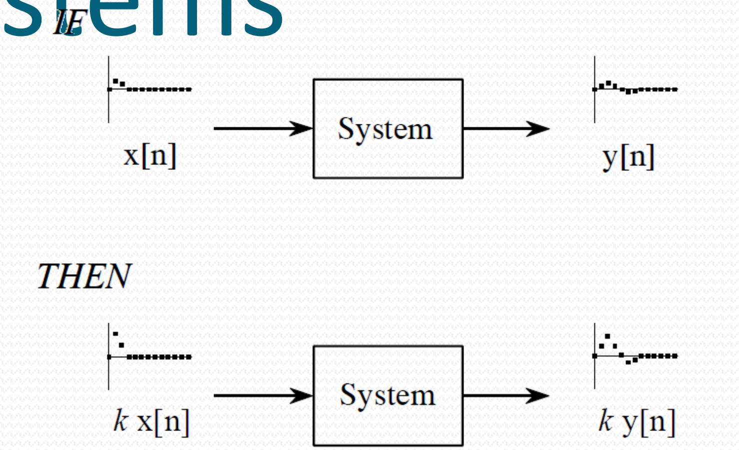
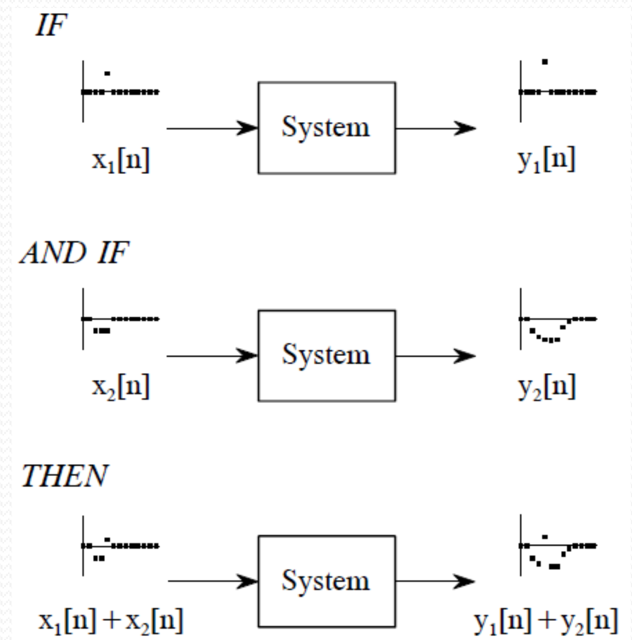
EE327 Digital Signal Processing Convolution

Yasser F. O. Mohammad

REMINDER 1:Linear Systems



- Linear = Homogeneous+Additive
- Homogeneity
  - If $X[n] \rightarrow Y[n]$
    then $k X[n] \rightarrow k Y[n]$
- Additive
  - If $X_1[n] \rightarrow Y_1[n]$ and $X_2[n] \rightarrow Y_2[n]$
    then $X_1[n]+X_2[n] \rightarrow Y_1[n]+Y_2[n]$



- Most DSP linear systems are also *shift invariant* (LTI)

REMINDER 2: Sinusoidal Fidelity

- Linear system → sinusoidal output for sinusoidal input
- Sinusoidal Fidelity $\not\rightarrow$ Linear System
  - (e.g. phase Lock Loop)

- This is why we can work with AC circuits using only two numbers (amplitude and phase)

- This is why Fourier Analysis is important

- This is partially why Linear Systems are important

- This is why you cannot see DSP without *sin*

REMINDER 3: Fundamental Concept of DSP

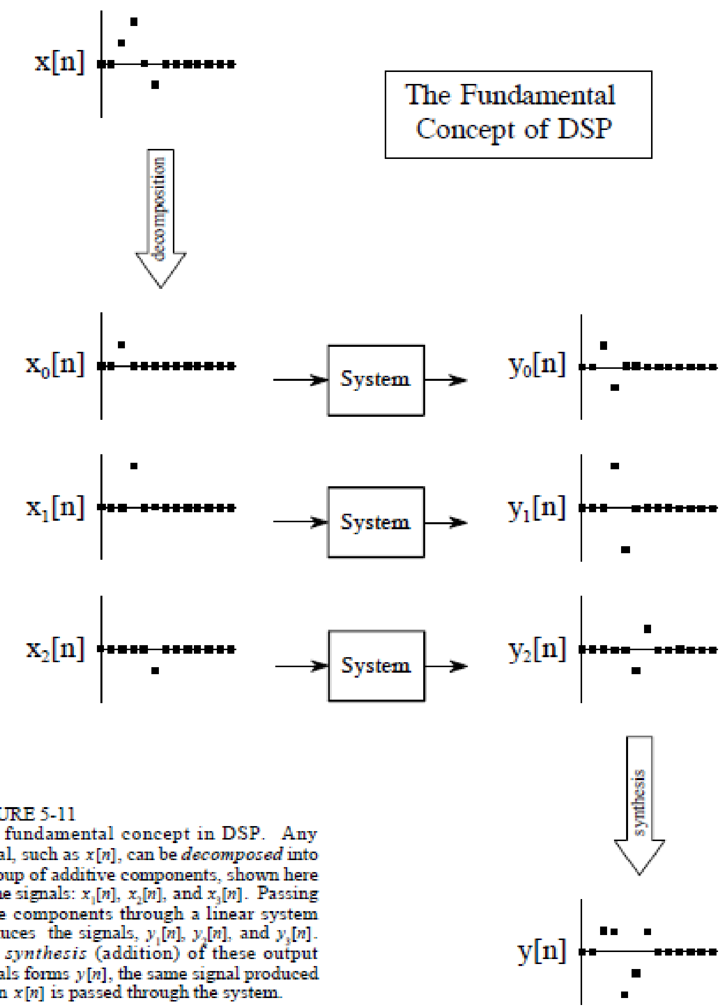
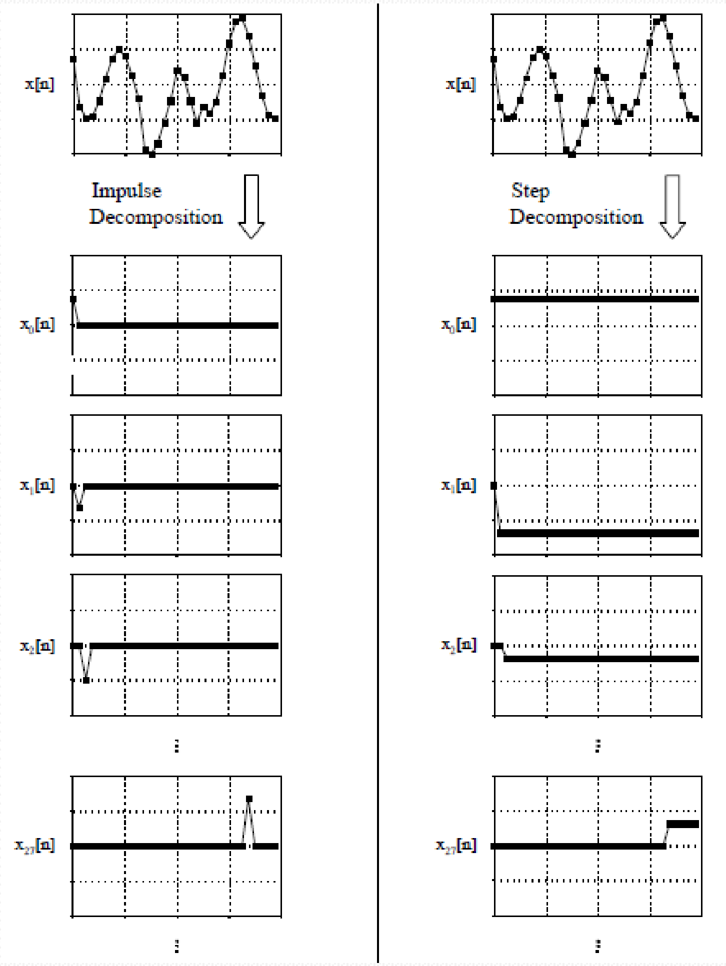


