CE 395 Special Topics in Machine Learning

Assoc. Prof. Dr. Yuriy Mishchenko

Fall 2017

DIGITAL FILTERS AND FILTERING

Why filters?

- Digital filtering is the workhorse of digital signal processing
- Filtering is a fundamental operation of Deep Neural Networks
- Digital filters are used to pre-process data in many ML algorithms
- Digital filters are everywhere in speech recognition and computer vision
- Many more – in essence, DSP is filters

What are filters?

- Filter is any procedure for changing a signal (essentially a transformation)
- Filter may transform a 1D signal (time, signal filter) or image (spatial filter). For clarity we focus on 1D signals first.

Filter: signal $x(t) \rightarrow y(t)$

$$x(t) \rightarrow \boxed{\text{Filter}} \rightarrow y(t)$$

Linear filters

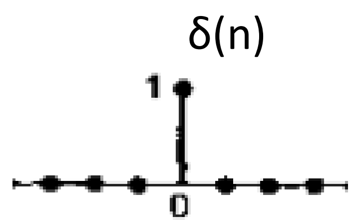
Linear filter is such a filter that transform signal in linear manner. That is, the output for a superposition of two input signals will be the superposition of the individual outputs of those two inputs.

$$F : x(t) \to x'(t) \text{ and } z(t) \to z'(t) \Rightarrow$$

$$F : x(t) + z(t) \to x'(t) + z'(t)$$

Linear filters

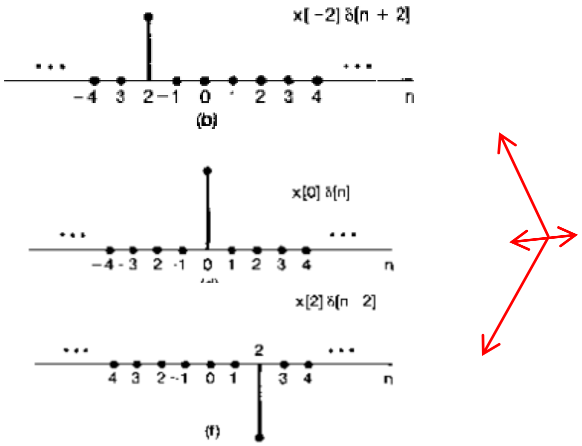
Delta function or **impulse signal** is a signal which equals to zero at all times except t0=0: δ(t0)=0 everywhere except t=t0, and δ(t0)=1 (we write δ(t-t0), to be precise)



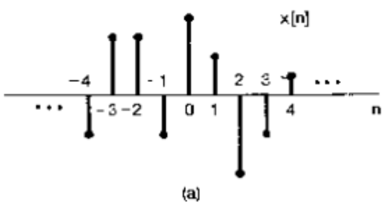
Linear filters

Fact: Any input signal can be represented as a sum of impulses.

$$\sum_{t_0=-\infty}^{\infty} x(t_0)\,\delta(t-t_0) = x(t)$$



Arbitrary signal as a sum of impulses:



Linear filters

Fact: Effect of filter on an impulse input signal at time t0 is **the response function $h_{t0}(t)$**

$$F[\delta(t - t_0)] = h_{t_0}(t) \Rightarrow$$

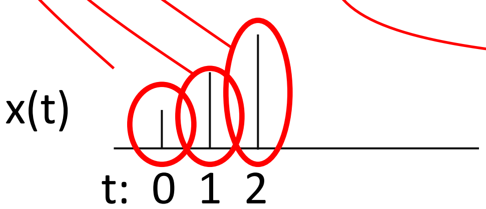
$$F[x(t) = \sum x(t_0)\delta(t - t_0)] = \sum h_{t_0}(t)x(t_0)$$

Linear filters

Imagine a signal as a sum of impulses at t=0,1,2…
Each impulse is transformed to independent time series $h_{t0=0}(t)$, $h_{t0=1}(t)$, $h_{t0=2}(t)$, …

h:t'/t	-2	-1	0	1	2
0	1	2	4	3	2
1	0	1	2	2	0
2	0	0	2	2	0

In the end, by linearity these are summed together!

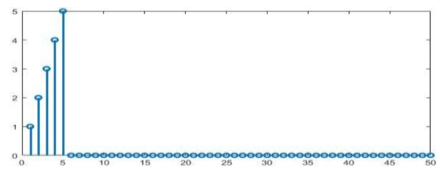


t=0→ [0 1 2 4 3 2 0]*x(0) ie $h_0(t)$
t=1→ [0 0 1 2 2 0 0]*x(1) ie $h_1(t)$
t=2→ [0 0 0 2 2 0 0]*x(2) ie $h_2(t)$
Σ: [0 1 3 8 7 2 0] = y(t)

t: -3 -2 -1 0 1 2 3

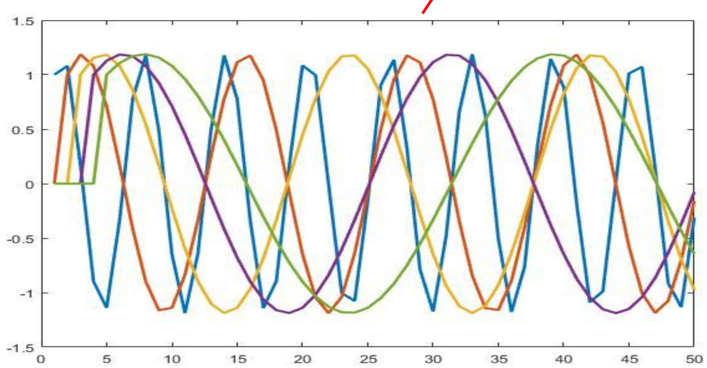
Linear filters as sum of responses:

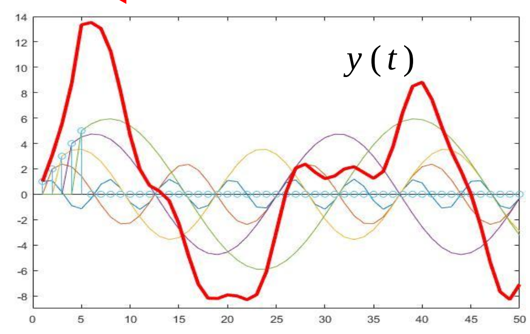
$$x(t) = t,\ 1 \le t \le 5$$



The filter output (thick line) is the sum of individual responses to x(t0) at times t0=1..5 (thin lines).

$$h_{t'}(t) = \sin(1)^{-1} \sin(t / t')$$





$$y(t) = \sum_{t_0} x(t_0) h_{t_0}(t)$$

LTI filters

Linear filter is called **<u>time-invariant</u> (LTI)** if its response is :

$$h_{t_0}(t) = h(t - t_0)$$

LTI filters

The main property of **time-invariant** filters is

$$F[x(t-t_0)](t) = F[x(t)](t-t_0)$$

Reads: shifting input by t0 only results in identical shift in the output signal

LTI filters

Using the definition property of linear filters, then we obtain for time-invariant filters the following formula:

$$F[x(t)] = \sum_{t_0=-\infty}^{\infty} h_{t_0}(t)\, x(t_0) \overset{LTI}{=} \sum_{t_0=-\infty}^{\infty} h(t - t_0)\, x(t_0)$$

LTI filters

The output of LTI filter is a superposition of single, time-invariant response function h(t) for each impulse x(t0) from the input

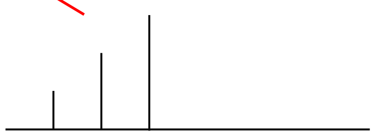
$$F_{LTI}[x(t)] = \sum_{t_0=-\infty}^{\infty} h(t-t_0)\, x(t_0)$$

Example of evaluation of LTI filter:

t	-2	-1	0	1	2
h(t)	0	1	2	3	4

LTI response function

x(t)



t: 0 1 2

t=0 $\rightarrow$ [0 0 1 2 3 4 0]*x(0) h(t-0)

t=1 $\rightarrow$ [0 0 0 1 2 3 4]*x(1) h(t-1)

t=2 $\rightarrow$ [0 0 0 0 1 2 3]*x(2) h(t-2)

Σ: [0 0 1 4 10 16 17] = y(t)

t: -3 -2 -1 0 1 2 3

Operation of Convolution

The peculiar mathematical operation that we saw in the time-invariant filters has a general significance and is called in mathematics **convolution**:

$$F_{LTI}[x(t)] = \sum_{t_0=-\infty}^{\infty} h(t-t_0)\,x(t_0)$$

$$x(t) * y(t) = \sum_{t'=-\infty}^{\infty} x(t')\,y(t-t')$$

Operation of Convolution

Convolution and time-invariant filters are basically the same thing: You can visualize convolution as summing up of h(t-t0) responses triggered by each incoming pulse x(t0)

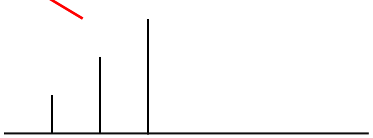
Operation of Convolution

This filter example from before is indeed a convolution:

t	-2	-1	0	1	2
h(t)	0	1	2	3	4

Convolution h(t)

x(t)



t: 0 1 2

t=0→ [0 0 1 2 3 4 0]*x(0) h(t-0)

t=1→ [0 0 0 1 2 3 4]*x(1) h(t-1)

t=2→ [0 0 0 0 1 2 3]*x(2) h(t-2)

Σ: [0 0 1 4 10 16 17] = y(t) = h(t)*x(t)

t: -3 -2 -1 0 1 2 3

Operation of Convolution

There may exist different methods for actually evaluating convolution:

- Sliding response mask (previous slide)
- Sliding inverted mask (next slide)
- Diagonal method [construct a table of $x(t) \cdot y(t')$ for all t and t', then sum the values on the counter-diagonals $t+t'=\tau \Rightarrow z(\tau)$ (**verify at home that that indeed gives the convolution**)]

Convolution calculation via diagonals:

t	-4	-3	-2	-1	0	1	2	3	4
x(t)	0	0	0	0	1	2	3	0	0
y(t)	0	1	2	3	4	0	0	0	0

$z(\tau = 0) = 4 + 6 + 6 = 18$

$z(\tau = 1) = 8 + 9$

$z(\tau = -2) = 2 + 2 = 4$

x\y	t=-3	t=-2	t=-1	t=0
t=0	1	2	3	4
t=1	2	4	6	8
t=2	3	6	9	12

τ=-2 τ=0 τ=1

z(t)	0	1	4	10	16	17	12	0	0

Convolution calculation via inverted mask:

t	-4	-3	-2	-1	0	1	2	3	4
x(t)	0	0	0	0	1	2	3	0	0
y(t)	0	1	2	3	4	0	0	0	0
y(-t)	0	0	0	0	4	3	2	1	0
z(t)	**0**	**1**	**4**	**10**	**16**	**17**	**12**	**0**	**0**

