Part 2
Signal Processing Systems

0368.3464

עיבוד ספרתי של אותות

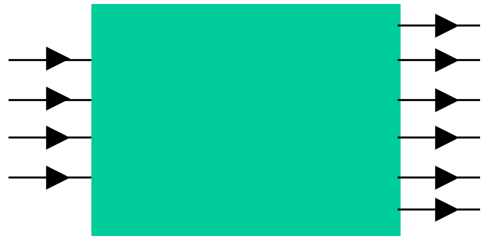
Digital Signal Processing for Computer Science

AKA

Digital Signal Processing – Algorithms and Applications

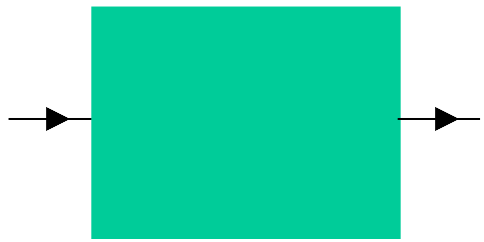
Systems

0 or more signals as inputs



1 or more signals as outputs

1 signal as input



1 signal as output

A **signal processing system** has signals as inputs and outputs

The most common type of system has a single input and output

A system is called ***causal***

if y_n depends on x_{n-m} for $m \geq 0$ but not on x_{n+m}

A system is called ***linear*** (note - does *not* mean $y_n = ax_n + b$!)

if $x_1 \rightarrow y_1$ and $x_2 \rightarrow y_2$ then $(ax_1 + bx_2) \rightarrow (ay_1 + by_2)$

A system is called ***time invariant*** if it has no internal clock

if $x \rightarrow y$ then $\hat{z}^n x \rightarrow \hat{z}^n y$

A system that is both **linear** and **time invariant** is called a ***filter***

Exercise time!

Which of the following are signal processing systems (we shall use x for inputs and y for outputs)? Explain. **x and y must be signals!**

1. The identity $y = x$
2. The constant $y = k$ irrespective of x
3. $y = \pm\sqrt{x}$
4. A device that inputs a pizza and outputs a list of its ingredients
5. $y = \sin(\frac{1}{t})$
6. $y(t) = \int_{-\infty}^{t} x(t)$
7. The Fourier transform
8. A television
9. A D/A converter

Example systems

- Identity system $y_n = x_n$
- Amplifier (gain) $y_n = g\, x_n$
- Saturator $y_n = \mathrm{sign}(x_n)$

  What does this do to a sinusoid ?

  How is it related to the amplifier ?

  Why is DSP better than electronics ? (see next slide)
- Time by time functions $y_n = f(x_n)$

These are not interesting since they don't involve *time*

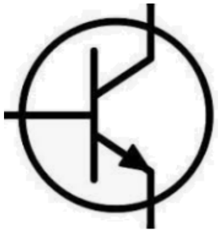
- Delay $y = \hat{z}^{-1}x$ (i.e., $y_n = x_{n-1}$)
- First time difference $y = \hat{\Delta}\, x$ (i.e., $y_n = x_n - x_{n-1}$)
- Smoother $y_n = \frac{1}{4} x_{n-1} + \frac{1}{2} x_n + \frac{1}{4} x_{n+1}$ (not causal!)

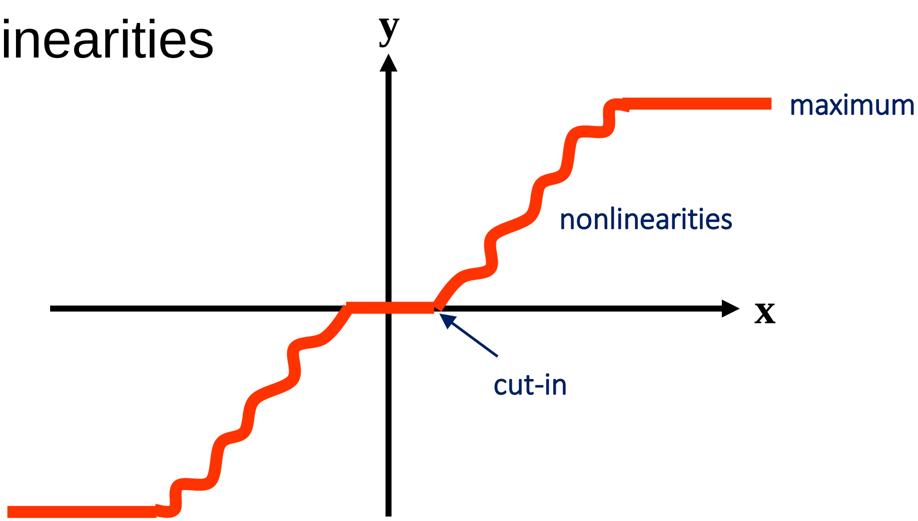
  How can we make it causal?

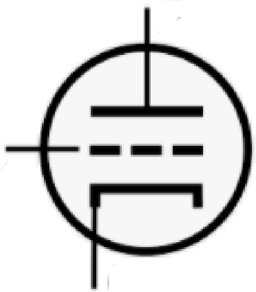
DSP is better than electronics

Analog electronic amplifiers have

- maximum output voltage (power supply voltage)
- cut-in voltage
- nonlinearities







In DSP we can multiply exactly

(we'll see later why overflow/underflow won't concern us)

Filters

Filters have a property in the frequency domain (the **filter law**)

$$\mathbf{Y(\omega) = H(\omega)\, X(\omega)} \qquad\qquad \mathbf{Y_k = H_k\, X_k}$$

In particular, if the input has no energy at frequency f

then the output also has no energy at frequency f

(what you get out of it depends on what you put into it)

This is the reason to call it a **filter**

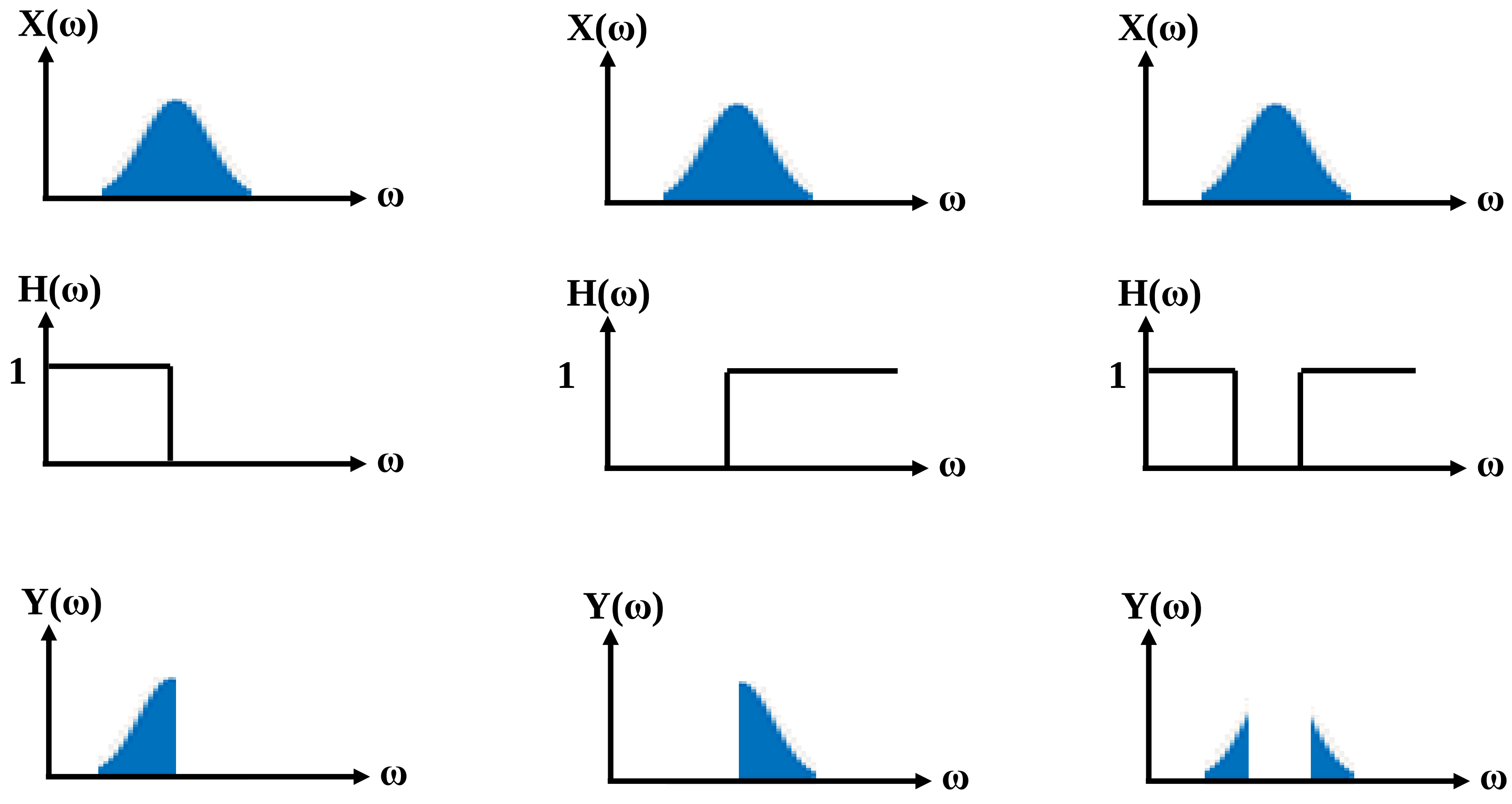
just like a colored light filter (or a coffee filter …)

Filters are used for many purposes, for example

- filtering out noise or narrowband interference
- separating two signals
- integrating and differentiating (why are these filters???)
- emphasizing or de-emphasizing frequency ranges

Why is the amplifier a filter? (explain why linear and TI, and in frequency domains)
What is $H(\omega)$ for the delay system ?

How does the *filter law* work?



H(ω) is called the ***frequency response***

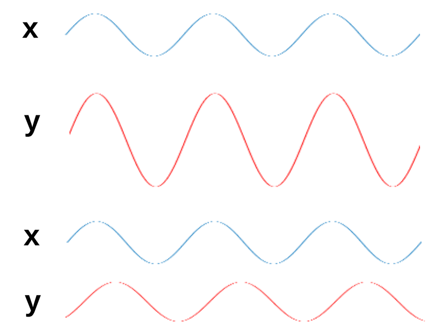
Frequency response

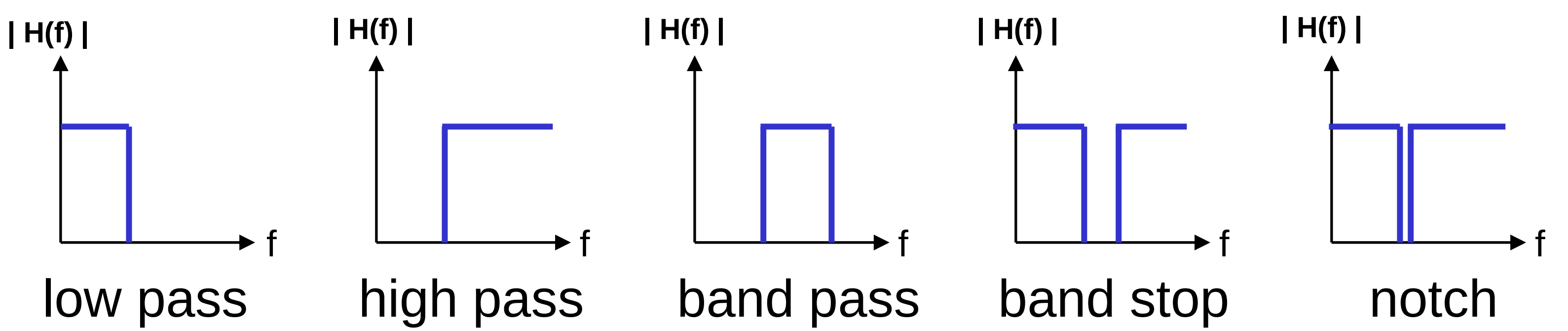
In general H(ω) is a *complex* number

- The absolute value is the gain
  how much the sinusoid is amplified or attenuated
- The phase is the phase shift
  how much the sinusoid is delayed

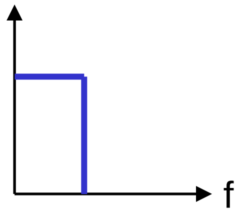
H(ω) is a function of ω
  a filter need not do the same thing to all frequencies!

Many time we use filters that are low-pass, high-pass, etc.
  but not all filters are like that

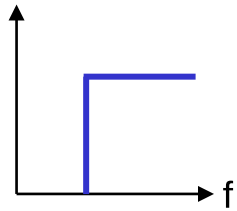




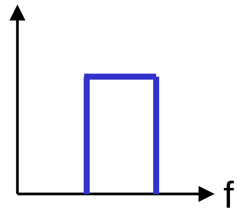
Types of filters



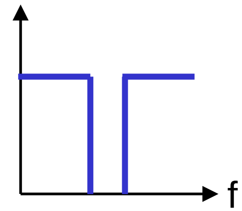
low pass



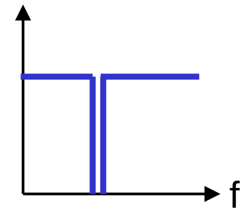
high pass



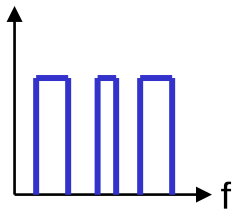
band pass



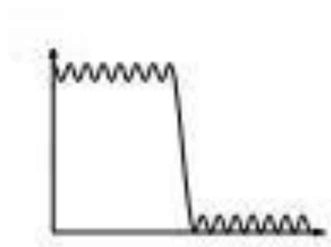
band stop



notch



multiband



realizable LP

When designing filters, we can specify :

- transition frequencies
- transition widths
- ripple in pass and stop bands
- linear phase (yes/no/approximate)
- computational complexity
- memory restrictions

What kind of analog filter is an anti-aliasing filter ?

Nonfilters

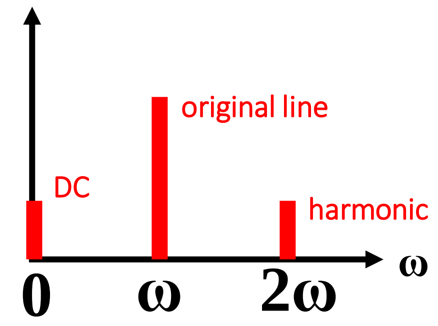
If a system is not linear it does not obey the filter law !

