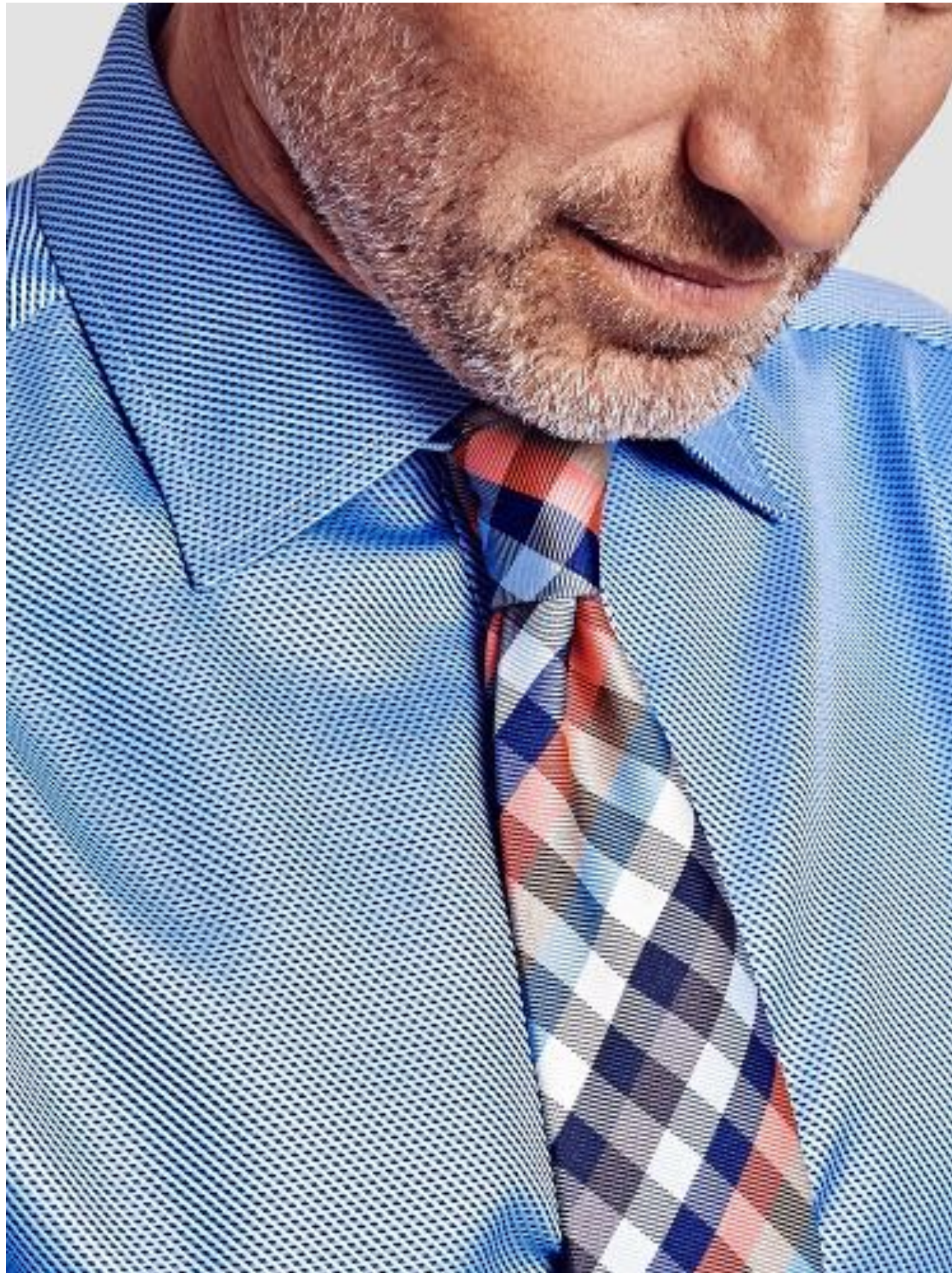


Computer Graphics

Sampling and Anti-aliasing

Subodh Kumar, Dept of Computer Sc. & Engg., IIT Delhi

Insufficient Samples



Original image



Sub-sampled image

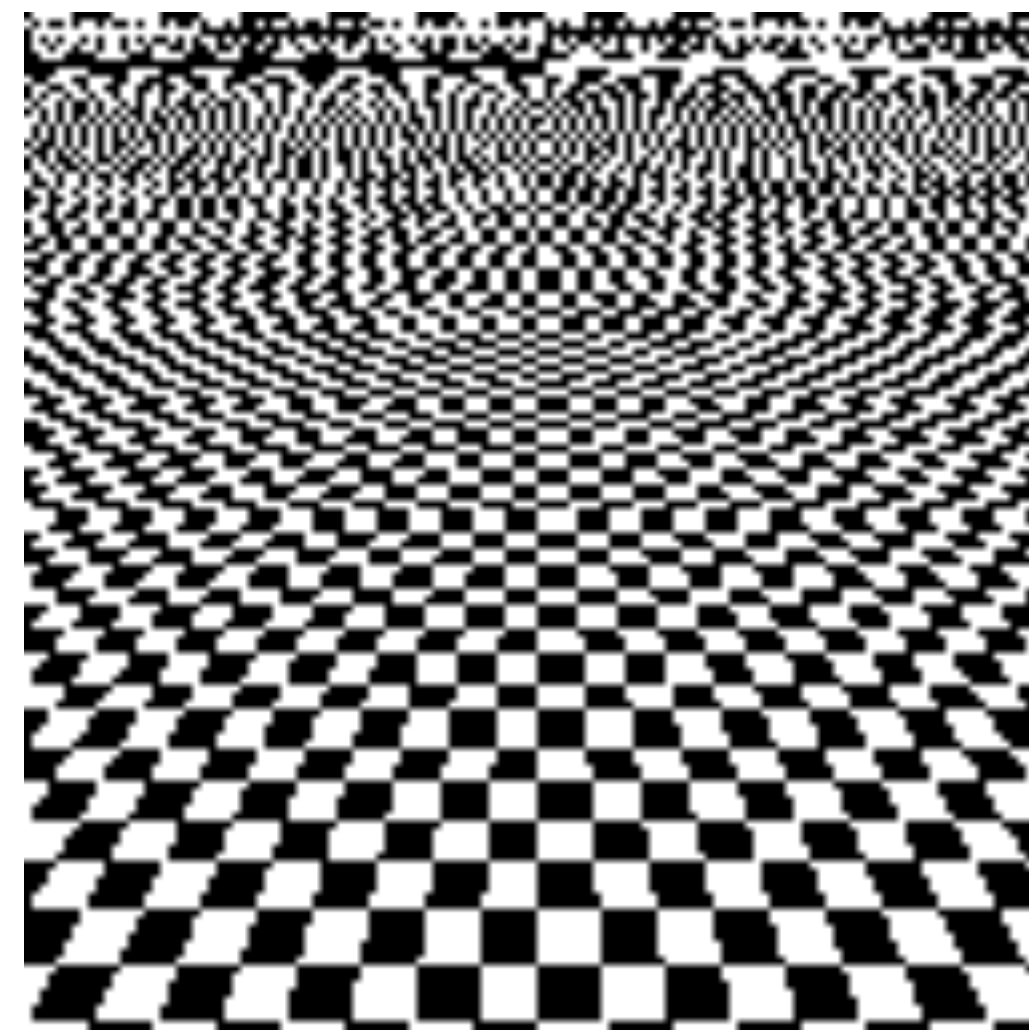
Aliasing



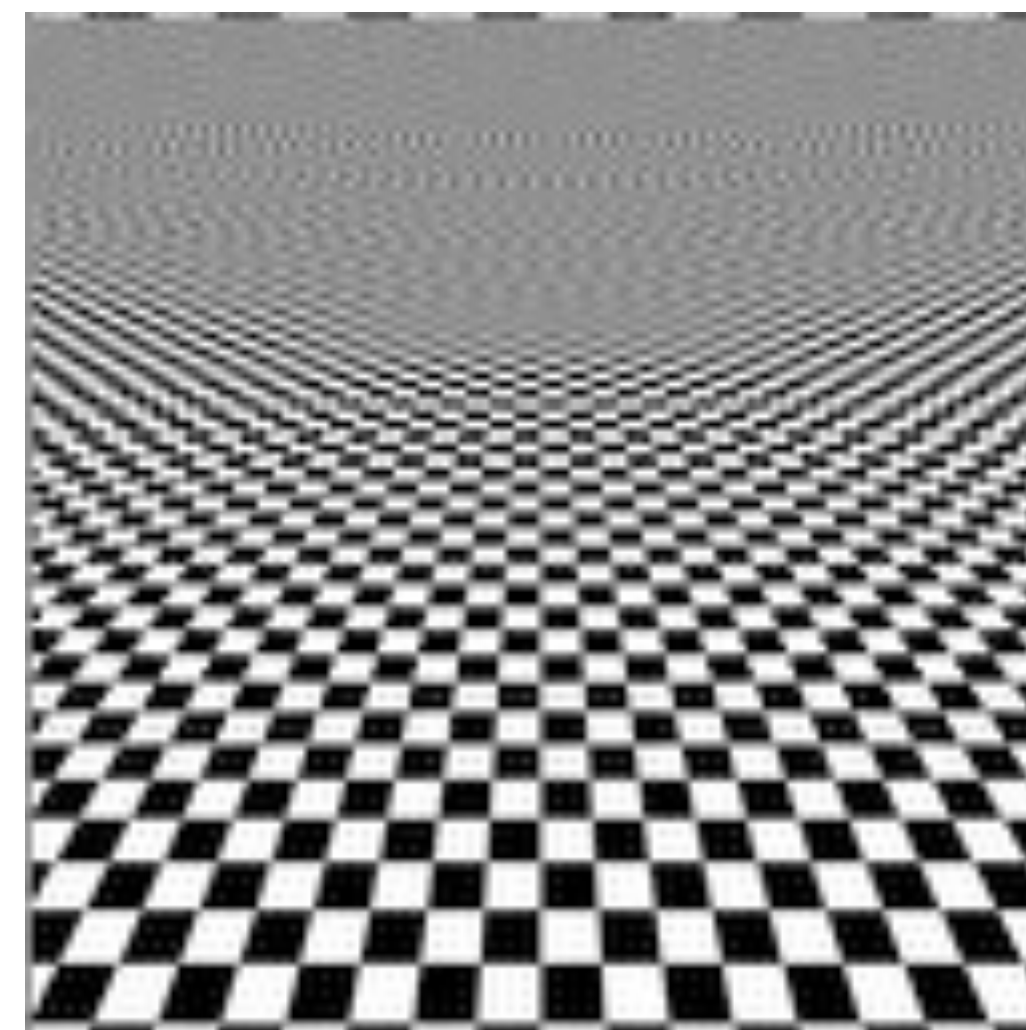
Aliasing

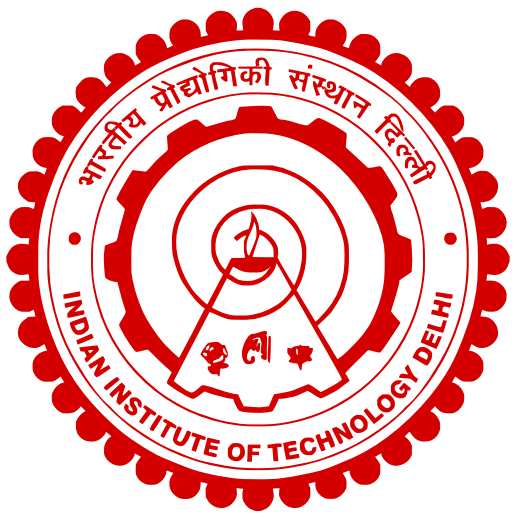


aliased



“anti-aliased”



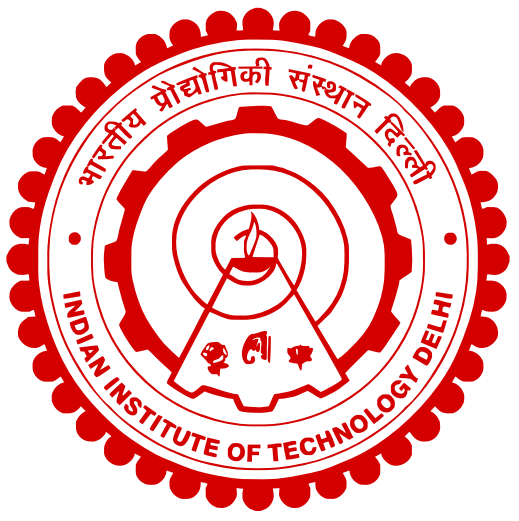


Aliasing and Anti-aliasing

- Fewer samples than required
 - ➡ Less than Nyquist rate
 - ➡ A lower rate sampling can cause a high-frequency noise in reconstructed signal
- Solution:
 - ➡ Increase sampling rate, or
 - ➡ Remove high frequency
 - Or a combination

Need to understand the frequency content of an image

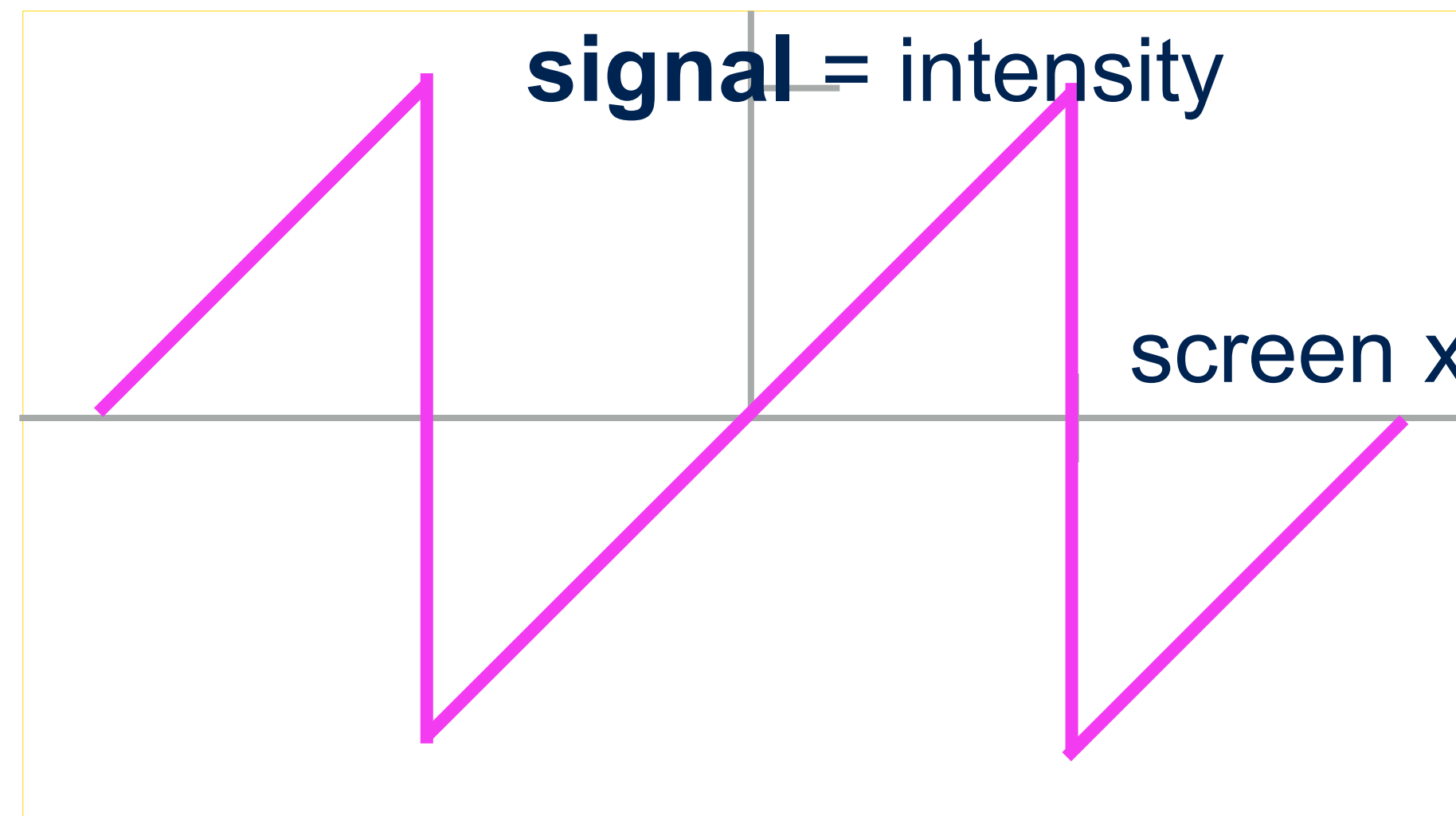
Fourier Transform



A function can be represented by a (possibly infinite) sum of harmonic sine and cosine functions.

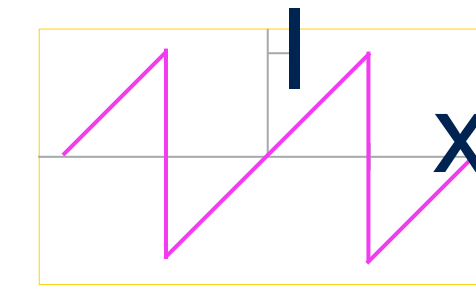
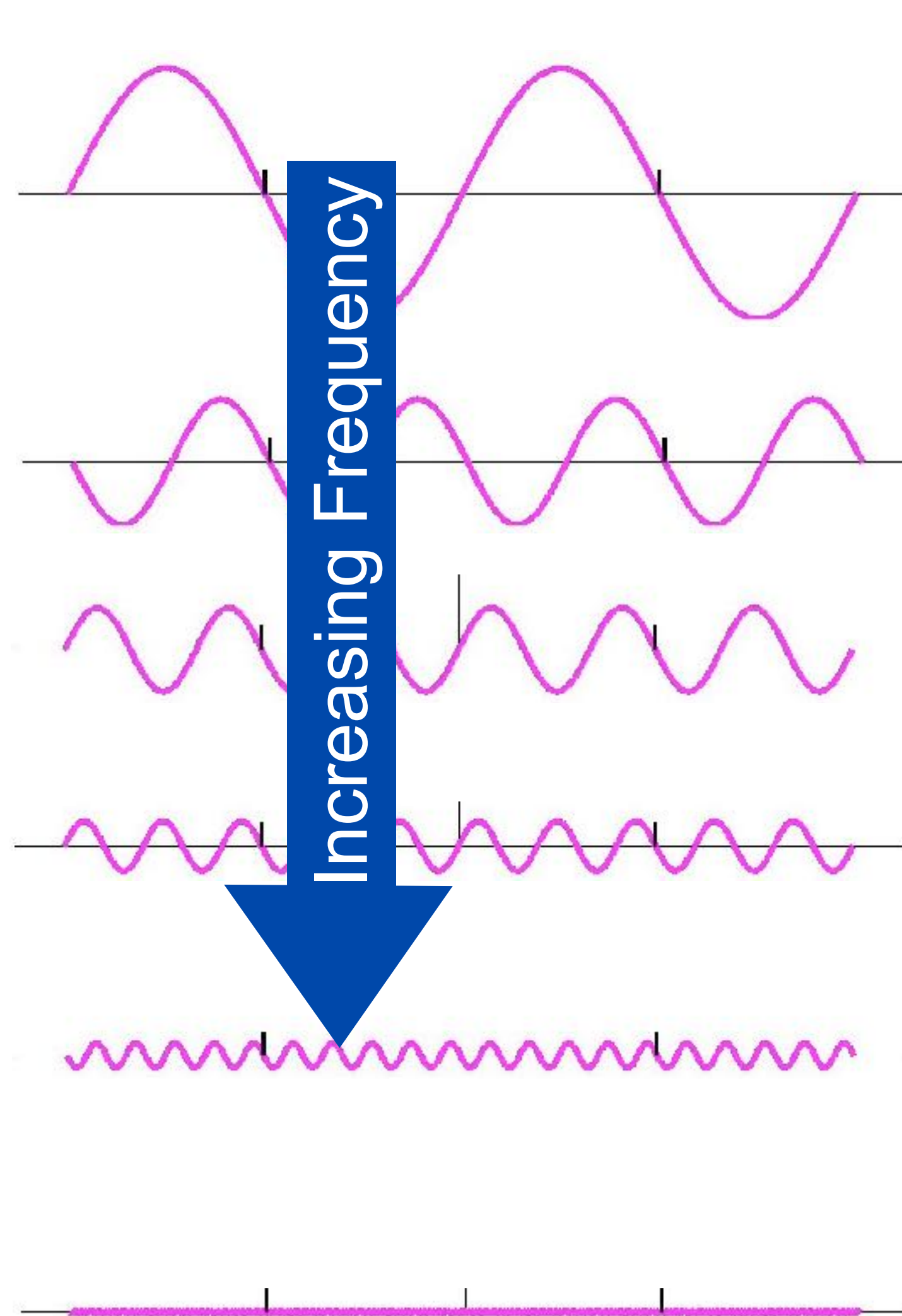
Harmonics = integer multiples of a fundamental frequency

Harmonic Summation



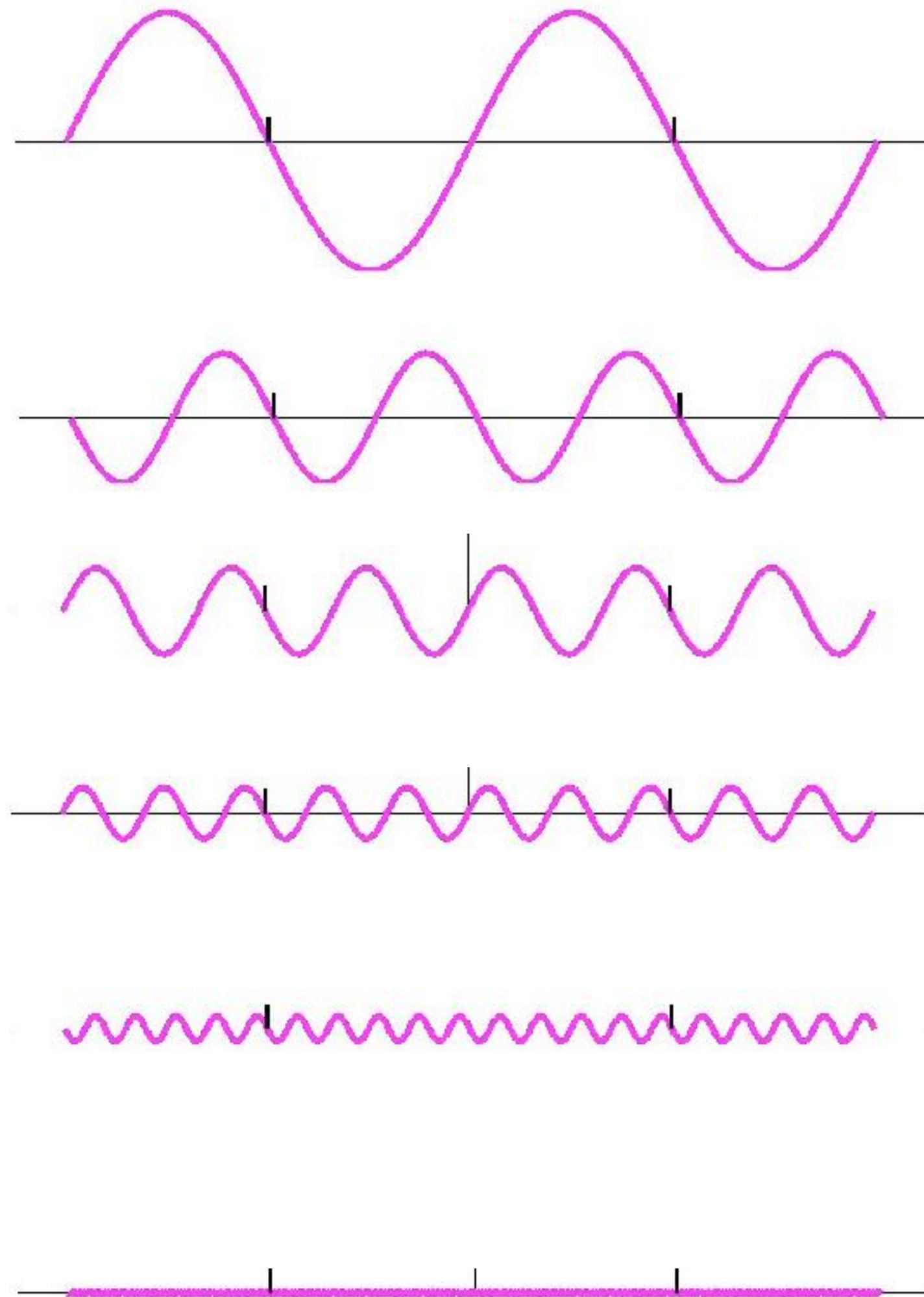
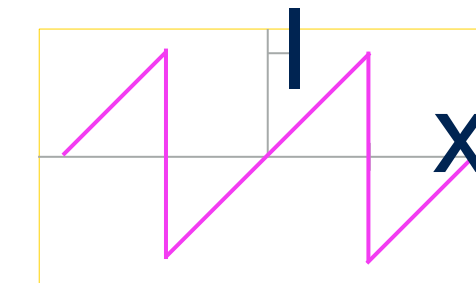