Inverted mask,
slide left right

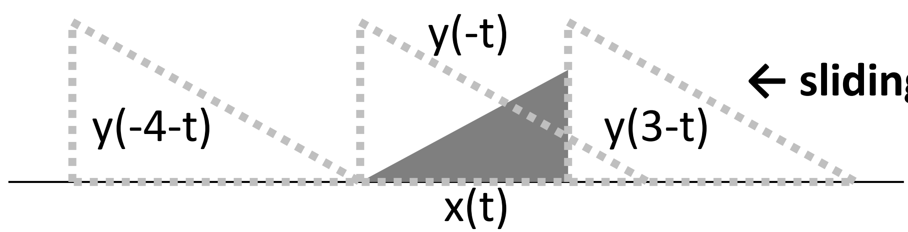
$$z(t=0)=\sum_{k=-\infty}^{\infty} x(k)\,y(0-k)=\sum_{k=-\infty}^{\infty} x(k)\,y(-k)=$$

$$=x(-1)\,y(1)+x(0)\,y(0)+x(1)\,y(-1)+x(2)\,y(-2)+x(3)\,y(-3)+x(4)\,y(-4)$$

=1*4+2*3+3*2=16

$$z(1)=\sum_{k=-\infty}^{\infty} x(k)\,y(1-k)=x(1)\,y(0)+x(2)\,y(-1)$$ =2*4+3*3=17

$$z(2)=\sum_{k=-\infty}^{\infty} x(k)\,y(2-k)=x(2)\,y(0)=12$$



← sliding inverted mask

Main properties of convolution

For a DSP engineer the most important thing is to know and understand the **properties of convolution**. All of these can be shown directly from the definition (exercise for home).

$x(t) * h(t) = h(t) * x(t)$	Commutative
$x(t) * (h(t) + g(t)) = x(t) * h(t) + x(t) * g(t)$	Distributive
$(x(t) * h(t)) * g(t) = x(t) * (h(t) * g(t))$	Associative
$x(t) * \delta(t) = x(t)$	Identity
$x(t) * \delta(t - t_0) = x(t - t_0)$	Time-shift

LTI filters

- FIR filters are such filters whose response h(t) is non-zero only in a small region around t=0. Such filters are also oftentimes called **masks**.
- The result of applying an FIR filter reduces to summing the signal x(t) with a sliding response mask h(t) or convolving it with inverted mask h(-t), see previous slides for examples

LTI filters

- LTI filter is called causal if response h(t)=0 for all t<0. That is, the signal can only affect the filtered signal y(t) in the future

$$F[x(t)] = \sum_{t_0=-\infty}^{t_0=t} h(t - t_0)\,x(t_0)$$

- For example, the filter in slide 18 is not causal, because x(t) affects y(t-1). However, if we set h(-1)=0, then that filter would be causal.

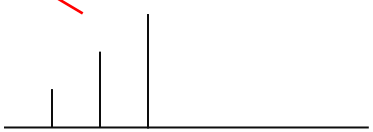
Causal and non-causal filters

t	-2	-1	0	1	2
h(t)	0	1	2	3	0

Response function

Note x(0) affects y(-1) therefore this filter is **not causal.** Setting h(-1)=0 on the other hand results in **causal** filter.

x(t)



t: 0 1 2

t=0→ [0 0 1 2 3 0 0]*x(0) h(t-0)

t=1→ [0 0 0 1 2 3 0]*x(1) h(t-1)

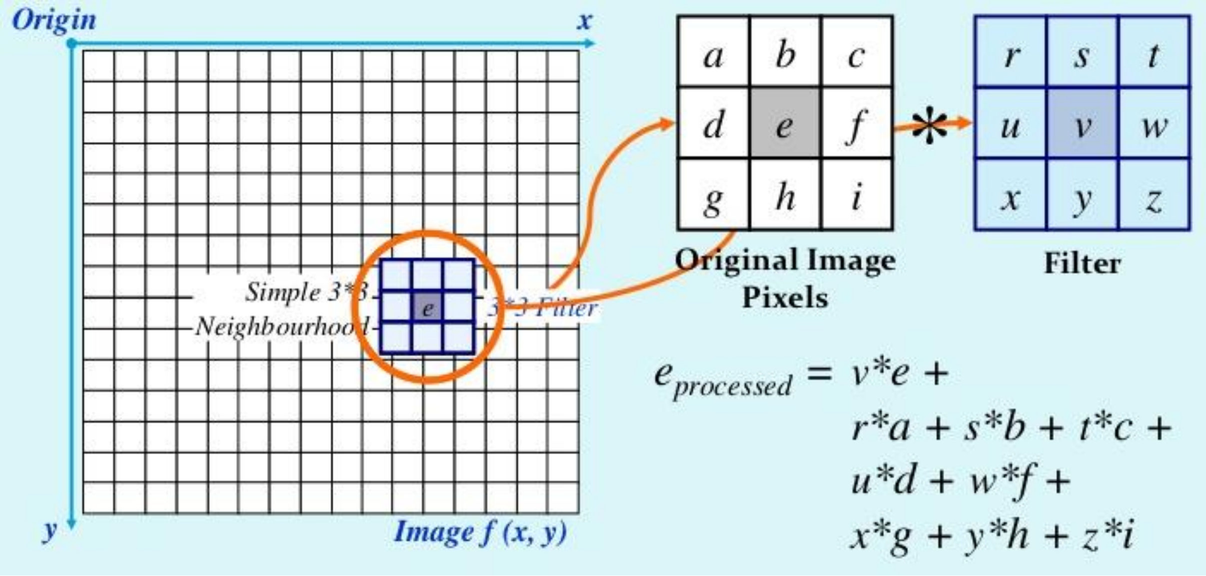
t=2→ [0 0 0 0 1 2 3]*x(2) h(t-2)

Σ: [0 0 1 4 10 12 9] = y(t)

t: -3 -2 -1 0 1 2 3

Spatial filters

Typical filters used in image processing are FIR filters that can be viewed as a 2D response mask (typically square) being moved around the image, multiplied with image pixel values, and then summed to define the new pixel values

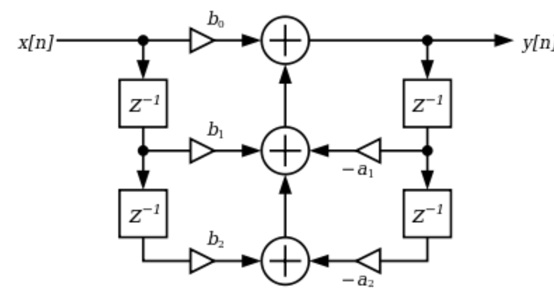


Spatial filters

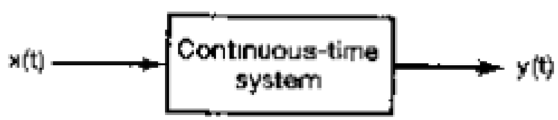
See more details about how a spatial filter works in these great animations on youtube!
https://www.youtube.com/watch?v=gFELyrIx010
https://www.youtube.com/watch?v=hRtmSh2gF48

System representations of filters

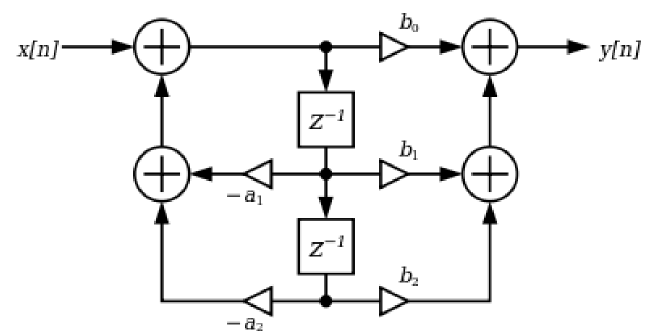
Examples of systems:



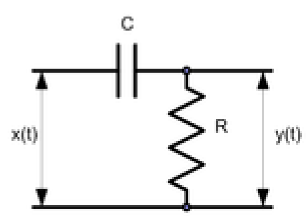
$$x(t) \rightarrow y(t).$$

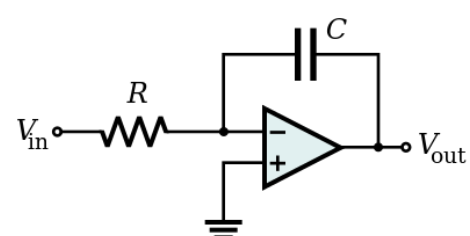


(a)



(b)





System representations of filters

- System can be simplistically viewed as definitions of sequences of processing steps for transforming input signal x(t) into an output signal y(t), $x(t) \rightarrow y(t)$
- As we can see, this definition is not much different from what we said about filters - In fact, systems **are** filters in many respects!

