
Transform Methods & Signal Processing

lecture 05

Matthew Roughan

<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

July 27, 2009

Filters

Filtering is a basic signal processing operation. We want to “filter out” some part of the signal so that we can see other more clearly. For instance, we want to filter out the “noise”. Common techniques for filtering either use transforms directly, or in their analysis and design and this is one of the most important applications of transform methods, but also, we will later see how we can implement some transforms using filter.

Filters

A filter takes some input $x(n)$, and produces an output $y(n)$, which has been filtered to extract certain features (e.g. trend, seasonality, ...)



References:

- Brockwell and Davis, 1996
- Box and Jenkins, 1976
- Anderson and Moore, 1979

Possible filter properties

- **invertibility:** The mapping $x(t) \rightarrow y(t)$ must be 1:1, so that each input signal has a unique output signal (don't need to invert all possible outputs).
- **memory:** $y(t_0)$ depends on $x(t)$ for $t \neq t_0$.
- **causality:** $y(t_0)$ only depends on $x(t)$ for $t \leq t_0$.
- **stability:** Bounded Input Bounded Output (BIBO).
If $|x(t)| \leq M$ for all t and some M , then $|y(t)| \leq R$ for all t and some R .
- **time invariance:** time shift doesn't matter, i.e.
 $x(t) \rightarrow y(t)$ implies $x(t - t_0) \rightarrow y(t - t_0)$.
- **linearity:** principle of superposition: $x_i \rightarrow y_i, i = 1, 2$
implies that for all $a_1, a_2 \in \mathbb{R}$, $a_1x_1 + a_2x_2 \rightarrow a_1y_1 + a_2y_2$.

Linear Filters

Response is linear in the input, e.g. given the filter,

$$\mathcal{L}\{x_1\} \rightarrow y_1$$

$$\mathcal{L}\{x_2\} \rightarrow y_2$$

Then

$$\mathcal{L}\{ax_1 + bx_2\} \rightarrow ay_1 + by_2$$

The output of linear filters can be written as a linear combination of the inputs.

$$y(m) = \sum_{i=-\infty}^{\infty} w(m, i)x(m - i)$$

Linear Time Invariant Filters

- time invariant filters don't change over time, so
 $w(m, i) = w(i)$

The output of linear filters can be written as a linear combination of the inputs.

$$y(m) = \sum_{i=-\infty}^{\infty} w(i)x(m-i)$$

Note that this is a discrete convolution!

Convolution

Definition: Discrete convolution

$$[x_1 * x_2](n) = \sum_{i=-\infty}^{\infty} x_1(i)x_2(n-i) = \sum_{i=-\infty}^{\infty} x_1(n-i)x_2(i)$$

Now remember the impact of convolutions in DFTs, e.g.

$$\mathcal{F}\{x_1 * x_2\} = X_1(k)X_2(k)$$

where $\mathcal{F}\{x_1(n)\} = X_1(k)$ and $\mathcal{F}\{x_2(n)\} = X_2(k)$.

Circular convolution

Convolution of finite, discrete-time sequences

- standard convolution assumes infinite series of data

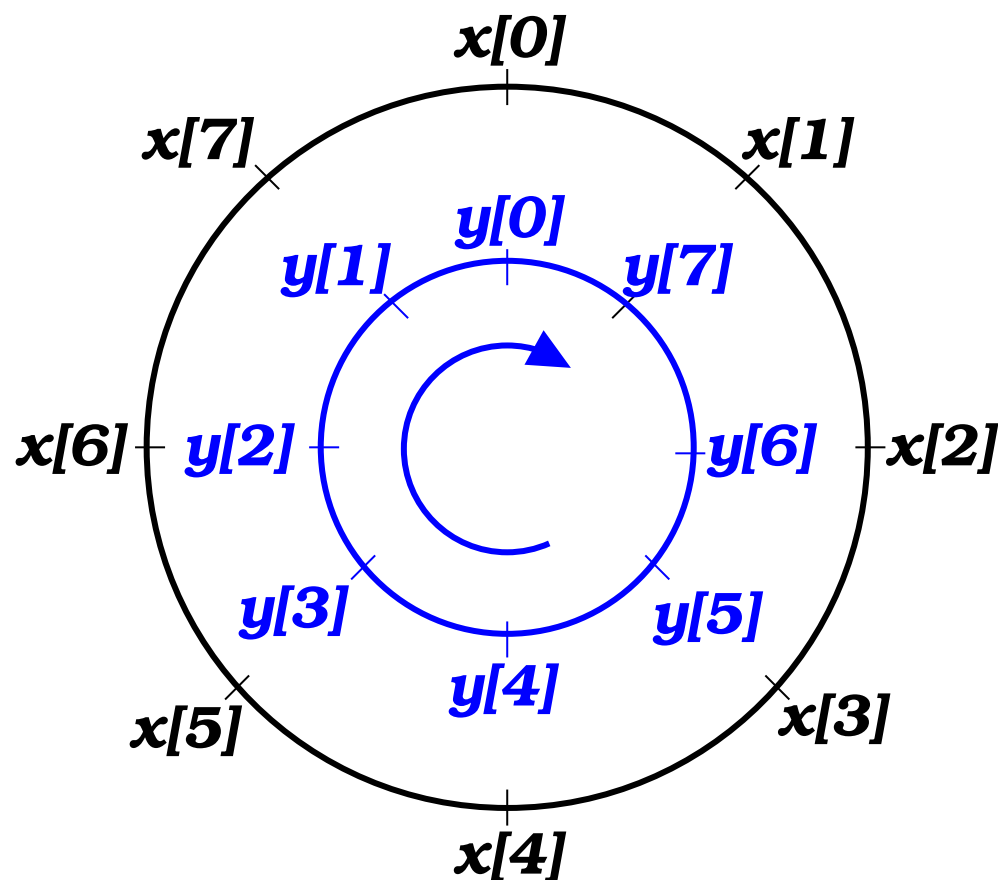
$$(x * y)[n] = \sum_{i=-\infty}^{\infty} x(i)y(n-i)$$

- note what happens at the edges of a standard convolution, when the series are finite
 - either zero pad (pretend series are infinite, but values are zero)
 - truncate convolution (only compute where edge effects are nil)
 - take **circular convolution**

Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

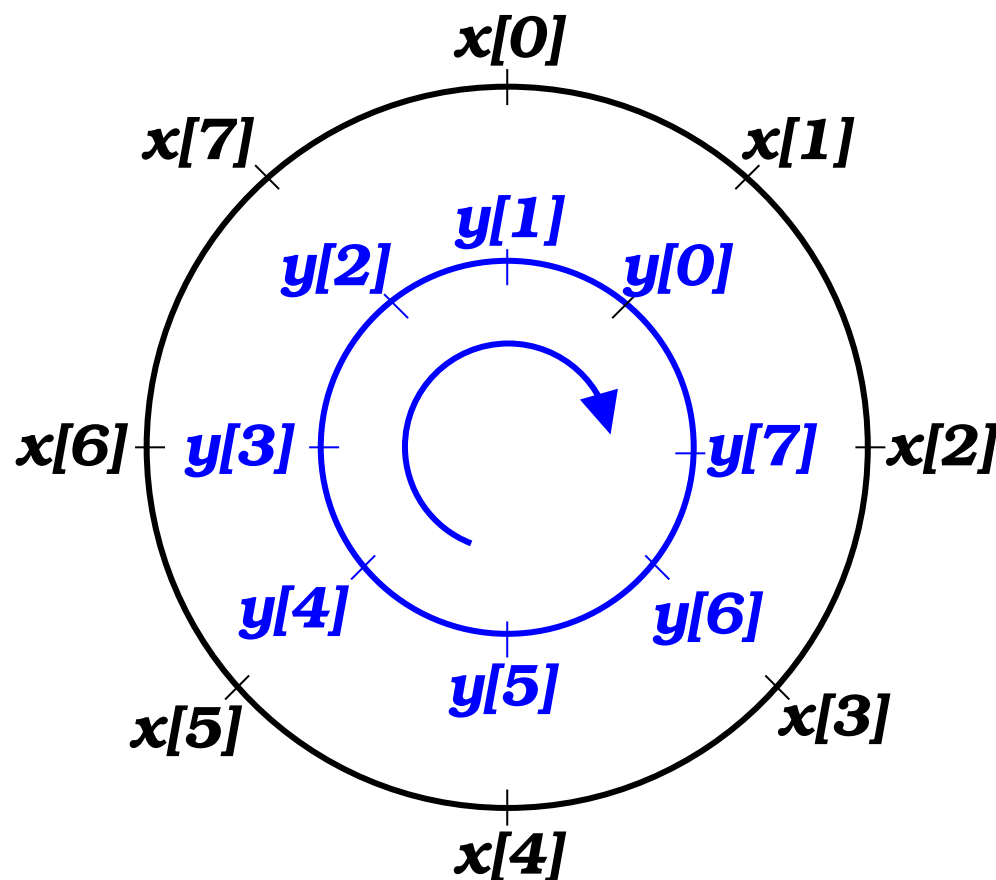
$(x * y)[0]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

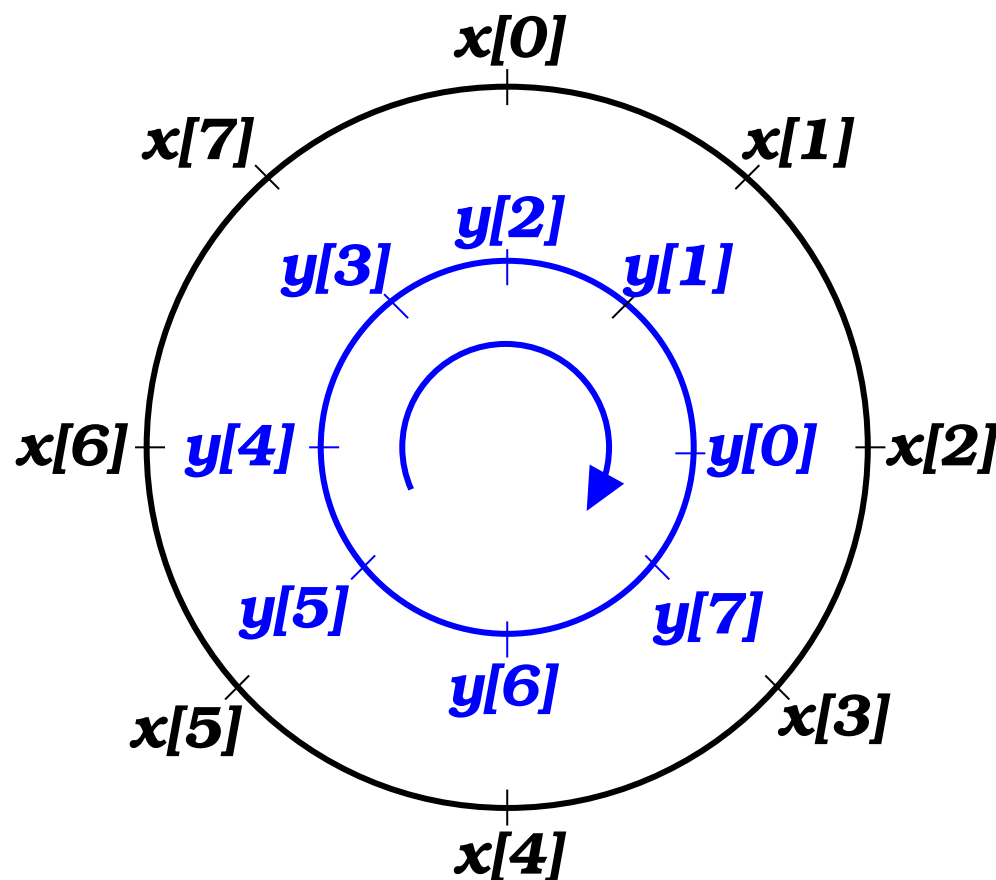
$(x * y)[1]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

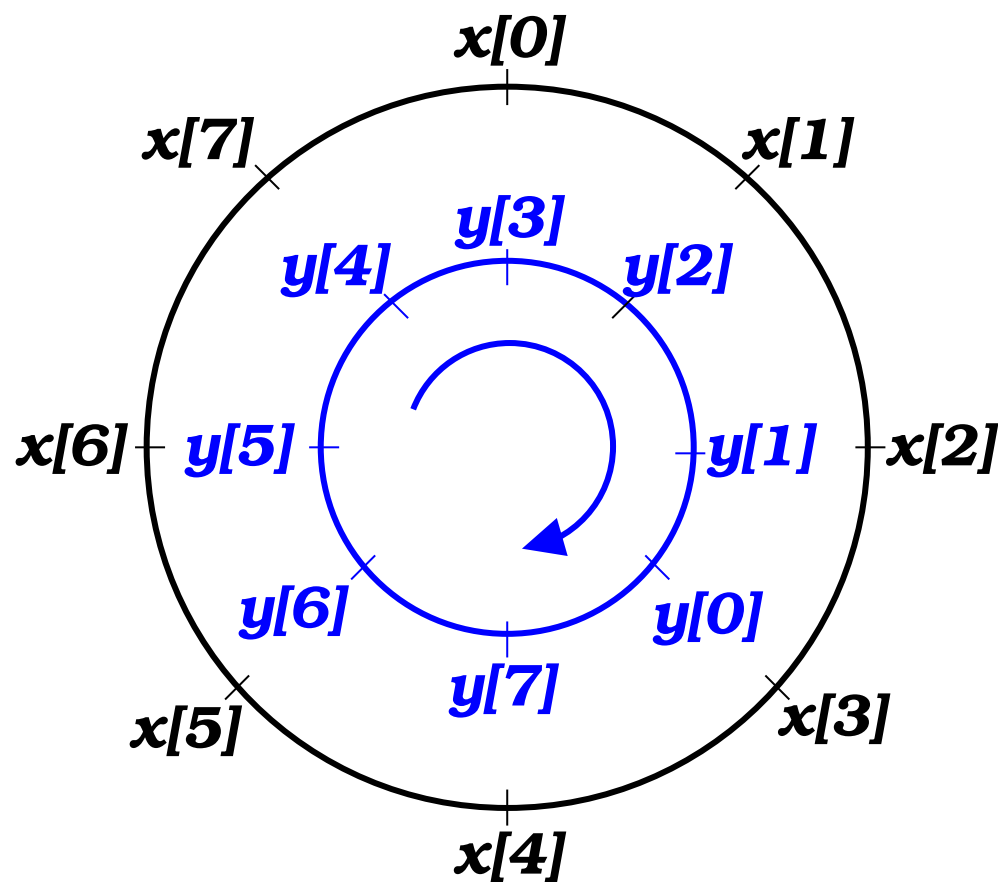
$(x * y)[2]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