Harmonic Summation



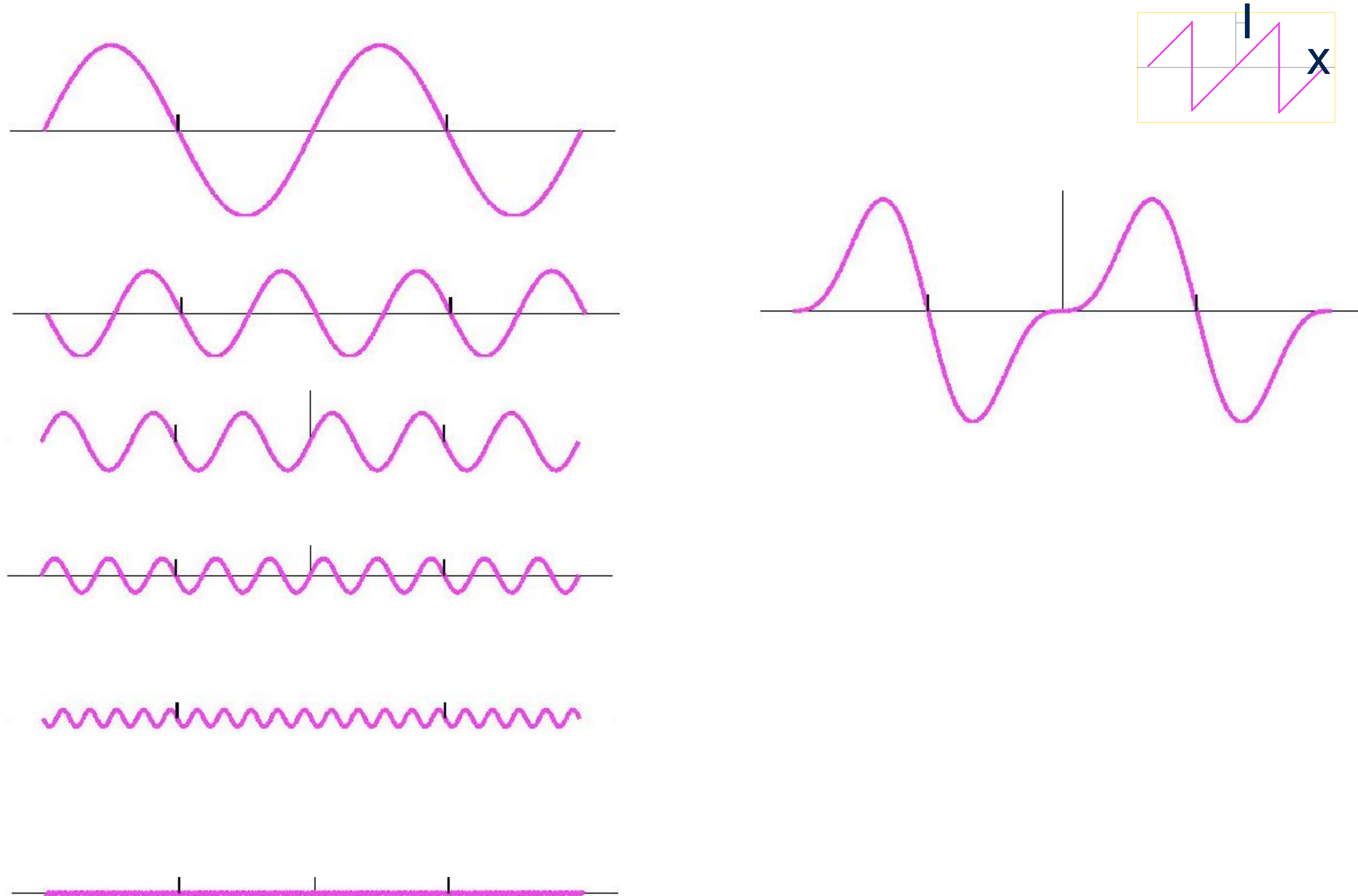


Harmonic Summation



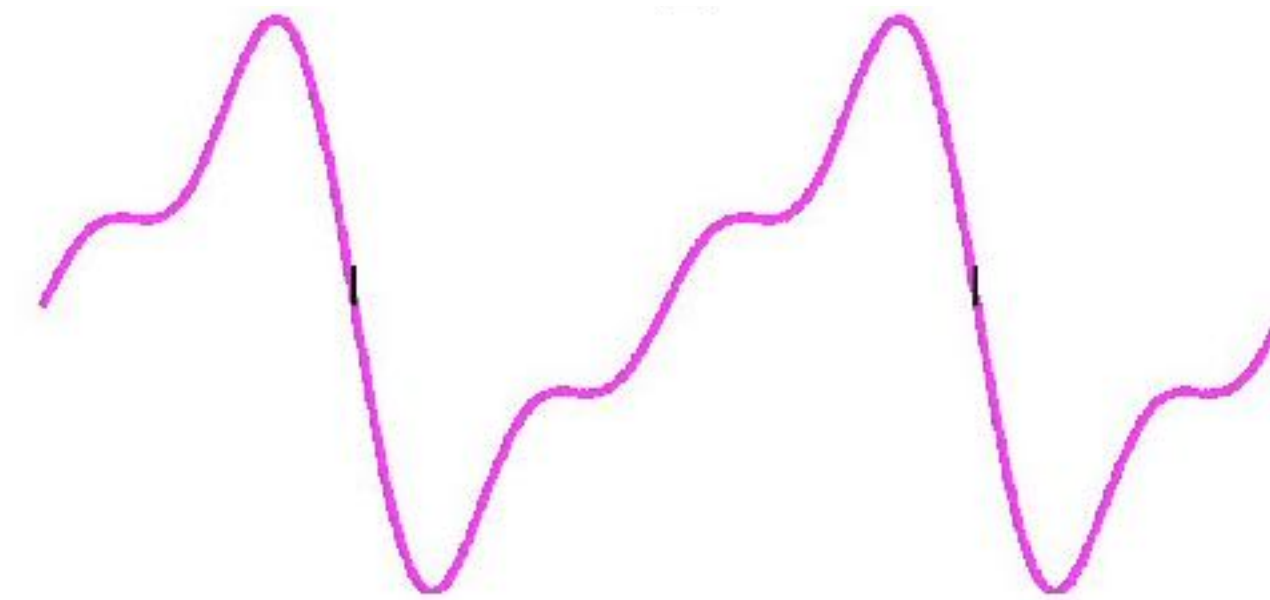
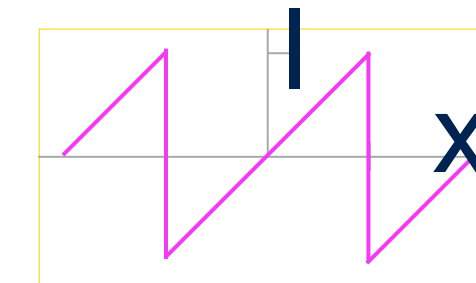
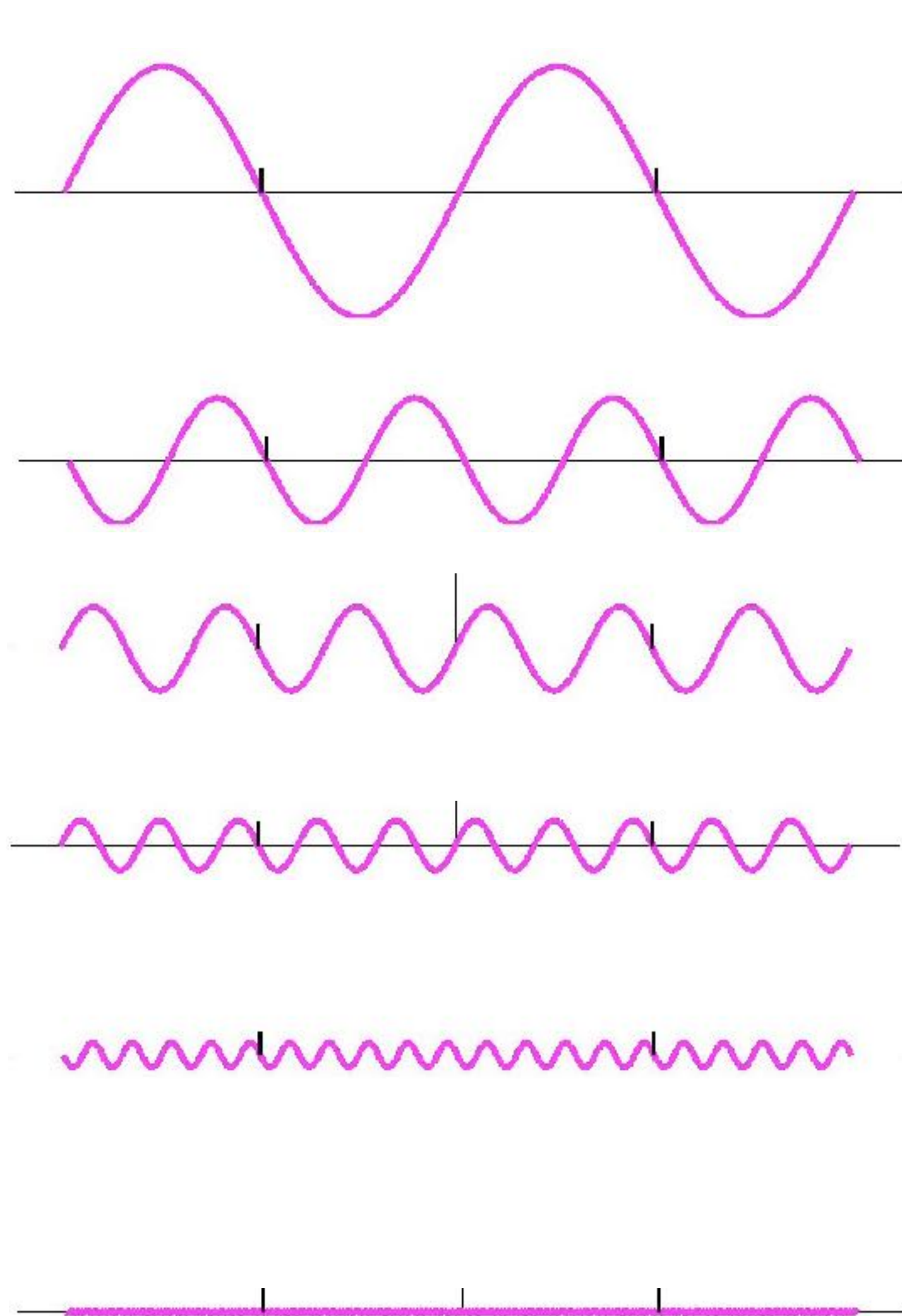


Harmonic Summation

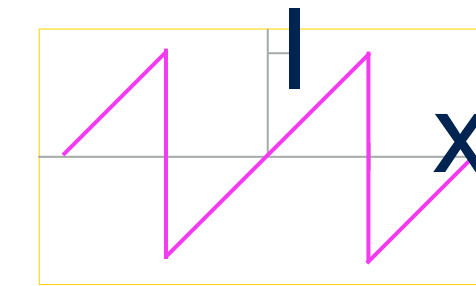
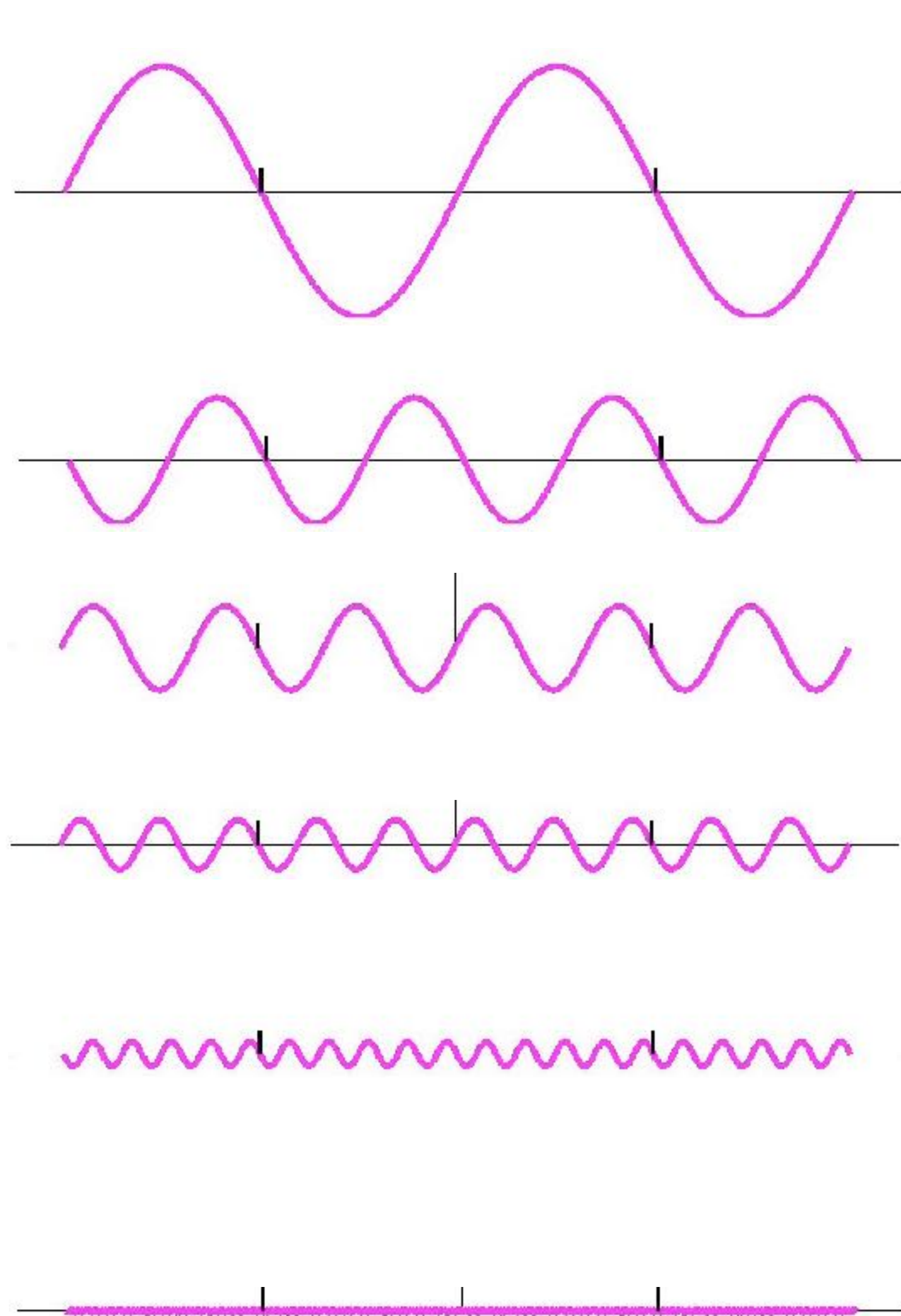




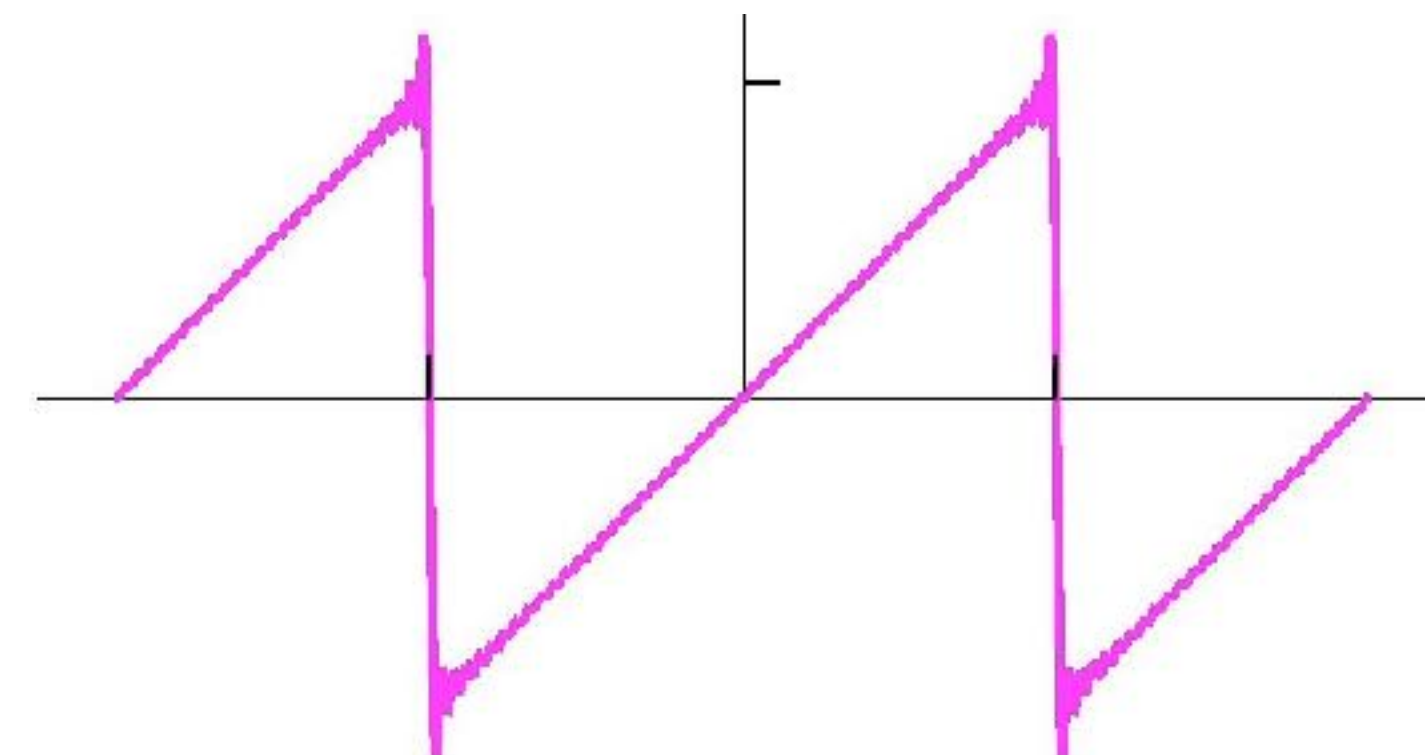
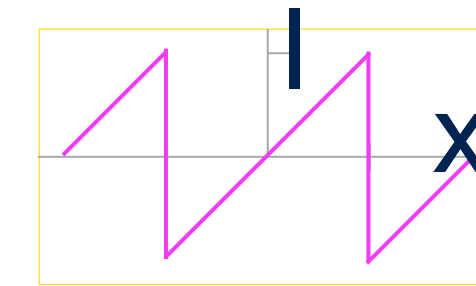
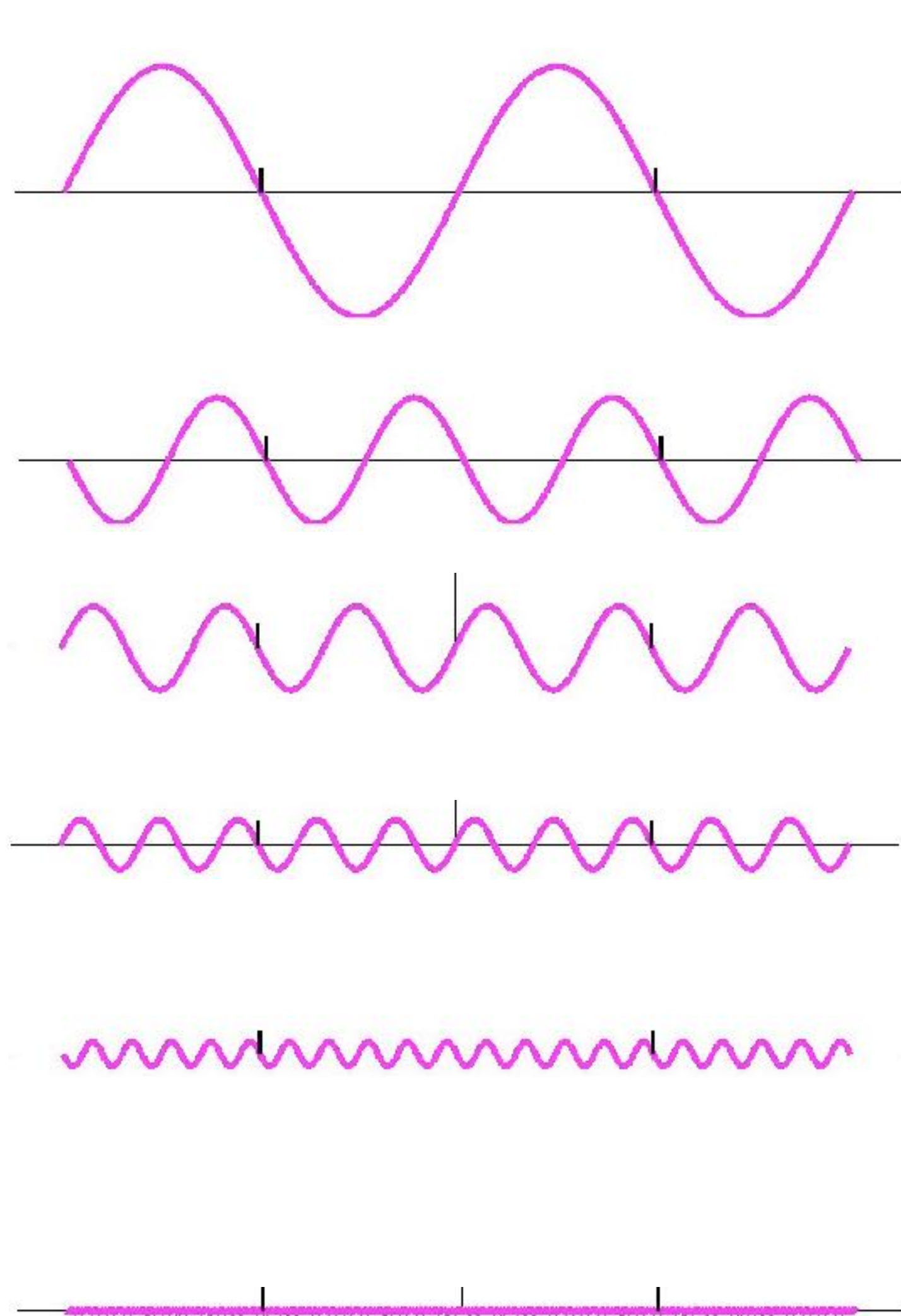
Harmonic Summation



