Machine Learning in Imaging

BME 590L
Roarke Horstmeyer

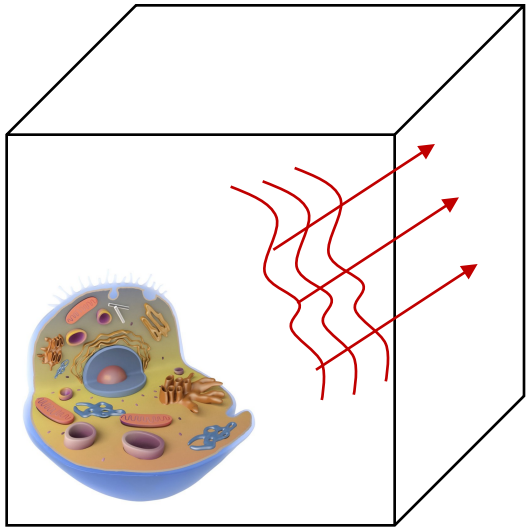
Lecture 3: Mathematical preliminaries for discrete functions

Labs & coding assignment notes

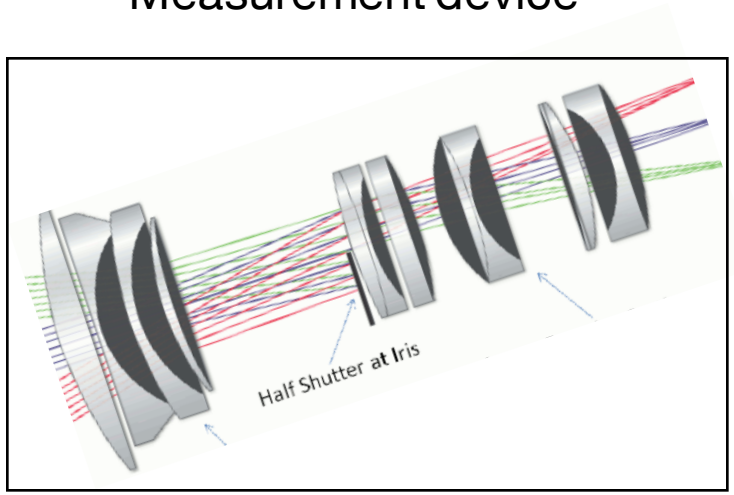
- Ouwen has updated the instructions for setting up Jupyter & Google Cloud:
    - https://github.com/BME-590-Machine-Learning-in-Imaging
- Extra TA office hours to help with this setup **@ Twinnies 4-6pm this Friday**
- **No lab on Monday 1/21** – Come on Wednesday (Hudson 208) or Friday (TBD)
- Main tasks needed now:
    1) Simple programming interface with Python
    2) Be able to run Jupyter locally, get comfortable with it (editing, saving)
    3) Be able to upload (‘commit’) code to your Github account & share link
    4) (soon) Run code in Google cloud via Jupyter notebook locally

ML+Imaging pipeline introduction

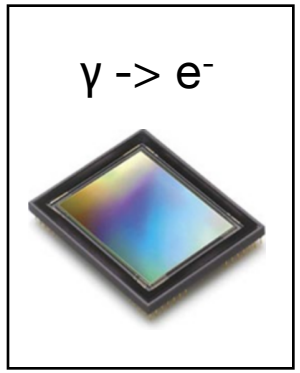
Real World



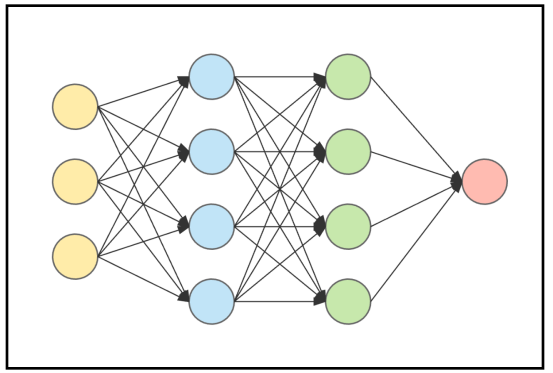
Measurement device



Digitization

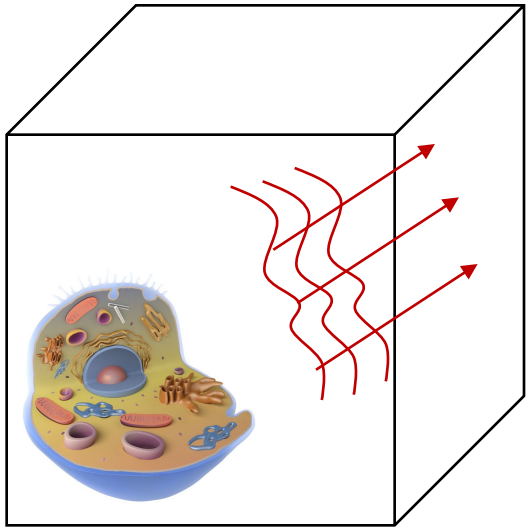


Machine Learning



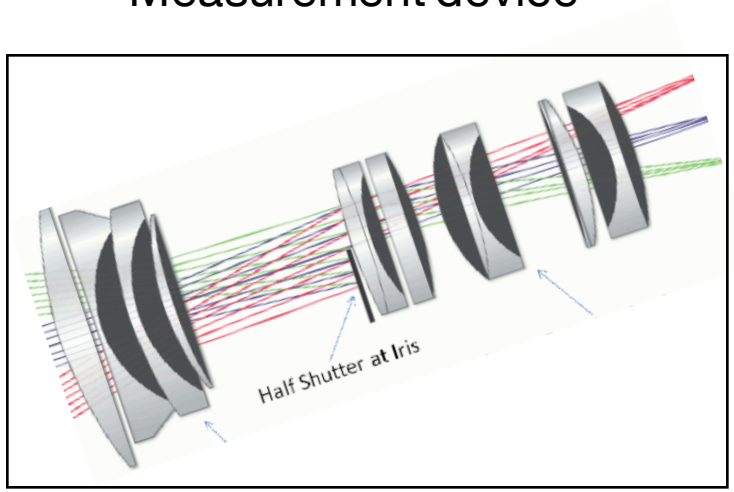
ML+Imaging pipeline introduction

Real World



Continuous complex fields

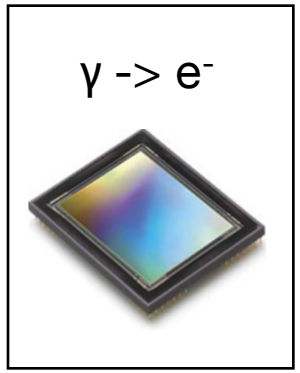
Measurement device



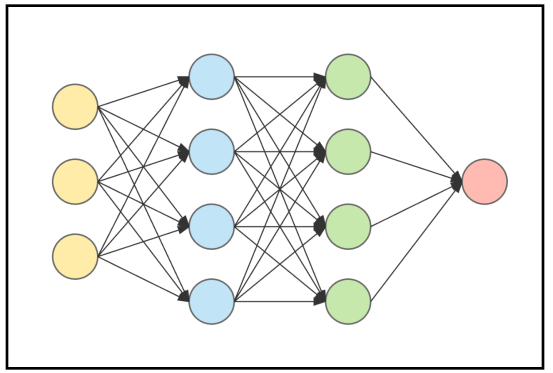
Black box transformations

- Convolution
- Fourier Transform

Digitization



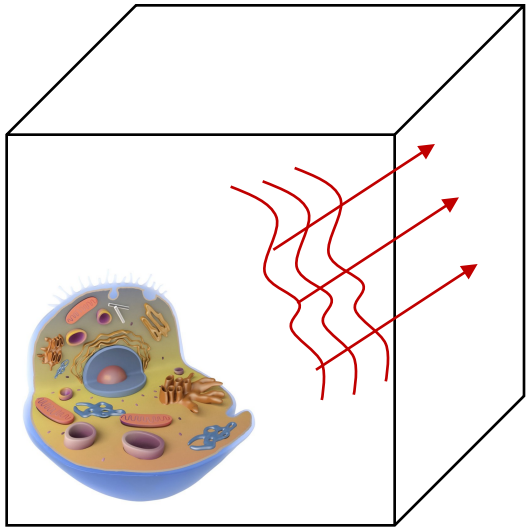
Machine Learning



(last class) → (last class, this class)

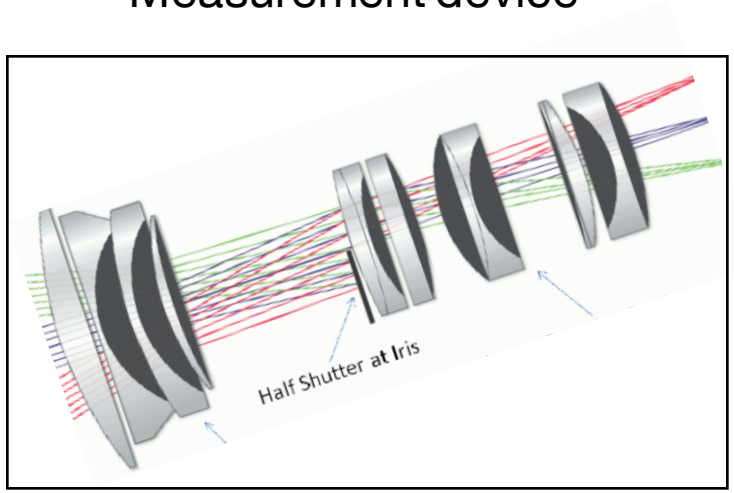
ML+Imaging pipeline introduction

Real World



Continuous complex fields

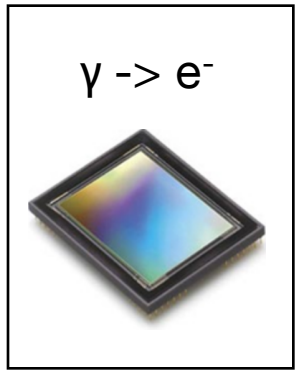
Measurement device



Black box transformations

- Convolution
- Fourier Transform

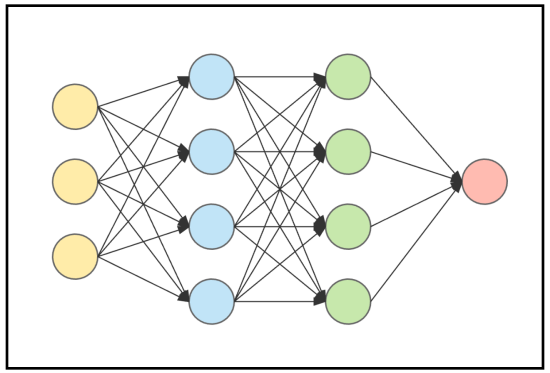
Digitization



Sampling Theorem

Discrete math &
Linear algebra

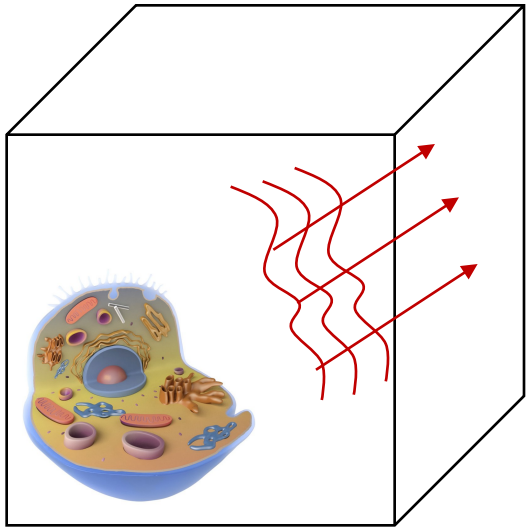
Machine Learning



(last class) → (last class, this class) → (this class)

ML+Imaging pipeline introduction

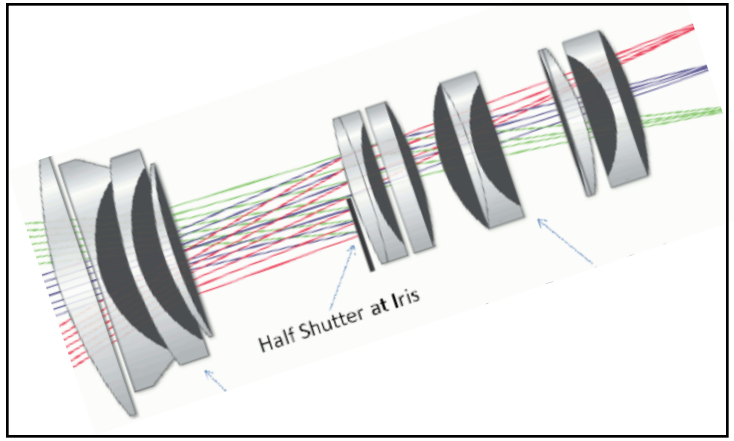
Real World



Continuous complex fields

(last class)

Measurement device

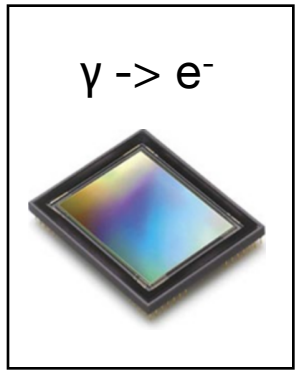


Black box transformations

- Convolution
- Fourier Transform

(last class, this class)

Digitization

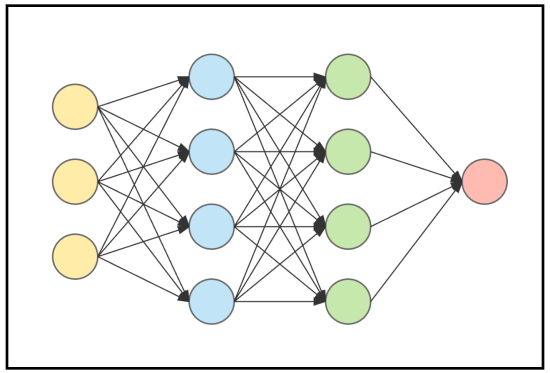


Sampling Theorem

Discrete math & Linear algebra

(this class)

Machine Learning



Optimization

Linear classification

Logistic classifier

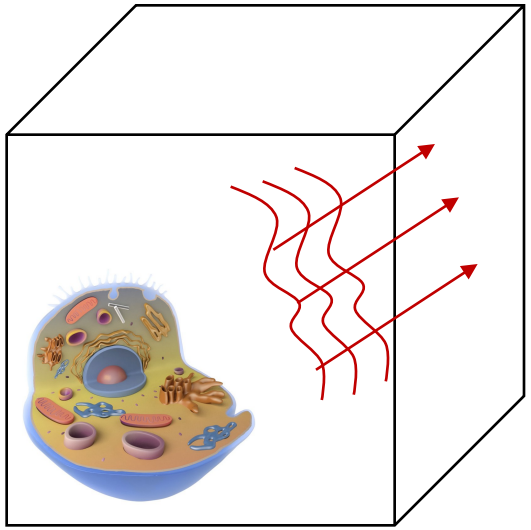
Neural networks

Convolutional NN's

(next few weeks)

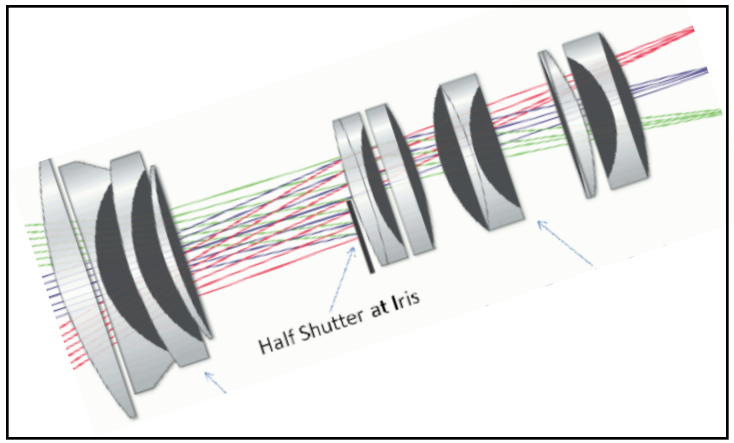
ML+Imaging pipeline introduction

Real World



Continuous complex fields

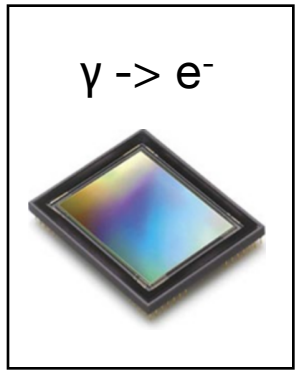
Measurement device



Black box transformations

- Convolution
- Fourier Transform

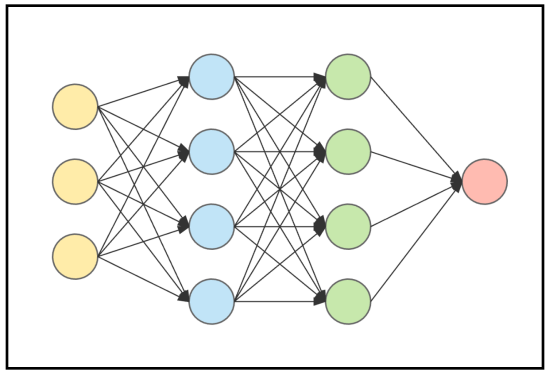
Digitization



Sampling Theorem

Discrete math & Linear algebra

Machine Learning



Optimization

Linear classification

Logistic classifier

Neural networks

Convolutional NN's

(last class) → (last class, this class) → (this class) → (next few weeks)