For example, $y_n = x_n + \epsilon\, x_n^2$ $\sin^2(\phi) = \frac{1}{2} - \frac{1}{2}\cos(2\,\phi)$

if $x_n = \sin(\omega n)$ then $y_n = \sin(\omega\, n) + \epsilon/2 - \epsilon/2 \cos(2\omega\, n)$

So the input spectrum has 1 component

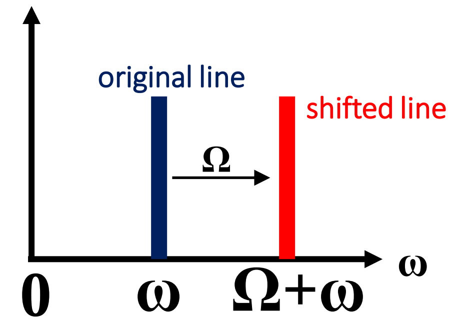
and the output spectrum has 3!



If a system is not time invariant it is not a filter!

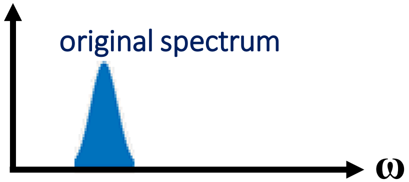
For example, $y(t) = e^{i\,\Omega\, t}\, x(t)$

if $x(t) = e^{i\,\omega\, t}$ then $y(t) = e^{i\,\Omega\, t}\, e^{i\,\omega\, t} = e^{i\,(\Omega + \omega)\, t}$



Why did we use complex exponentials here ?

What happens to a more general spectrum ?



Question 1

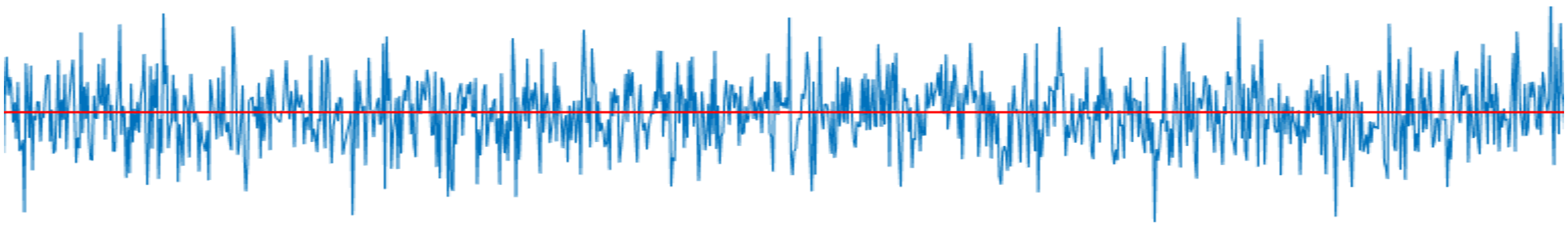
To understand our first kind of filter we'll look at an example

We know that a signal is DC (a constant $s_n = k$)

but only see a noisy version $x_n = s_n + v_n$

where the noise signal v_n is DC-free (zero average)

How do we discover k (recover s_n) ?



We **average** over as much time as we can

$k = < x_n > = < s_n + v_n > = < s_n > + < v_n > = < s_n > + 0$

In practice, we take N samples

$$k = \frac{1}{N} \sum_{n=0}^{N-1} x_n$$

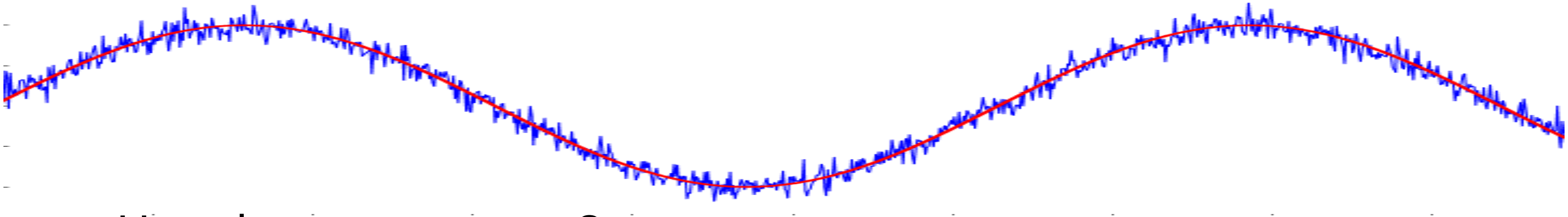
Question 2

We know that a signal s_n changes very slowly

(has only low frequencies in its spectrum)

but only see a noisy version $x_n = s_n + v_n$

where the noise signal v_n is DC-free (zero average)



How do we recover s_n ?

We **average** over a *window*

long enough for the noise to average out

but not so long as to destroy the signal

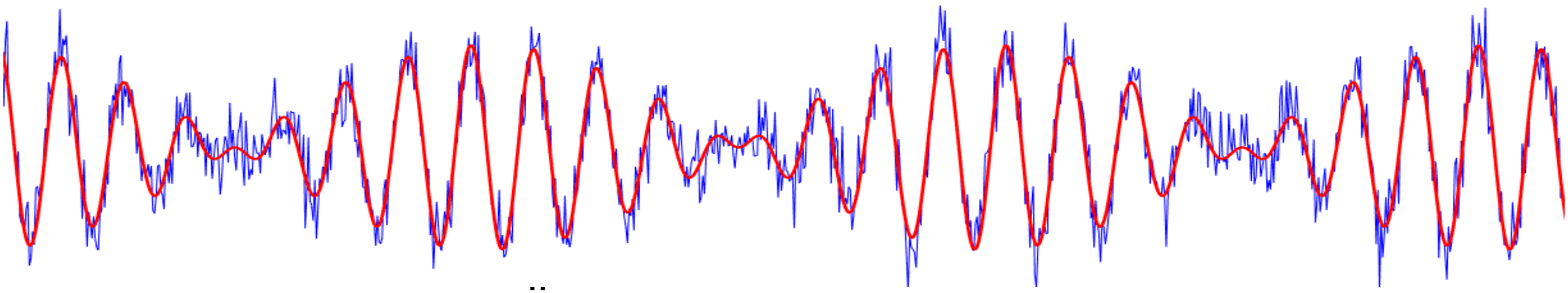
And then we move on to the next window

This is called **M**oving **A**verage

$$y_n = \frac{1}{L} \sum_{l=-L/2}^{+L/2} x_{n-l}$$ (if L is odd then 1/(L+1))

Question 3

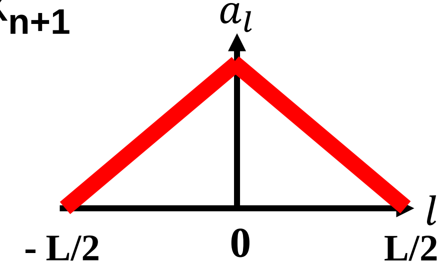
The same, but signal s_n doesn't change so slowly
(there are higher frequencies in its spectrum)



We perform a (generalized) **M**oving **A**verage but with non-equal coefficients

$$y_n = \sum_{l=n-L/2}^{n+L/2} a_l \, x_{n-l} \quad \text{where} \sum a_l = 1$$

For example, the *smoother* $y_n = \frac{1}{4} x_{n-1} + \frac{1}{2} x_n + \frac{1}{4} x_{n+1}$

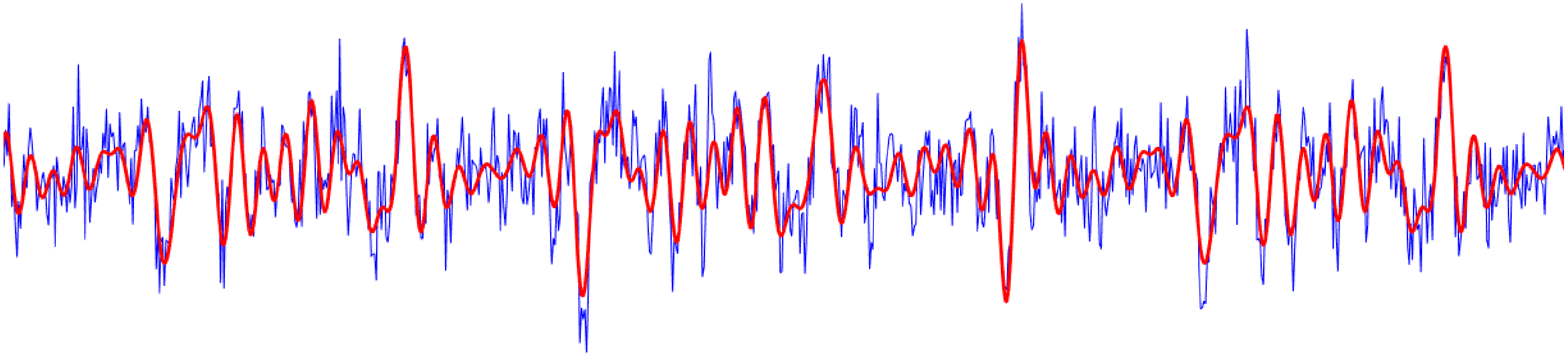


What coefficients return us to the original MA ?

Why do we often use *triangular* coefficients ?

Question 4

What if the signal s_n has a spectrum with all frequencies ?



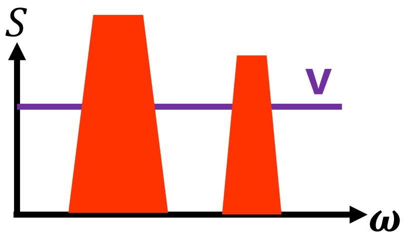
We can still perform **M**oving **A**verage

but need to find the coefficients based on the frequency domain

such that we allow the signal to pass

but block as much noise as possible

This will only work if MA is a filter!



MA is always a filter

Let's check that **M**oving **A**verage is a *filter* (linear and time invariant)

LINEARITY

If we multiply the input by a gain g

$$y_n' = \sum a_l \, g x_{n-l} = g \sum a_l \, x_{n-l} = g \, y_n \quad ✔$$

If we add two inputs u and v which give outputs x and y

$$\sum a_l (u+v)_{n-l} = \; + \sum a_l \; u_{n-l} \; + \sum a_l \; v_{n-l} \; = x_n + y_n \quad ✔$$

TIME INVARIANCE

If we shift the input signal by m times (m positive or negative)
and the coefficients don't change!

$$y_n' = \sum a_l \; x_{(n+m)-l} = \left(\sum a_l \; x_{j-l}\right)_{j=n+m} = y_{n+m} \quad ✔$$

Note that sometimes it is useful to have coefficients
that change slowly over time (to *adapt* to changing circumstances)

In which case we *almost* have a filter …

How to design a digital filter

20 years ago a large part of every DSP course was devoted to how to design digital filters, i.e., given H(ω) how to find a_l

It is *not* enough to take the function H(ω) and perform an iFT since in practice we would do this in the digital domain S_k and we would have no control over what happens between the discrete frequency points

Here is the algorithm I recommend today ☺

- Google **digital filter design software free download**
- Download and install
- We'll learn later about the different filter types
  for now pick MA (also called *FIR*) filter
- Enter or draw the desired frequency characteristics
- Press **compute coefficients**
- View the spectrum
- Try it out

Convolution

We saw that to filter out noise we used the signal processing system

$$y_n = \sum_{l=n-L/2}^{n+L/2} a_l \, x_{n-l}$$

This is not causal, a similar causal filter is

$$y_n = \sum_{l=0}^{L-1} a_l \, x_{n-l}$$

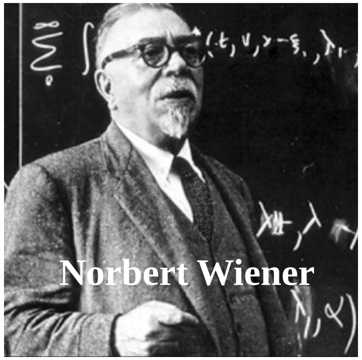
These forms of computation are called (finite) ***convolution***

Note that *convolution* is the sum of products
with one index **going up** and the other index **going down**
in this way the sum of the 2 indexes stays the same (n)
We could have made both indexes go in the same direction
which is called **correlation** (used to compare 2 signals x and y)

$$C_{x,y}(m) = \sum x_{l+m} \, y_l$$

Note that here the indexes both go up together!

Convolution (2)



The word convolution was invented by Norbert Wiener
the inventor of cybernetics and DSP

Some DSP courses emphasize *correlation*
and some emphasize *convolution*
the difference being relabeling the coefficients

We'll use convolution
and we'll see later why it is the best choice

Convolution (or correlation) appears in many DSP contexts
in fact, convolution is so important
that a processor that performs convolution optimally
is called a Digital Signal Processor