System representations of filters

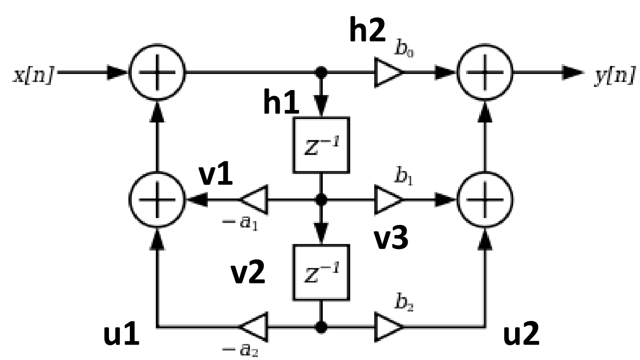
- The difference between systems and filters is the point of view: systems are viewed primarily as sequences of operations or machines, while filters are treated more as mathematical transformations
- Respectively, systems are naturally represented by diagrams of signal processing steps, while filters are expressed by mathematical formulas

System representations of filters

- **System diagrams** represent the sequence of operations to be carried out on an input signal to obtain the output, in a diagram form
- **Main elements of system diagrams** are elementary operations (+,-,*, unary -x), signal branching (copy), and time delays (labeled by Z^{-1} means x[t-1])
- Additionally, **systems may use other systems** as building blocks

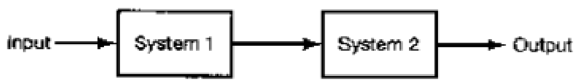
Example of reading a system diagram

- Copy x[n] into two branches (h1 and h2)
- Signal in branch h2 is multiplied with b_0
- Signal in branch h1 is time-delayed by 1 step, x[n-1]
- Thus time-delayed signal h1 is copied three ways – v1, v2 and v3
- Signal v2 is further time delayed by 1 step (x[n-2]) and then copied two ways – u1 and u2
- Signal in branch u1 is multiplied by $-a_2$
- Signal in branch u2 is multiplied by b_2
- Signal in branch v1 is multiplied by $-a_1$ and added to u1
- Signal in branch v3 is multiplied by b_1 and added to u2
- Signal in branch u1 is added back to input x[n] (feedback)
- Signal in branch u2 is added to h2, thus forming the result (y[n]) to be served as output



System representations of filters

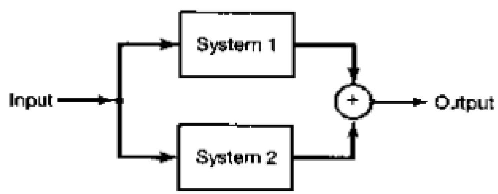
Systems can be interconnected to create more complex systems:



(a)

$$y = s2(s1(x)) = s2 \circ s1(x)$$

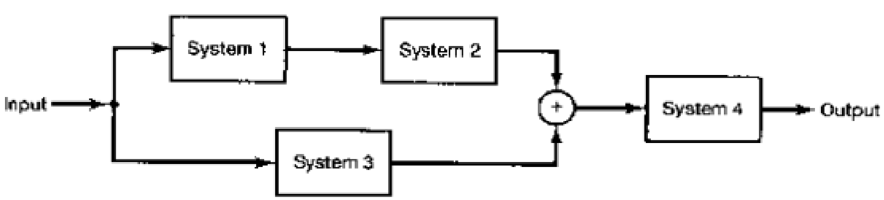
(series: output of system 1 becomes input to system 2**)**



(b)

$$y = s1(x) + s2(x) = (s1 + s2)(x)$$

(parallel: after processing signal over two parallel branches results are plus'ed together**)**



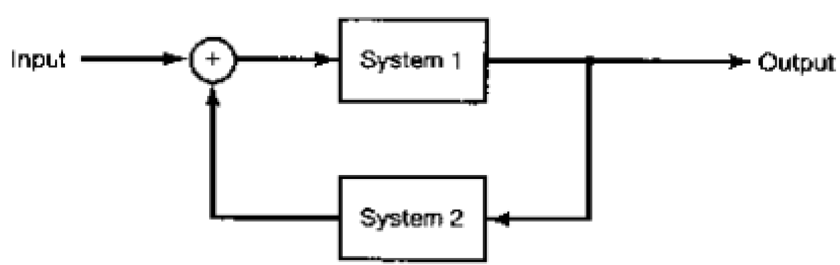
(c)

$$y = s4(s2(s1(x)) + s3(x))$$

NOW TRY AND OBTAIN THIS

System representations of filters

Extremely important is **feedback interconnection:** *output* of a system is used back as its *input* (possibly after additional processing)



Shown feedbacks is

$$x(t) \to y(t) = S_1(x(t) + S_2(y(t)))$$

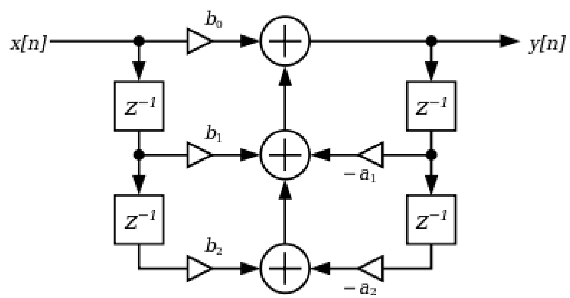
System representations of filters

- **Try to design and draw the system diagrams for following systems or digital filters:**
  - *Differentiator* filter D_1 $x(t) \to x(t)-x(t-1)=y(t)$
  - *Integrator* filter S_1 $x(t) \to x(t)+y(t-1)=y(t)$

System representations of filters

For us the important point is that one can use systems language to design digital filters

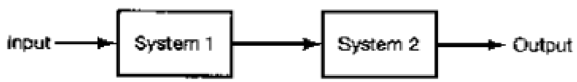
- This in fact is a filter:



- In systems language, we can combine multiple filters in a simple way, parallel, series, or feedbacks, to create very complex filters!

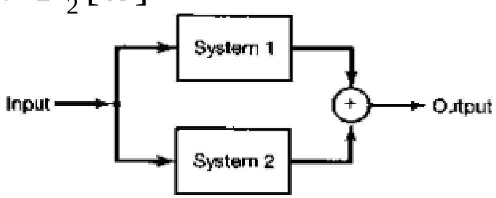
System representations of filters

$$y = F_2[F_1[x]]$$

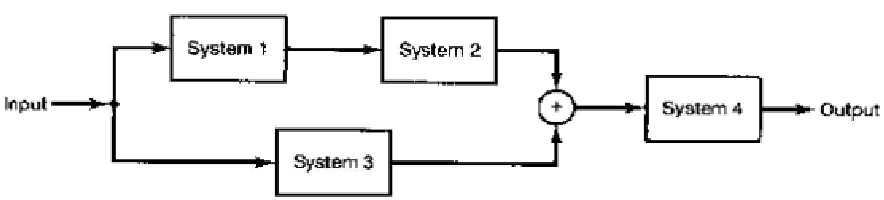


(a)

$$y = F_1[x] + F_2[x]$$



(b)



(c)

- **Series:** A new filter is obtained by applying filter F1 and then F2 to F1's result
- **Parallel:** A new filter is obtained by applying F1 and F2 and then summing their outputs

←Now try and explain what kind of filter this last diagram is

System representations of filters

Like a filter, a system is called **linear** if

$$s(x_1 + x_2) = s(x_1) + s(x_2)$$

A system is called **time-invariant** if its characteristics don’t change over time, that is

$$s(x(t + t_0)) = y(t + t_0)$$

System representations of filters

A system is **LTI (Linear Time-Invariant)** if it is linear and time invariant at the same time

LTI systems always can be described by convolution just like the LTI filters

$$s(x(t)) = \sum_{t'=-\infty}^{\infty} x(t')h(t-t') = x(t) * h(t)$$

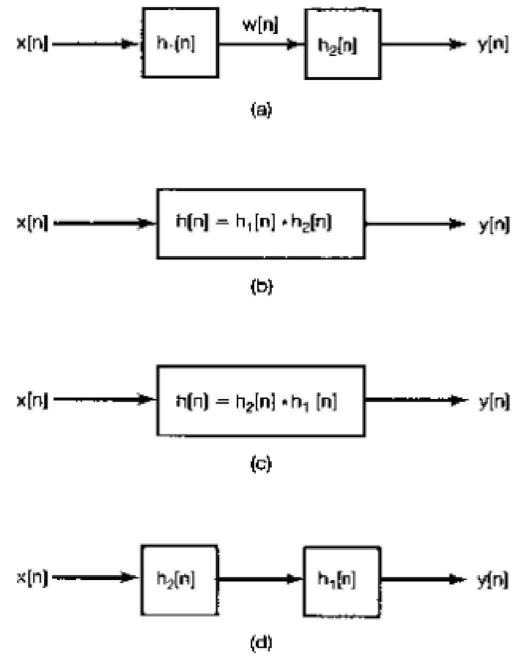
Helpful interconnection identities for LTI systems/filters

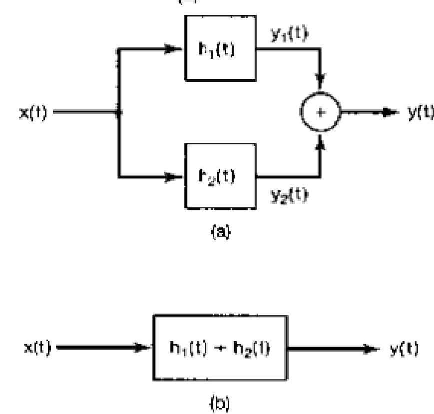
$$s_1(t) * s_2(t) = s_2(t) * s_1(t)$$ **Commutative**

$$s_1(t) * (s_2(t) + s_3(t)) = s_1(t) * s_2(t) + s_1(t) * s_3(t)$$ **Distributive**

$$(s_1(t) * s_2(t)) * s_3(t) = s_1(t) * (s_2(t) * s_3(t))$$ **Associative**

$$x(t) * \delta(t) = x(t)$$ **Identity**

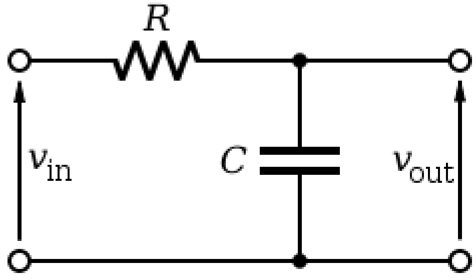




Follow from the properties of <u>convolution</u> now

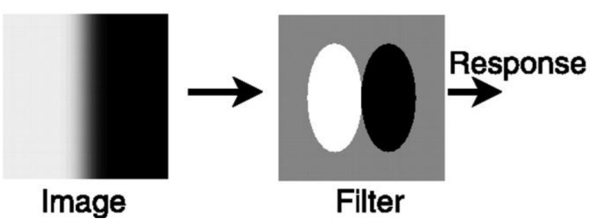
Digital Signal Processing

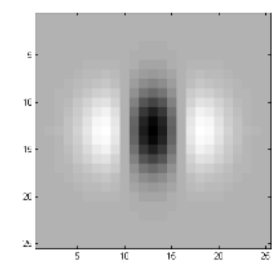
- Original usage of filters:



- Modern usage of filters - Digital Signal Processing (**DSP**)