FIGURE 5-11
The fundamental concept in DSP. Any signal, such as $x[n]$, can be *decomposed* into a group of additive components, shown here by the signals: $x_1[n]$, $x_2[n]$, and $x_3[n]$. Passing these components through a linear system produces the signals, $y_1[n]$, $y_2[n]$, and $y_3[n]$. The *synthesis* (addition) of these output signals forms $y[n]$, the same signal produced when $x[n]$ is passed through the system.

REMINDER 4: Impulse and Step Decompositions



What is convolution?

- A mathematical operation that takes two signals and produces a third one.

  - X[n]*Y[n]=Z[n]

- For us:
  - A way to get the output signal given the input signal and a representation of system function

From now one we will deal only with discrete signals if not otherwise specified

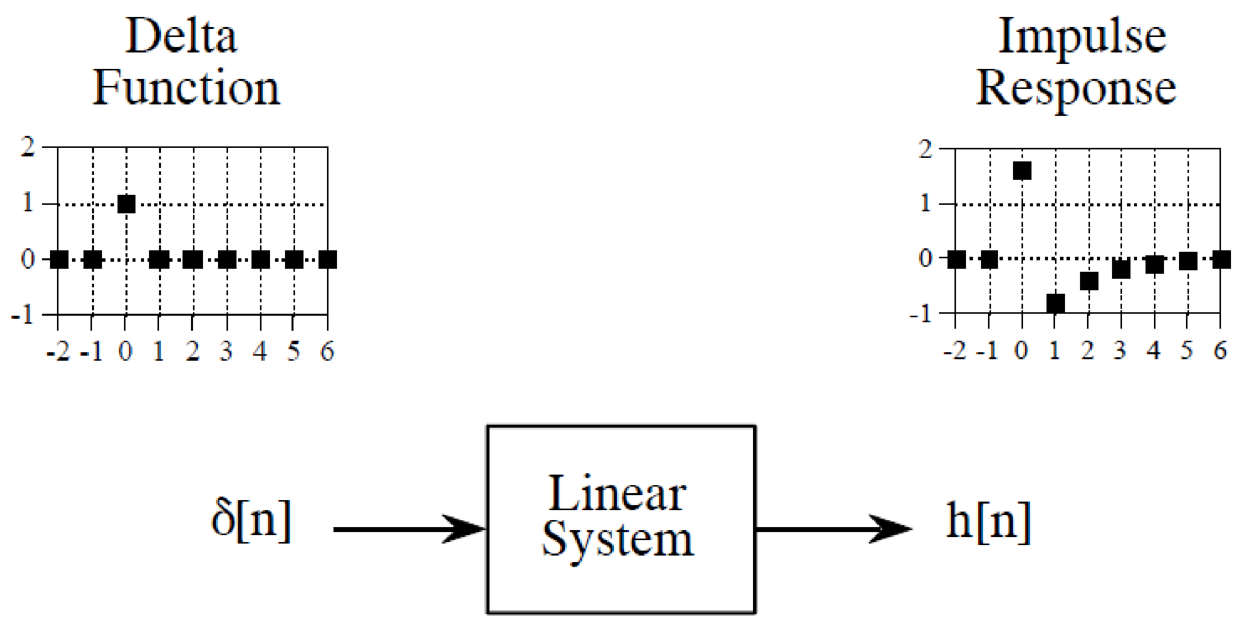
Delta function

- Delta function=Unit impulse = δ[n]

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & otherwise \end{cases}$$

Impulse Response

- Describes a SYSTEM not a signal
  - We use *h[n]* for it
- Gives the output signal if the input to the system was a unit impulse



