PS403 - Digital Signal processing

II. DSP - Impulse Response and Convolution

Key Text:

Digital Signal Processing with Computer Applications (2nd Ed.)

Paul A Lynn and Wolfgang Fuerst, (Publisher: John Wiley & Sons, UK)

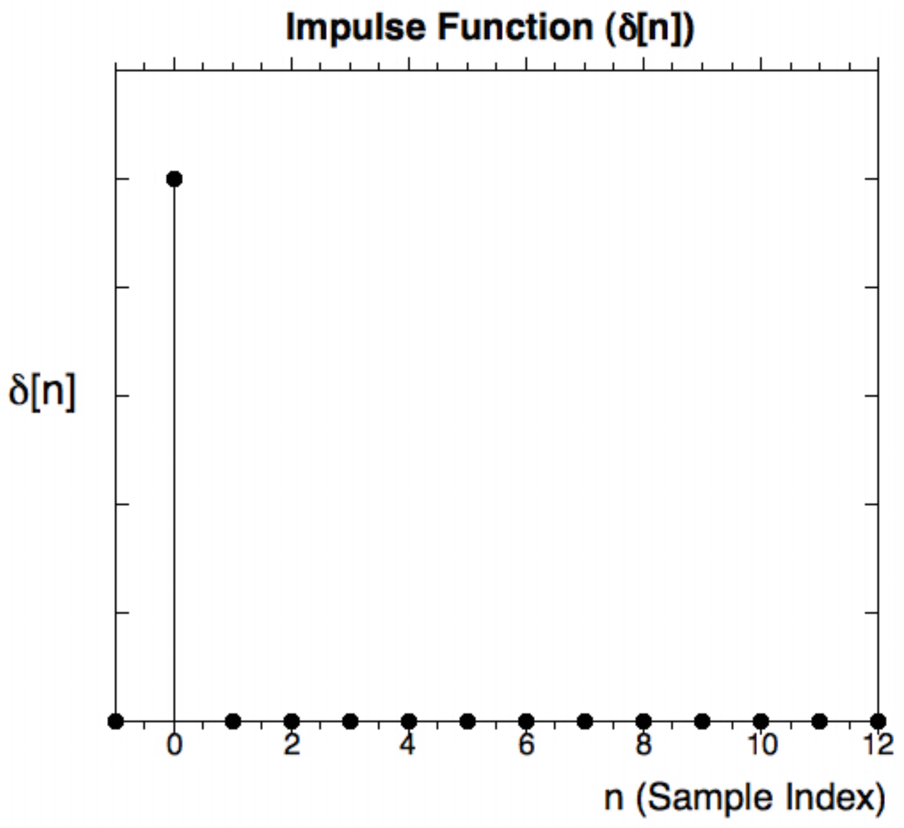
We will cover in this section

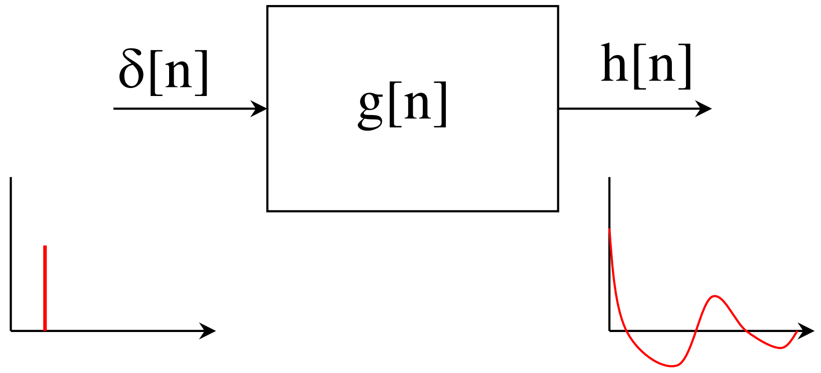
How to compute the impulse reponse h[n] of a digital system

How to use this to determine a system's reponse to any arbitrary input x[n]

How to use it to restore a signal distorted by a measurement system

Definition: The **impulse response** of a system is its natural (unforced) response - obtained by inputting an impulse into the system





In general we will be writing y[n] = f(x[n]) where y[n] is the table of outputs and x[n] the list of input values

Impulse Response

h[n] = convolution of δ[n] and g[n], h[n] = δ[n]*g[n] - *we will see later how easy this is to accomplish graphically !*

We will also show that the processor (filter/instrument) function G[n] = h[n] !