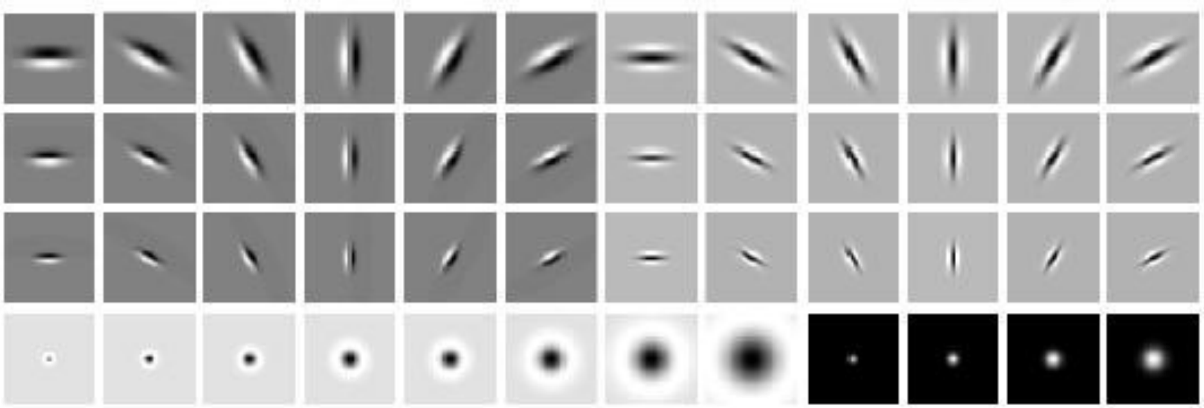
Digital Signal Processing





Digital Signal Processing

Banks of filters



Features for
ML algorithms

Digital Signal Processing

Detect edges using a FIR image filter

-1	0	+1
-2	0	+2
-1	0	+1

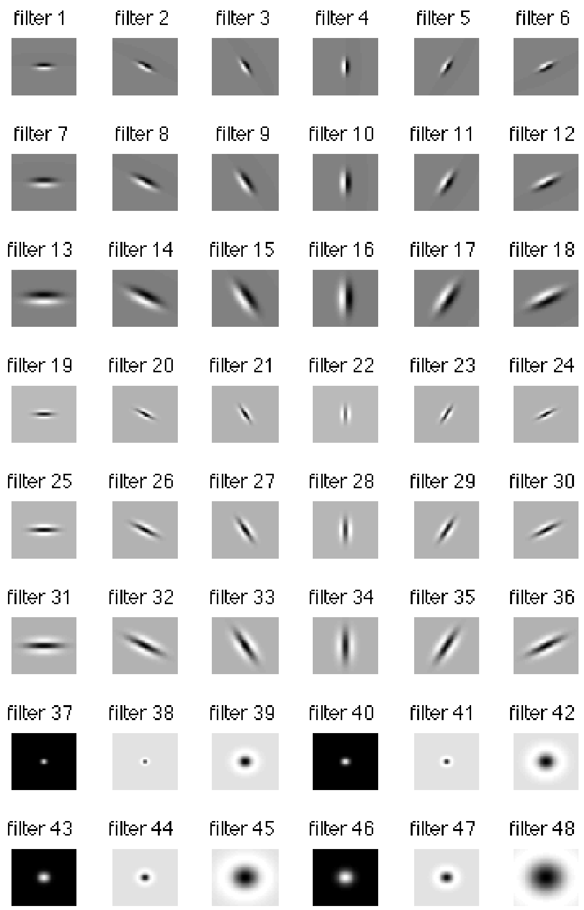
Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

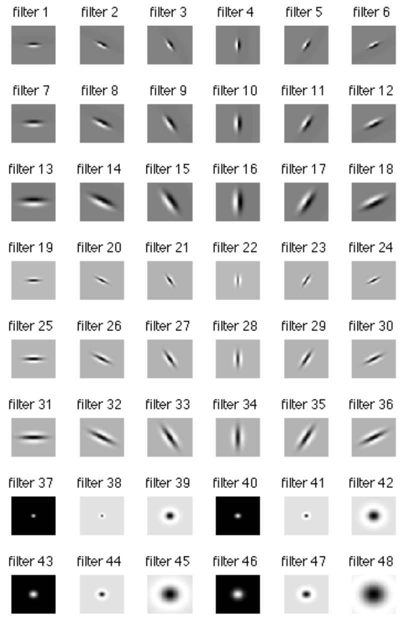
Digital Signal Processing

Banks of multi-scale filters



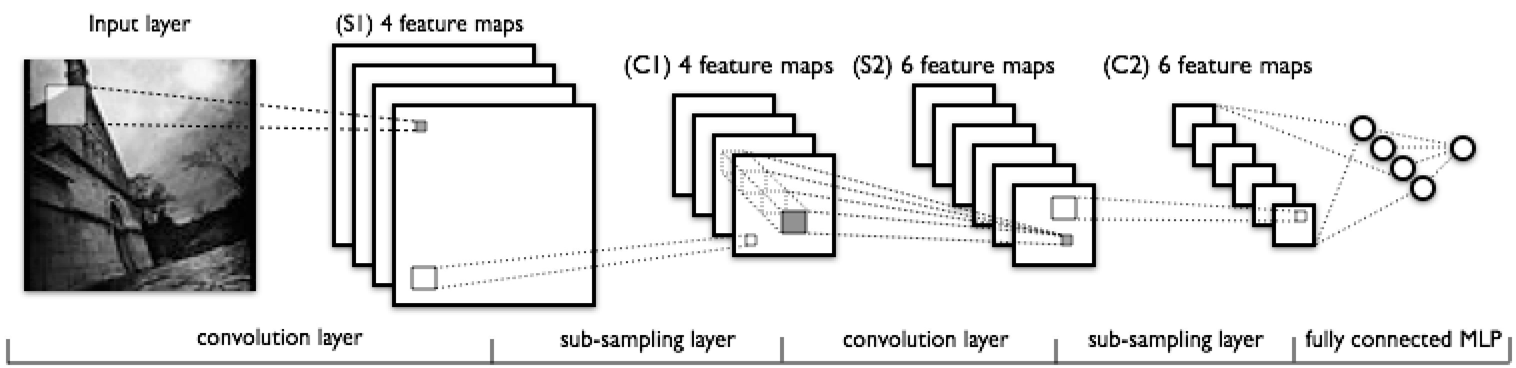
Across scales features for ML algorithm

Digital Signal Processing

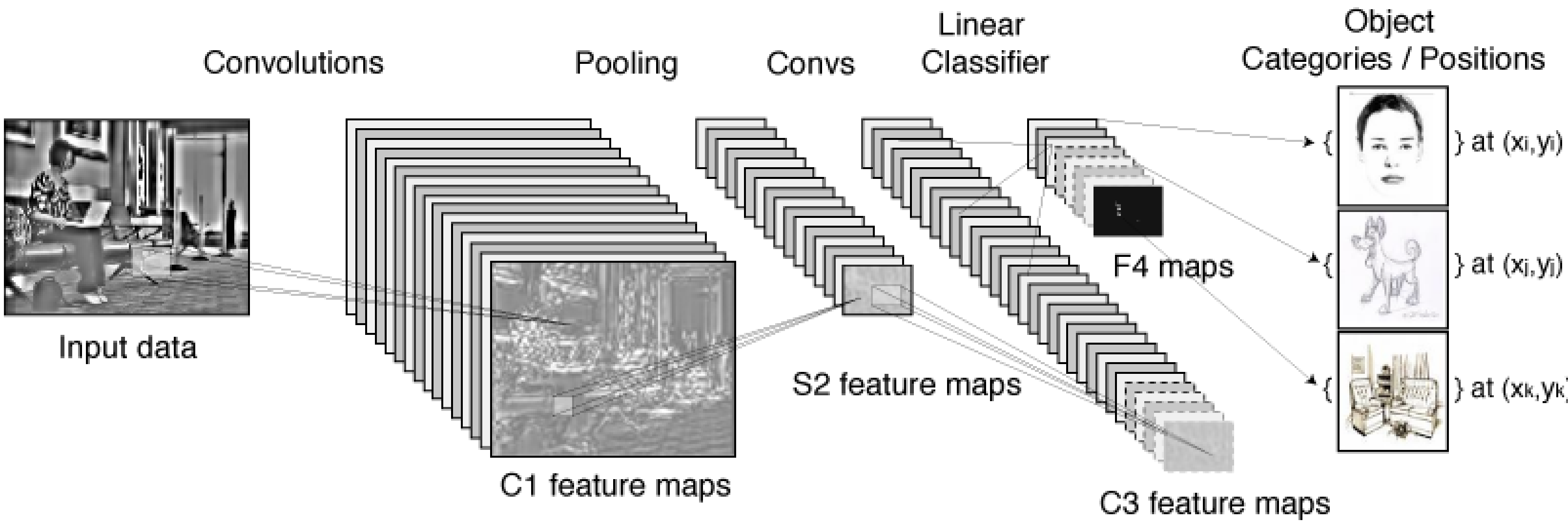


Multi-scale features for ML algorithms

A Convolutional Neural Network



Digital Signal Processing



Digital Signal Processing

- **Problem:** design a filter that has required properties (for example – responds strongly to vertical edges in an image)
- There are powerful software tools for that – you can learn those when/if you need them

Digital filters' representation

- **Problem:** how can we calculate a result of filtering a signal on a computer?