Harmonic Summation

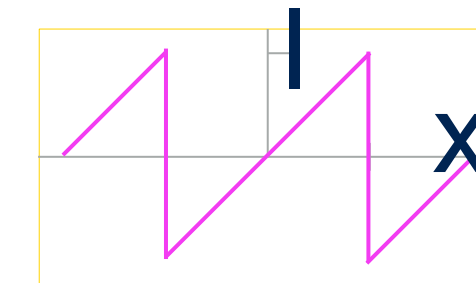
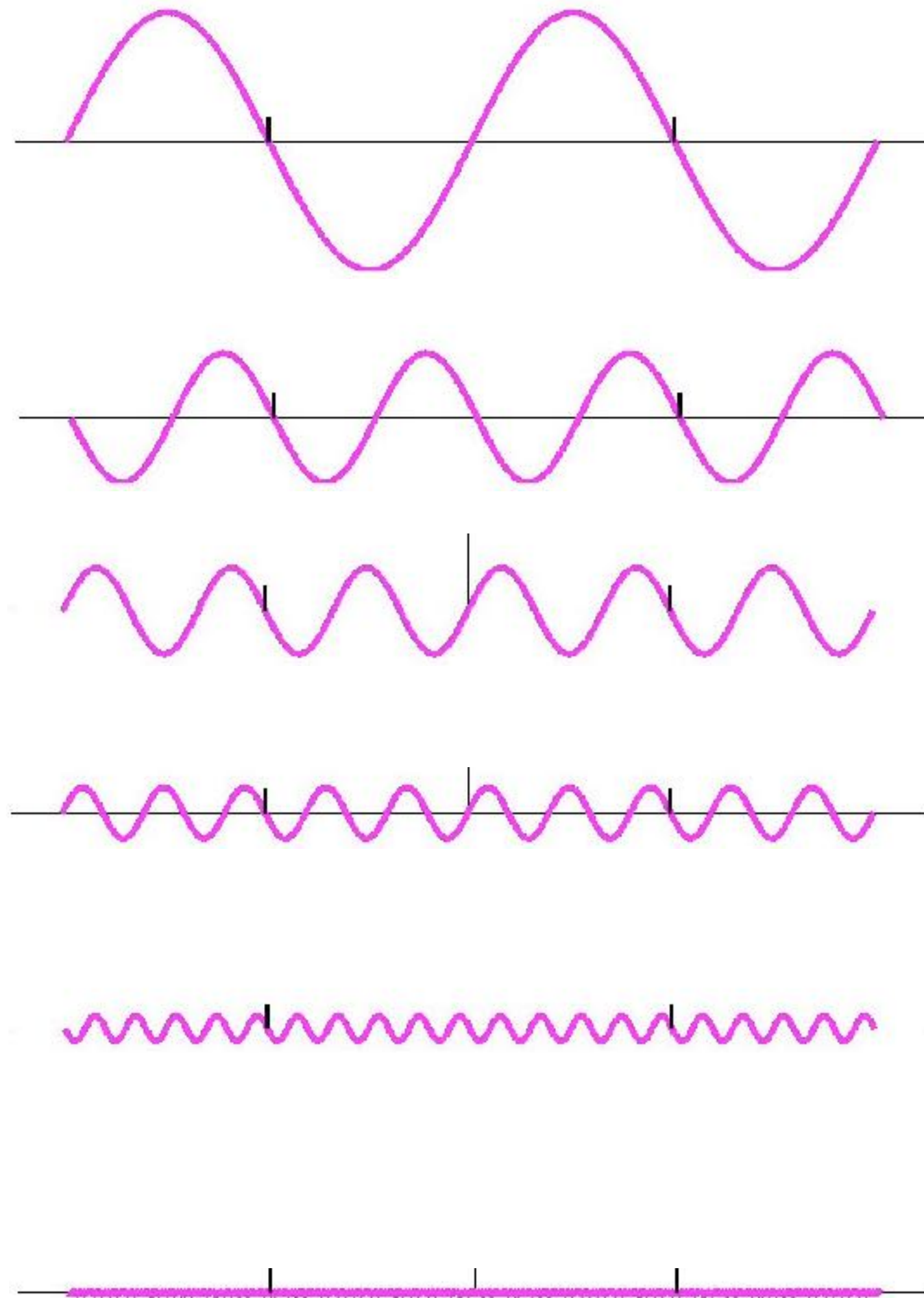