The echo cave 1

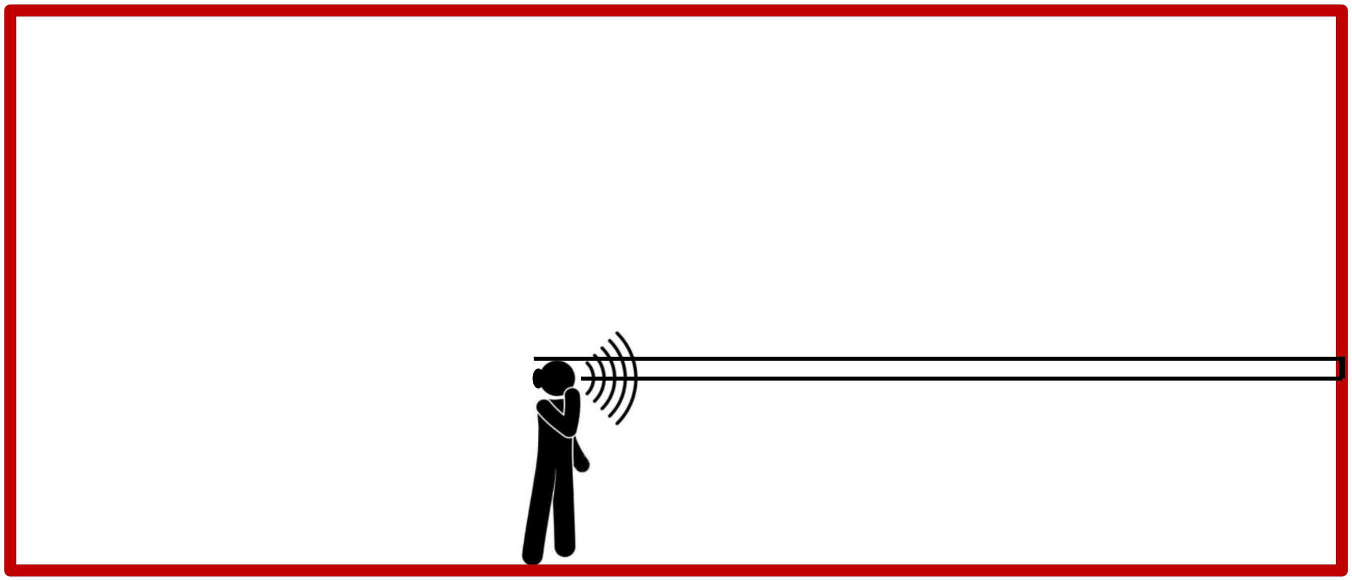
Here is another way convolution occurs

Consider shouting in a cave

the echo you hear is an attenuated copy

of what you shouted a roundtrip time ago

$$y_n = x_n + a\, x_{n-\ell} \qquad \text{where } \ell \text{ is the RTT}$$



The echo cave 2

But there can be *many* echoes

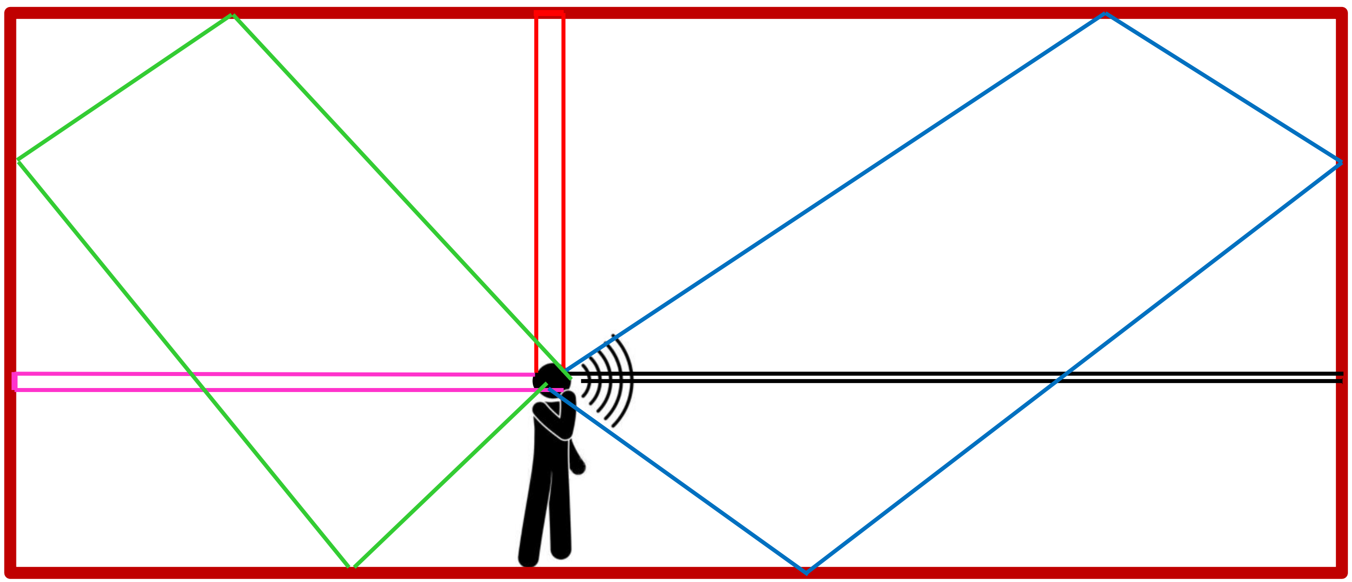
$$y_n = x_n + a_1\, x_{n-1} + a_2\, x_{n-2} + a_3\, x_{n-3} + a_4\, x_{n-4} + \ldots$$

If the longest possible echo returns after L times

$$y_n = \sum_{l=0}^{L-1} a_l\, x_{n-l} \quad \text{(where } a_0 = 1\text{, all other } 0 \le |a_l| < 1\text{)}$$

convolution!

What does $a_l < 0$ mean ?



You already know all about convolution!

LONG MULTIPLICATION

$$
\begin{array}{ccccccc}
 & & & B_3 & B_2 & B_1 & B_0 \\
 & & * & A_3 & A_2 & A_1 & A_0 \\
\hline
 & & & A_0B_3 & A_0B_2 & A_0B_1 & A_0B_0 \\
 & & A_1B_3 & A_1B_2 & A_1B_1 & A_1B_0 & \\
 & A_2B_3 & A_2B_2 & A_2B_1 & A_2B_0 & & \\
A_3B_3 & A_3B_2 & A_3B_1 & A_3B_0 & & & \\
\hline
\end{array}
$$

$$C_{\mathbf{3}} = A_0B_3 + A_1B_2 + A_2B_1 + A_3B_0$$

You learned about convolution in grade school,

and then again in high school!

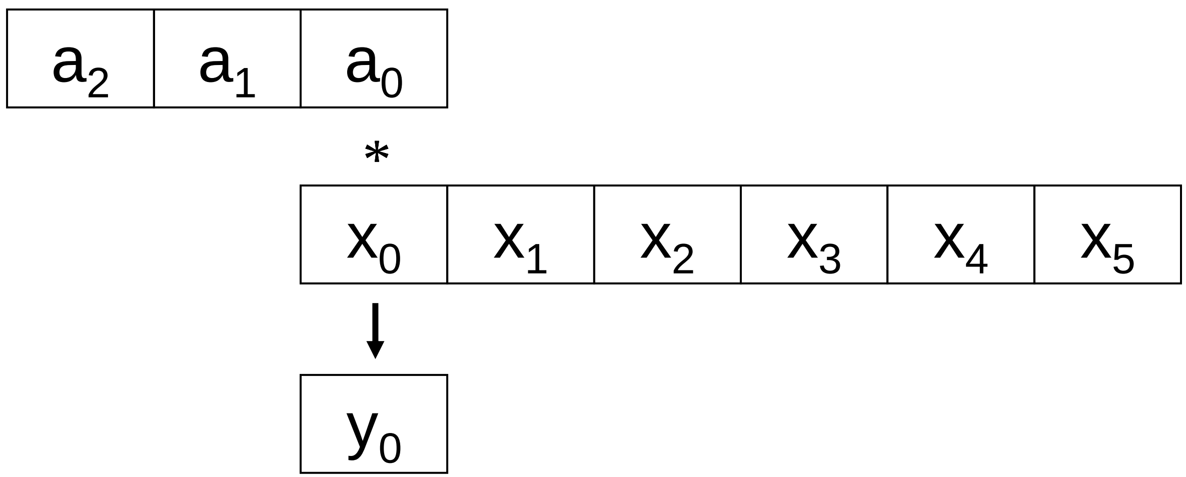
POLYNOMIAL MULTIPLICATION

$$(a_3\, x^3 + a_2\, x^2 + a_1\, x + a_0)\ (b_3\, x^3 + b_2\, x^2 + b_1\, x + b_0) =$$

$$a_3\, b_3\, x^6 + \ldots + (a_3\, b_0 + a_2\, b_1 + a_1\, b_2 + a_0\, b_3\,)\ x^3 + \ldots + a_0\, b_0$$

What's the connection between these?

Picturing Convolution - 0



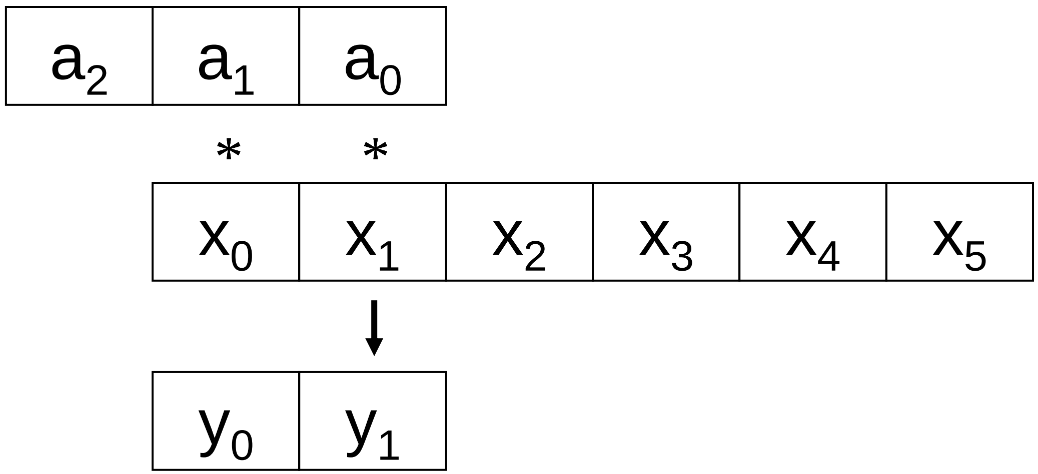
We chose the coefficients so that the indexes of a and x go in opposite directions

Note that the sum of the input indexes equals the output's index !

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l}$$

+

Picturing Convolution - 1



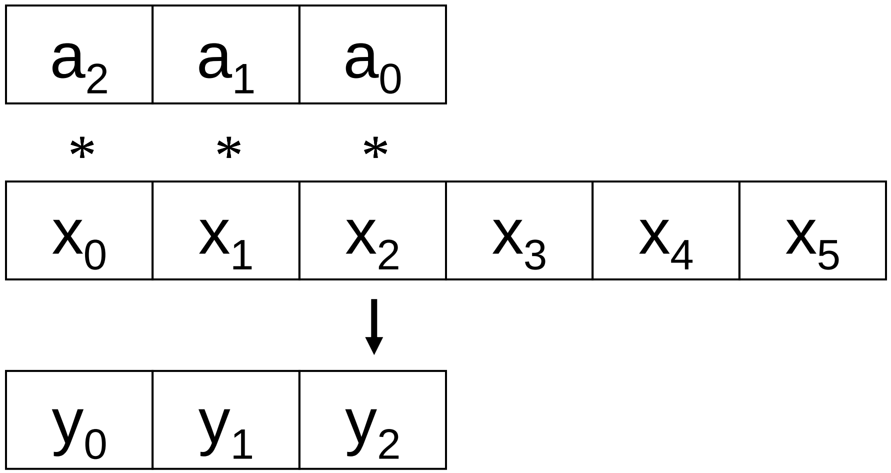
We chose the coefficients so that the indexes of a and x go in opposite directions

Note that the sum of the input indexes equals the output's index !

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l}$$

+

Picturing Convolution - 2



We chose the coefficients so that the indexes of a and x go in opposite directions

Note that the sum of the input indexes equals the output's index !

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l}$$

Picturing Convolution - 3

a_2	a_1	a_0

$*$ $*$ $*$

x_0	x_1	x_2	x_3	x_4	x_5

$\downarrow$

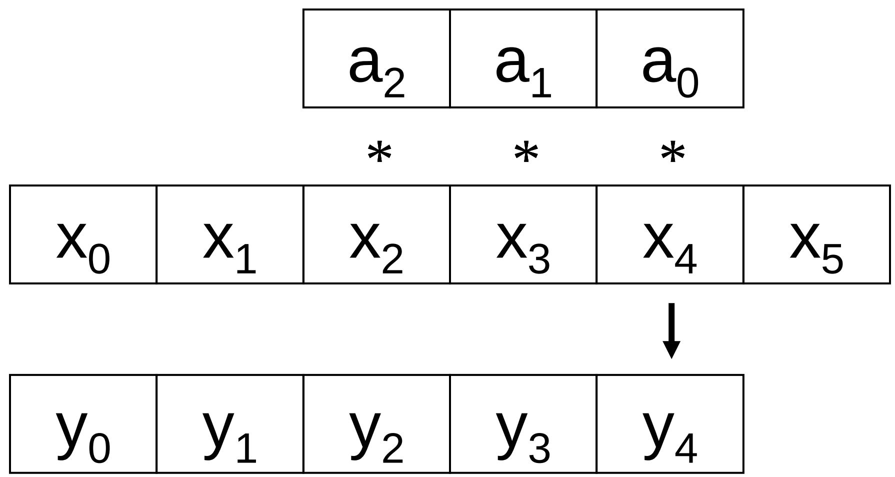
y_0	y_1	y_2	y_3

We chose the coefficients so that the indexes of a and x go in opposite directions

Note that the sum of the input indexes equals the output's index !

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l}$$

Picturing Convolution - 4



We chose the coefficients so that the indexes of a and x
go in opposite directions

Note that the sum of the input indexes equals the output's index !

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l}$$

+

Picturing Convolution - 5

			a_2	a_1	a_0
			*	*	*
x_0	x_1	x_2	x_3	x_4	x_5
					↓
y_0	y_1	y_2	y_3	y_4	y_5

We chose the coefficients so that the indexes of a and x go in opposite directions

Note that the sum of the input indexes equals the output's index !

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l}$$

+

Multiply and Accumulate (MAC)

How do we compute a convolution? (or a correlation?)

We iterate on a basic operation

$$y \leftarrow y + a_i * x_j$$

Since this **M**ultiplies a times x and then **AC**cumulates the answers it is called a **MAC**

The MAC is the most basic computational block in DSP

Even computing energy can be done using (degenerate) MACs

$$E \leftarrow E + x_i * x_i$$

Digital **S**ignal **P**rocessors are optimized to compute **MAC**s

In the frequency domain

Remember that in DSP
we are interested in time and frequency domains

We know what an MA filter does in the time domain – convolution!
What does it do in the frequency domain?

What does an MA filter do to a sinusoid of arbitrary frequency ω ?

Here it is much easier to use complex exponentials than sines

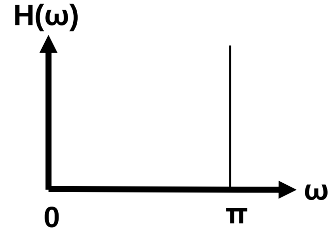
So we ask, what does an MA filter do to $x_n = e^{i\omega n}$ for arbitrary ω

We haven't proven it yet (don't worry – we will later)
but we said that for ***all*** filters $Y(\omega) = H(\omega)\, X(\omega)$

That means that sinusoids are *eigensignals* of filters

$$\text{MA-filter}(\, e^{i\omega n}\,) = H(\omega)\, e^{i\omega n}$$

$H(\omega)$ is called the MA frequency response $(0 \le \omega \le \pi)$



Why do we look at $H(\omega)$ for all ω instead of H_k ?
Why don't we look at $H(\omega)$ for negative ω ?