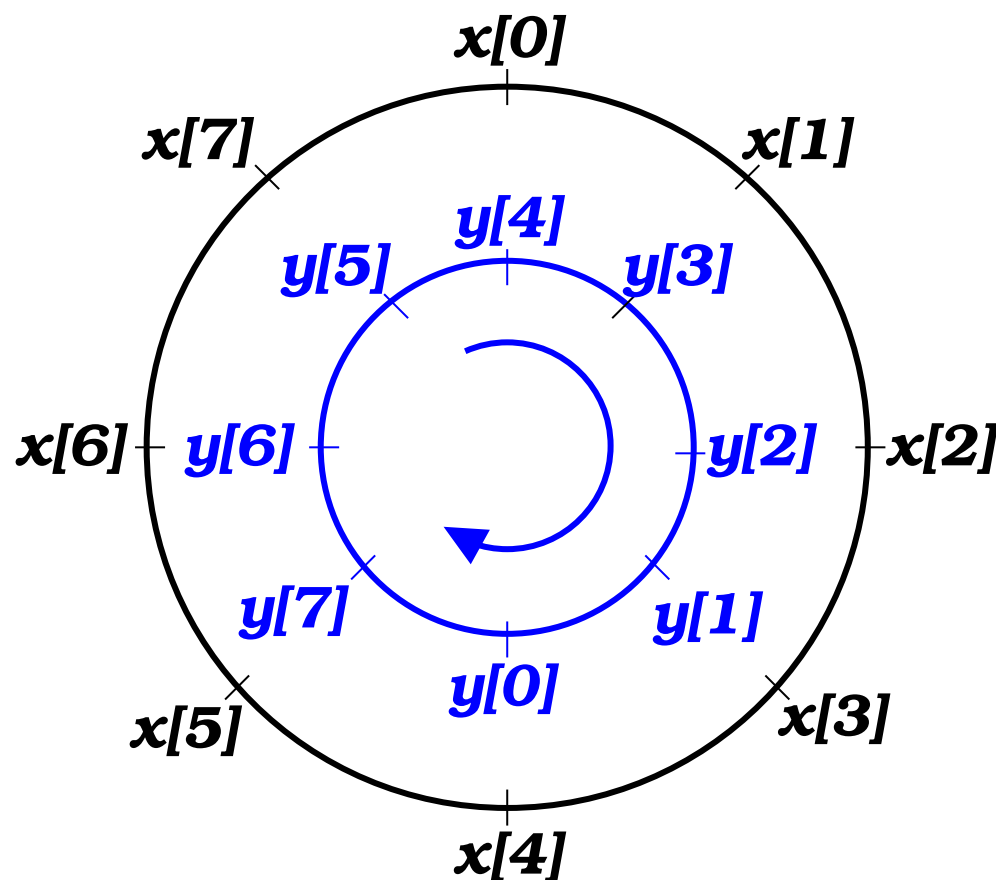
$(x * y)[3]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

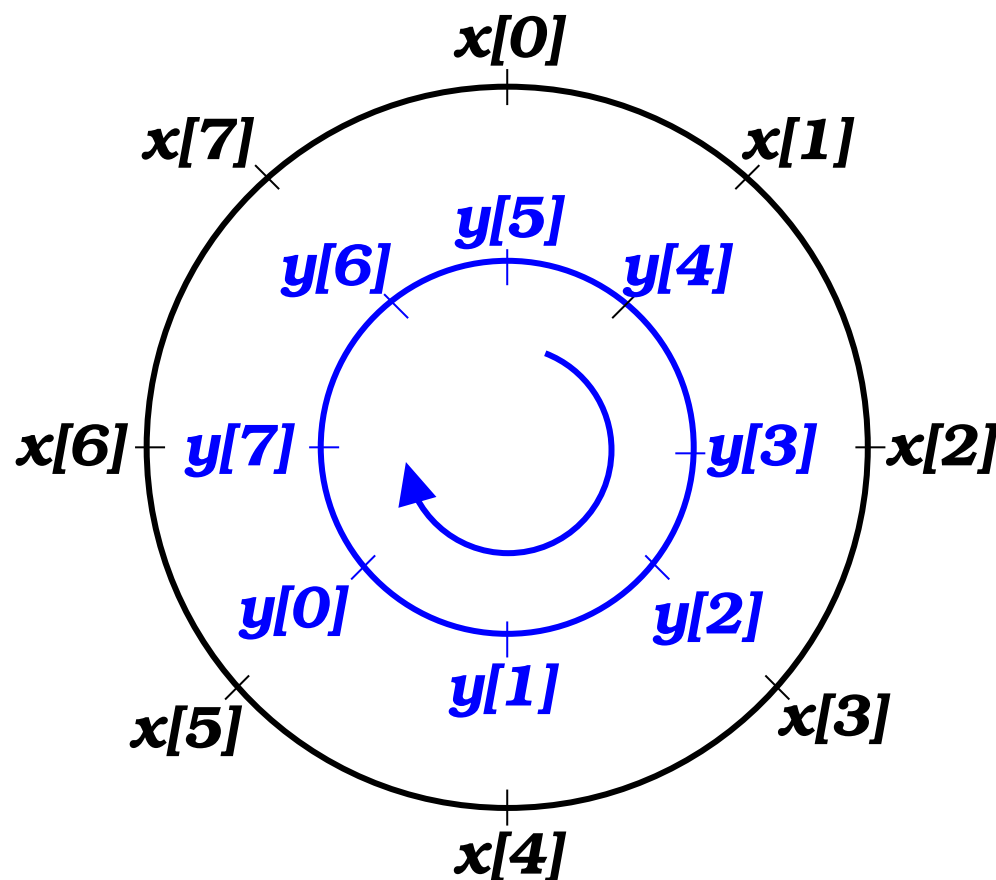
$(x * y)[4]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

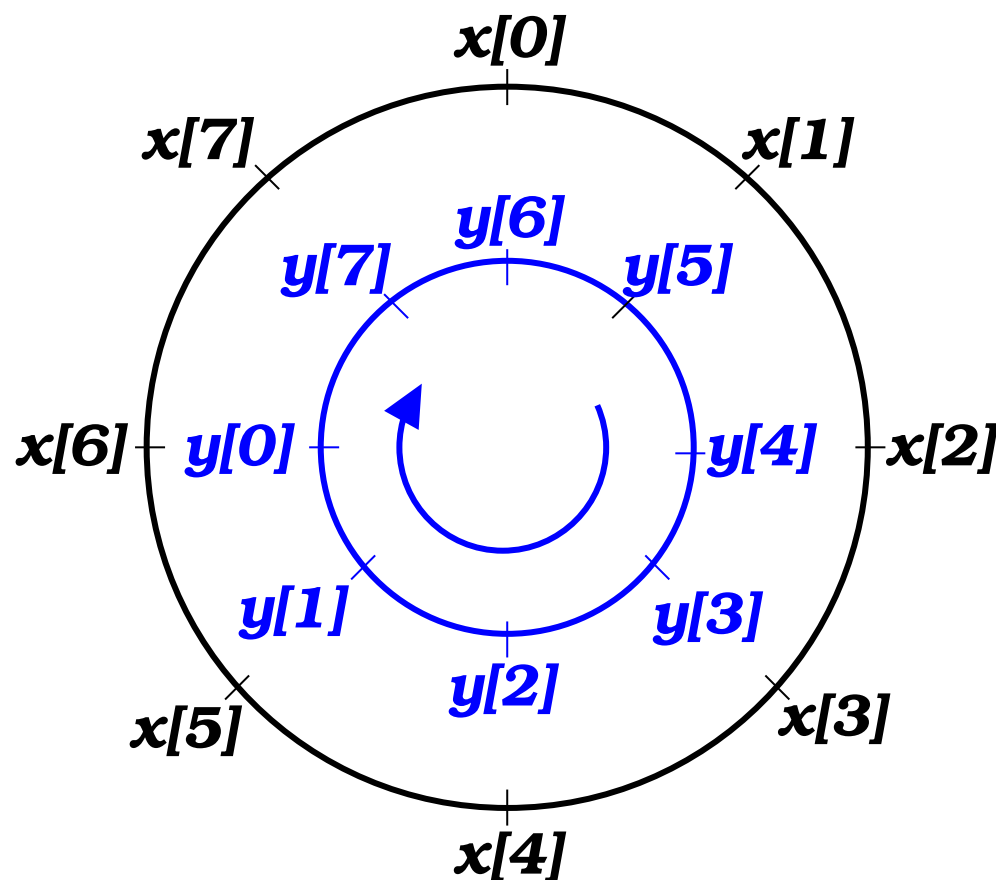
$(x * y)[5]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

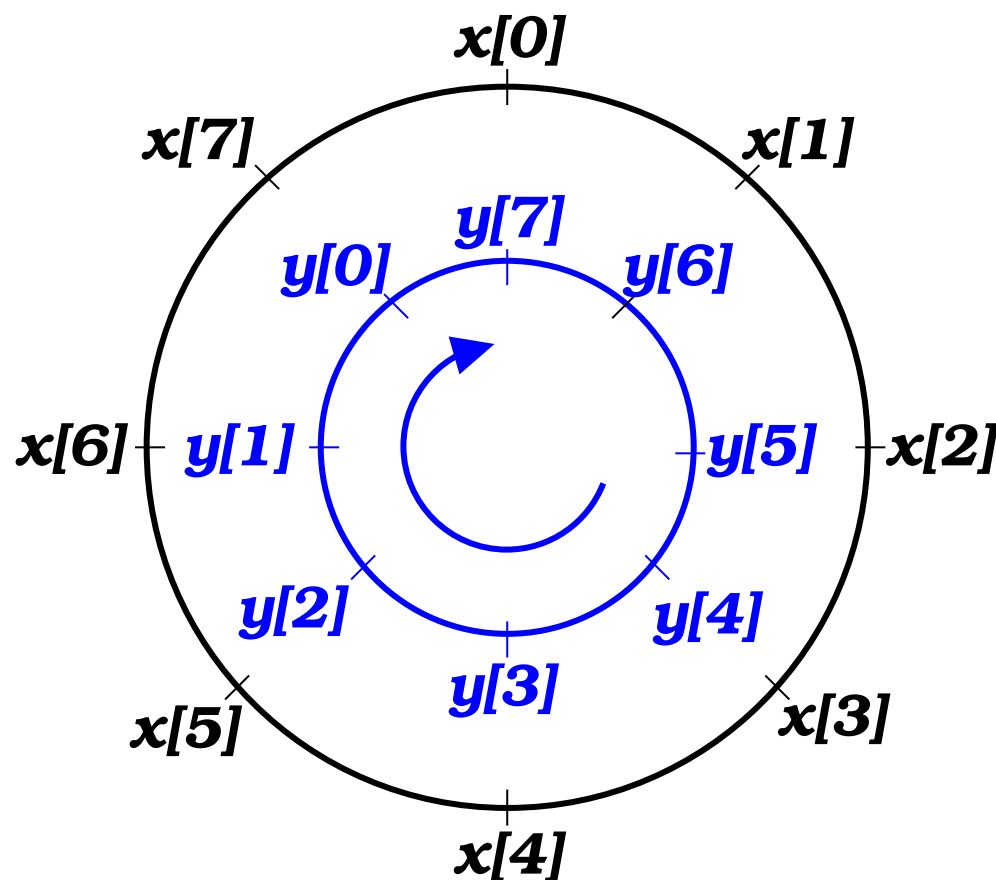
$(x * y)[6]$



Circular convolution

$$\text{Circular convolution } (x * y)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

$(x * y)[7]$



Example

Take $y(n) = [x * x](n) = (1, 1, 0, 0) * (1, 1, 0, 0) = (1, 2, 1, 0)$

$$(x * x)[n] = \sum_{i=0}^{N-1} x(i)y(n - i \bmod N)$$

$$\begin{aligned} y(0) &= x(0)x(0) + x(1)x(3) + x(2)x(2) + x(3)x(1) \\ &= 1 + 0 + 0 + 0 &= 1 \end{aligned}$$

$$\begin{aligned} y(1) &= x(0)x(1) + x(1)x(0) + x(2)x(3) + x(3)x(2) \\ &= 1 + 1 + 0 + 0 &= 2 \end{aligned}$$

$$\begin{aligned} y(2) &= x(0)x(2) + x(1)x(1) + x(2)x(0) + x(3)x(3) \\ &= 0 + 1 + 0 + 0 &= 1 \end{aligned}$$

$$\begin{aligned} y(3) &= x(0)x(3) + x(1)x(2) + x(2)x(1) + x(3)x(0) \\ &= 0 + 0 + 0 + 0 &= 0 \end{aligned}$$

Example DFT of circular convolution

Convolution theorem still holds

$$\begin{aligned}x(n) &= (1, 1, 0, 0) \\X(k) &= (2, 1 - i, 0, 1 + i)\end{aligned}$$

$$\begin{aligned}y(n) &= [x * x](n) &= (1, 2, 1, 0) \\Y(k) &= X(k)X(k) &= (4, -2i, 0, 2i)\end{aligned}$$

Direct calculation of $Y(k) = \sum_{n=0}^{N-1} y(n)e^{-i2\pi kn/N}$

$$\begin{aligned}Y(0) &= e^{-i2\pi 0/4} + 2e^{-i2\pi 0/4} + e^{-i2\pi 0/4} = 1 + 2 + 1 = 4 \\Y(1) &= e^{-i2\pi 0/4} + 2e^{-i2\pi 1/4} + e^{-i2\pi 2/4} = 1 - 2i - 1 = -2i \\Y(2) &= e^{-i2\pi 0/4} + 2e^{-i2\pi 2/4} + e^{-i2\pi 4/4} = 1 - 2 + 1 = 0 \\Y(3) &= e^{-i2\pi 0/4} + 2e^{-i2\pi 3/4} + e^{-i2\pi 6/4} = 1 + 2i - 1 = 2i\end{aligned}$$

Linear Time Invariant Causal Filters

- time invariant filters don't change over time, so $w(m, i) = w(i)$
- causal filters only depend on the past, so $w(-i) = 0$, for $i > 0$.

The output of linear filters can be written as a linear combination of the inputs.

$$y(m) = \sum_{i=0}^{\infty} w(i)x(m-i)$$

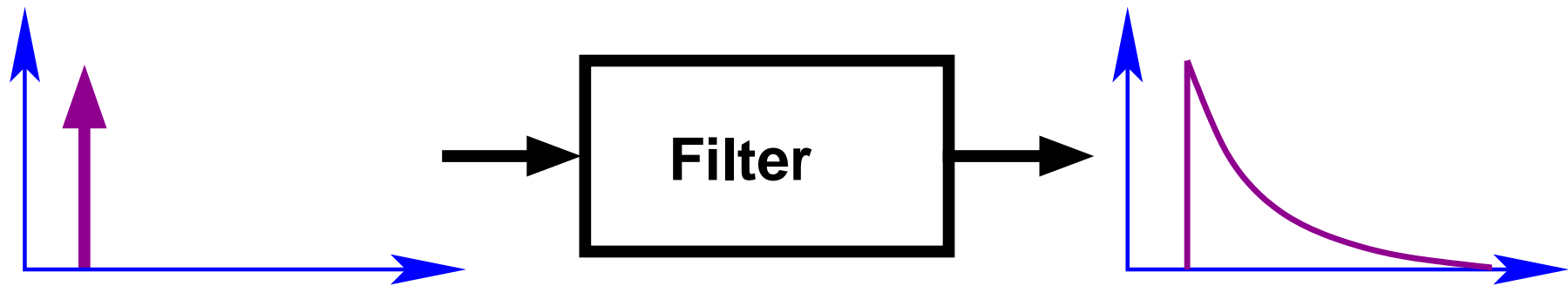
Note that this is also a discrete convolution!

Impulse response

Given a filter:



The impulse response is the output of the filter given an impulse as the input.



Impulse response

For a linear, time-invariant filter F , the impulse response is

$$I_F(m) = \sum_{i=-\infty}^{\infty} w(i)\delta_{mi} = w(m)$$

where δ_{nk} is the Kronecker delta, defined by

$$\delta_{nk} = \begin{cases} 1 & \text{if } n = k \\ 0 & \text{otherwise} \end{cases}$$

So a **linear time-invariant filter** can be completely characterized by its **impulse response**.