(most general form)

$$y(n) = b(1)x(n) + b(2)x(n-1) + \dots + b(n_b+1)x(n-n_b)$$
$$-a(2)y(n-1) - \dots - a(n_a+1)y(n-n_a).$$

Digital filters' representation

- Direct form 1:

$$y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b)$$
$$-a(2)y(n-1) - \ldots - a(n_a+1)y(n-n_a).$$

$$\Updownarrow$$

$$\left(1 + a(2)Z^{-1} + \ldots + a(n_a+1)Z^{-n_a}\right)y(n) = \left(b(1) + b(2)Z^{-1} + \ldots + b(n_b+1)Z^{-n_b}\right)x(n)$$

filter order

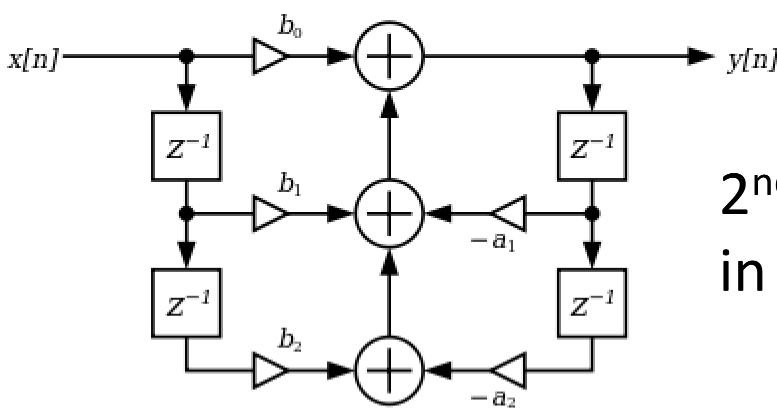
Z^{-1} indicates time-shift

$$Z^{-1}x(n) = x(n-1)$$

Digital filters' representation

- Direct form 1:

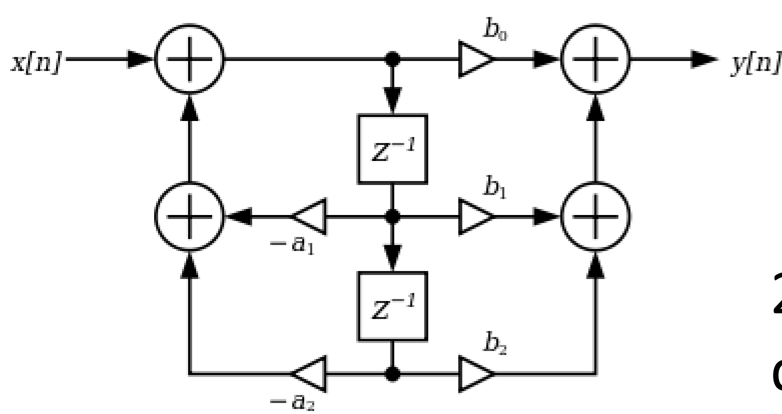
$$y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b)$$
$$-a(2)y(n-1) - \ldots - a(n_a+1)y(n-n_a).$$



2nd order filter in direct form 1

Digital filters' representation

- Direct form 2:
  Try and follow this system diagram to see how this type of filters works.



2^{nd} order filter in direct form 2

Digital filters' representation

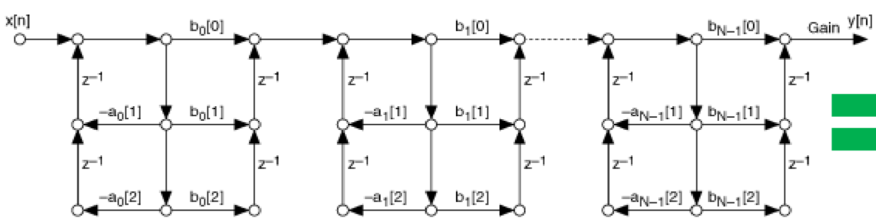
Digital filters in *direct form* may be unstable for evaluation because of numerical overflows in the long chains of additions of possibly very different scale-numbers – **this problem is typically exponentially bad in the filter order N**

$$y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b)$$
$$-a(2)y(n-1) - \ldots - a(n_a+1)y(n-n_a).$$

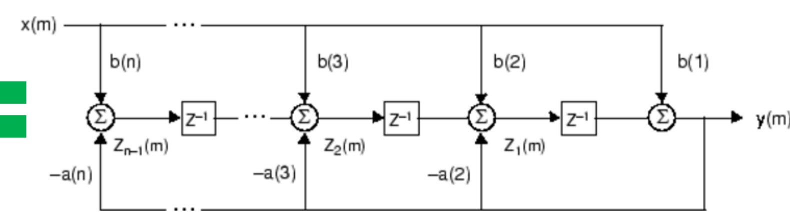
Digital filters’ representation

SOS – Second Order Sections (biquadratic cascade) filter representation form

Theorem: Any digital filter can be represented as 2^{nd} order filters cascade.



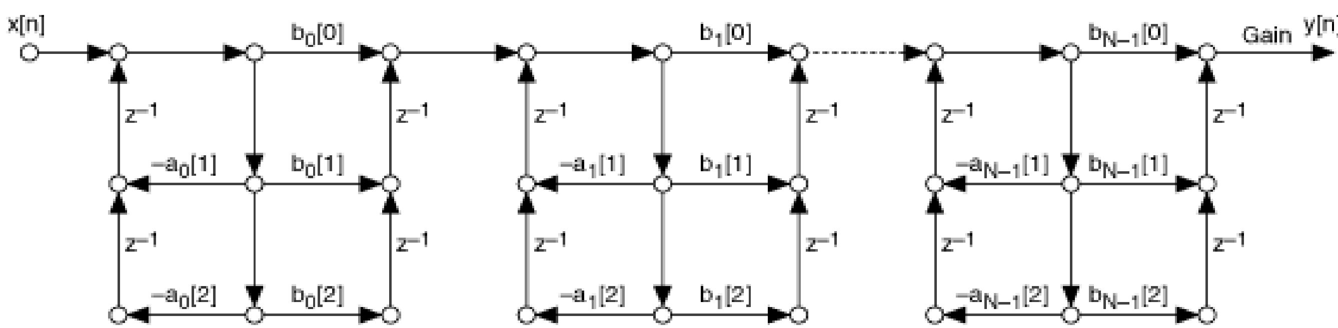
SOS



Direct form 1

Digital filters' representation

SOS filters have much better numerical stability and are the preferred form of representation for digital filters



Digital filters' representation

Transfer function representation of filters we will come back to after learning about Fourier transform

Nonlinear filters

- A nonlinear filter contains a nonlinear transformation in addition to linear filtering
- Very different ways to implement, active area of research
- Some examples:

$$y(t) = f\left(\sum_{t'=-\infty}^{\infty} h(t-t')x(t') \right)$$

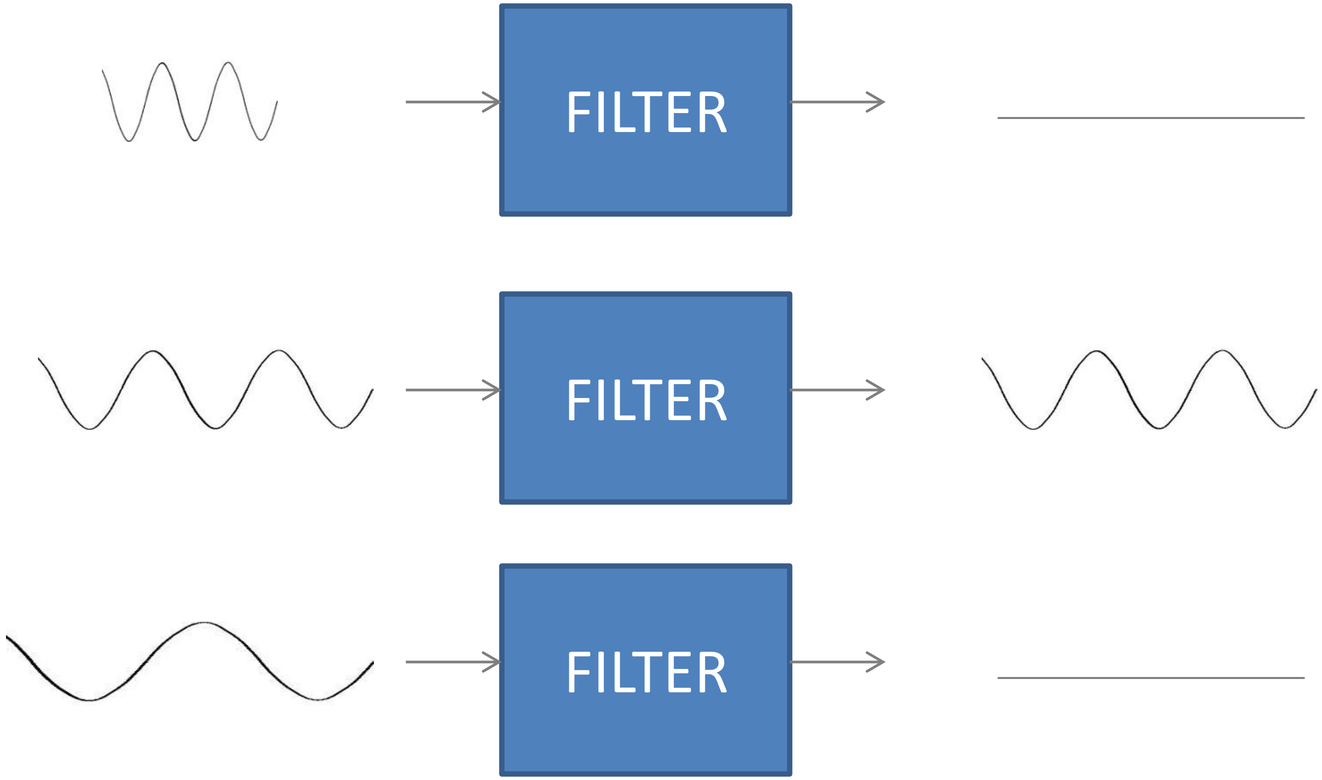
$$y(t) = \sum_{t'=-\infty}^{\infty} h(t-t') f\left(x(t')\right)$$

$$\Delta y(t) = f\left(Ay(t) + Bx(t)\right)$$

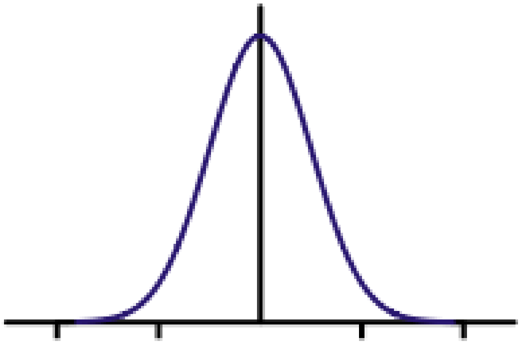
Some important DSP filters

- **Low pass** – suppress high frequencies in signals or images resulting in generally smoother signal
- **High pass** – enhance high frequencies in signals or images resulting in generally sharper looking features in signals with emphasize on edges
- **Band pass** – allow frequencies in signals or images in certain range (band) to pass
- **Band stop (Notch)** – remove frequencies in signals or images in certain range