Simple MA Frequency response

Let's start with a simple noncausal 3-point MA

$$y_n = \frac{1}{3}(x_{n-1} + x_n + x_{n+1})$$

First let's ask what this filter does to DC

remembering that for DC we can take $x_n = 1$ (for all n)

$y_n = \frac{1}{3}(1+1+1) = 1$ (for all n) so y is also DC (of course - it had to be!)

and $H(\text{DC}) = y_n / x_n = 1$

Next let's ask what it does to Nyquist frequency ($\omega = \pi$)

remembering that for Nyquist we take $x_n = \ldots -1 +1 -1 +1 \ldots$

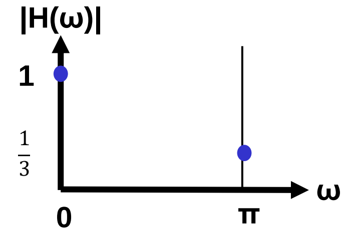
For even n: $y_n = \frac{1}{3}(-1+1-1) = -\frac{1}{3}$ and for odd n: $y_n = \frac{1}{3}(+1-1+1) = \frac{1}{3}$

So y_n is $\ldots +\frac{1}{3} - \frac{1}{3} + \frac{1}{3} - \frac{1}{3} \ldots$ which is also a Nyquist signal (it had to be!)

and $H(\omega) = y_n / x_n = -\frac{1}{3}$

We'll only care about $|H(\omega)|$ for now, and $|H(\text{Nyquist})| = \frac{1}{3}$

Simple MA Frequency response (cont)

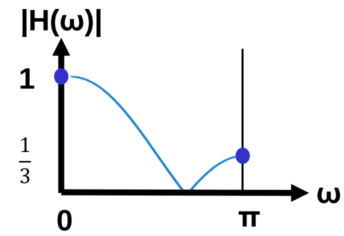


We have already found 2 points on the |H(ω)| plot

Now let's find the rest!

We substitute $x_n = e^{i\omega n}$ for arbitrary ω (0 ≤ ω ≤ π)

$$
\begin{aligned}
y_n &= \frac{1}{3}\left(e^{i\omega(n-1)} + e^{i\omega n} + e^{i\omega(n+1)}\right) \\
&= \frac{1}{3}\left(e^{i\omega n}e^{-i\omega} + e^{i\omega n} + e^{i\omega n}e^{+i\omega}\right) \\
&= \frac{1}{3}\left(e^{-i\omega} + 1 + e^{+i\omega}\right)e^{i\omega n} \\
&= \frac{1}{3}\left(1 + 2\cos(\omega)\right)e^{i\omega n}
\end{aligned}
$$



So y_n is a constant times $e^{i\omega n}$ (of course - it had to be!)

and $H(\omega) = y_n \,/\, x_n = \frac{1}{3}\left(1 + 2\cos(\omega)\right)$

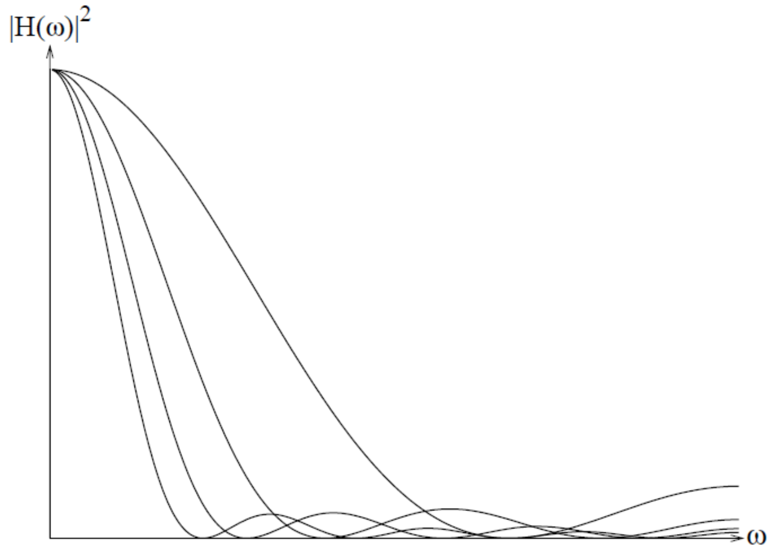
Why isn't this a nice enough low-pass filter?

What about averaging more?

The frequency response of the more general averaging filter

$$y_n = \frac{1}{2L+1} \sum_{l=-L}^{L} x_{n+l} \qquad \text{is} \qquad \frac{\sin(\frac{Lx}{2})}{L\sin(\frac{x}{2})}$$

proof in textbook



The more we average the more low-pass the filter becomes!

Why?

Frequency response 2

As another example let's look at the simple smoother filter

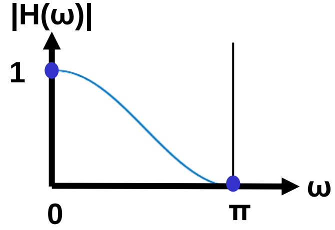
$$y_n = \tfrac{1}{4}\, x_{n-1} + \tfrac{1}{2}\, x_n + \tfrac{1}{4}\, x_{n+1}$$

For DC $y_n = \frac{1}{4} + \frac{1}{2} + \frac{1}{4} = 1$ (for all n) so H(DC) = $y_n / x_n = 1$

For Nyquist (even n) $y_n = \frac{1}{4}(-1) + \frac{1}{2}(+1) + \frac{1}{4}(-1) = 0$ so H(Nyquist) = 0

For general frequency we substitute $x_n = e^{i\omega n}$

$$\begin{aligned} y_n &= \tfrac{1}{4}e^{\mathrm{i}\omega(n-1)} + \tfrac{1}{2}e^{\mathrm{i}\omega n} + \tfrac{1}{4}e^{\mathrm{i}\omega(n+1)} \\ &= \tfrac{1}{4}e^{\mathrm{i}\omega n}e^{-\mathrm{i}\omega} + \tfrac{1}{2}e^{\mathrm{i}\omega n} + \tfrac{1}{4}e^{\mathrm{i}\omega n}e^{+\mathrm{i}\omega} \\ &= \left(\tfrac{1}{4}e^{-\mathrm{i}\omega} + \tfrac{1}{2} + \tfrac{1}{4}e^{+\mathrm{i}\omega}\right)e^{\mathrm{i}\omega n} \\ &= \tfrac{1}{2}\left(1 + \cos(\omega)\right)e^{\mathrm{i}\omega n} \end{aligned}$$



y is a sinusoid of the same frequency (of course – it has to be!)

and $|H(\omega)| = ½\ (\ 1 + 2\cos(\omega)\ )$ Why is this better?

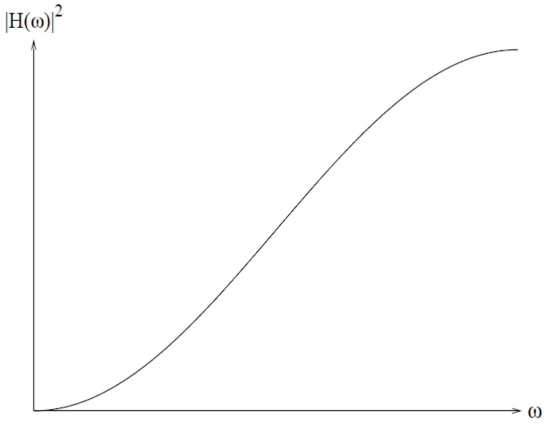
The first finite difference

Last example - the first finite difference in the frequency domain

$$y = \hat{\Delta}\, x \quad \text{(i.e., } y_n = x_n - x_{n-1} \text{)}$$

$H(\omega)\, e^{i\omega n} = e^{i\omega n} - e^{i\omega(n-1)}$ so $H(\omega) = 1 - e^{-i\omega} = e^{-i\omega/2}\left(e^{i\omega/2} - e^{-i\omega/2}\right)$

So $|H(\omega)| = 2 \sin(\omega/2)$



This is a high-pass filter!

Why must it be high-pass?

Why is this complex (i.e., why does it have a phase shift)?

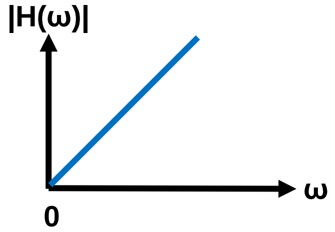
Differentiation and Integration

What does the analog derivative look like in the frequency domain?

Here is it easy enough to use sines

The derivative of x(t) = sin(ωt) is y(t) = $\frac{d\,x(t)}{dt}$ = ω cos(ωt)

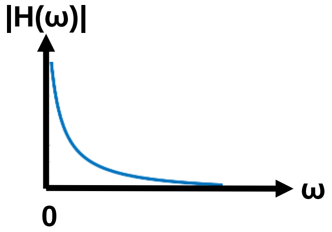
so | H(ω) | = ω   and there is a 90 degree phase shift



What about the analog integral?

The integral of x(t) = sin(ωt) is y(t) = $\int x(t)dt$ = – (1/ω) cos(ωt)

so | H(ω) | = 1/ω   and there is a 90 degree phase shift



Convolution and multiplication

We already saw 2 connections between convolution

multiplying numbers and polynomials are convolutions

But we now understand a deeper connection

The filter law means that a *convolution* in the time domain

$$\mathbf{y_n} = \sum_{l=0}^{L-1} a_l \, x_{n-l}$$ many people even write $y = a * x$

corresponds to a *multiplication* in the frequency domain $\mathbf{Y_k = H_k \, X_k}$

So, instead of convolving a and x in the time domain

we can move to the frequency domain and multiply

a_5	a_4	a_3	a_2	a_1	a_0
*	*	*	*	*	*
x_0	x_1	x_2	x_3	x_4	x_5

6*6 multiplications! ↓

y_0	y_1	y_2	y_3	y_4	y_5

→

A_0	A_1	A_2	A_3	A_4	A_5
*	*	*	*	*	*
X_0	X_1	X_2	X_3	X_4	X_5
↓	↓	↓	↓	↓	↓
Y_0	Y_1	Y_2	Y_3	Y_4	Y_5

6 multiplications

Why 6*6 and not 1+2+3+4+5+6 = 21 ?

The complexity of convolution

In DSP we use either the time domain or the frequency domain whichever is better for the task at hand

To perform convolutions over N elements in the time domain

$\mathbf{y_n} = \sum_{l=0}^{N-1} a_l \, x_{n-l}$ for n = 0 … N-1

requires N times N multiplications (and another N*(N-1) additions) and so the complexity is $O(N^2)$

To perform the same thing in the frequency domain $\mathbf{Y_k = H_k X_k}$ requires only N multiplications

But if we are working in the time domain

we need to first convert a and x into frequency domain A and X

and at the end convert Y back into time domain y

To do that we need to perform 2 DFTs $X_k = \sum_{n=0}^{N-1} W_N^{-nk} \, x_n$

and 1 iDFT $y_n = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{-nk} \, Y_k$ each of which is $O(N^2)$!

What we really need is a lower complexity DFT algorithm!

AR

Computation of convolution is ***iteration***

In CS there is a more general form of 'loop' - ***recursion***

Example: let's average values of input signal up to present time

$$
\begin{aligned}
y_0 &= x_0 &&= x_0 \\
y_1 &= (x_0 + x_1) / 2 &&= 1/2\, x_1 + 1/2\, y_0 \\
y_2 &= (x_0 + x_1 + x_2) / 3 &&= 1/3\, x_2 + 2/3\, y_1 \\
y_3 &= (x_0 + x_1 + x_2 + x_3) / 4 &&= 1/4\, x_3 + 3/4\, y_2 \\
y_n &= 1/(n+1)\, x_n + n/(n+1)\, y_{n-1} &&= (1-\beta)\, x_n + \beta\, y_{n-1}
\end{aligned}
$$

So the **present** output

depends on the **present input** and **previous outputs**

In DSP recursion is called **A**uto**R**egression (term invented by Udny Yule)

Note: to be time-invariant, β must be non-time-dependent (not like here!)

Unraveling the recursion

Given an AR form we can swap the recursion for an infinite iteration

For example, the simplest AR filter is $y_n = x_n + \beta\, y_{n-1}$

(we'll leave out the input gain for now)

$$
\begin{aligned}
y_n &= x_n + \beta y_{n-1} \\
&= x_n + \beta(x_{n-1} + \beta y_{n-2}) = x_n + \beta x_{n-1} + \beta^2 y_{n-2} \\
&= x_n + \beta x_{n-1} + \beta^2(x_{n-2} + \beta y_{n-3}) = x_n + \beta x_{n-1} + \beta^2 x_{n-2} + \beta^3 y_{n-3} \\
&= \cdots \\
&= x_n + \beta x_{n-1} + \beta^2 x_{n-2} + \beta^3 x_{n-3} + \beta^4 x_{n-4} + \cdots \\
&= x_n + \sum_{m=1}^{\infty} \beta^m x_{n-m} \;=\; \sum_{m=0}^{\infty} \beta^m x_{n-m}
\end{aligned}
$$

In general AR filters can be written as *infinite* convolutions

$$y_n = \sum_{l=0}^{\infty} h_l \, x_{n-l}$$

Try this for the AR with 2 time delays

AR is a filter

The recursive AR form is an AR (autoregressive) *filter*

LINEARITY

Start from the infinite convolution form
the proof is the same as for the MA filter

TIME INVARIANCE

Start from the infinite convolution form
the proof is the same as for the MA filter
as long as the coefficients are not time dependent

Frequency response of an AR filter

Let's try our simple AR example $y_n = (1-\beta)\, x_n + \beta\, y_{n-1}$

What happens for DC?

we purposely put the input gain back in!

We know that for all n $x_n = 1$ and $y_n = H_0$

so $H_0 = (1-\beta) + \beta H_0$ or $H_0 (1 - \beta) = (1-\beta)$ so $H_0 = 1$

What happens for Nyquist ?