March → April

Review: signals as complex fields

Simplification #1: Let's forget about light changing as a function of time. It does so way too fast, and way too slow:

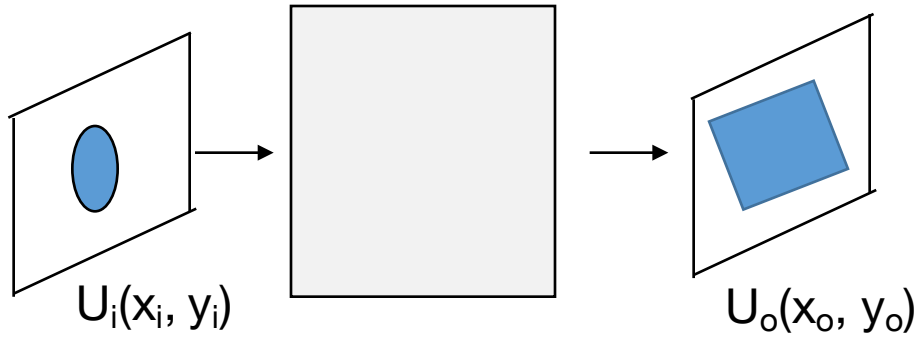
$$A(\mathbf{r})\cos(\mathbf{kr} - \omega t) \rightarrow A(\mathbf{r})\cos(\mathbf{kr})$$

Simplification #2: We'll use complex numbers when required, it'll make our lives easier. This leads to the *complex* field, U(**r**):

$$A(\mathbf{r})\cos(\mathbf{kr}) \leftrightarrow \boxed{A(\mathbf{r})\, e^{i\mathbf{k}\cdot\mathbf{r}} = U(\mathbf{r})}$$

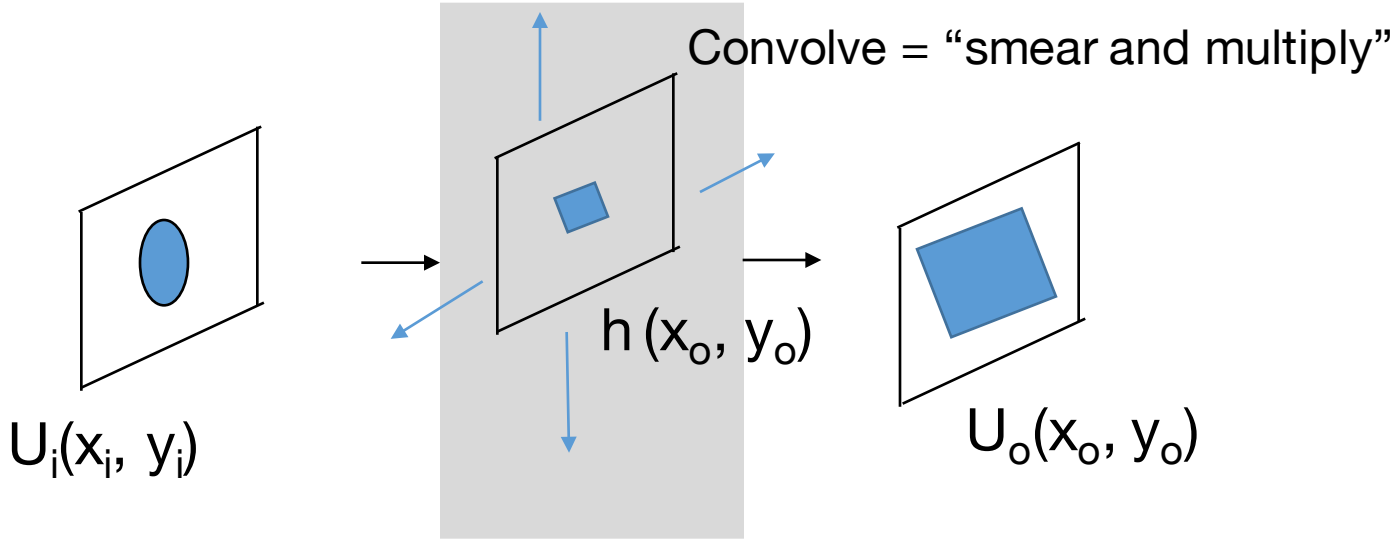
Simplification #3: Just consider mappings between planes across space. This is a critically important way of thinking for optics. Think "index card 1 to index card 2".

$$U(\mathbf{r}) \rightarrow U(x,y)$$



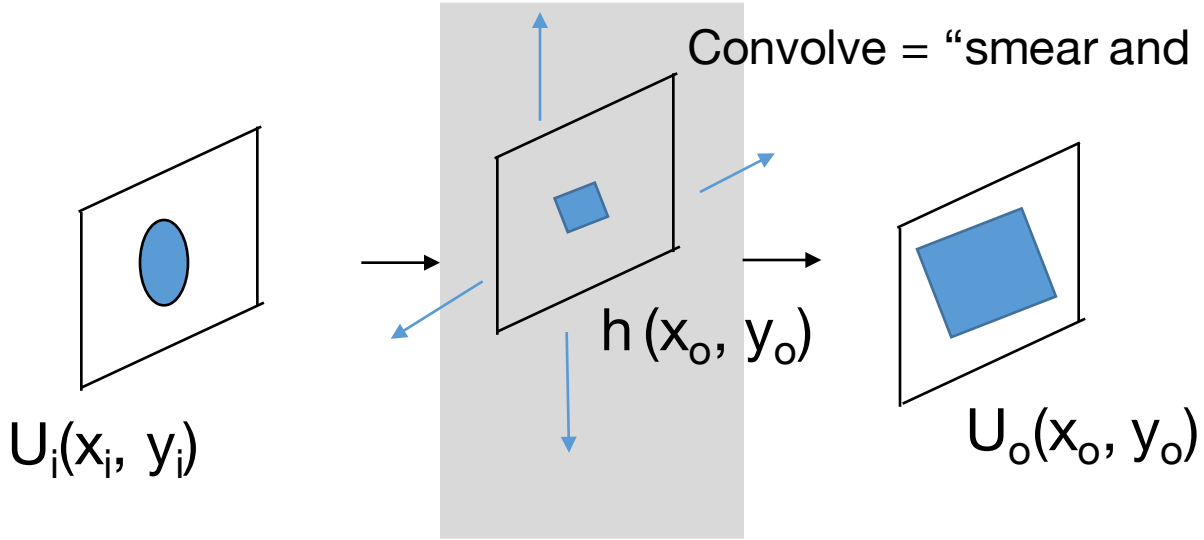
Review: black box transforms as a convolution

Knowing the point-spread function, it is direct to model any output of the black box, given an input:



Review: black box transforms as a convolution

Knowing the point-spread function, it is direct to model any output of the black box, given an input:



$$U_o(x_o, y_o) = \iint_{-\infty}^{\infty} U_i(x_i, y_i) h(x_o - x_i, y_o - y_i) dx_i dy_i$$

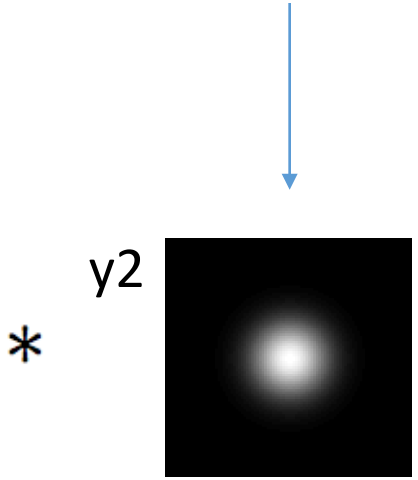
Output of linear system is a convolution of the input with its point-spread function

2D convolution example

High-res. real-world object

$U_1(x,y)$

Blur caused by camera lens



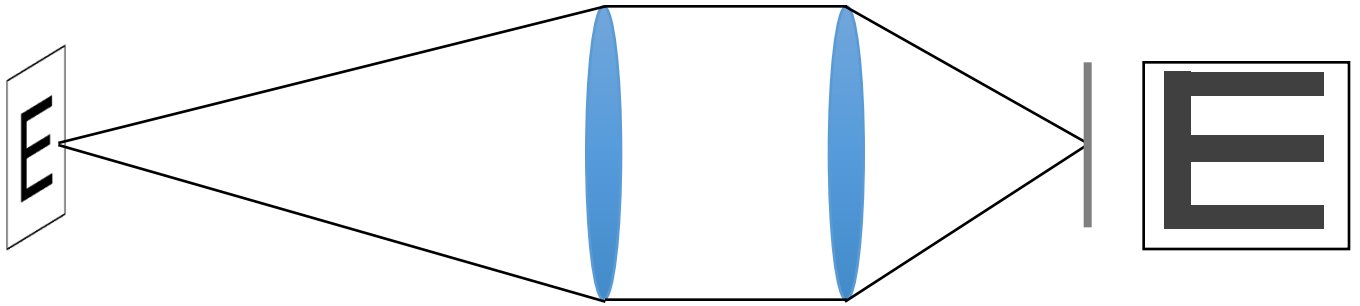
*

=

Image at camera sensor plane

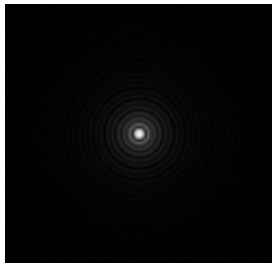
$U_0(x,y)$

Optical modification Ex. #1: The cubic phase mask

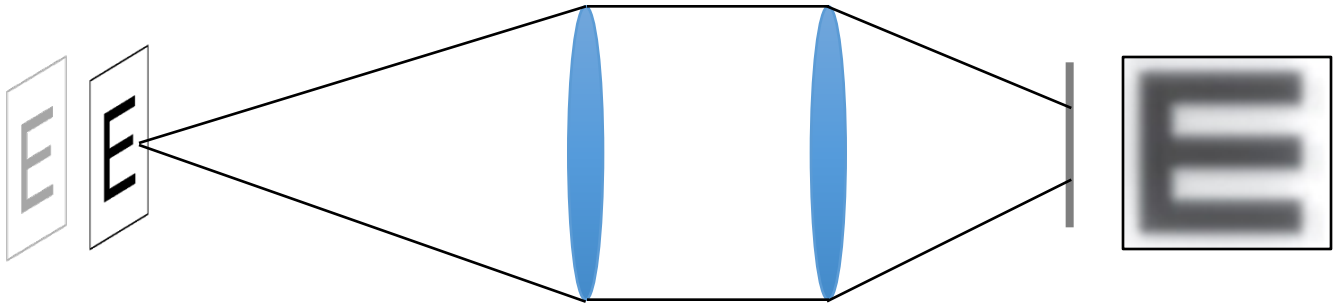


Standard camera
Point-spread function

in focus

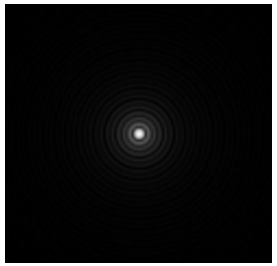


Optical modification Ex. #1: The cubic phase mask

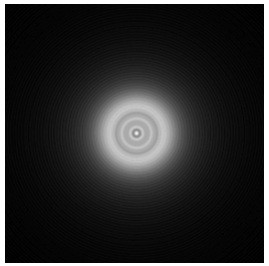


Standard camera:
Limited depth-of-field

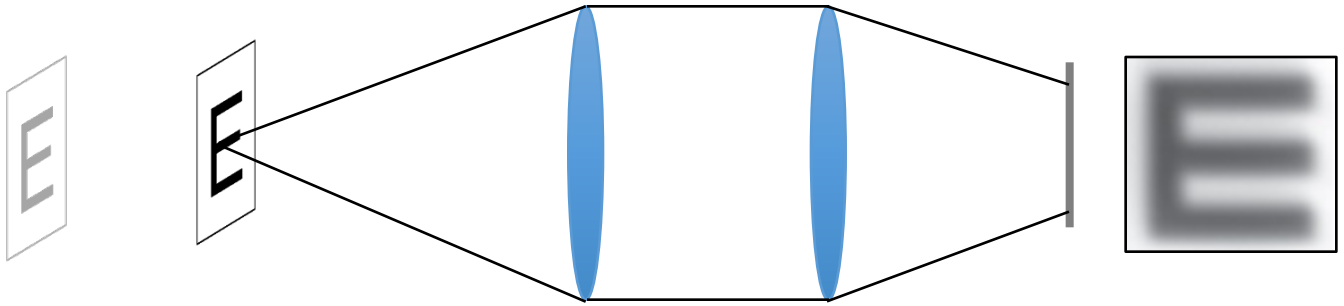
in focus



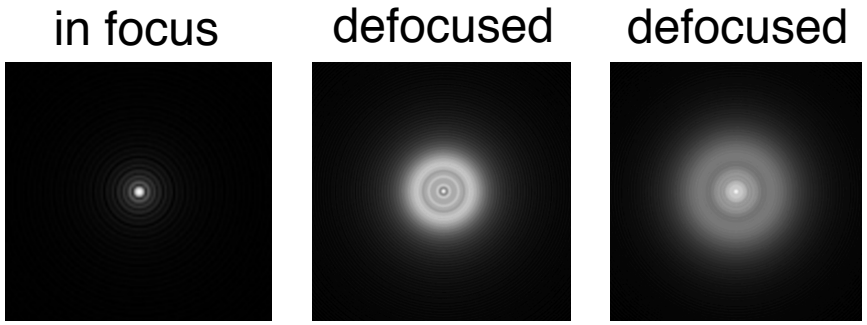
defocused



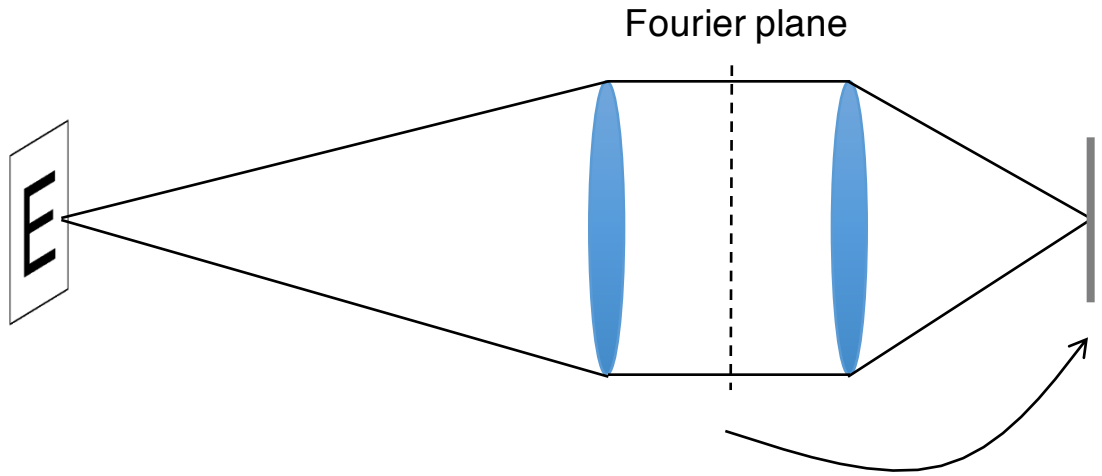
Optical modification Ex. #1: The cubic phase mask



