Discrete-Time Signals: Time-Domain Representation

- Signals represented as sequences of numbers, called **samples**
- Sample value of a typical signal or sequence denoted as $x[n]$ with n being an integer in the range $-\infty \leq n \leq \infty$
- $x[n]$ defined only for integer values of n and undefined for noninteger values of n
- Discrete-time signal represented by $\{x[n]\}$

Discrete-Time Signals: Time-Domain Representation

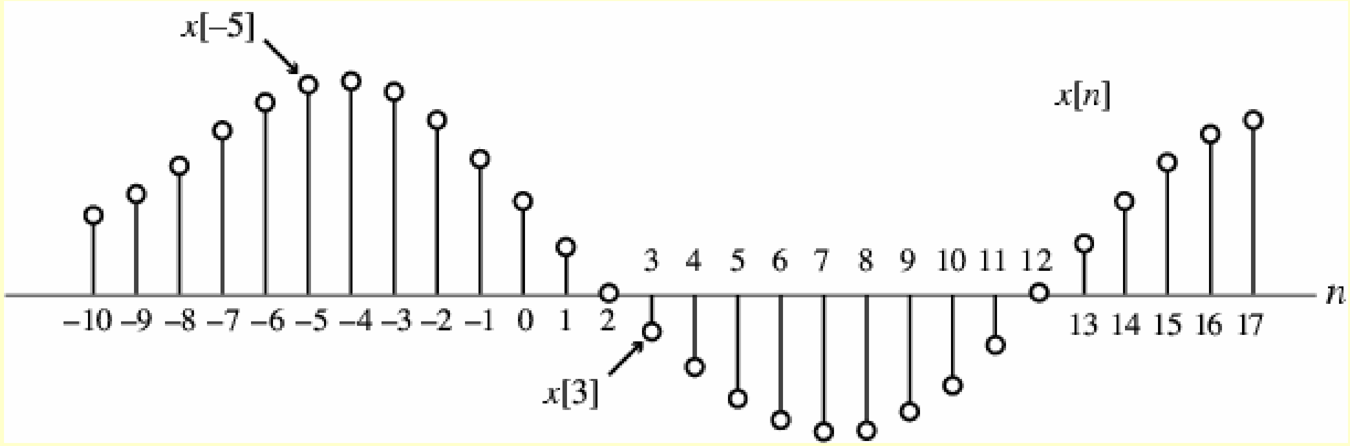
- Discrete-time signal may also be written as a sequence of numbers inside braces:

$$\{x[n]\} = \{\ldots, -0.2, \underset{\uparrow}{2.2}, 1.1, 0.2, -3.7, 2.9, \ldots\}$$

- In the above, $x[-1] = -0.2$, $x[0] = 2.2$, $x[1] = 1.1$, etc.
- The arrow is placed under the sample at time index $n = 0$

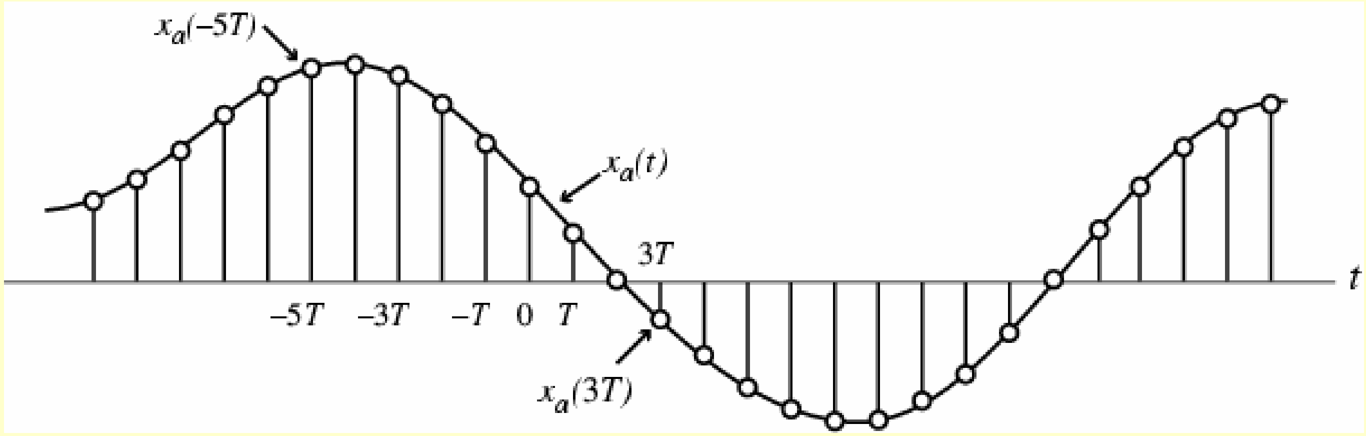
Discrete-Time Signals: Time-Domain Representation

- Graphical representation of a discrete-time signal with real-valued samples is as shown below:



Discrete-Time Signals: Time-Domain Representation

- In some applications, a discrete-time sequence $\{x[n]\}$ may be generated by periodically sampling a continuous-time signal $x_a(t)$ at uniform intervals of time



Discrete-Time Signals: Time-Domain Representation

- Here, n-th sample is given by

$$x[n] = x_a(t)\big|_{t=nT} = x_a(nT),\ n = \ldots, -2, -1, 0, 1, \ldots$$

- The spacing T between two consecutive samples is called the **sampling interval** or **sampling period**

- Reciprocal of sampling interval T, denoted as F_T, is called the **sampling frequency**:

$$F_T = \frac{1}{T}$$

Discrete-Time Signals: Time-Domain Representation

- Unit of sampling frequency is cycles per second, or **hertz** (Hz), if T is in seconds
- Whether or not the sequence $\{x[n]\}$ has been obtained by sampling, the quantity $x[n]$ is called the n**-th sample** of the sequence
- $\{x[n]\}$ is a **real sequence**, if the n-th sample $x[n]$ is real for all values of n
- Otherwise, $\{x[n]\}$ is a **complex sequence**

Discrete-Time Signals: Time-Domain Representation

- A complex sequence $\{x[n]\}$ can be written as $\{x[n]\} = \{x_{re}[n]\} + j\{x_{im}[n]\}$ where $x_{re}[n]$ and $x_{im}[n]$ are the real and imaginary parts of $x[n]$
- The complex conjugate sequence of $\{x[n]\}$ is given by $\{x^*[n]\} = \{x_{re}[n]\} - j\{x_{im}[n]\}$
- Often the braces are ignored to denote a sequence if there is no ambiguity

Discrete-Time Signals: Time-Domain Representation

- Example - $\{x[n]\} = \{\cos 0.25n\}$ is a real sequence
- $\{y[n]\} = \{e^{j0.3n}\}$ is a complex sequence
- We can write

$$\{y[n]\} = \{\cos 0.3n + j\sin 0.3n\}$$

$$= \{\cos 0.3n\} + j\{\sin 0.3n\}$$

where $\{y_{re}[n]\} = \{\cos 0.3n\}$

$$\{y_{im}[n]\} = \{\sin 0.3n\}$$

Discrete-Time Signals: Time-Domain Representation

- Example -

  $$\{w[n]\} = \{\cos 0.3n\} - j\{\sin 0.3n\} = \{e^{-j0.3n}\}$$

  is the complex conjugate sequence of $\{y[n]\}$

- That is,

  $$\{w[n]\} = \{y^*[n]\}$$

Discrete-Time Signals: Time-Domain Representation

- Two types of discrete-time signals:

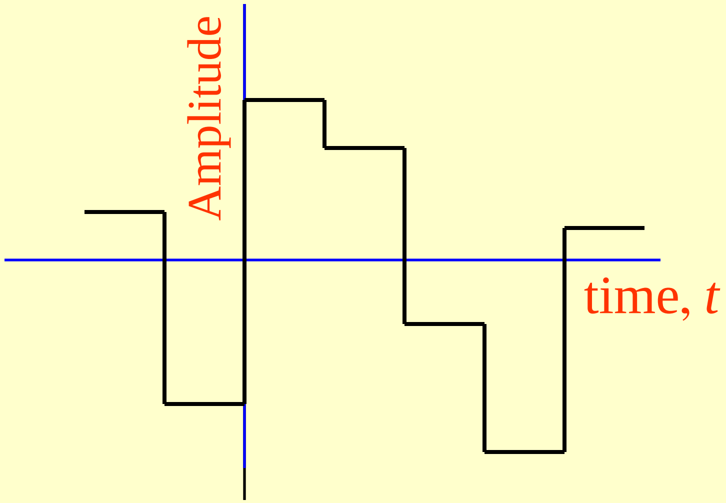
  - **Sampled-data signals** in which samples are continuous-valued

  - **Digital signals** in which samples are discrete-valued

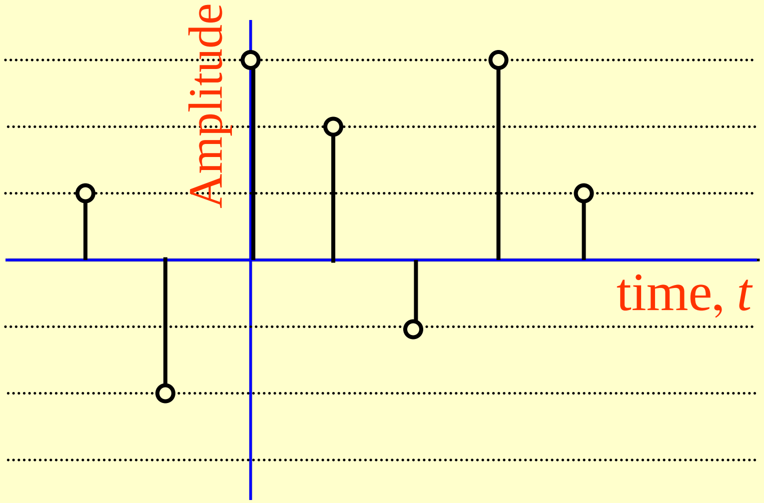
- Signals in a practical digital signal processing system are digital signals obtained by quantizing the sample values either by **rounding** or **truncation**

Discrete-Time Signals: Time-Domain Representation

- Example -



Boxedcar signal



Digital signal

Discrete-Time Signals: Time-Domain Representation

- A discrete-time signal may be a **finite-length** or an **infinite-length sequence**
- Finite-length (also called **finite-duration** or **finite-extent**) sequence is defined only for a finite time interval: $N_1 \leq n \leq N_2$

  where $-\infty < N_1$ and $N_2 < \infty$ with $N_1 \leq N_2$
- **Length** or **duration** of the above finite-length sequence is $N = N_2 - N_1 + 1$

Discrete-Time Signals: Time-Domain Representation

- Example - $x[n] = n^2, -3 \le n \le 4$ is a finite-length sequence of length $4-(-3)+1=8$

$y[n] = \cos 0.4n$ is an infinite-length sequence

Discrete-Time Signals: Time-Domain Representation

- A length-N sequence is often referred to as an N**-point sequence**

- The length of a finite-length sequence can be increased by **zero-padding**, i.e., by **appending** it **with zeros**