Since any arbitrary digital signal waveform can be formed by a weighted sum of impulses, i.e.,

$$x[n] = \sum_{i=1}^{k} a_i \delta[n-i]$$

as a result of the superposition property of LTI systems, the output is;

$$y[n] = \sum_{i=1}^{k} a_i h[n-i]$$

Example: Band Pass Filter.

$$y[n] = 1.5\, y[n-1] - 0.85\, y[n-2] + x[n]$$

$$\text{So } \underbrace{h[n]}_{\text{Impulse response}} = 1.5\, h[n-1] - 0.85\, h[n-2] + \underbrace{\delta[n]}_{\text{Impulse input}}$$

$h[0] = 1.5\, h[-1] - 0.85\, h[-2] + \delta[0] = 0 - 0 + 1 = 1$
$h[1] = 1.5\, h[0] - 0.85\, h[-1] + \delta[1] = 1.5 - 0 + 0 = 1.5$
$h[2] = 1.5\, h[1] - 0.85\, h[0] + \delta[2] = 2.25 - 0.85 + 0 = 1.4$
....

Continue on term by term - see Fig 2.4

h[n] is a damped oscillation at 10 samples/cycle - does that suggest anything to you ?

Step Response

Step function is a running sum of time delayed impulses

$$u[n] = \sum_{i=0}^{\infty} \delta[n-i]$$

Step response is a running sum of time delayed impulse responses

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

Also we can write: h[n] = s[n] - s[n-1]
i.e., impulse response is the 1st order difference of the step response

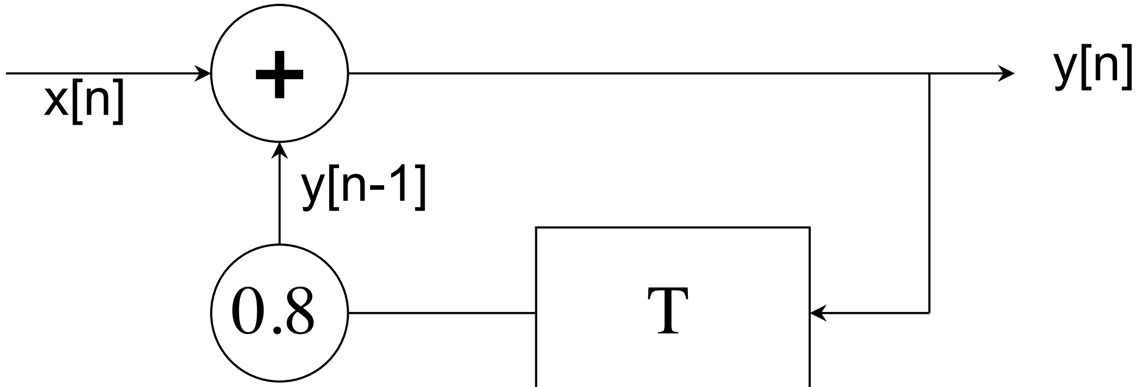
Commutative Property:

Approach 1 - $\delta[n]$ - running sum - u[n] - LTI system - s[n]

Approach 2 - $\delta[n]$ - LTI system - h[n] - running sum - s[n]

Example: y[n] = 0.8 y[n-1] + x[n]. Find h[n] and s[n]

Impulse Response



$h[0] = 0.8\, h[-1] + \delta[0] = 0 + 1 = 1$

$h[1] = 0.8\, h[0] + \delta[1] = 0.8 + 0 = 0.8$

$h[2] = 0.8\, h[1] + \delta[2] = 0.8 \times 0.8 + 0 = 0.64$

$h[3] = 0.8^3$

....

