

---

*EEE 507 – Multidimensional Digital Signal Processing*

<http://www.fulton.asu.edu/~karam/eee507/>

## Introduction – 2D Signals and Systems

Prof. Lina Karam  
Department of Electrical Engineering  
Arizona State University  
*karam@asu.edu*

---

# What is different from 1-D?

- *Many things are the same: concepts generalize, details change*
- *Some mathematics do not generalize*
  - *Polynomials do not factor*
  - *Remez algorithm will not generalize*
- *Application assumptions different*
  - *Not causal*
  - *Inputs finite extent*
- *Problems bigger*
  - **Example: HDTV 1920x1080x30 frames/sec > 62 million color samples/sec**
    - ~ 186 M bytes/sec
    - ~ 5.3 nanosec/byte
  - Processor speed?
  - Memory?
  - Addressing capability?
  - Need to minimize computations whenever possible
- *New possibilities*

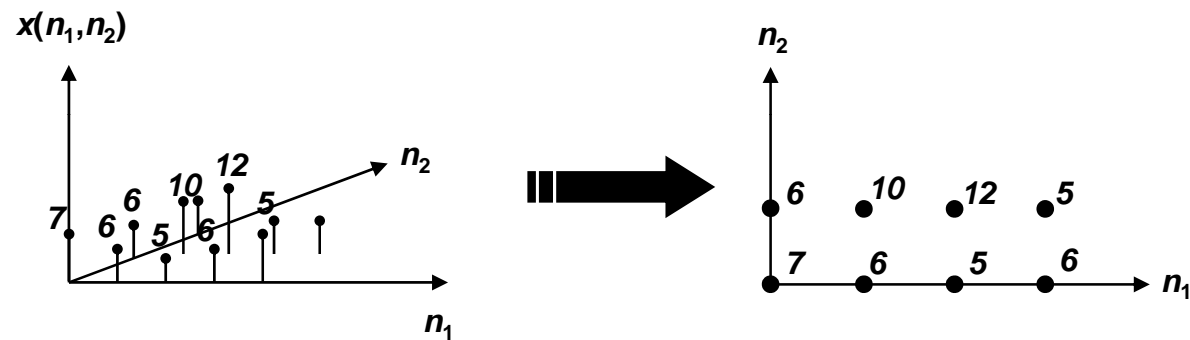
---

## 2-D Discrete Signals

- $x(n_1, n_2)$  : discrete signal
  - $x$  : real or complex value
  - $(n_1, n_2)$  : pair of integer indices

# 2D Signal Representation

$x(n_1, n_2)$  has 2 axes ( $n_1, n_2$ ) + amplitude axis



---

# Examples

- **Sampled Black & White Photograph:  $x(n_1, n_2)$**

$x(n_1, n_2)$  scalar indicating pixel intensity at location  $(n_1, n_2)$

For example:  $x = 0$   $\longrightarrow$  Black

$x = 1$   $\longrightarrow$  White

$0 < x < 1$   $\longrightarrow$  In-between

- **Sampled color video/TV signal**

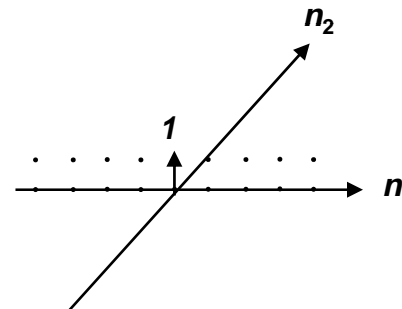
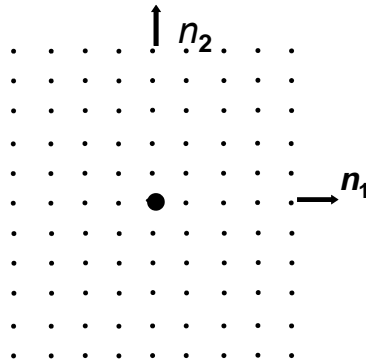
$$\begin{pmatrix} x_R(n_1, n_2, n_3) \\ x_G(n_1, n_2, n_3) \\ x_B(n_1, n_2, n_3) \end{pmatrix}$$