Standard camera:
Limited depth-of-field

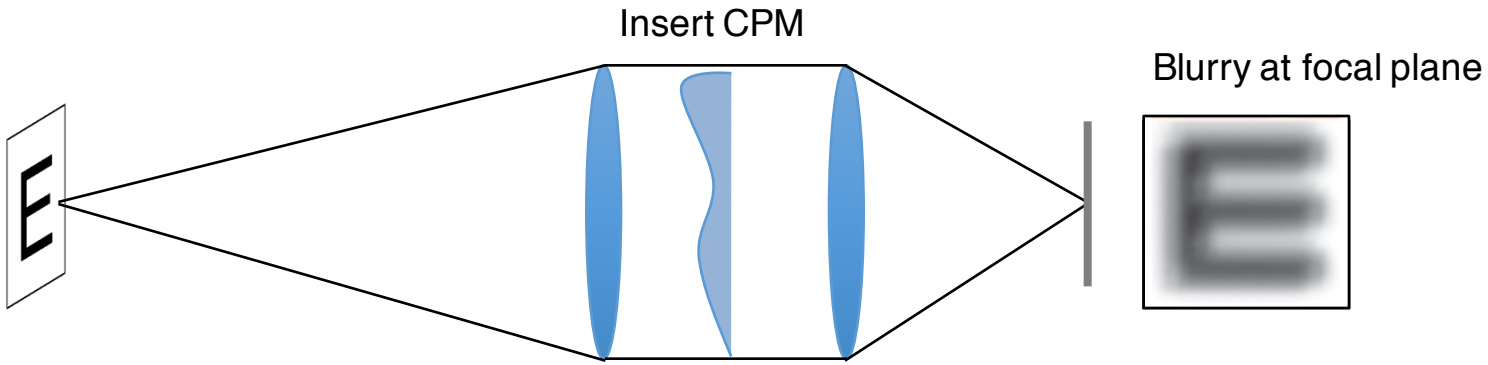


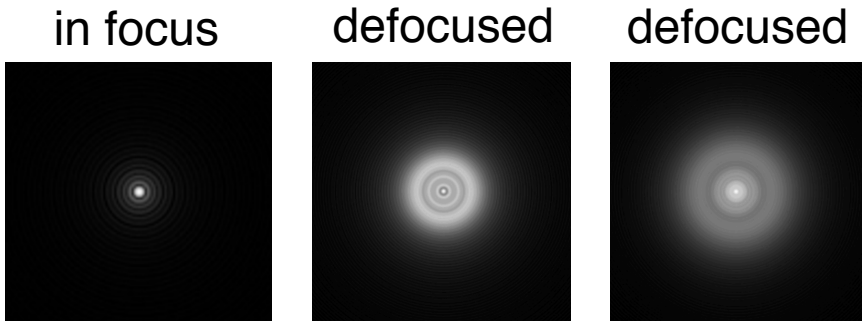
Optical modification Ex. #1: The cubic phase mask



Blur proportional to Fourier transform of shape of aperture

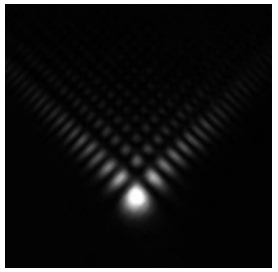
Optical modification Ex. #1: The cubic phase mask





Standard camera:
Limited depth-of-field

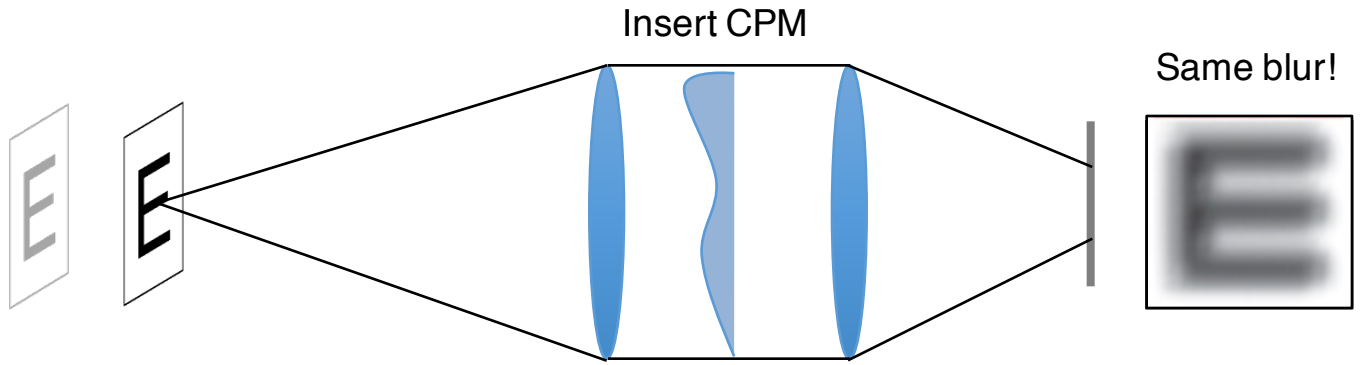
Cubic phase mask:
extended depth-of-field

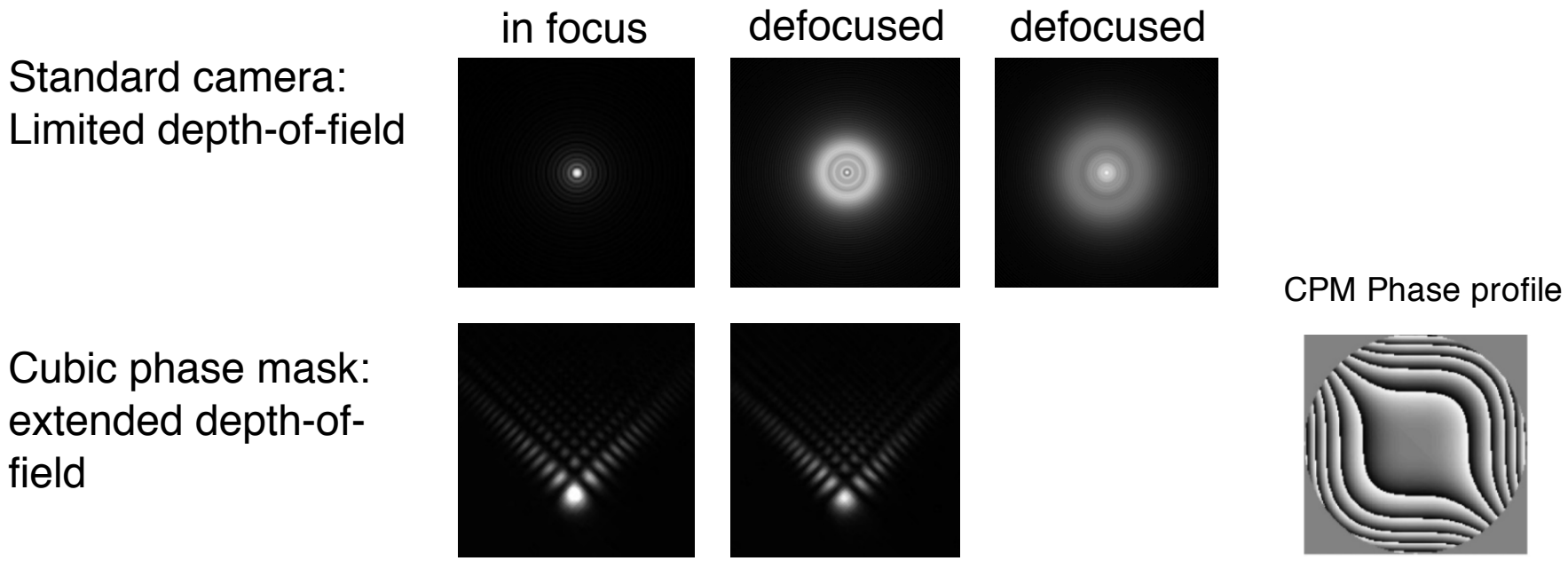


CPM Phase profile

E. Dowski and W. T Cathey, "Extended depth of field through wave-front coding," Appl Opt. 1994

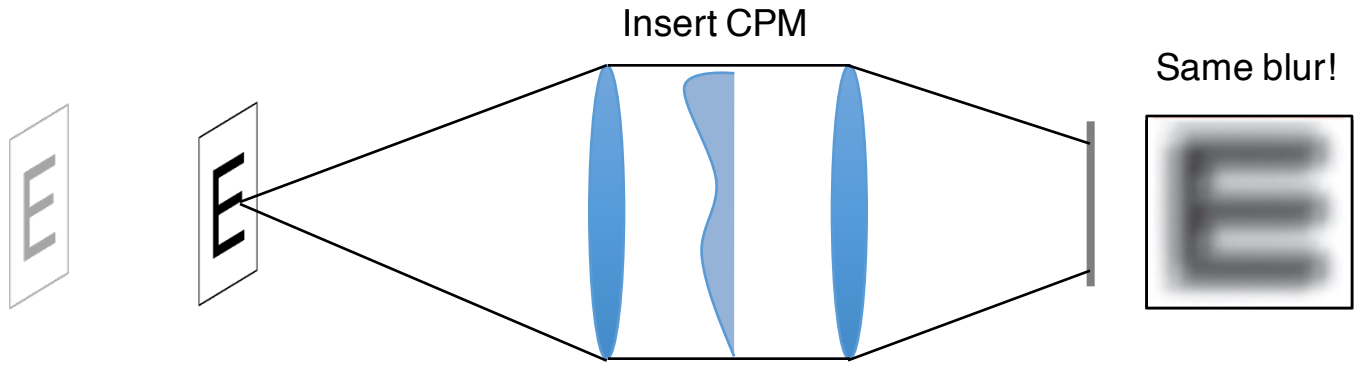
Optical modification Ex. #1: The cubic phase mask

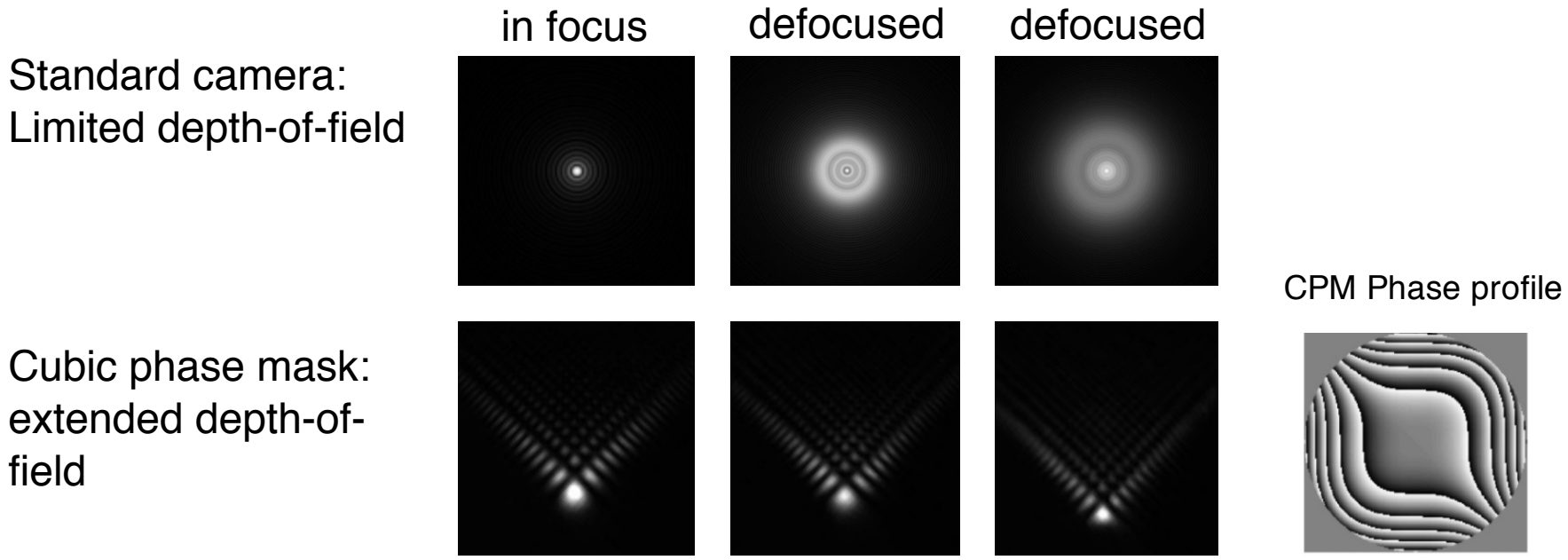




E. Dowski and W. T Cathey, "Extended depth of field through wave-front coding," Appl Opt. 1994

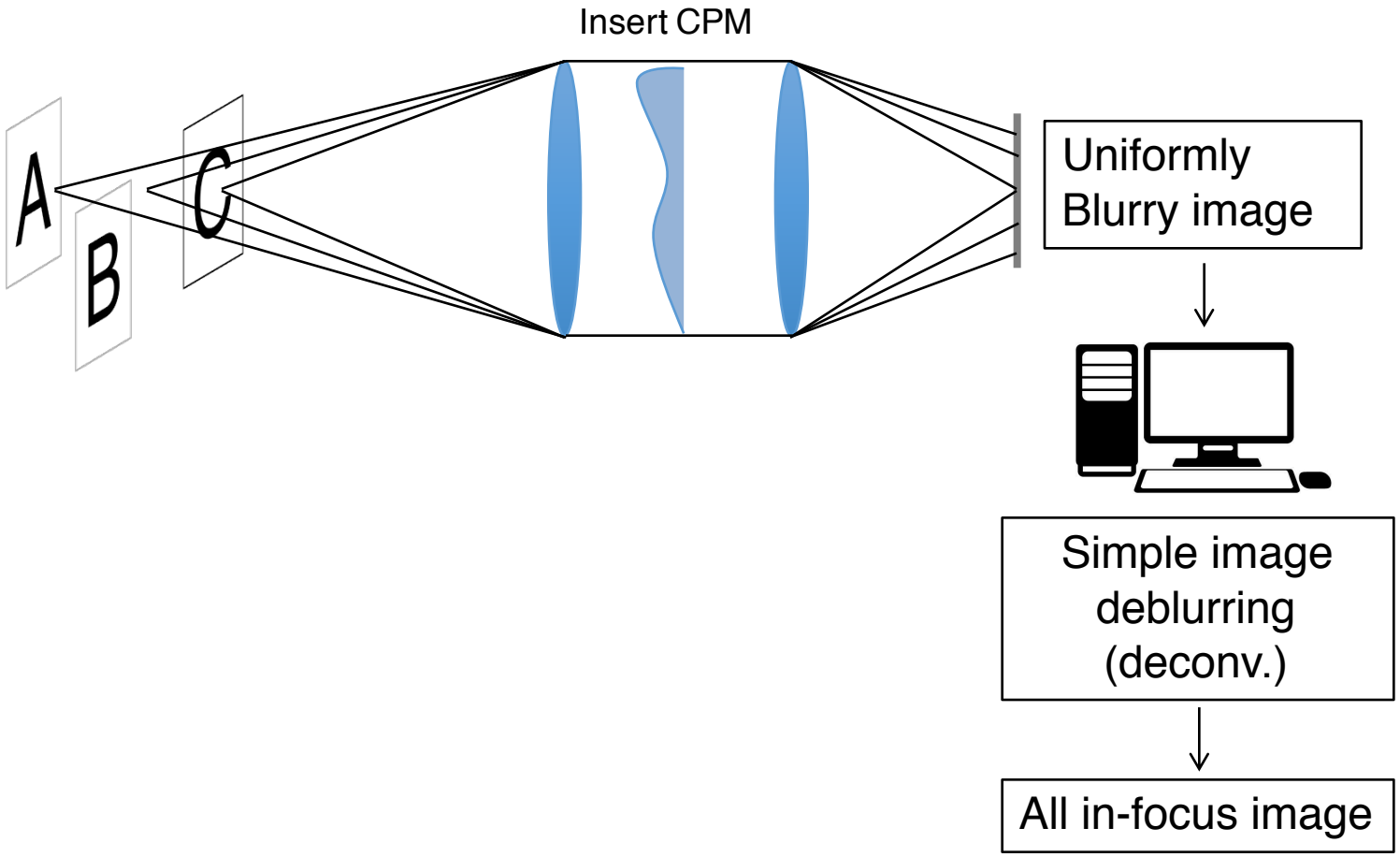
Optical modification Ex. #1: The cubic phase mask





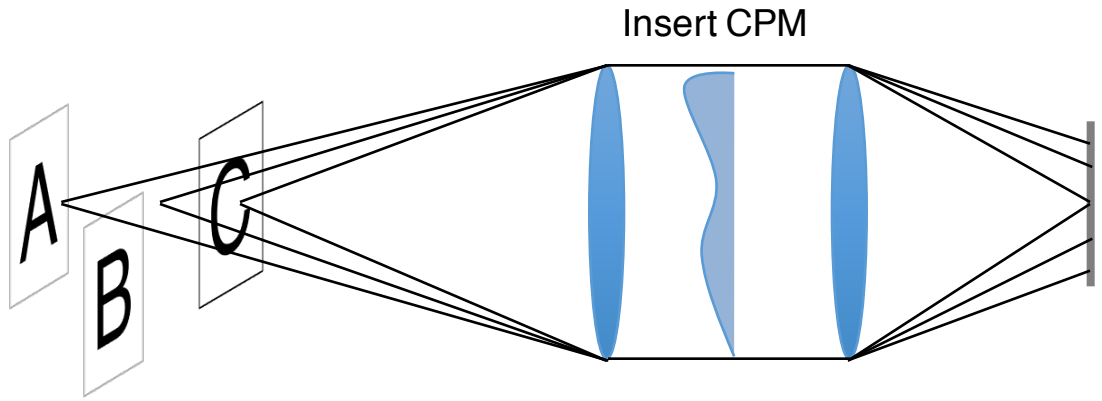
E. Dowski and W. T Cathey, "Extended depth of field through wave-front coding," Appl Opt. 1994

Optical modification Ex. #1: The cubic phase mask



E. Dowski and W. T Cathey, "Extended depth of field through wave-front coding," Appl Opt. 1994

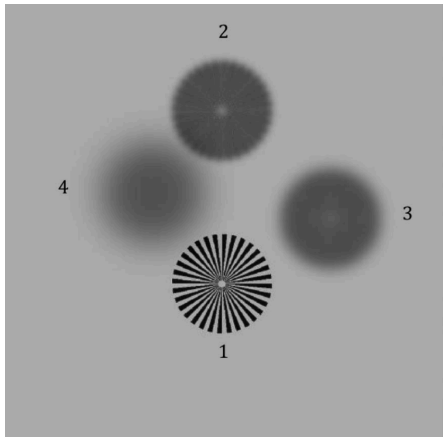
Optical modification Ex. #1: The cubic phase mask



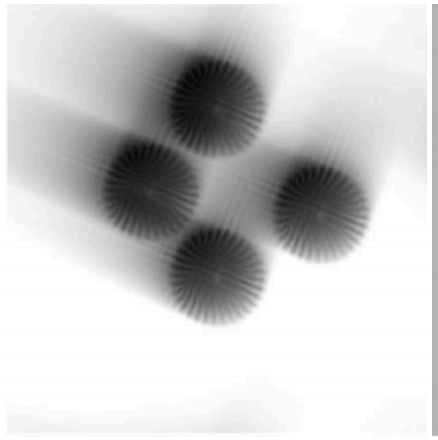
Uniformly Blurry image

Simple image deblurring (deconv.)