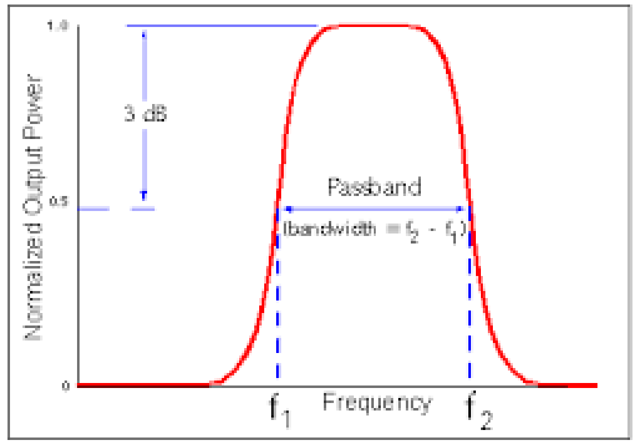
Some important DSP filters



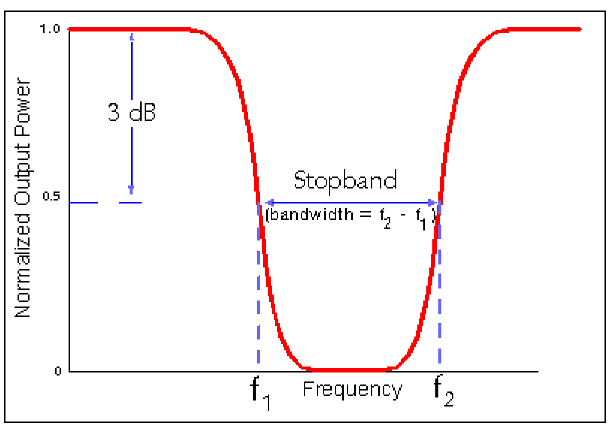
Some important DSP filters



Gaussian low pass
or smoothing filter

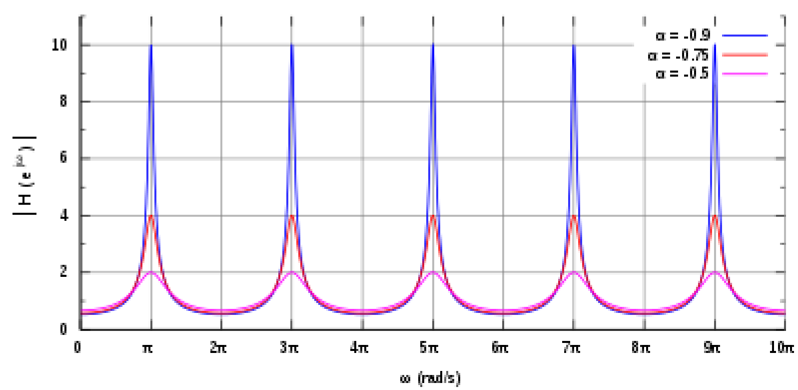


Band pass filter



Notch (band stop) filter

Comb filter

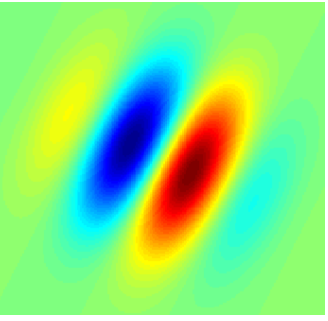


Some important DSP filters

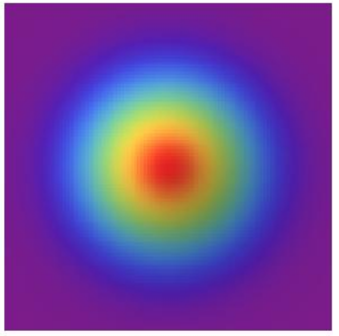
- **FIR (Finite Impulse Response Filters)** – FIR filters have finite span of the response function h(t) and can contain no feedback, reduce to mask application over signal
- **IIR (Infinite Impulse Response Filters)** – IIR filters can have infinite span of response function h(t) and are the general filters with feedback

Some important DSP filters

Most filters used in spatial or image filtering are **FIR** filters and can be described by a 2D, usually square **filter mask** being applied over local image patches to produce filtered image's pixel values



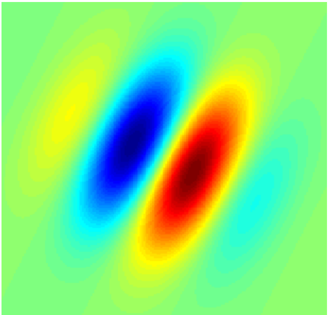
Gabor (edge) filter mask



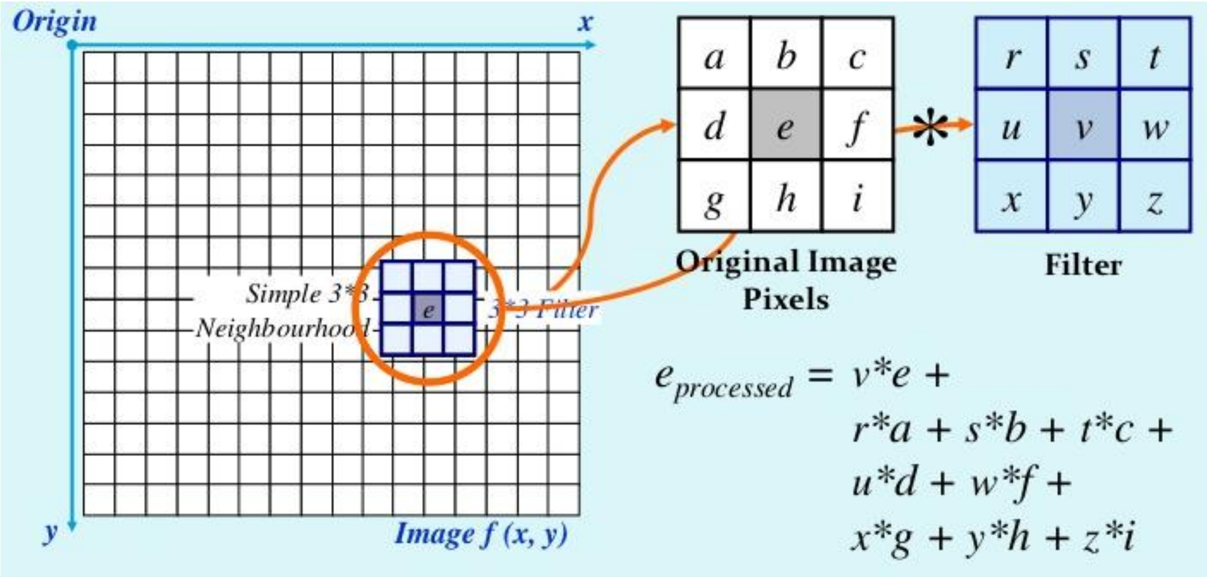
Gaussian (smoothing) filter mask

Some important DSP filters

We have talked about how such spatial masks are applied earlier.



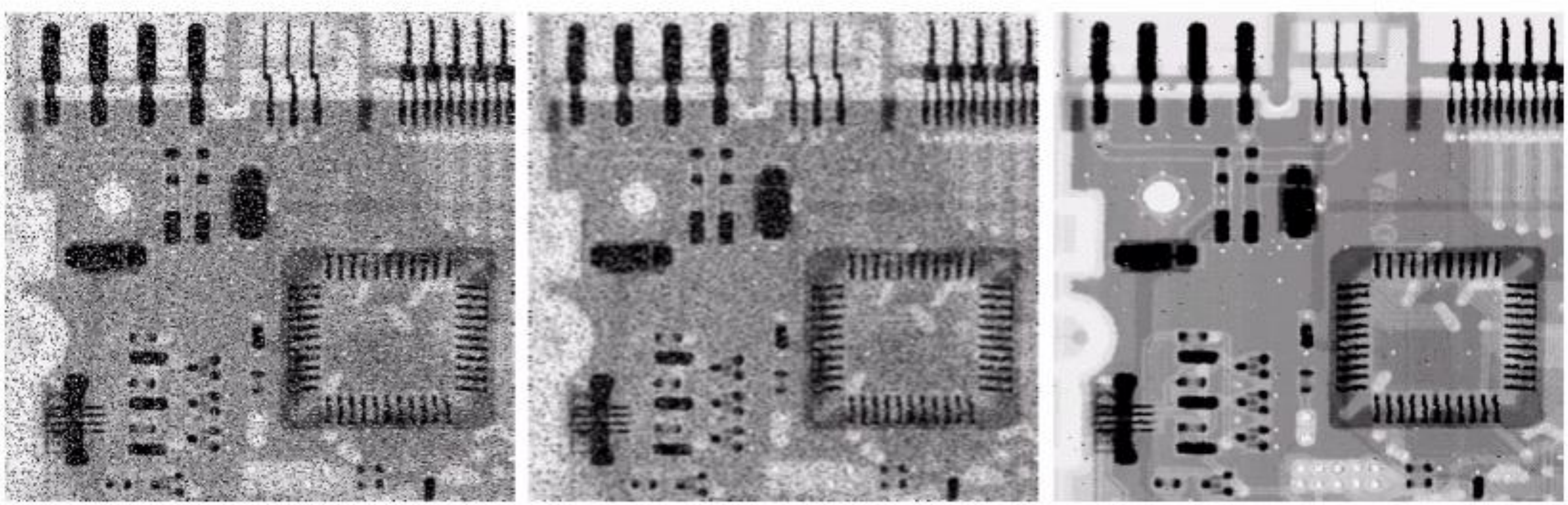
Gabor (edge) filter mask



Some important DSP filters

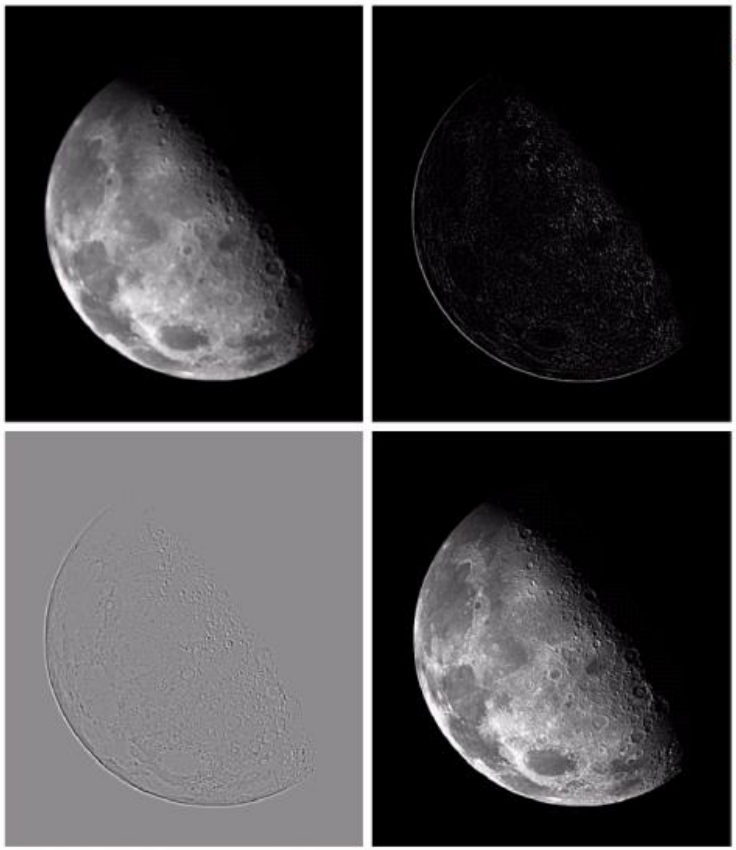
Averaging, smoothing and denoising:

3x3 average/mean 3x3 median



Some important DSP filters

Sharpening and edge detectors:



$$H_h = \frac{1}{4}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$H_s = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 16 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Some important DSP filters

- Simple edge detectors: Sobel, Prewitt, Roberts, Gradient, Laplacian, Gradient of Gaussian (GoG)
- Ridge detectors - Laplacian, Laplacian of Gaussian (LoG)

- Sobel

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Prewitt

$$\frac{1}{6}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

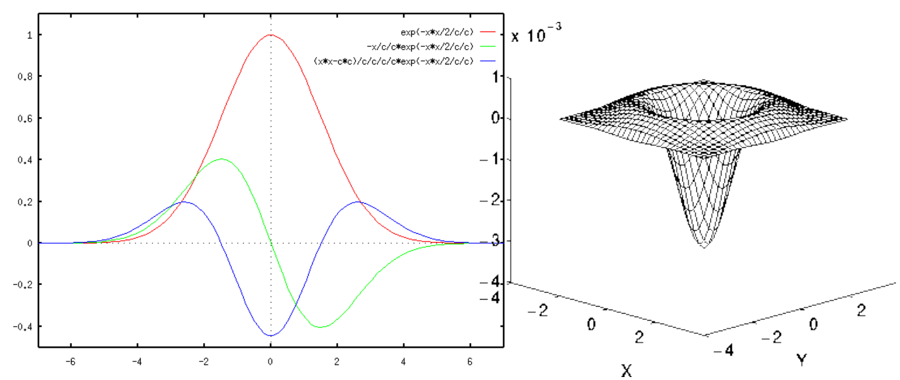
- Roberts

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- Laplacian

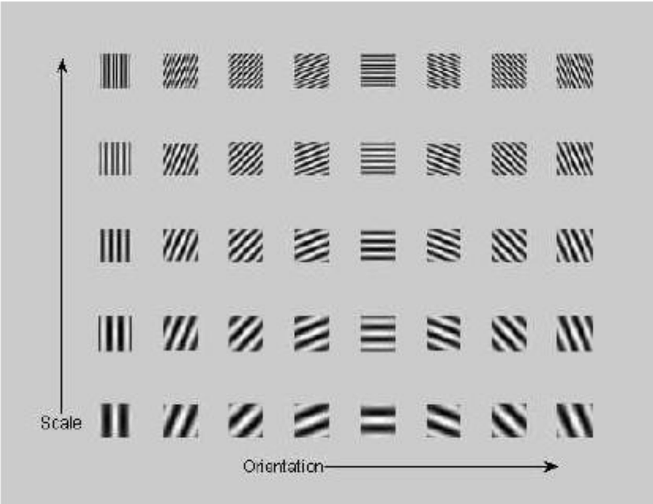
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplacian of Gaussian filter

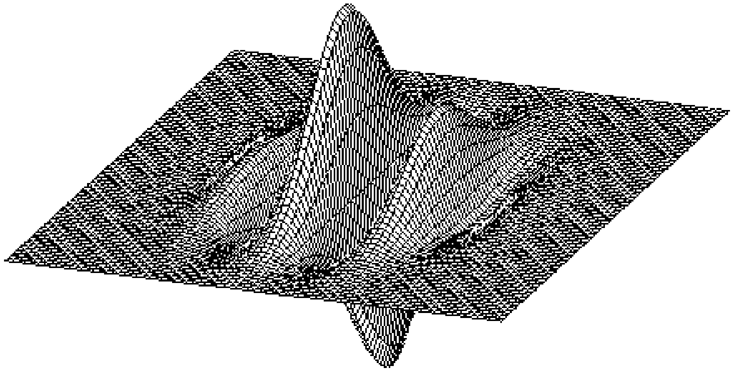


Some important DSP filters

- Texture detectors – Gabor filters, wavelet filters
- Hough Transform – used to detect shapes in images such as lines and circles



Gabor



QUESTIONS FOR SELF-CONTROL

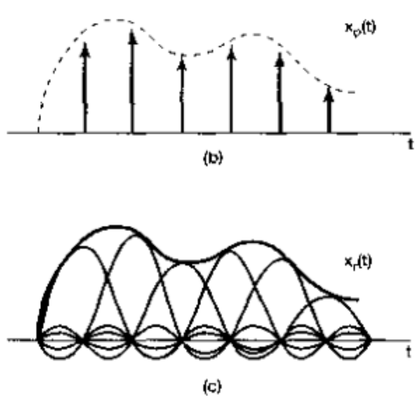
- Define what a filter is. What is linear filter?
- What do we mean when we say that linear filter is a sum of filter responses?
- Define LTI filter.
- Define causal filter.
- Define the operation of convolution.
- List main properties of convolution.
- Define the three main system interconnection types.
- Give definition of LTI system.
- Draw differentiator and integrator systems, are these LTI?
- In what sense do we say that LTI systems and linear filters are equivalent?
- What is filter bank and how is it used for image processing?

- Define direct forms 1 and 2.
- Define what order of filter is.
- What would be the order of the filter corresponding to the integrator system you constructed in a previous question?
- Define SOS filter.
- Explain why SOS filters are better for digital signal processing than direct form filters?
- Define nonlinear filter.
- Define low pass, high pass, band pass, and band stop filters.
- What is the difference between FIR and IIR filters?
- Explain how spatial filtering by a filter mask works on the example of an edge detector image filter.

ADVANCED

Ideal low pass filter

Signal reconstruction by ideal low pass filter can be viewed **as a convolution of sampled signal with sinc function**



$$x(t) = \sum_{n=-\infty}^{\infty} x(nT)\,\mathrm{sinc}(t - nT) = x(t) * \mathrm{sinc}(t)$$

2D convolution

Although we considered only 1D convolution, it can be generalized to any number of dimensions

$$f(x,y) * h(x,y) = \sum_{x'=-\infty}^{\infty} \sum_{y'=-\infty}^{\infty} f(x',y')h(x-x',y-y')$$

Some general properties of systems

- Memory – systems with memory produce output depending on many past values from x(t) not just the current value
- Invertability – an inverse system g exists for a system s if their series interconnection produces identity – $g*s=\delta$, we say $g=s^{-1}$
- Causality – a system is causal of its output at time t depends only on the past values of x(t’), t’<t)
- Stability – a system is stable if its output is bounded for any finite input x(t)

Some properties of LTI systems

$y(t) = \sum_{t'=-\infty}^{\infty} h(t')x(t-t') = x(t) * h(t)$ implies LTI **must have memory** (the only memory-less LTI system is the identity system δ)

Some properties of LTI systems

LTI system h is invertible if there is another system g such that $g*h=\delta(t)$ (the resolution of this problem is obtained in the theory of Fourier transform next lecture)

Some properties of LTI systems

LTI system is stable if its output is bounded in magnitude for every bounded input. This implies and requires (sufficient and necessary):

$$\sum_{t'=-\infty}^{\infty} \left| h(t') \right| < \infty$$

$$\begin{aligned}
|y(t)| &= \left| \int_{-\infty}^{+\infty} h(\tau)x(t-\tau)d\tau \right| \\
&\le \int_{-\infty}^{+\infty} |h(\tau)||x(t-\tau)|d\tau \\
&\le B \int_{\infty}^{+\infty} |h(\tau)|d\tau.
\end{aligned}$$

Some properties of LTI systems

LTI is causal iff $h(t)=0$ for all $t<0$

$$y(t) = \sum_{t'=0}^{\infty} x(t')h(t-t') = x(t) * h(t)$$

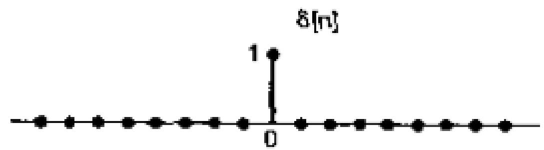
Step response of causal LTI system:

$$LTI\ [u(t)] = \sum_{t'=0}^{t} h(t')$$

Some properties of LTI systems

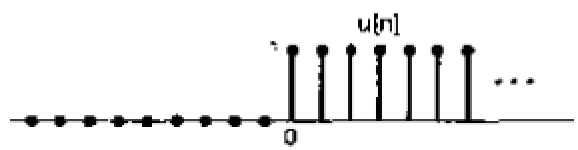
Delta or impulse signal and unit step signal

$$\delta(t) = \begin{cases} 0, t \neq 0 \\ 1, t = 0 \end{cases}$$

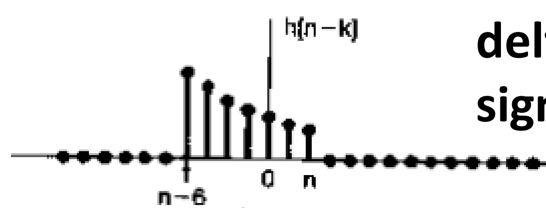


$$y(t) = h(t)$$

$$u(t) = \begin{cases} 0, t < 0 \\ 1, t \geq 0 \end{cases}$$



$$y(t) = \sum_{t'=-\infty}^{t} h(t')$$



Response of LTI system to delta/impulse and step signal

LTI systems and dynamic systems

Many causal LTI can be described by differential or finite difference equations - such equations describe effectively how the output of the system is computed:

$$\frac{dy(t)}{dt} - Ay(t) = Bx(t) \qquad \Delta y(t) = Ay(t) + Bx(t)$$ **1st order LTI**

$$\frac{d^2 y(t)}{dt^2} + A_1 \frac{dy(t)}{dt} - A_0 y(t) = Bx(t)$$ **2nd order LTI**