- **Seismic array:  $x(n_1, n_2)$**

# Special 2-D Sequences

- 2D Unit Impulse

$$x(n_1, n_2) = \delta(n_1, n_2) = \begin{cases} 1, & n_1 = n_2 = 0 \\ 0, & \text{else} \end{cases}$$
$$= \delta(n_1) \delta(n_2)$$



# Line impulses

- Vertical line impulse:

$$x(n_1, n_2) = \delta(n_1)$$

- Horizontal line impulse:

$$x(n_1, n_2) = \delta(n_2)$$

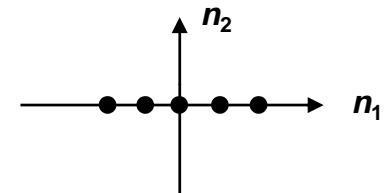
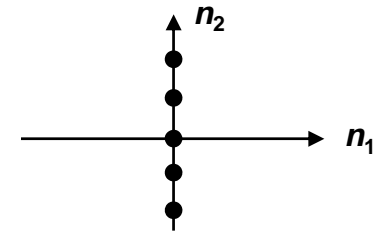
- Other line impulses:

$$\delta(n_1 + n_2),$$

$$\delta(n_1 - n_2),$$

$$\delta(2n_1 - n_2),$$

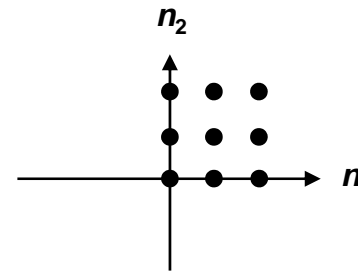
$$\delta(Pn_1 + Qn_2), \quad \text{slope} = -P/Q$$



---

- **2D Unit Step**

$$u(n_1, n_2) = \begin{cases} 1, & n_1 \text{ \& } n_2 \geq 0 \\ 0, & \text{otherwise} \end{cases}$$



**Note:**  $u(n_1, n_2) = u(n_1)u(n_2)$

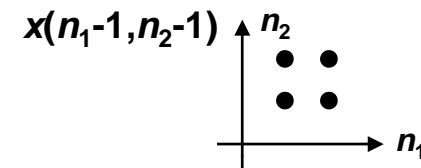
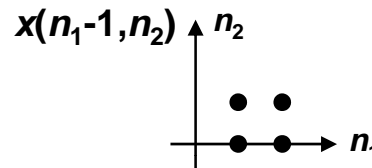
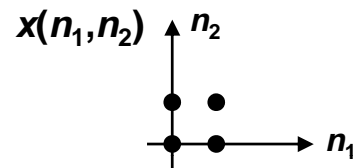
---

# Some Useful Definitions

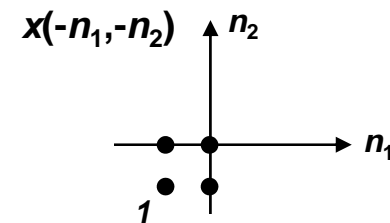
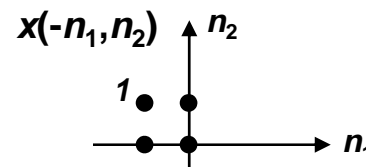
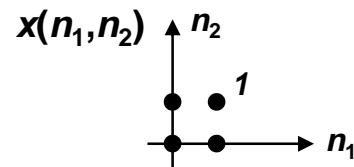
- A 2-D signal  $x(n_1, n_2)$  is **separable** if  $x(n_1, n_2) = f(n_1)g(n_2)$ .
- A  $M$ -Dimensional signal is **separable** if  $x(n_1, n_2, \dots, n_M) = f_1(n_1) \cdot f_2(n_2), \dots, f_M(n_M)$ .
- A **finite-extent** signal is a signal with a finite number of non-zero samples.
- **Region of support of a signal:**
  - If  $R$  is the region of support of a signal  $x(n_1, n_2)$ , then  $x(n_1, n_2) = 0$  for  $(n_1, n_2) \notin R$  (i.e., outside  $R$ ).
  - The region of support of a signal is the set of points where it can be nonzero.