Discrete-Time Signals: Time-Domain Representation

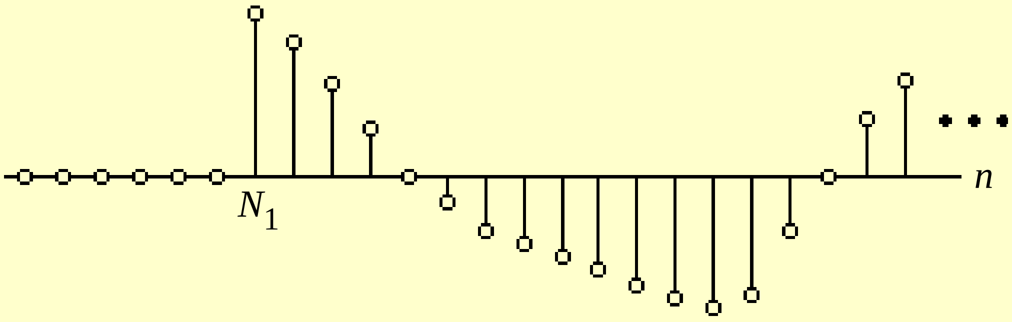
- Example -

$$x_e[n] = \begin{cases} n^2, & -3 \le n \le 4 \\ 0, & 5 \le n \le 8 \end{cases}$$

is a finite-length sequence of length 12 obtained by zero-padding $x[n] = n^2, -3 \le n \le 4$ with 4 zero-valued samples

Discrete-Time Signals: Time-Domain Representation

- A **right-sided sequence** $x[n]$ has zero-valued samples for $n < N_1$

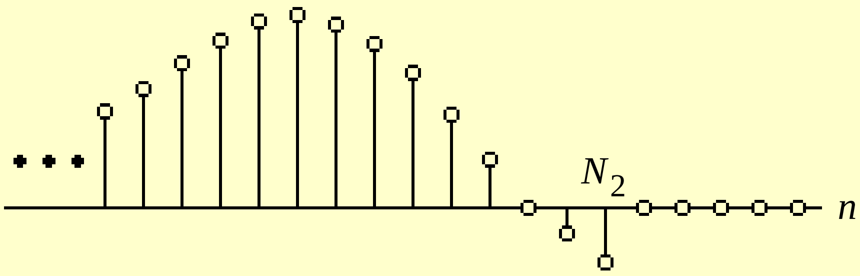


A right-sided sequence

- If $N_1 \geq 0$, a right-sided sequence is called a **causal sequence**

Discrete-Time Signals: Time-Domain Representation

- A **left-sided sequence** $x[n]$ has zero-valued samples for $n > N_2$



A left-sided sequence

- If $N_2 \leq 0,$ a left-sided sequence is called a **anti-causal sequence**

Discrete-Time Signals: Time-Domain Representation

- **Size of a Signal**

Given by the norm of the signal

$\mathcal{L}_p$-norm

$$\|x\|_p = \left(\sum_{n=-\infty}^{\infty} |x[n]|^p \right)^{1/p}$$

where p is a positive integer

Discrete-Time Signals: Time-Domain Representation

- The value of p is typically 1 or 2 or ∞

$\mathcal{L}_2$-norm

$$\|x\|_2$$

is the root-mean-squared (rms) value of $\{x[n]\}$

Discrete-Time Signals: Time-Domain Representation

$\mathcal{L}_1$-norm $\|x\|_1$

is the mean absolute value of $\{x[n]\}$

$\mathcal{L}_\infty$-norm $\|x\|_\infty$

is the peak absolute value of $\{x[n]\}$, i.e.

$$\|x\|_\infty = |x|_{\max}$$

Discrete-Time Signals: Time-Domain Representation

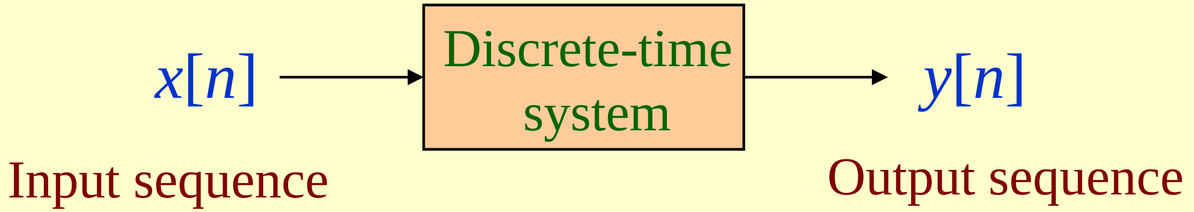
Example

- Let $\{y[n]\}, 0 \le n \le N-1$, be an approximation of $\{x[n]\}$, $0 \le n \le N-1$
- An estimate of the relative error is given by the ratio of the $\mathcal{L}_2$-norm of the difference signal and the $\mathcal{L}_2$-norm of $\{x[n]\}$:

$$E_{rel} = \left(\frac{\sum_{n=0}^{N-1} \left| y[n] - x[n] \right|^2}{\sum_{n=0}^{N-1} \left| x[n] \right|^2} \right)^{1/p}$$

Operations on Sequences

- A single-input, single-output discrete-time system operates on a sequence, called the **input sequence**, according some prescribed rules and develops another sequence, called the output sequence, with more desirable properties

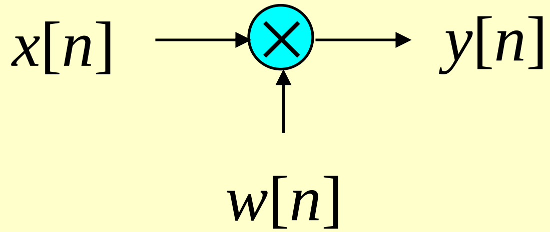


Operations on Sequences

- For example, the input may be a signal corrupted with additive noise
- Discrete-time system is designed to generate an output by removing the noise component from the input
- In most cases, the operation defining a particular discrete-time system is composed of some **elementary operations**

Elementary Operations

- **Product (modulation)** operation:
  - Modulator

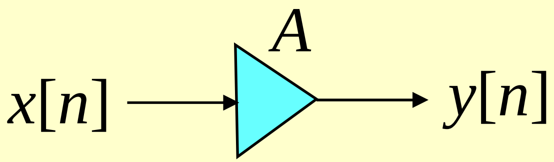


$$y[n] = x[n] \cdot w[n]$$

- An application is in forming a finite-length sequence from an infinite-length sequence by multiplying the latter with a finite-length sequence called an **window sequence**
- Process called **windowing**

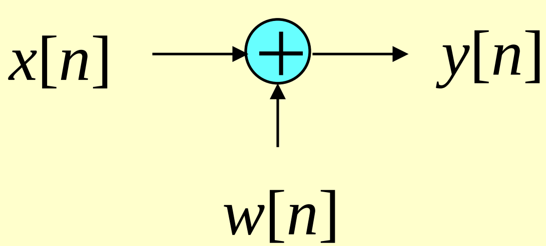
Elementary Operations

- **Multiplication** operation
  - Multiplier



$$y[n] = A \cdot x[n]$$

- Addition operation
  - Adder



$$y[n] = x[n] + w[n]$$

Addition Operation

$$y[n] = x[n] + w[n]$$

- Subtraction operation

  By inverting the signs of all samples of $w[n]$, an adder can also implement the subtraction operation

Elementary Operations

- **Time-shifting** operation: $y[n] = x[n-N]$
  where N is an integer
- If $N > 0$, it is **delaying** operation
  – Unit delay

$$x[n] \longrightarrow \boxed{z^{-1}} \longrightarrow y[n] \qquad y[n] = x[n-1]$$

- If $N < 0$, it is an **advance** operation

$$x[n] \longrightarrow \boxed{z} \longrightarrow y[n] \qquad y[n] = x[n+1]$$

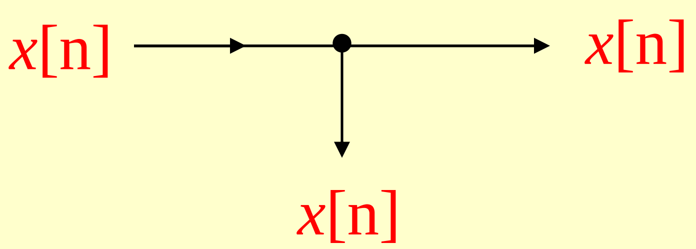
  – Unit advance

Elementary Operations

- **Time-reversal (folding)** operation:

$$y[n] = x[-n]$$

- **Branching** operation: Used to provide multiple copies of a sequence



Elementary Operations

- <u>Example</u> - Consider the two following sequences of length 5 defined for $0 \le n \le 4$:

$$\{a[n]\} = \{3 \quad 4 \quad 6 \quad -9 \quad 0\}$$

$$\{b[n]\} = \{2 \quad -1 \quad 4 \quad 5 \quad -3\}$$

- New sequences generated from the above two sequences by applying the basic operations are as follows:

Elementary Operations

$$\{c[n]\} = \{a[n] \cdot b[n]\} = \{6 \quad -4 \quad 24 \quad -45 \quad 0\}$$

$$\{d[n]\} = \{a[n] + b[n]\} = \{5 \quad 3 \quad 10 \quad -4 \quad -3\}$$

$$\{e[n]\} = \frac{3}{2}\{a[n]\} = \{4.5 \quad 6 \quad 9 \quad -13.5 \quad 0\}$$

- As pointed out by the above example, operations on two or more sequences can be carried out if all sequences involved are of same length and defined for the same range of the time index n

Elementary Operations

- However if the sequences are not of same length, in some situations, this problem can be circumvented by appending zero-valued samples to the sequence(s) of smaller lengths to make all sequences have the same range of the time index
- Example - Consider the sequence of length 3 defined for $0 \le n \le 2$: $\{f[n]\} = \{-2 \;\; 1 \;\; -3\}$

Elementary Operations

- We cannot add the length-3 sequence $\{f[n]\}$ to the length-5 sequence $\{a[n]\}$ defined earlier
- We therefore first append $\{f[n]\}$ with 2 zero-valued samples resulting in a length-5 sequence $\{f_e[n]\} = \{-2 \;\; 1 \;\; -3 \;\; 0 \;\; 0\}$
- Then

$$\{g[n]\} = \{a[n]\} + \{f_e[n]\} = \{1 \quad 5 \quad 3 \quad -9 \quad 0\}$$

Elementary Operations

Ensemble Averaging

- A very simple application of the addition operation in improving the quality of measured data corrupted by an additive random noise
- In some cases, actual uncorrupted data vector **s** remains essentially the same from one measurement to next