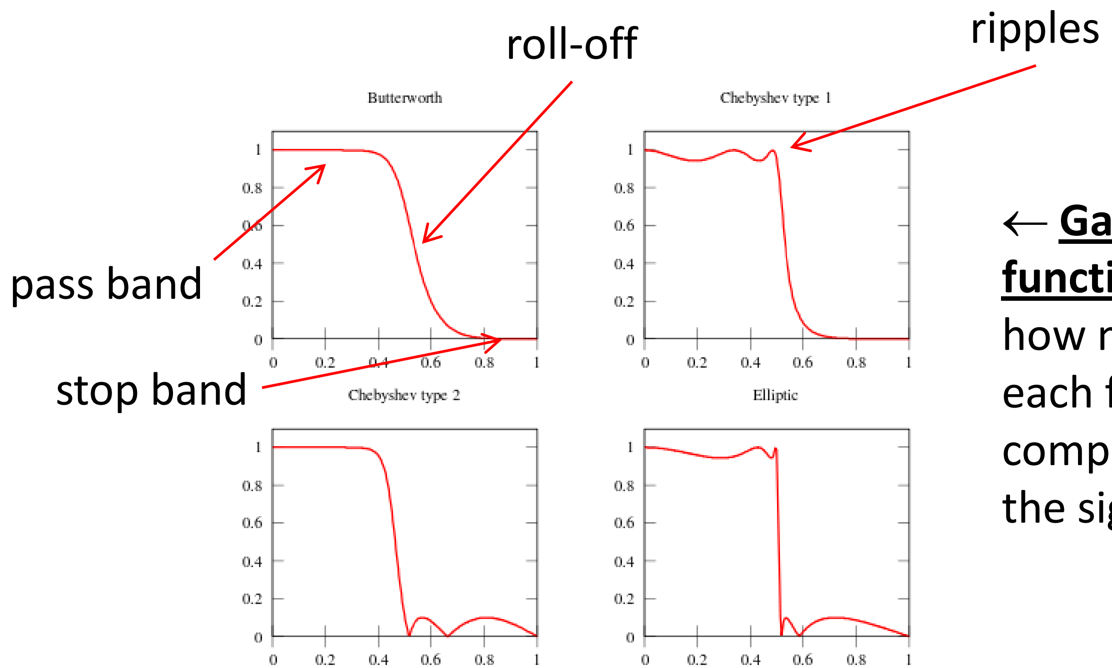
$$\Delta\Delta\, y(t) + A_1 \Delta y(t) = A_0 y(t) + Bx(t)$$

LTI systems and dynamic systems

- **Question:** Can any causal LTI system be described via a differential equation?

Low pass filters

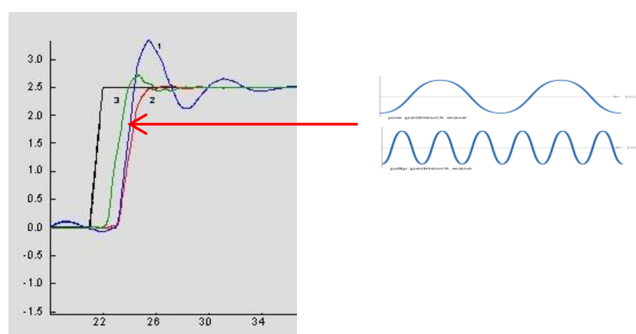
- **Butterworth** – smooth pass band, slow cutoff (roll-off)
- Chebyshev type 1 – smooth stop band, faster cutoff
- Chebyshev type 2 – smooth pass band, faster cutoff
- Bessel - no group delay, all smooth, very slow cutoff
- Gaussian – no ripples in all bands, very slow
- Elliptic – ripples in all bands, very fast



← **<u>Gain</u>** or **<u>transfer function</u>** – describes by how much strength of each frequency component changes in the signal after filtering

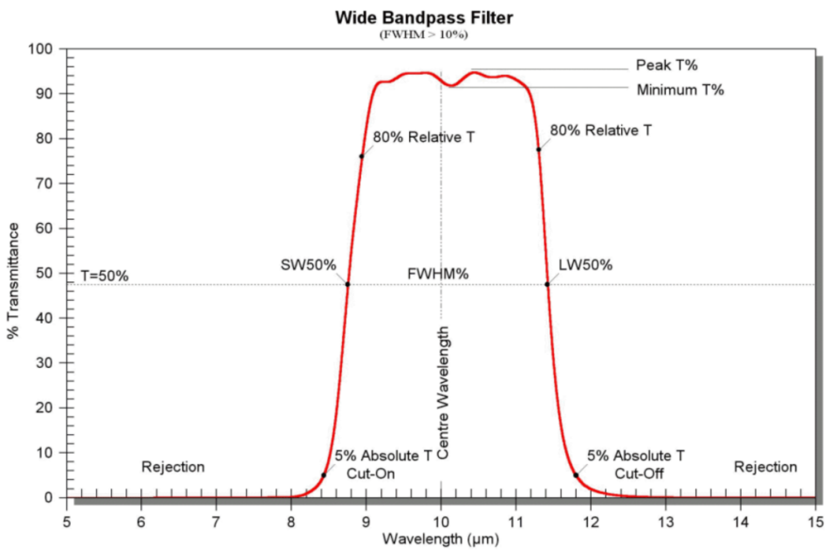
High pass digital signal filters

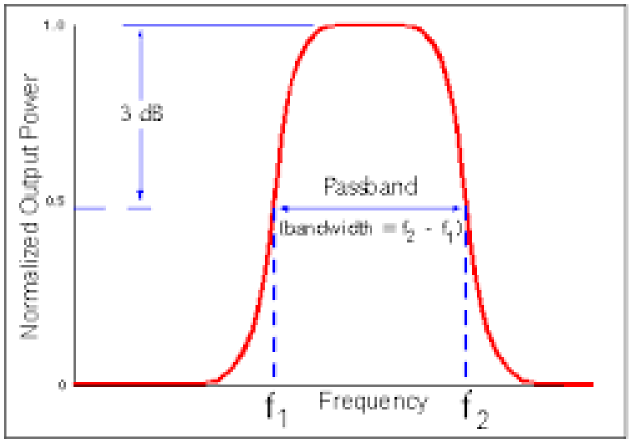
- High pass filter removes low frequency components from signal – **sharpening**
- Essentially it is a “1 – low pass” filter (that is – same design options as in the last slide)
- Sharp edges in signals and images generally are composed of very high frequencies needed to achieve fast rise! Therefore, high pass filters can be used to extract or enhance the edges.



Band pass digital signal filters

- Band pass filter allows frequency components in certain “band(s)” to pass
- Band pass filter can be represented as a low and high pass filter in series, but generally needs to be designed separately per pass-band specifications, as shown below!

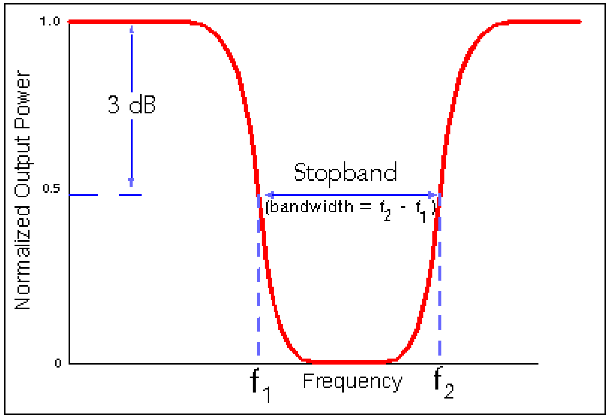




bandpass filter

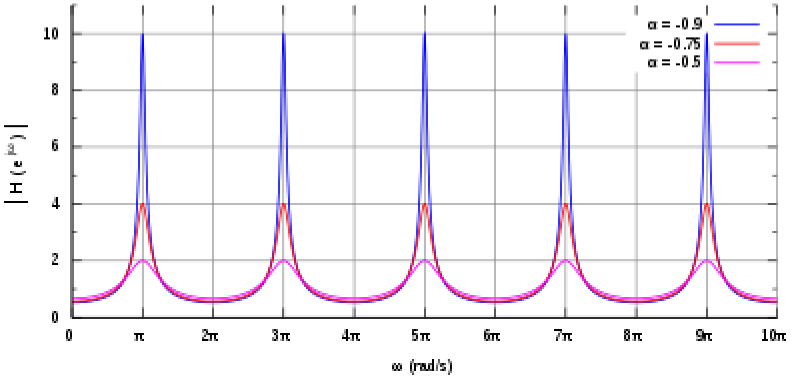
Band stop digital signal filters

- Band stop filter stops frequency components in certain “band”- exact opposite of band pass
- Band stop filters are also called **notch filters**



band-stop filter

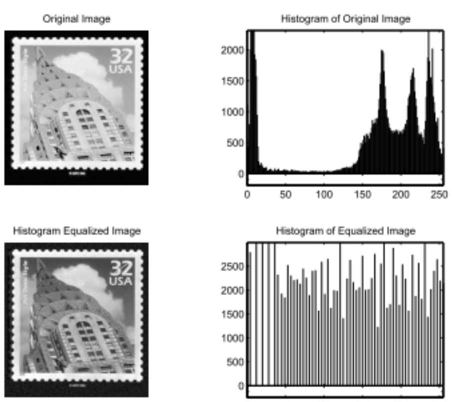
Comb filter



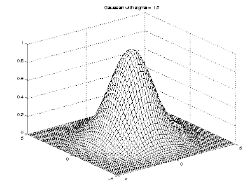
Some additional types of image filters

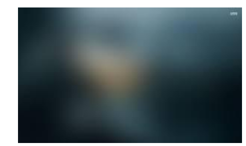
- Point transformations
  - Contrast adjustment
  - BW-thresholding
  - Complement (inversion)
  - Histogram equalization

histeq

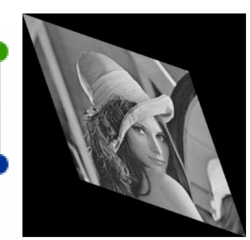


- Denoising (smoothing)
  - Average filter, median filter, Gaussian filter





- Sharpening
  - Laplacian
- Geometric transforms
  - Translation, rotation, scaling, **affine**



Some additional types of image filters

- **Morphologic filters or operations** are nonlinear transformations of images used to implement various miscellaneous transformations. Examples of morphological operations include *closing*, *opening*, *hat-opening*, *dilation*, *erosion*, and many other.