# Basic 2D Operations

- Shifting

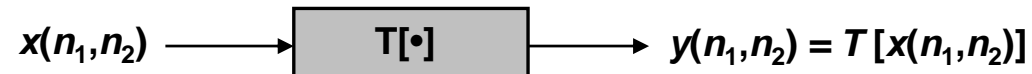


- Flipping



---

## 2-D Systems



### Why LSI Systems?

- Most frequently used
- Easy to design and analyze
- Need to simplify
- Mathematically tractable.

---

## 2-D LSI Systems

- Linear  $\Rightarrow$  superposition principle holds

$$\sum_i a_i x_i(n_1, n_2) \longrightarrow \boxed{\text{T}} \longrightarrow \sum_i a_i y_i(n_1, n_2), \quad \text{where } y_i(n_1, n_2) = T[x_i(n_1, n_2)]$$

- Shift-invariant  $\Rightarrow$  Shift in the input generates same shift in corresponding output

$$x(n_1 - s_1, n_2 - s_2) \longrightarrow \boxed{\text{T}} \longrightarrow y(n_1 - s_1, n_2 - s_2), \quad \text{where } y(n_1, n_2) = T[x(n_1, n_2)]$$

---

# 2D LSI Systems

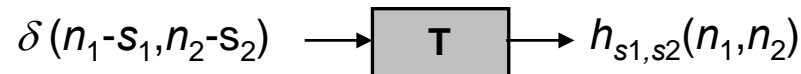
□ Note: System is characterized by its impulse response.

- Impulse response

➤ What is it?



- General system (Not SI)



- Shift Invariant (SI) system



---

## Output of a 2D LSI System: 2D Convolution Sum

- An arbitrary 2-D sequence can be decomposed into a linear combination of shifted impulses

$$x(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) \delta(n_1 - k_1, n_2 - k_2)$$

- The output of a general linear system is given by

$$y(n_1, n_2) = T[x(n_1, n_2)] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h_{k_1 k_2}(n_1, n_2)$$

where

$$h_{k_1 k_2}(n_1, n_2) = T[\delta(n_1 - k_1, n_2 - k_2)]$$

---

## 2D Convolution Sum

- The output of an LSI system is then the same linear combination of shifted impulse responses

$$\begin{aligned}y(n_1, n_2) &= T[x(n_1, n_2)] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) T[\delta(n_1 - k_1, n_2 - k_2)] \\ &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2)\end{aligned}$$

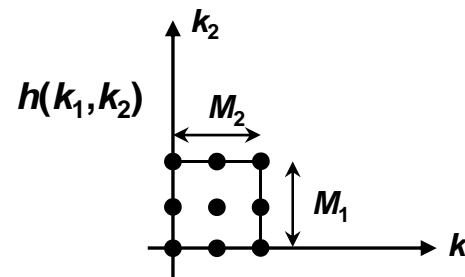
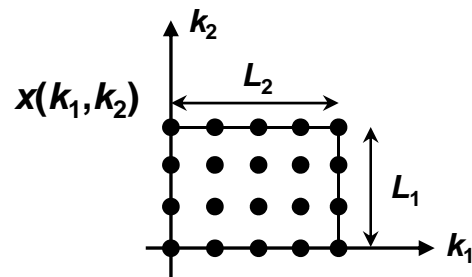
- Conceptually: Same as 1-D
- Mathematically: a lot more work
- Notation for 2D Convolution:  $y(n_1, n_2) = x(n_1, n_2) ** h(n_1, n_2)$

# 2D Convolution Sum

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2)$$

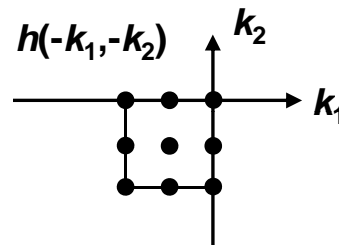
Notation:  $y = x ** h$

## Example



# 2D Convolution Sum

- $h(-k_1, -k_2)$  corresponds to a 180° rotated version of  $h(k_1, k_2) \Rightarrow$  reflect flip around  $k_1$  axis and then  $k_2$  axis (or vice-versa)