We know that $x_n = \ldots -1 +1 -1 +1 \ldots$ and $y_n = \ldots -H_\pi + H_\pi - H_\pi + H_\pi \ldots$

so $H_\pi = (1-\beta) - \beta H_\pi$ or $H_\pi (1+\beta) = (1-\beta)$ so $H_\pi = (1-\beta) / (1+\beta)$

For general frequency ω : $x_n = e^{i\omega n}$ and $y_n = H(\omega) e^{i\omega n}$

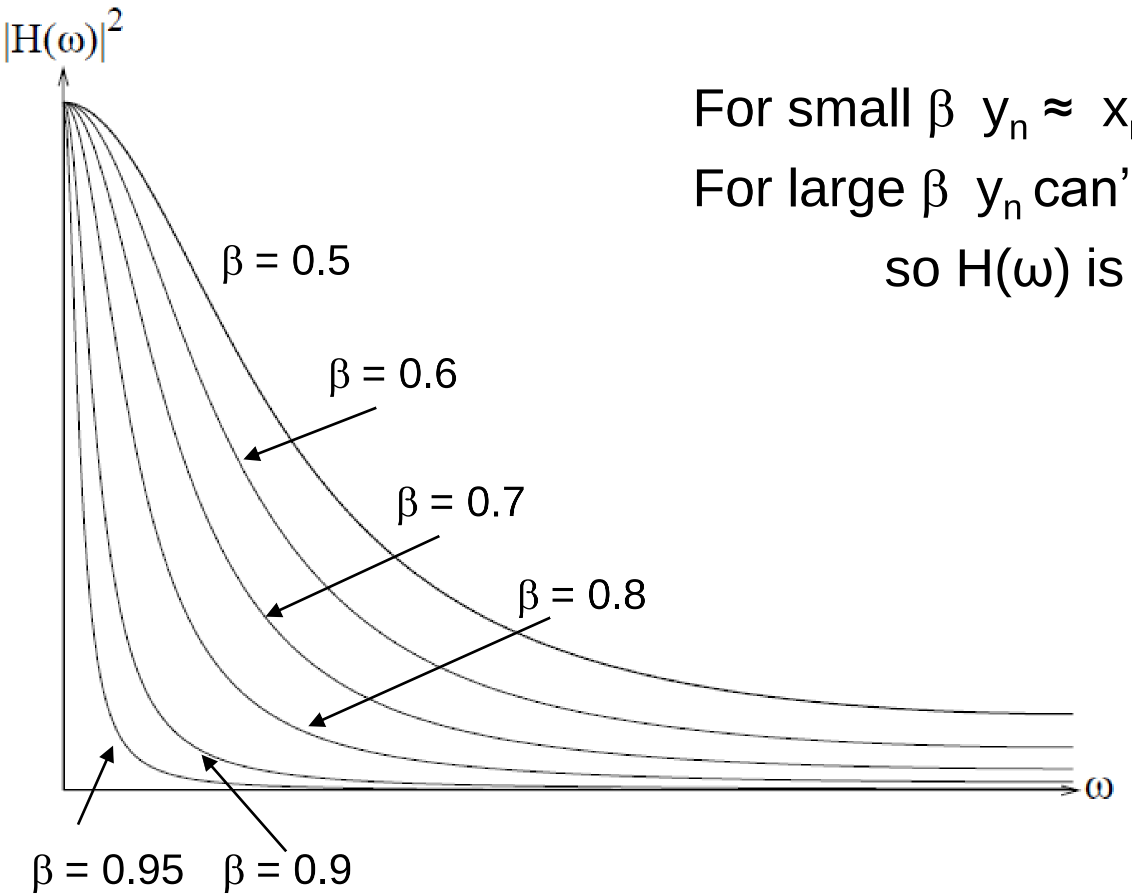
so $H(\omega) e^{i\omega n} = (1-\beta) e^{i\omega n} + \beta H(\omega) e^{i\omega(n-1)}$ so $H(\omega) = (1-\beta) + \beta H(\omega)\, e^{-i\omega}$

so $H(\omega) = (1-\beta) / (1 - \beta\, e^{-i\omega})$ Why is this complex (i.e., has a phase shift)?

and $|H(\omega)|^2 = 1 / (1 - 2\beta \cos(\omega) + \beta^2)$

The AR frequency response

$y_n = (1- \beta)\, x_n + \beta\, y_{n-1}$



For small β $y_n \approx x_n$ so $H(\omega) \approx 1$

For large β y_n can't keep up with x

so $H(\omega)$ is very low-pass

The harder way

But we cheated! We haven't yet proven the **filter law**

We can find the frequency response of the AR filter
from the *unraveled form,* but without using the filter law

$$
\begin{aligned}
y_n &= (1-\beta)e^{i\omega n} + \beta(1-\beta)e^{\mathrm{i}\omega(n-1)} + \beta^2(1-\beta)e^{\mathrm{i}\omega(n-2)} + \ldots \\
&= (1-\beta)\sum_{k=0}^{\infty}(\beta e^{-\mathrm{i}\omega})^k e^{\mathrm{i}\omega n} \\
&= \frac{(1-\beta)}{(1-\beta e^{-\mathrm{i}\omega})} e^{\mathrm{i}\omega n}
\end{aligned}
$$

(we used the formula for the sum of an infinite series

$\sum_{k=0}^{\infty} q^k = \frac{1}{1-q}$ with $q = \beta\, e^{-\mathrm{i}\omega}$)

The accumulator

We once defined the accumulator y = $\hat{\Upsilon}$ x

by $y_n = \sum_{m=0}^{\infty} x_{n-m}$

(the inverse of the first finite difference - $\hat{\Upsilon}\,\hat{\Delta} = \hat{\Delta}\hat{\Upsilon} = 1$)

We can write the accumulator as an AR filter

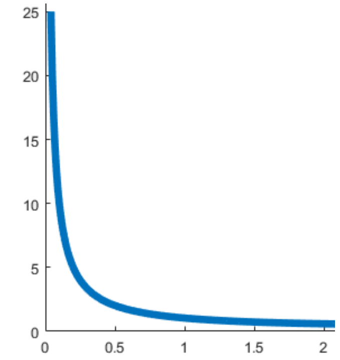
$$y_n = x_n + y_{n-1}$$

If we input DC this explodes! (AR filters can be *unstable*)

What is the frequency response?

$$H(\omega)e^{\mathrm{i}\omega n} = e^{i\omega n} + H(\omega)e^{\mathrm{i}\omega(n-1)}$$

Thus $$H(\omega) = \frac{1}{1-e^{-\mathrm{i}\omega}} = -\mathrm{i}e^{\mathrm{i}\omega/2}\frac{1}{2\sin(\omega/2)}$$



So |H(ω)| is very similar to the FR of the true integrator!

MA, AR and ARMA

The general causal system looks like this:

$$y_n = f\,(\,x_n\,,\,x_{n-1}\,\ldots\,x_{n-l}\,;\,y_{n-1}\,,\,y_{n-2}\,,\,\ldots y_{n-m}\,\;;\,n\,)$$

But the general ***causal filter*** has to be

a linear combination of the inputs and outputs

$$y_n = \sum_{n=0}^{L-1} a_l x_{n-l} + \sum_{m=1}^{M} b_m y_{n-m}$$

This is called ARMA (it would be hard to say *MAAR*)

if b_m=0 then it is MA

if a_0=0 and $a_{\ell>0}$=0 but $b_m \neq 0$ then it is AR

Why doesn't the ARMA filter depend explicitly on n ?

Why does the sum only include previous inputs and outputs?

Why must the function be a linear combination of them ?

Why does m start at 1 and not 0 ?

Symmetric form of writing ARMA

We can write the ARMA equation in symmetric form by terms moving from side to side

$$
\begin{aligned}
y_n &= \sum_{l=0}^{L-1} \alpha_l x_{n-l} + \sum_{m=1}^{M} \beta_m y_{n-m} \\
y_n - \sum_{m=1}^{M} \beta_m y_{n-m} &= \sum_{l=0}^{L-1} \alpha^l x_{n-l}
\end{aligned}
$$

where

$$\forall l \;\; \alpha_l = a_l \qquad \beta_0 = 1 \qquad \forall m > 0 \;\; \beta_m = -b_m$$

This form is a called a *difference equation* since it can be rewritten as $\sum B_m \hat{\Delta}^m y = \sum A_l \hat{\Delta}^l x$

What is the connection between the coefficients?

3 ways of writing the ARMA filter

So far we can write the causal ARMA filter in 3 ways

ARMA form

$$y_n = \sum_{l=0}^{L-1} a_l x_{n-l} + \sum_{m=1}^{M} b_m y_{n-m}$$

Symmetric form
(difference equation)

it looks *nicer* with L

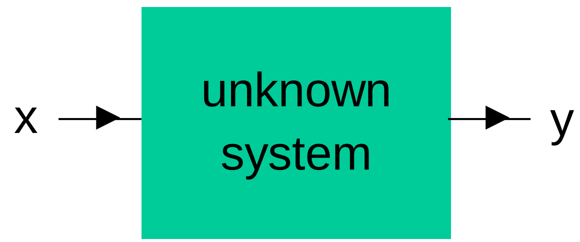
$$\sum_{m=0}^{M} \beta_m y_{n-m} = \sum_{l=0}^{L} \alpha_l x_{n-l}$$

Infinite convolution

$$y_n = \sum_{l=0}^{\infty} h_l x_{n-l}$$

What happens when the filter is MA? AR?
How can we translate between representations?

System identification



Up to now we have discussed
what a known ARMA system does to a given input

Now let's consider the converse problem

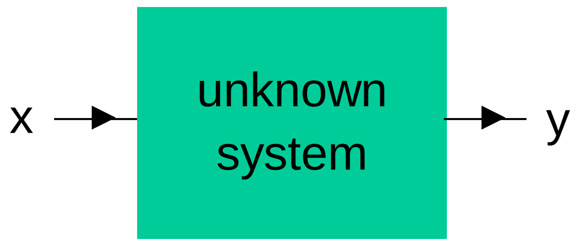
We are given an *unknown* system with one input and one output
think of the system as inside a black box which can't be opened

What is known are the input and output to the black box

Can we figure out what is inside the box ?

This is called the *system identification problem*

Identification?



What do we mean by identifying the system ?

You are given the unknown system for some amount of time

You need to be able to predict the output for *any* given input

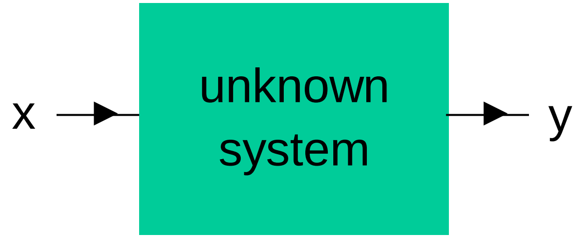
For ARMA systems, it is enough to know any of these:

- ARMA form – L **a** coefficients and M **b** coefficients
- symmetric form (difference equation) L **α** coefficients, M **β** coefficients
- infinite convolution form all h_l
- the frequency response all $H_{\mathbf{k}}$

since from any of these we can calculate the output y for all times

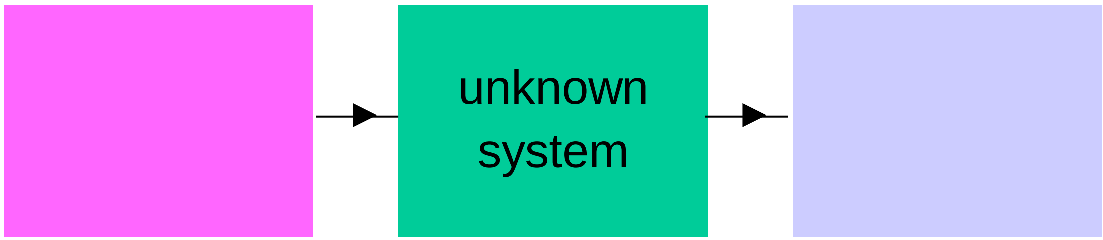
Two flavors

There are two different ways this *game* can be played



Easy system identification problem

- we are allowed to input any **x** we want and observe the output **y**
- what input should we use?



Hard system identification problem

- the system is already "hooked up"

    we can only *observe* the input **x** and output **y**

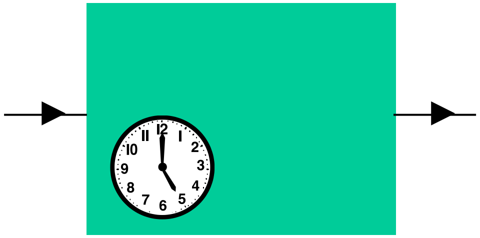
The *hard* problem is indeed harder than the easy problem

for example - what happens if the input is always 0?

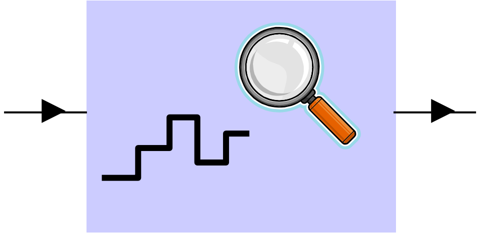
Filter identification

Is the system identification problem always *solvable* ?

Not if the system characteristics can change over time
Since you can't predict what it will do next
So only solvable if system is **time invariant**



Not if system can have a hidden *trigger* signal
So only solvable if system is **linear**
Since for linear systems



- any signal is the sum of the trigger plus the difference
- small changes in input lead to bounded changes in output

So only solvable if system is a **filter !**

Easy problem
Impulse Response (IR)

To solve the easy problem (where we can input any signal(s) we want)
we need to decide which input signal x to use

One common choice is the *unit impulse*
the signal that is zero everywhere except at time zero n=0

The response of the filter to an impulse at time zero (UI)
is called the **impulse response** IR (not a surprising name !)
תגובה להלם

The impulse response of a filter is universally called h_n

What can we say about the impulse response for a causal system?

Some impulse responses

What is the impulse response for an **MA** filter?

$$\mathbf{h_n} = \sum_{l=0}^{L-1} a_l \, \delta_{n-l,0} = a_n$$

So, the MA coefficients are exactly the impulse response

What is the impulse response for an **ARMA** filter ?

Use the infinite convolution form!

$$\mathbf{h_n} = \sum_{l=0}^{\infty} h_l \, \delta_{n-l,0} = h_n$$

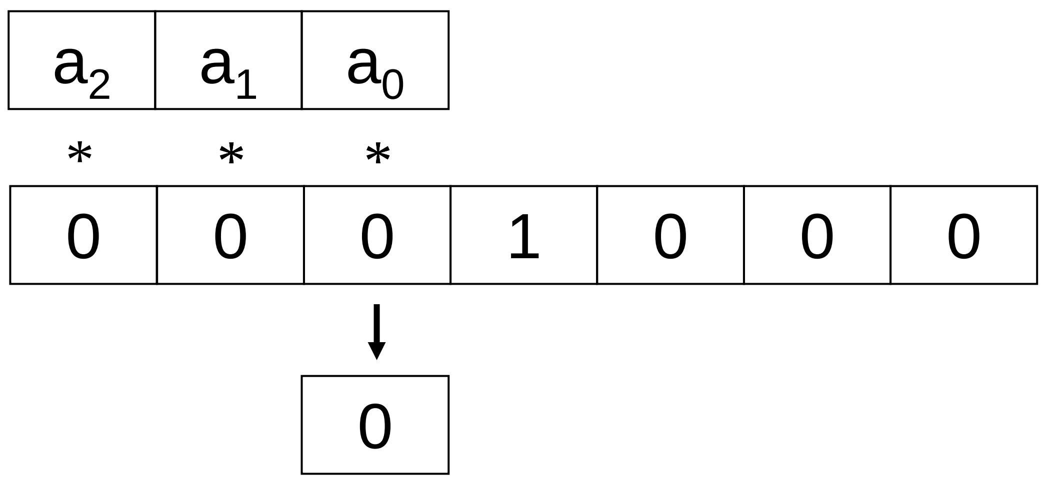
which is why we called these coefficients h in the first place!