Elementary Operations

- While the additive noise vector is random and not reproducible
- Let $\mathbf{d}_i$ denote the noise vector corrupting the i-th measurement of the uncorrupted data vector $\mathbf{s}$:

$$\mathbf{x}_i = \mathbf{s} + \mathbf{d}_i$$

Measured data vector

Uncorrupted data vector

Noise vector

Elementary Operations

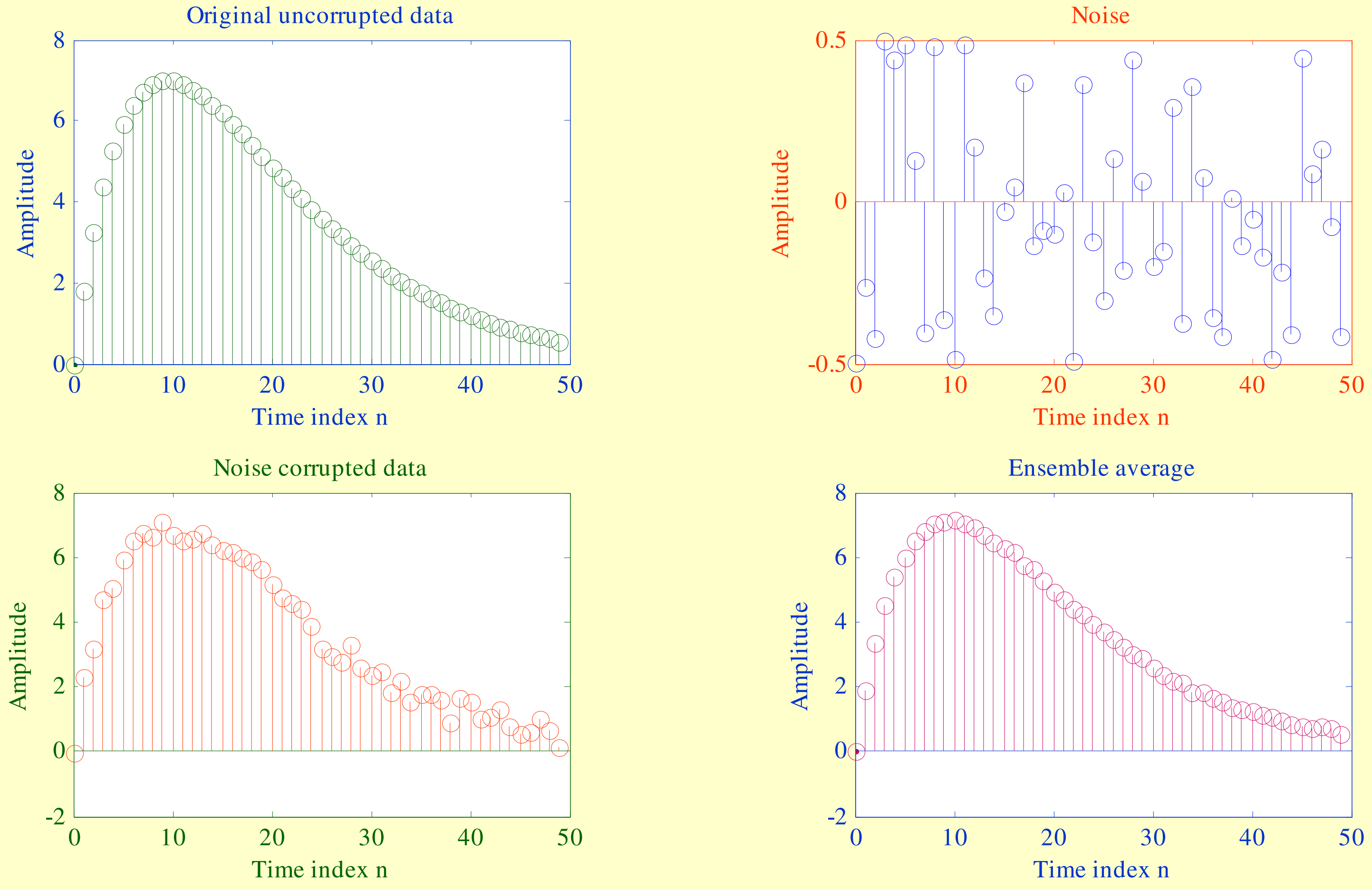
- The average data vector, called the ensemble average, obtained after K measurements is given by

$$\mathbf{x}_{ave} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{x}_i = \frac{1}{K}\sum_{i=1}^{K}(\mathbf{s}+\mathbf{d}_i) = \mathbf{s} + \frac{1}{K}\sum_{i=1}^{K}\mathbf{d}_i$$

- For large values of K, $\mathbf{x}_{ave}$ is usually a reasonable replica of the desired data vector $\mathbf{s}$

Elementary Operations

- **Example**



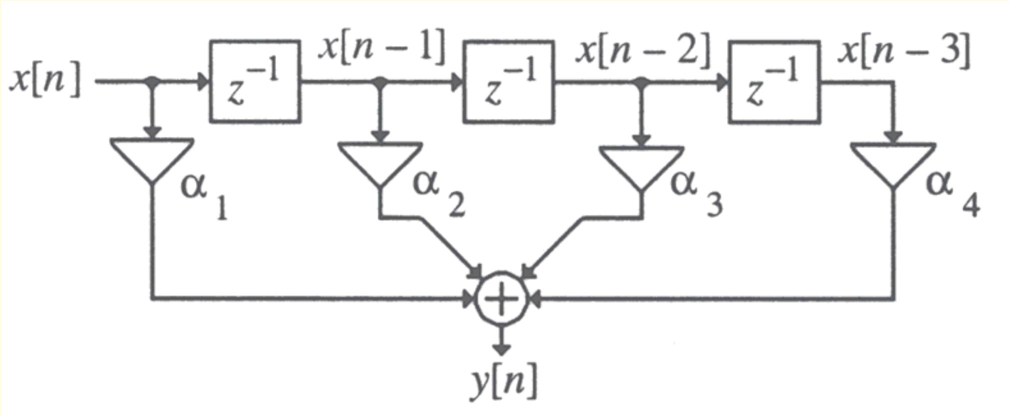
Elementary Operations

- We cannot add the length-3 sequence $\{f[n]\}$ to the length-5 sequence $\{a[n]\}$ defined earlier
- We therefore first append $\{f[n]\}$ with 2 zero-valued samples resulting in a length-5 sequence $\{f_e[n]\} = \{-2 \quad 1 \quad -3 \quad 0 \quad 0\}$
- Then

$$\{g[n]\} = \{a[n]\} + \{f_e[n]\} = \{1 \quad 5 \quad 3 \quad -9 \quad 0\}$$

Combinations of Basic Operations

- Example -



$$y[n] = \alpha_1 x[n] + \alpha_2 x[n-1] + \alpha_3 x[n-2] + \alpha_4 x[n-3]$$

Convolution Sum

- The summation

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

is called the **convolution sum** of the sequences $x[n]$ and $h[n]$ and represented compactly as

$$y[n] = x[n] \circledast h[n]$$

Convolution Sum

- We illustrate the convolution operation for the following two sequences:

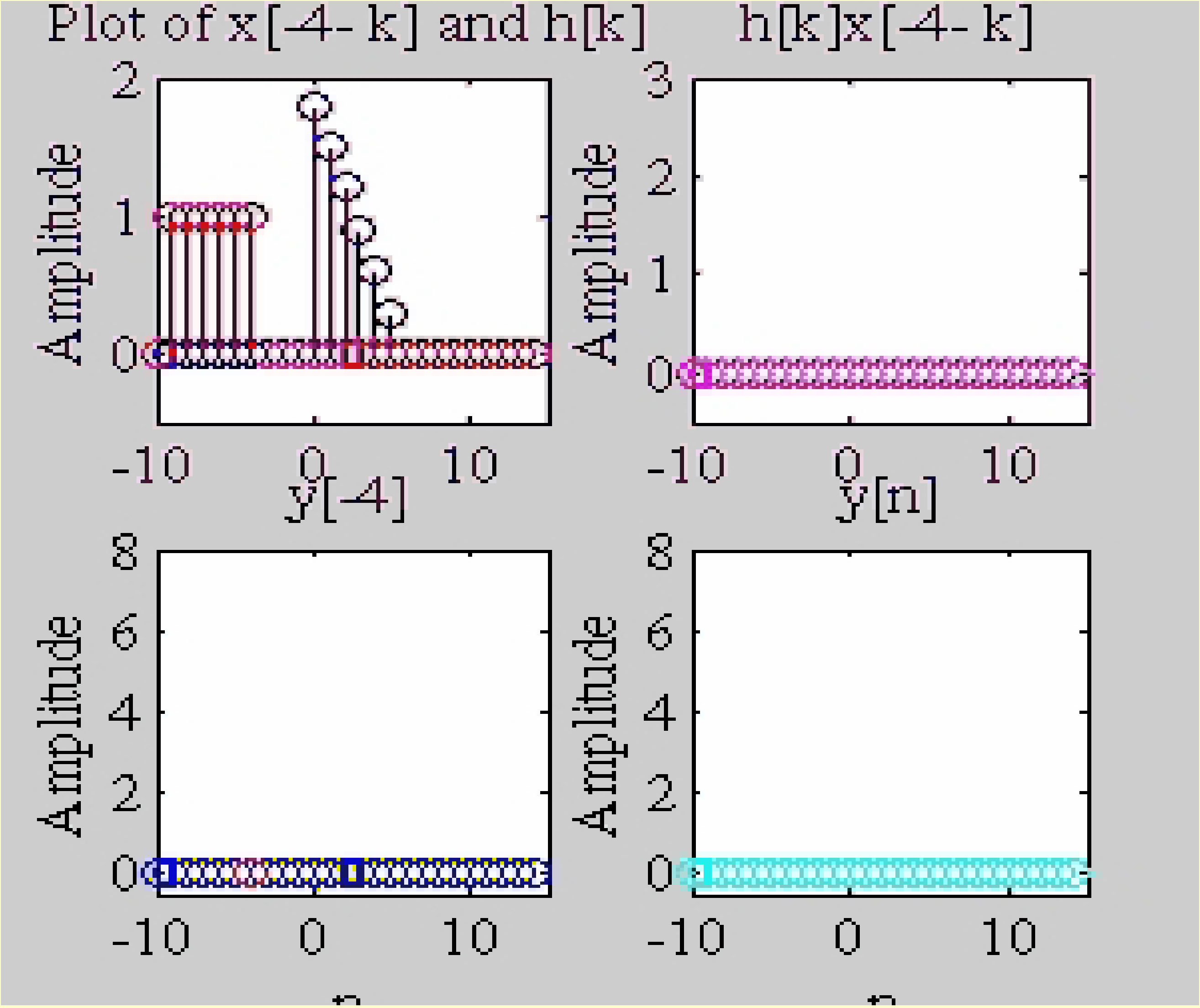
$$x[n] = \begin{cases} 1, & 0 \le n \le 5 \\ 0, & \text{otherwise} \end{cases}$$

$$h[n] = \begin{cases} 1.8 - 0.3n, & 0 \le n \le 5 \\ 0, & \text{otherwise} \end{cases}$$

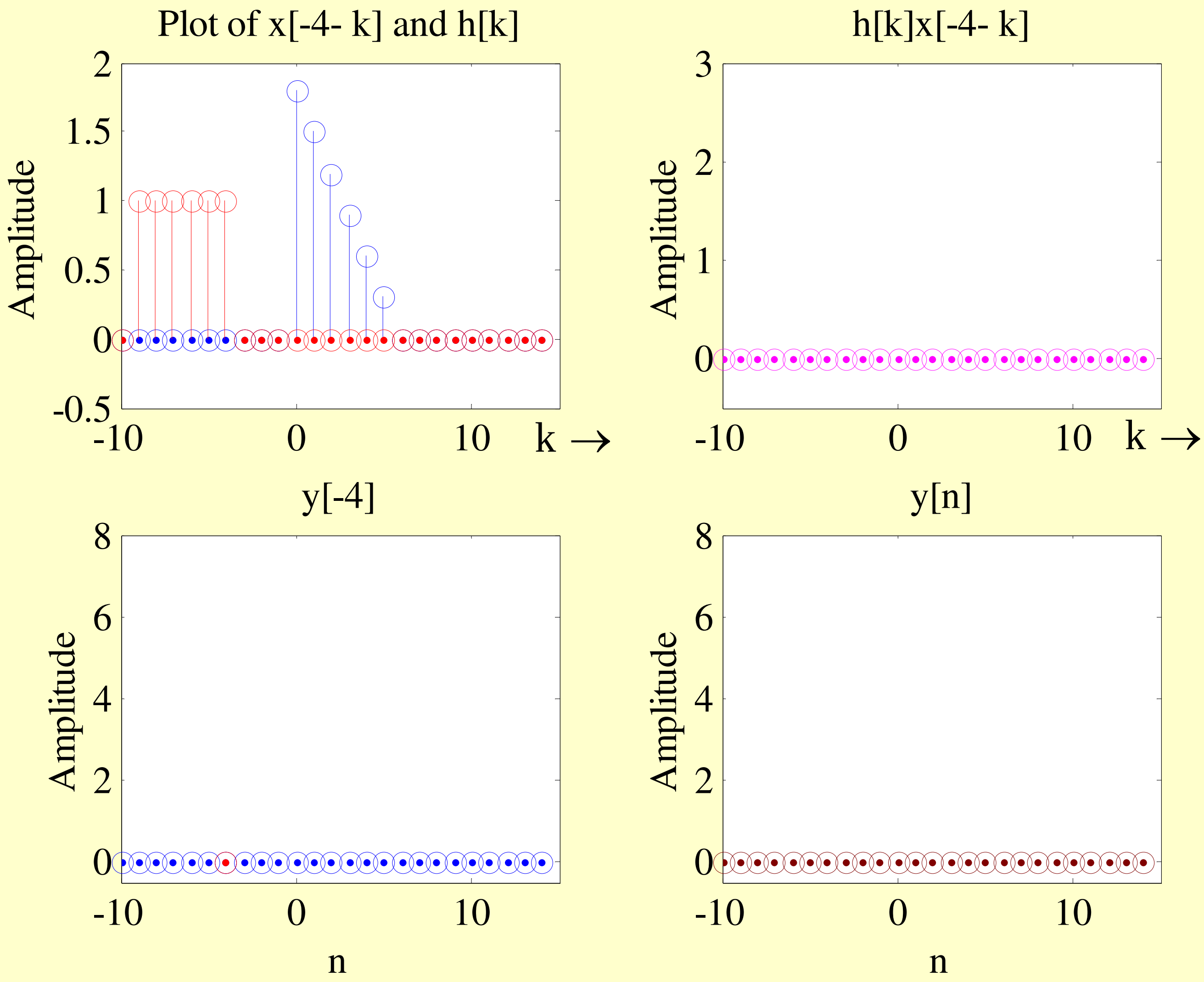
- Figures on the next several slides the steps involved in the computation of