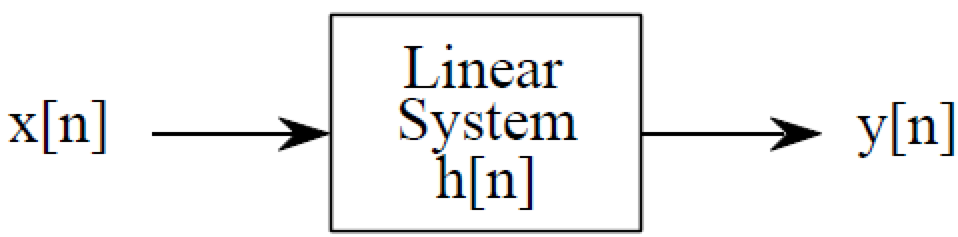
Other names of impulse response

- Filters
  - Filter Kernel
  - Kernel
  - Convolution Kernel

- Image processing
  - Point Spread Function

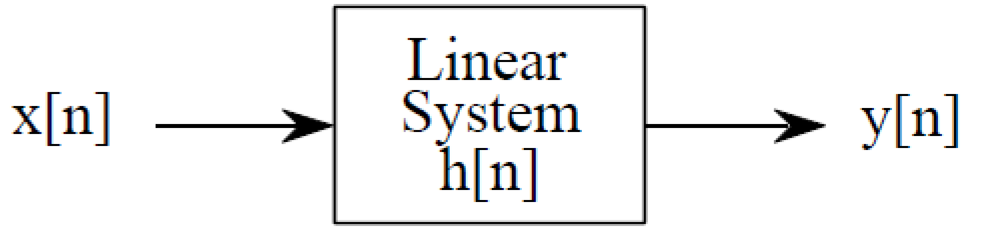
Why impulse response is important?

- It COMPLETELY describes systems FUNCTION
  - Any input can be decomposed into an impulse train
  - Linearity → Superposition → Any input
  - [Usually] Shift invariance → Any time



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

How to calculate the output



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

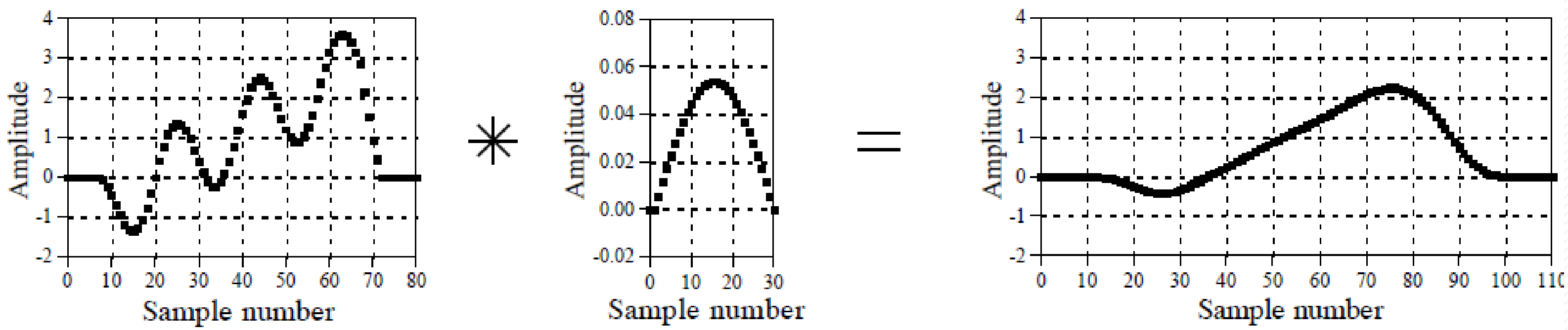
- Input length = N
- Impulse Response length = M

- Output length = N+M-1

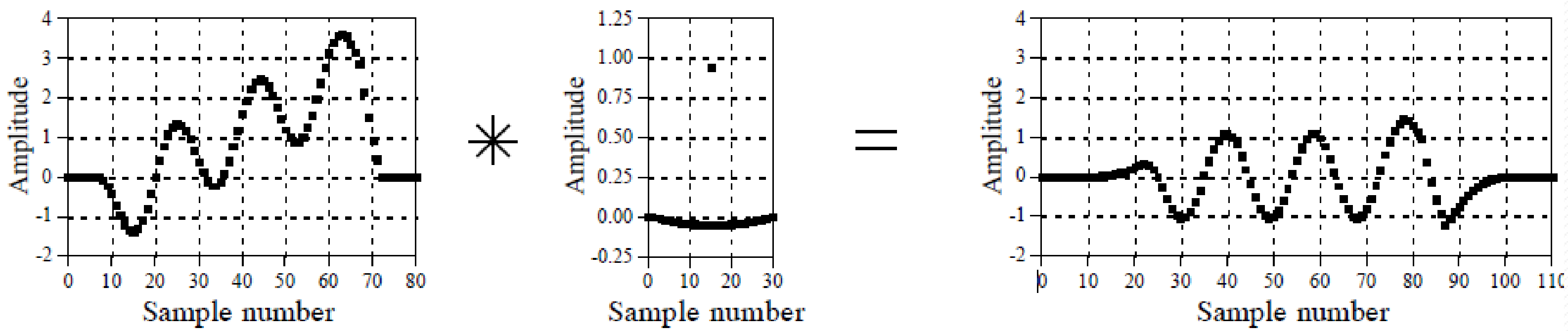
- For example a 81 points input convolved with a 31 points impulse response gives 111 points output