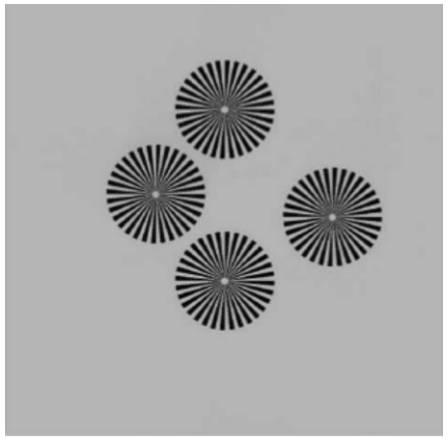
All in-focus image



Standard camera image



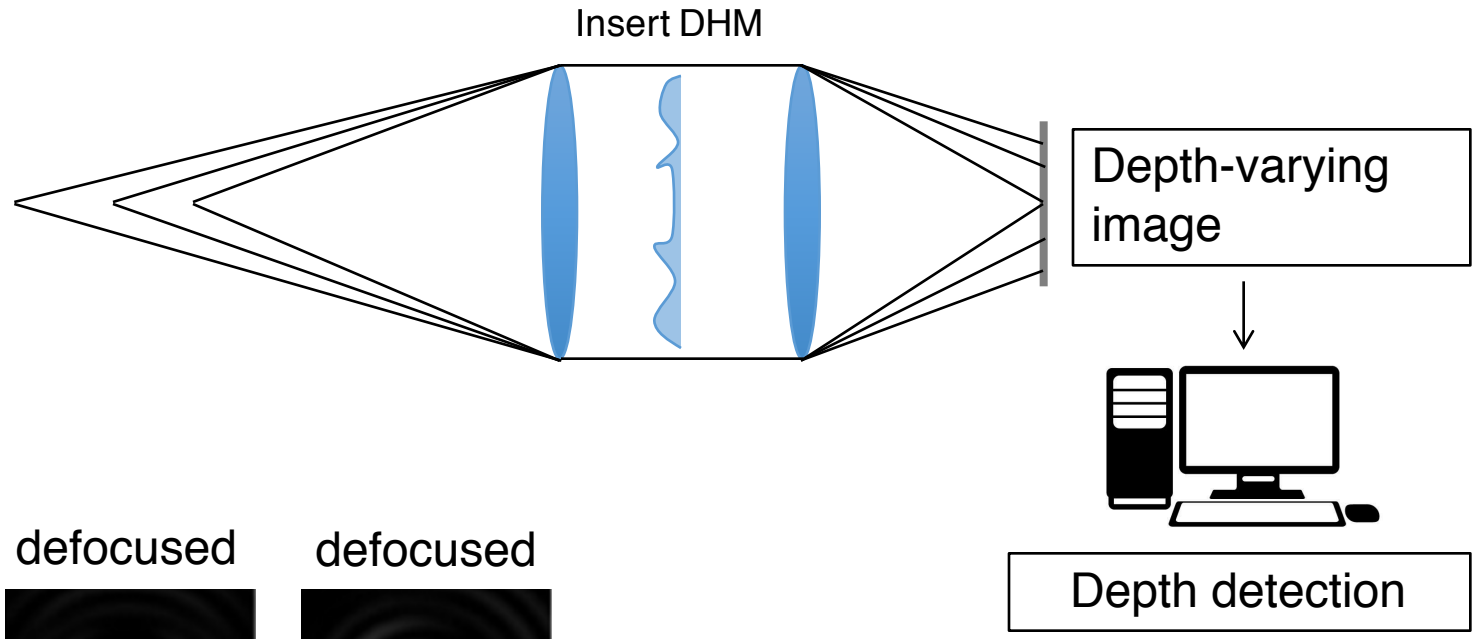
CPM image, raw

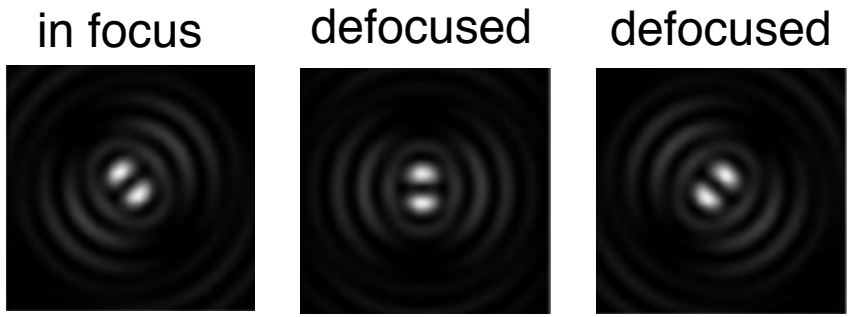


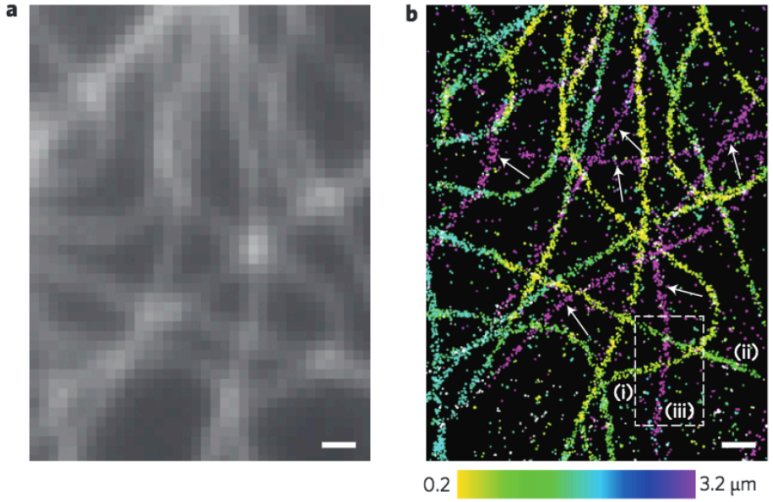
CPM image, deconvolved

E. Dowski and W. T Cathey, "Extended depth of field through wave-front coding," Appl Opt. 1994

Optical modification Ex. #1b: Double helix mask



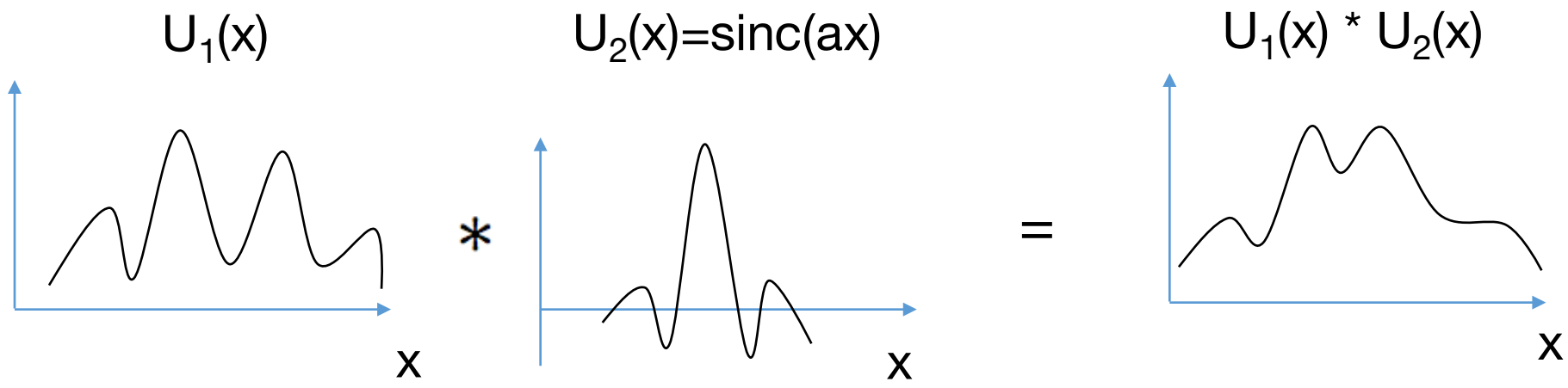




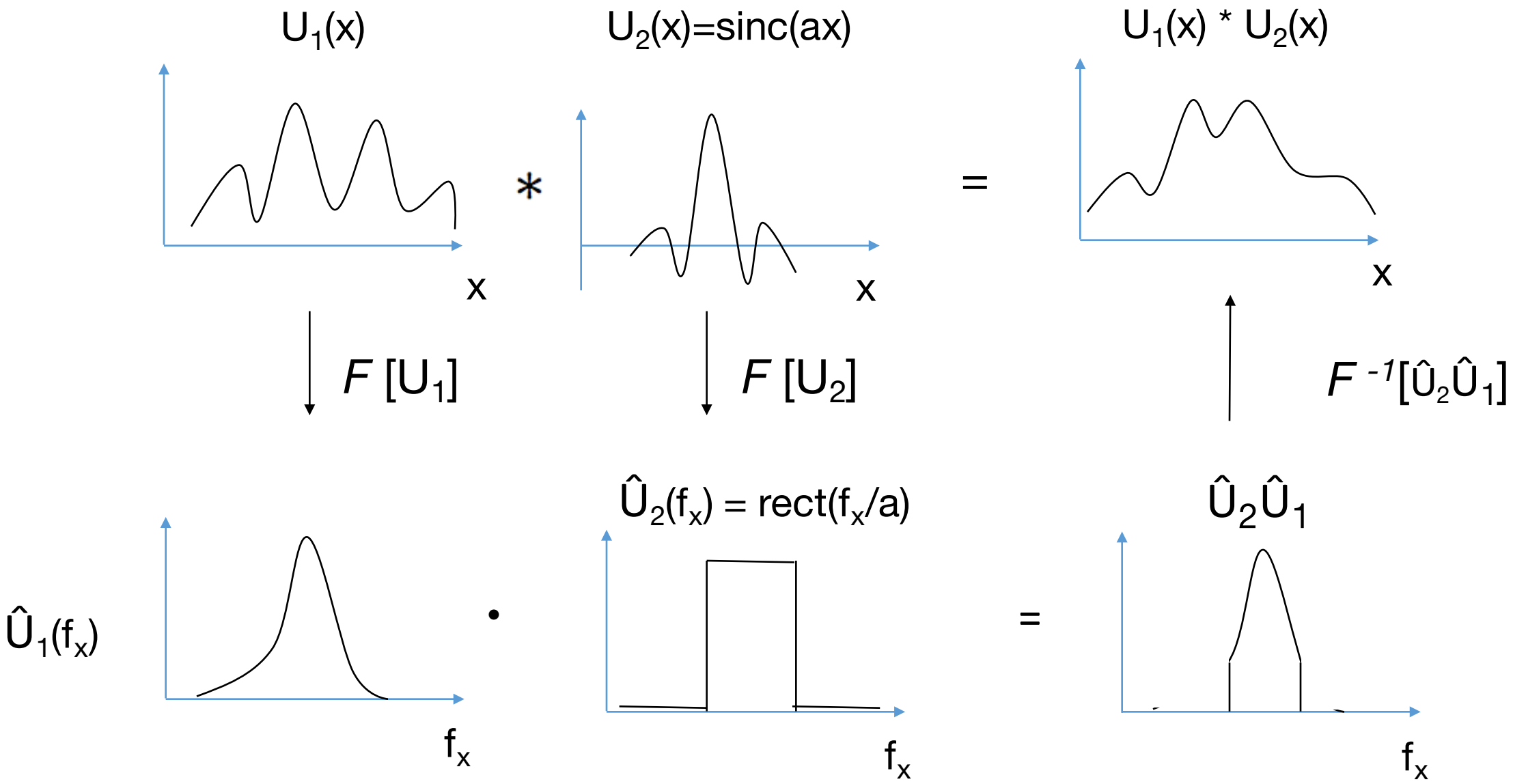
Moerner Lab
Nobel Prize in Chemistry, 2014

Jia et al., Nature Photonics 2014

Review: Convolution theorem

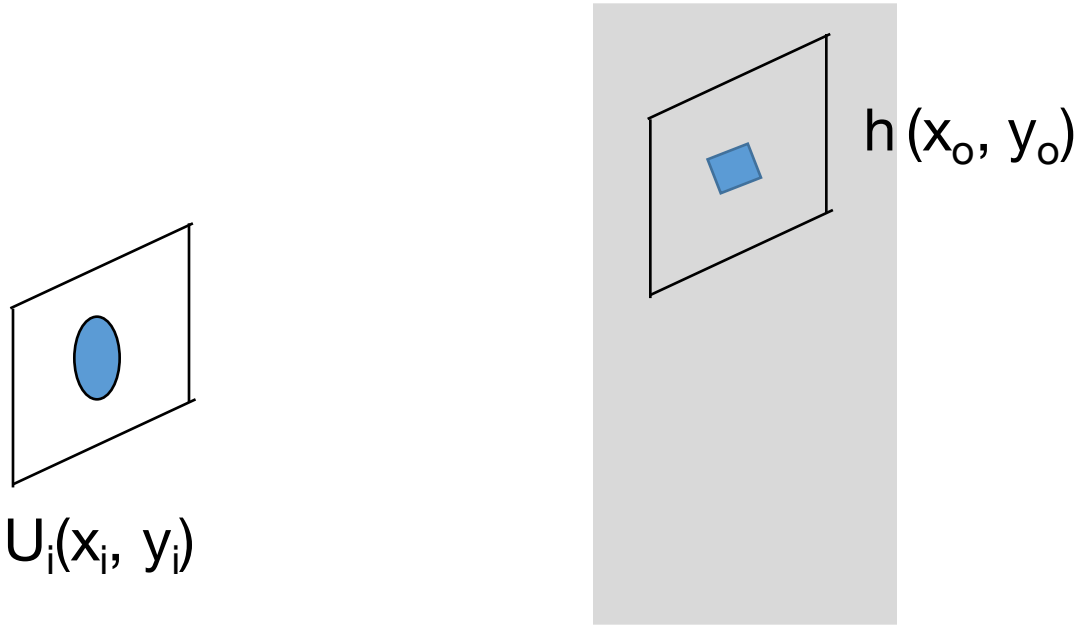


Review: Convolution theorem



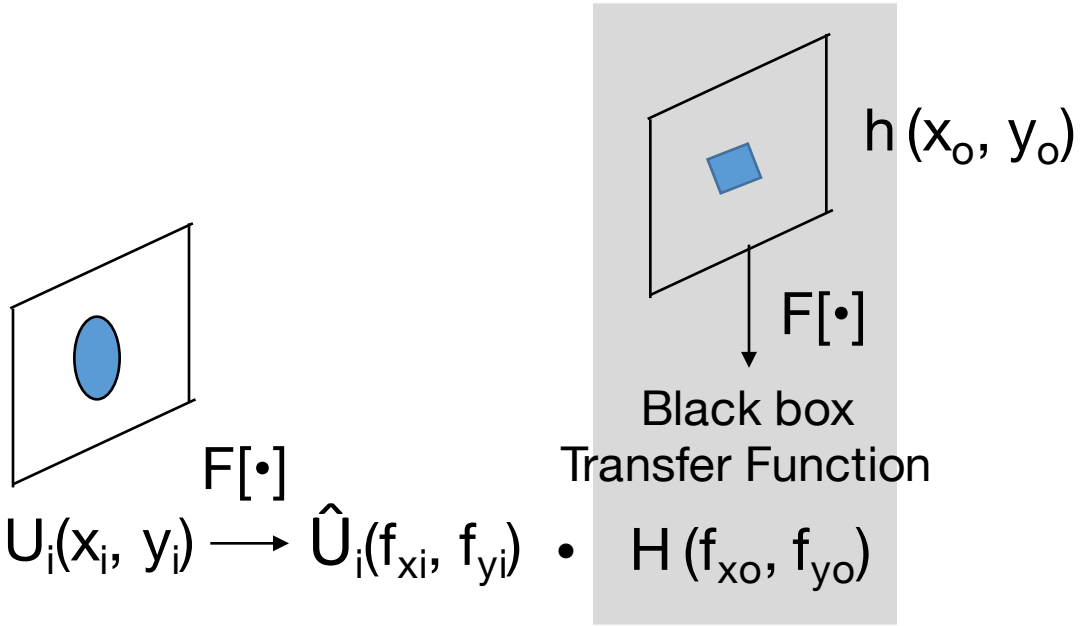
Review: black box transforms as a convolution