$$y[n] = x[n] \circledast h[n]$$

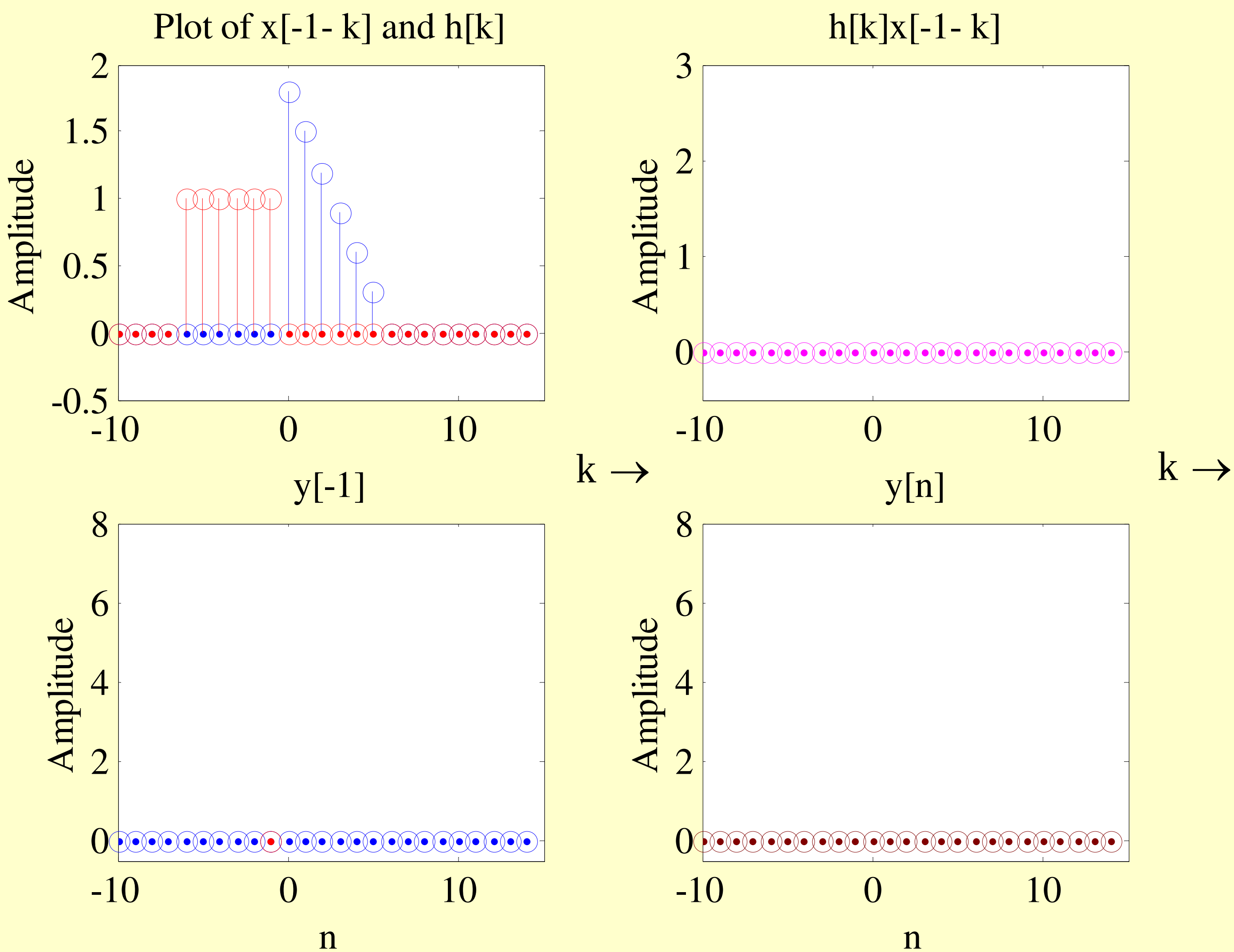
Convolution Sum



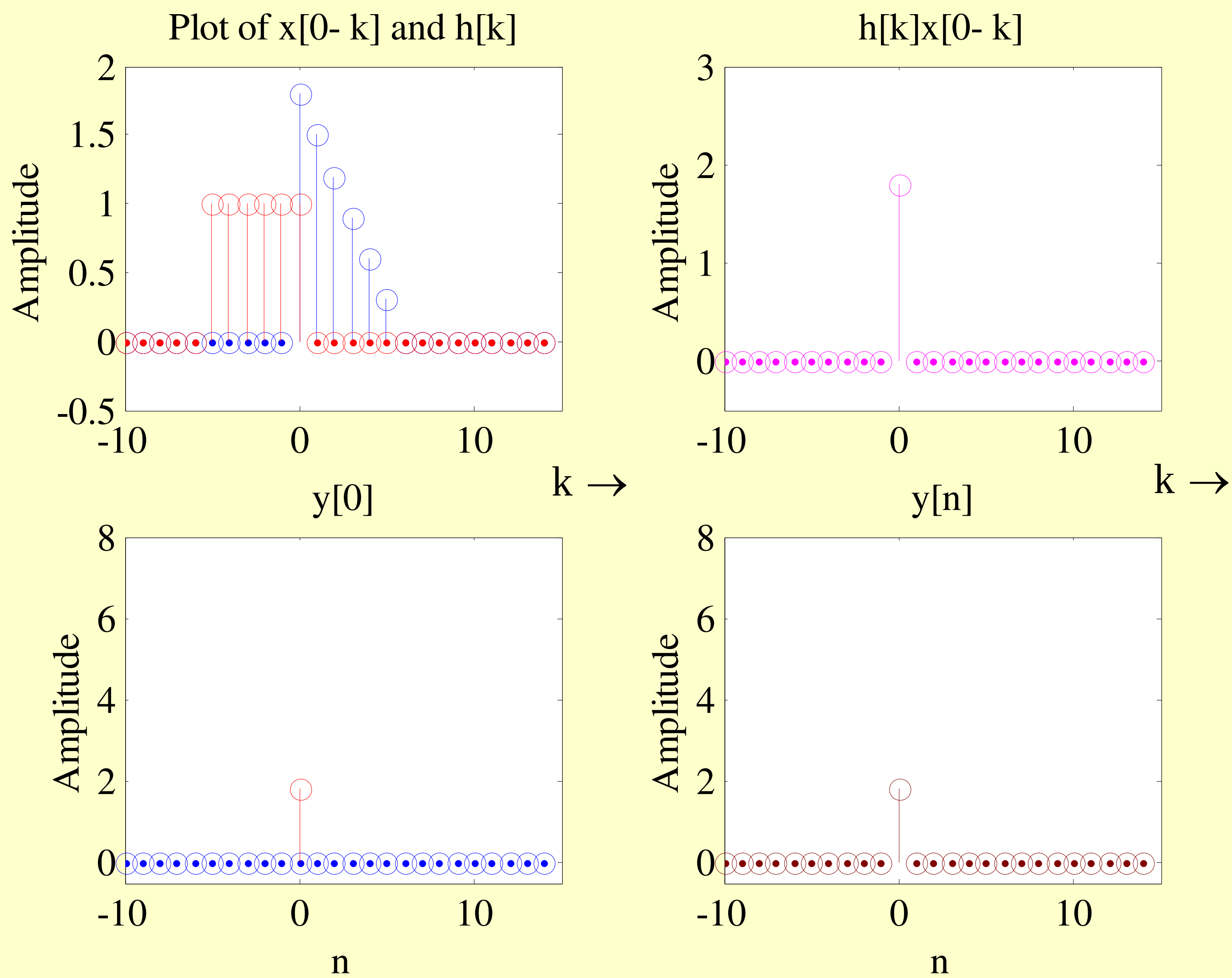
Convolution Sum



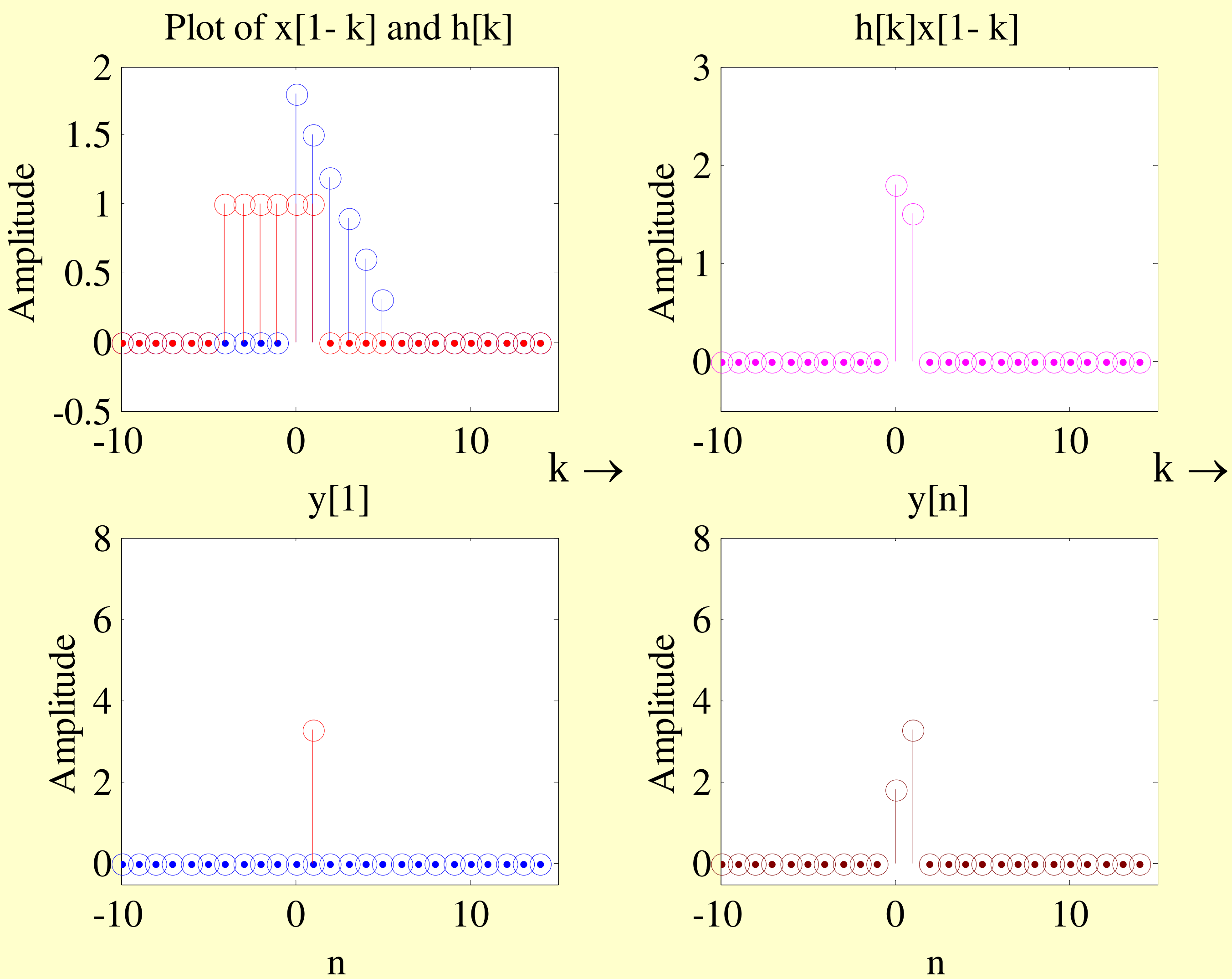
Convolution Sum



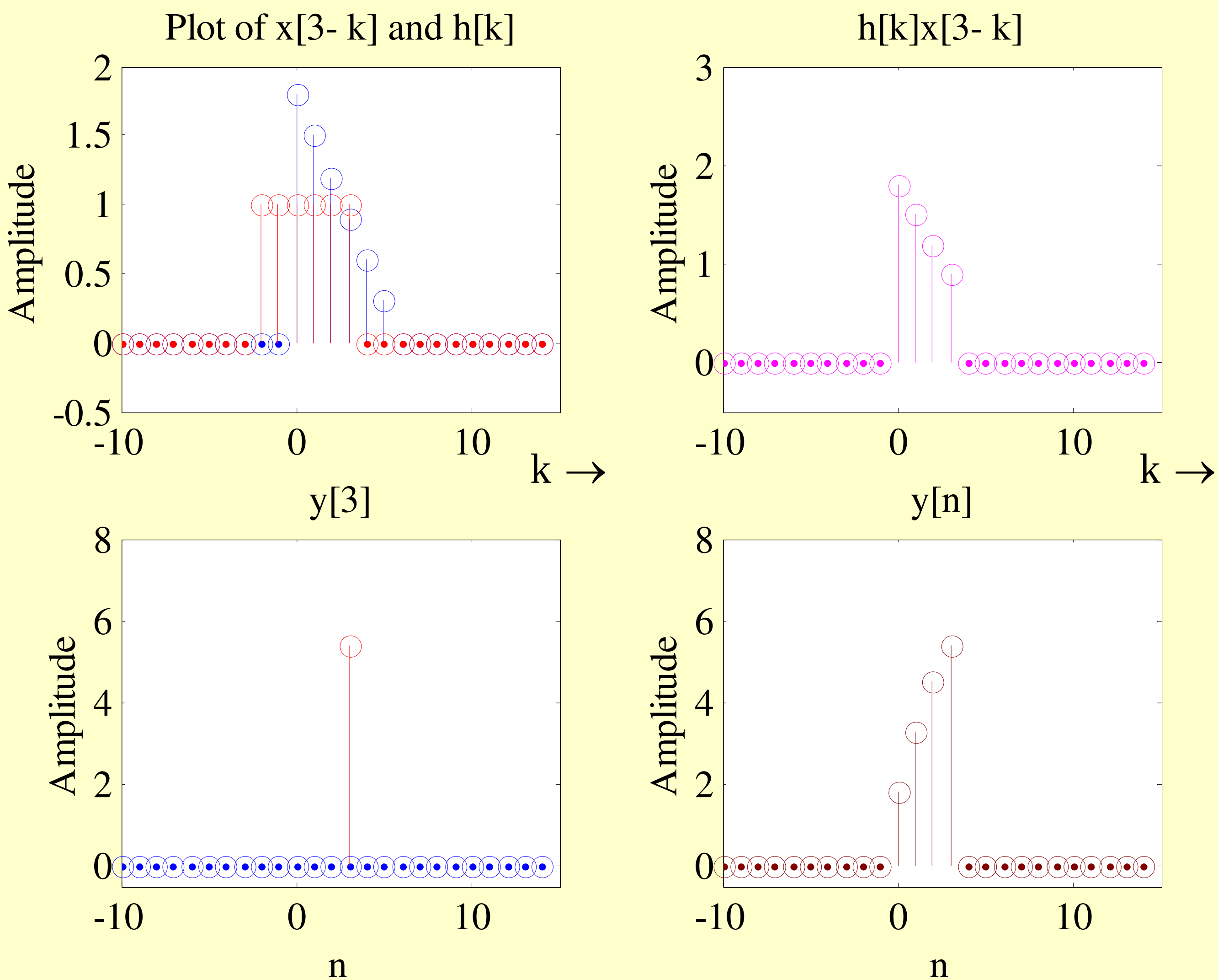
Convolution Sum



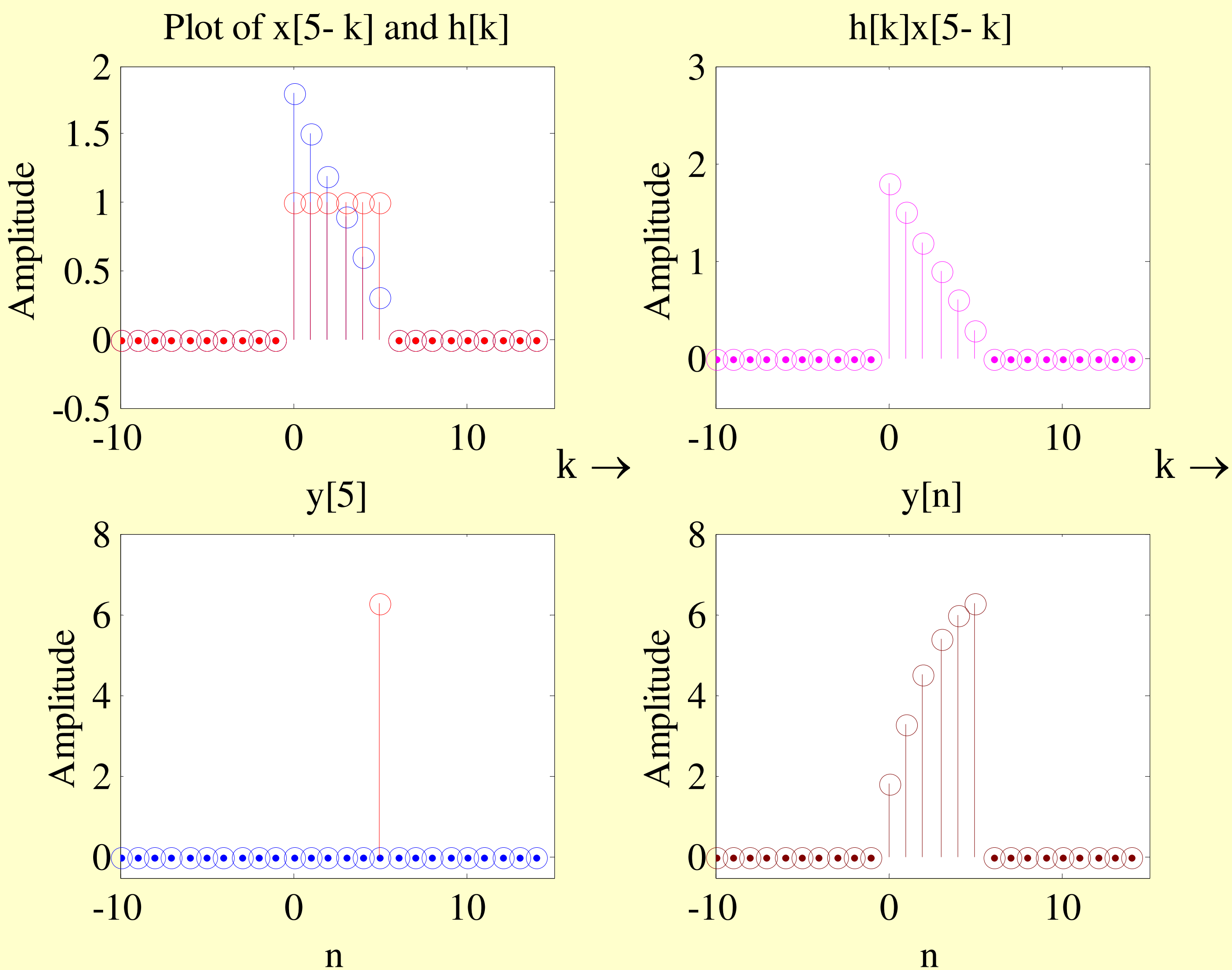
Convolution Sum



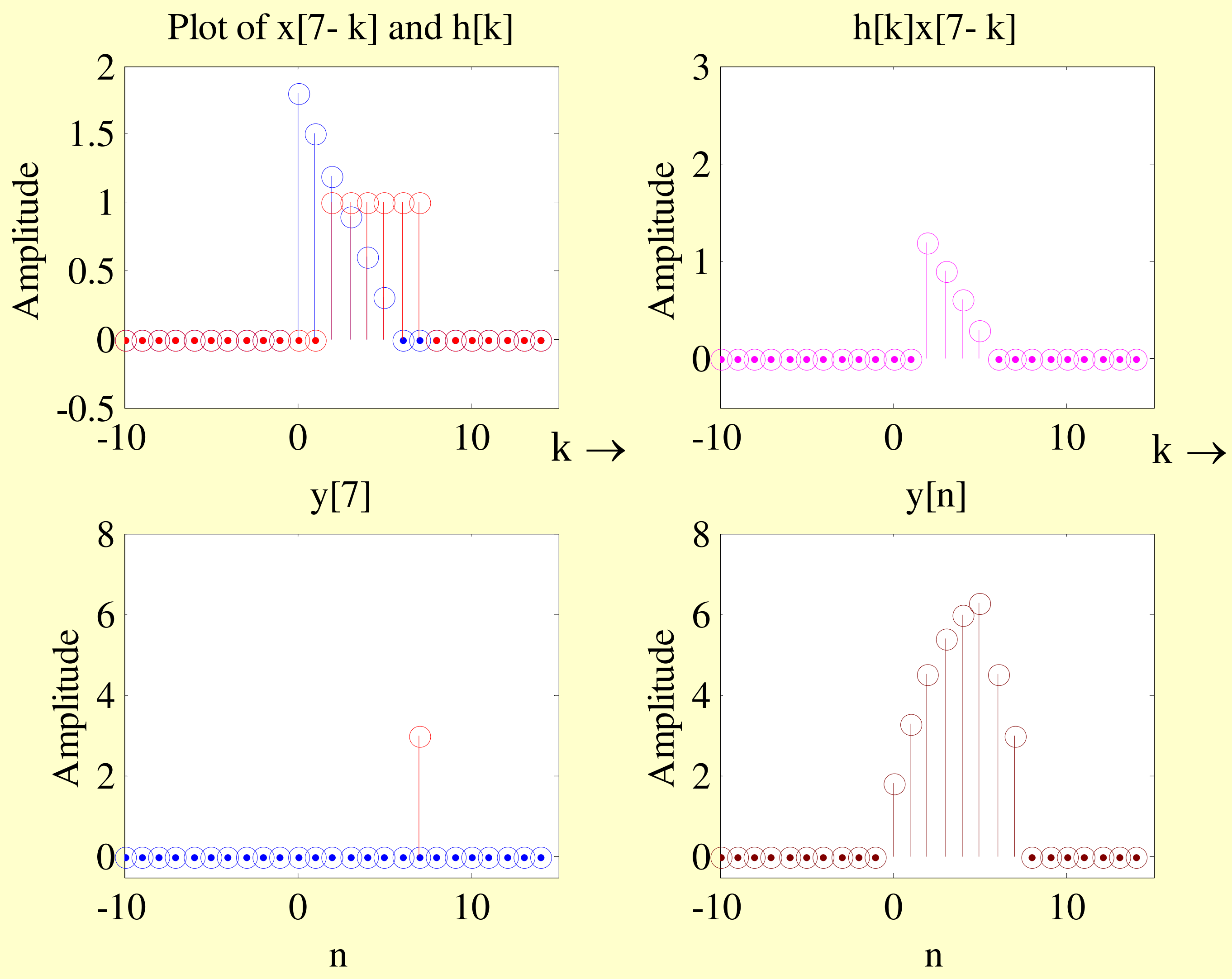
Convolution Sum



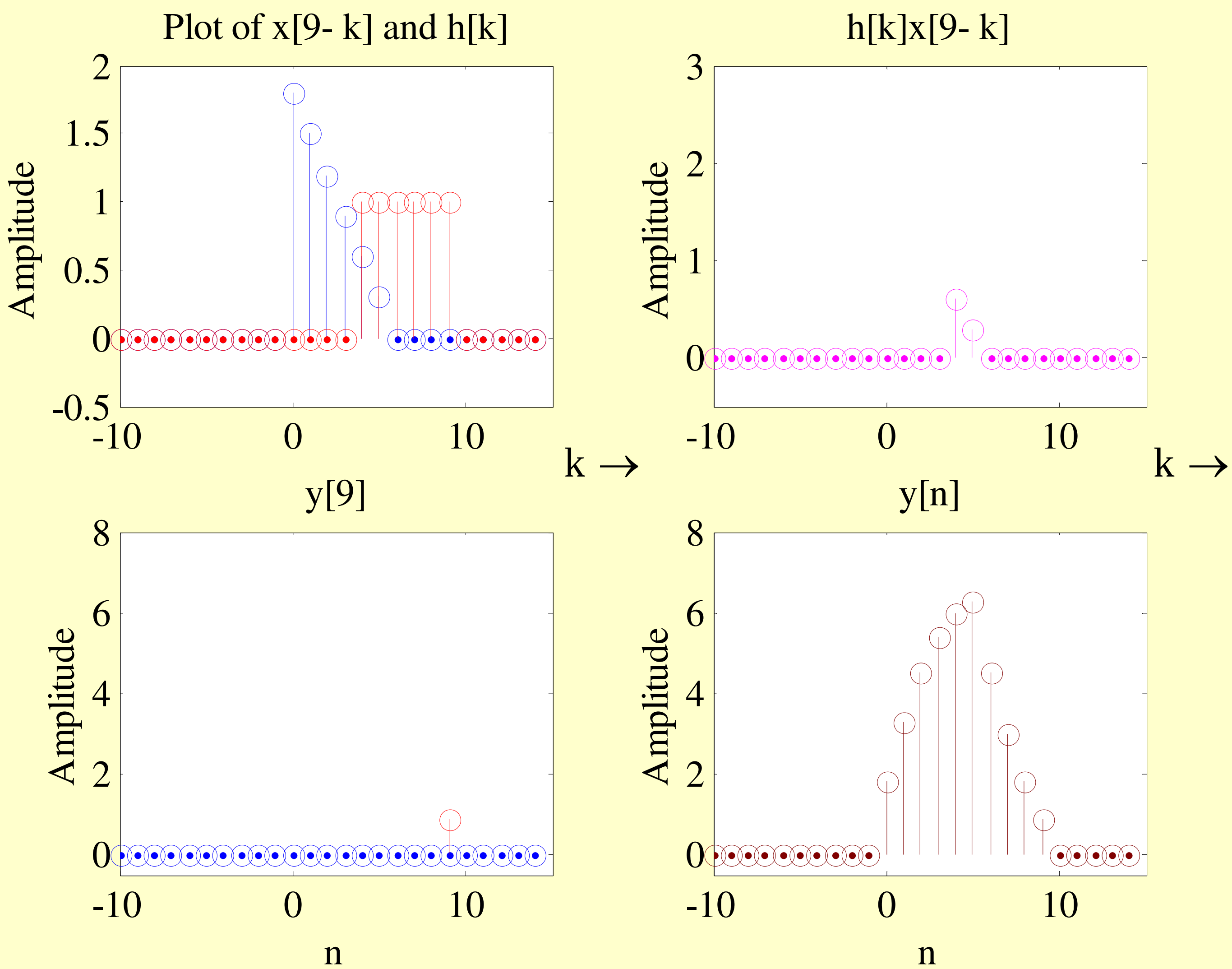
Convolution Sum



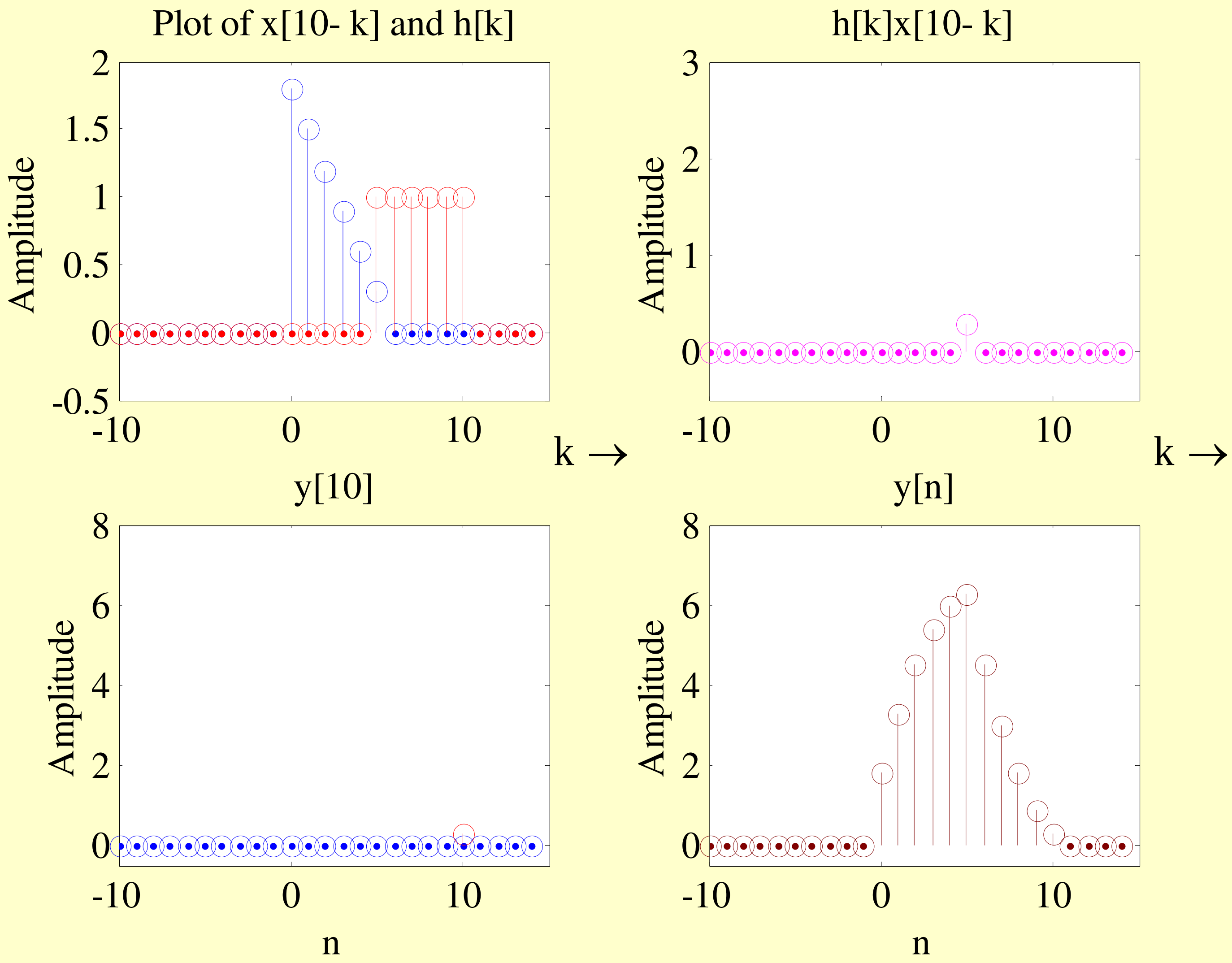