$\boxed{\textbf{h[n] = 0.8}^{n} \textbf{ - PLOT THIS !!}}$

Step Response - Approach 2

Note that:

s[0] = h[0], s[1] = h[0] + h[1], s[2] = h[0] + h[1] + h[2],..............

which can be written as:

s[0] = h[0], s[1] = s[0] + h[1], s[2] = s[1] + h[2],...... s[n]=s[n-1] + h[n]

Using, s[0] = h[0], s[1] = h[0] + h[1], s[2] = h[0] + h[1] + h[2],....

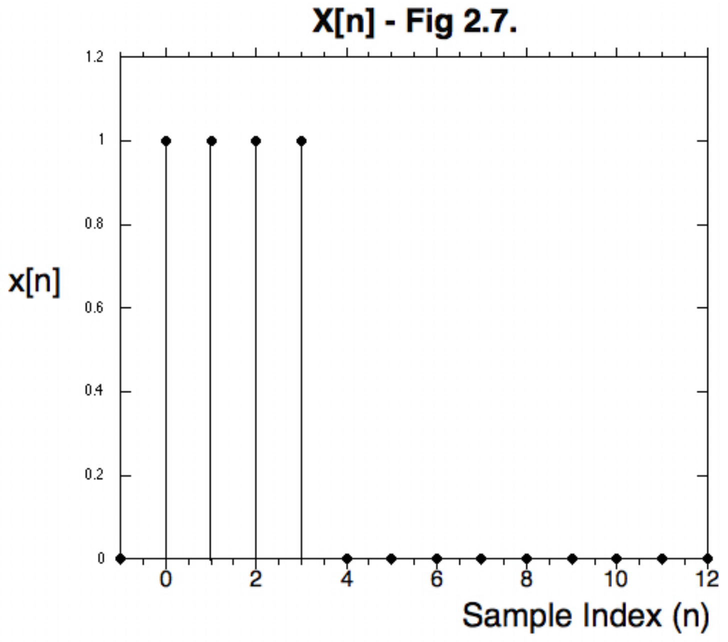
we get, s[0] = 1, s[1] = 1.8, s[2] = 2.44,..............

$s[\infty] = 1 + 0.8 + 0.8^2 + .. = 1/(1-0.8) = 5$ - PLOT THIS YOURSELVES !

See Figure 2.7, Page 40.

Response to any arbitrary input pulse x[n] - Approach 1

Consider the pulse x[n] shown in figure 2.7(b) - pulse of 4 samples duration.



So, from the superposition property, x[n] is simply 4 time shifted unit impulses and so the output can be computed by adding 4 time shifted impulse responses h[n] together

n	0	1	2	3	4	5	6
h[n]	1	0.8	0.64	0.512	0.410	0.328	0.262
h[n-1]		1	0.8	0.64	0.512	0.410	0.328
h[n-2]			1	0.8	0.64	0.512	0.410
h[n-3]				1	0.8	0.64	0.512
y[n]	1	1.8	2.44	2.952	2.362	1.889	1.571

Notice the initial step response followed by fall-off..... (Fig 2.7)

Response to any arbitrary input pulse x[n] - Using step responses

Notice also that x[n] can be formed by subtracting a step with Amplitude -1, beginning at n=4 from a step starting at n = 0.