Knowing the point-spread function, it is direct to model any output of the black box, given an input:



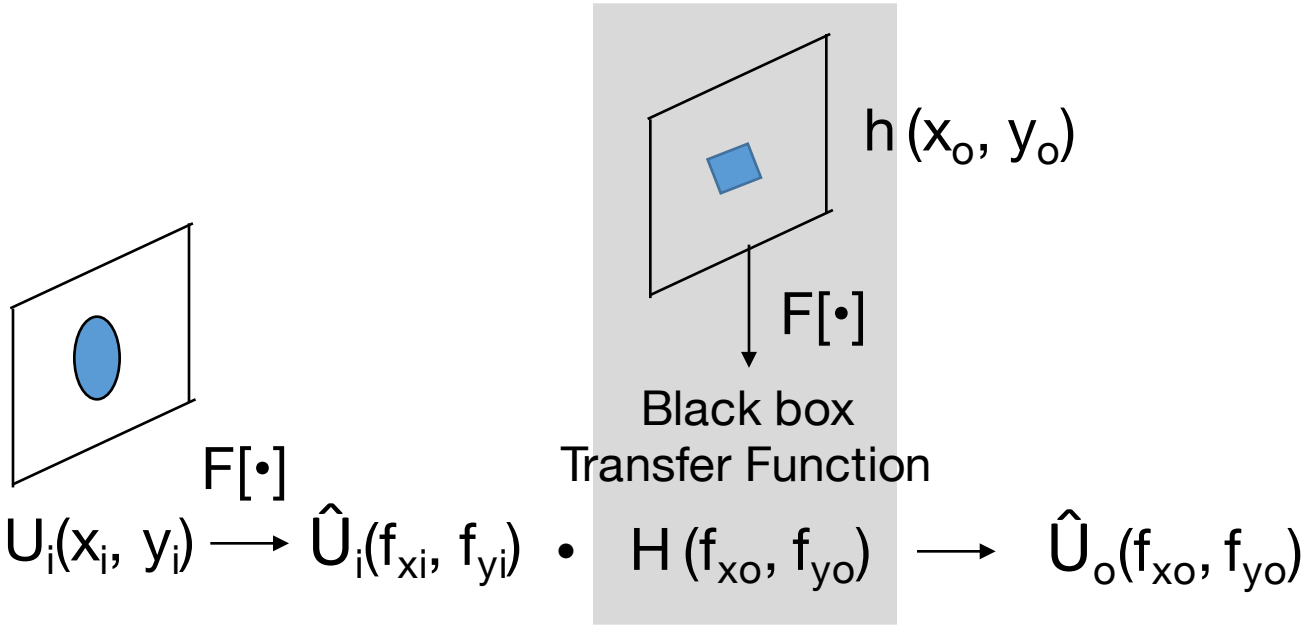
Review: black box transforms as a convolution

Knowing the point-spread function, it is direct to model any output of the black box, given an input:



Review: black box transforms as a convolution

Knowing the point-spread function, it is direct to model any output of the black box, given an input:

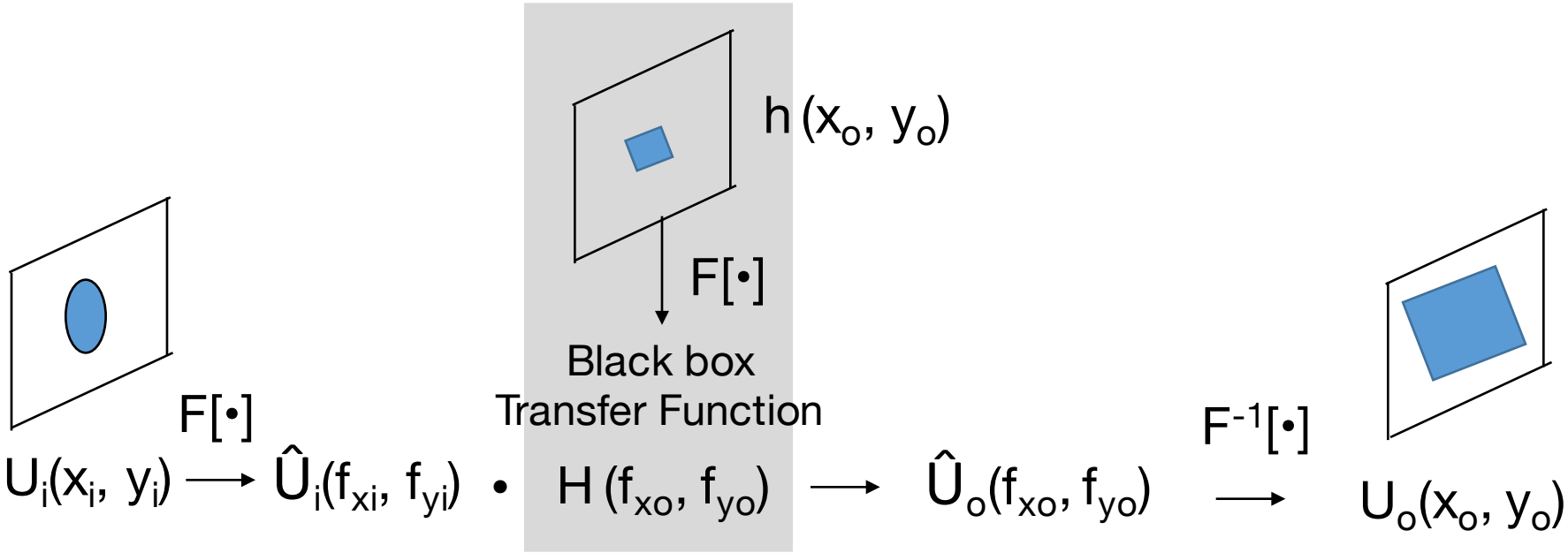


$$\hat{U}_o(f_x, f_y) = \hat{U}_i(f_x, f_y) H(f_x, f_y)$$

Can also multiply Fourier transform of input with transfer function H to obtain Fourier transform of output

Review: black box transforms as a convolution

Knowing the point-spread function, it is direct to model any output of the black box, given an input:

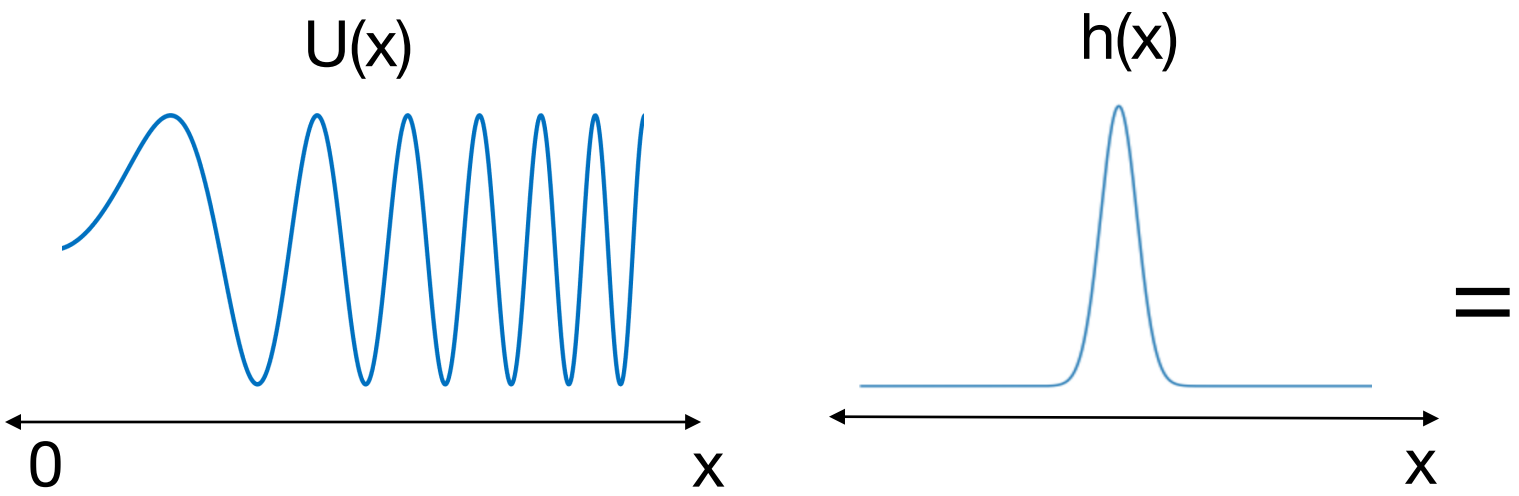


$$\hat{U}_o(f_x, f_y) = \hat{U}_i(f_x, f_y) H(f_x, f_y)$$

Can also multiply Fourier transform of input with transfer function H to obtain Fourier transform of output

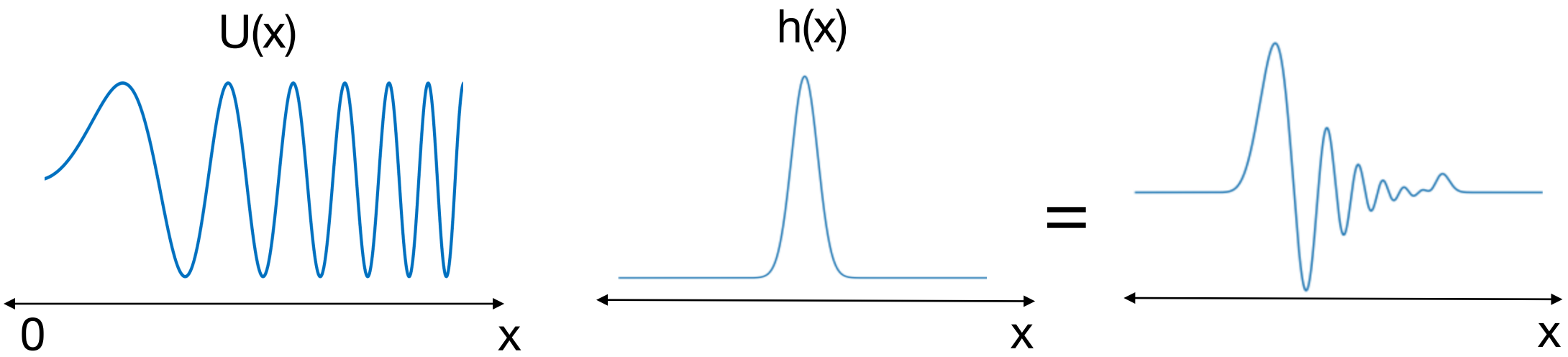
Conceptual questions for the class:

1. Draw what you think the convolution of these two functions looks like:



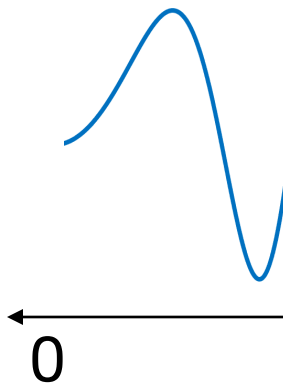
Conceptual questions for the class:

1. Draw what you think the convolution of these two functions looks like:

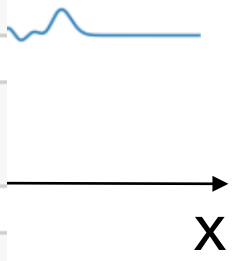


Conceptual questions for the class:

1. Dra

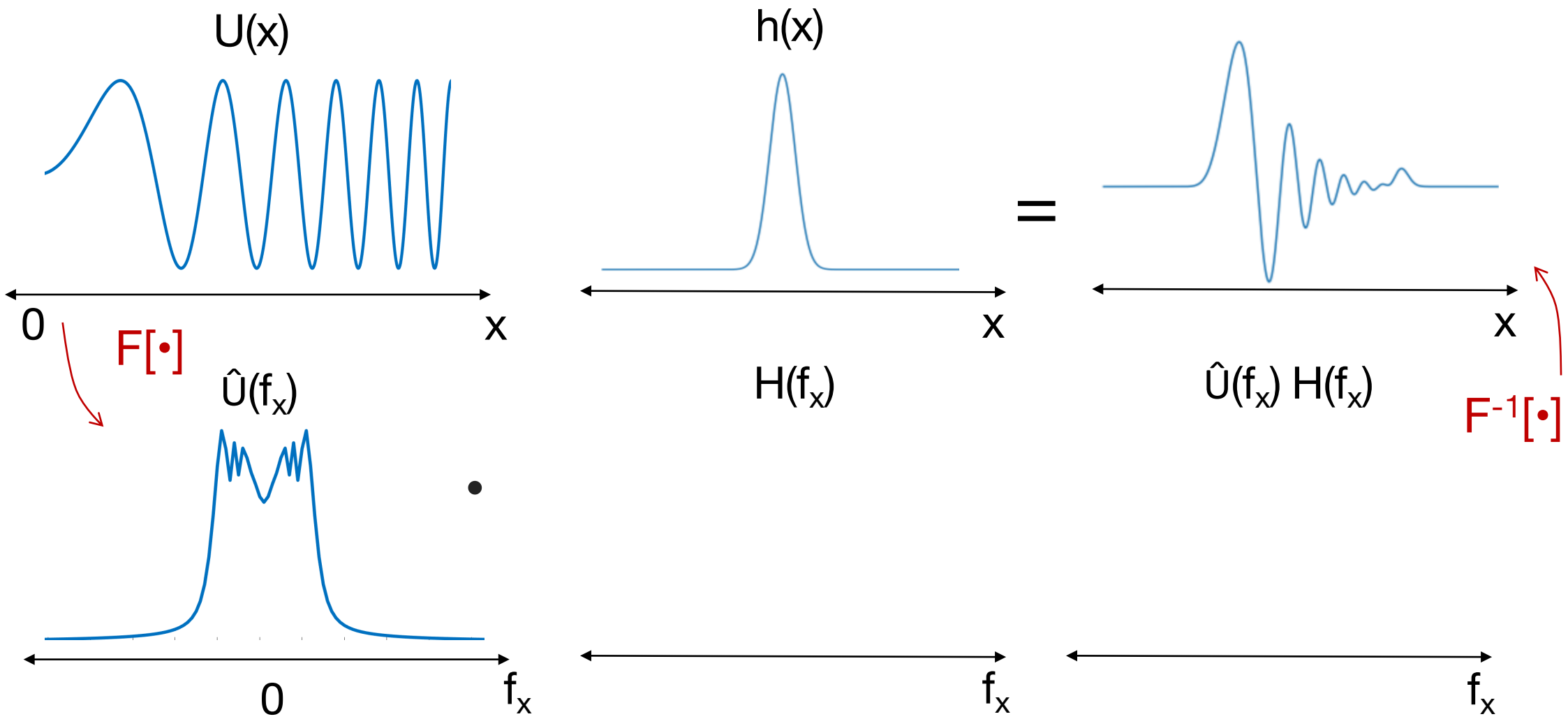



```
In [1]: import numpy as np

In [50]: x = np.linspace(0,1,1000)

In [72]: U=np.sin(40*np.square(x))
         V=np.exp(-300*np.square(x-0.5))

In [73]: W=np.convolve(U,V)

In [74]: import matplotlib.pylab as plt

In [75]: plt.plot(np.linspace(0,1,1999),W)
Out[75]: [<matplotlib.lines.Line2D at 0x108f42828>]

In [76]: plt.show()
```