Convolution Sum



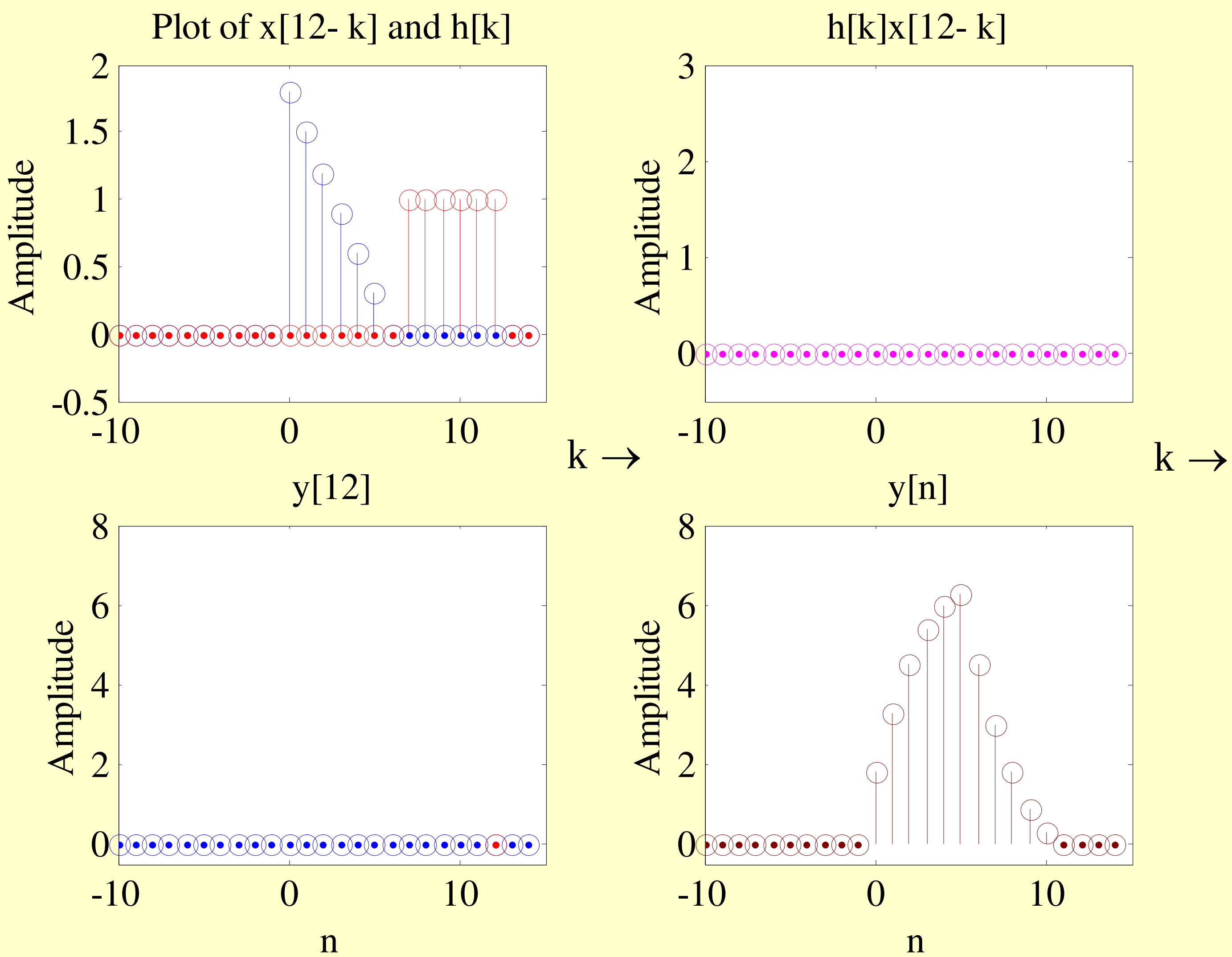
Convolution Sum



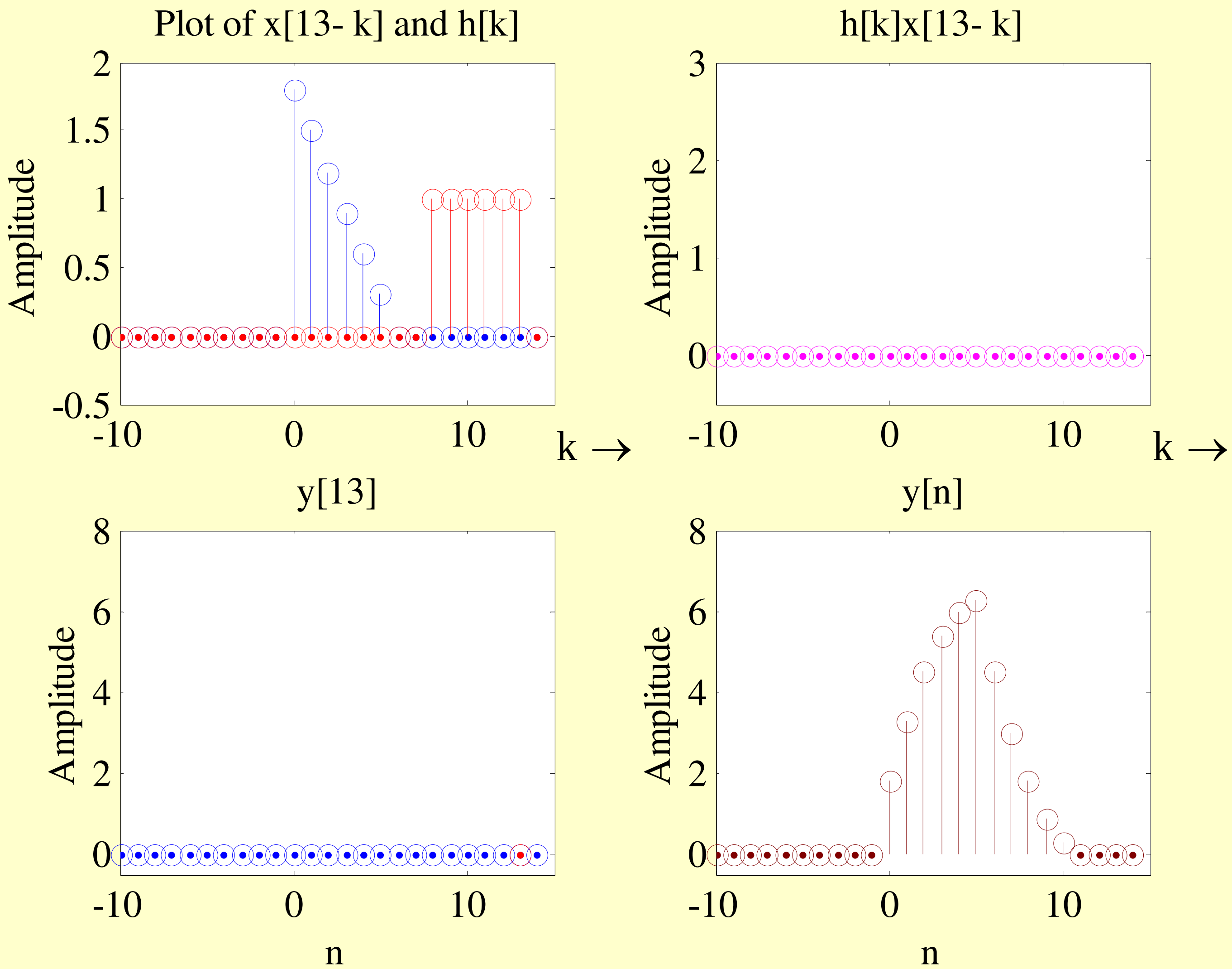
Convolution Sum



Convolution Sum

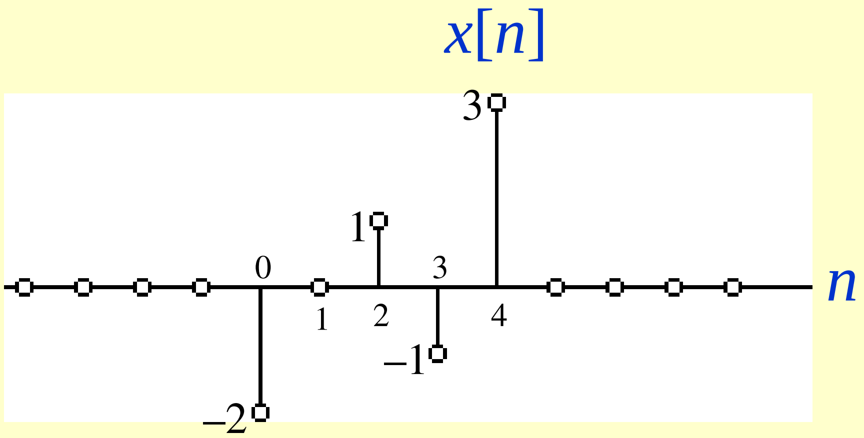


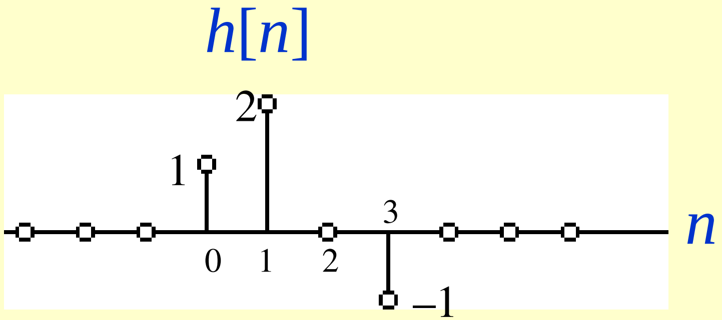
Convolution Sum



Convolution Sum

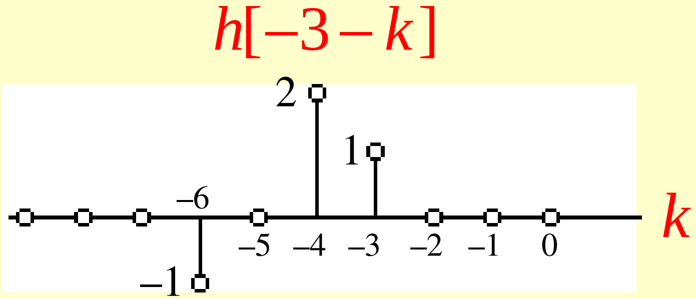
- Example - Develop the sequence $y[n]$ generated by the convolution of the sequences $x[n]$ and $h[n]$ shown below





Convolution Sum