The output y[n] is obtained by subtracting the corresponding step responses

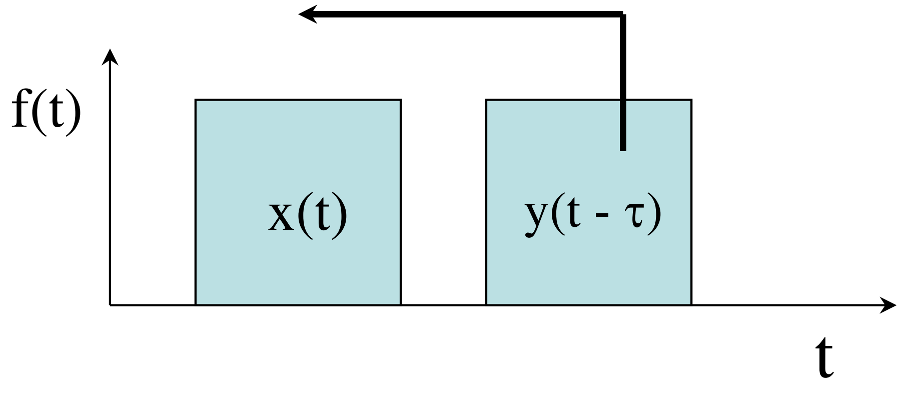
n	0	1	2	3	4	5	6
s[n]	1	1.8	2.44	2.952	3.362	3.689	3.95
s[n-4]					-1.000	-1.800	-2.44
y[n]	1	1.8	2.44	2.952	2.362	1.889	1.571

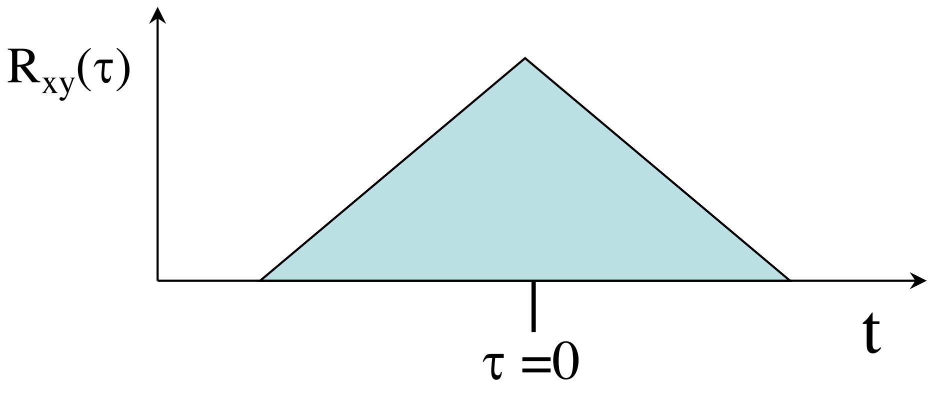
Convolution

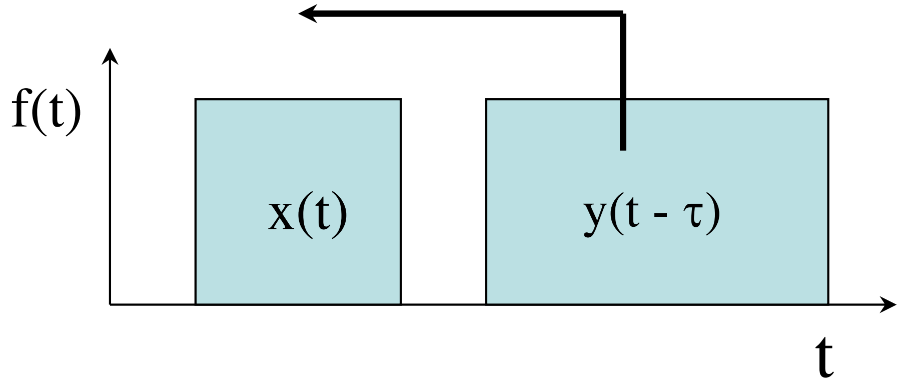
Convolution - important operation in signal (image) processing.