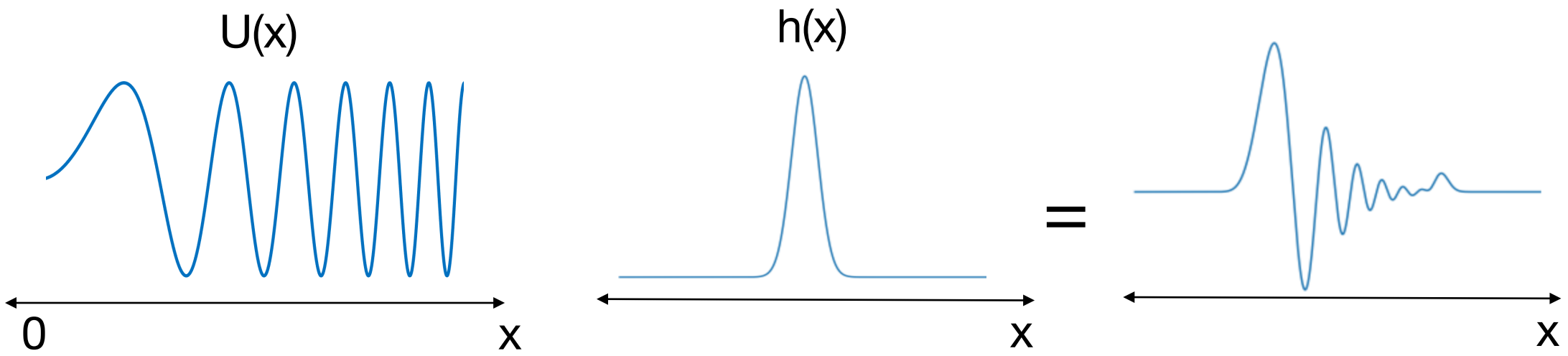
Conceptual questions for the class:

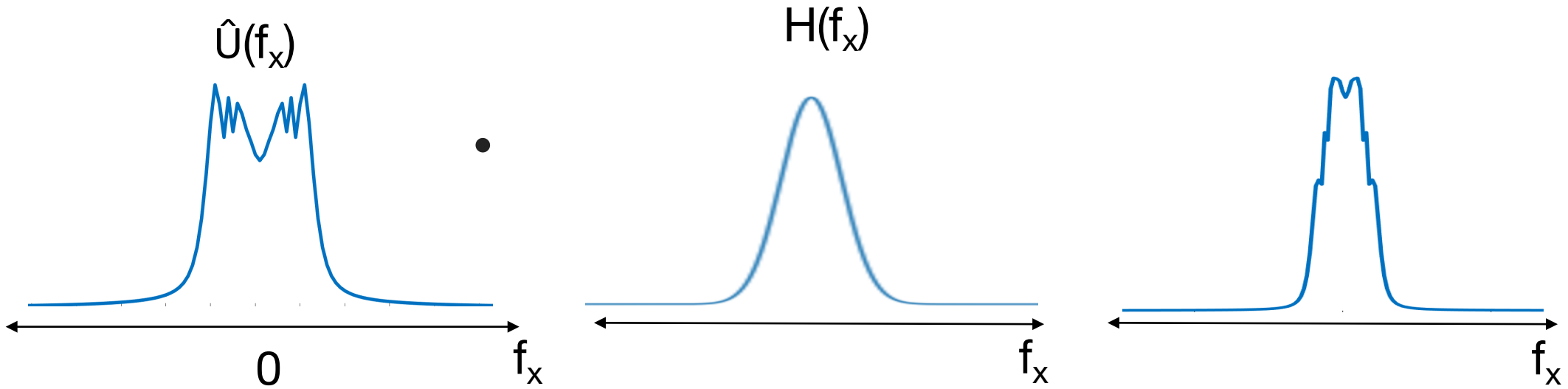
1. Draw what you think the convolution of these two functions looks like:



Conceptual questions for the class:

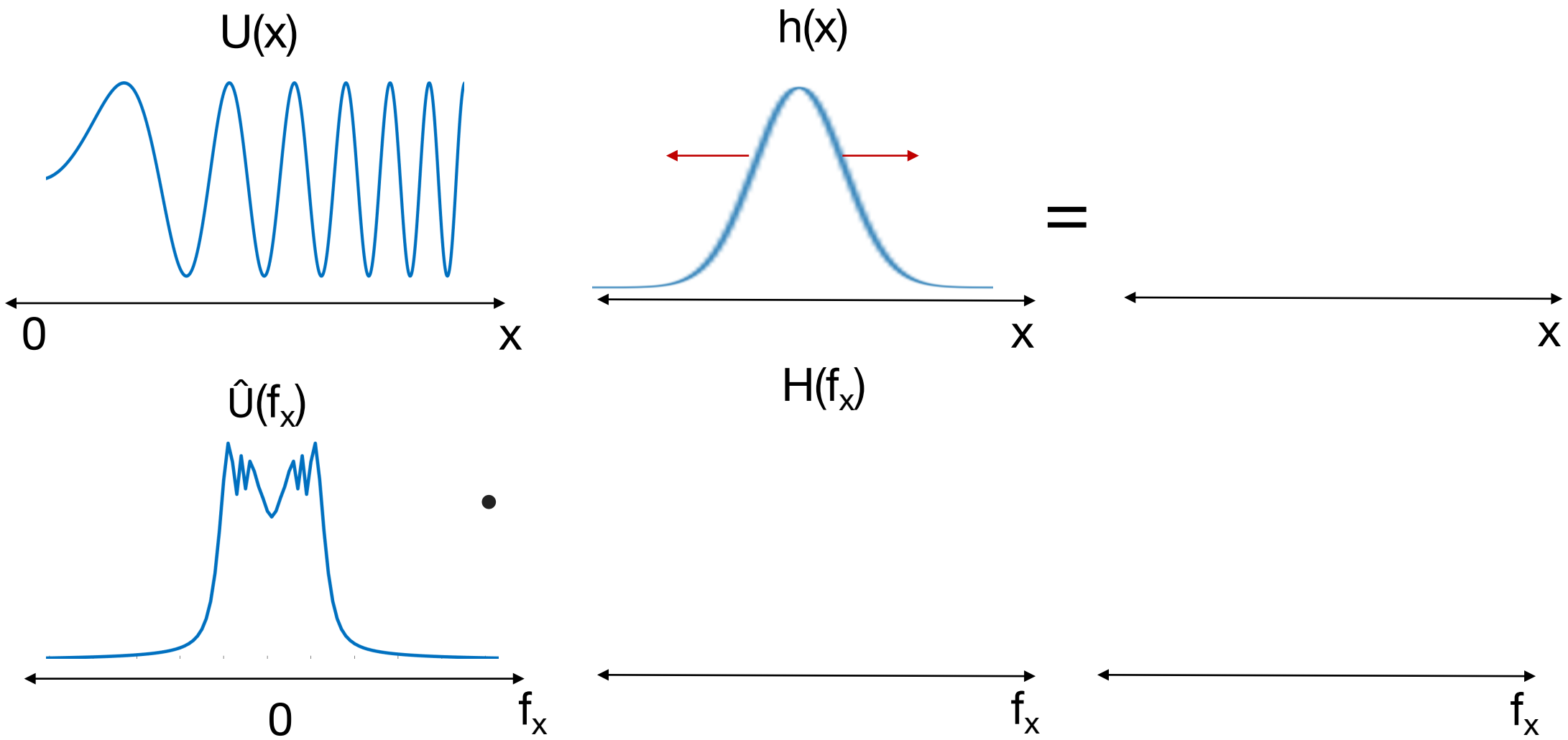
1. Draw what you think the convolution of these two functions looks like:





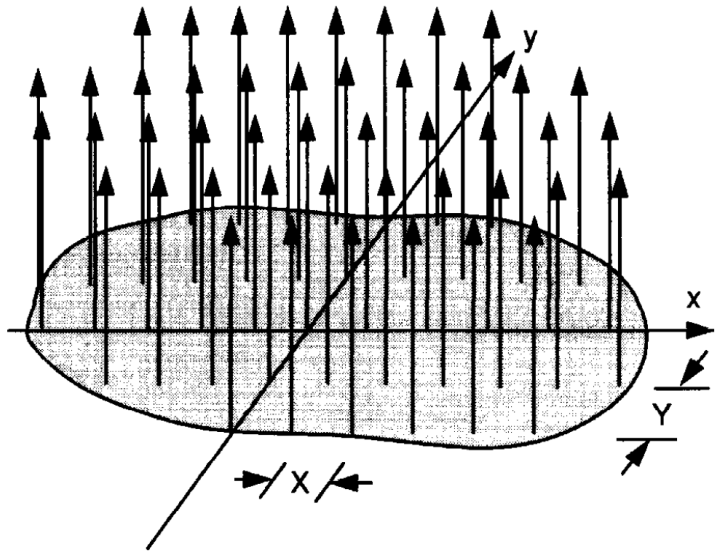
Conceptual questions for the class:

2. Repeat with wider convolution filter:



The Sampling Theorem – from Goodman Section 2.4.1

$$U_s(x,y) = \mathrm{comb}(x/X)\mathrm{comb}(y/Y)U(x,y)$$



Signal sampling occurs with:

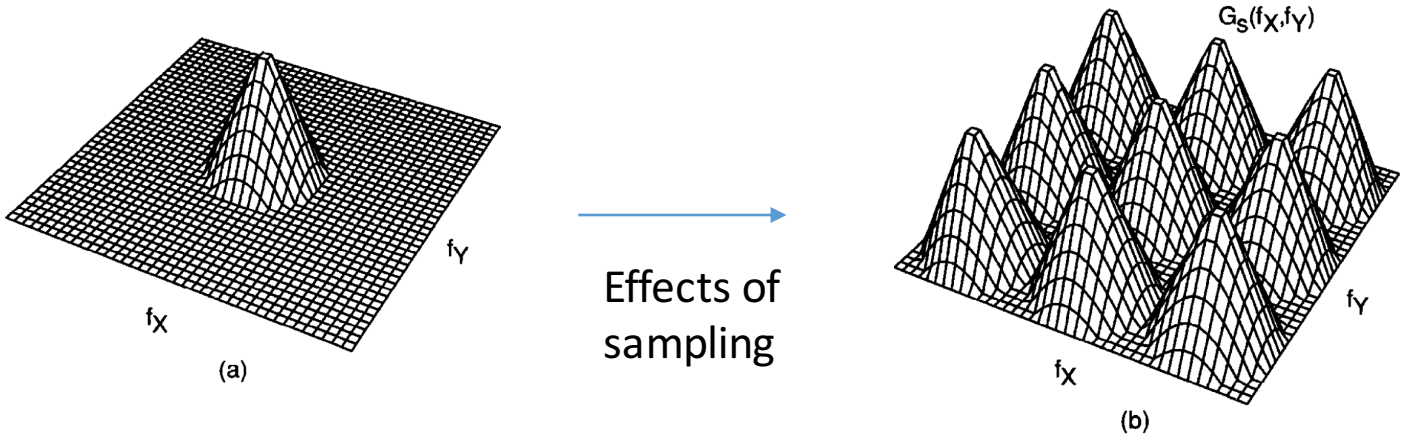
- CMOS (pixel) sensors, PMTs, SPADs
- A-to-D after antennas
- A-to-D after acoustic transducers

Sampling interval width X and Y

The Sampling Theorem – from Goodman Section 2.4.1

$$\mathcal{F}\left[\mathrm{comb}(x/X)\mathrm{comb}(y/Y)\right] = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\delta\left(f_x - \frac{n}{X}, f_y - \frac{m}{Y}\right)$$

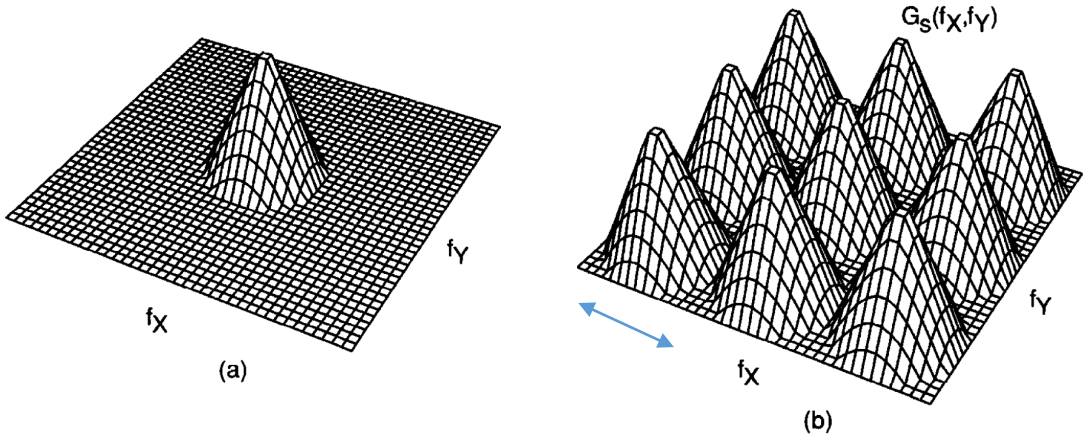
$$\hat{U}_s(f_x, f_y) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\hat{U}\left(f_x - \frac{n}{X}, f_y - \frac{m}{Y}\right)$$



The Sampling Theorem – from Goodman Section 2.4.1

$$\mathcal{F}\left[\mathrm{comb}(x/X)\mathrm{comb}(y/Y)\right] = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\delta\left(f_x - \frac{n}{X}, f_y - \frac{m}{Y}\right)$$

$$\hat{U}_s(f_x, f_y) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\hat{U}\left(f_x - \frac{n}{X}, f_y - \frac{m}{Y}\right)$$



Signal extends from (-B_x, -B_y) to (B_x, B_y) in Fourier domain

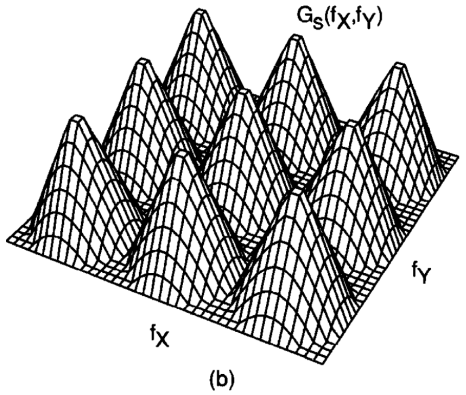
Filter out extra copies:

$$\mathrm{rect}\left(\frac{f_x}{2B_x}\right)\mathrm{rect}\left(\frac{f_y}{2B_y}\right)\hat{U}_s(f_x, f_y) = \hat{U}(f_x, f_y)$$

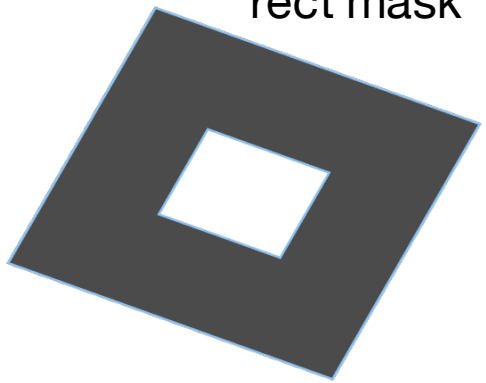
Bandwidth (B_x, B_y) of signal

The Sampling Theorem – from Goodman Section 2.4.1

$$\text{rect}\left(\frac{f_x}{2B_x}\right)\text{rect}\left(\frac{f_y}{2B_y}\right)\hat{U}_s(f_x,f_y)=\hat{U}(f_x,f_y)$$



•



rect mask

$F[\bullet]$

$$h(x,y)=4B_xB_y\text{sinc}(2B_xx)\text{sinc}(2B_yy)$$

Can manipulate the above masking to show,

$$U(x,y)=4B_xB_yXY\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

The Sampling Theorem

When sampled appropriately, a discrete signal can *exactly* reproduce a continuous signal:

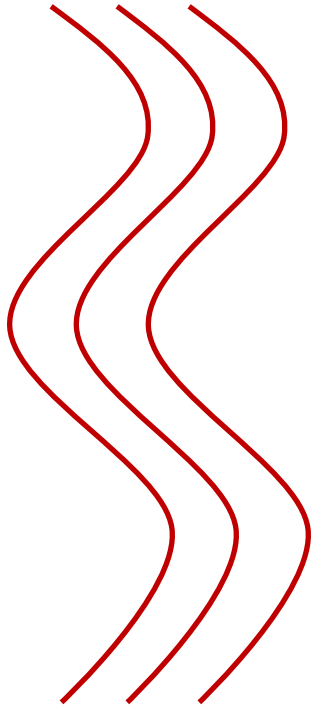
$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\,\mathrm{sinc}\left[2B_x(x-nX)\right]\mathrm{sinc}\left[2B_y(y-mY)\right]$$

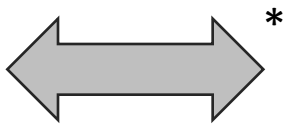
Continuous signal:
- EM field
- Sound wave
- MR signal

Discretized signal:
- Detected EM field
- Sampled sound wave
- Sampled MR signal

What does the Sampling Theorem mean for us?