The IR of an MA filter is nonzero for a finite number (L) of times
and so MA filters are called **F**inite **I**mpulse **R**esponse filters

The IR of AR or general ARMA filters
is nonzero for an infinite number of times (due to the recursion!)
and so they are called **I**nfinite **I**mpulse **R**esponse filters

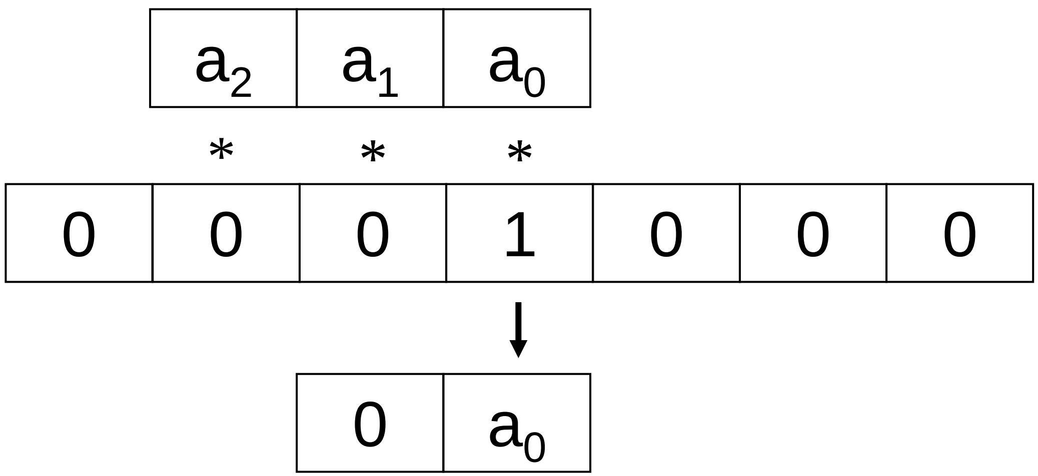
IR for MA filter

We can see why MA filters are FIR
by the following graphical construction



IR for MA filter

We can see why MA filters are FIR
by the following graphical construction



IR for MA filter

We can see why MA filters are FIR
by the following graphical construction

a_2	a_1	a_0

$*$ $*$ $*$

0	0	0	1	0	0	0

$\downarrow$

0	a_0	a_1

IR for MA filter

We can see why MA filters are FIR
by the following graphical construction

a_2	a_1	a_0

$*$ $*$ $*$

0	0	0	1	0	0	0

$\downarrow$

0	a_0	a_1	a_2

IR for MA filter

We can see why MA filters are FIR
by the following graphical construction

				a_2	a_1	a_0
				*	*	*
0	0	0	1	0	0	0

↓

0	a_0	a_1	a_2	0

You see why we chose this direction?
convolution, not **correlation**!

IR solves the easy SI problem!

It is enough to input one simple signal to know the system !

- if we know the response of a filter to the UI
  we know its response to any SUI
  because of time invariance (just shift the impulse!)

- if we know the response of a filter to all SUIs
  we know its response to any weighted combination of SUIs
  because of linearity (add the weighted outputs!)

- any input signal **x**
  can be written as the weighted combination of SUIs
  since SUIs are a basis

Easy problem
Frequency Response (FR)

We have found one solution to the easy SI problem

Another common choice of input are the *sinusoids*

$$x_n = \sin(k\,n)$$

But we need to enter all possible sinusoids (k=0, 1, …)

However, from the filter law we know that

sinusoids are *eigensignals* of filters

the response to a sinusoid of frequency ω : $\sin(\omega\, n)$

is a sinusoid of frequency ω (or zero output)

$$y_n = A_\omega \sin(\omega\, n + \phi_\omega)$$

So we input all possible sinusoids

but record only the **frequency response** FR

- the gain A_k
- the phase shift ϕ_k

k	A_k	ϕ_k
0		
1		
2		

FR solves the easy SI problem!

It is enough to input these sinusoids to know the system !

- if we know the response of a filter to $x_n = \sin(k\,n)$
  we know its response to $x_n = \sin(k\,n + \phi)$
  because of time invariance

- if we know the response of a filter to arbitrary sinusoids
  we know its response to weighted combination of them
  because of linearity (add the weighted outputs!)

- any input signal **x**
  can be written as the weighted combination of sinusoids
  since they are the Fourier basis

Does this make sense?

In the first solution we only needed one trial

we entered one input the UI

and recorded the impulse response h_n

In the second solution we had to enter many inputs – all the sinusoids!

Does this make sense?

For the impulse response we needed to record many time values

for the frequency response

we only needed one complex number for each input

For example, assume a signal with N values (in time/frequency domain)
and an MA with N coefficients

- for h_n we need to record N values
- for H_k we need to record N coefficients : 1 for each frequency

Why do you think we call the IR **h** and the FR **H** ?

The easiest hard problem

The hard problem is so hard
that we will start with a simple case

- Assume an MA filter with 3 coefficients

$$\mathbf{y_n} = \sum_{l=0}^{L=2} a_l\, x_{n-l} = a_0\, x_n + a_1\, x_{n-1} + a_2\, x_{n-2}$$

- We further assume that the input was zero until time n=0
  (we can always take the time the signal starts to be n=0 …)

  so $x_{n<0} = 0$

We need to find 3 unknowns – a_0, a_1, and a_2
so we will need three equations to solve

The easiest hard problem (cont.)

First let's write the equation for n=0

$$y_0 = a_0\, x_0 + a_1\, x_{-1} + a_2\, x_{-2} = a_0\, x_0$$

Since $x_0 \neq 0$ we can divide to find $a_0 = y_0 \,/\, x_0$

Next we write the equation for n=1

$$y_1 = a_0\, x_1 + a_1\, x_0 + a_2\, x_{-1} = a_0\, x_1 + a_1\, x_0$$

What do we already know?

$$y_1 = a_0\, x_1 + a_1\, x_0 \quad \text{so} \quad a_1 = (y_1 - a_0\, x_1) \,/\, x_0$$

which is OK since $x_0 \neq 0$

Finally we write the equation for n=2

$$y_2 = a_0\, x_2 + a_1\, x_1 + a_2\, x_0$$

so $a_2 = (y_2 - a_0\, x_2 - a_1\, x_1) \,/\, x_0$ which is OK since $x_0 \neq 0$

The easiest hard problem – matrix form

First can rewrite the three equations

$$\mathbf{y_0} = a_0\, x_0$$
$$\mathbf{y_1} = a_0\, x_1 + a_1\, x_0$$
$$\mathbf{y_2} = a_0\, x_2 + a_1\, x_1 + a_2\, x_0$$

in matrix format (with the coefficient as the vector)

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_0 & 0 & 0 \\ x_1 & x_0 & 0 \\ x_2 & x_1 & x_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$$

which can be solved by inverting the matrix

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} x_0 & 0 & 0 \\ x_1 & x_0 & 0 \\ x_2 & x_1 & x_0 \end{pmatrix}^{-1} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

The easiest hard problem – some more

The matrix to invert $\begin{pmatrix} x_0 & 0 & 0 \\ x_1 & x_0 & 0 \\ x_2 & x_1 & x_0 \end{pmatrix}$ is lower triangular

which is why it was so easy to solve

In fact, our solution was the straightforward inversion!

But this matrix has another characteristic

it has Toeplitz (Töplitz) form

– the same value along diagonals

$$\begin{pmatrix} x_0 & 0 & 0 \\ x_1 & x_0 & 0 \\ x_2 & x_1 & x_0 \end{pmatrix}$$

A slightly harder problem

- Assume an MA filter with 3 coefficients

$$\mathbf{y_n} = \sum_{l=0}^{L=2} a_l\, x_{n-l} = a_0\, x_n + a_1\, x_{n-1} + a_2\, x_{n-2}$$

- but the input does not start nonzero

We still need to find the 3 unknowns – a_0, a_1, and a_2
so we will need three equations to solve

So pick any n (there is nothing special about any time)
and write three consecutive equations

$$\begin{aligned}
\mathbf{y_n} &= a_0\, x_n + a_1\, x_{n-1} + a_2\, x_{n-2} \\
\mathbf{y_{n+1}} &= a_0\, x_{n+1} + a_1\, x_n + a_2\, x_{n-1} \\
\mathbf{y_{n+2}} &= a_0\, x_{n+2} + a_1\, x_{n+1} + a_2\, x_n
\end{aligned}$$

Note that we need to observe 5 consecutive times

n-2 x_{n-2} **n-1** x_{n-1} **n** x_n and $\mathbf{y_n}$

n+1 x_{n+1} and $\mathbf{y_{n+1}}$ **n+2** x_{n+2} and $\mathbf{y_{n+1}}$

Solving the slightly harder problem

Let's jump directly to the matrix form

$$\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \end{pmatrix} = \begin{pmatrix} x_n & x_{n-1} & x_{n-2} \\ x_{n+1} & x_n & x_{n-1} \\ x_{n+2} & x_{n+1} & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$$

So, here is yet another connection
    between convolution and (matrix) multiplication

The solution is once again to invert the matrix
    but this time it is not lower triangular
        but it is still Toeplitz

Inverting a general matrix is $O(N^3)$ and may be unstable
    (well actually $O(n^{\log 2(7)}) = O(n^{2.807})$ or even less)
but inverting a Toeplitz matrix takes only $O(N^2)$
    and is always stable (Levinson Durbin algorithm)

BTW – another connection

We wanted to solve for the coefficients
and thus put them into a vector

In other circumstances we may want to rewrite the equations

$$
\begin{aligned}
y_n &= a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} \\
y_{n+1} &= a_0 x_{n+1} + a_1 x_n + a_2 x_{n-1} \\
y_{n+2} &= a_0 x_{n+2} + a_1 x_{n+1} + a_2 x_n
\end{aligned}
$$

in another form

$$
\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \end{pmatrix} = \begin{pmatrix} a_2 & a_1 & a_0 & 0 & 0 \\ 0 & a_2 & a_1 & a_0 & 0 \\ 0 & 0 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} x_{n-2} \\ x_{n-1} \\ x_n \\ x_{n+1} \\ x_{n+2} \end{pmatrix}
$$

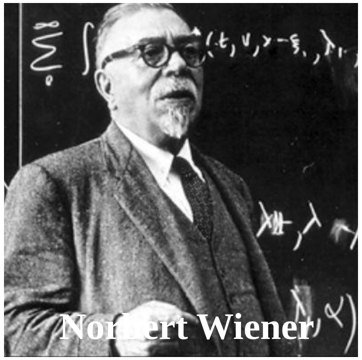
Here the matrix is Toeplitz but not square!

This is yet another connection between convolution
and multiplication by a Toeplitz matrix!

Wiener-Hopf equations

The equations we found

$$\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \end{pmatrix} = \begin{pmatrix} x_n & x_{n-1} & x_{n-2} \\ x_{n+1} & x_n & x_{n-1} \\ x_{n+2} & x_{n+1} & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$$



are called the **Wiener-Hopf equations**

However, you will never see them written in this simple way!

That is because of noise!

If we solve them twice for different **n**
we won't get exactly the same answer

So you might solve many times and average the solutions
but that would require many matrix inversions
and is not even the right thing to do!

What's the right way?

Let's go back to the original MA equation

$$\mathbf{y_n} = \sum_{l=0}^{L} \boldsymbol{a}_l \, \boldsymbol{x}_{n-l}$$

Multiple both sides by $\mathrm{x}_{\mathbf{n+j}}$ and sum over all n

$$\sum_n \boldsymbol{y}_n \, \boldsymbol{x}_{n+j} = \sum_n \sum_l \boldsymbol{a}_l \, \boldsymbol{x}_{n-l} \, \boldsymbol{x}_{n+j}$$

we can reverse the summation order!

Remember that we mentioned the *correlation of x and y ?*

$$C_{xy}(j) = \sum_n \boldsymbol{x}_n \, \boldsymbol{y}_{n+j}$$

So the Wiener-Hopf equations can be written:

$$C_{yx}(j) = \sum_l \boldsymbol{a}_l \, C_{xx}(j-l)$$

This has the same form, but need be solved only once!

What about AR filters?

Now let's assume that the unknown system
is an **AR** filter with 3 coefficients

$\mathbf{y_n = x_n} + \sum_{m=1}^{M=3} b_m\, y_{n-m} = x_n + b_1\, y_{n-1} + b_2\, y_{n-2} + b_3\, y_{n-3}$

Once again we have three coefficients to find
so we need to write 3 equations

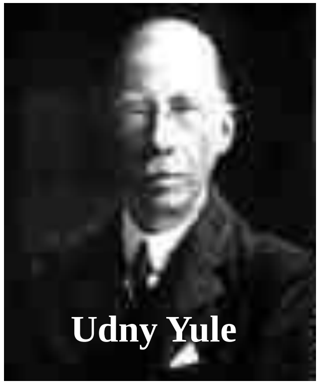
$$\begin{aligned} y_n &= x_n + b_1 y_{n-1} + b_2 y_{n-2} + b_3 y_{n-3} \\ y_{n+1} &= x_{n+1} + b_1 y_n + b_2 y_{n-1} + b_3 y_{n-2} \\ y_{n+2} &= x_{n+2} + b_1 y_{n+1} + b_2 y_n + b_3 y_{n-1} \end{aligned}$$

Note that we need to observe 6 times - 6 **y**s and 3 **x**s

Yule-Walker equations

Let's write the equations in matrix form

$$\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \end{pmatrix} = \begin{pmatrix} x_n \\ x_{n+1} \\ x_{n+2} \end{pmatrix} + \begin{pmatrix} y_{n-1} & y_{n-2} & y_{n-3} \\ y_n & y_{n-1} & y_{n-2} \\ y_{n+1} & y_n & y_{n-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

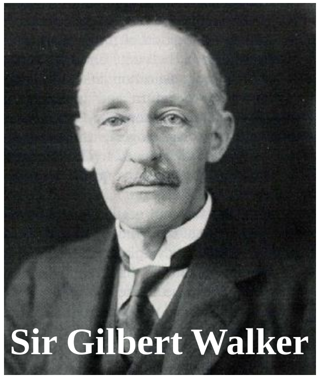


vector $\underline{y} = \underline{x} + \underline{\underline{y}}\,\underline{b}$ matrix so $\underline{b} = \underline{\underline{y}}^{-1}\,\underline{(y - x)}$

These are called the **Yule Walker equations**

The matrix has Toeplitz form

and so is solved by Levinson-Durbin



Your cellphone solves YW equations

thousands of times per second !