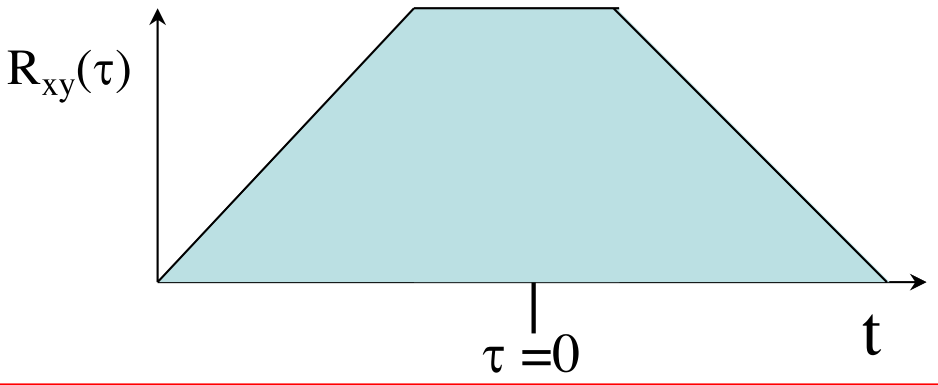
Mathematically:

$$R_{xy}(\tau) = \frac{1}{2T}\int_{-T}^{+T} x(t).y(t-\tau)dt$$









Convolution - effectively a measure of the common overlap area between two functions as you slide one over the other.

DSP 'operators' are usually convolved with a signal x[n] to produce the processed output signal y[n-1] - y[n] = x[n]*w[n]

Convolution is a fundamental operation in DSP

Digital Convolution

We know that any arbitrary input x[n] is composed of a succession of appropriately weighted impulses. The output is then a corresponding series of impulse responses weighed by the value of 'impulse' elements of the array x[n] which gave rise to it - see e.g., Fig 2.8.

So in general we can write:

y[n] = ..+ x[-2].h[n+2] + x[-1].h[n+1] +x[0].h[n] +x[1].h[n-1] +x[2].h[n-2] +....

$$y[n] = \sum_{k=-\infty}^{k=+\infty} x[-k].h[n+k] = \sum_{k=-\infty}^{k=+\infty} x[k].h[n-k]$$

Digital Convolution Sum

$$\boxed{y[n] = x[n]*h[n]}$$

Digital Convolution

Notice that each impulse response is weighted by the value of x[n] that give rise to it and shifted to begin at the correct instant. y[n] is then found by superimposing the individual impulse responses.

If you look at it graphically - you sweep the impulse response h[n] past the input x[n] one sampling interval at a time. At each interval you cross-multiply the two waveforms (matrices or list) and add up all the product values (integration).

Very powerful technique - you can determine impulse response, perform the digital convolution and thereby ultimately compute the output y[n] for any arbitrary input x[n]……………..

Digital Convolution

Example: Savitsky Golay (multi-point averaging) filter

Used to smooth noisy data. We take the case of a 5 point adjacent channel averaging action - see Fig 2.11.