Examples

a. Low-pass Filter

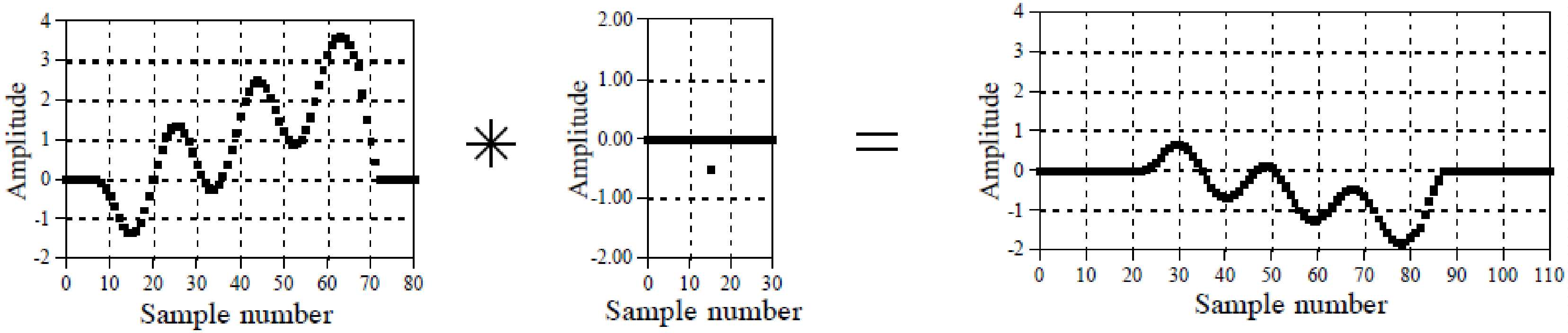


b. High-pass Filter

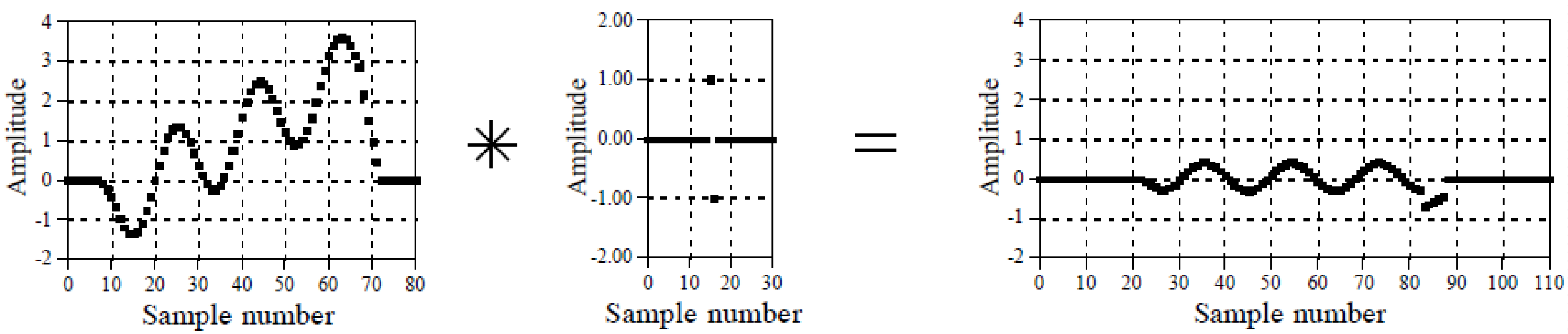


More Examples

a. Inverting Attenuator



b. Discrete Derivative



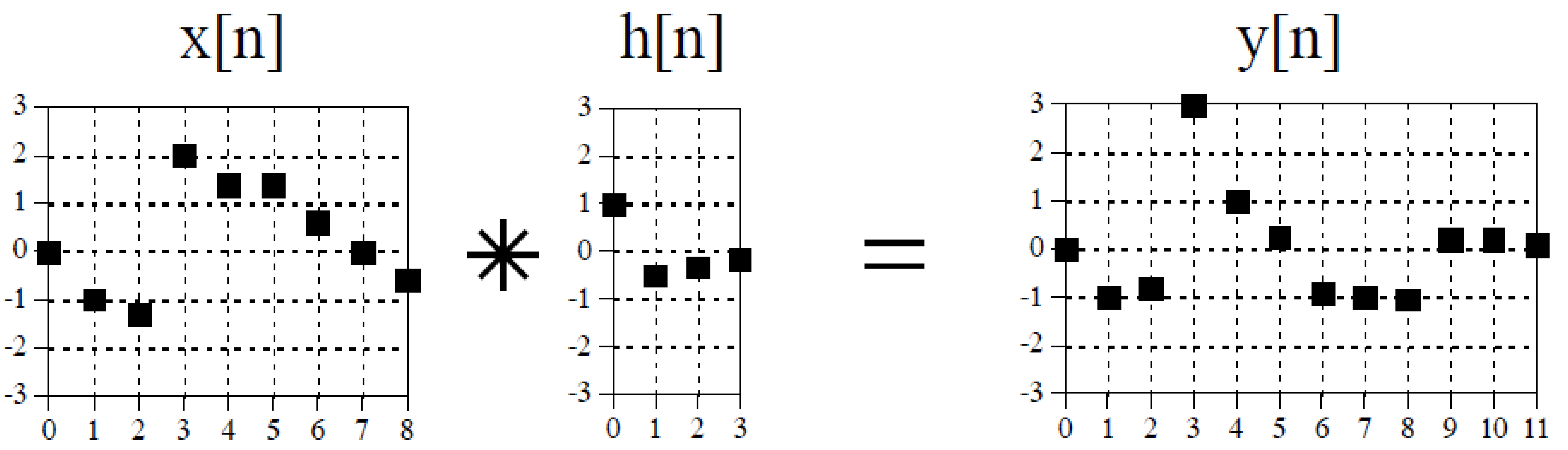
Two ways to understand it

- Input Signal Viewpoint *(Input Side Algorithm)*
  - How each input impulse contributes to the output signal.
  - Good for your understanding

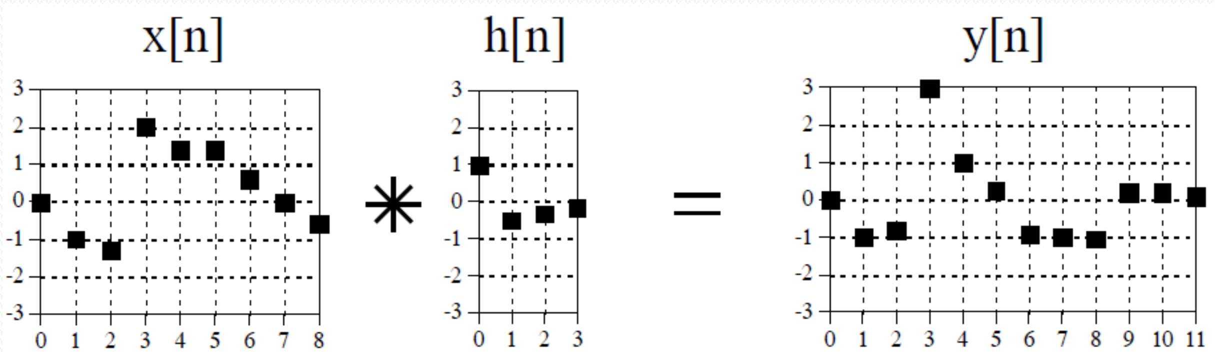
- Output Signal Viewpoint *(Output Side Algorithm)*
  - How each output impulse is calculated from input signal.
  - Good for your calculator