Harmonic Summation

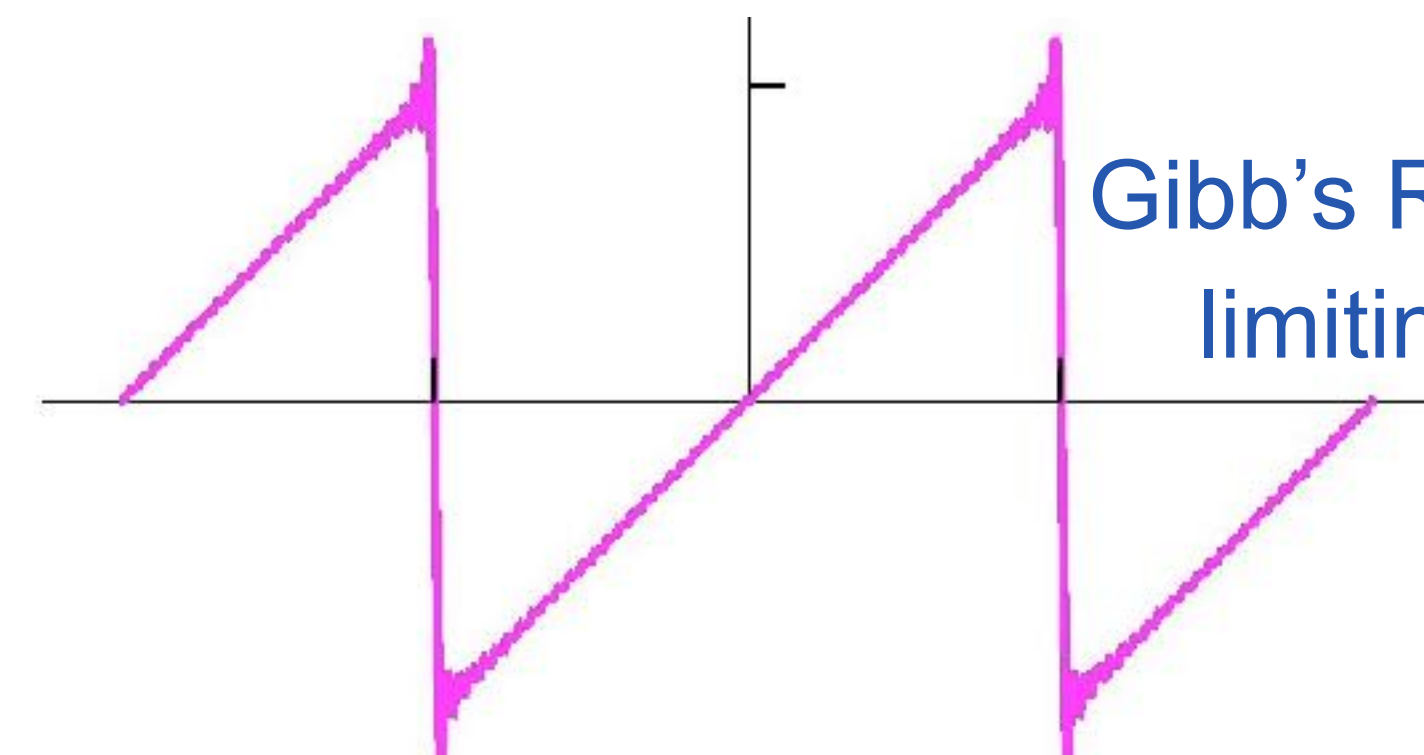




Harmonic Summation



Approximately equal
Becomes more similar with increasing n



Gibb's Ringing due to band-limiting the frequency

Fourier Transform



$$F(\mu) = \int_{-\infty}^{+\infty} f(x) e^{-i2\pi\mu x} dx$$

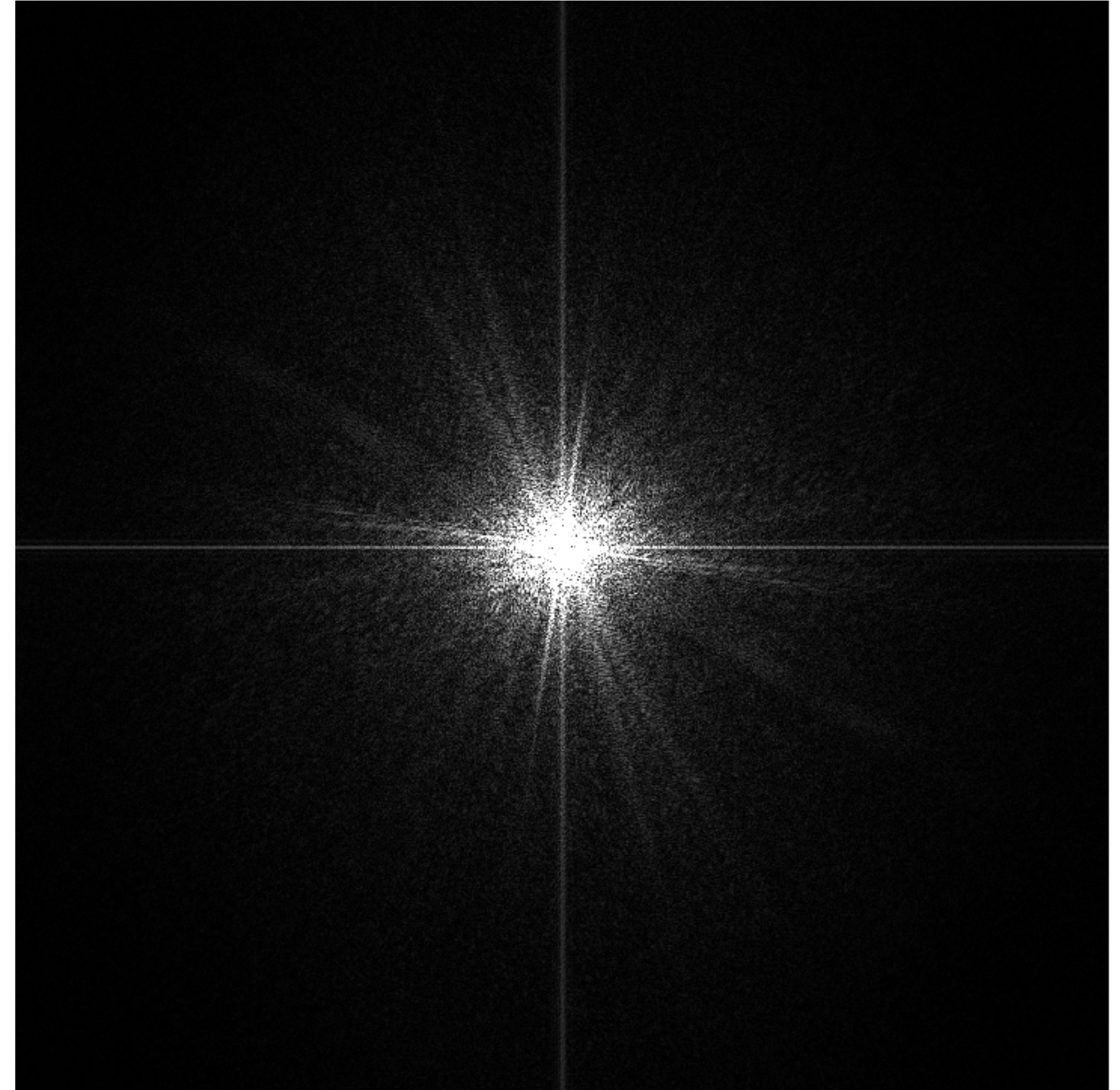


$$f(x) = \int_{-\infty}^{+\infty} F(\mu) e^{i2\pi\mu x} d\mu$$

Fourier transform of an image



Spatial domain

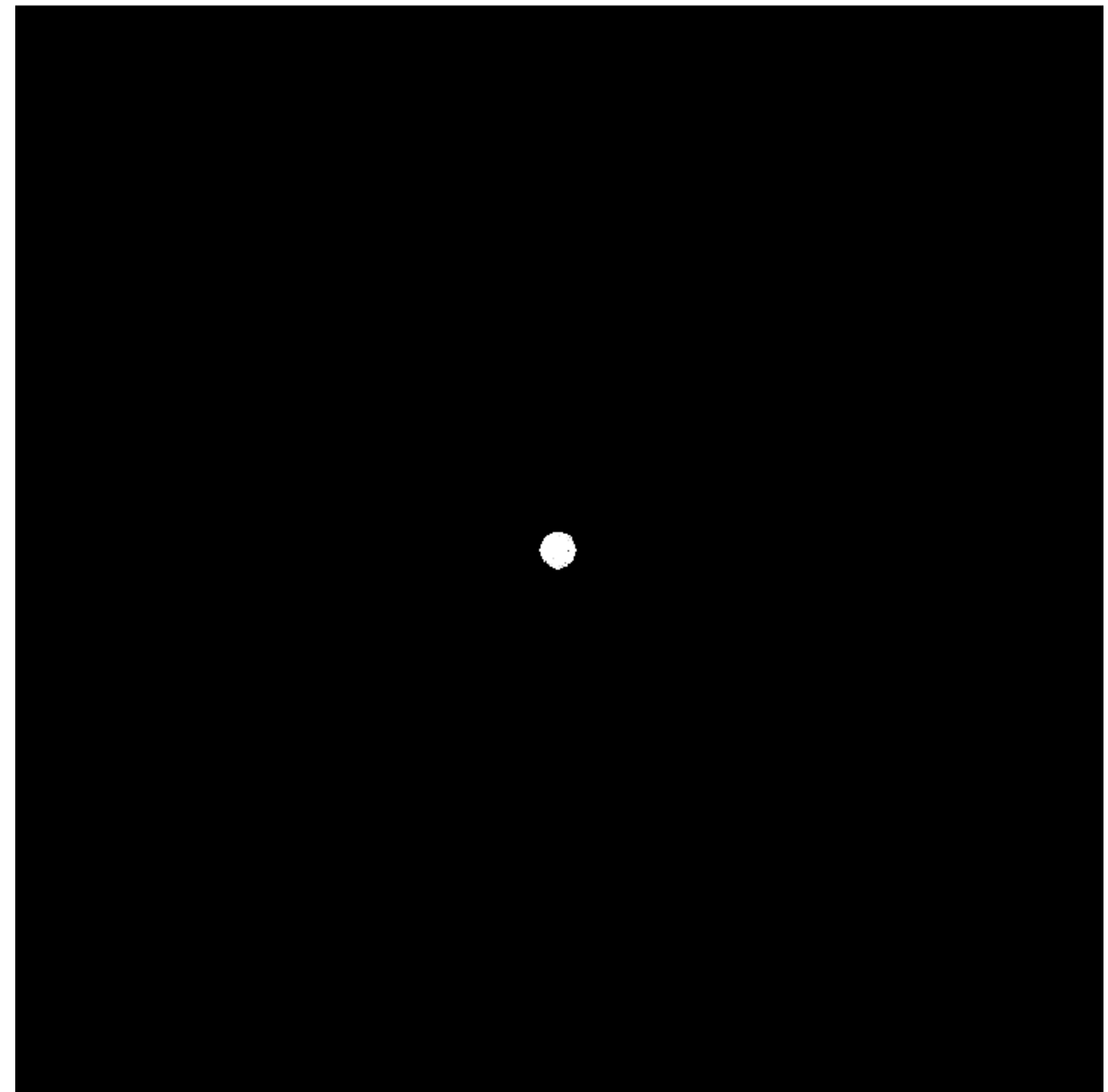


Frequency domain

Low frequencies



Spatial domain

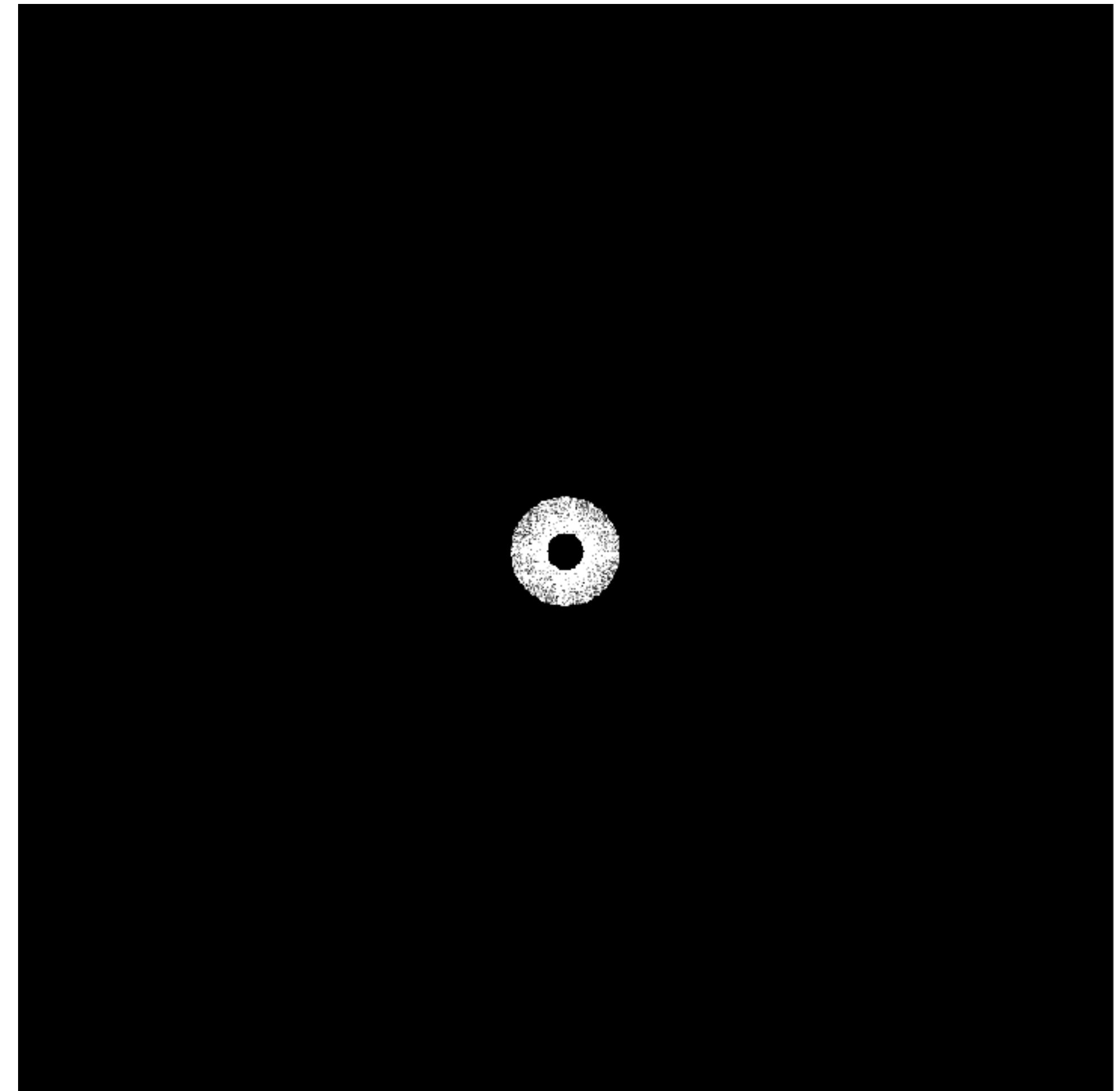


Frequency domain

Medium frequencies



Spatial domain

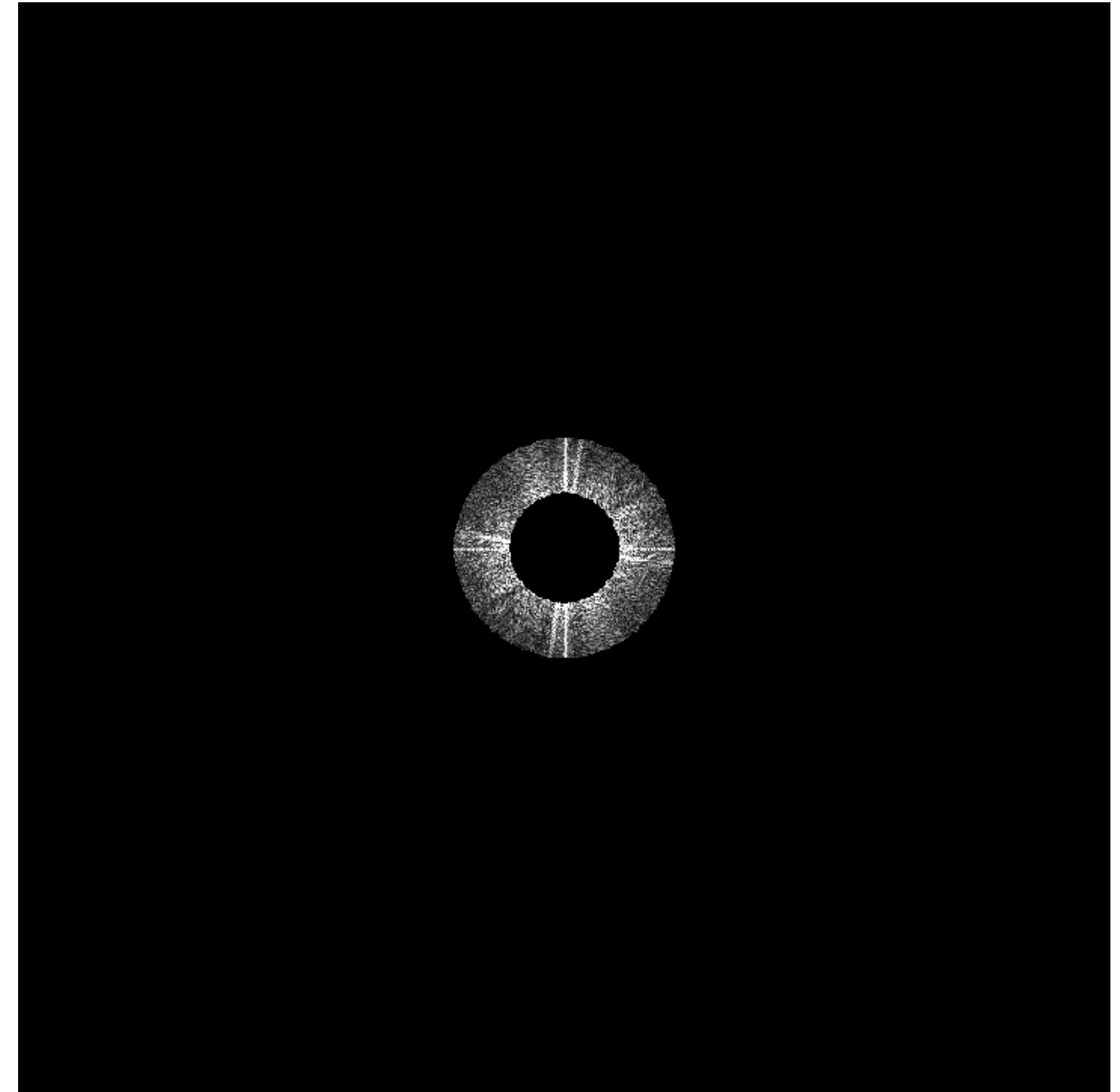


Frequency domain

Medium frequencies



Spatial domain

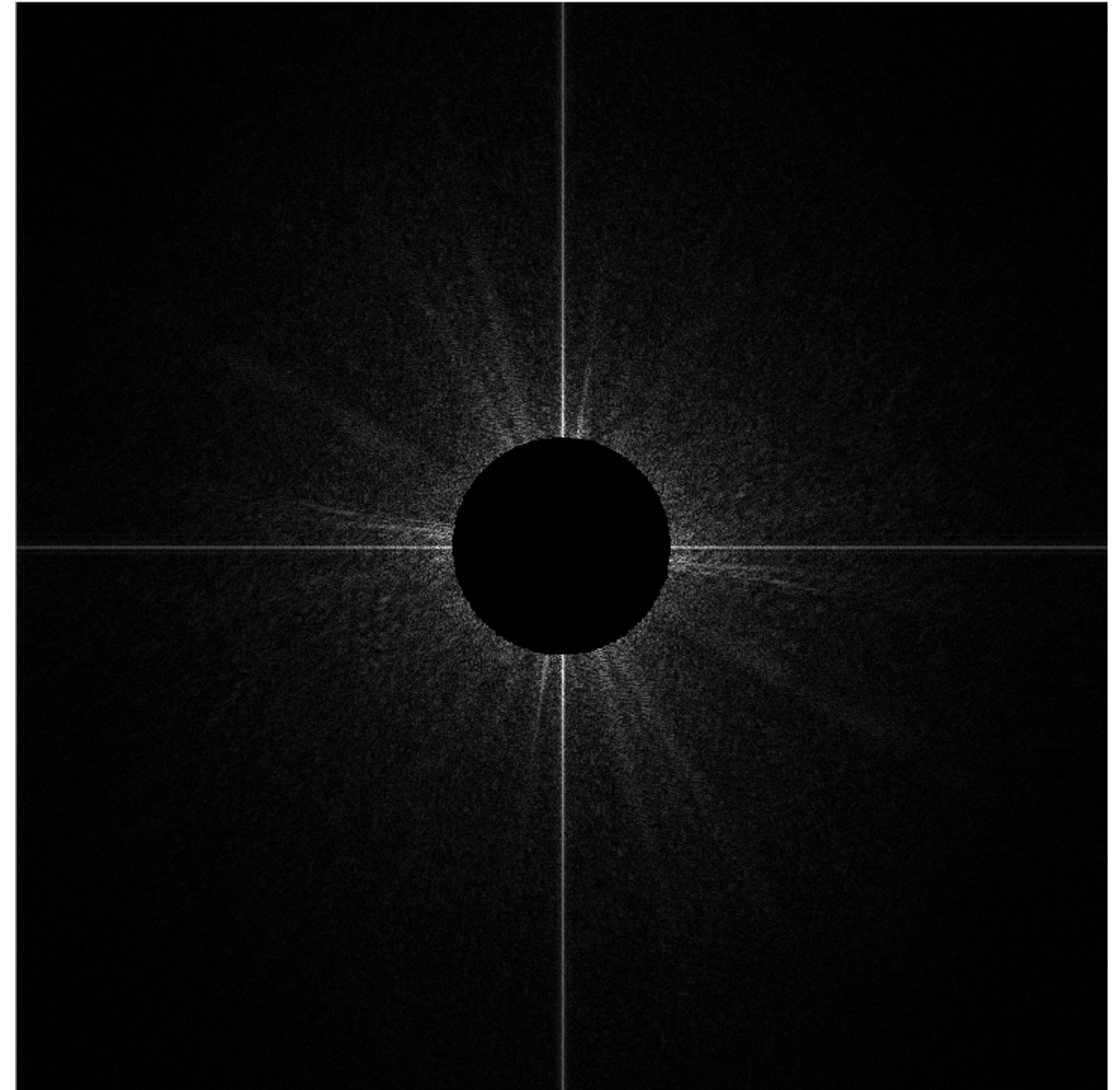


Frequency domain

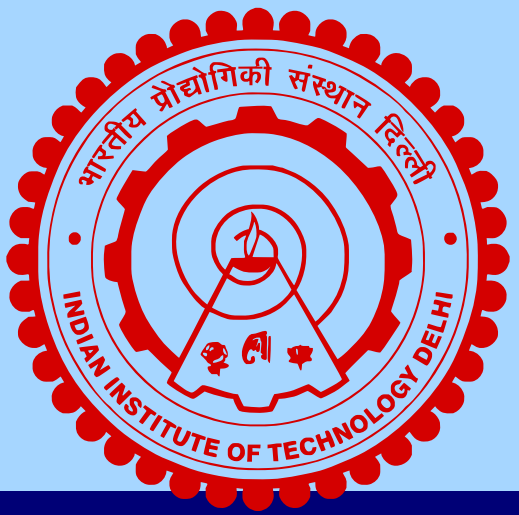
High frequencies



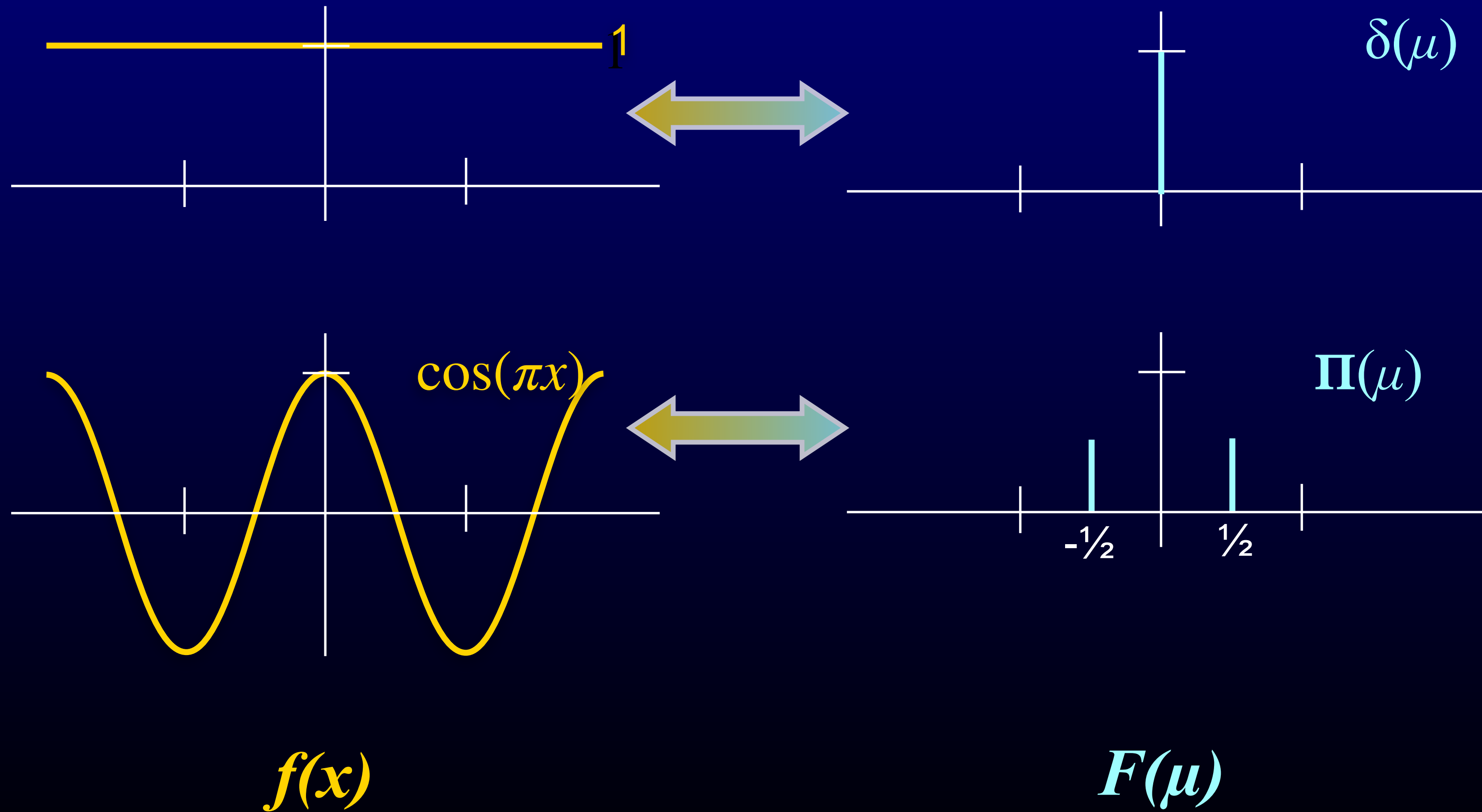
Spatial domain



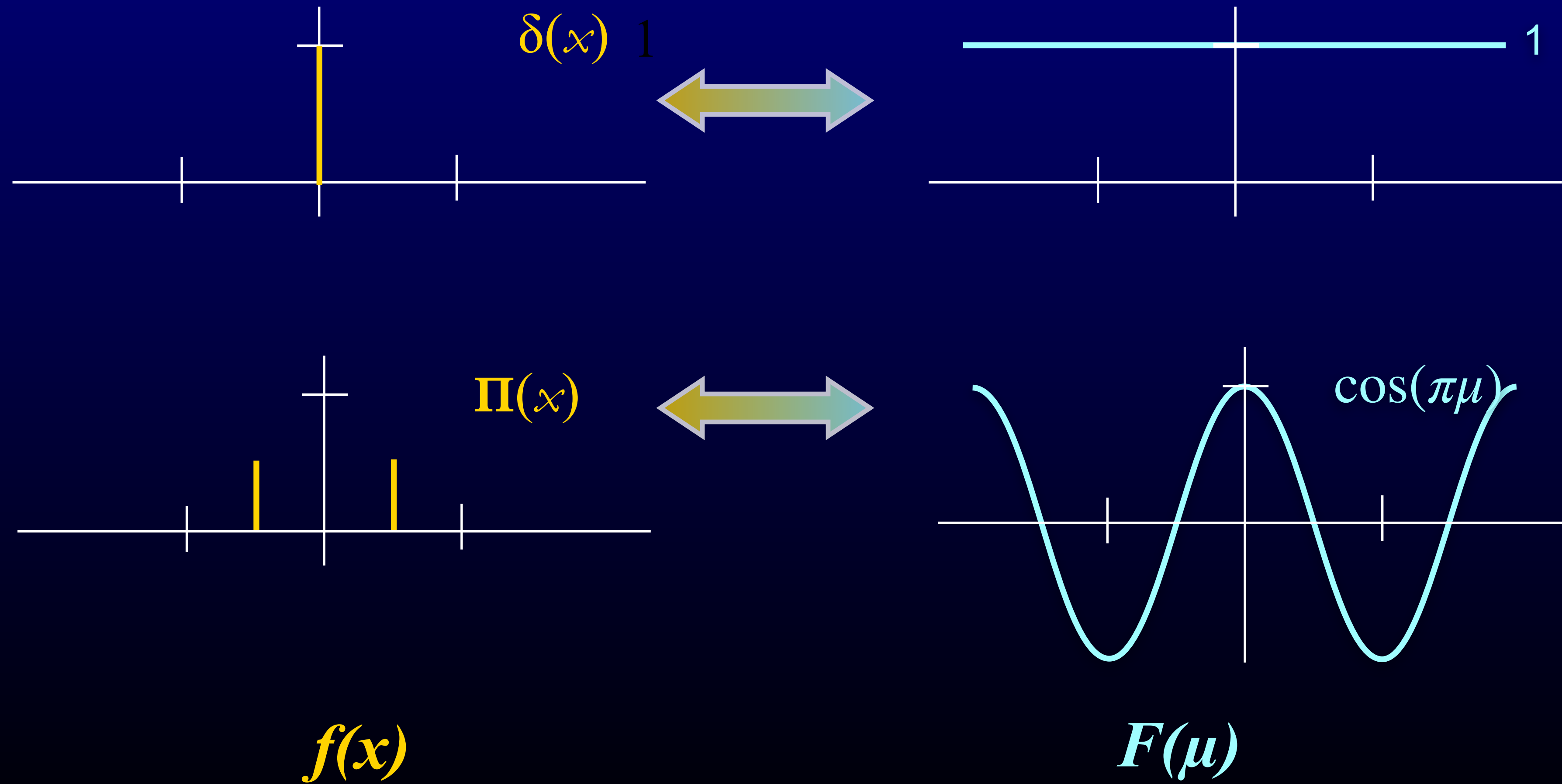
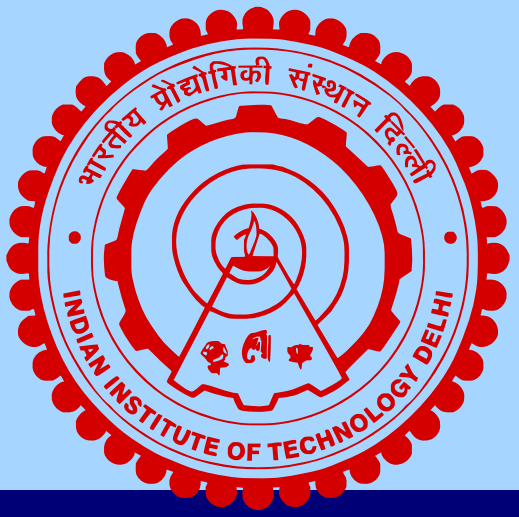
Frequency domain

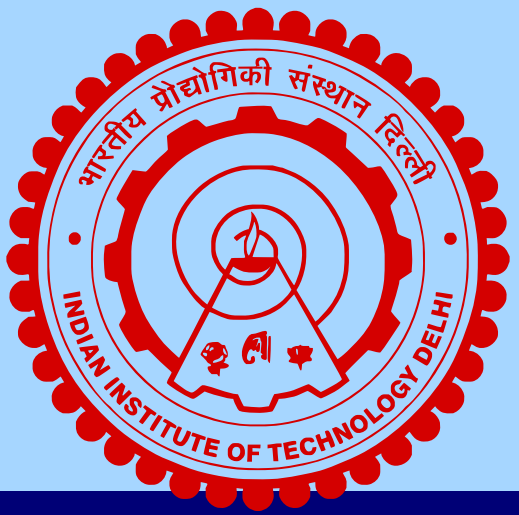


Fourier Transform Pairs

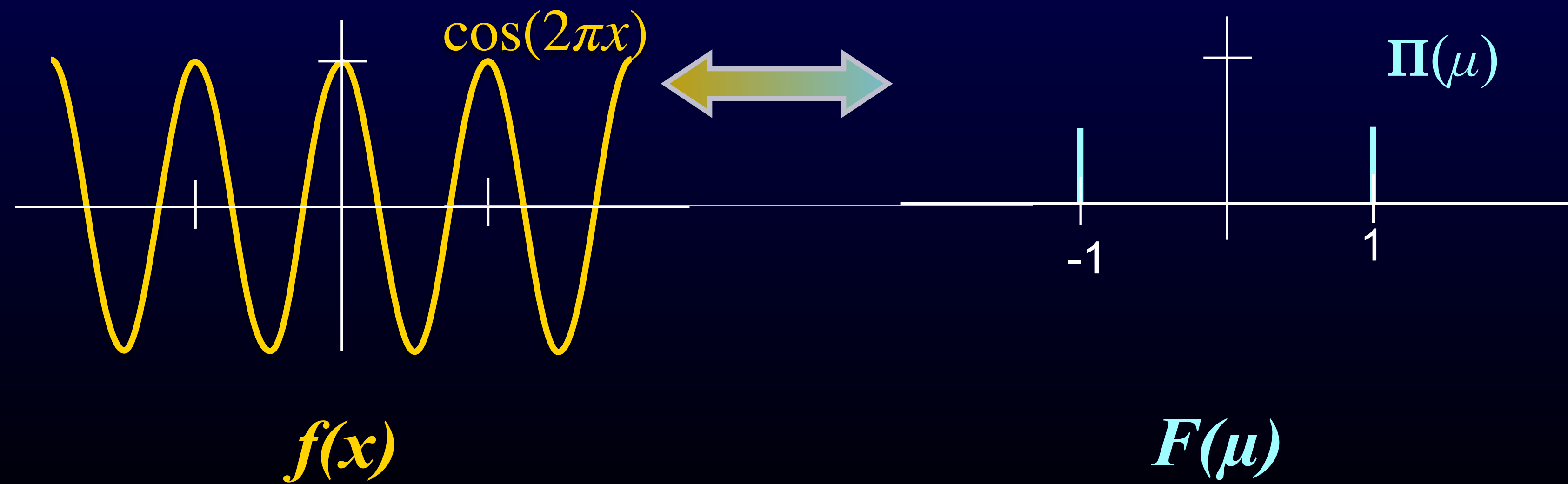
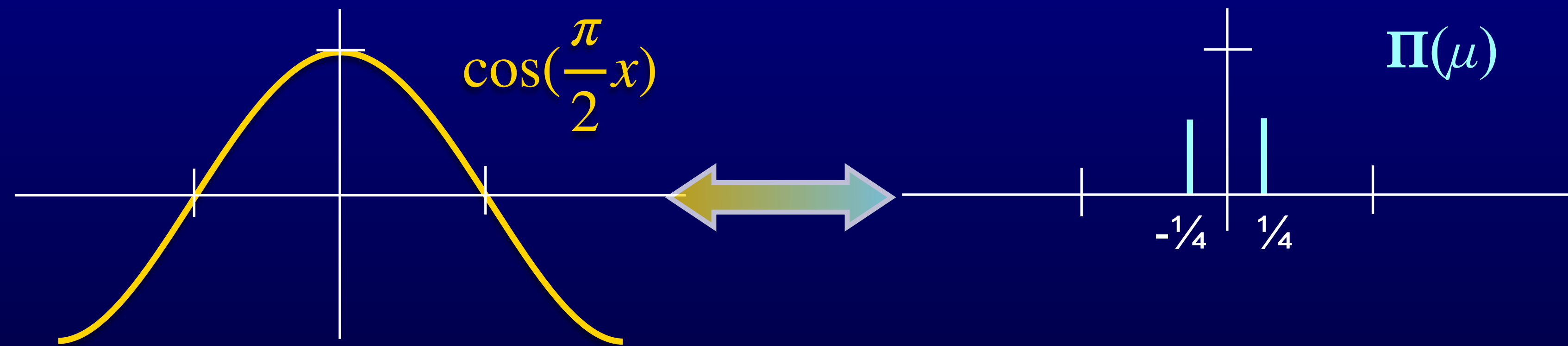