Impulse response

Note that any signal $x(n)$ can be written as a linear combination of impulses, e.g.

$$x(n) = \sum_{k=-\infty}^{\infty} \delta_{nk}x(k)$$

Given linearity of the filter, the output can be written as the same linear combination of the impulse responses, e.g.

$$\begin{aligned} y(m) &= \sum_{i=-\infty}^{\infty} w(i) \left[\sum_{k=-\infty}^{\infty} \delta_{m-i,k}x(k) \right] \\ &= \sum_{i=-\infty}^{\infty} w(i)x(m-i) \end{aligned}$$

Memory

Filters can have finite, or infinite memory

- **FIR:** Finite Impulse Response filters have an impulse response which have a finite number of terms, i.e. $\exists N$ such that

$$w(n) = 0, \quad \forall |n| > N$$

- **IIR:** Infinite Impulse Response filters have an impulse response with an infinite number of terms.

though for BIBO we require a finite sum, e.g.

$$\sum_{i=-\infty}^{\infty} |w(i)| < \infty$$

FIR example: Moving Average

(finite) **Moving Average (MA)**

$$y(n) = \sum_{i=-N}^N b(i)x(n-i)$$

typical example, symmetric rectangular windowed MA

$$y(n) = \frac{1}{2N+1} \sum_{i=-N}^N x(n-i)$$

NB: this is a non-causal filter

FIR example: difference

A **difference operator** (or filter) looks like

$$y(n) = x(n) - x(n - 1)$$

Note this is a special case of the MA above

$$b(0) = 1, b(1) = -1$$

but this terminology is used differently in different fields

- signal processing and stats: MA as defined above
- financial time series: MA \Rightarrow low pass

NB: this is a causal filter

Example of IIR filter: EWMA

Exponentially Weighted Moving Average (EWMA)

$$y(n) = ay(n-1) + (1-a)x(n)$$

alternative IIR representation

$$y(n) = (1-a) \sum_{i=0}^{\infty} a^i x(n-i)$$

gives **exponentially** decreasing weight to historical data

More general case **Autoregressive (AR)** filters

$$y(n) = \sum_{i=1}^p a(i)y(n-i) + b(0)x(n)$$

Transfer function

- we can represent LTI filter as convolution
- in Fourier domain, convolution becomes a simple product
- LTI filter is completely characterized by FT of its impulse response
- we call the FT of the impulse response the **Transfer function**, e.g.

$$W(k) = DFT(w(n))$$

- The transfer function tells us the impact of the filter on different components of the spectrum of a signal

Types of filters

- **low pass:** pass low frequencies, stop high frequencies.

these filters act as smoothers of the data.
e.g. EWMA, MA

- **high pass:** pass high frequencies, stop low frequencies.

e.g. differencer - highlights edges

- **band pass:** pass a band of frequencies

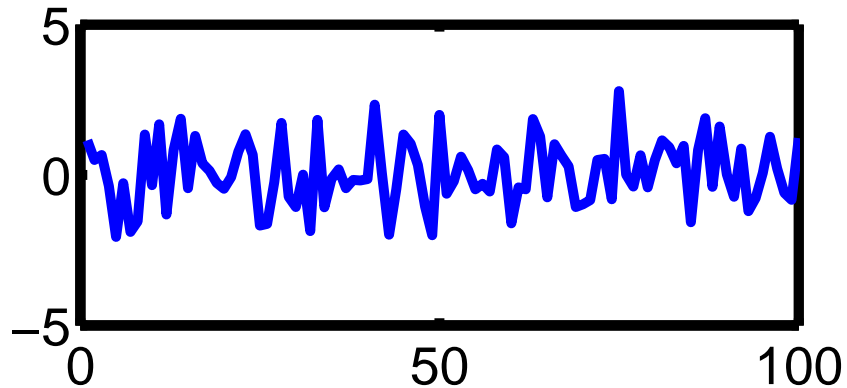
- **notch:** exclude a band (sometimes called bandstop)

e.g. remove signal at a particular frequency to prevent feedback ("ringing out")

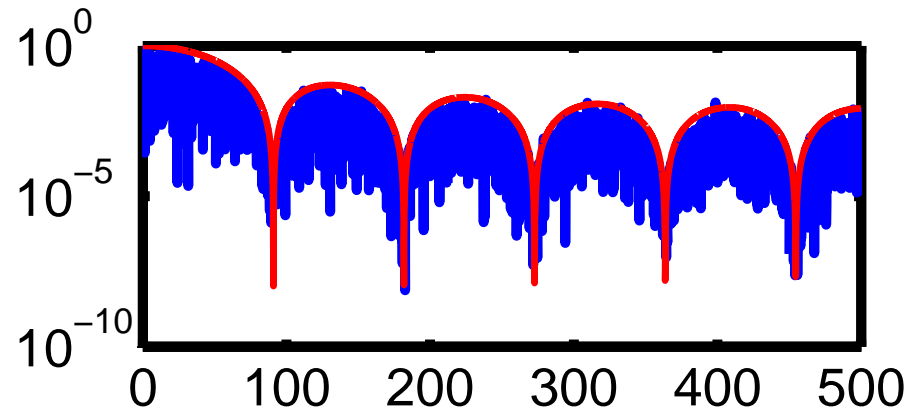
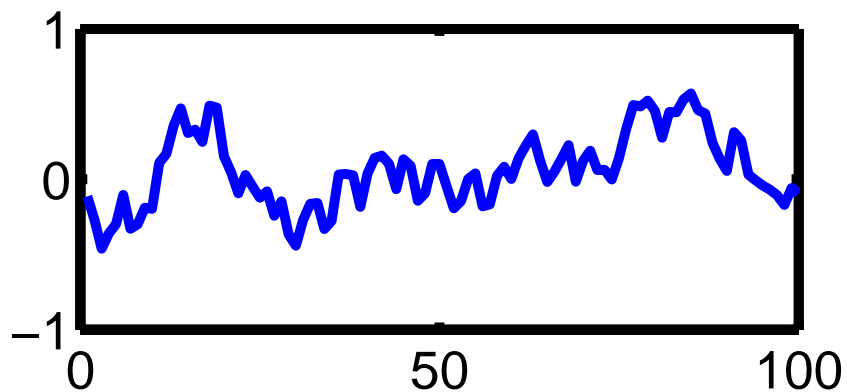
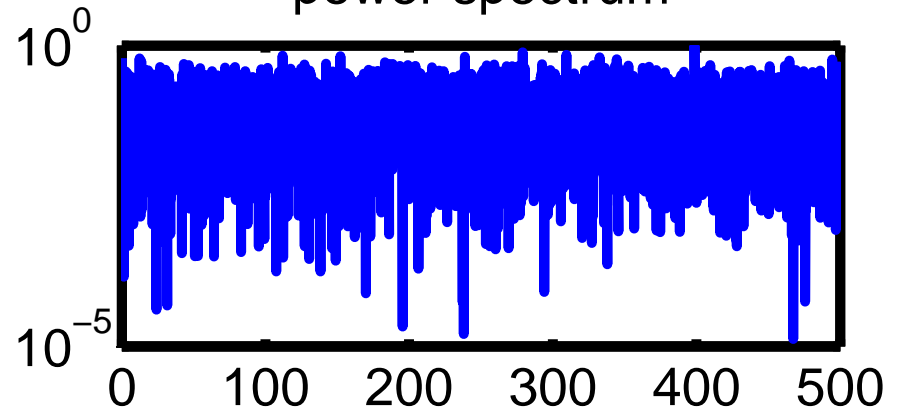
Example: MA $y(n) = \frac{1}{2N+1} \sum_{i=-N}^N x(n-i)$

$f_s = 1000$, $N = 10,000$, input white noise, $N = 5$

signal segment



power spectrum

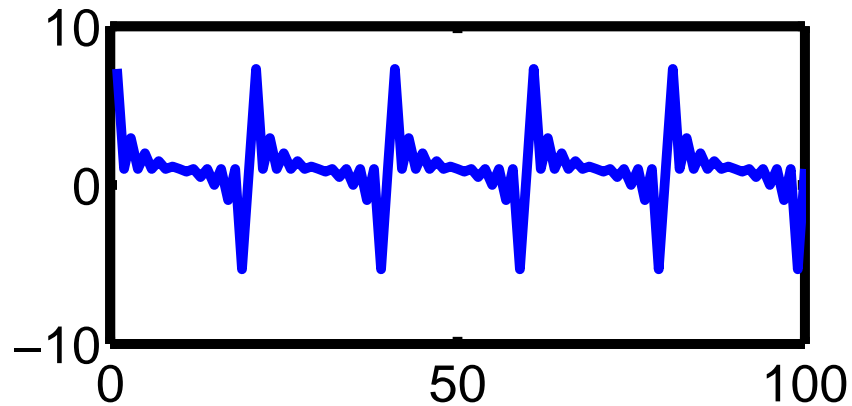


low pass

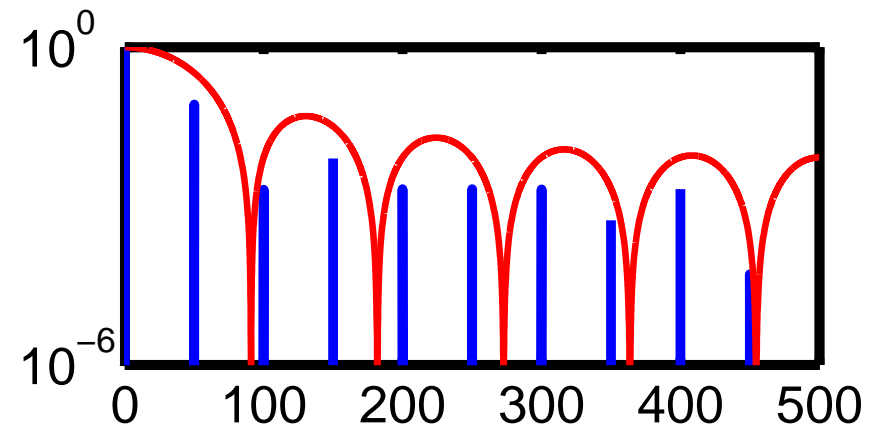
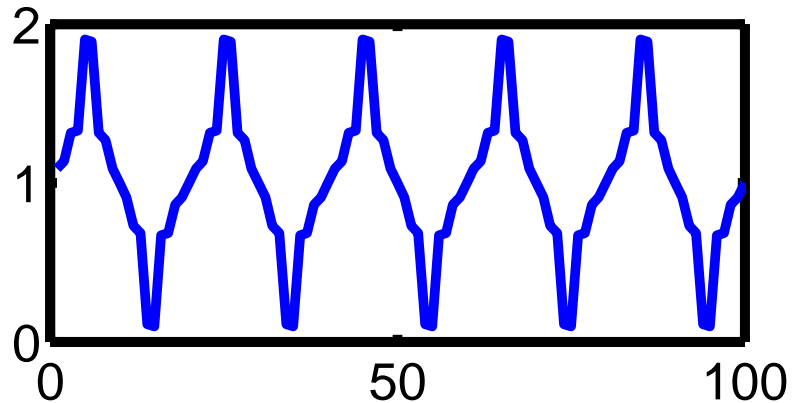
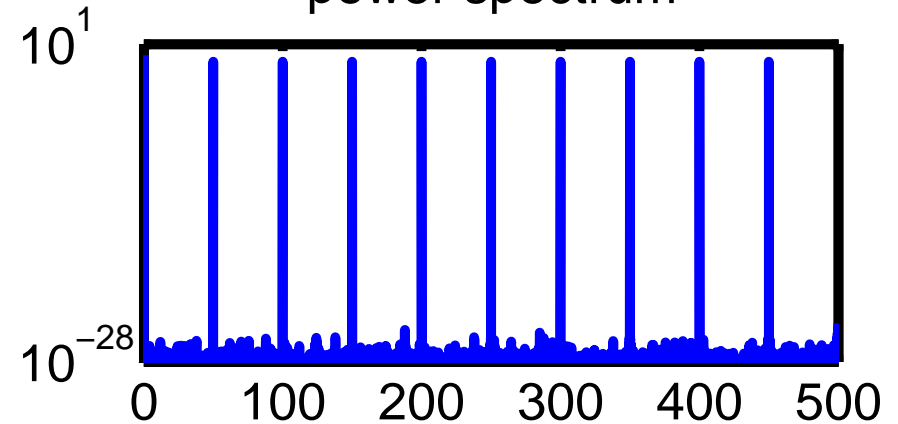
Example: MA $y(n) = \frac{1}{2N+1} \sum_{i=-N}^N x(n-i)$

$f_s = 1000$, $N = 10,000$, input 10 sines evenly spaced freq.

signal segment



power spectrum

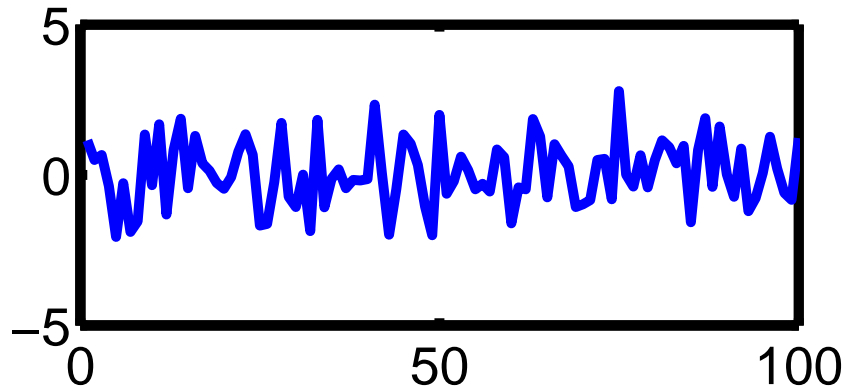


low pass

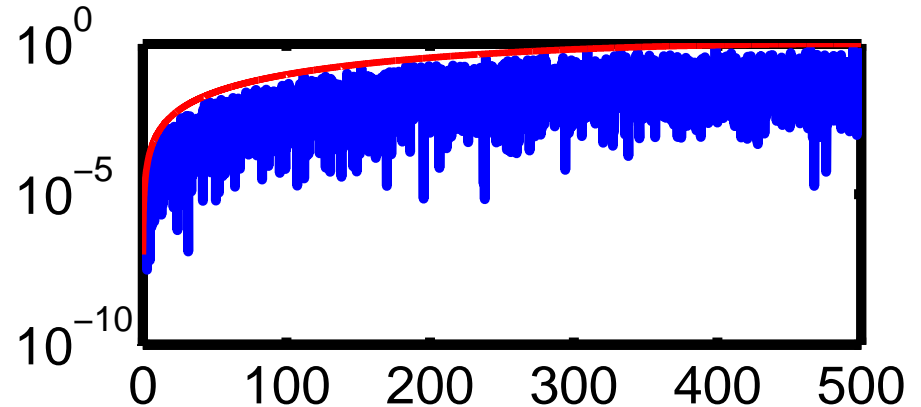
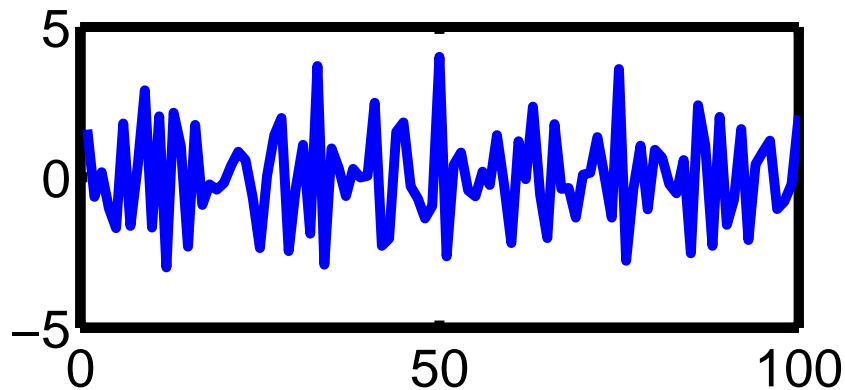
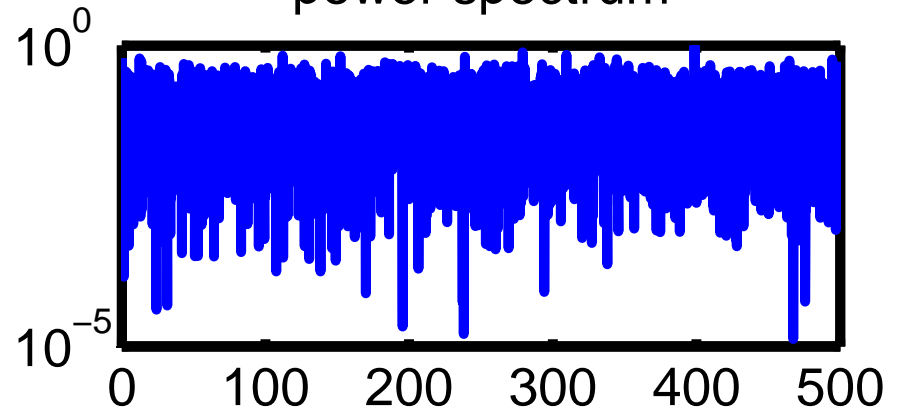
Example: difference $y(n) = x(n) - x(n-1]$