- As can be seen from the shifted time-reversed version $\{h[n-k]\}$ for $n < 0$, shown below for $n = -3$, for any value of the sample index k, the k-th sample of either $\{x[k]\}$ or $\{h[n-k]\}$ is zero

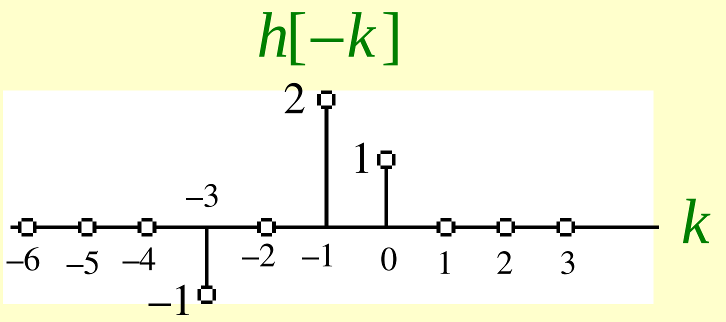


Convolution Sum

- As a result, for $n < 0$, the product of the k-th samples of $\{x[k]\}$ and $\{h[n-k]\}$ is always zero, and hence

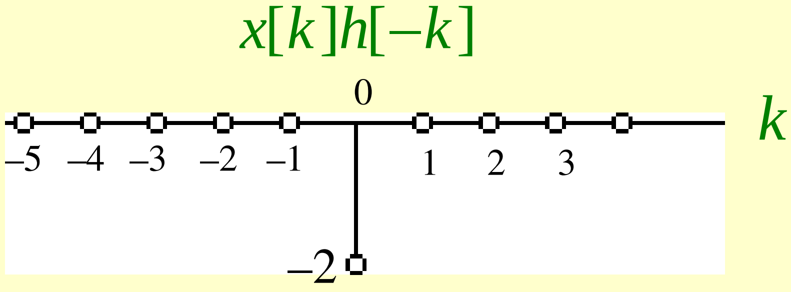
$$y[n] = 0 \qquad \text{for } \; n < 0$$

- Consider now the computation of $y[0]$