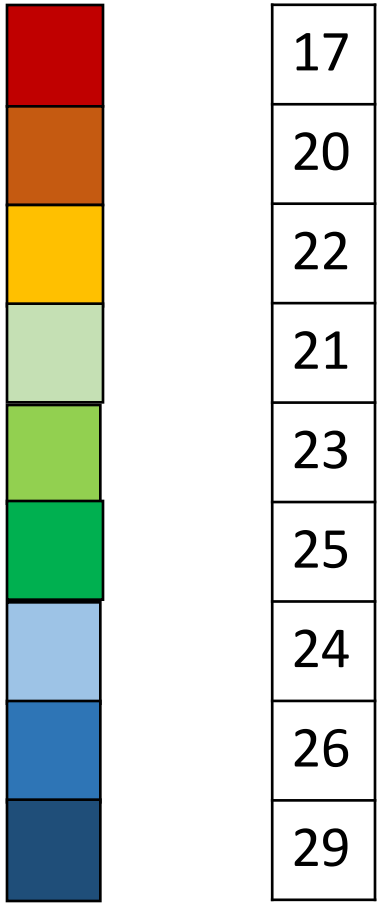
Continuous fields





(*) Under certain conditions

Discretize vectors (and matrices)



17
20
22
21
23
25
24
26
29

Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

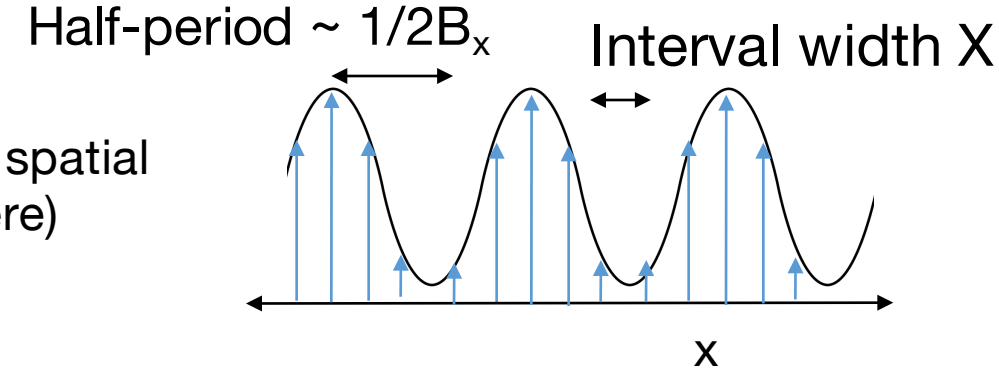
- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$

Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$

(Assume largest spatial freq. in signal here)



U(x)comb(x/X)

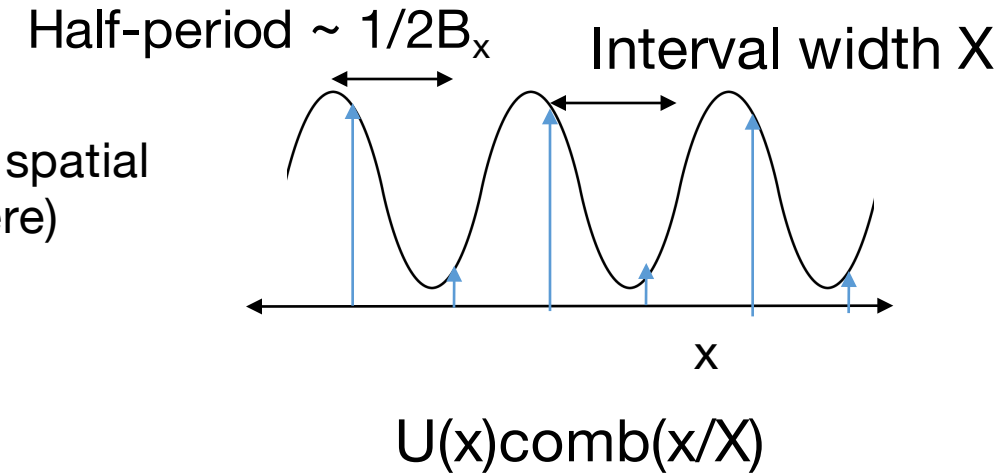
$X < 1/2B_x$ ✔

Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$

(Assume largest spatial freq. in signal here)



$X = 1/2B_x$ ✔

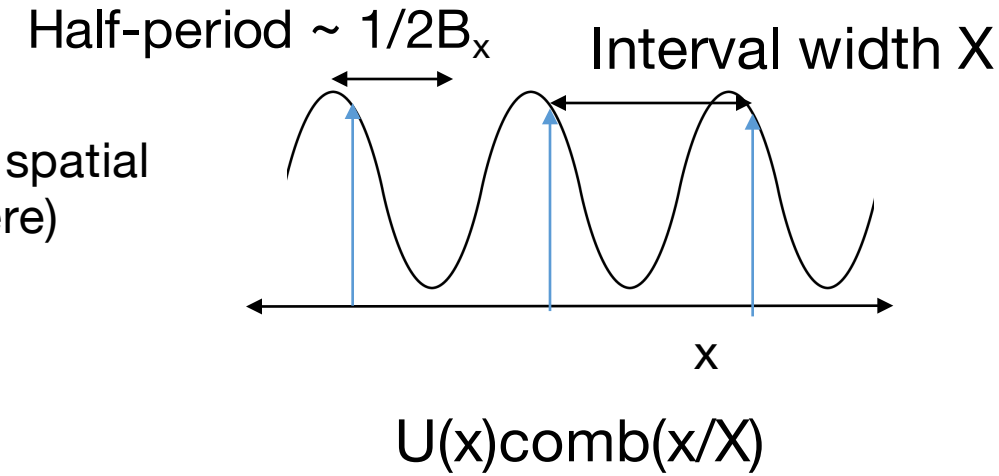
Nyquist sampling – still sampling peak and trough

Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$

(Assume largest spatial freq. in signal here)



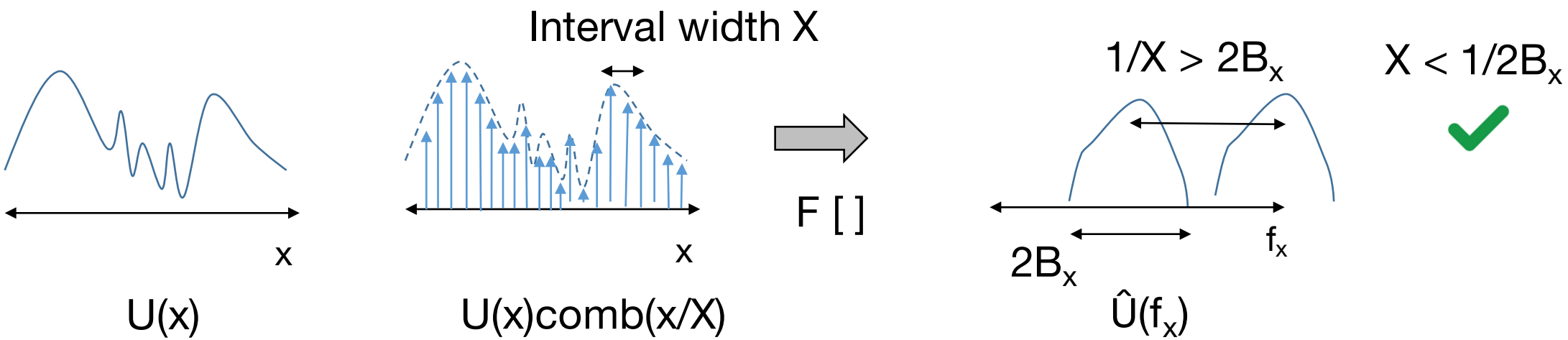
$X > 1/2B_x$ 

Can't detect the frequency anymore!

Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

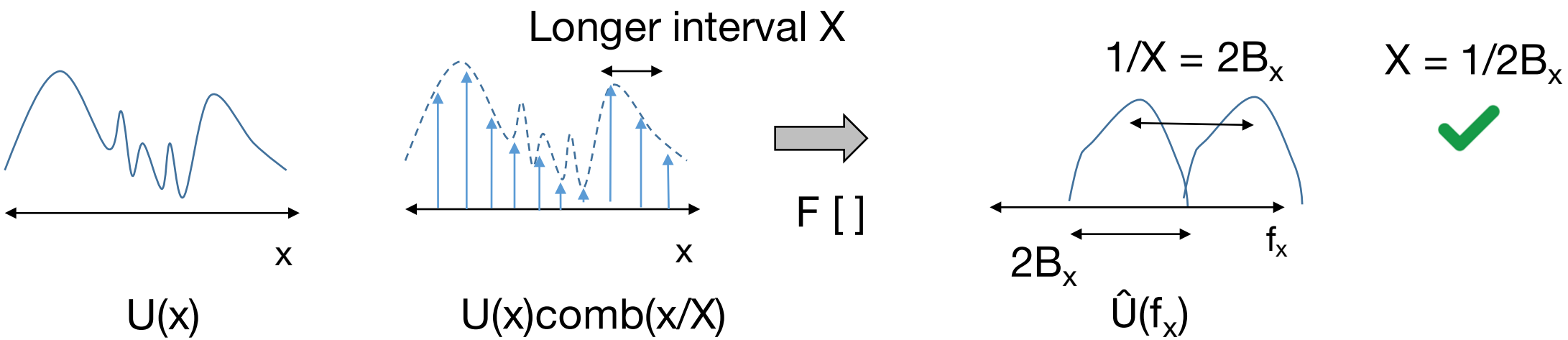
- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$
  - Needed to avoid aliasing



Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

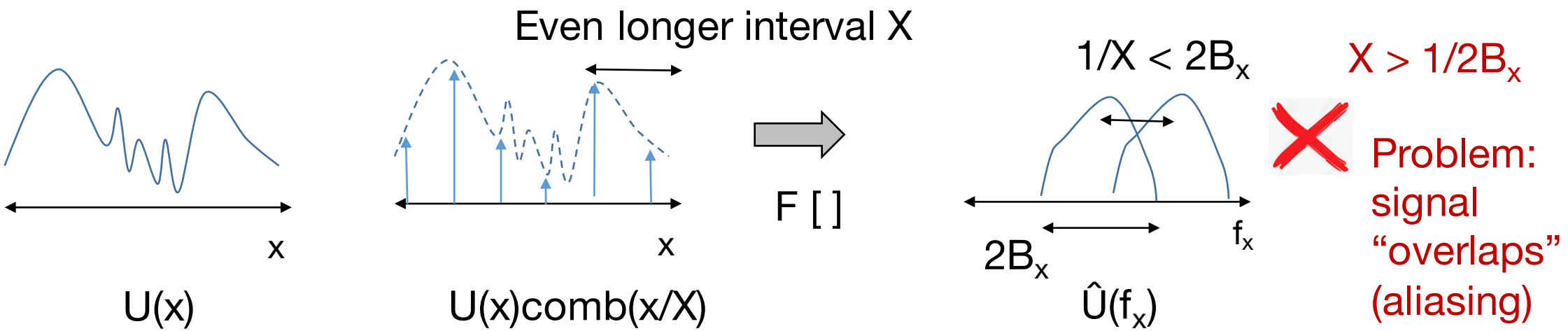
- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$
  - Needed to avoid aliasing



Conditions to safely apply the sampling theorem

$$U(x,y) = 4B_xB_yXY \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} U(nX,mY)\text{sinc}\left[2B_x(x-nX)\right]\text{sinc}\left[2B_y(y-mY)\right]$$

- Sampling must be proportional to bandwidth ($2B_x$ and $2B_y$)
  - "Nyquist" sampling: $X = 1/2B_x$, $Y = 1/2B_y$
  - Needed to avoid aliasing



Linear Algebra – notation and basics

Linear Algebra – notation and basics

- We'll (try to) write *column* vectors as lower case variables w/ brackets
- Row vectors will be denoted as the transpose
- We'll try to write matrices as upper case variables
- We'll try to denote if a matrix/vector is real, complex etc. and its size with the following notation

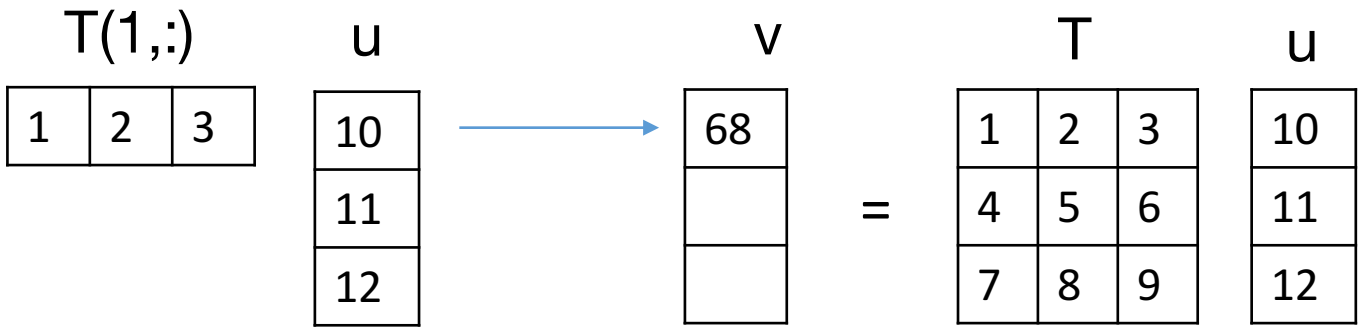
Linear Algebra – notation and basics

Some basic vector operations you should know:

- Conjugate, transpose, conjugate transpose
- Inner product
- Hadamard (element-wise, dot-times) product
- outer product
- Vector (matrix) addition
- matrix-vector product
- convolution

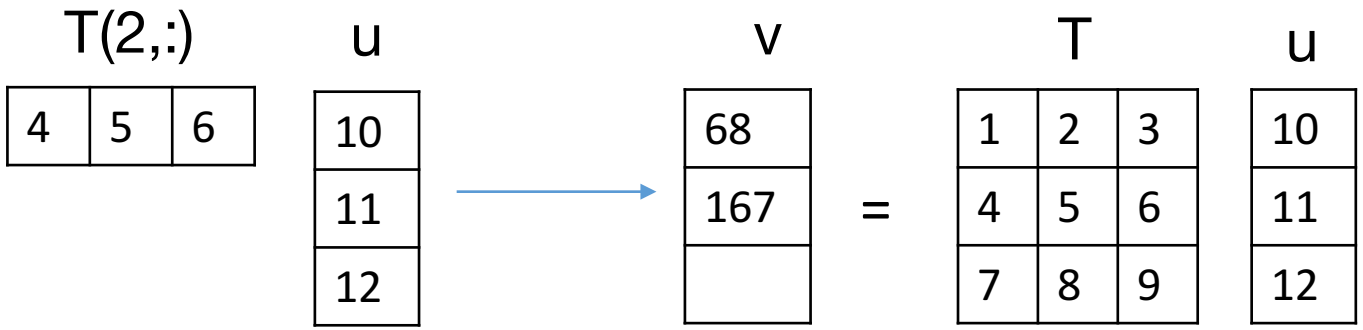
Matrix-vector products – two useful interpretations

1. Inner products per entry:



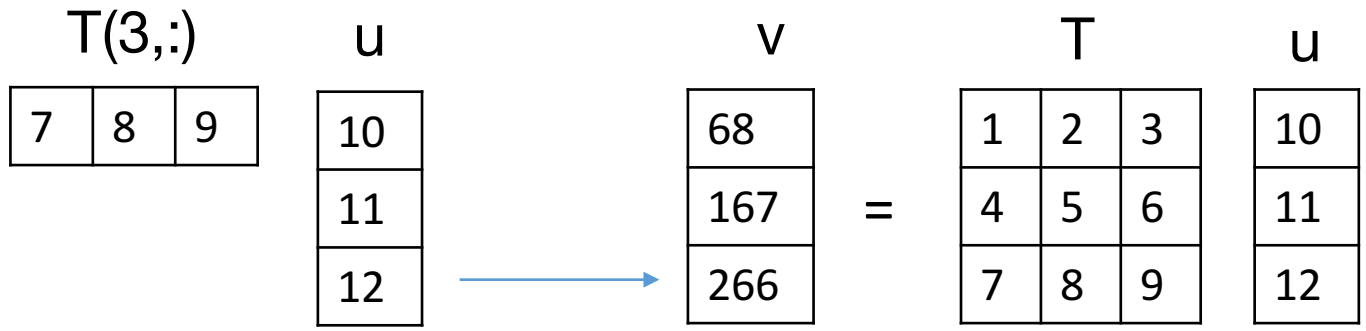
Matrix-vector products – two useful interpretations

1. Inner products per entry:



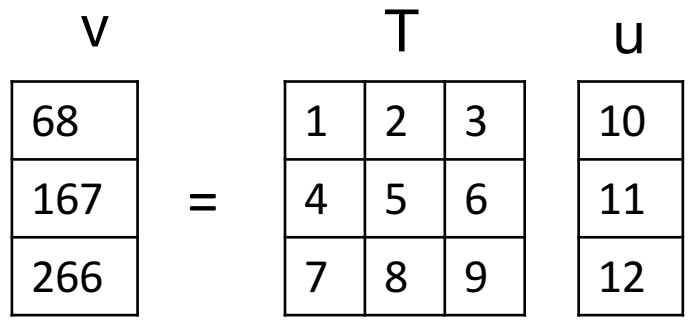
Matrix-vector products – two useful interpretations