$f_s = 1000$, $N = 10,000$, input white noise

signal segment



power spectrum

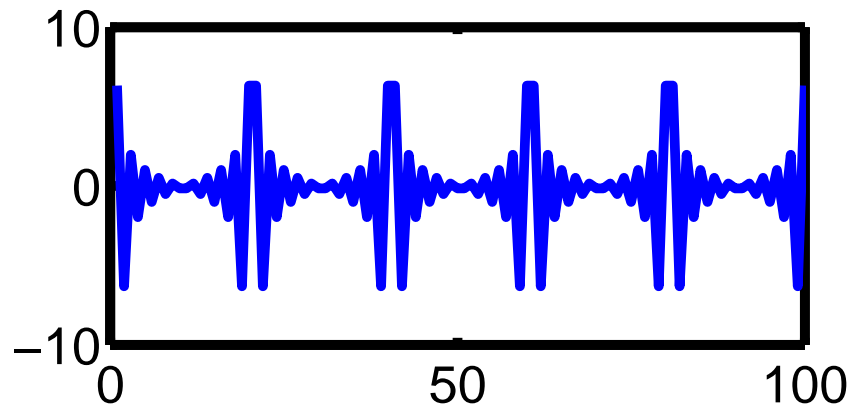
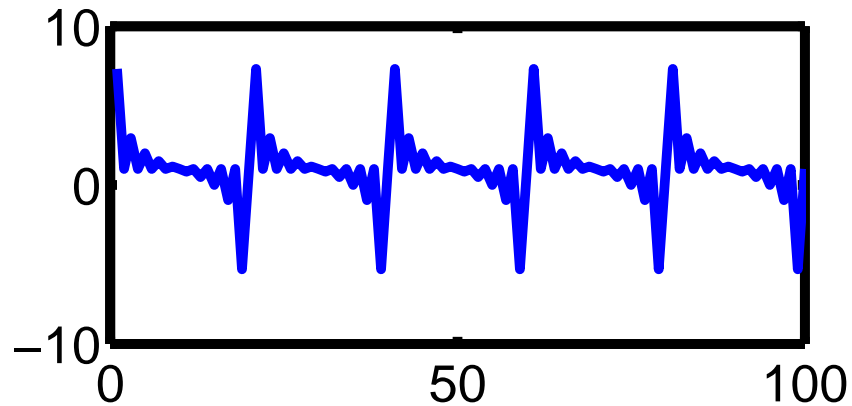


high pass

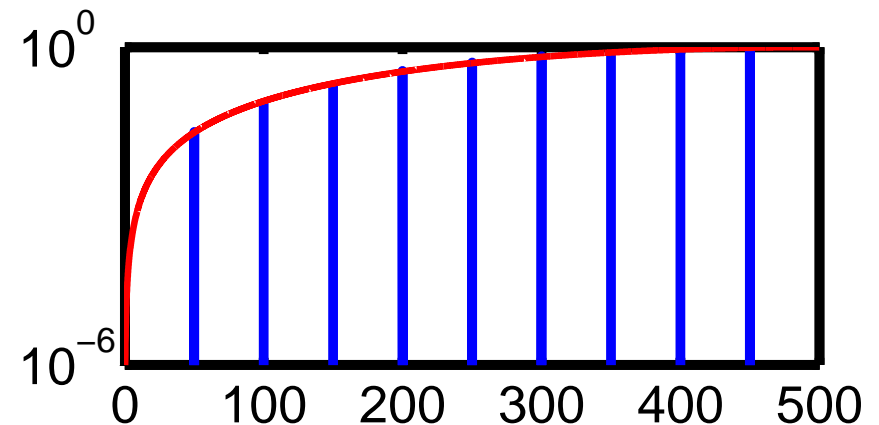
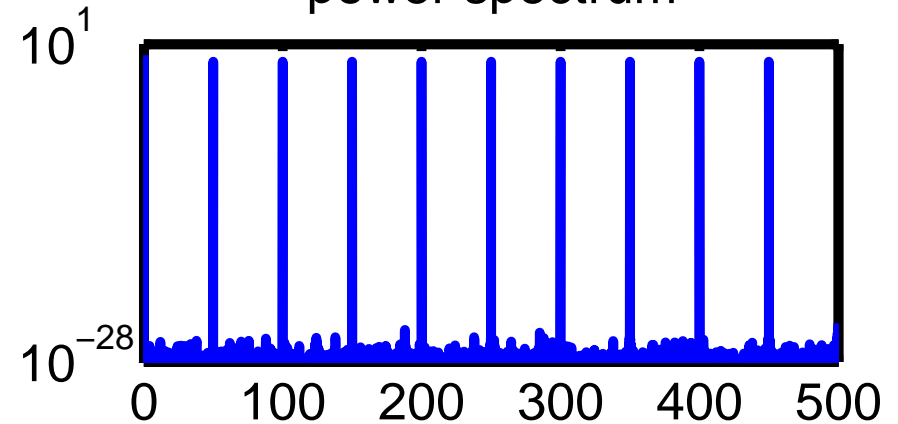
Example: difference $y(n) = x(n) - x(n-1]$

$f_s = 1000$, $N = 10,000$, input 10 sines evenly spaced freq.

signal segment



power spectrum

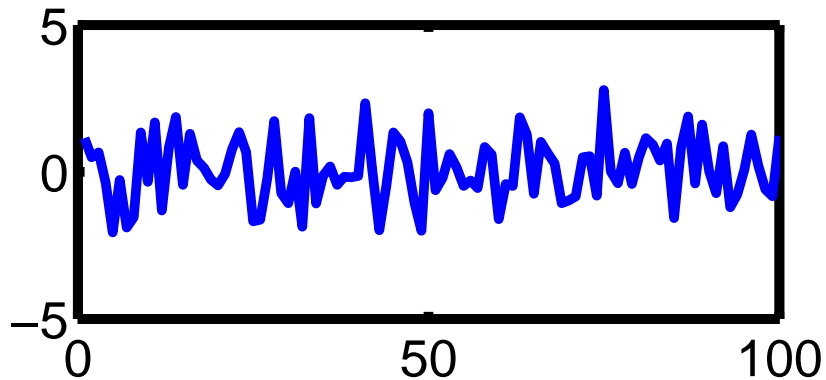


high pass

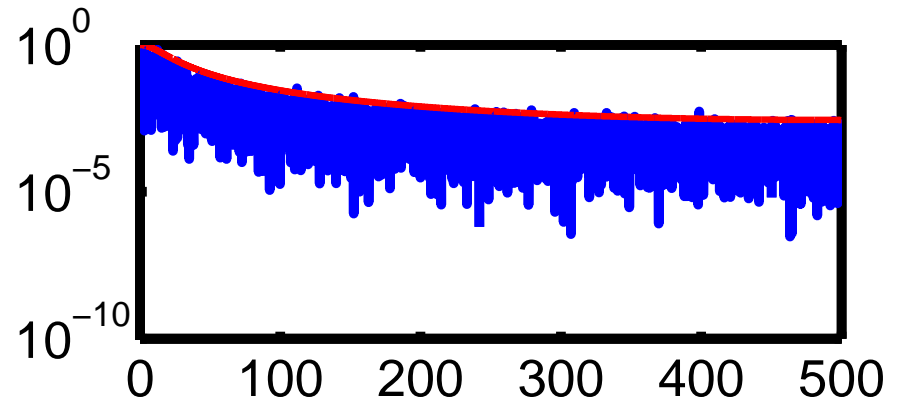
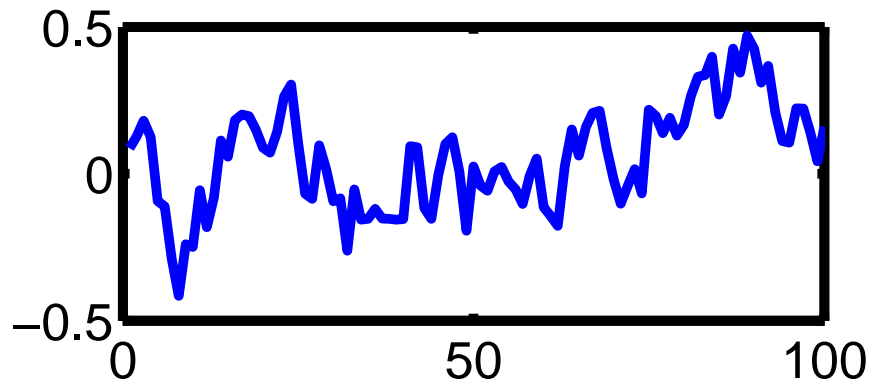
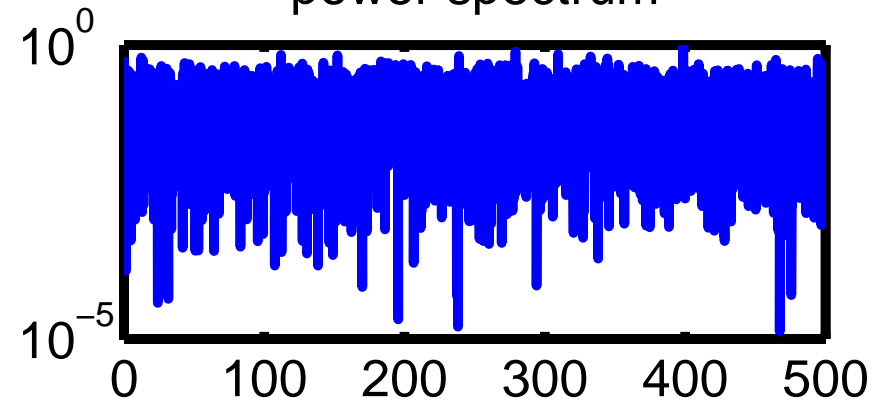
Example: EWMA $y(n) = ay(n-1) + (1-a)x(n)$

$f_s = 1000$, $N = 10,000$, input white noise, $a = 0.9$

signal segment



power spectrum

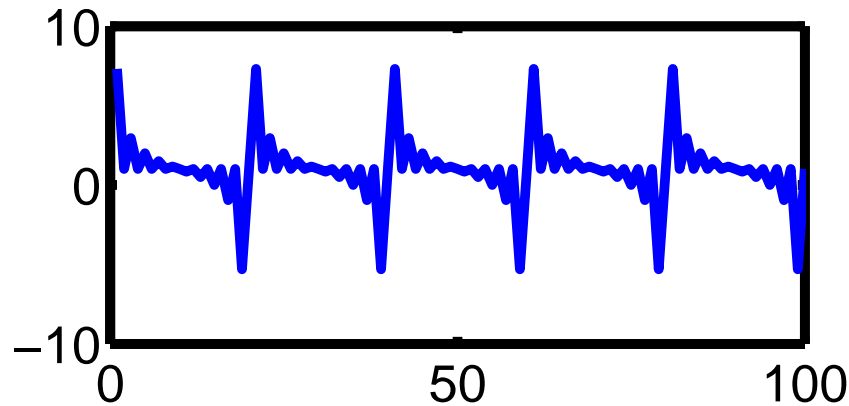


low pass

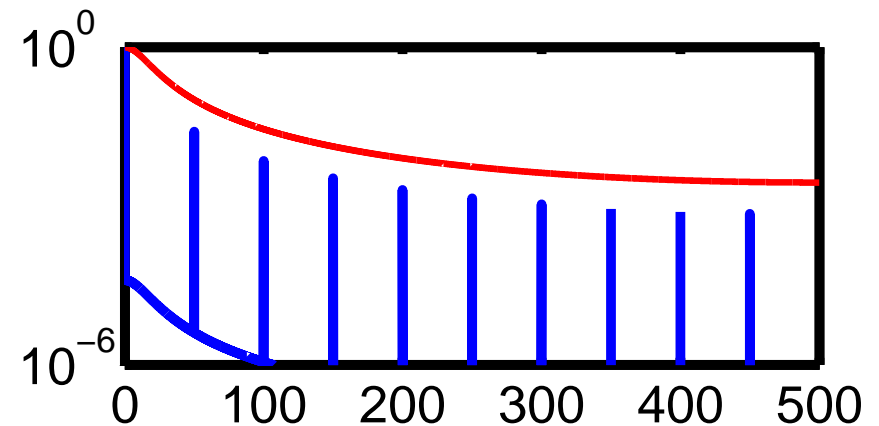
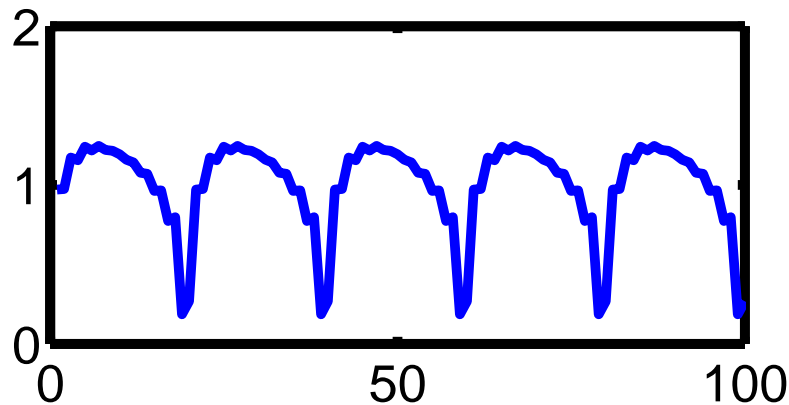
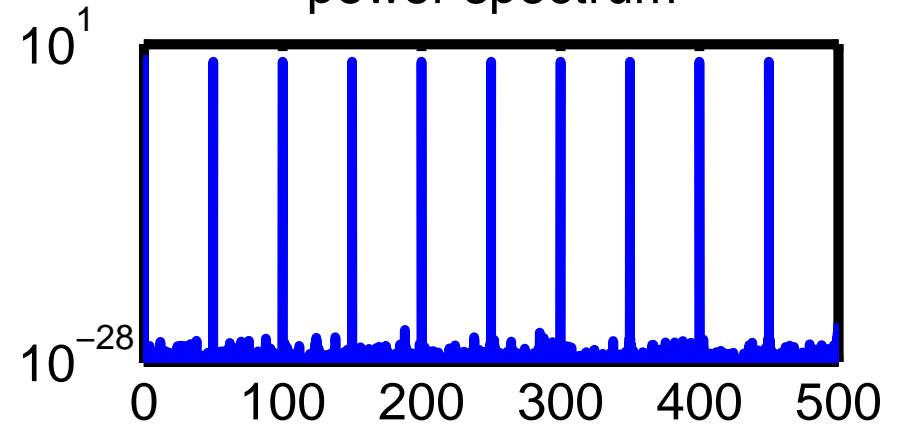
Example: EWMA $y(n) = ay(n-1) + (1-a)x(n)$

$f_s = 1000$, $N = 10,000$, input 10 sines evenly spaced freq.

signal segment







power spectrum





low pass



What do they sound like?