- The sequence $\{h[-k]\}$ is shown on the right



Convolution Sum

- The product sequence $\{x[k]h[-k]\}$ is plotted below which has a single nonzero sample $x[0]h[0]$ for $k = 0$

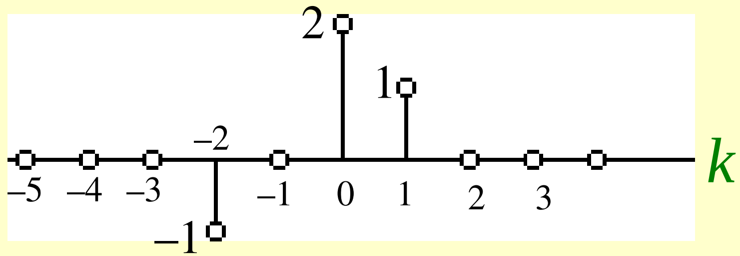


- Thus $y[0] = x[0]h[0] = -2$

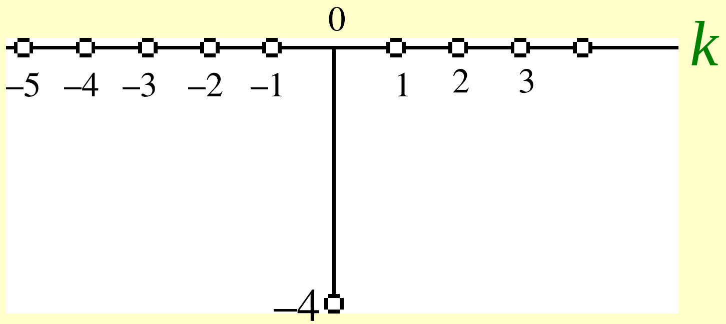
Convolution Sum

- For the computation of y[1], we shift $\{h[-k]\}$ to the right by one sample period to form $\{h[1-k]\}$ as shown below on the left
- The product sequence $\{x[k]h[1-k]\}$ is shown below on the right