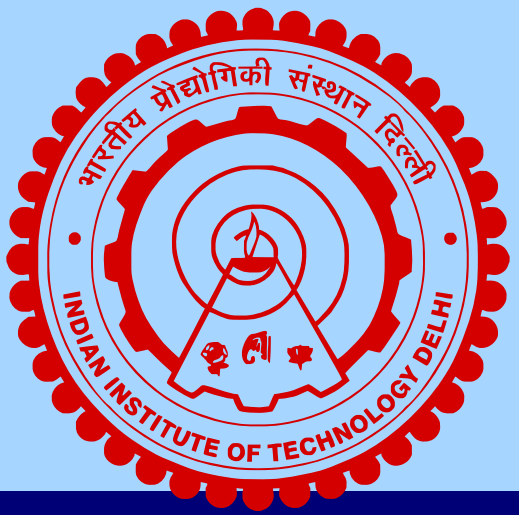
Reciprocity



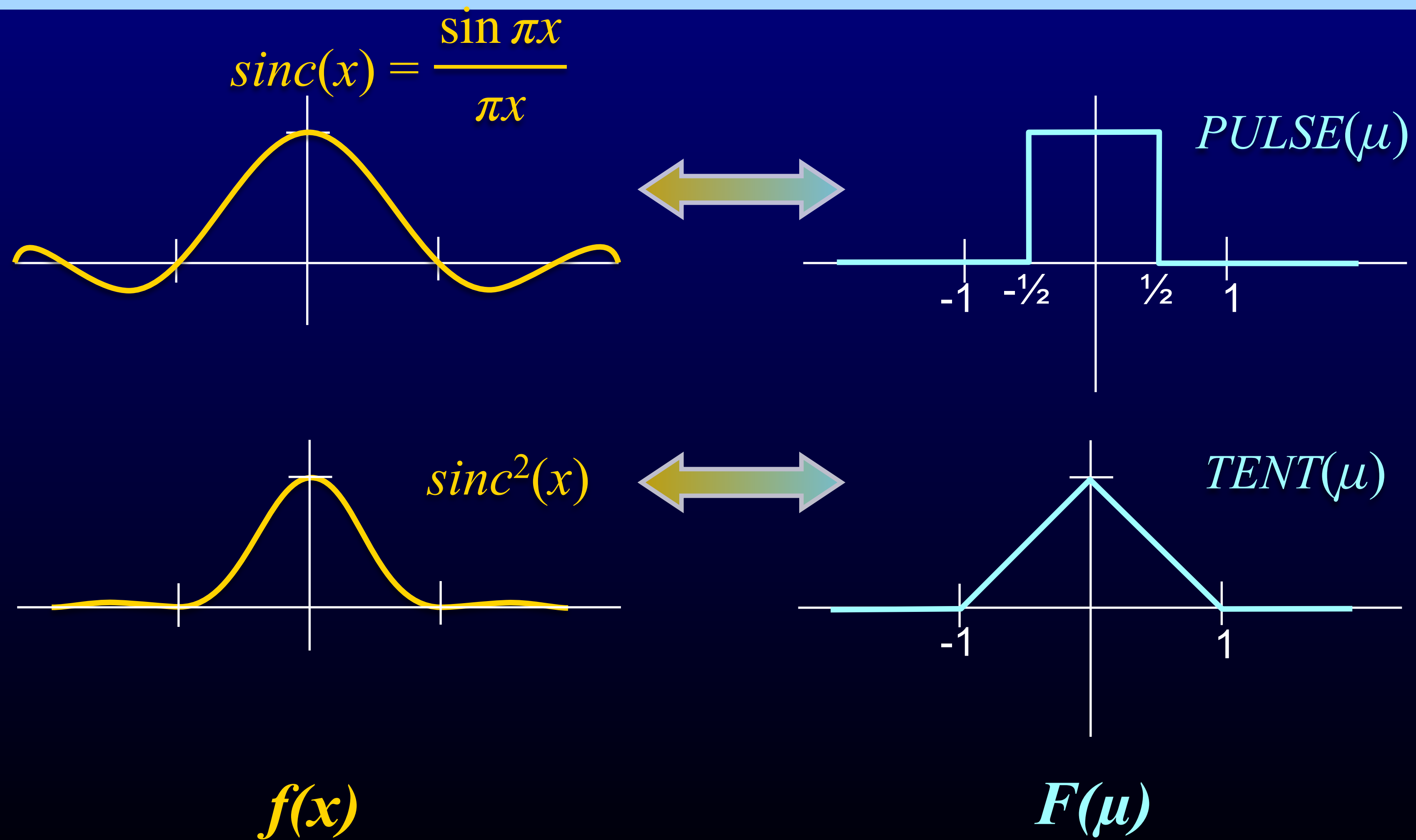


Inverse Scaling

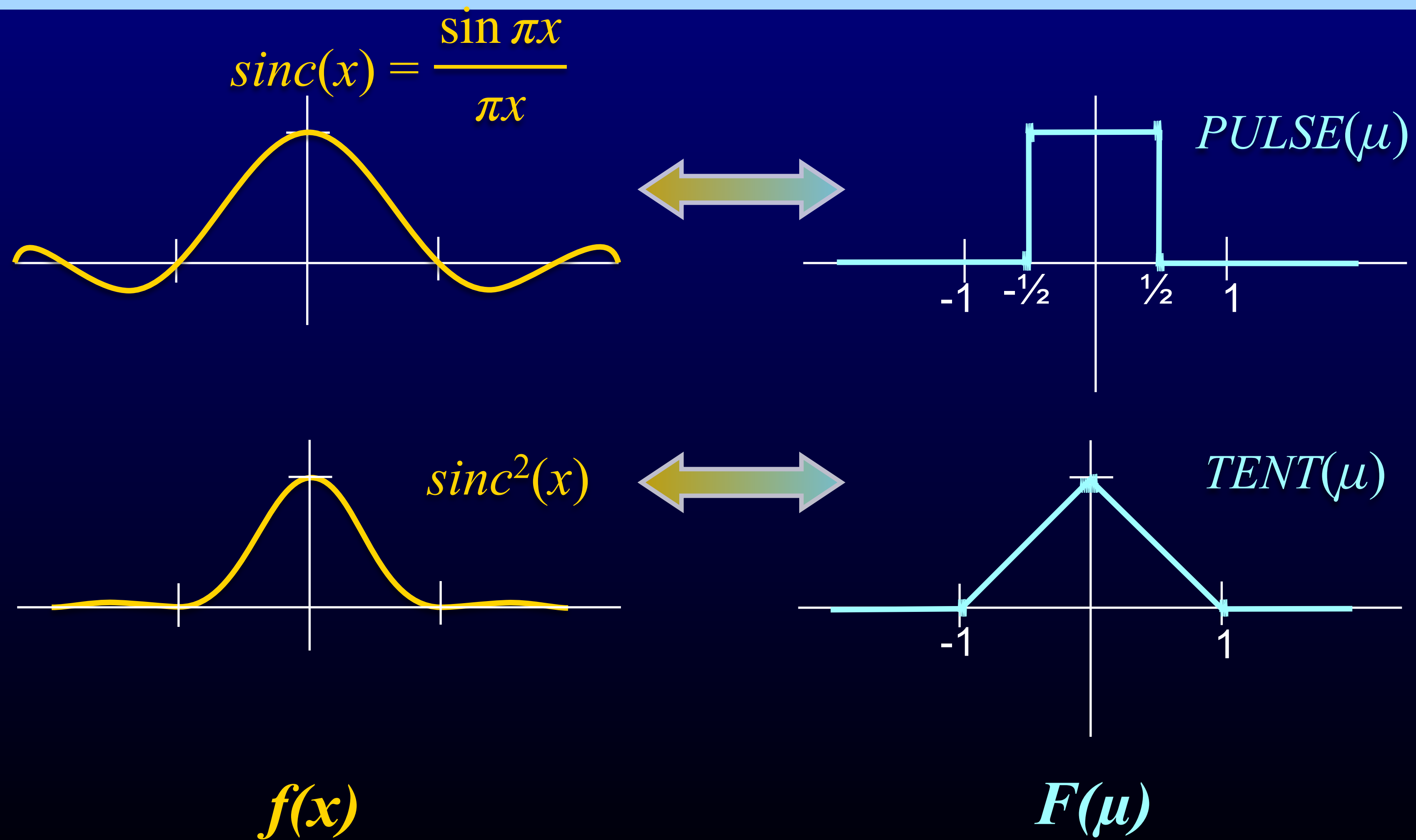
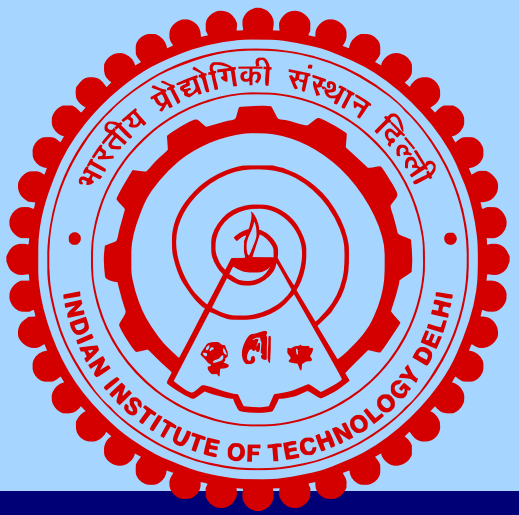


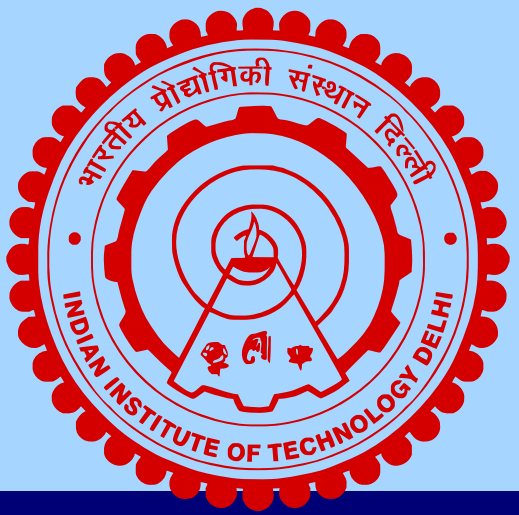


Band-limited Transform Pairs

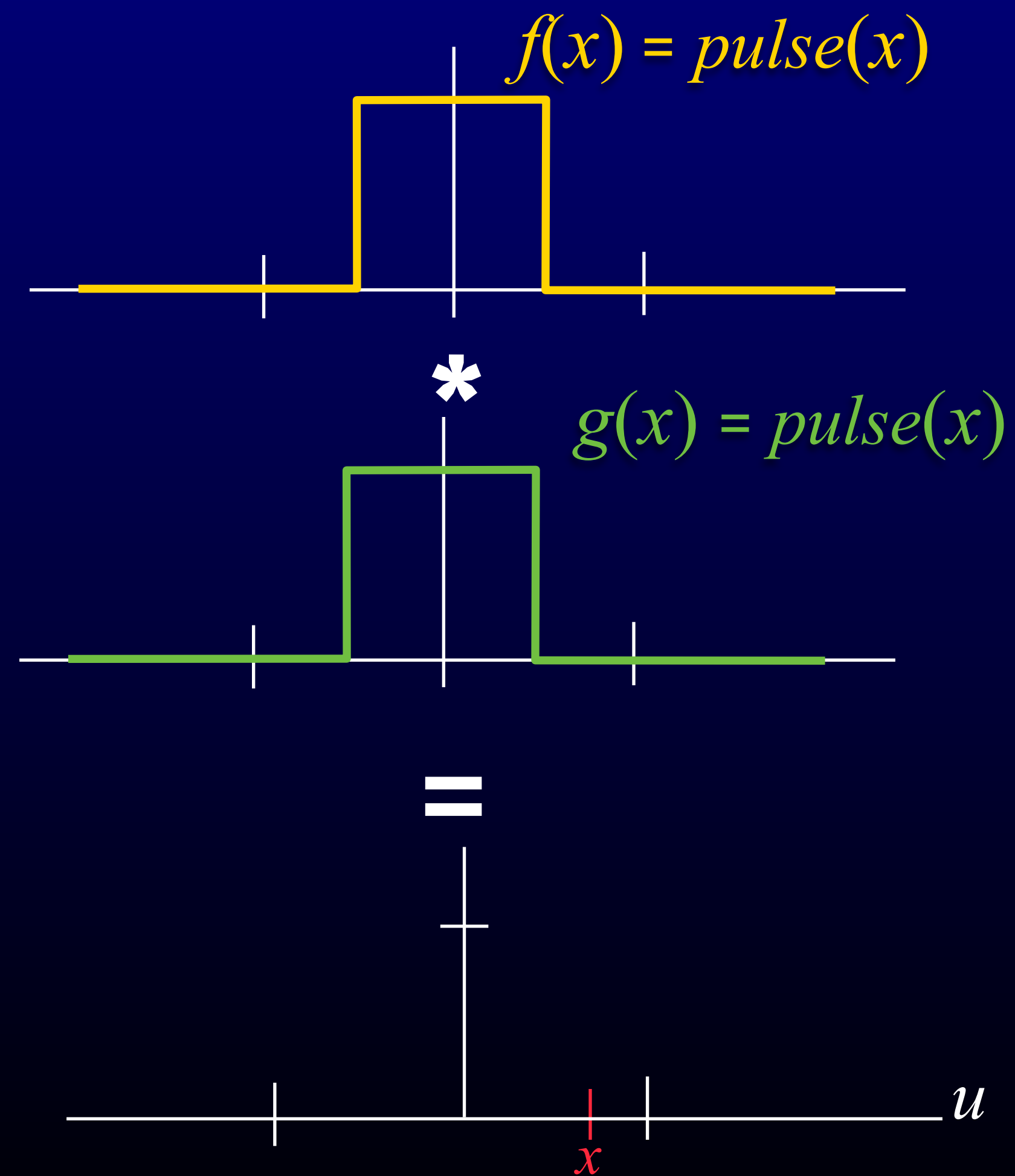


Band-limited Transform Pairs



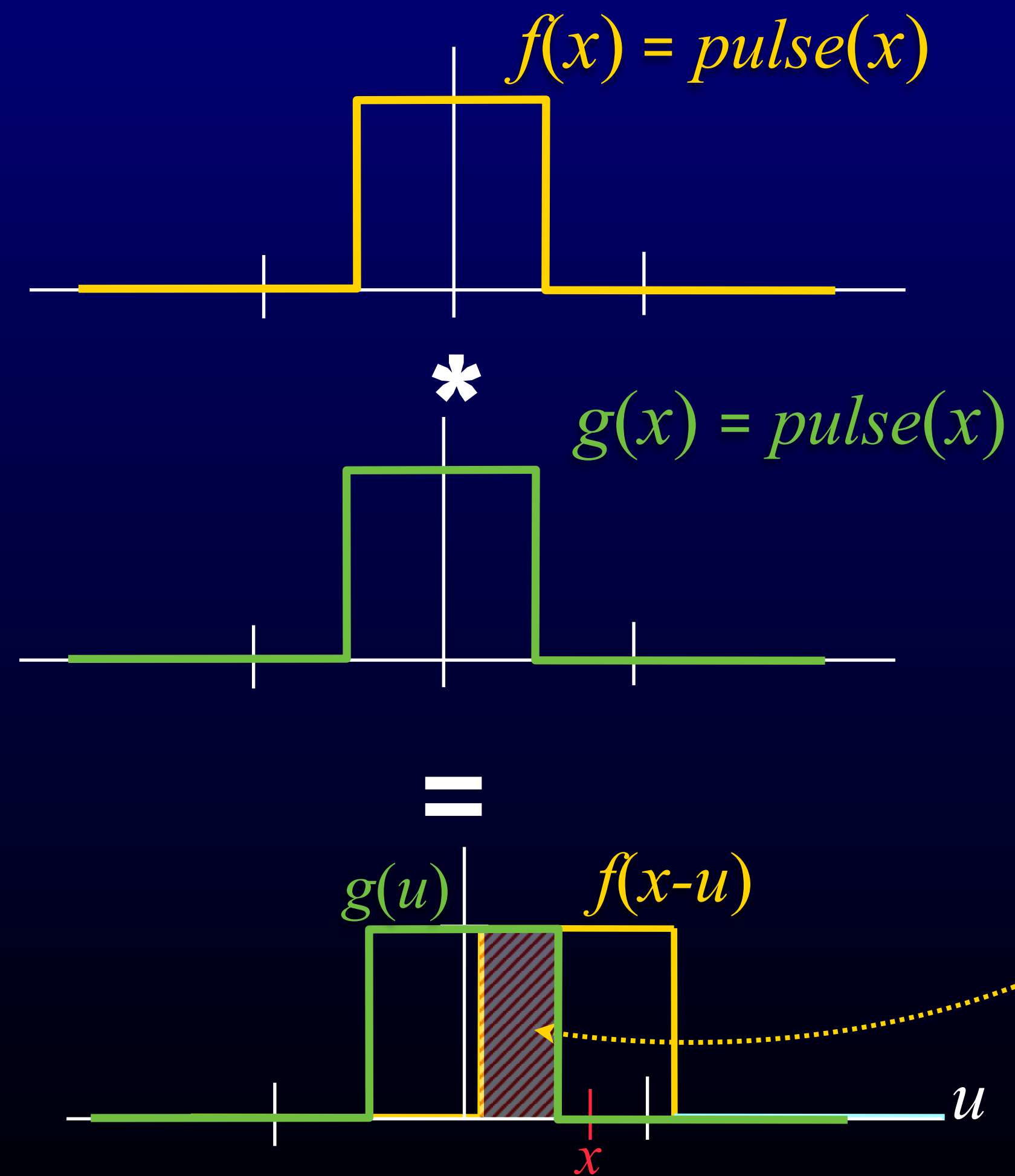


Convolution



$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(x-u)g(u)du$$

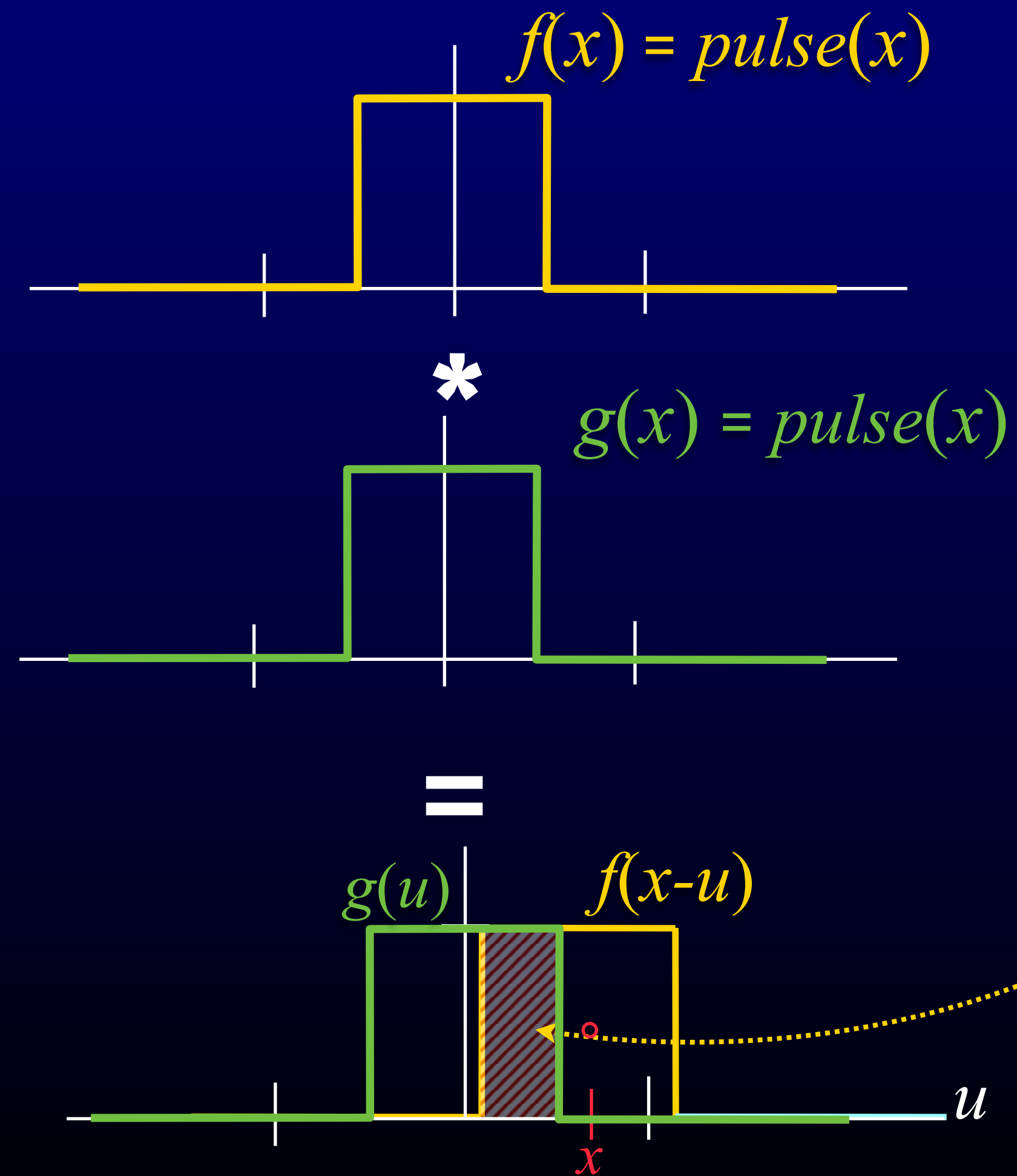
Convolution



$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(x-u)g(u)du$$

shaded area

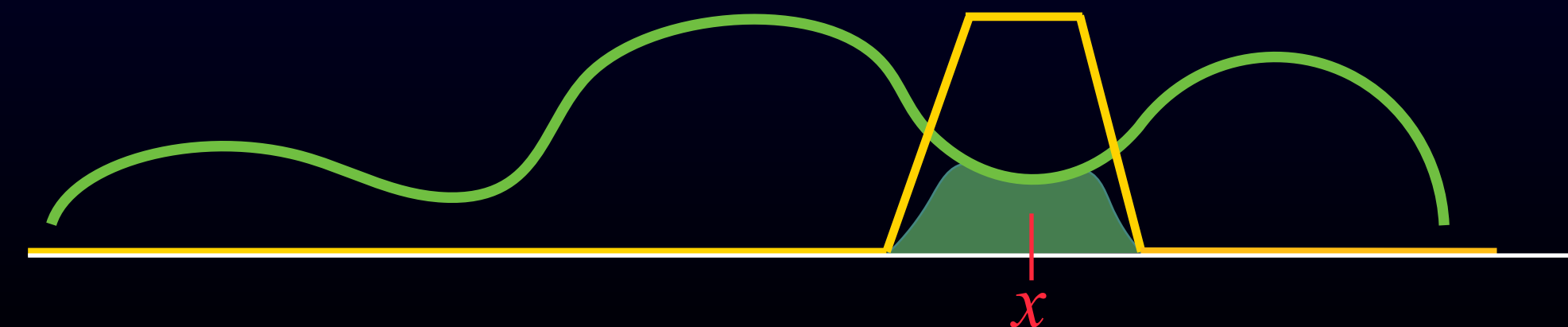
Convolution



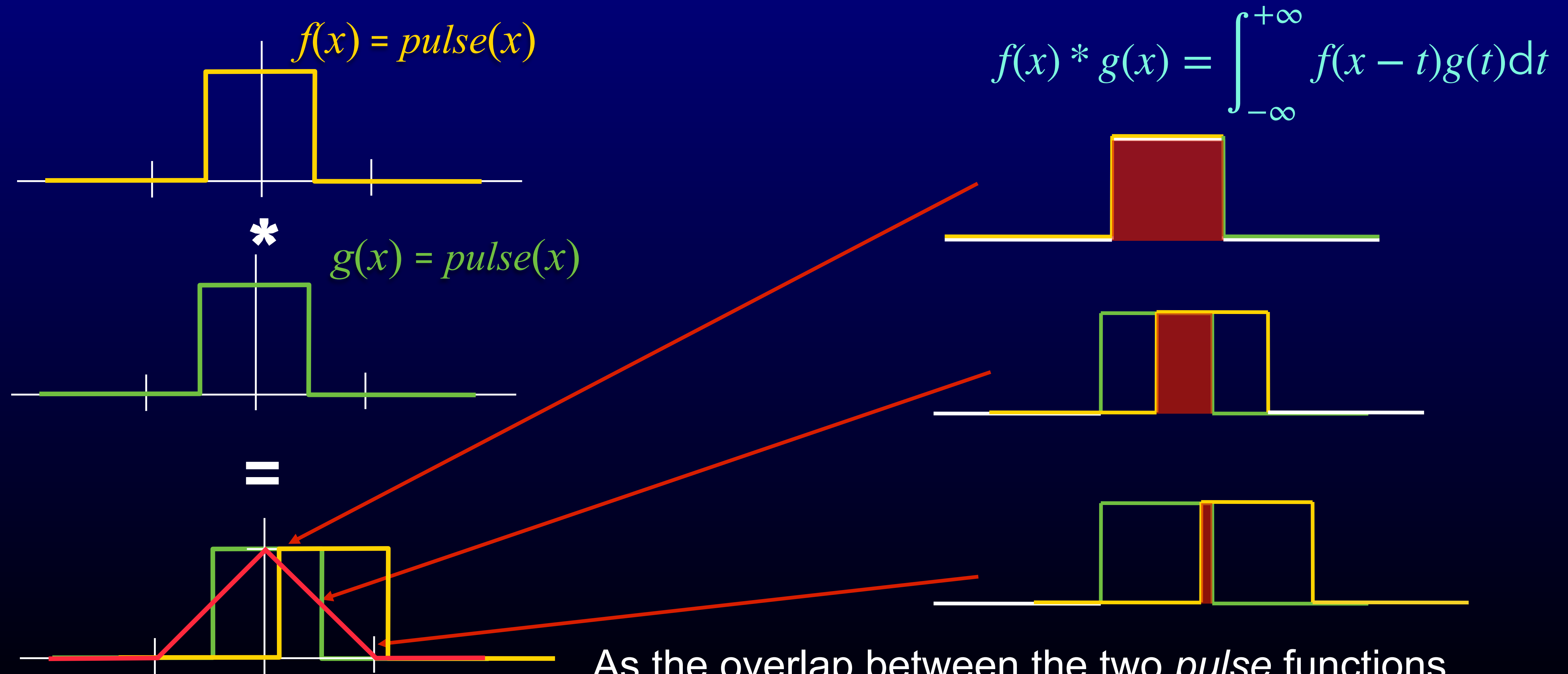
$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(x-u)g(u)du$$

shaded area

Weighted average of $g(u)$,
weights given by $f(x-u)$



Convolution





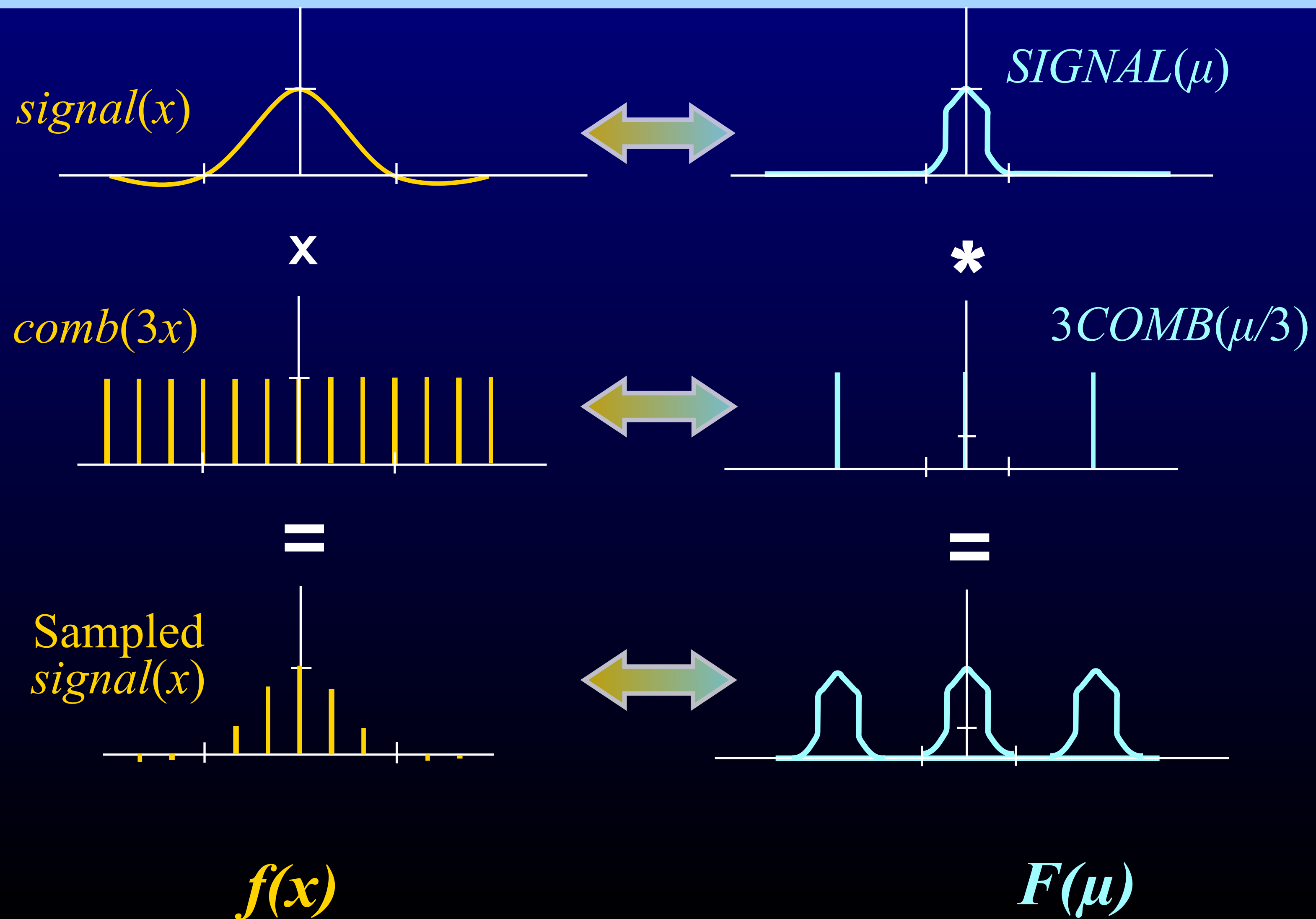
Convolution Theorem

If F and G be the transforms of f and g :