- white noise (static) 
- MA (low-pass, length 11) 
- MA (low-pass, length 11) 
- difference (high-pass) 

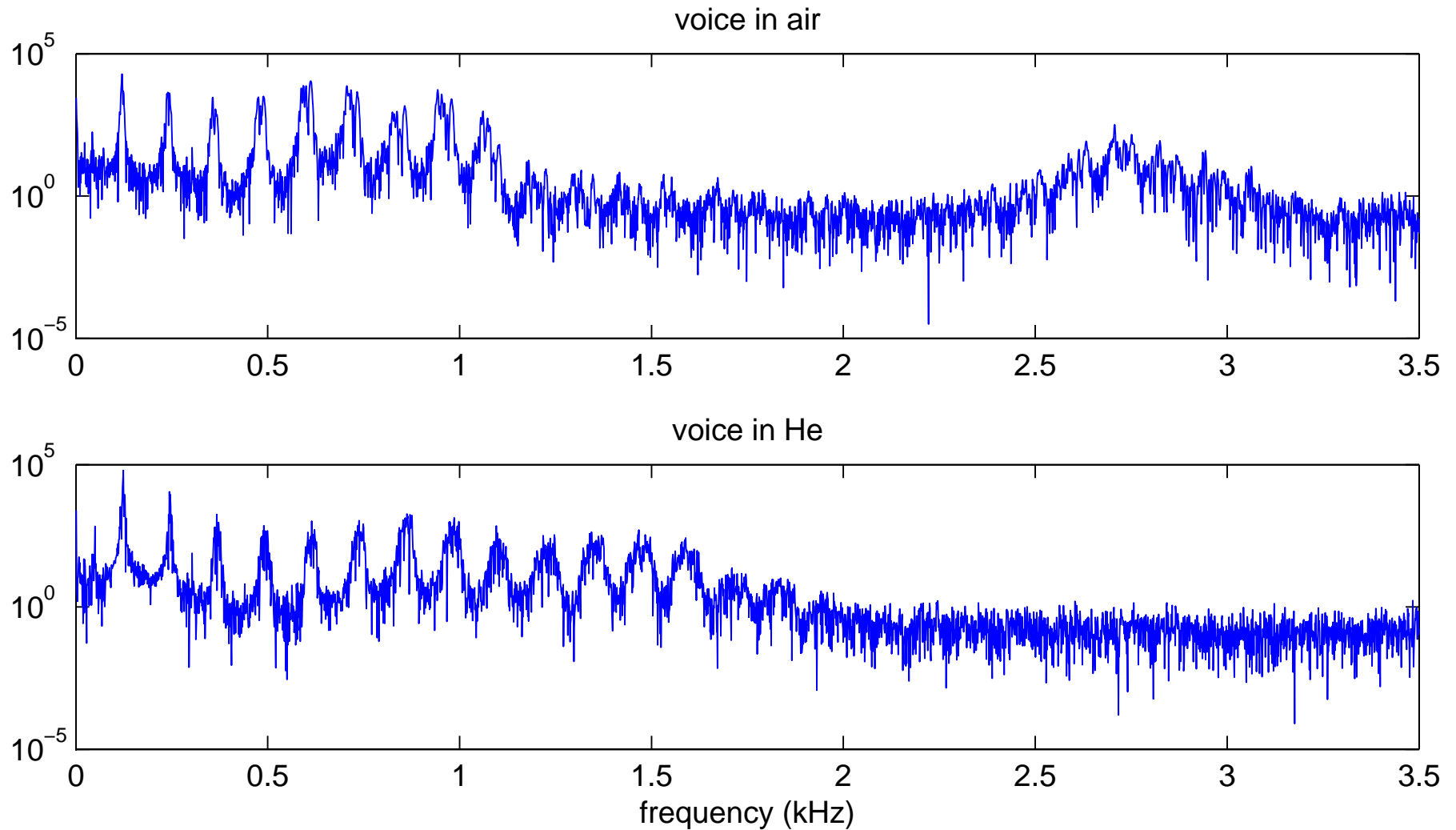
Why does He make your voice funny?

- inhaling Helium (He) makes your voice sound funny
 - Ordinary speech 
 - Helium speech 
- conventional explanation: He is much lighter than air, and the speed of sound is around 3 times as fast, hence vibrations are faster, and so the pitch of your voice is higher.
 - **But this is wrong!**
 - vibrations in our voice are generated by vocal cords, which whose vibrational frequency is independent of gas.

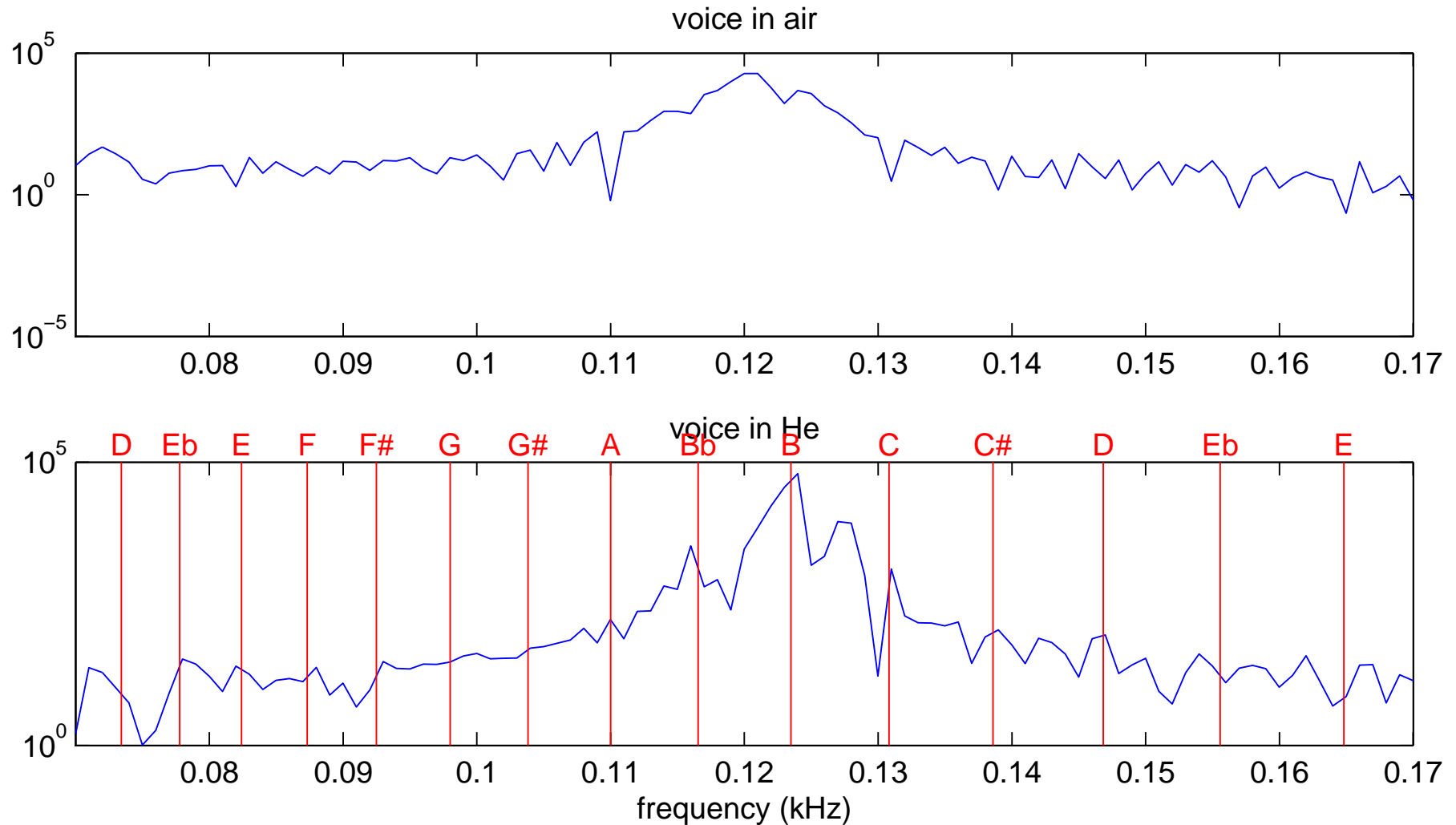
Why does He make your voice funny?

- actually voice is generated by two processes
 - vocal chords generate vibrations
 - vocal tract (mouth, tongue, etc.) filters the sounds
- He in vocal tract changes the transfer function so that the filter becomes "higher" pass than before.
 - pitch is not changed
 - only timbre (harmonics) are changed
- Example
 - Pitch in Air 
 - Pitch in He 

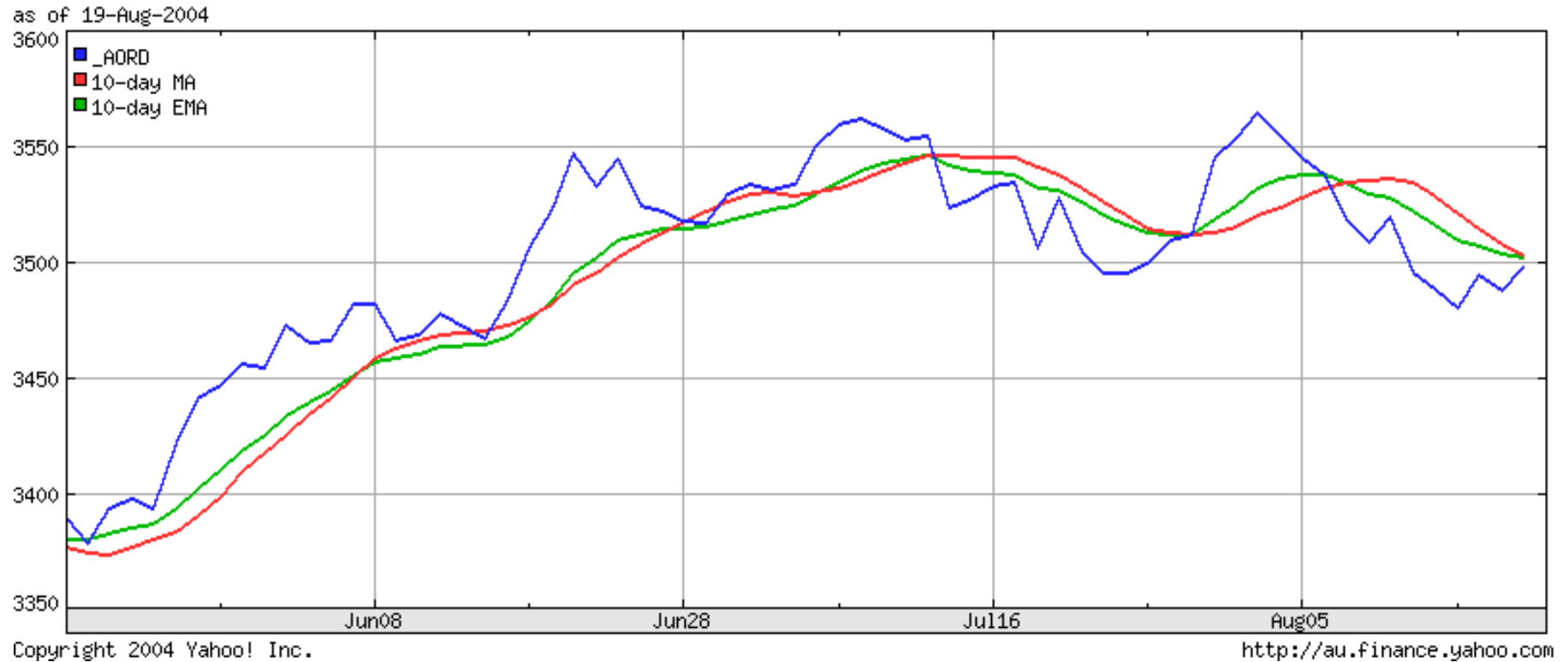
Why does He make your voice funny?



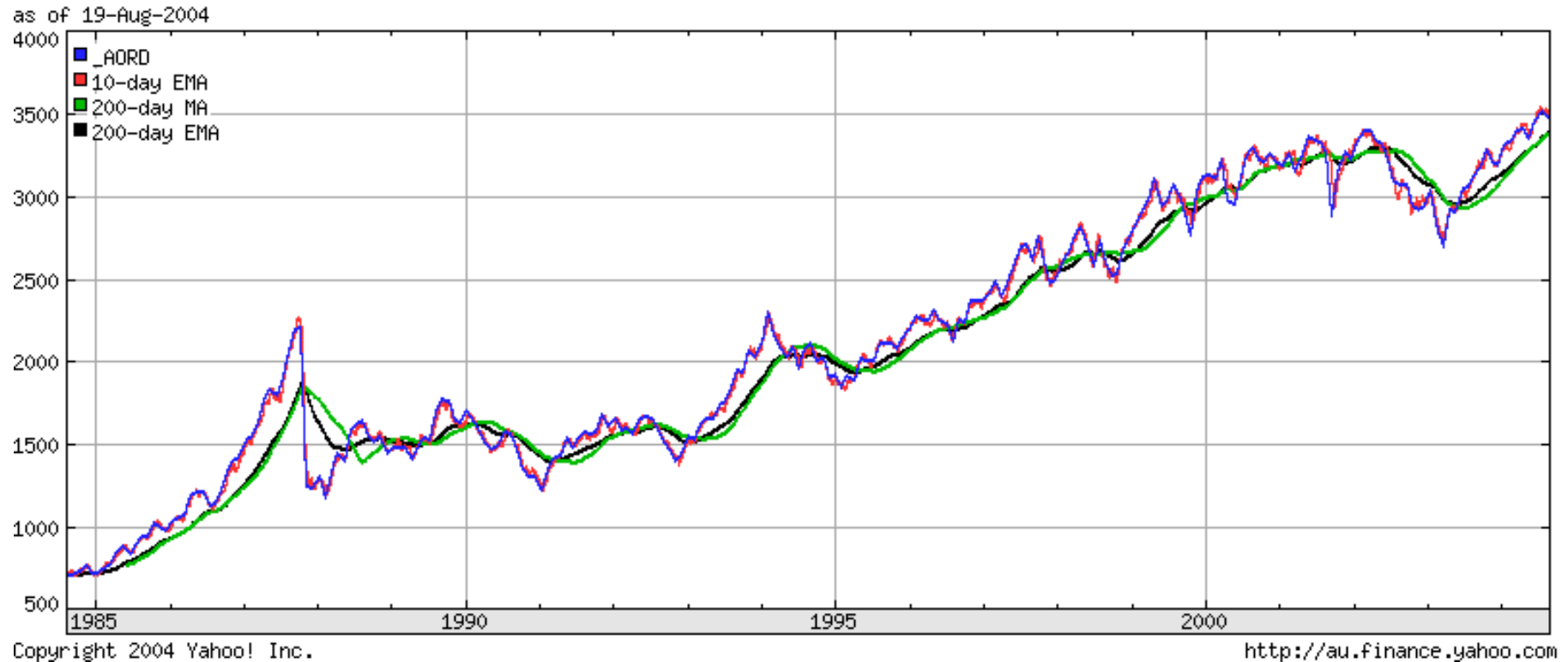
Why does He make your voice funny?