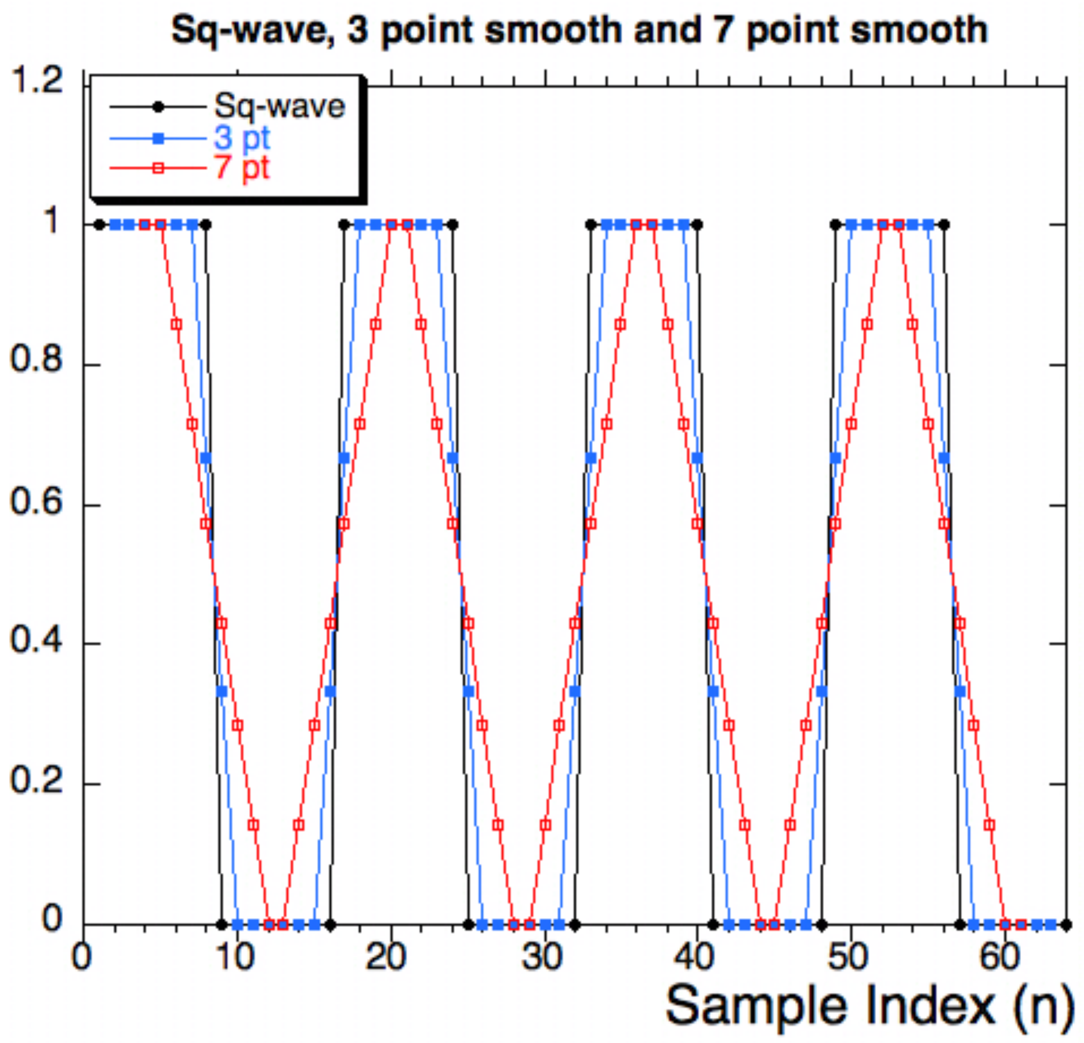
$$y[n] = 0.2 \{x[n+2] + x[n+1] + x[n] + x[n-1] + x[n-2]\}$$