Input Side Algorithm

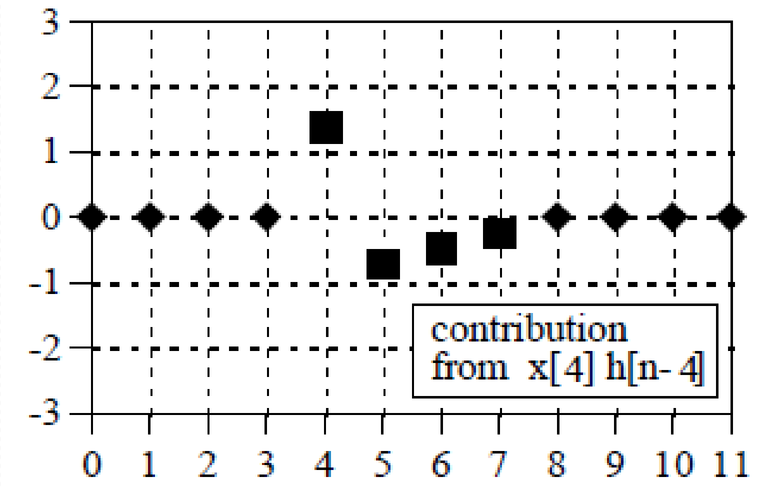
- Each sample is considered a scaled impulse
- Each scaled impulse results in a scaled impulse response
- Add all scaled impulse responses together