Financial data example



Financial data example



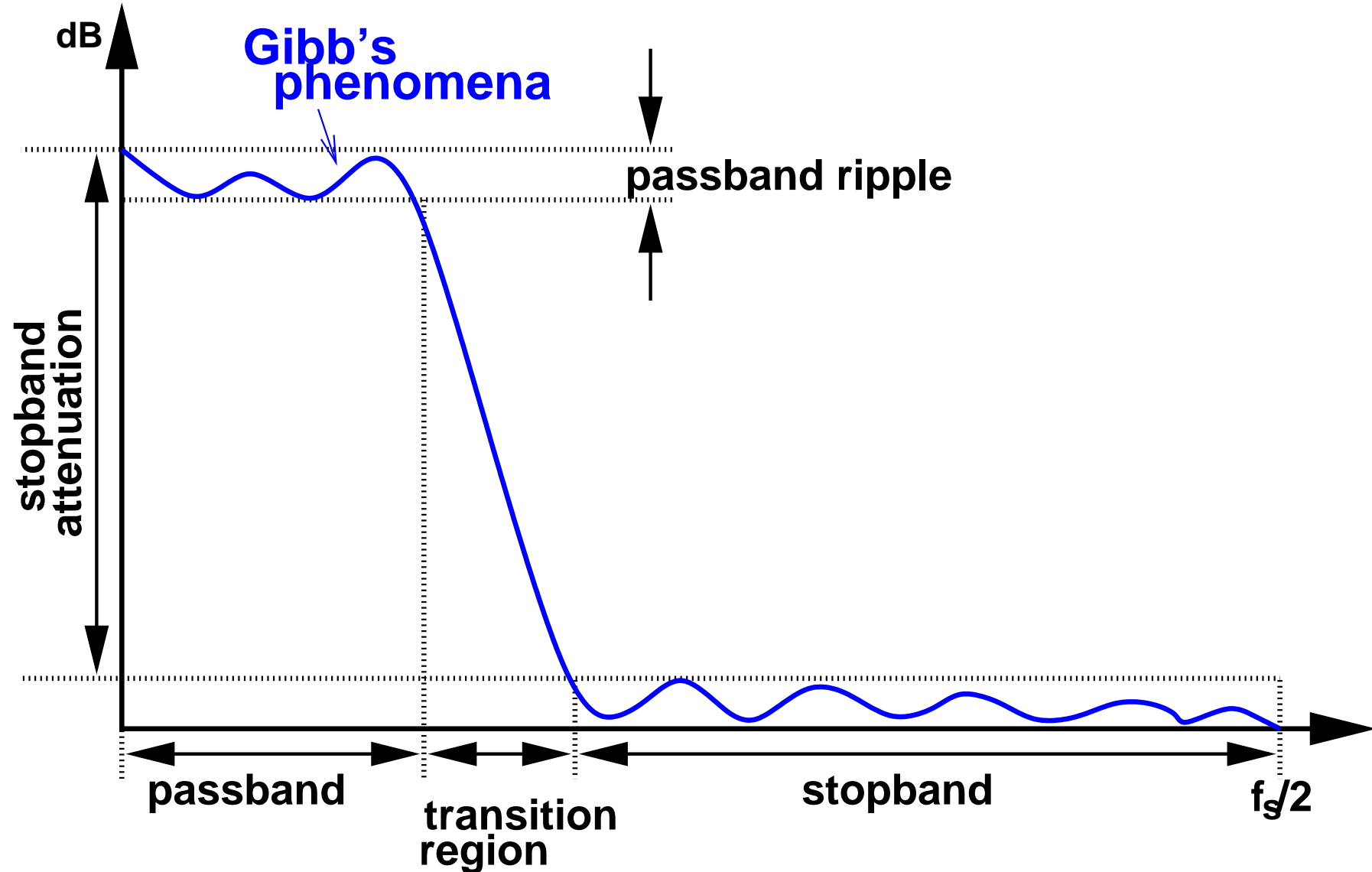
Better filters

These don't look like very good filters?

- they don't have a very distinct **pass band**
- the **transition region** between **pass band** and **stop bands** is large
- the **stop-band attenuation** is poor

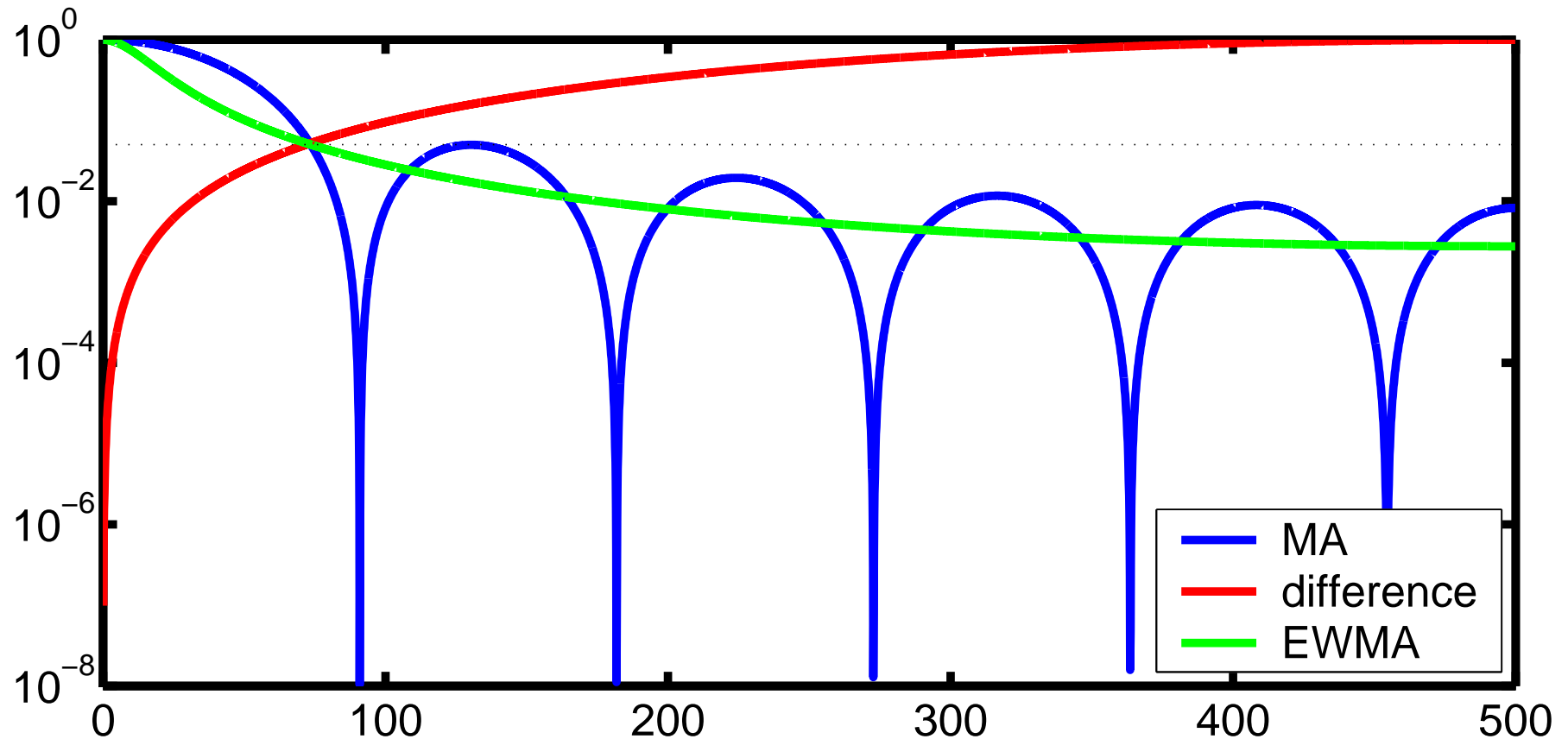
It would be nice to have filters with better properties, so we can more precisely specify filter out particular bands.

Terminology



Example filter comparison

Stop-band attenuation $\simeq -13$ dB



Filtering in the frequency domain

We could filter thus:

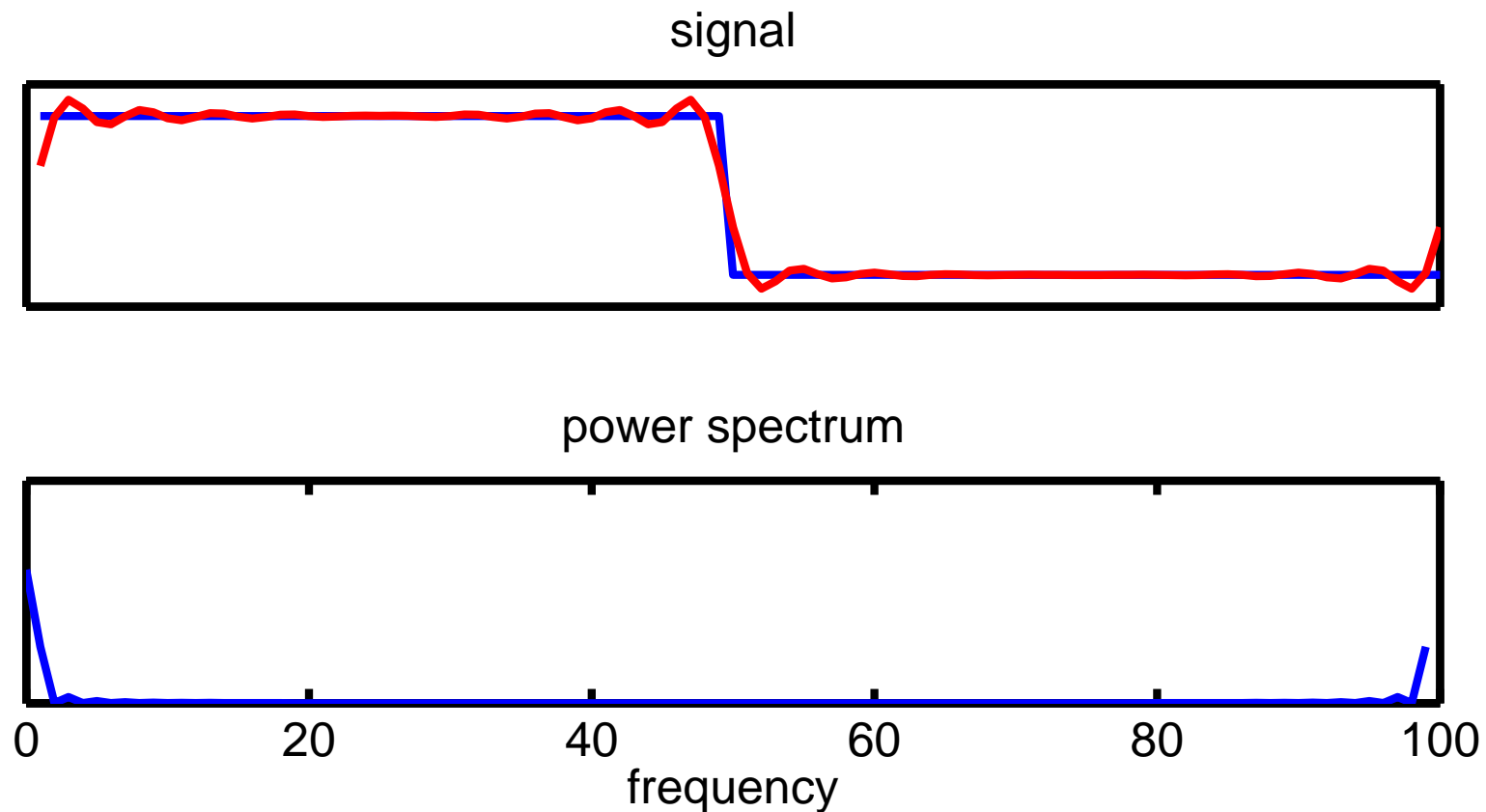
- `fft`
- `filter`
- `ifft`,

but this requires $O(N \log N)$ operations, which grows non-linearly in N . For many applications, we can't afford to have filtering operations grow faster than $O(N)$, e.g. real-time applications,

- The number of data points will be $f_s T$
- the time available for computation is T
- time available per data point is $1/t_s$, which is constant with respect to N .

Perfect filters and Gibb's phenomena

Filtering in frequency domain might not give you what you want. For example, rectangular low-pass filter to smooth the data.



Creates Gibb's phenomena in time (worse in images).

Filtering in the time domain

We see that there are two possible representations for linear, time-invariant filters

- frequency domain
- time domain

We can convert between them, but what we want is a filter that has good properties in both domains.

- good stop-band attenuation, and short transition region, with not too much ripple
- short number of **taps**, and not too much ripple.

Z-transforms

This is a convenient point to introduce a new (though closely related) transform called the Z-transform, which is ideal for analyzing LTI filters.

Z-transforms

The Z-transform is defined by

$$W(z) = \sum_{n=-\infty}^{\infty} w(n)z^{-n}$$

Similar to Probability Generating Functions (PGF)

$$P(z) = \sum_{n=-\infty}^{\infty} p_n z^n$$

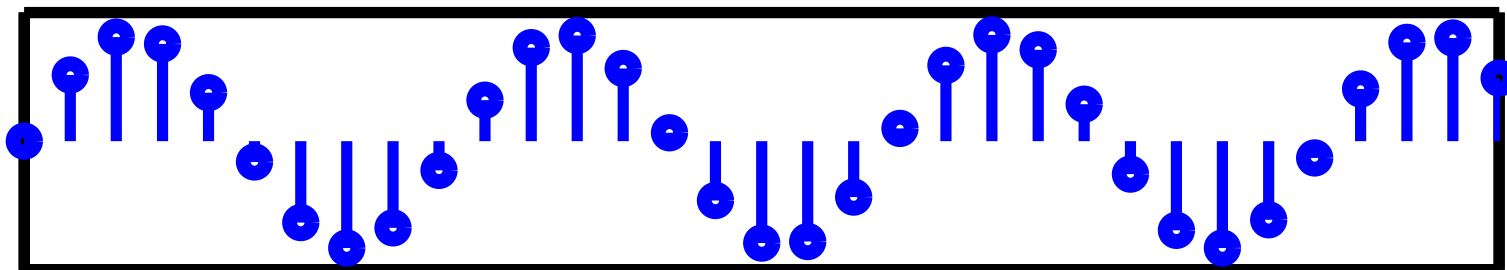
The Z-transform extends the Fourier transform onto the complex plane