What is the right way?

However, you will never see the Yule Walker equations written in this simple way because of noise

Instead take the original AR equation (without the x)

$$\mathbf{y_n} = \sum_{m=1}^{M} b_m\, y_{n-m}$$

Multiply both sides by y_{n+j} and sum over all n

$$\sum_n y_{n+j}\, y_n = \sum_n \sum_m b_m\, y_{n-m}\, y_{n+j}$$

which can be written

we can reverse the summation order!

$$C_{yy}(j) = \sum_m b_m\, C_{yy}(j-m)$$

What about (full) ARMA filters?

We can repeat the entire exercise for general ARMA filters

$$y_n = \sum_{n=0}^{L-1} a_l x_{n-l} + \sum_{m=1}^{M} b_m y_{n-m}$$

We have L+M variables and so have to write L+M equations

But the matrix will not turn out to be Toeplitz
and thus the equations will be difficult to solve!

So, in DSP we try to make every system identification problem
either MA or AR !

Another way to solve

So far we have worked in the time domain

Why can't we use the filter law directly?
Since **$Y_k = H_k X_k$** we can divide to find **$H_k = Y_k / X_k$**
and knowing H_k determines the filter!

The problem is that X_k can be zero!

So, instead we will use the z transform
and at last *prove* the filter law!

We will start with
the *infinite convolution* form of the ARMA filter

$$y_n = \sum_{k=-\infty}^{\infty} h_k x_{n-k}$$

Using z transform

the same tricks we've done before!

$$
\begin{aligned}
Y(z) &= \sum_{n=-\infty}^{\infty} y_n z^{-n} \\
&= \sum_{n=-\infty}^{\infty} \left\{ \sum_{k=-\infty}^{\infty} h_k x_{n-k} \right\} z^{-n} \\
&= \sum_{k=-\infty}^{\infty} h_k \sum_{n=-\infty}^{\infty} x_{n-k} z^{-n} \\
&= \sum_{k=-\infty}^{\infty} h_k z^{-k} \sum_{m=-\infty}^{\infty} x_m z^{-m} \\
&= \quad H(z) \qquad\qquad X(z)
\end{aligned}
$$

The transfer function

So we have found that Y(z) = H(z) X(z)

H(z) is called the **transfer function**

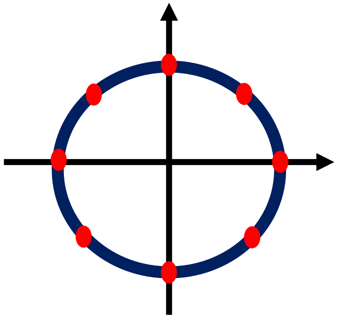
We defined $\mathrm{H(z)} = \sum_{n=-\infty}^{\infty} h_n \, z^{-n}$

which means that H(z) is the zT of the *impulse response*

In particular, if we look only on the unit circle

we find Y(ω) = H(ω) X(ω)

and on the *digital* points $Y_{\mathbf{k}} = H_{\mathbf{k}} X_{\mathbf{k}}$



Which is precisely the **filter law**

Furthermore, we see that the *frequency response* $H_{\mathbf{k}}$

is the FT of the *impulse response* $h_{\mathbf{n}}$

This explains why we called them **h** and **H** !

Let's do that again!

To find out even more We will do the same kind of calculation but this time start with the symmetric form of the ARMA filter

$$\sum_{m=0}^{M} \beta_m y_{n-m} = \sum_{l=0}^{L} \alpha_l x_{n-l}$$

remember $\beta_0 = 1$

Now we take the zT of both sides

$$\sum_{n=-\infty}^{\infty} \left(\sum_{m=0}^{M} \beta_m y_{n-m} \right) z^{-n} = \sum_{n=-\infty}^{\infty} \left(\sum_{l=0}^{L} \alpha_l x_{n-l} \right) z^{-n}$$

and do our usual tricks to get

$$\sum_{m=0}^{M} \beta_m z^{-m} \sum_{n=-\infty}^{\infty} y_n z^{-n} = \sum_{l=0}^{L} \alpha_l z^{-l} \sum_{n=-\infty}^{\infty} x_n z^{-n}$$

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

The entire calculation

$$
\begin{aligned}
\sum_{n=-\infty}^{\infty}\left(\sum_{m=0}^{M}\beta_m y_{n-m}\right)z^{-n} &= \sum_{n=-\infty}^{\infty}\left(\sum_{l=0}^{L}\alpha_l x_{n-l}\right)z^{-n}\\
\sum_{n=-\infty}^{\infty}\sum_{m=0}^{M}\beta_m y_{n-m} z^{-n} &= \sum_{n=-\infty}^{\infty}\sum_{l=0}^{L}\alpha_l x_{n-l} z^{-n}\\
\sum_{n=-\infty}^{\infty}\sum_{m=0}^{M}\beta_m y_{n-m} z^{-n} &= \sum_{n=-\infty}^{\infty}\sum_{l=0}^{L}\alpha_l x_{n-l} z^{-n}\\
\sum_{m=0}^{M}\beta_m \sum_{n=-\infty}^{\infty} y_{n-m} z^{-n} &= \sum_{l=0}^{L}\alpha_l \sum_{n=-\infty}^{\infty} x_{n-l} z^{-n}\\
\sum_{m=0}^{M}\beta_m z^{-m} \sum_{n=-\infty}^{\infty} y_{n-m} z^{-(n-m)} &= \sum_{l=0}^{L}\alpha_l z^{-l} \sum_{n=-\infty}^{\infty} x_{n-l} z^{-(n-l)}\\
\sum_{m=0}^{M}\beta_m z^{-m} \sum_{n=-\infty}^{\infty} y_n z^{-n} &= \sum_{l=0}^{L}\alpha_l z^{-l} \sum_{n=-\infty}^{\infty} x_n z^{-n}\\
B(z) \qquad Y(z) &= A(z) \qquad X(z)
\end{aligned}
$$

H(z) is a rational function

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

so $Y(z) = A(z) / B(z)\ X(z)$

but we know $Y(z) = H(z)\,X(z)$

so $H(z) = A(z) / B(z)$

A(z) and B(z) are polynomials in z^{-1}

By multiplying by z^L and z^M respectively
we can make them into polynomials in z

This means that the transfer function is a **rational function**
that is, the ratio of two polynomials in z

In complex function theory

- the *roots* of the *numerator* are called ***zeros*** of H(z)
- the *roots* of the *denominator* are called ***poles*** of H(z)

Poles and zeros

From the **fundamental theory of algebra** we know that
every polynomial of degree n has n roots over the complex numbers

Hence we can write

$$
\begin{aligned}
A(z) &= \sum_{l=0}^{L} \alpha_l z^l = \prod_{l=0}^{L} (z - \zeta_l) \\
B(z) &= \sum_{m=0}^{M} \beta_m z^m = \prod_{m=0}^{M} (z - \pi_m)
\end{aligned}
$$

where we see the *zeros* and *poles* of the transfer function

An important theorem in complex functions states
that the zeros and poles determine a rational function
to within a multiplicative constant

So the poles and zeros of the transfer function determine the filter
to within an overall gain

In diagrams zeros are shown as ● and poles as **x**

Special cases

If the ARMA form is actually an MA filter
then there are α coefficients
but all the β are zero except $\beta_0 = 1$

So $H(z) = A(z) / B(z) = A(z)$ has **zeros** but no poles!

If the ARMA form is actually an AR filter
then there are β coefficients
but all the α are zero except $\alpha_0 = 1$

So $H(z) = A(z) / B(z) = 1/B(z)$ has **poles** but no zeros

If the ARMA filter is *general* (not MA or AR)
then it has both poles and zeros

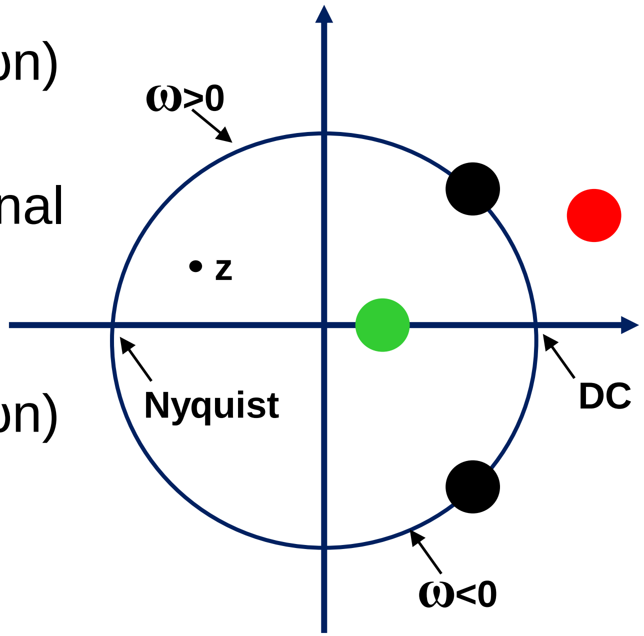
Summary of filter names

FIR	MA	all zeros
IIR	AR	all poles
	ARMA	zeros and poles

What do zeros/poles mean?

Since $Y(z) = H(z)\, X(z)$, if the input is $x_n = z^n$ the output is $y_n = H(z)\, z^n$

- A zero at z means that if the input is $x_n = z^n$ the output is zero!
  - A zero **on** the unit circle
    means an input sinusoid $x_n = \sin(\omega n)$
    for which the output is zero
- A pole means that if the input is *that* signal
  the output explodes!
  - A pole **on** the unit circle
    means an input sinusoid $x_n = \sin(\omega n)$
    for which the output explodes



Why do zeros and poles not on the real axis come in pairs?

Why don't we allow poles on or outside the unit circle
while zeros can be anywhere?

Are zeros important?

The filter law $Y(\omega) = H(\omega)\, X(\omega)$ tells us that

no new frequencies are created

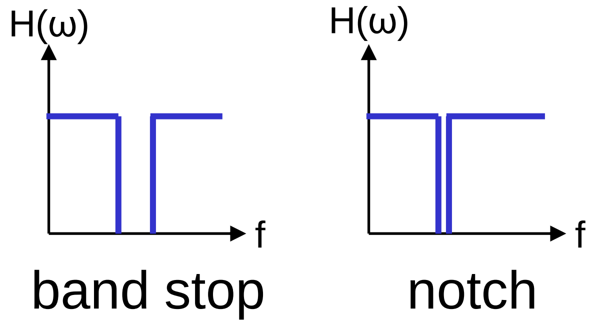
but frequencies can disappear (when $H(\omega)=0$) !

We call a frequency that disappears a **zero** of the filter

and more generally a signal z^n that disappears

We already saw examples of MA filters with zeros!

- $y_n = x_n - x_{n-1}$ (first finite difference) has a zero at DC
- $y_n = x_n + x_{n-1}$ has a zero at Nyquist
- $y_n = x_n + x_{n-2}$ has a zero at half Nyquist ($\omega = \pi/2$)
- Bandstop and notch filters are used because of their zeros



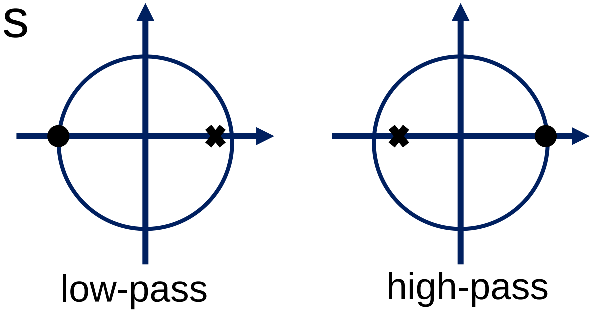
Are poles important?

The filter law in the z plane $Y(z) = H(z) X(z)$ tells us that
no new z^n signals are created
but these signals can explode (when $H(\omega)=\infty$) !

We call a z^n signal that explodes a **pole** of the filter

We already saw examples of AR filters with poles!

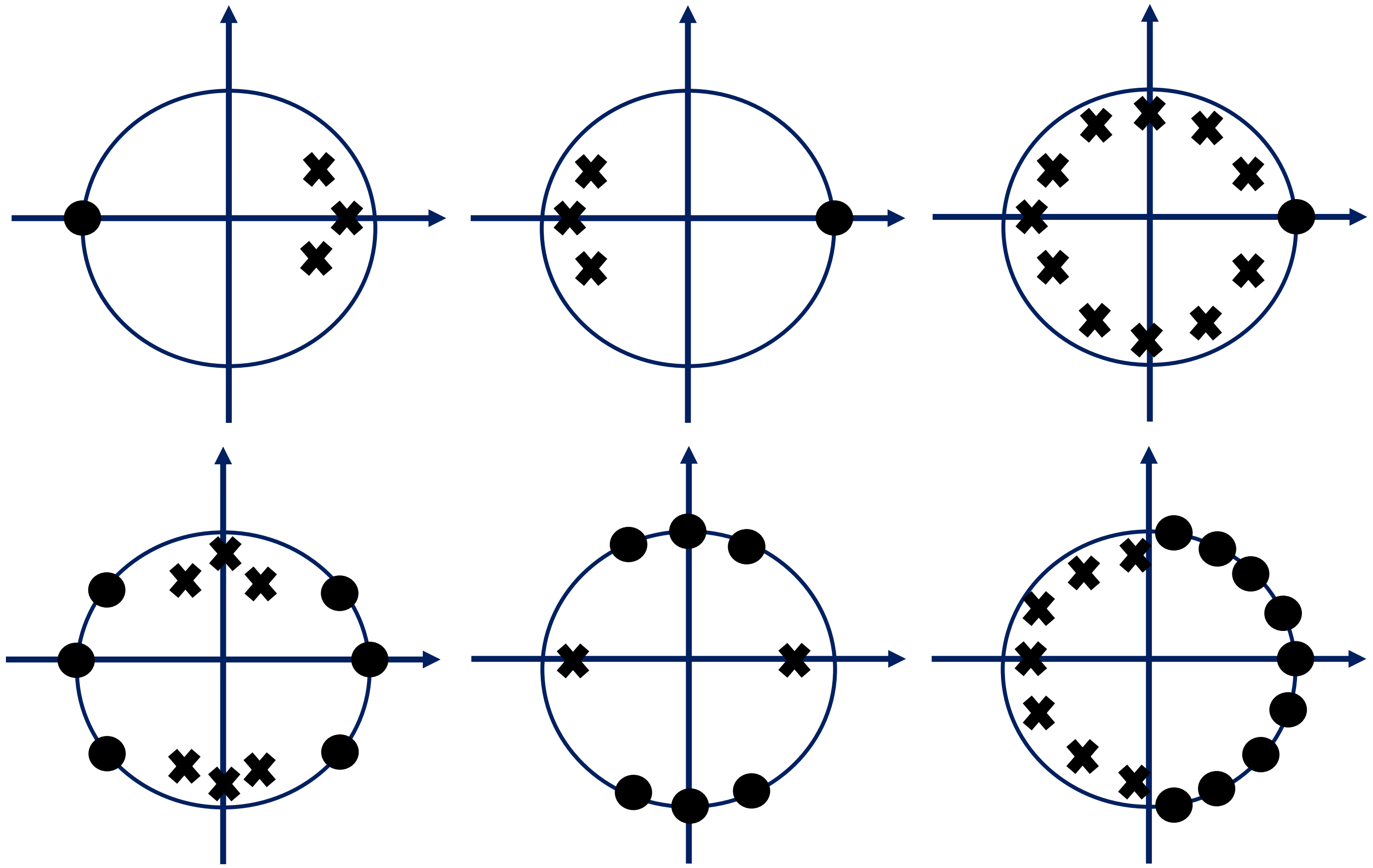
- $y_n = x_n + y_{n-1}$ (accumulator) has a pole at DC
- $y_n = x_n - y_{n-1}$ has a pole at Nyquist
- $y_n = x_n - y_{n-2}$ has a pole at half Nyquist ($\omega = \pi/2$)
- we don't want poles on the unit circle (for sinusoids!)
  but sinusoids that are amplified
  require nearby poles



Designing filters by poles and zeros

DSP experts sometimes design filters directly using poles and zeros!