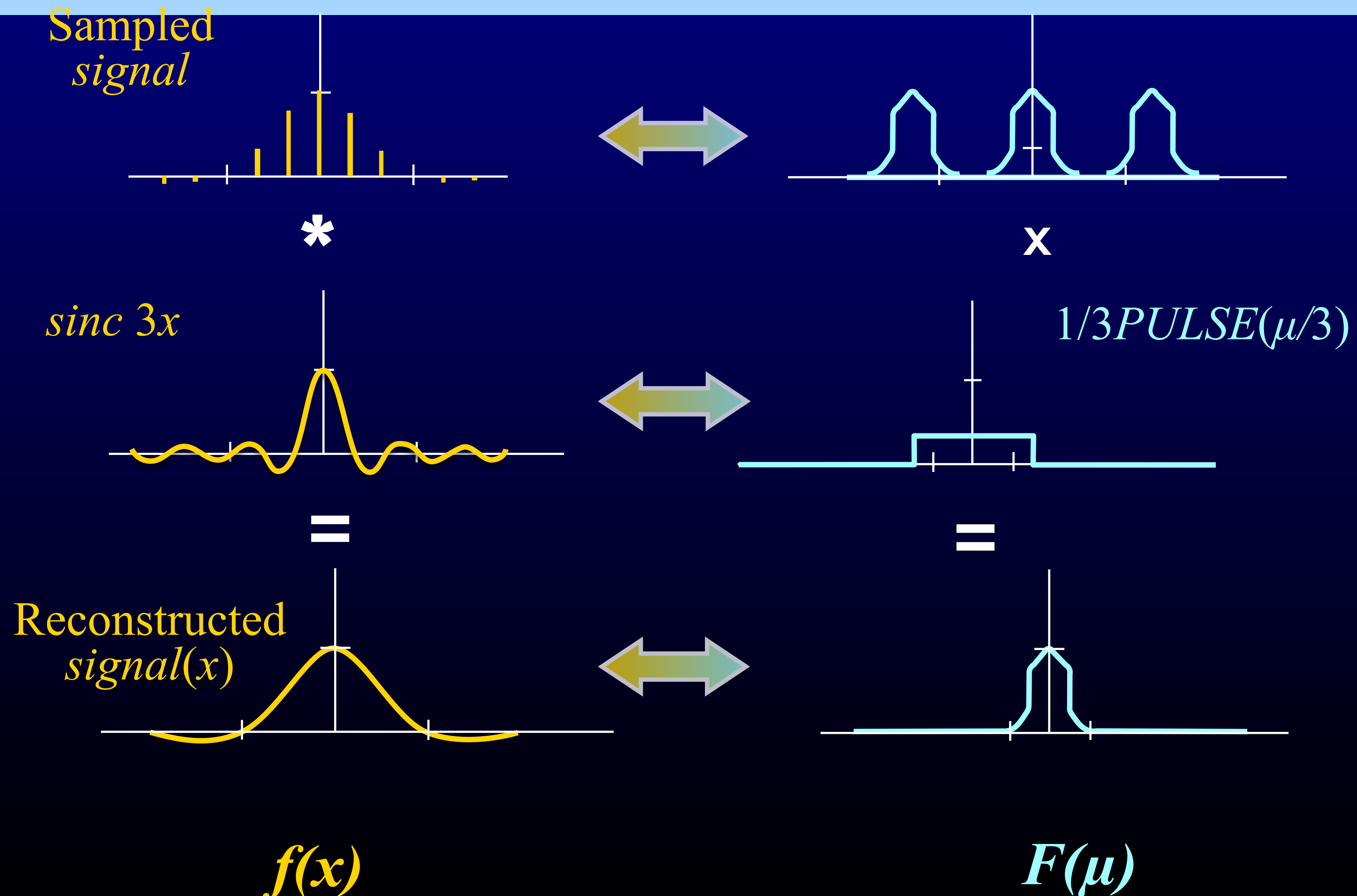
$$f(x) * g(x) \longleftrightarrow F(\mu) \times G(\mu)$$