- note that $W(e^{i2\pi k}) = F(k)$, where $F(k)$ is the FT of w

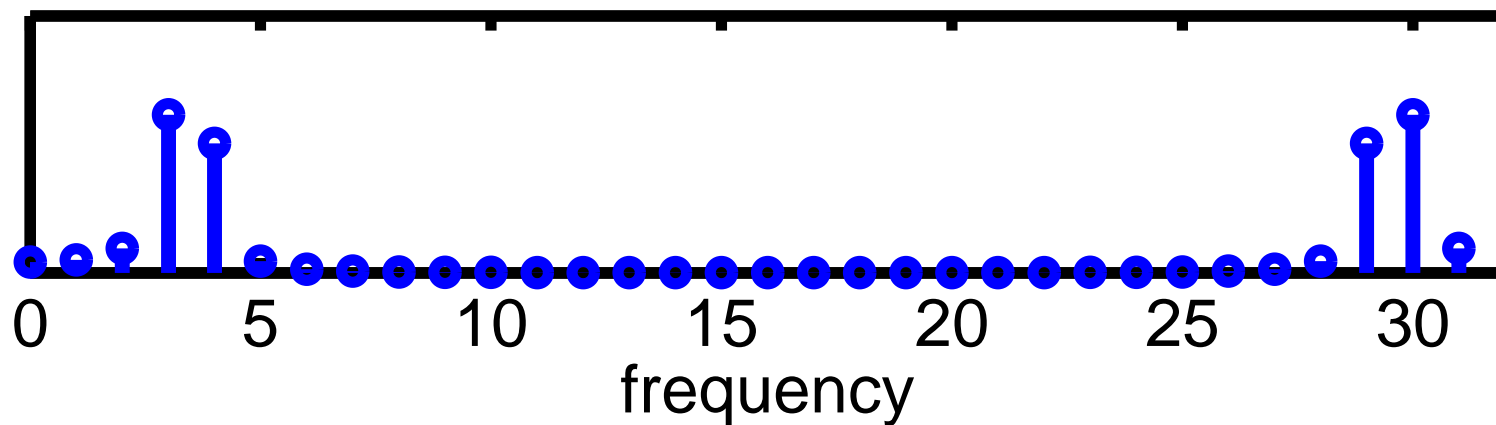
Z-transforms

A Fourier transform

signal

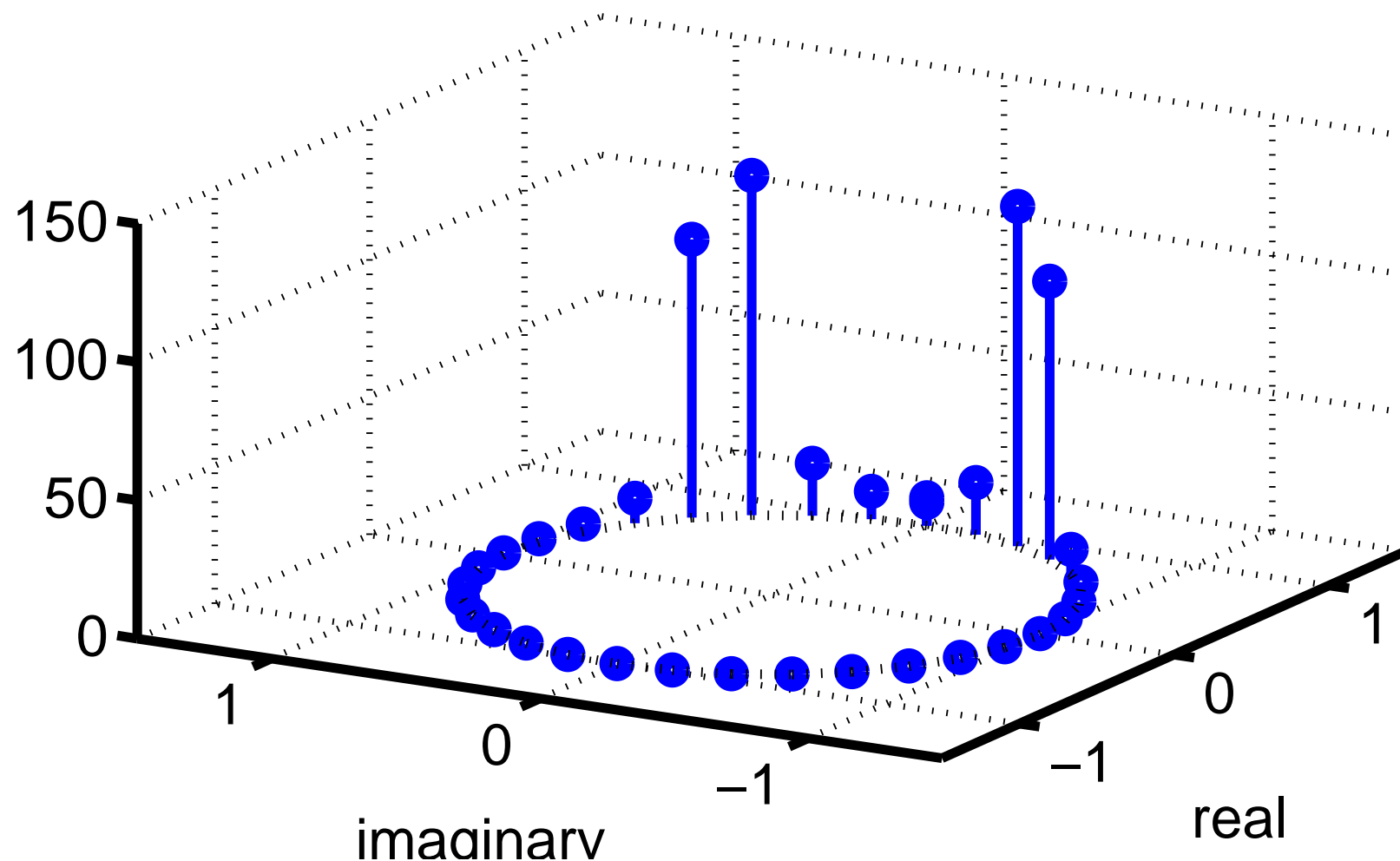


power spectrum



Z-transforms

A Fourier transform viewed as special points of the Z-transform



Z-transform and convolutions

Given a discrete convolution

$$w(n) = [x * y](n) = \sum_{i=-\infty}^{\infty} x(i)y(n-i) = \sum_{i=-\infty}^{\infty} x(n-i)y(i)$$

then the Z-transform of w is

$$W(z) = X(z)Y(z)$$

where $X(z)$ and $Y(z)$ are the Z-transforms of x and y .

Inverse Z-transform

Can see by analogy to the DFT that we could invert by integrating the Z-transform around the unit circle in the complex plane.

In fact we can use any counter-clockwise contour integral which goes around all of the poles of the Z-transform.

$$w(n) = \frac{1}{2\pi i} \oint_{\Gamma} W(z) z^{n-1} dz$$

where Γ is such a contour in the complex plane.

More general IIR filters

ARMA (Auto-regressive Moving Average)

$$y(n) = - \sum_{i=1}^p a(i)y(n-i) + \sum_{i=0}^q b(i)x(n-i)$$

Alternatively write this as two convolutions

$$\sum_{i=0}^p a(i)y(n-i) = \sum_{i=0}^q b(i)x(n-i)$$

Take Z-transform

$$A(z)Y(z) = B(z)X(z)$$

ARMA filters

$$Y(z) = \frac{B(z)}{A(z)}X(z)$$

- $A(z)$ and $B(z)$ are polynomials of degree p and q in z
- given a particular desired transfer function (written in Z-transform terms as $W(z)$), filter design problem is to approximate this using a rational polynomial $A(z)/B(z)$ of as low order as possible.
- $A(z)$ has p zeros in complex plane, called **poles**
- $B(z)$ has q zeros in complex plane, called **zeros**
- for causal, linear, time-invariant filter to be **stable** (BIBO), the poles have to be inside the unit circle.

Example: EWMA

$$y(n) = ay(n-1) + (1-a)x(n)$$

So

$$A(z) = 1 - az^{-1}$$

$$B(z) = 1 - a$$

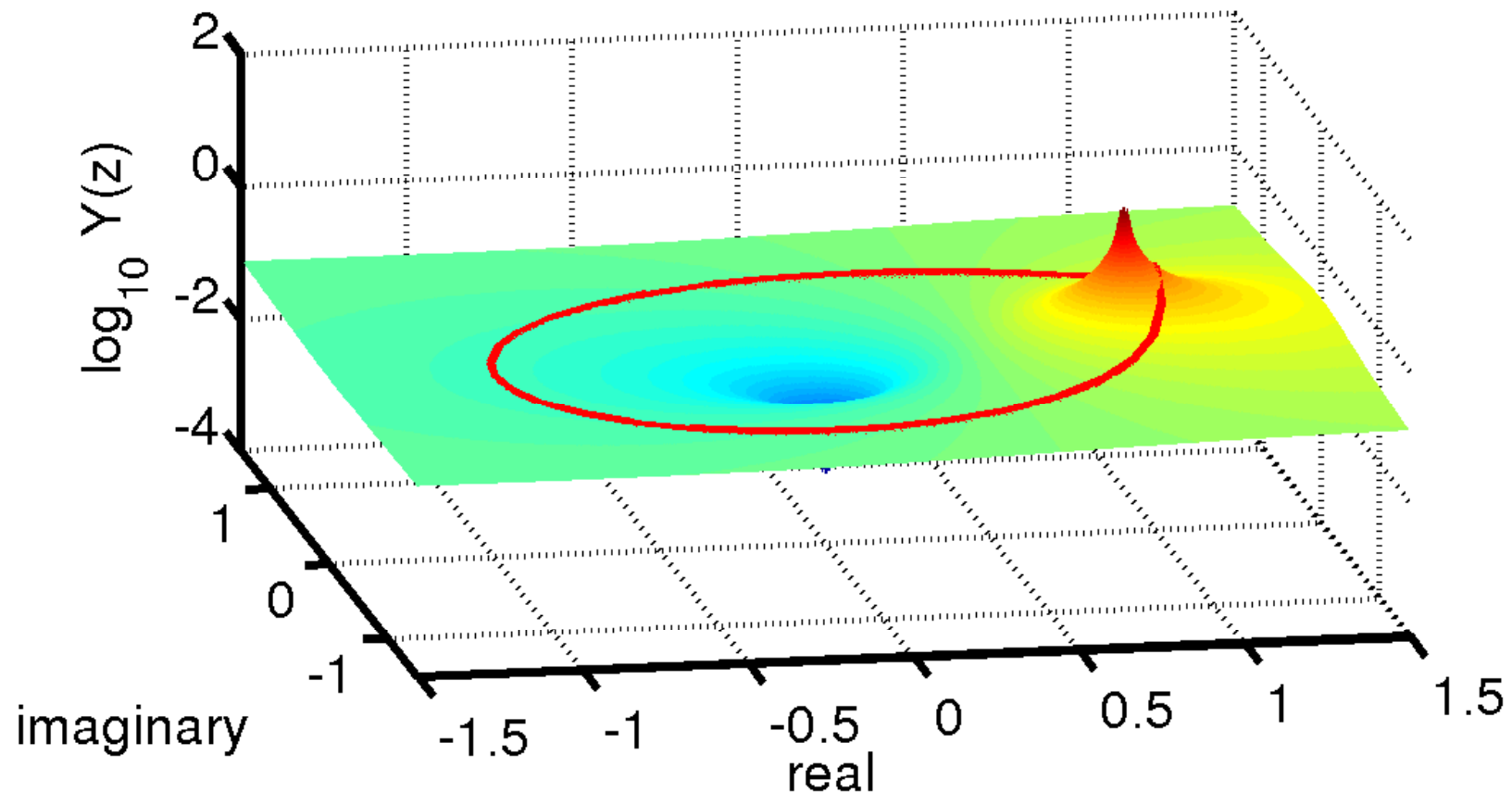
$$Y(z) = \frac{1-a}{1-az^{-1}} = \frac{(1-a)z}{z-a}$$

- single zero at 0
- single pole is at $z = a$

For the filter to be stable $|a| < 1$.

Example: EWMA

$$Y(z) = \frac{1-a}{1-az^{-1}}, \quad a = 0.9$$



Example: EWMA impulse response

The EWMA

$$y(n) = ay(n-1) + (1-a)x(n)$$

has Z-transform (where stable, i.e. $|a| < 1$)

$$Y(z) = \frac{1-a}{1-az^{-1}} = (1-a) \sum_{n=0}^{\infty} a^n z^{-n}$$

which we can invert by inspection to see that

$$y(n) = (1-a) \sum_{i=0}^{\infty} a^i x(n-i)$$

Hence the **Exponential** (or geometric) decrease in the impulse response of EWMA filter.

Filter invertibility

The transfer function of the inverse of a filter (with transfer function $H(z)$) should be

$$H^{-1}(z) = 1/H(z)$$

because the product of these two transfer functions should cancel.

- for a filter to be stable, the poles of $H(z)$ must lie inside the unit circle in the complex plane.
- when we invert, poles become zeros, and visa versa
- for the inverse to be stable, the zeros of $H(z)$ must lie inside the unit circle in the complex plane.

Some simple filters

- All zero filter = MA
- All pole filter = AR
- Laplace, Sobel, Prewitt (2D, next lecture)
- I have only really looked at magnitude, but phase is also important for filters.

Note that we only consider discrete filters here, there is an interesting set of problems in designing analogue filters.

Applications

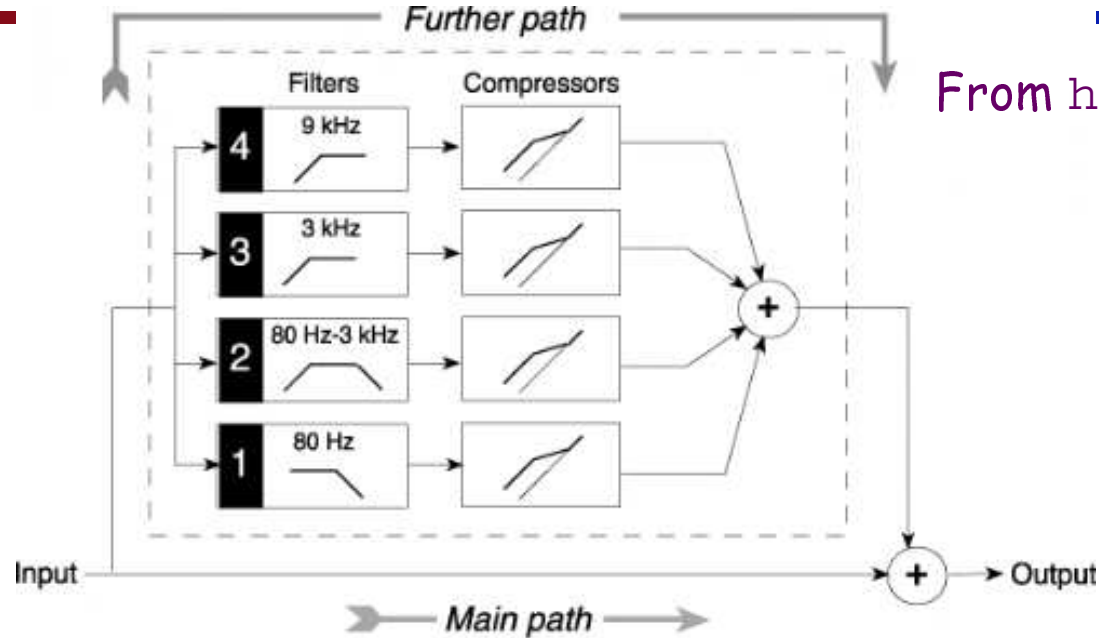
Applications of filters include noise reduction

Application: Dolby noise reduction

Goal: reduce the tape hiss on a cassette tape.