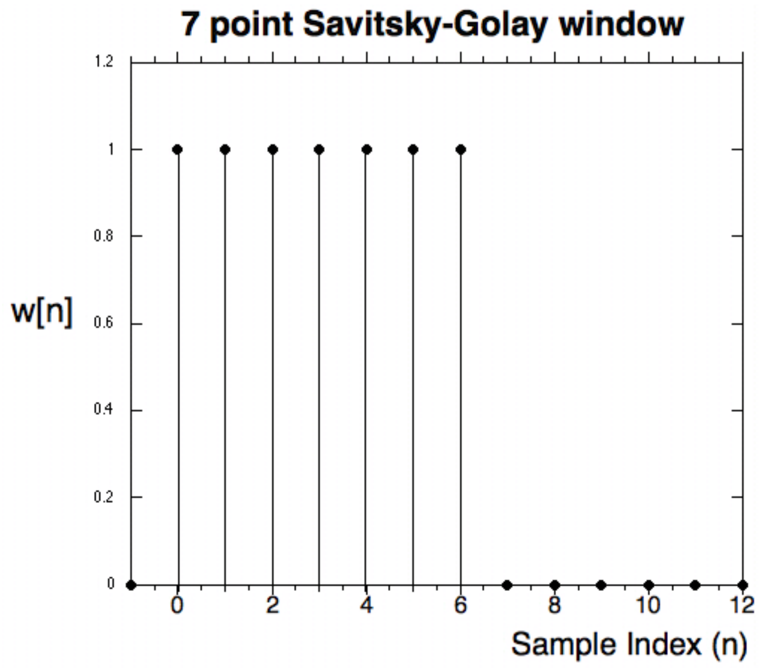
Referring to figure 2.11, the input signal is multiplied by the weighting function and all five finite product values are summed to produce one value of the output signal (matrix/list). The weighting function (filter function) is time shifted by one sampling interval and the process (cross multiplication and summation) repeated - SOUND FAMILIAR ?

Of course the process is just digital convolution but with the impulse response replaced by the filter (weighting function)

Digital Convolution

Example: **Square wave of 16 samples per cycle.**
3 and 7 point average on 64 point sample set





Digital Convolution

$$y[n] = \sum_{k=-3}^{k=+3} x[k].w[n-k]$$

7 point SG smoothing operation expressed as a convolution sum

Hence to smooth any signal x[n] using SG Golay ' m+1' point averaging, m = 2, 3, 4......, you simply pass it through an LTI processor with an impulse response h[n] exactly = w[n]

Notice that the filter depends on 'future values' of x[n], i.e., y[n] depends on x[n+1], x[n+2] & x[n+3], *ergo it is not causal !*

Recursive version of 5 point SG smooth: y[n]=y[n-1] + 0.2{x[n+2] - x[n-3]}
Write out the corresponding 7 point filter function.

Digital Convolution - Moving Average Cont'd

Consider the following input signal:

$$x[n] = Sin\left(\frac{2\pi n}{60}\right) + Sin\left(\frac{2\pi n}{10}\right), 60 \le n \le 320$$

Sum of two sinusoids with 'frequencies' of:
60 samples/cycle and 10 samples/cycle

You build a simple digital convolution code which has as input any arbitrary signal array x[n] which is convolved arbitrary filter array h[n].

Choose a 10 term SG moving average filter:

w[n] = h[n] = 0.1, $0 \le n \le 9$
h[n] = 0.0 elsewhere

Run the code to evaluate y[n] = x[n] * h[n] - what does y[n] look like ?

Digital Convolution - Moving Average Cont'd

See Fig 2.12 !!

Digital Convolution - Moving Average Cont'd

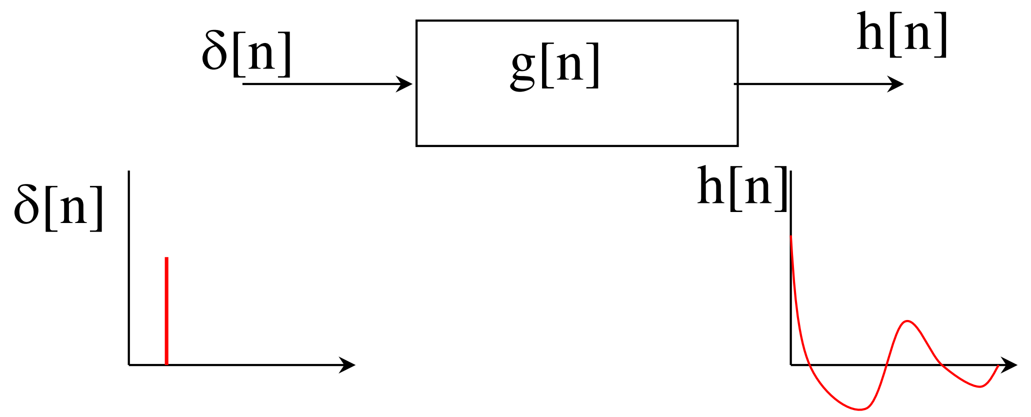
x[n] has 2 frequency components:

@ 10 samples/cycle - Sin($2\pi n/10$) - high 'frequency'
@ 60 samples/cycle - Sin($2\pi n/60$) - low 'frequency'

Since the smoothing filter w[n] averages over exactly 10 samples, it eliminates the sinusoidal component @ 10 samples/cycle - you add the 5 positively valued samples over the positive 1/2 cycle to the 5 negatively valued samples over the succeeding negative 1/2 cycle !

Notice the switching transient at the beginning of y[n] in figure 2.12.

Signal Restoration - Deconvolution - Brief



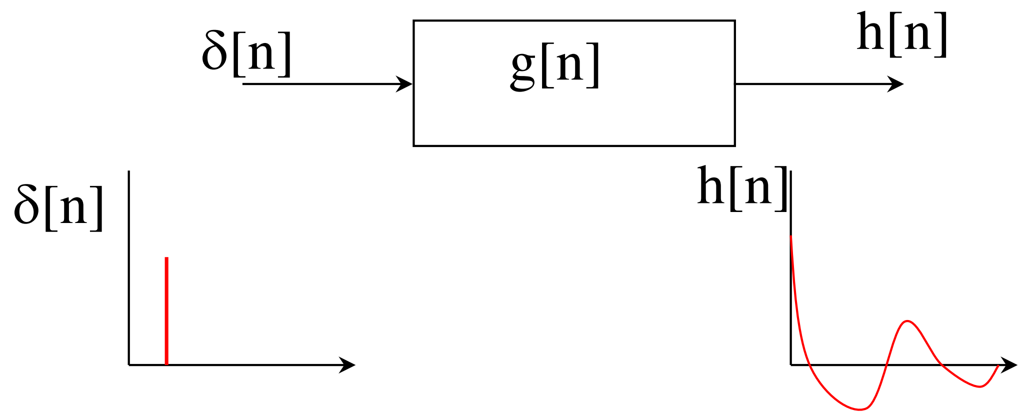
The principle is simple. Every measurement system distorts the measurand. Say the undistorted signal (measurand) is x[n] - but the measuring instrument (circuit, oscilloscope, etc.) actually gives y[n] - the distorted signal.

So y[n] = x[n] * g[n], where g[n] is the instrument function.

But we know that g[n] = h[n] = instrument impulse response -

So (for any measuring system):
The instrument function is the impulse response !

Signal Restoration - Deconvolution - Cont'd



Since: distorted signal = instrument function * undistorted measurand,

$$y[n] = h[n] * x[n]$$

we can obtain x[n] (restore the measurand) by carrying out the inverse operation -

this is referred to as 'deconvolution' - a hugely powerful technique……

$$\boxed{x[n] = h^{-1}[n] * y[n]}$$

Signal Restoration - Deconvolution - Cont'd

So, since h[n] embodies the information on how the system distorted the measurand, we can use it to restore the signal to close to its original form.

The process can be generalised to 2D for images.

This is not a panacea for all (ill) distorted signals !!

As we shall see in the next section you can work in the complementary frequency domain where we will see how a signal is distorted at each frequency - *so the instrument function is effectively a 'frequency transfer operator'* and so the 'deconvolution operator' will become the the inverse of the 'frequency transfer operator or function'.

In general, an instrument will smooth off the sharp edges of a signal - i.e., act like a low pass filter. So the deconvolution operator will have high gain at high frequency. The problem will be that real signals have noise superimposed which will also be preferentially amplified in deconvolution. So we will need to prefilter noise and so the deconvolution operator (in time or frequency domains) will not simply by the inverse of the inst. function !!