What do the following pole/zero diagrams mean?



How to find the transfer function?

If you are given the filter in the time domain

it is easy to find its *transfer function* and *poles/zeros*

by the following steps:

1. Move all the **y**s to one side and **x**s to the other
   to create the symmetric form
2. Write the equation in terms of signals using delay operators
3. Take the zT of both sides using our rule $zT(\hat{\mathbf{z}}^{-1} x) = z^{-1}\, zT(x)$
4. Divide leaving Y(z) on the LHS
5. Change numerator and denominator into polynomials (discard z^M)
6. Find the roots of the polynomials

Summary - filters

FIR = MA = all zero
IIR: AR = all pole
ARMA = zeros and poles

The following contain everything about the filter
(are can predict the output given the input)

- **a** and **b** coefficients
- α and β coefficients
- impulse response $\mathbf{h_n}$
- frequency response $\mathbf{H(\omega)}$
- transfer function **H(z)**
- pole-zero diagram + overall gain

How do we convert between them ?

Conversions

to → from ↓	a, b coefficients	α,β coefficients	impulse response	frequency response	transfer function	gain and pole-zero diagram
a, b coefficients	identity	subtraction of y terms	MA: h=a AR + ARMA: recursion	substitute $x=e^{ikn}$	write using z^{-1} and extract	through transfer function
α,β coefficients	addition of y terms	identity	same as a,b	same as a,b	same as a,b	same as a,b
impulse response	MA: a=h ARMA: recursion	through a,b	identity	DFT	zT	through transfer function
frequency response	through IR or transfer function	same as a,b	iDFT	identity	analytic continuation	through transfer function
transfer function	through α,β	$B(z)\,Y(z) = A(z)\,X(z)$	izT	substitute $z = e^{i\omega}$	identity	find roots
gain and pole-zero diagram	through transfer function	through transfer function	through transfer function	substitution	multiply terms to get polynomial	identity

Exercise - causal MA

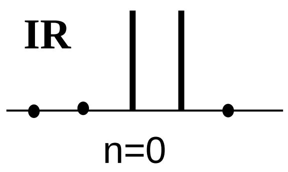
$y_n = x_n + x_{n-1}$

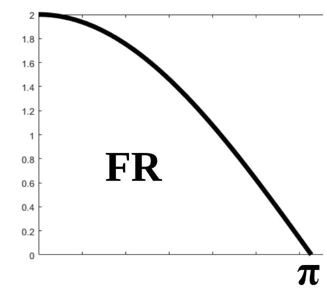
- this filter is causal MA
- we can immediately guess that it is low-pass since
  - it averages!
  - H(DC) = 2 (from H = 1 + 1) – gain=2!
  - H(Nyquist) = 0 (from H = 1 +1)
- impulse response is 1, 1
- frequency response: $H(\omega)e^{i\omega n} = e^{i\omega n} + e^{i\omega(n-1)}$

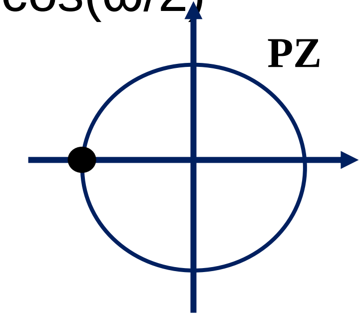
  so $H(\omega) = 1 + e^{-i\omega} = e^{-i\omega/2} (e^{+i\omega/2} + e^{-i\omega/2}) =$ phase*$2\cos(\omega/2)$

  causality results in phase!
- transfer function

  $y = (1 + \hat{z}^{-1})\, x$ so $H(z) = 1 + 1/z \rightarrow z+1$







Try $y_n = x_n - x_{n-1}$

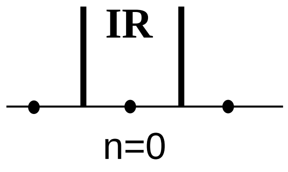
Exercise -noncausal MA

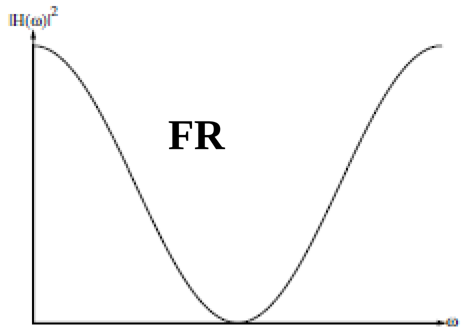
$y_n = x_{n-1} + x_{n+1}$

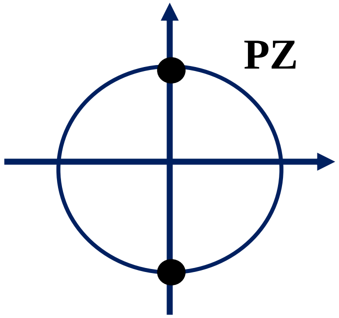
- this filter is **non**causal MA
- we can immediately guess that it is band-stop since
  - H(DC) = 2   (from H = 1 + 1)
  - H(Nyquist) = 2 (from H = 1 +1)
  - H(mid) = 0 (use x = -1 0 +1 0)
- impulse response is 1, 0, 1
- frequency response: $H(\omega)e^{i\omega n} = e^{i\omega(n-1)} + e^{i\omega(n+1)}$

  so $H(\omega) = e^{i\omega} + e^{-i\omega} = 2\cos(\omega)$
- transfer function

  $y = (\hat{z}^{-1} + \hat{z}^{+1})\,x$ so $H(z) = z + 1/z = z^2+1 = (z+i)(z-i)$



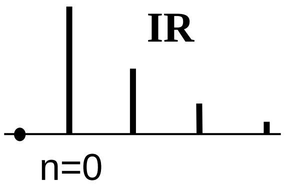




Why are there 2 zeros?

Exercise - AR

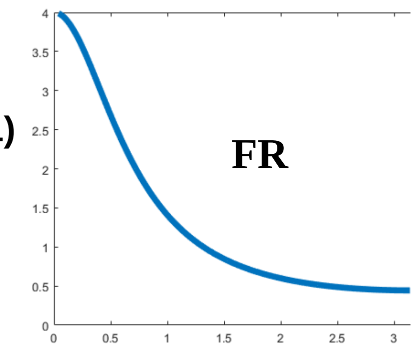
$y_n = x_n + ½\, y_{n-1}$



- this filter is causal AR
- we can immediately guess that it is LP since
  - H(DC) = 2   (from H = 1 + ½ H)
  - H(Nyquist) = 2/3 (from H = 1 – ½ H)
- impulse response is 1, 1/2, 1/4, 1/8, …
- frequency response: $H(\omega)e^{i\omega n} = e^{i\omega n} + ½\, H(\omega)e^{i\omega(n-1)}$

  so $H(\omega) = 1 / (1-½e^{-i\omega}) = 2/(2 - e^{-i\omega})$

  $|H(\omega)|^2 = 4 / (5 – 4 \cos(\omega))$

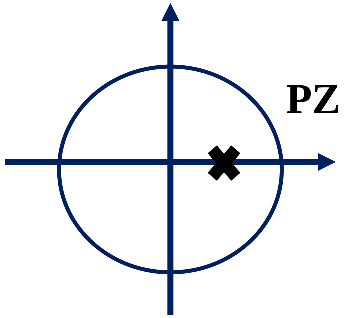


- transfer function

  $(1 – ½\, \hat{z}^{-1})\, y = x$

  so $H(z) = 1/(1- ½\, z^{-1}) = 1 / (z- ½)$

  i.e., 1 pole at ½



Exercise - ARMA

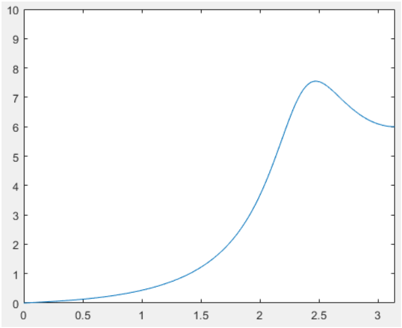
$$y_n = x_n - \frac{3}{2} x_{n-1} + \frac{1}{2} x_{n-2} - y_{n-1} - \frac{1}{2} y_{n-2}$$

- this filter is causal ARMA
- impulse response 1, $-\frac{5}{2}$, $+\frac{5}{2}$, $-\frac{5}{4}$, 0, …
- at DC $H(DC) = 1 - \frac{3}{2} + \frac{1}{2} - H(DC) - \frac{1}{2} H(DC)$ so $H(DC) = 0$
- at Nyquist $H(Nyq) = 1 + \frac{3}{2} + \frac{1}{2} + H(Nyq) - \frac{1}{2} H(Nyq)$ so $H(Nyq) = 6$
- frequency response

$$H(\omega)e^{i\omega n} = e^{i\omega n} - \frac{3}{2} e^{i\omega(n-1)} + \frac{1}{2} e^{i\omega(n-2)} - H(\omega)e^{i\omega(n-1)} - \frac{1}{2} H(\omega)e^{i\omega(n-2)}$$

$$H(\omega) = \frac{1-\frac{3}{2}e^{i\omega}+\frac{1}{2}e^{2i\omega}}{1+e^{i\omega}+\frac{1}{2}e^{2i\omega}} = \frac{-\frac{3}{2}+\cos(\boldsymbol{\omega})+\frac{1}{2}e^{i\omega}}{1+\cos(\boldsymbol{\omega})+\frac{1}{2}e^{i\omega}}$$

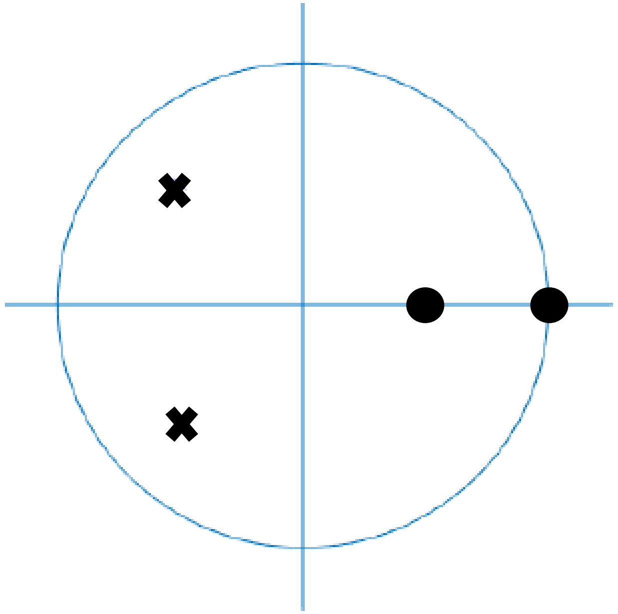
$$|\,H(\omega)\,|^2 = \frac{\frac{10}{4} - \frac{9}{2}\cos(\omega) + 2cos^2(\omega)}{\frac{5}{4} + 3\cos(\omega) + 2cos^2(\omega)}$$



Exercise - ARMA (cont.)

$$y_n = x_n - \frac{3}{2}x_{n-1} + \frac{1}{2}x_{n-2} - y_{n-1} - \frac{1}{2}y_{n-2}$$

see example for an ARMA filter



1. $$y_n + y_{n-1} + \frac{1}{2}y_{n-2} = x_n - \frac{3}{2}x_{n-1} + \frac{1}{2}x_{n-2}$$

2. $$\left(1 + \hat{z}^{-1} + \frac{1}{2}\hat{z}^{-2}\right) y = \left(1 - \frac{3}{2}\hat{z}^{-1} + \frac{1}{2}\hat{z}^{-2}\right) x$$

3. $$\left(1 + z^{-1} + \frac{1}{2}z^{-2}\right) Y(z) = \left(1 - \frac{3}{2}z^{-1} + \frac{1}{2}z^{-2}\right) X(z)$$

4. $$Y(z) = \frac{\left(1 - \frac{3}{2}z^{-1} + \frac{1}{2}z^{-2}\right)}{\left(1 + z^{-1} + \frac{1}{2}z^{-2}\right)} X(z)$$

$$H(z) = \frac{\left(1 - \frac{3}{2}z^{-1} + \frac{1}{2}z^{-2}\right)}{\left(1 + z^{-1} + \frac{1}{2}z^{-2}\right)} = \frac{\left(z^2 - \frac{3}{2}z + \frac{1}{2}\right)}{\left(z^2 + z + \frac{1}{2}\right)} = \frac{(z-1)(z-\frac{1}{2})}{(z+\frac{1}{2}(1+\mathrm{i}))(z+\frac{1}{2}(1-\mathrm{i}))}$$

5. 6.

substitute $z=e^{i\omega}$ for the frequency response