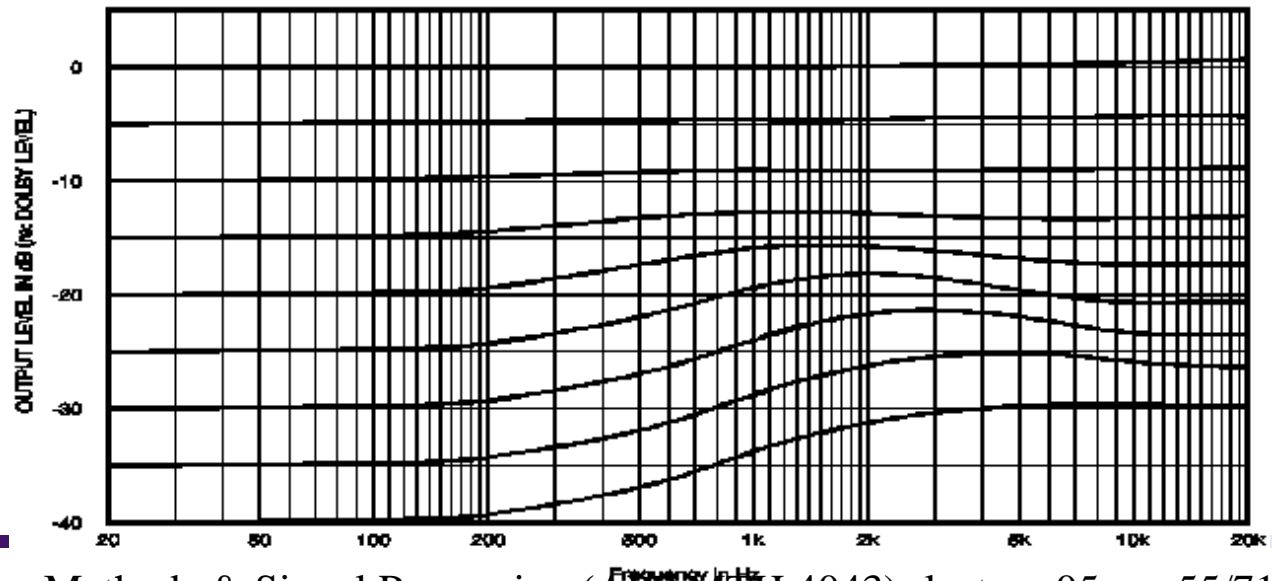
- Note, this is an analogue problem, not digital!
 - hiss results from imperfections in the analogue magnetic media on the tape
- Dolby A solution
 - note that tape hiss is a bigger problem at high frequencies, where there is less musical content to mask the hiss
 - two stage: used at both encoding and decoding
 - amplify higher frequency music content on recording
 - reverse effect on playback

Application: Dolby noise reduction



From <http://www.dolby.com/>

Dolby A-type compressor



An applications: denoising

Goal: remove noise from a signal

- types of noise

- white Gaussian noise, salt & pepper noise (because of appearance in images), or uncorrelated noise
- correlated Gaussian noise
- ticks and pops: small but high power bursts of noise

- single ended

- don't get to encode data on recording as with Dolby
- just get a signal including the noise

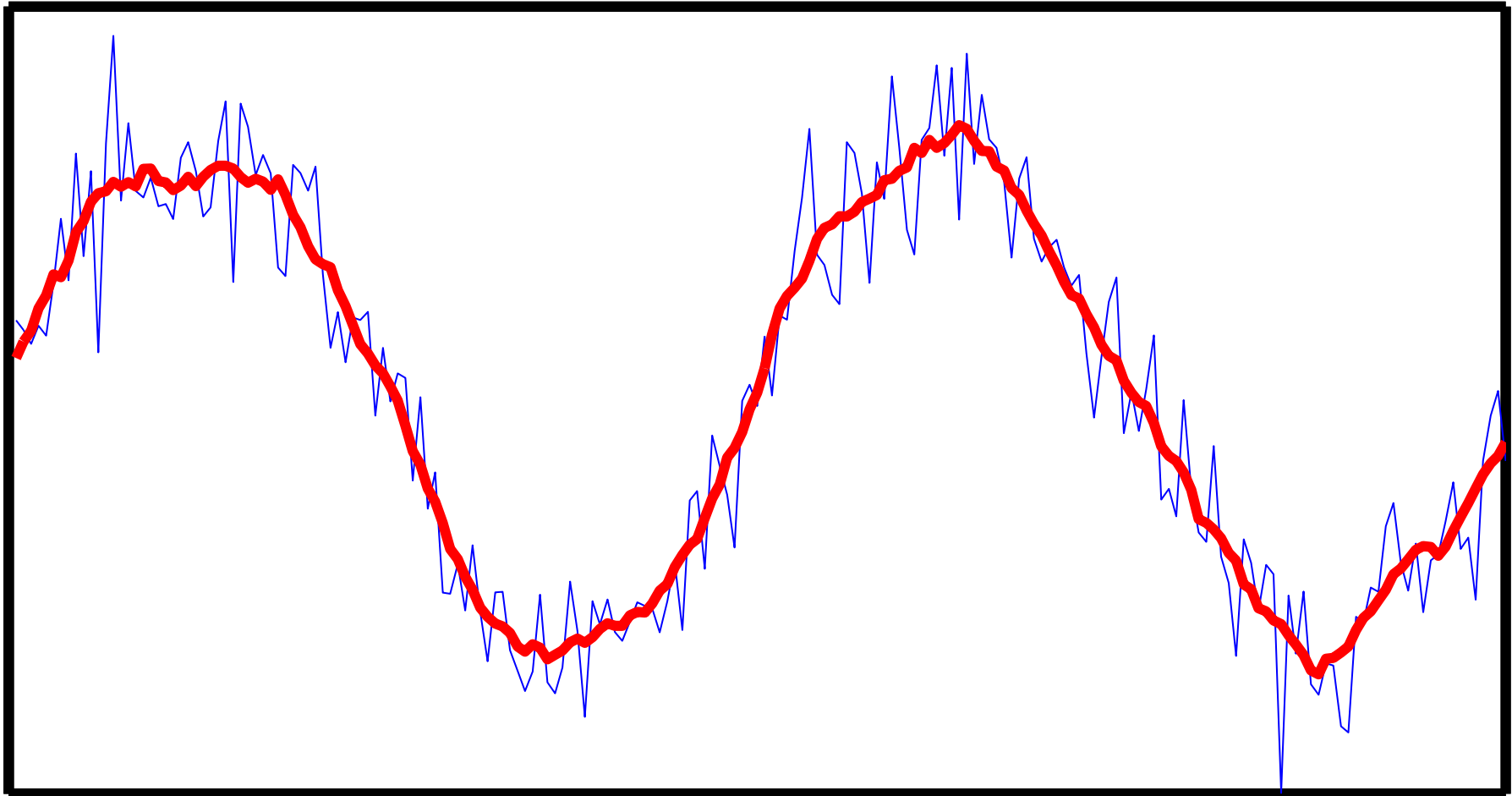
An applications: denoising

Goal: remove noise from a signal




- Approach: use a low-pass filter
- works because often high-frequency content isn't as important, but white noise is spread over the whole frequency spectrum.
 - works well for uncorrelated noise
 - not as good for correlated noise (spectrum is not uniform)
 - not much use for ticks and pops
- if high-frequency content is important, introduces artifacts
 - e.g. blurred edges in images

Example

Sinusoid + noise, filtered using rectangular MA $N = 5$



Example 2

- Music 
- Music plus white noise 
- Music plus white noise, filtered using rectangular
MA $N = 11$ 

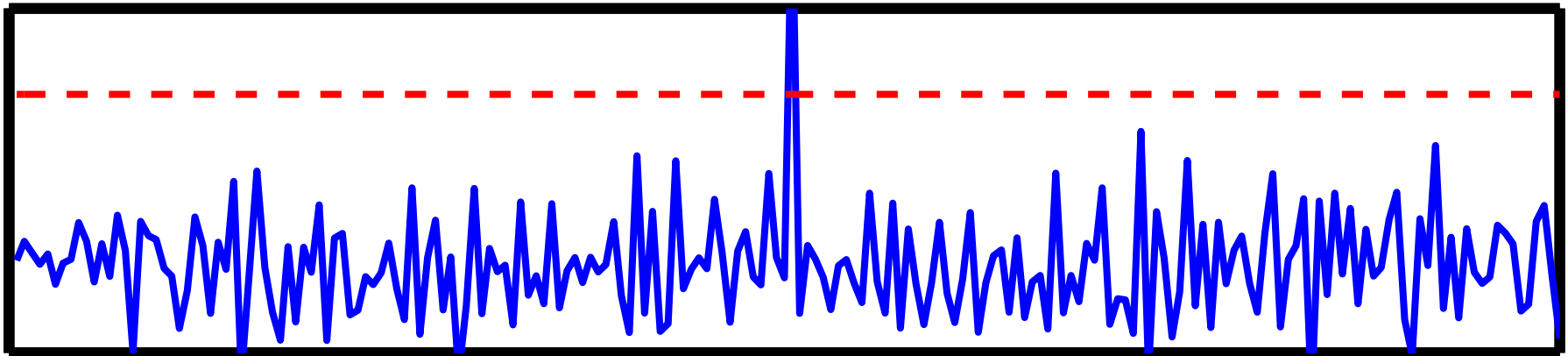
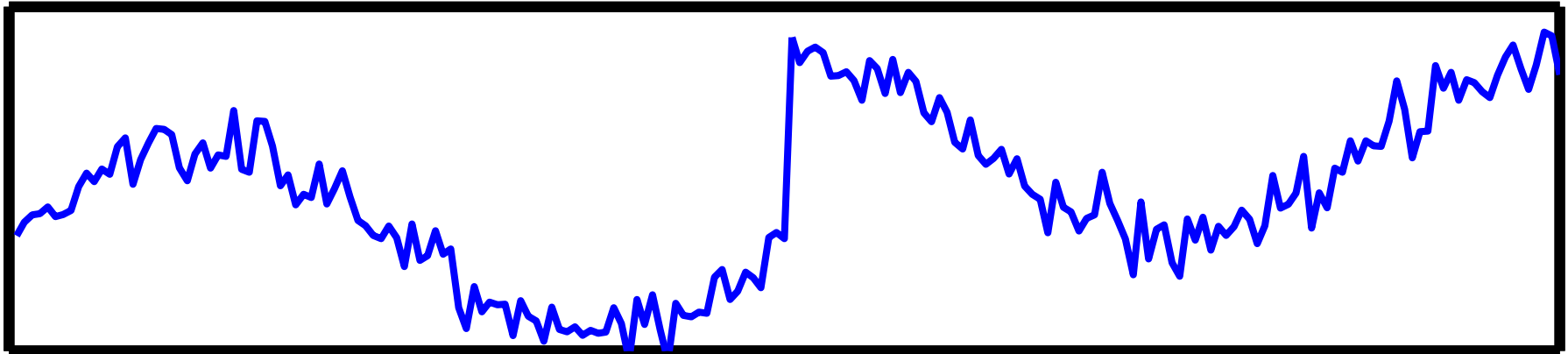
An applications: detect level changes

Goal, detect level changes in the signal

- Approach: use a high-pass filter
- edge detection in images
- often use something as simple as a differencer
- threshold on the filter output to detect changes

Example

Sinusoid + noise, with an edge, filtered using differencer



Linear systems

This is not a systems course, but the analogy between filters and systems is so close it would be a shame to miss the opportunity to compare the two.

Systems

A system is represented by a transformation of an input signal $x(t)$ into an output signal $y(t)$.



This might represent, e.g.

- a pendulum, or a vibrating string, where $x(t)$ is a forcing term, and $y(t)$ is the pendulum's position
- and electronic circuit where $x(t)$ is the input voltage and $y(t)$ is the output.
- A sensor where the input is the quantity to be sensed (e.g. temperature) and the output is what we see, e.g. the resistance of a thermistor.

System properties

- **invertibility:** The mapping $x(t) \rightarrow y(t)$ must be 1:1, so that each input signal has a unique output signal (don't need to invert all possible outputs).
- **memory:** $y(t_0)$ depends on $x(t)$ for $t \neq t_0$.
- **causality:** $y(t_0)$ only depends on $x(t)$ for $t \leq t_0$.
- **stability:** Bounded Input Bounded Output (BIBO).
If $|x(t)| \leq M$ for all t and some M , then $|y(t)| \leq R$ for all t and some R .
- **time invariance:** time shift doesn't matter, i.e.
 $x(t) \rightarrow y(t)$ implies $x(t - t_0) \rightarrow y(t - t_0)$.
- **linearity:** principle of superposition: $x_i \rightarrow y_i, i = 1, 2$
implies that for all $a_1, a_2 \in \mathbb{R}, a_1x_1 + a_2x_2 \rightarrow a_1y_1 + a_2y_2$.

Linear systems

Systems are just filters! The difference is

- we design filters for certain goals (e.g. low pass)
- systems occur in nature

Problems are different

- optimal design of filters
- estimation or control of system

Linear time-invariant systems

- Simple, tractable
- As with filters characterized by impulse response, or frequency response (transfer function)
- Frequency response is Fourier transform of impulse response.
- we don't (necessarily) design system, so we can't ensure linearity; its OK sometimes, but ...

Non-linear systems

An example

$$y(t) = x(t)^2$$

Given input signal $x(t) = \sin(2\pi f_0 t) + \sin(2\pi f_1 t)$

The output will be

$$\begin{aligned} y(t) &= \sin^2(2\pi f_0 t) + \sin^2(2\pi f_1 t) + 2 \sin(2\pi f_0 t) \sin(2\pi f_1 t) \\ &= \sin^2(2\pi f_0 t) + \sin^2(2\pi f_1 t) + \\ &\quad \cos(2\pi(f_0 - f_1)t) - \cos(2\pi(f_0 + f_1)t) \end{aligned}$$

We get frequencies that didn't exist in original signal!

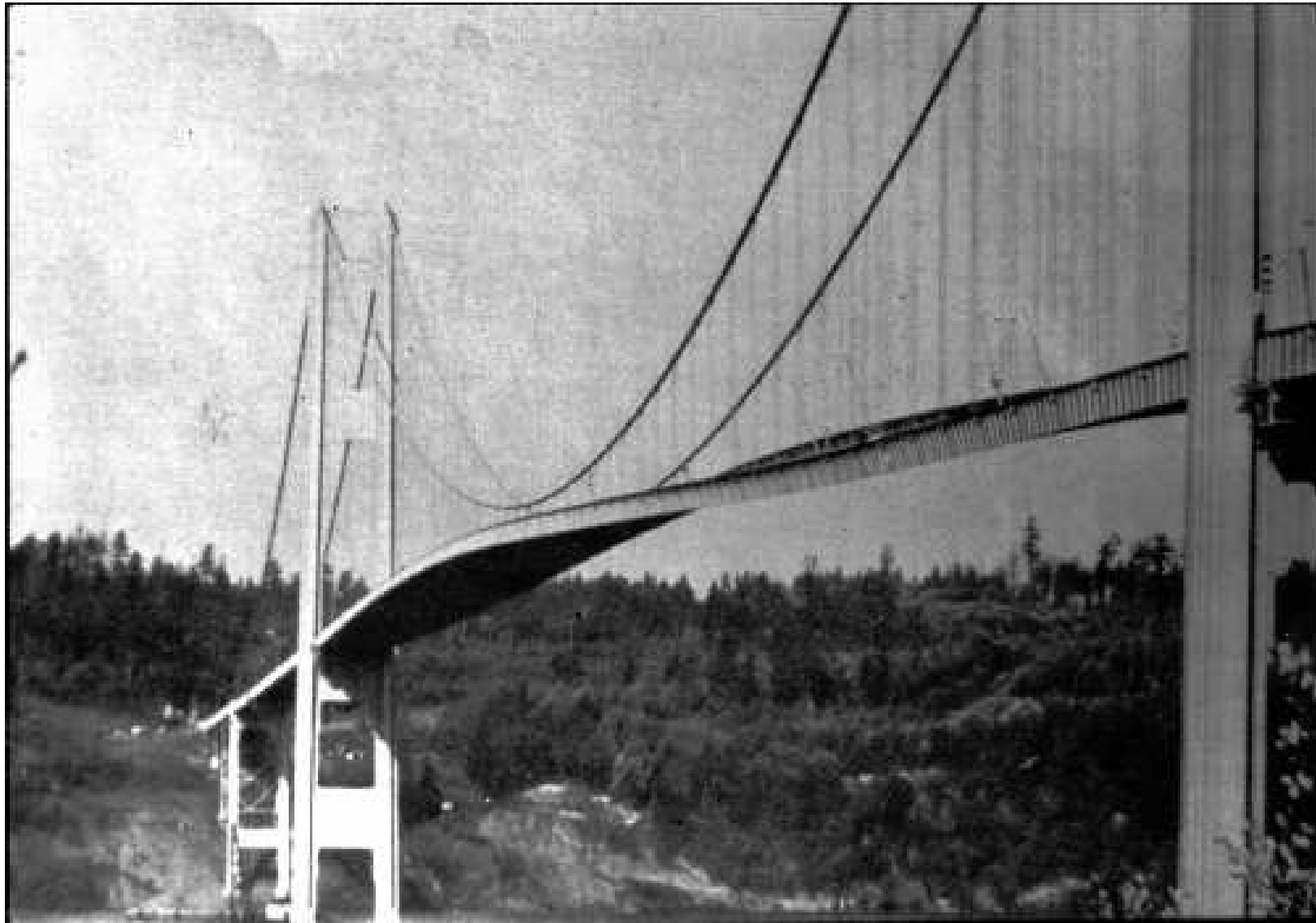
- linear systems can be characterized by transfer function, so they can only change the amount of a particular frequency component (and its phase), not introduce new frequencies

Resonance

Think of resonant system as

- narrow band pass filter
- input signal is enhanced at certain frequencies, and attenuated at all others
- examples
 - forced pendulum
 - plucked string
- may be unstable, e.g. Tacoma narrows

Tacoma narrows bridge



Tacoma narrows bridge



Tacoma narrows bridge