$$f(x) \times g(x) \longleftrightarrow F(\mu) * G(\mu)$$

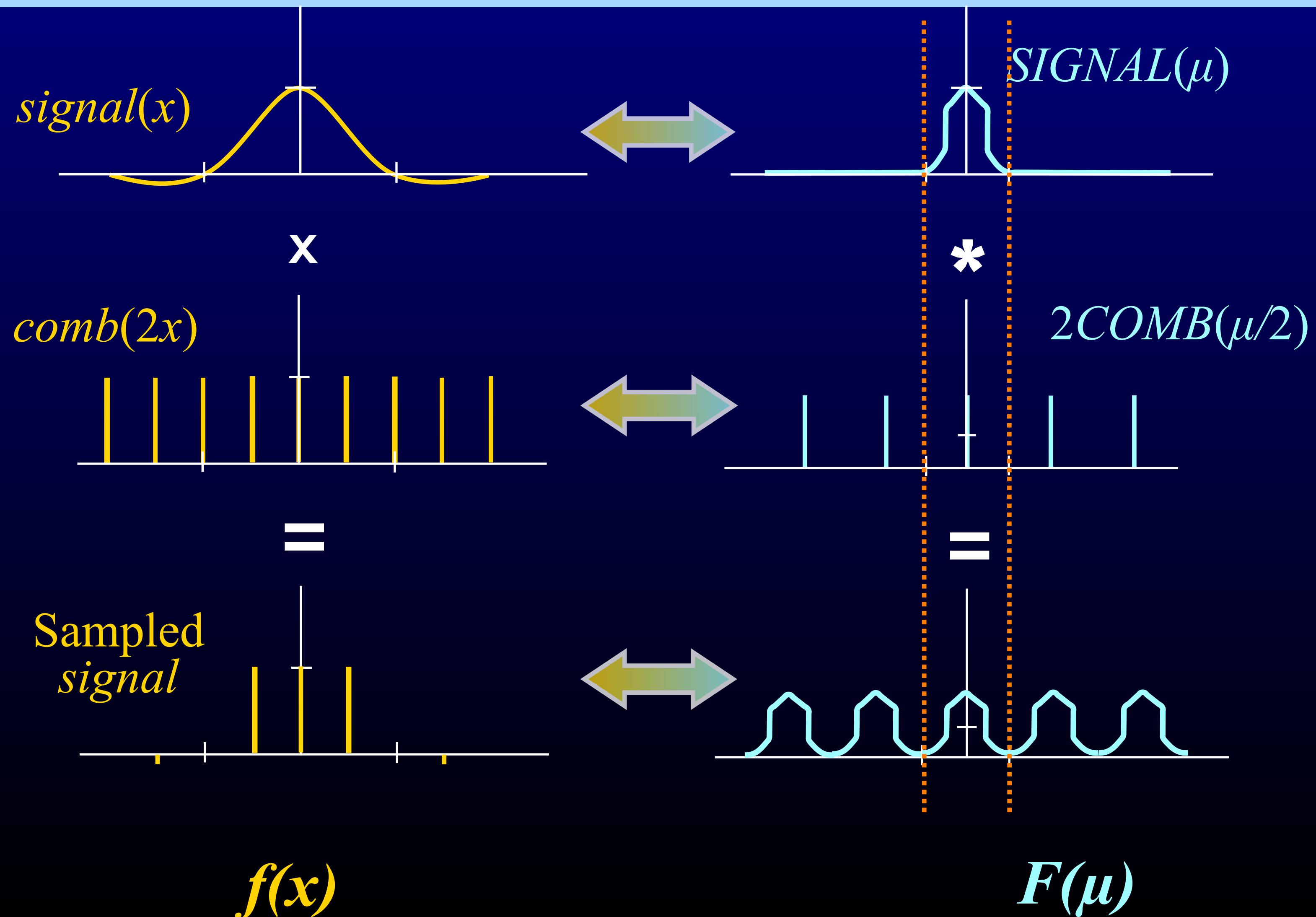
Sampling



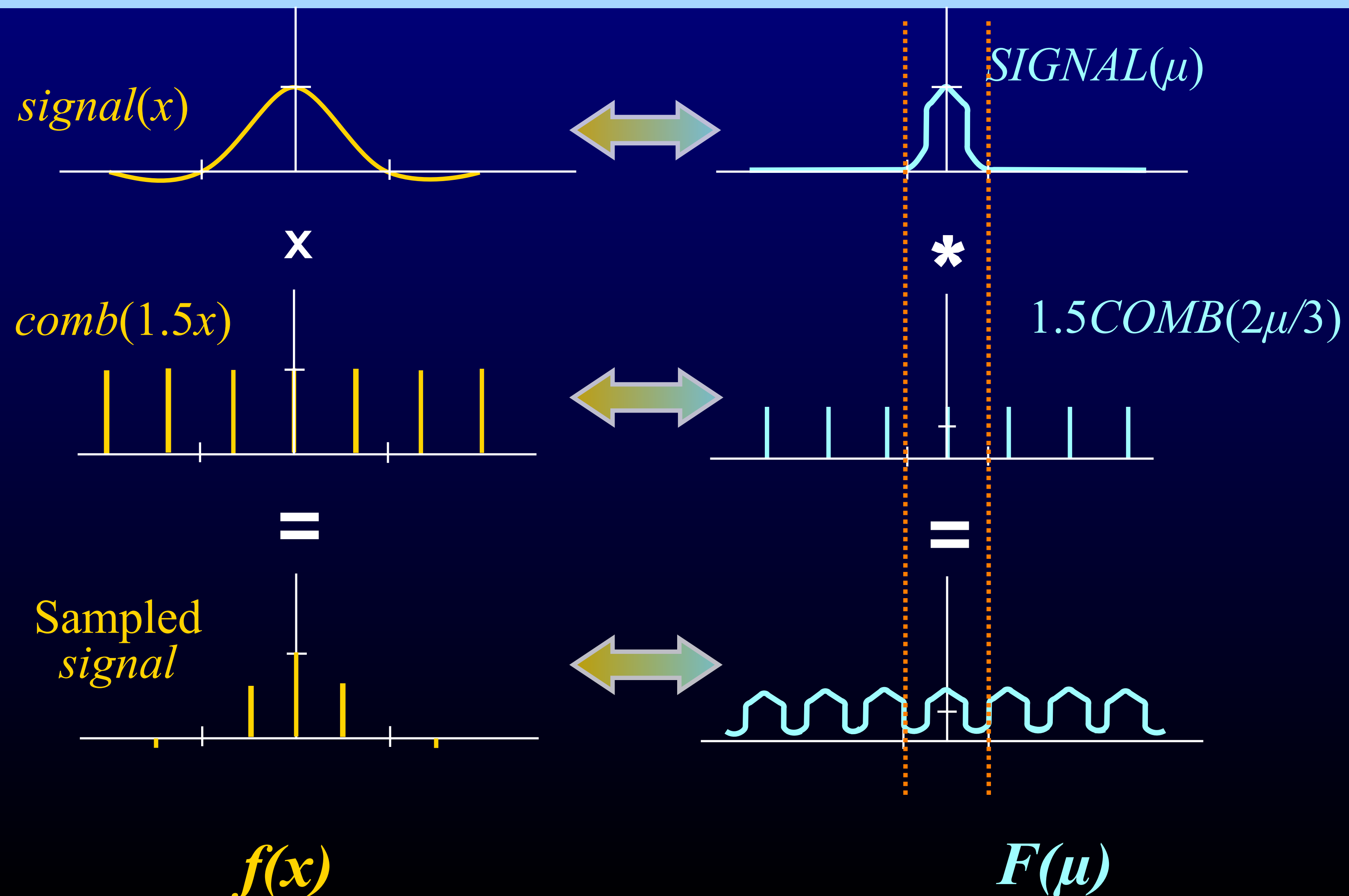
Reconstruction



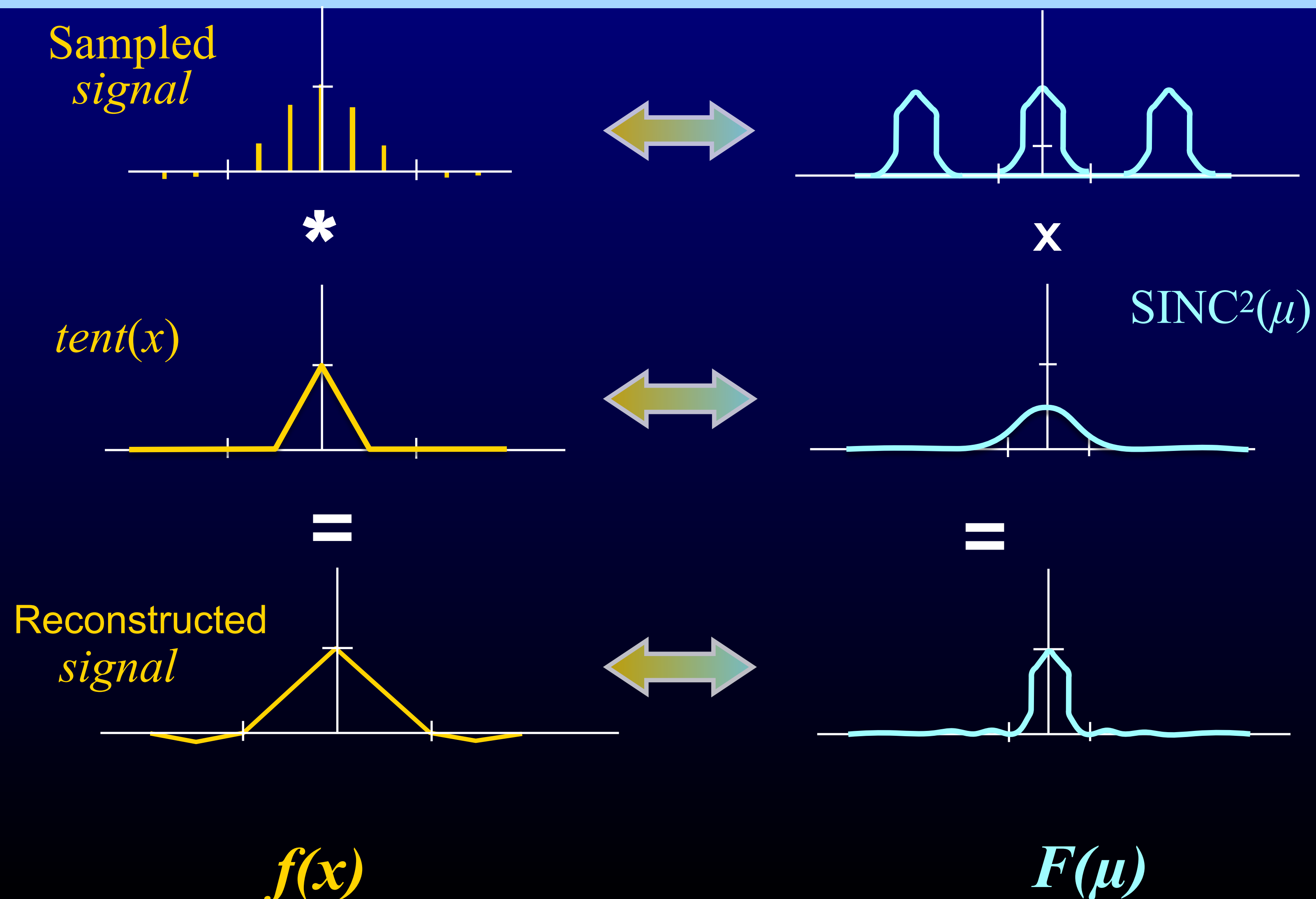
Sufficient Samples



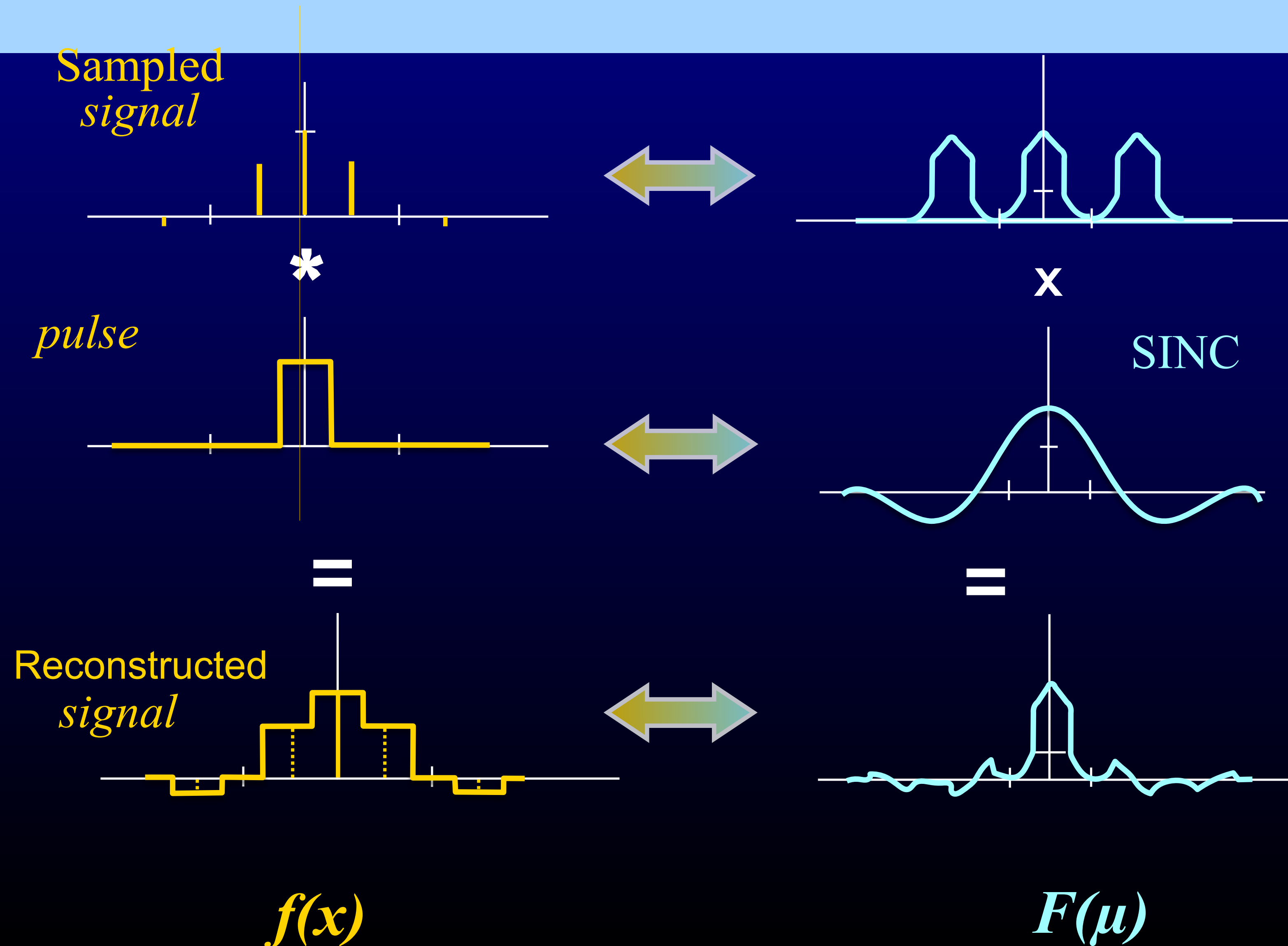
Insufficient Samples

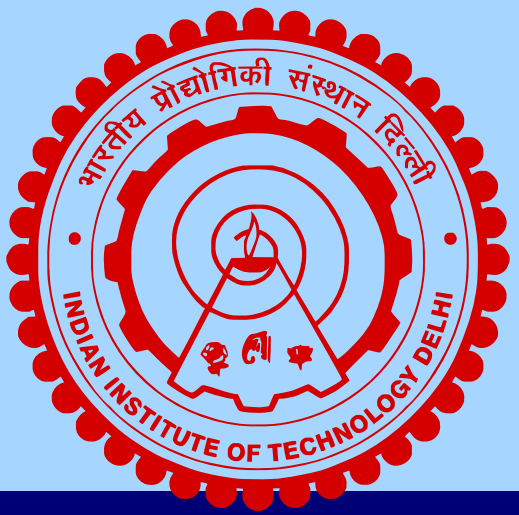


Triangle Filter

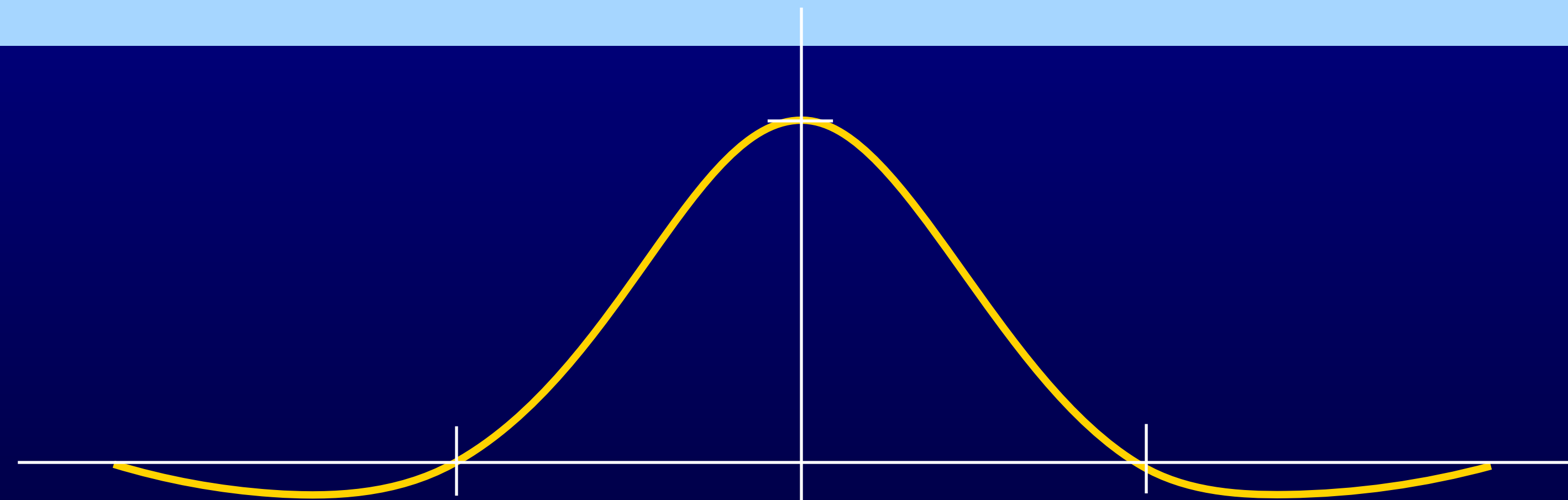


Box Filter

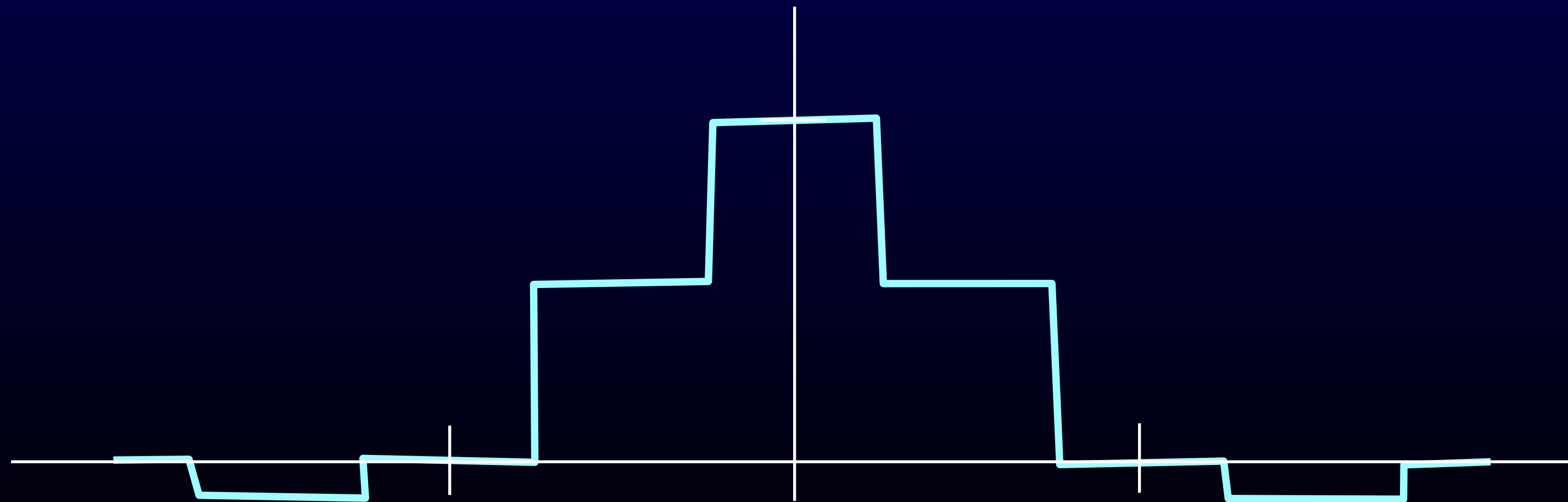




Box Reconstruction Error

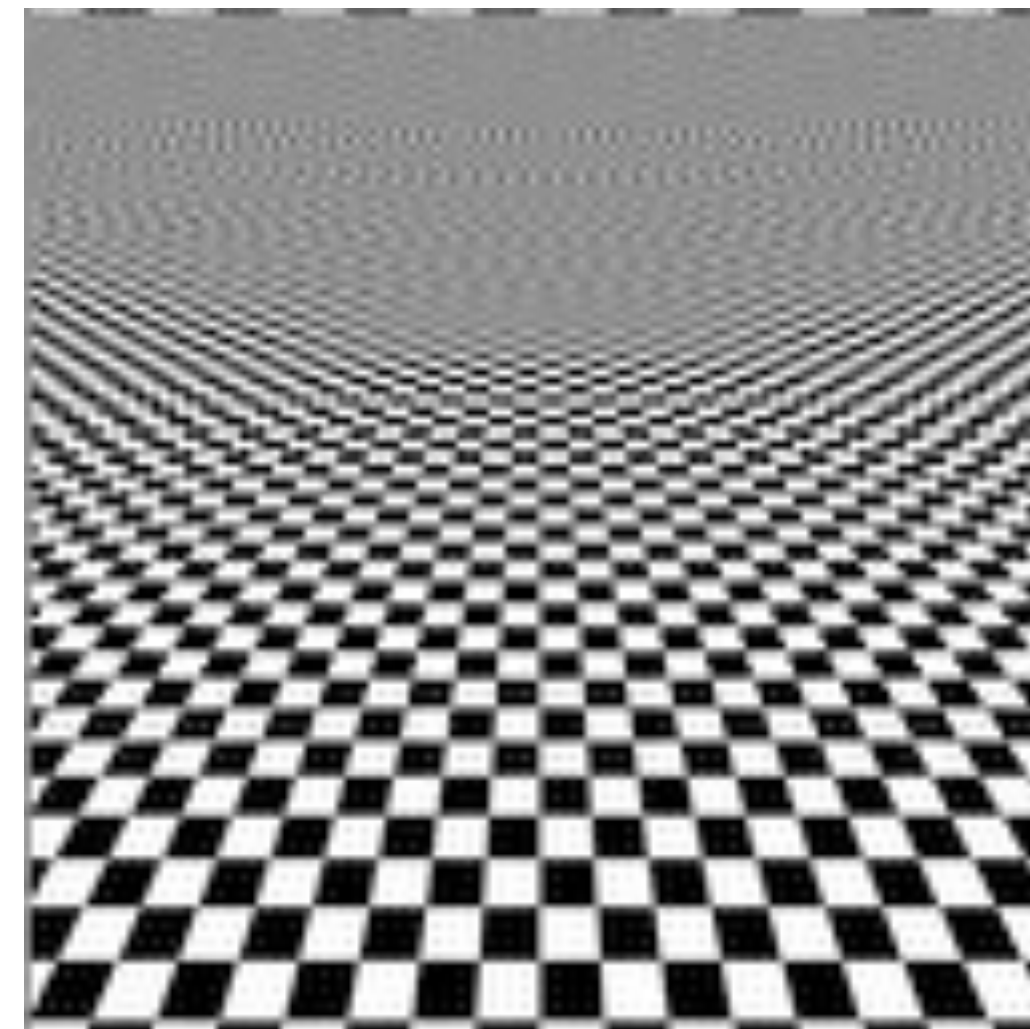
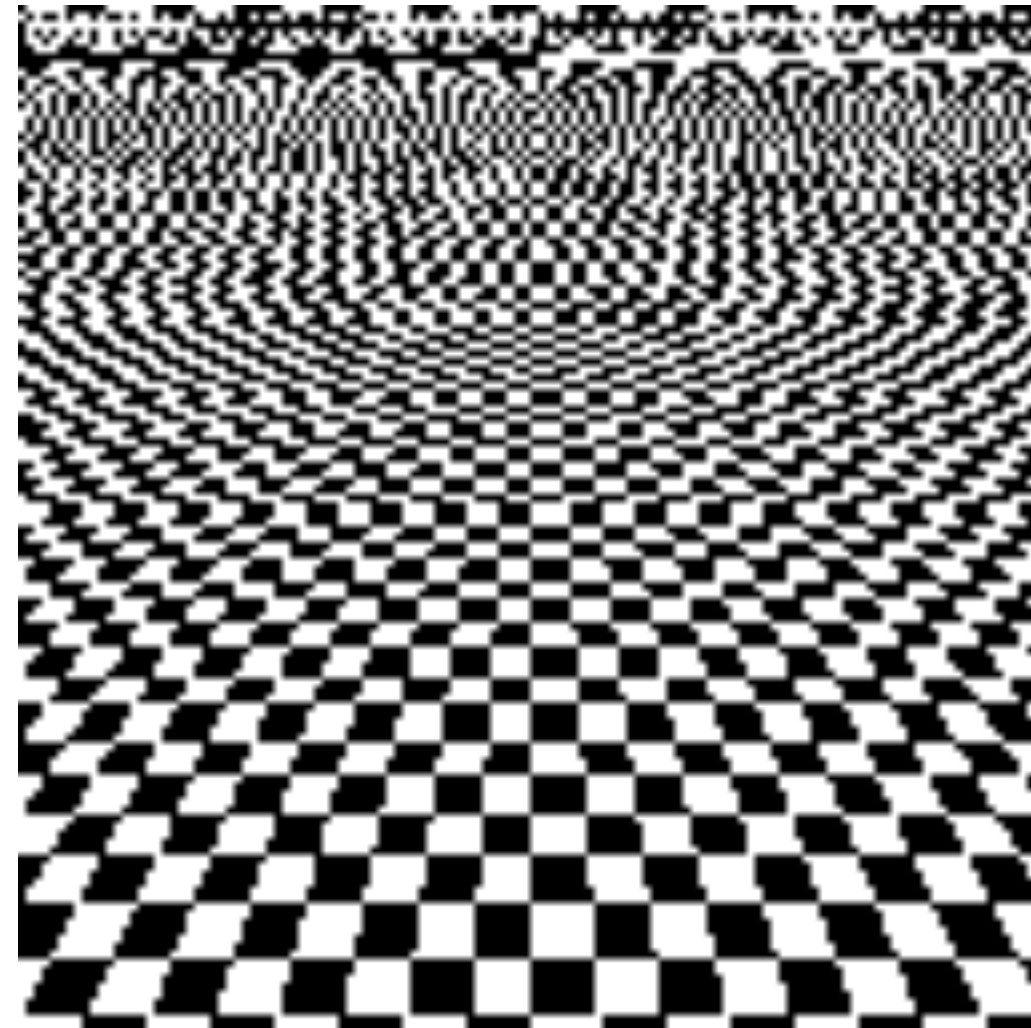


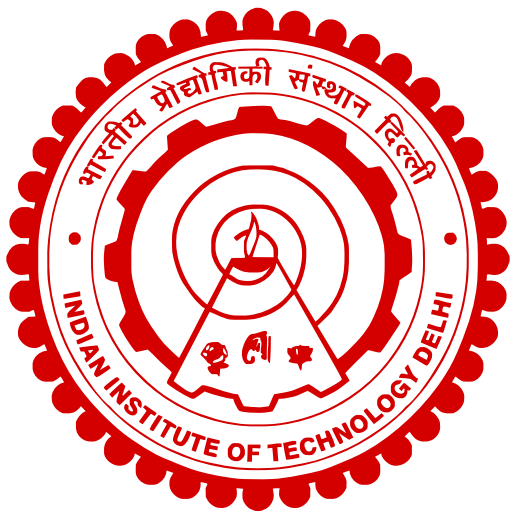
Original Signal



Box Reconstruction

Aliasing





What is Needed?

- Image to be displayed should be band-limited
 - ➔ less than 0.5 Hz (i.e., per pixel)
- Frequency band-limiting can increase spatial domain of the signal
 - ➔ We will need to generate more of the new signal to further process
- A possible solution:
 - ➔ Band-limit = Low-pass filter $\Rightarrow$ convolve *sinc*
 - ➔ Re-sample $\Rightarrow$ multiply *comb(x)*
- Do not compute the entire original signal

Anti-Alias: Eliminate High Frequency



- Low pass filter

$$f(x) * \text{sinc}(x) \leftrightarrow F(\mu) \times \text{PULSE}(\mu)$$

➡ Do not want to compute full f (nor convolve with *Sinc* really)

➡ But recall (for a large enough n , number of comb teeth per pixel):

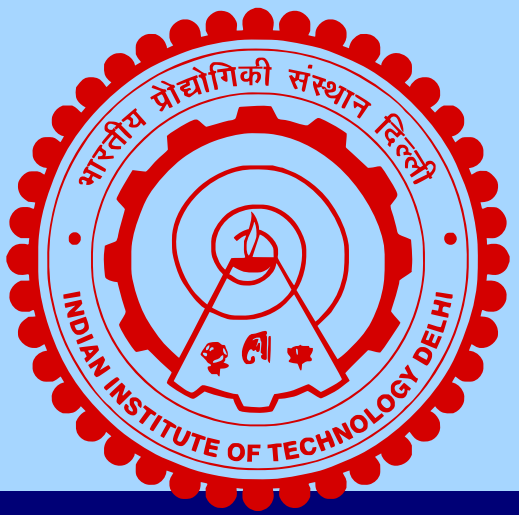
$$f(x) = f(x) \times \text{comb}_n(x) * \text{sinc}(x)$$

$$\Rightarrow \text{low pass filter: } f(x) \times \text{comb}_n(x) * \text{sinc}(x) * \text{sinc}(x)$$

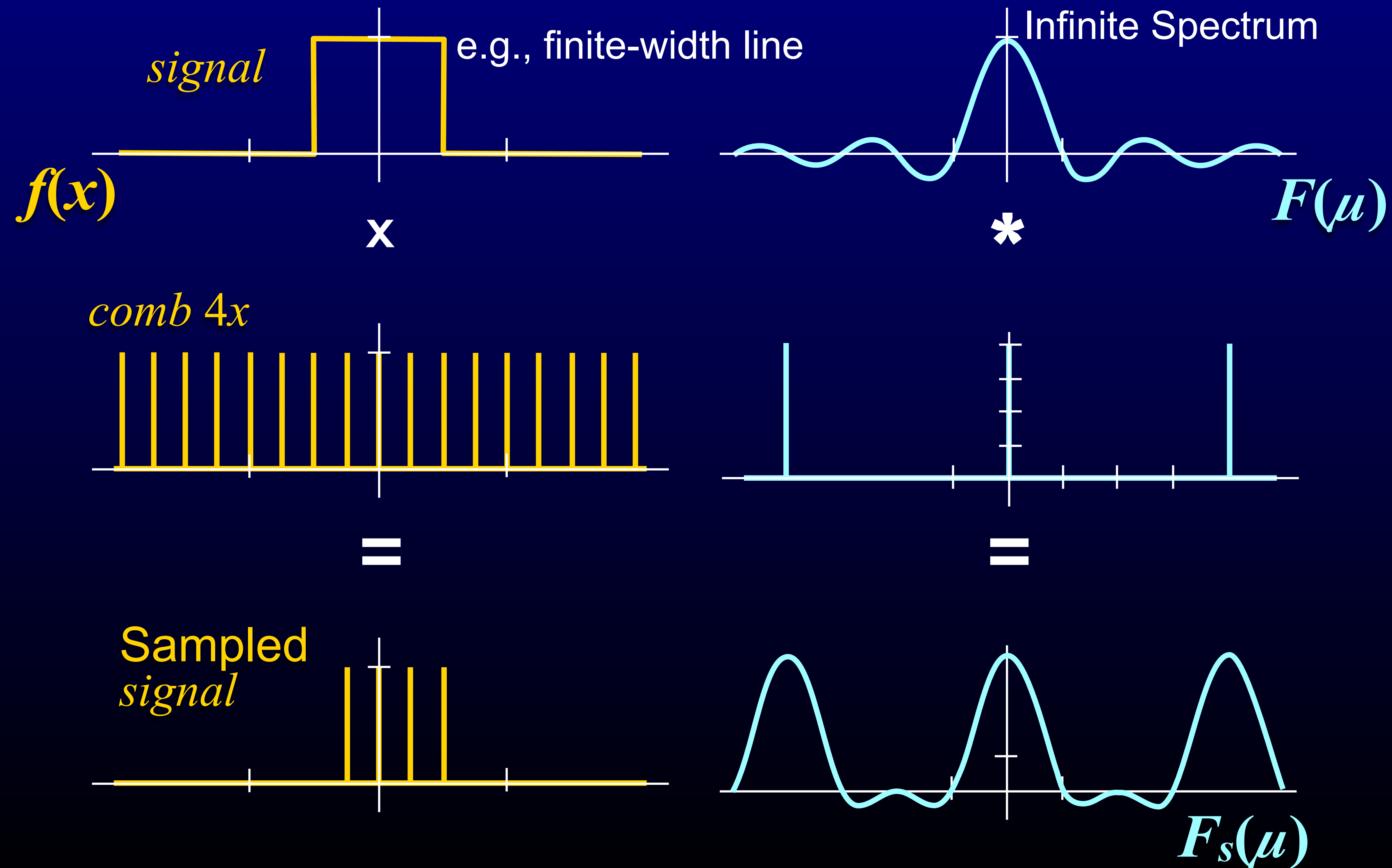
- Sample: Dense samples convolve (weighted average)
Pulse or Triangle instead?

$$[f(x) \times \text{comb}_n(x) * \text{sinc}(x)] \times \text{comb}_1(x)$$

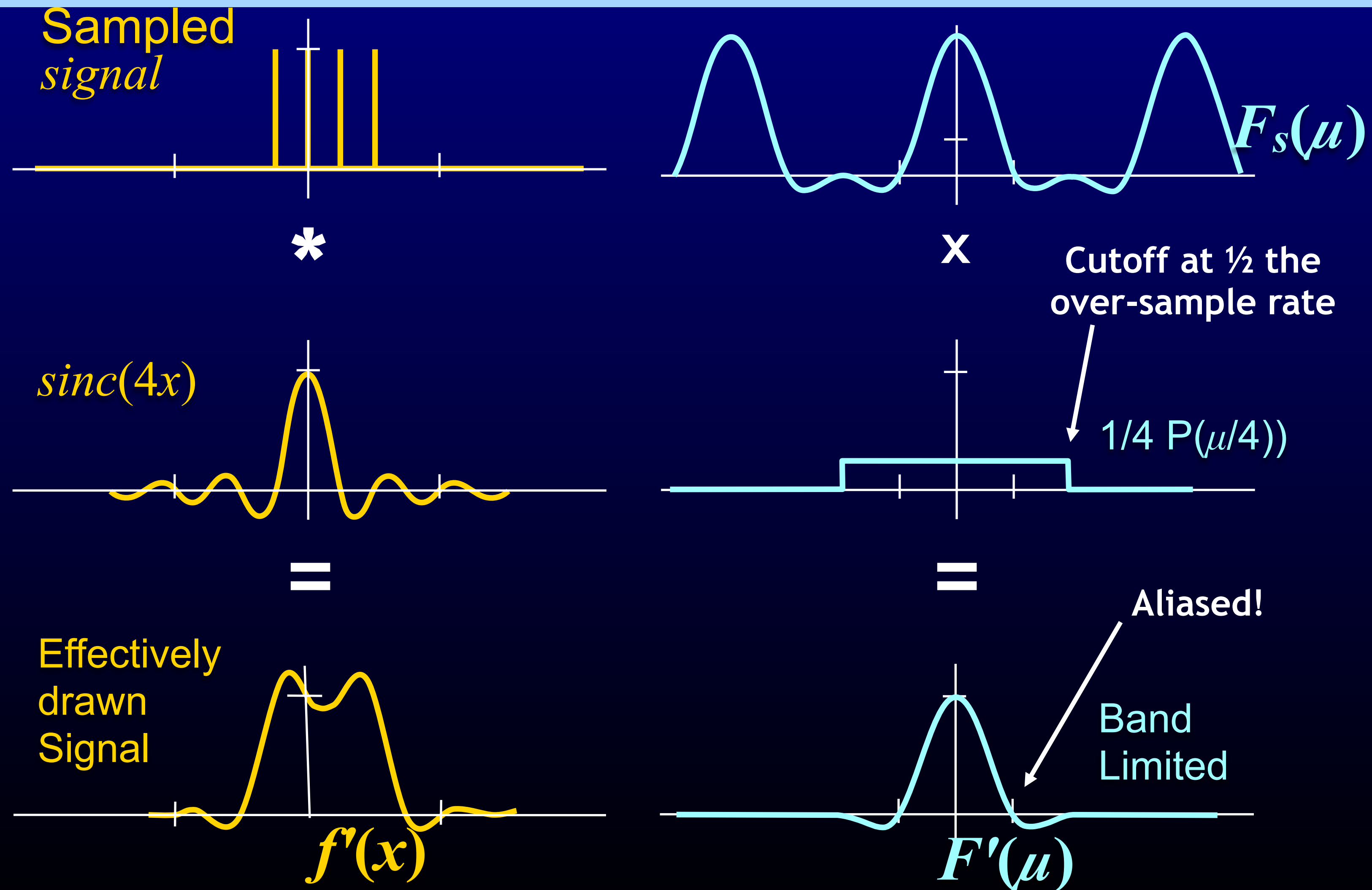
Compute at each pixel

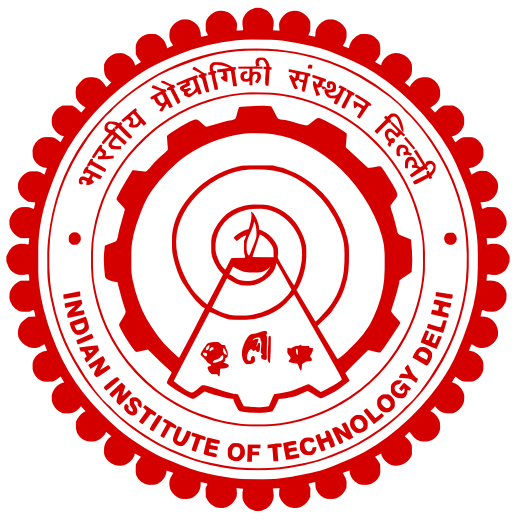


e.g., Line: Sample at 4x pixel rate



Reconstruct





Useful Filters

- Box
- Triangle
- Gaussian
- Sinc
 - ➡ Truncated sinc
- In 2D
 - ➡ Tensor Product of 1D functions