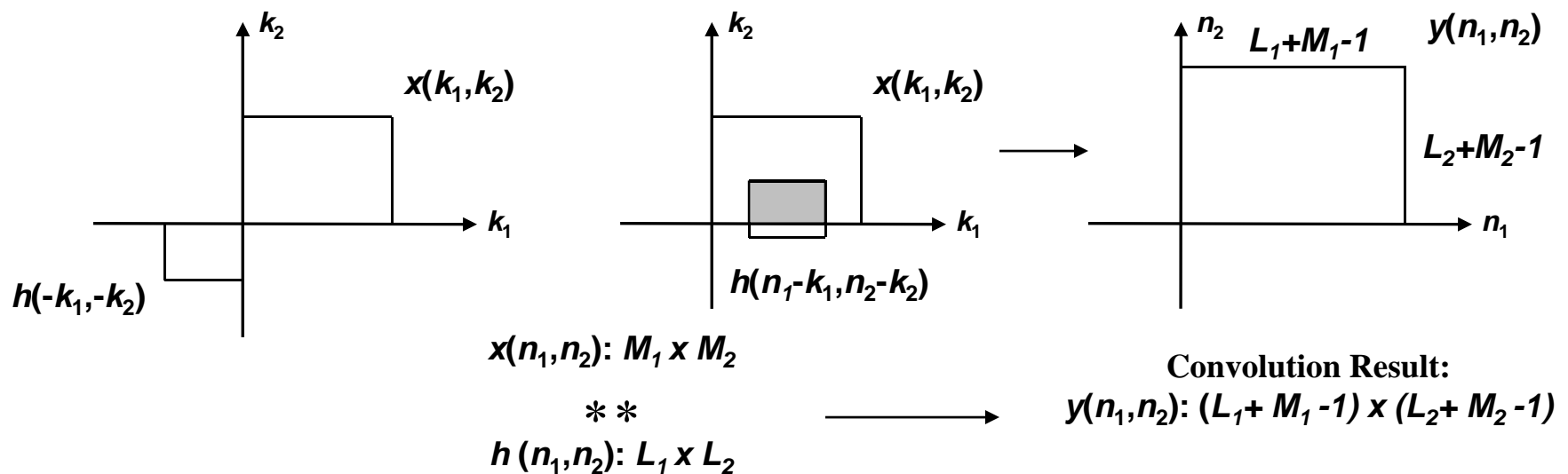


- $h(n_1 - k_1, n_2 - k_2)$  is a 180° rotated and shifted version of  $h(k_1, k_2)$



# 2D Convolution Sum

- Convolution:
  - Flip, slide to  $(n_1, n_2)$ , multiply and sum
  - Repeat for other desired values of  $(n_1, n_2)$



---

# Convolution Steps

1. Reflect  $h$  about to both  $k_1$  and  $k_2$ .
2. Translate so that sample  $h(0,0)$  lies at the point  $(n_1, n_2)$ .
3. Multiply sequences  $x(k_1, k_2)$  and  $h(n_1 - k_1, n_2 - k_2)$ .
4. Sum non-zero samples of obtained product sequence to compute the output sample  $y(n_1, n_2)$ .
5. Vary  $(n_1, n_2)$  and repeat from Step 2.

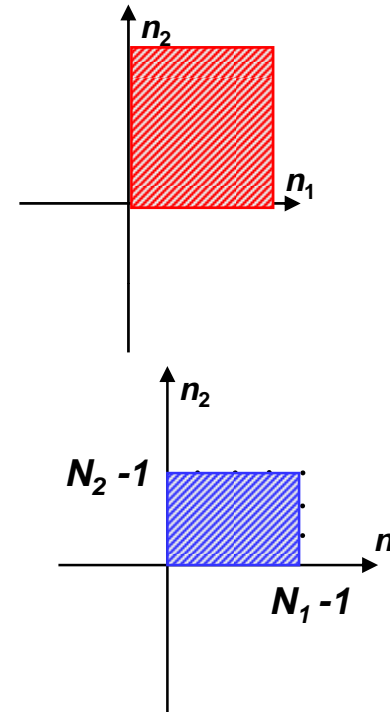
## 2D Convolution Sum

Example:

$$h(n_1, n_2) = u(n_1, n_2) = u(n_1) \cdot u(n_2)$$

$$x(n_1, n_2) = \begin{cases} 1, & 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1 \\ 0, & \text{otherwise} \end{cases}$$