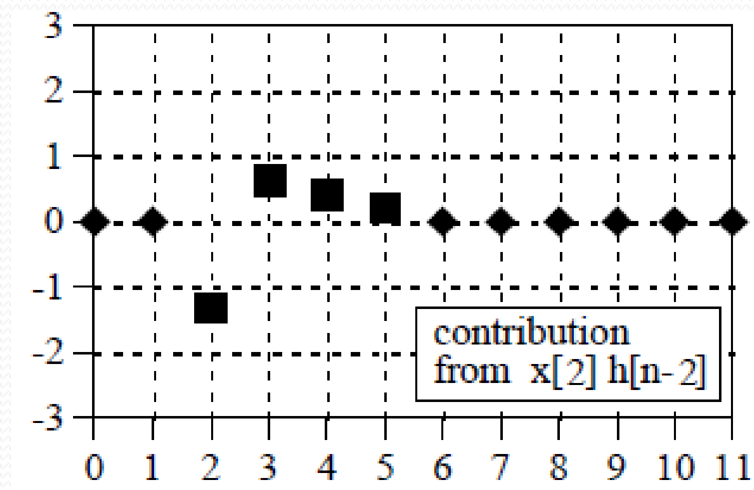
Example Input Side Algorithm



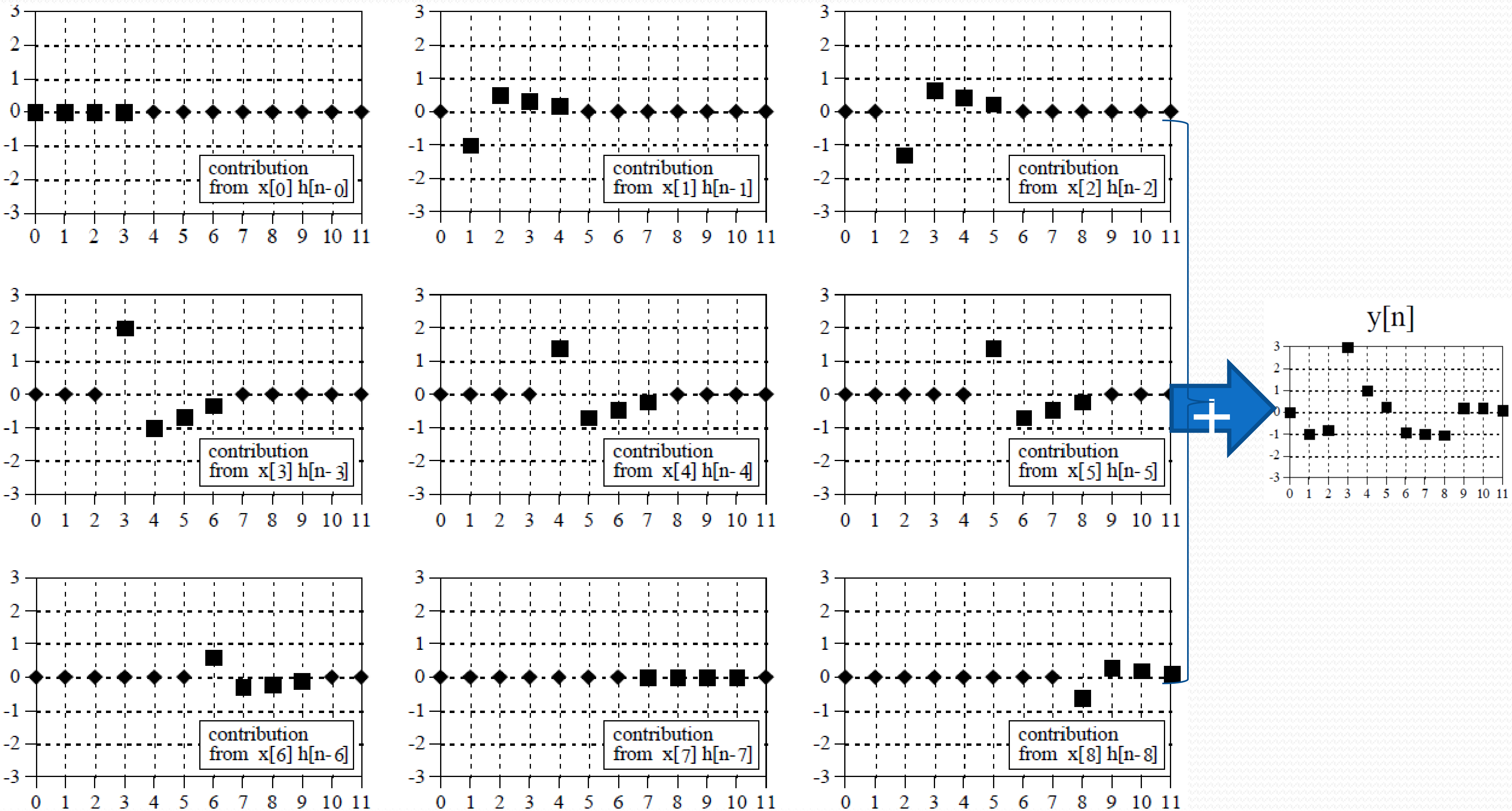
n=4



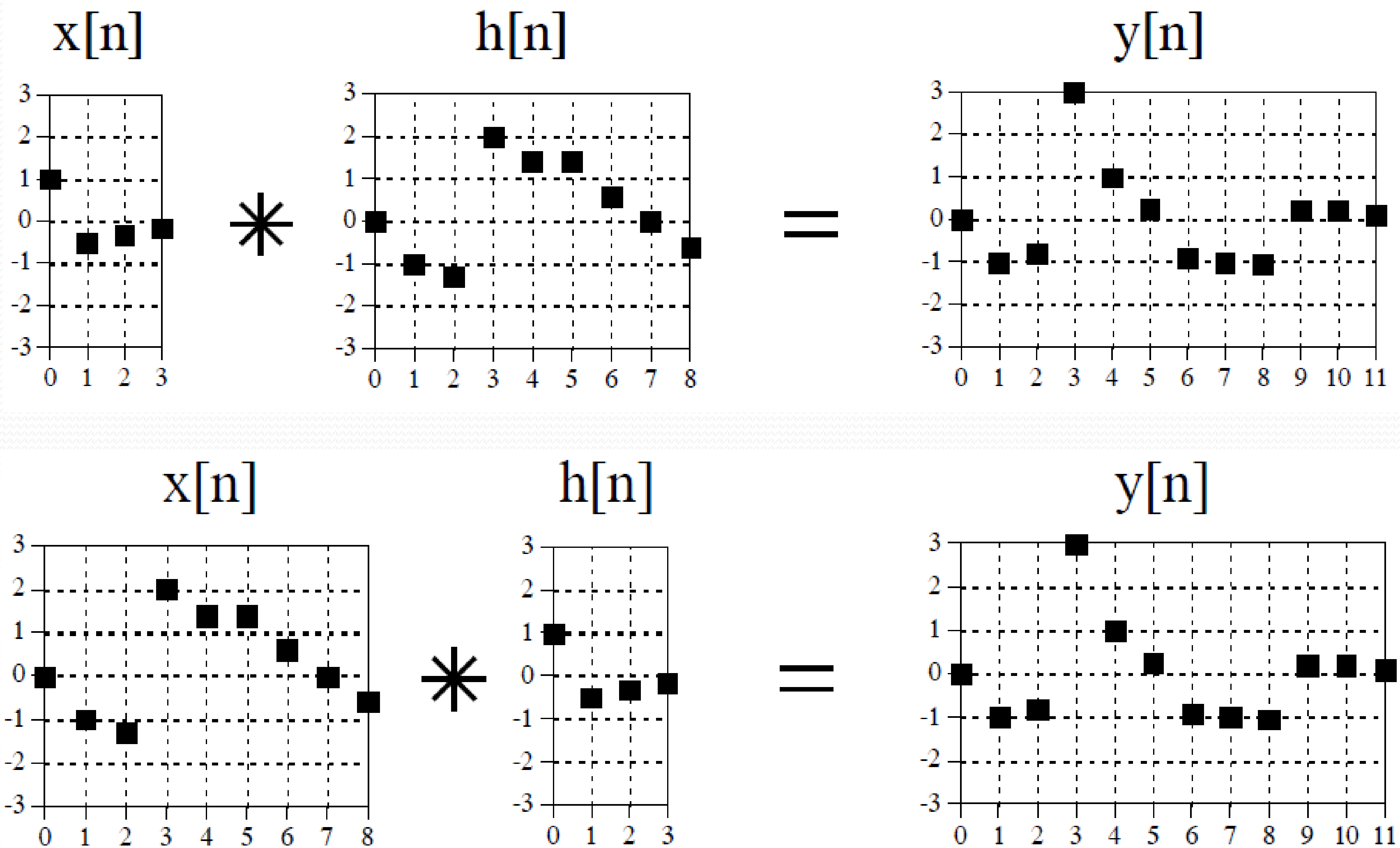
n=2



The nine responses=Total Response



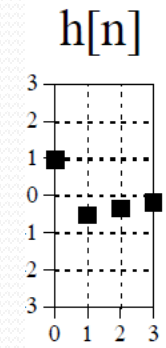
X[n]*h[n]=h[n]*X[n]