1. Inner products per entry:

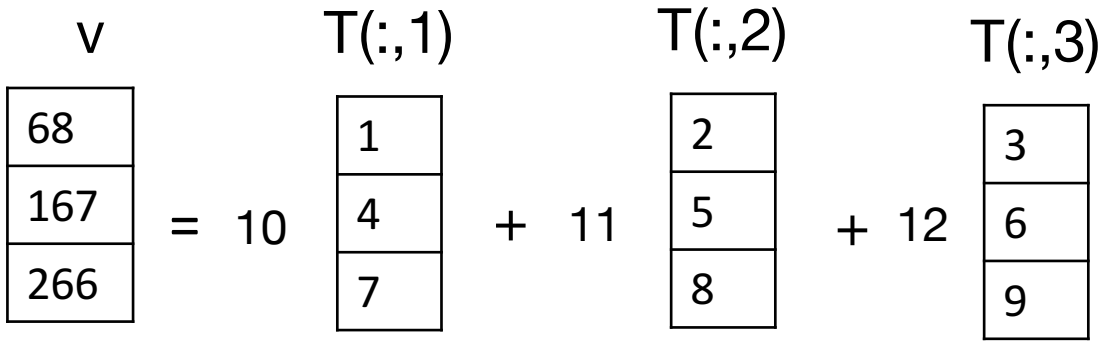


Matrix-vector products – two useful interpretations

1. Inner products per entry:



2. Weighted column sum:



Discrete convolution

$$V(x_o) = \int_{-\infty}^{\infty} U(x_i) h(x_o - x_i) dx_i$$

$$v[x_0] = \sum_{x_i=-M}^{M} u[x_i] h[x_o - x_i]$$

Discrete 1D Convolution – an example

Steps to follow:

Step 1	List the index 'k' covering a sufficient range
Step 2	List the input x[k]
Step 3	Obtain the reversed sequence h[-k], and align the rightmost element of h[n-k] to the leftmost element of x[k]
Step 4	Cross-multiply and sum the nonzero overlap terms to produce y[n]
Step 5	Slide h[n-k] to the right by one position
Step 6	Repeat step 4; stop if all the output values are zero or if required.

http://host.uniroma3.it/laboratori/sp4te/teaching/sp4bme/documents/LectureConvolution.pdf

Example 2: Find the convolution of the two sequences x[n] and h[n] given by,

x[k] = [3 1 2] h[k] = [3 2 1]

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3

Hint: The value of k starts from **(– length of h + 1)** and continues till **(length of h + length of x – 1)**

Here ***k*** starts from **-3 + 1 = -2** and continues till **3 + 3 – 1 = 5**

http://host.uniroma3.it/laboratori/sp4te/teaching/sp4bme/documents/LectureConvolution.pdf

Example 2: Find the convolution of the two sequences x[n] and h[n] given by,

$x[k] = [3\ 1\ 2]$ $h[k] = [3\ 2\ 1]$

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3

y: 9

Example 2: Find the convolution of the two sequences x[n] and h[n] given by,

$$x[k] = [3\ \underset{\uparrow}{1}\ 2] \qquad h[k] = [3\ \underset{\uparrow}{2}\ 1]$$

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3
y:			9	6+3				

Example 2: Find the convolution of the two sequences x[n] and h[n] given by,

x[k] = [3 1 2] h[k] = [3 2 1]

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3

y: 9 6+3 3+2+6

Example 2: Find the convolution of the two sequences x[n] and h[n] given by,

$x[k] = [3\ \underset{\uparrow}{1}\ 2]$ $\quad h[k] = [3\ \underset{\uparrow}{2}\ 1]$

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3

y: $\quad 9 \quad 6+3 \quad 3+2+6 \quad 1+4+0$

Example 2: Find the convolution of the two sequences x[n] and h[n] given by,

x[k] = [3 1 2] h[k] = [3 2 1]

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3

y: 9 6+3 3+2+6 1+4+0

y: [9 9 11 5 2 0]

Discrete convolution

$$V(x_o) = \int_{-\infty}^{\infty} U(x_i)h(x_o - x_i)dx_i$$

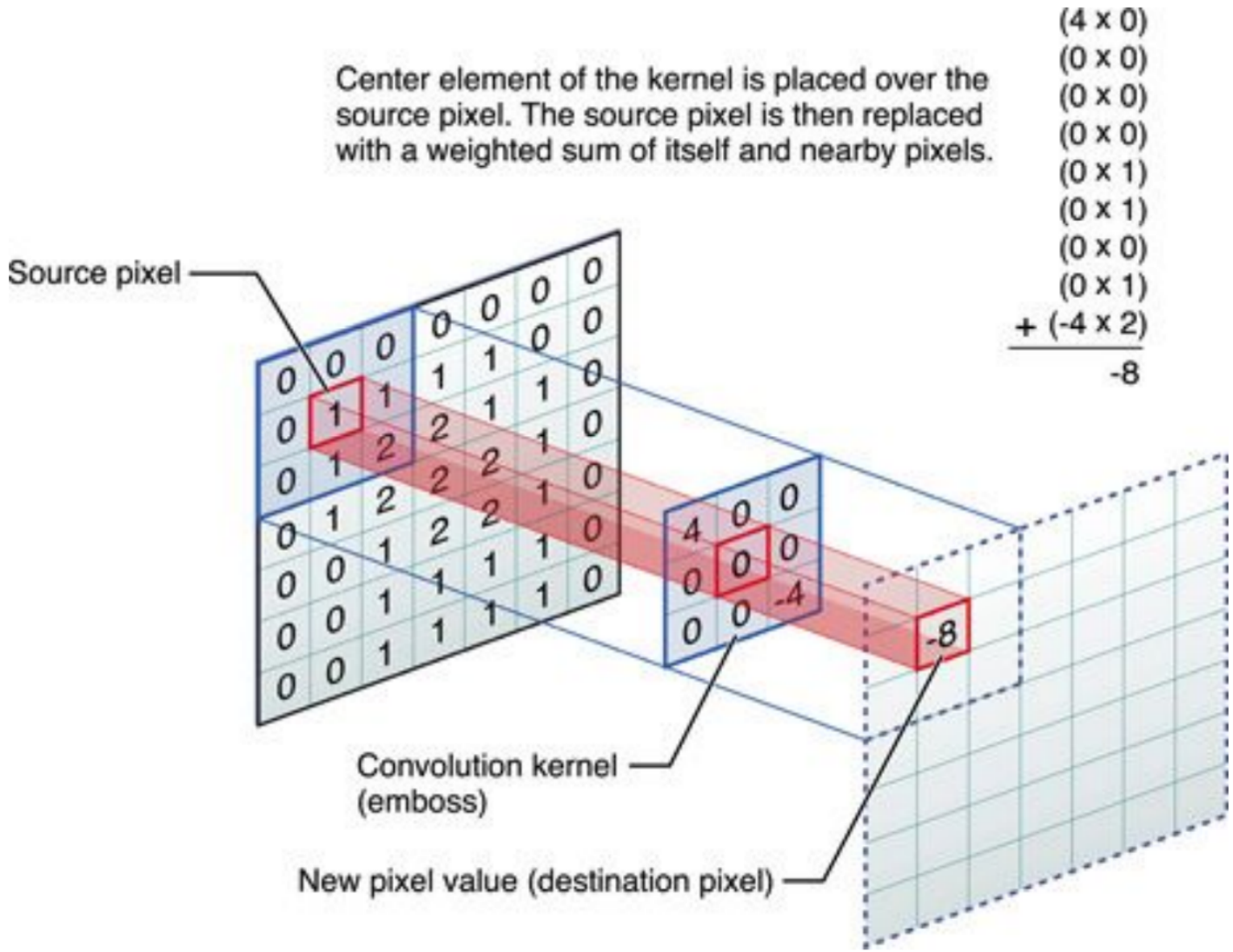
$$v[x_0] = \sum_{x_i=-M}^{M} u[x_i]h[x_o - x_i]$$

Discrete 2D convolution

$$V(x_o, y_o) = \iint_{-\infty}^{\infty} U(x_i, y_i)h(x_o - x_i, y_o - y_i)dx_i dy_i$$

$$v[x_0, y_o] = \sum_{y_i=-L}^{L} \sum_{x_i=-M}^{M} u[x_i, y_i]h[x_o - x_i, y_o - y_i]$$

Discrete 2D convolution



https://www.psi.toronto.edu/~jimmy/ece521/Tut1.pdf

Linear Algebra – notation and basics

Some basic types of matrices & terms that you should know about:

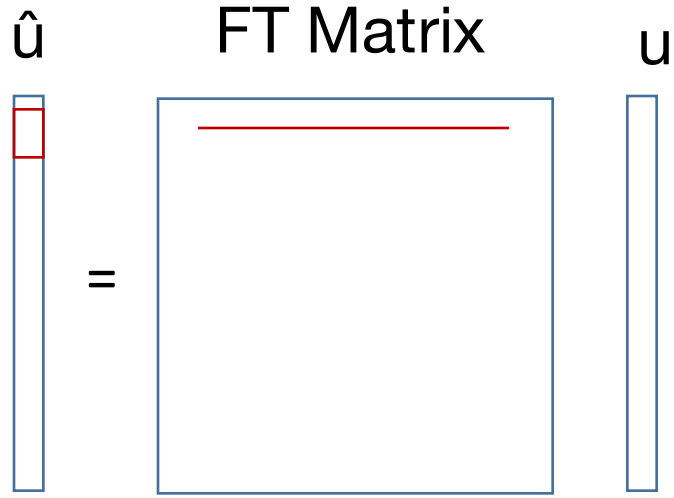
- Symmetric (Hermitian) matrix: $A=A^T$ if A is real, $A=A^H$ if A is complex
- Square, hot-dog and hamburger matrices
- Invertible matrix
- Diagonal matrix
- Toeplitz matrix
- Banded matrix

Discrete Fourier Transforms

$$\hat{U}(f_x) = \int_{-\infty}^{\infty} U(x) \exp\left(-2\pi i (f_x x)\right) dx$$

$$\hat{u}[f_x] = \sum_{x=0}^{M-1} u[x] \exp\left(-2\pi i f_x x / M\right)$$

Inner product of u with different complex expon.



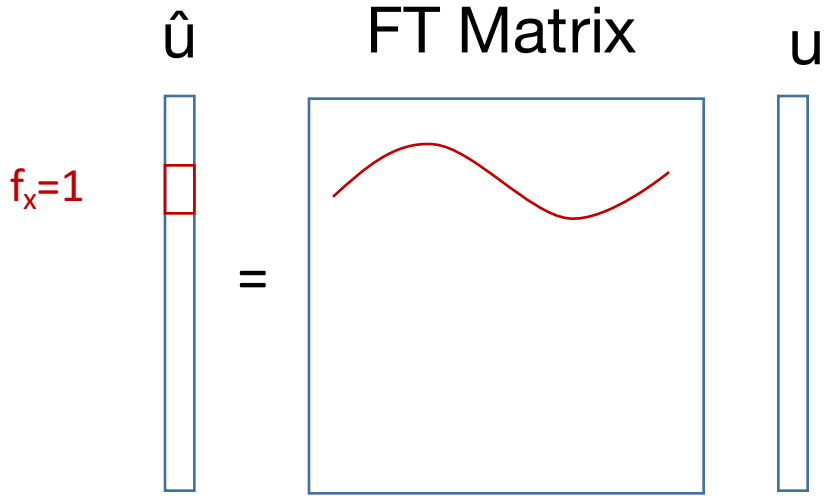
- np.fft(u), np.fftshift(np.fft(np.ifftshift(u)))
- fft = fast Fourier transform, much more comp. efficient than matrix multiplication!

Discrete Fourier Transforms

$$\hat{U}(f_x) = \int_{-\infty}^{\infty} U(x) \exp\left(-2\pi i (f_x x)\right) dx$$

$$\hat{u}[f_x] = \sum_{x=0}^{M-1} u[x] \exp\left(-2\pi i f_x x / M\right)$$

Inner product of u with different complex expon.



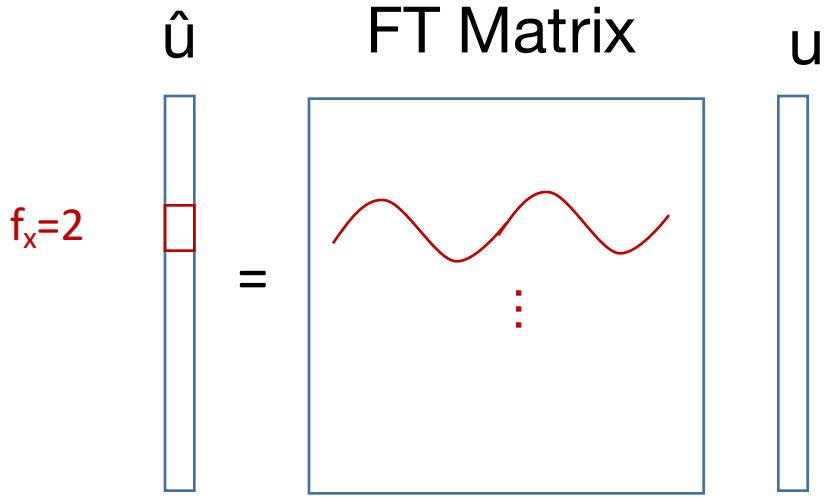
- np.fft(u), np.fftshift(np.fft(np.ifftshift(u)))
- fft = fast Fourier transform, much more comp. efficient than matrix multiplication!

Discrete Fourier Transforms

$$\hat{U}(f_x) = \int_{-\infty}^{\infty} U(x) \exp\left(-2\pi i (f_x x)\right) dx$$

$$\hat{u}[f_x] = \sum_{x=0}^{M-1} u[x] \exp\left(-2\pi i f_x x / M\right)$$

Inner product of u with different complex expon.

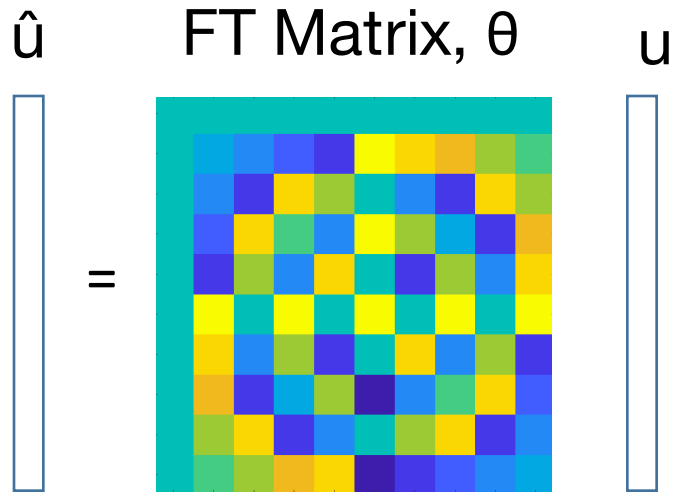


- np.fft(u), np.fftshift(np.fft(np.ifftshift(u)))
- fft = fast Fourier transform, much more comp. efficient than matrix multiplication!

Discrete Fourier Transforms

$$\hat{U}(f_x) = \int_{-\infty}^{\infty} U(x) \exp\left(-2\pi i (f_x x)\right) dx$$

$$\hat{u}[f_x] = \sum_{x=0}^{M-1} u[x] \exp\left(-2\pi i f_x x / M\right)$$



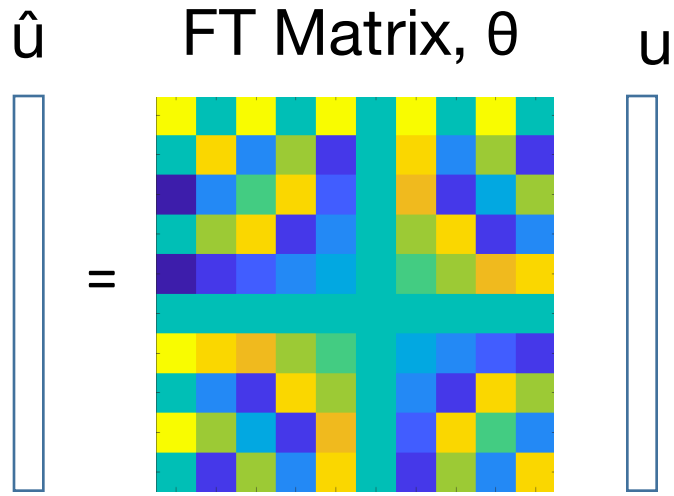
np.fft(np.eye(10))

Treats 1st entry of û as f_x=0

Discrete Fourier Transforms

$$\hat{U}(f_x) = \int_{-\infty}^{\infty} U(x) \exp\left(-2\pi i (f_x x)\right) dx$$

$$\hat{u}[f_x] = \sum_{x=0}^{M-1} u[x] \exp\left(-2\pi i f_x x / M\right)$$



np.fftshift(np.fft(np.ifftshift(np.eye(10)))))

Treats middle entry of û as f_x=0