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2)$$

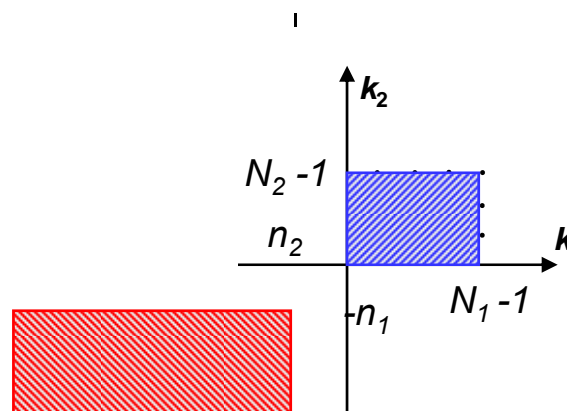


## 2D Convolution Sum - Example

- The shape of the product sequence can assume one of five different forms depending on the values of  $(n_1, n_2)$

Case #1:  $n_1 < 0$  or  $n_2 < 0$   $x(k_1, k_2) \cdot h(n_1 - k_1, n_2 - k_2) = 0$

$$y(n_1, n_2) =$$

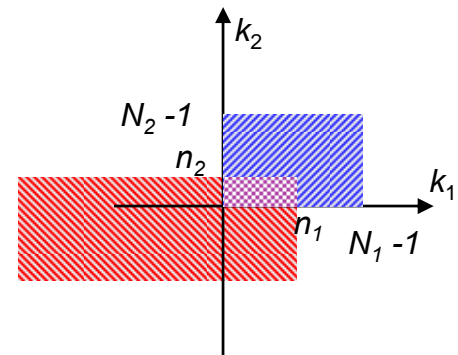


---

## 2D Convolution Sum - Example

Case #2:  $0 \leq n_1 \leq N_1-1, 0 \leq n_2 \leq N_2-1$

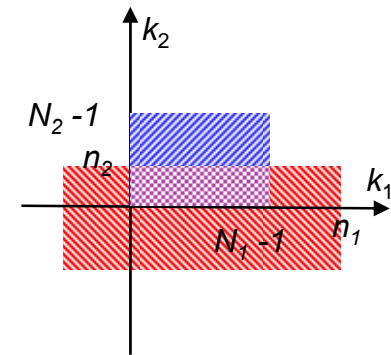
$$y(n_1, n_2) =$$



## 2D Convolution Sum - Example

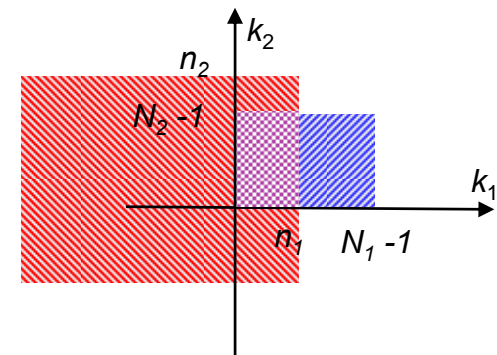
Case #3:  $n_1 \geq N_1 - 1, 0 \leq n_2 \leq N_2 - 1$

$$y(n_1, n_2) =$$



Case #4:  $0 \leq n_1 \leq N_1 - 1, n_2 \geq N_2 - 1$

$$y(n_1, n_2) =$$

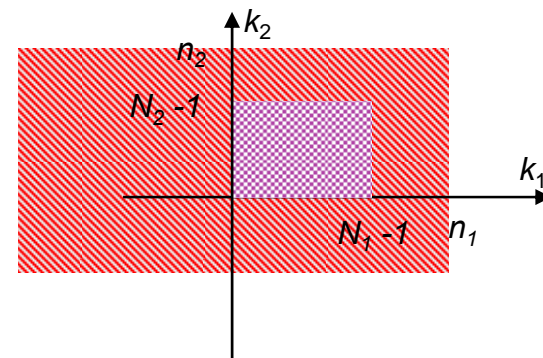


---

## 2D Convolution Sum - Example

Case #5:  $n_1 \geq N_1 - 1$ ,  $n_2 \geq N_2 - 1$

$$y(n_1, n_2) =$$



---

## 2D Convolution Sum - Example

Final result:

$$y(n_1, n_2) =$$

---

## 2D Convolution Sum

- Summary of 2D Convolution steps (Graphical Method):
  - Reflect  $h(k_1, k_2)$  about both  $k_1$  and  $k_2$  axes
  - Translate so that sample  $h(0,0)$  lies at the point  $(n_1, n_2)$
  - Multiply the sequences  $x(k_1, k_2)$  and  $h(n_1 - k_1, n_2 - k_2)$  point-by-point
  - Sum non-zero samples of obtained product sequence to compute the output sample  $y(n_1, n_2)$
  - Vary  $(n_1, n_2)$  and repeat from Step 2
- Direct evaluation of the 2D convolution sum can be sometimes easier than using the graphical method.
- Convolution sum is directly used or is used in combination with the graphical method when both or one of the signals cannot be easily represented graphically.

---

## 2D Convolution Sum

- Convolution with a 2D impulse:

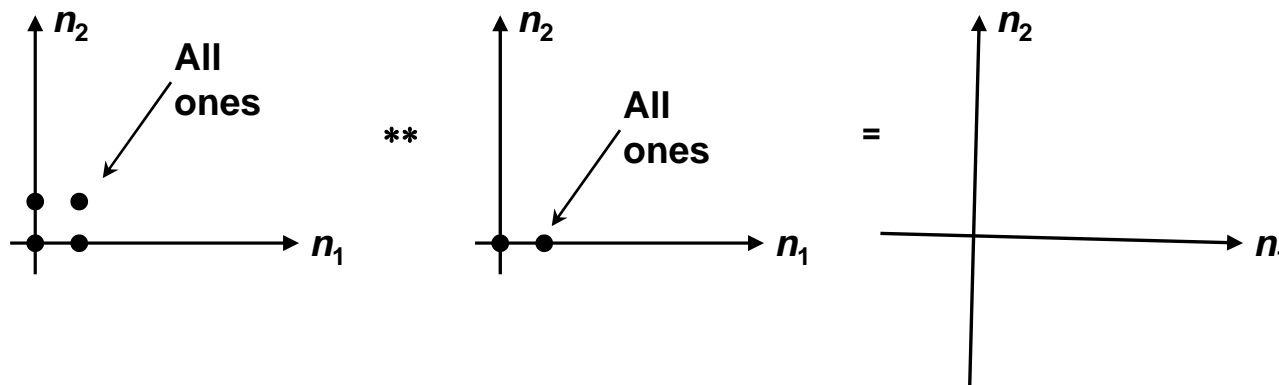
$$x(n_1, n_2) ** \delta(n_1 - k_1, n_2 - k_2) = ?$$

## 2D Convolution Sum

- Convolution with a 2D Impulse:

$$x(n_1, n_2) ** \delta(n_1 - k_1, n_2 - k_2) =$$

### 2D Convolution Example:

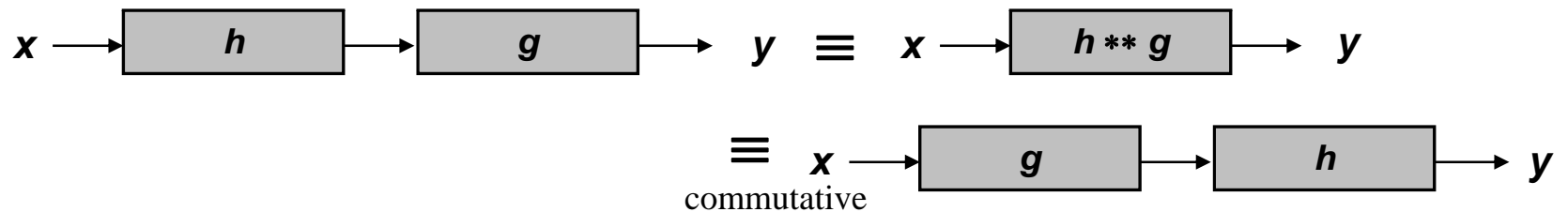


# 2D Convolution Sum

- **Properties of 2D convolution:**

- $x ** h = h ** x \Rightarrow$  commutative

- $(x ** h) ** g = x ** (h ** g) \Rightarrow$  associative



- $x ** (h + g) = x ** h + x ** g \Rightarrow$  distributive