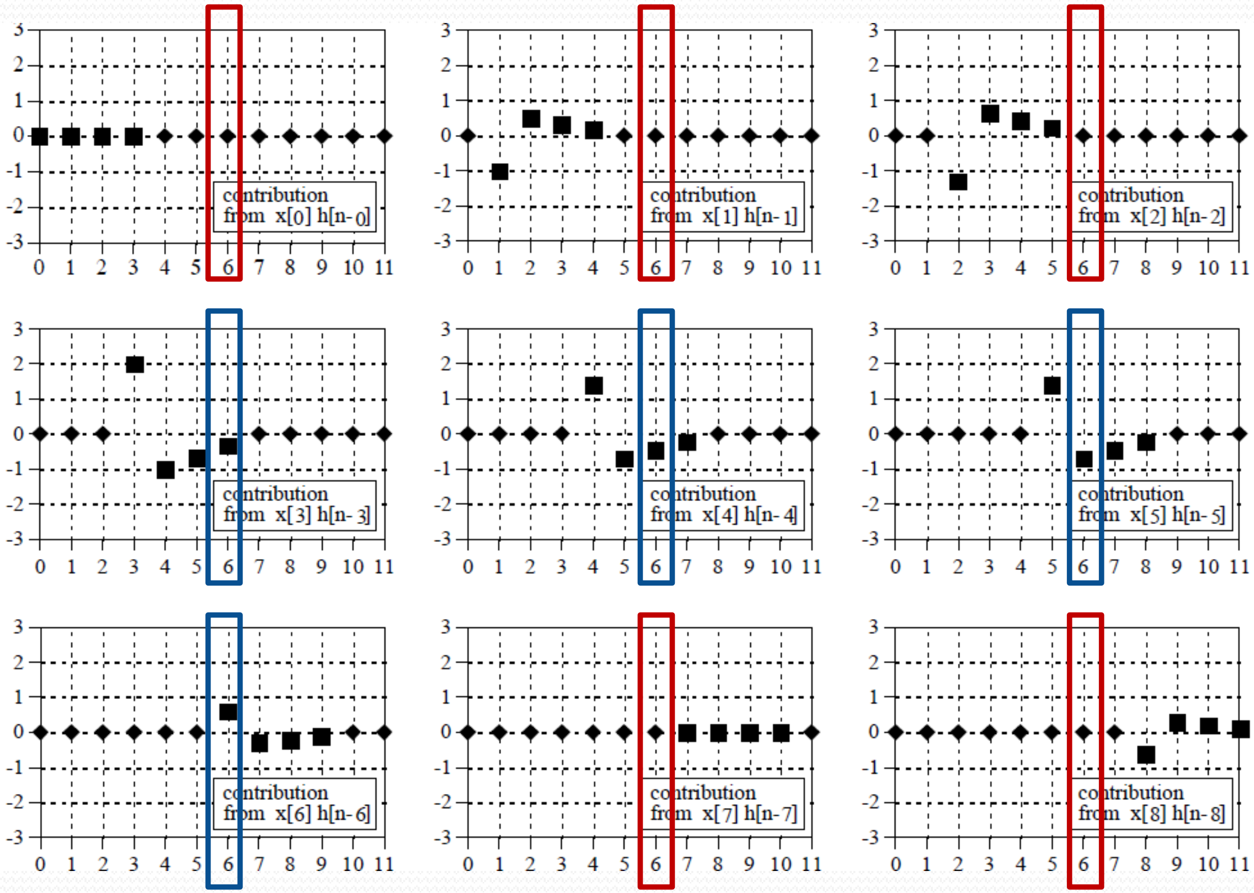


Input Side Algorithm

```
100 'CONVOLUTION USING THE INPUT SIDE ALGORITHM
110                                 '
120 DIM X[80]                       'The input signal, 81 points
130 DIM H[30]                       'The impulse response, 31 points
140 DIM Y[110]                      'The output signal, 111 points
150                                 '
160 GOSUB XXXX                      'Mythical subroutine to load X[ ] and H[ ]
170                                 '
180 FOR I% = 0 TO 110               'Zero the output array
190   Y(I%) = 0
200 NEXT I%
210                                 '
220 FOR I% = 0 TO 80                'Loop for each point in X[ ]
230   FOR J% = 0 TO 30              'Loop for each point in H[ ]
240     Y[I%+J%] = Y[I%+J%] + X[I%]*H[J%]
250   NEXT J%
260 NEXT I%                         '(remember, * is multiplication in programs!)
270                                 '
280 GOSUB XXXX                      'Mythical subroutine to store Y[ ]
290                                 '
300 END
```

Calculating a single output point





Output 6 is affected by the response to the following inputs (blue):

$x[3]\times h[3]$,
$x[4]\times h[2]$,
$x[5]\times h[1]$,
$x[6]\times h[0]$

This is true for ANY point

Output sample *j* is calculated As:

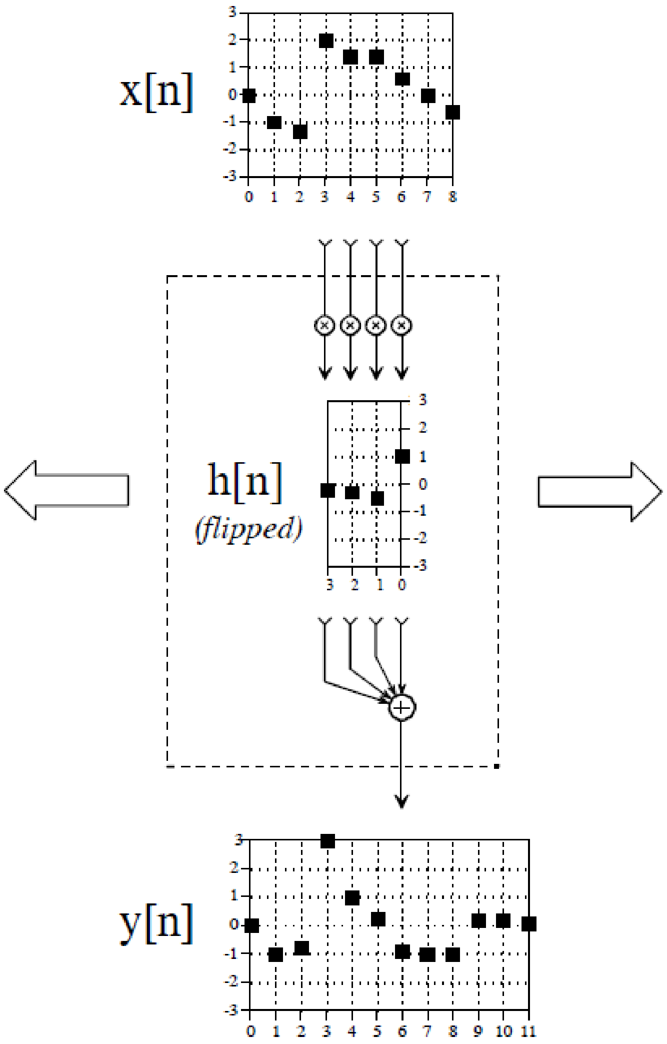
$$y[j]=\sum_{i=0}^{M-1} x[j-i]h[i]$$

General Output Side Flowchart

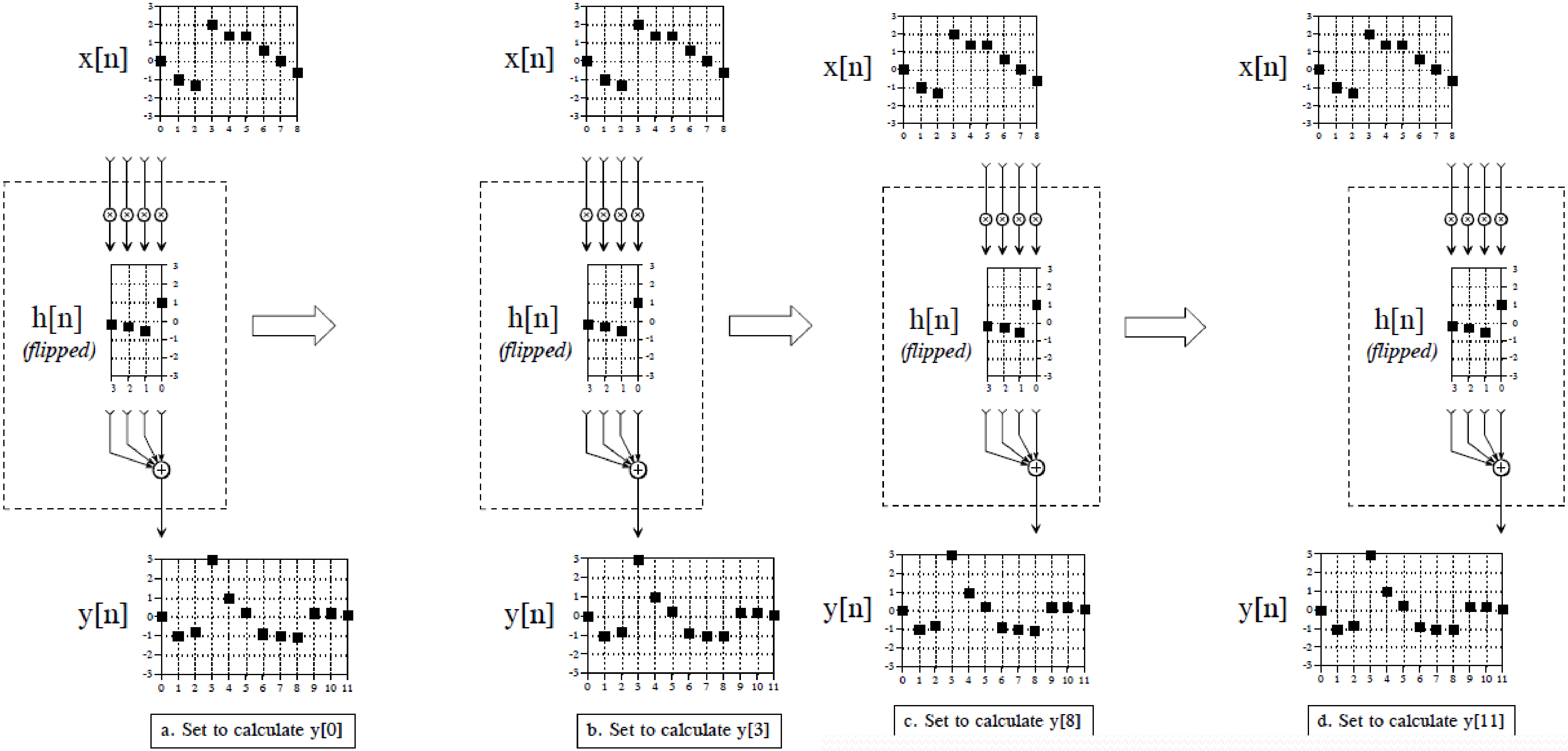
- Flip the second signal (h[n])
- Move it over the first signal (x[n])
- Each time calculate:

$$y[j] = \sum_{i=0}^{M-1} x[j-i]h[i]$$

- Continue until first signal is finished

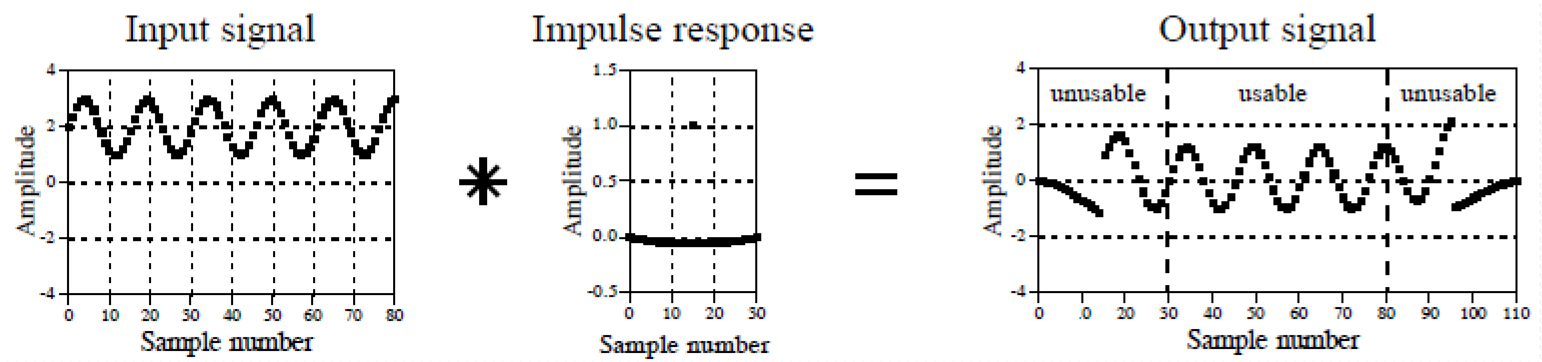


Example Output Side Algorithm



Boundary Effect

- At the first and last M-1 points the impulse response is not fully immersed into the signal
- These points are unreliable



Output Side Algorithm

```
100  'CONVOLUTION USING THE OUTPUT SIDE ALGORITHM
110                                    '
120 DIM X[80]                          'The input signal, 81 points
130 DIM H[30]                          'The impulse response, 31 points
140 DIM Y[110]                         'The output signal, 111 points
150                                    '
160 GOSUB XXXX                         'Mythical subroutine to load X[ ] and H[ ]
170                                    '
180 FOR I% = 0 TO 110                  'Loop for each point in Y[ ]
190   Y[I%] = 0                        'Zero the sample in the output array
200   FOR J% = 0 TO 30                 'Loop for each point in H[ ]
210     IF (I%-J% < 0)                 THEN GOTO 240
220     IF (I%-J% > 80)                THEN GOTO 240
230     Y(I%) = Y(I%) + H(J%) * X(I%-J%)
240   NEXT J%
250 NEXT I%
260                                    '
270 GOSUB XXXX                         'Mythical subroutine to store Y[ ]
280                                    '
290 END
```