$h[1-k]$



$x[k]h[1-k]$

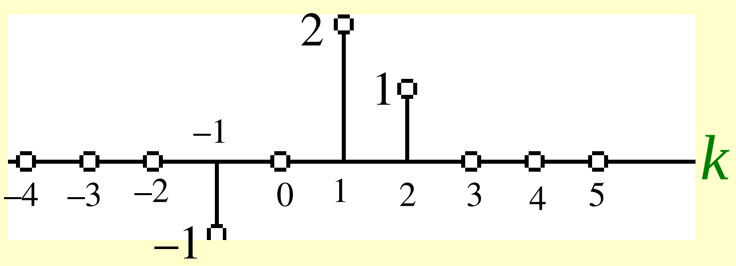


- Hence, $y[1] = x[0]h[1] + x[1]h[0] = -4 + 0 = -4$

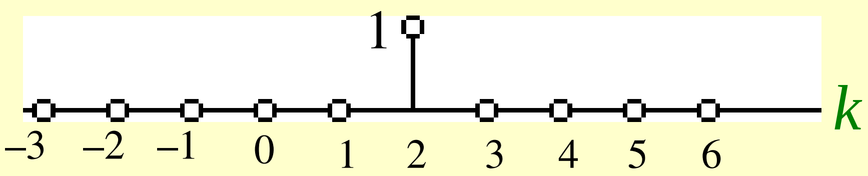
Convolution Sum

- To calculate $y[2]$, we form $\{h[2-k]\}$ as shown below on the left
- The product sequence $\{x[k]h[2-k]\}$ is plotted below on the right

$h[2-k]$



$x[k]h[2-k]$



$$y[2] = x[0]h[2] + x[1]h[1] + x[2]h[0] = 0 + 0 + 1 = 1$$

Convolution Sum

- Continuing the process we get

$$y[3] = x[0]h[3] + x[1]h[2] + x[2]h[1] + x[3]h[0]$$
$$= 2 + 0 + 0 + 1 = 3$$

$$y[4] = x[1]h[3] + x[2]h[2] + x[3]h[1] + x[4]h[0]$$
$$= 0 + 0 - 2 + 3 = 1$$

$$y[5] = x[2]h[3] + x[3]h[2] + x[4]h[1]$$
$$= -1 + 0 + 6 = 5$$

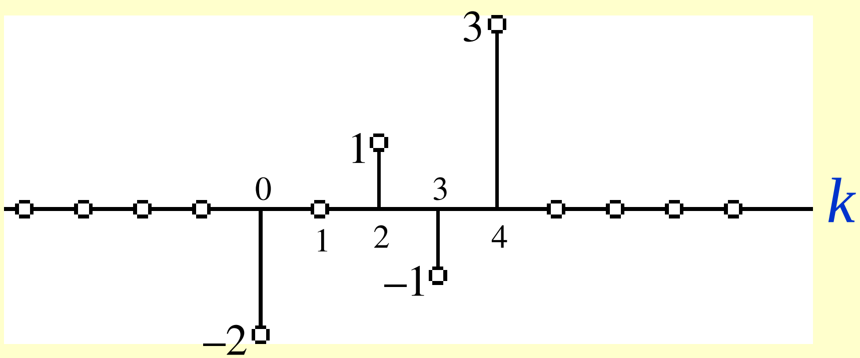
$$y[6] = x[3]h[3] + x[4]h[2] = 1 + 0 = 1$$

$$y[7] = x[4]h[3] = -3$$

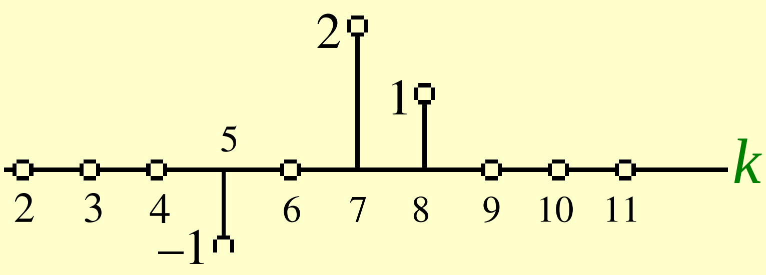
Convolution Sum

- From the plot of $\{h[n-k]\}$ for $n > 7$ and the plot of $\{x[k]\}$ as shown below, it can be seen that there is no overlap between these two sequences
- As a result $y[n] = 0$ for $n > 7$

$x[k]$

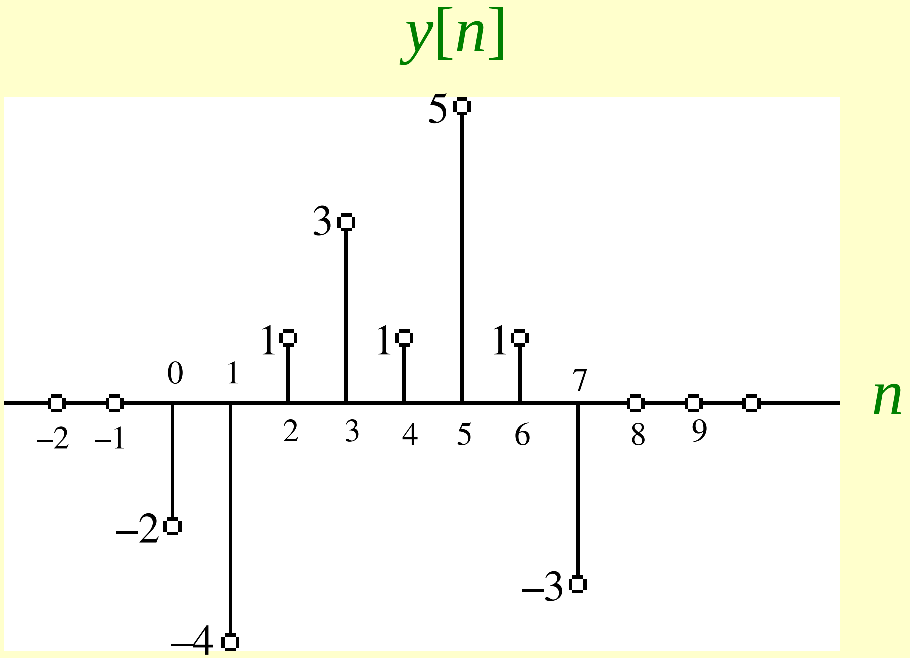


$h[8-k]$



Convolution Sum

- The sequence $\{y[n]\}$ generated by the convolution sum is shown below



Convolution Sum

- Note: The sum of indices of each sample product inside the convolution sum is equal to the index of the sample being generated by the convolution operation
- For example, the computation of $y[3]$ in the previous example involves the products $x[0]h[3]$, $x[1]h[2]$, $x[2]h[1]$, and $x[3]h[0]$
- The sum of indices in each of these products is equal to 3

Convolution Sum

- In the example considered the convolution of a sequence $\{x[n]\}$ of length 5 with a sequence $\{h[n]\}$ of length 4 resulted in a sequence $\{y[n]\}$ of length 8
- In general, if the lengths of the two sequences being convolved are M and N, then the sequence generated by the convolution is of length $M+N-1$

Sampling Rate Alteration

- Employed to generate a new sequence $y[n]$ with a sampling rate $F_T^{'}$ higher or lower than that of the sampling rate F_T of a given sequence $x[n]$
- **Sampling rate alteration ratio** is $R = \frac{F_T^{'}}{F_T}$
- If $R > 1$, the process called **interpolation**
- If $R < 1$, the process called **decimation**

Sampling Rate Alteration

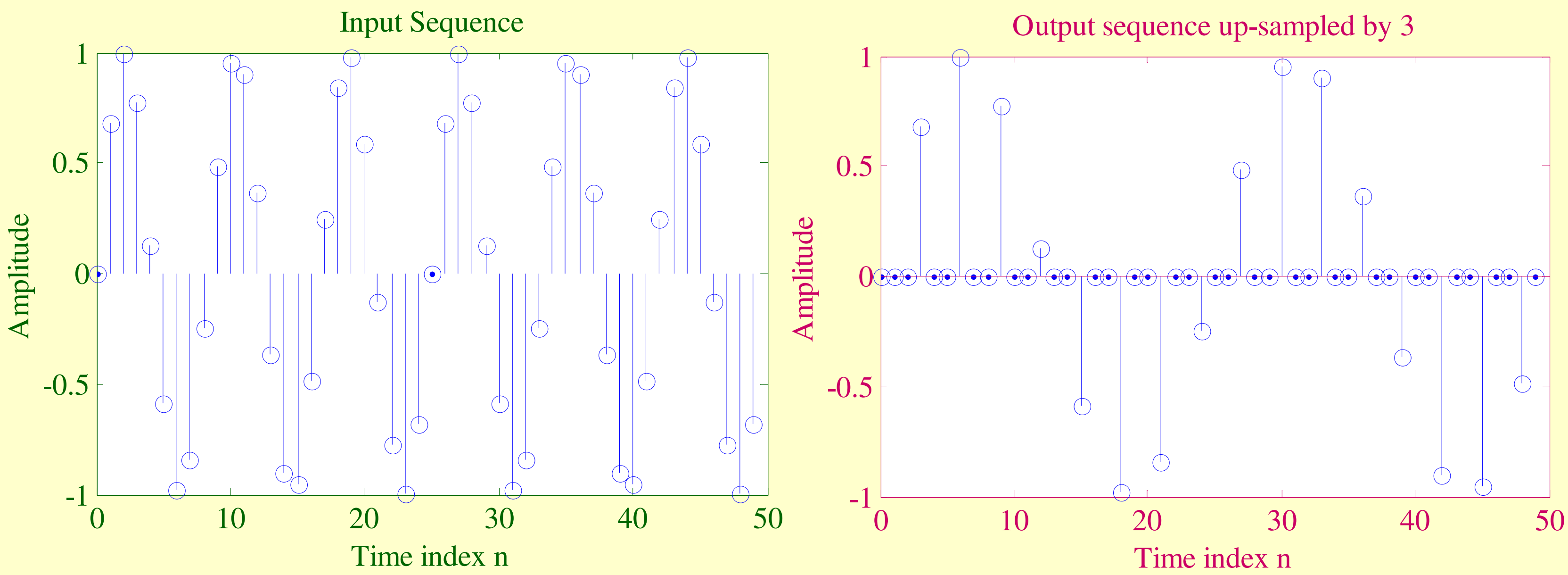
- In **up-sampling** by an integer factor $L > 1$, $L-1$ equidistant zero-valued samples are inserted by the **up-sampler** between each two consecutive samples of the input sequence $x[n]$:

$$x_u[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \cdots \\ 0, & \text{otherwise} \end{cases}$$

$$x[n] \longrightarrow \boxed{\uparrow L} \longrightarrow x_u[n]$$

Sampling Rate Alteration

- An example of the up-sampling operation



Sampling Rate Alteration

- In **down-sampling** by an integer factor $M > 1$, every M-th samples of the input sequence are kept and $M-1$ in-between samples are removed:

$$y[n] = x[nM]$$

$$x[n] \longrightarrow \boxed{\downarrow M} \longrightarrow y[n]$$

Sampling Rate Alteration

- An example of the down-sampling operation