Super-Sampling

- Implementation

- ➔ Regular sample spacing simplifies attribute interpolation
- ➔ Large sample count is expensive

- Image quality

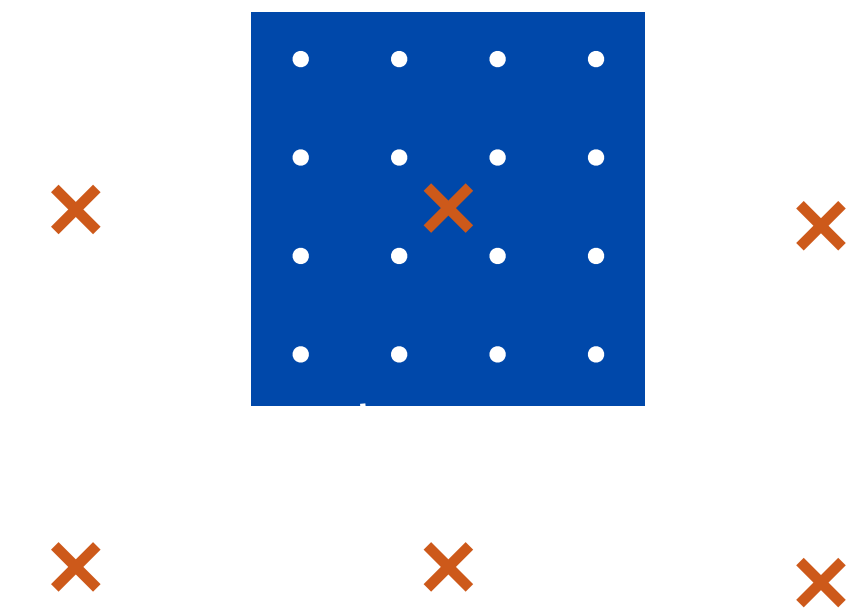
- ➔ Better with large number of samples
- ➔ Overkill for most samples

- In OpenGL pipeline

- ➔ Create a larger off-screen framebuffer
- ➔ Resample the large buffer to on-screen buffer
 - Can use texture

But see multi-sampling later

Pixel width = 1p



1p x 1p rectangle
around a single Pixel:
16x super-sampled

Pixel centers are
marked by x, ■ is
where the samples are



Regular-grid Subset Sampling

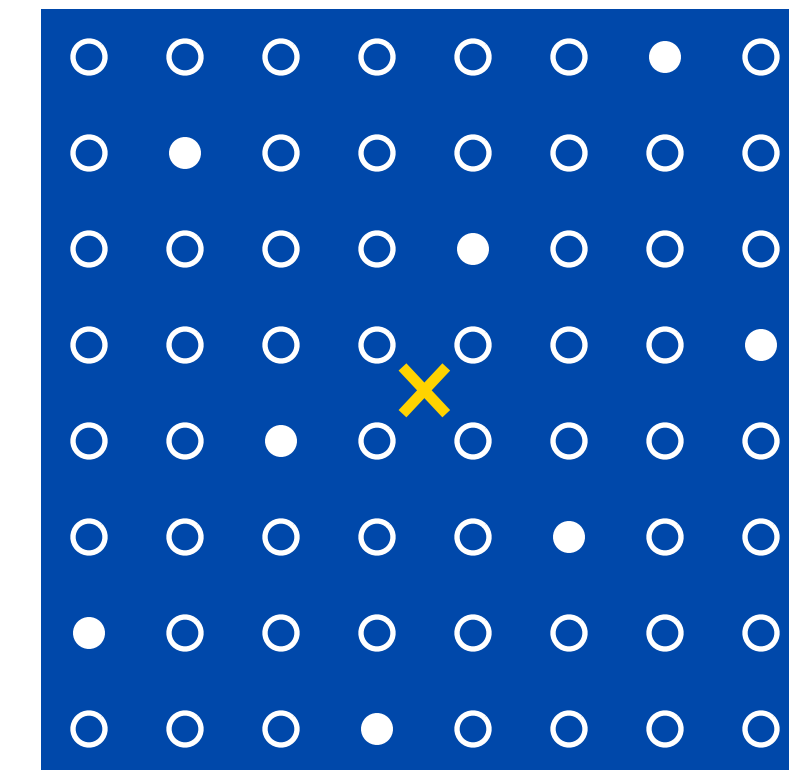
- Implementation

- ➔ Choose some samples on a higher resolution grid
- ➔ Moderate sample count is cost-effective

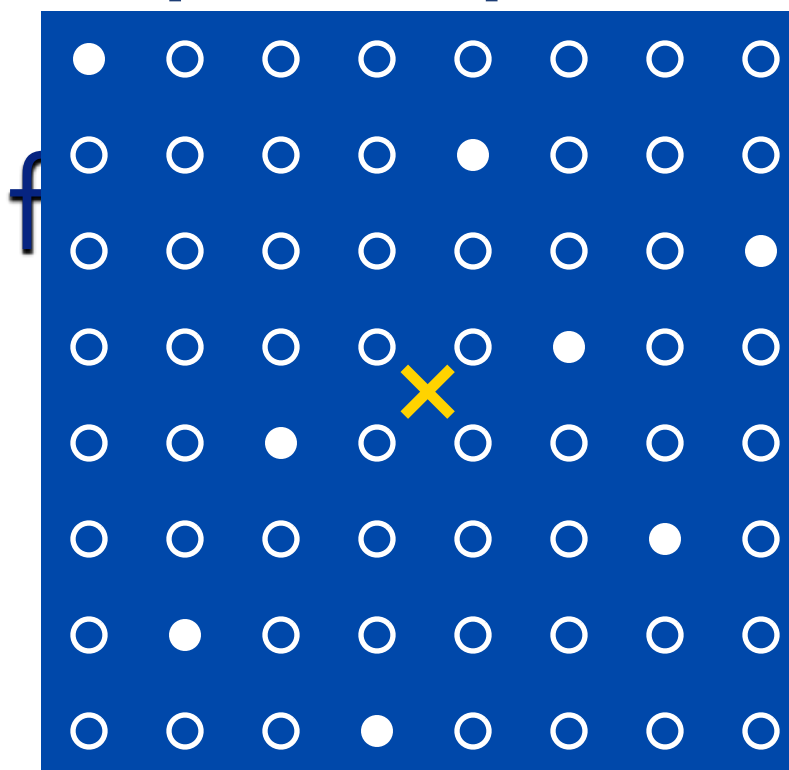
- Image quality

- ➔ Often OK with a moderate numbers of samples (8x, f

8-rooks pattern



8-queens pattern



One Pixel width



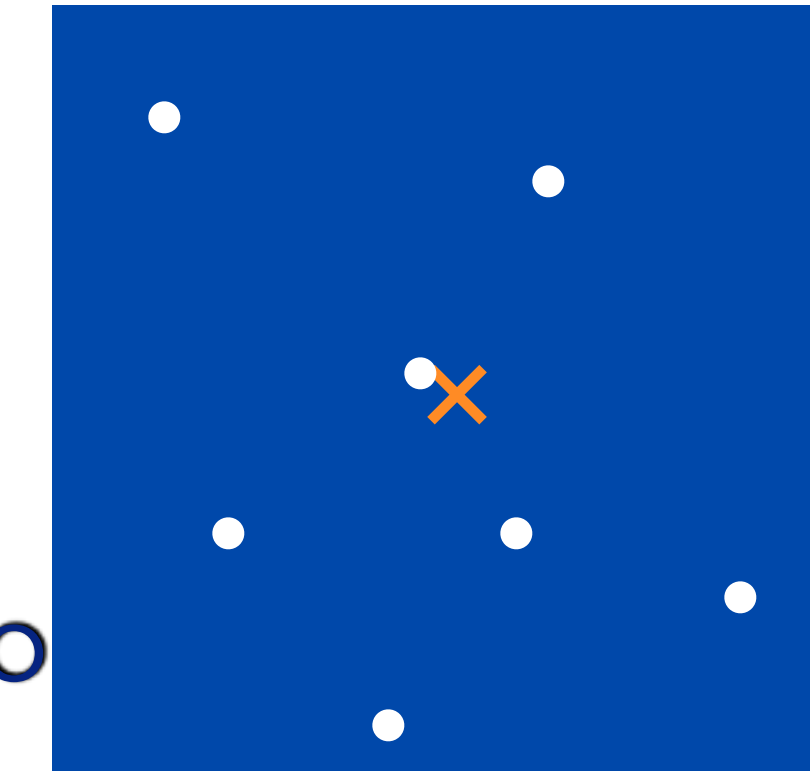
Stochastic Sampling

- Implementation

- ➔ Choose the locations in a random manner
- ➔ Irregular sample spacing can complicate attribute interpolation
- ➔ Moderate sample count is cost-effective (4-8x)

- Image quality

- ➔ Better than a pre-set pattern per



pixel



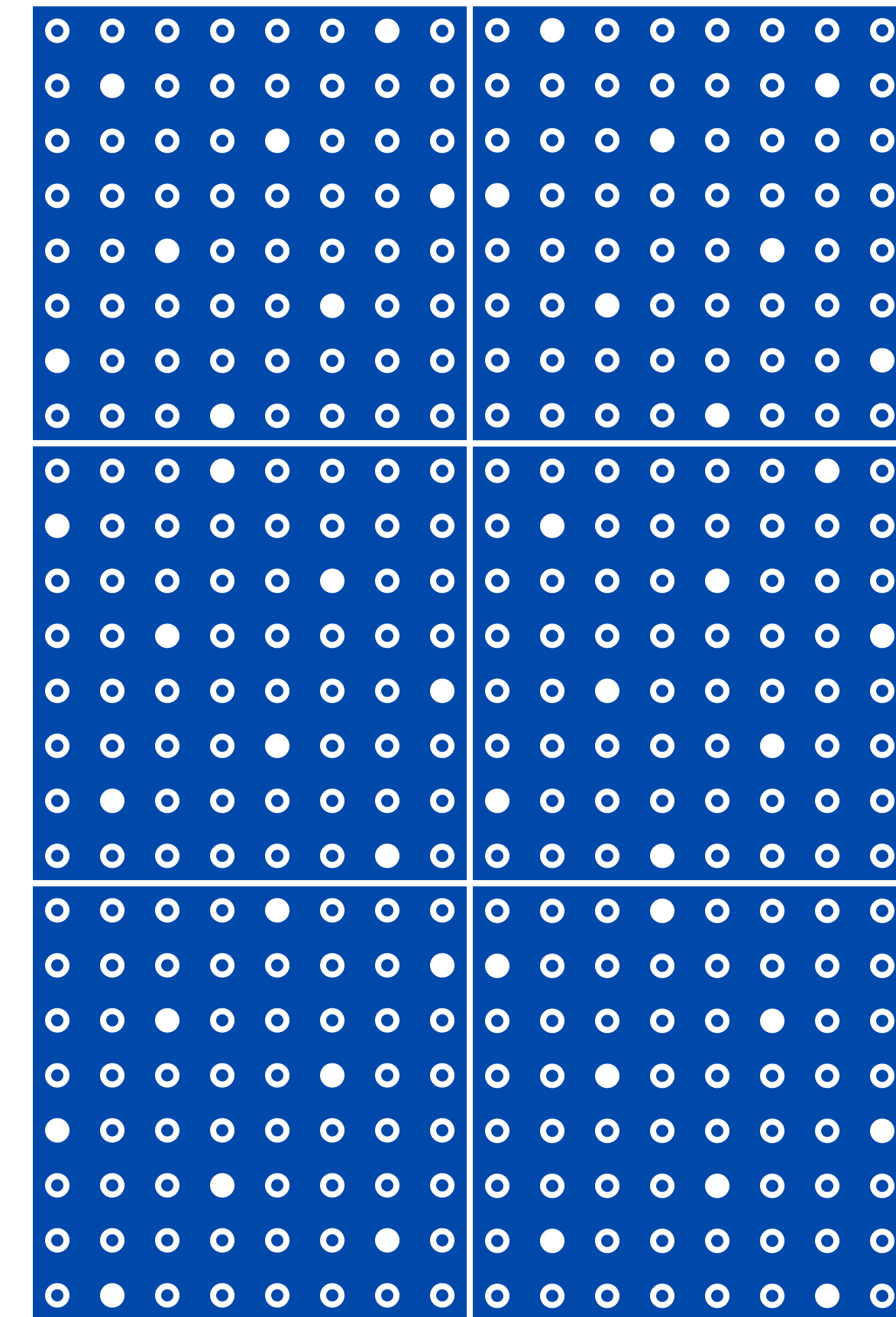
One Pixel width

random sample
pattern



Variation in Sample Patterns

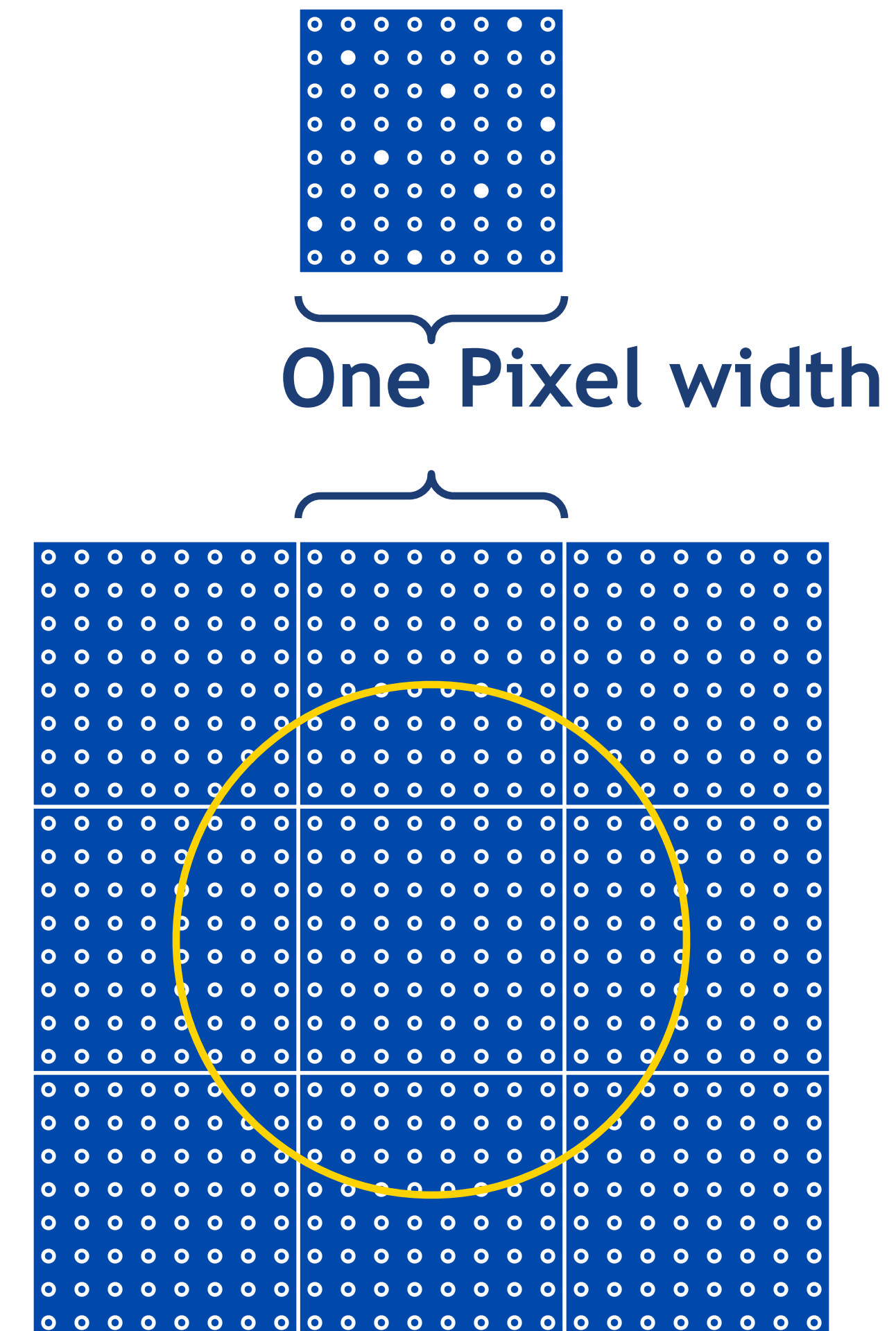
- Vary patterns
 - ➔ Temporally: frame to frame
 - ➔ Spatially: pixel to pixel
- Improves image quality
 - ➔ replaces aliasing patterns with noise
- Complicates implementation

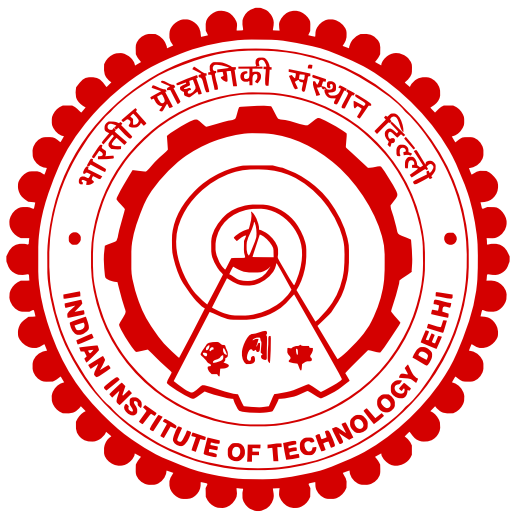




Filter Width

- Spatially wider Filter =
 - ➔ Lower pass (Narrow Frequency)
- Pixel-wide filter
 - ➔ OK for small sample counts (≤ 8)
 - ➔ Box filter is not uncommon
 - ➔ Easily parallelizable
 - ➔ Simplifies immediate filtering
- Wide foot-print filter
 - ➔ Use for large sample counts
 - ➔ Higher quality filter like *sinc*

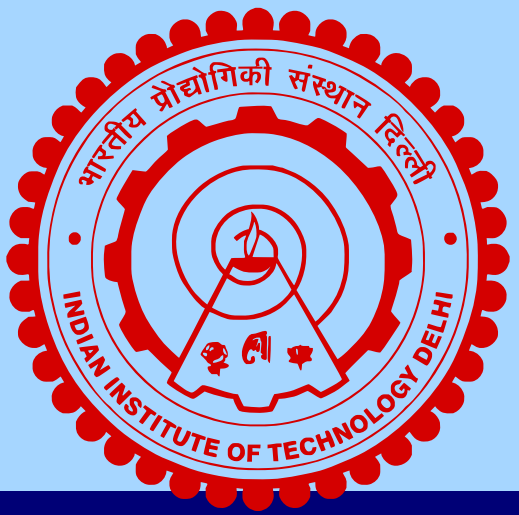




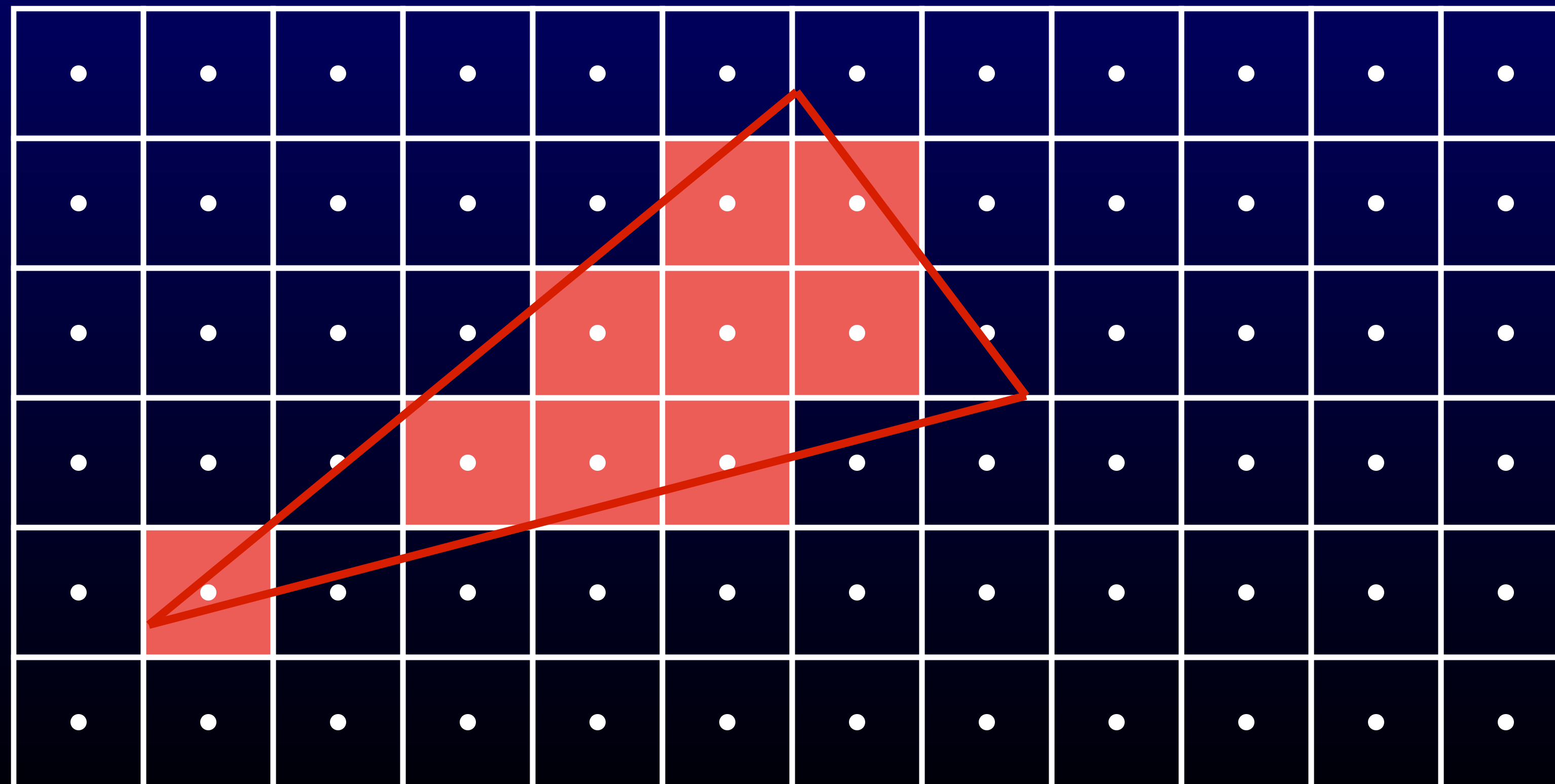
Multisampling

- Super-sampling: Shade each sample, then filter to get final pixel color
- Multisampling = approximate super-sampling
 - ➔ Super-sampling sized depth and color buffers (store per sample data)
 - ➔ Single invocation of fragment shader per pixel (not per sample)
 - ➔ Rasterizer generates a coverage mask (for each fragment)
 - "1" for those samples within the pixel that are covered by the current primitive
 - Different primitives may cover different samples of a pixel
 - ➔ Fragment shader output is written to all covered samples that pass the depth test
 - ➔ The final pixel color by filtering all sample colors

Pixel-center Fragment Selection

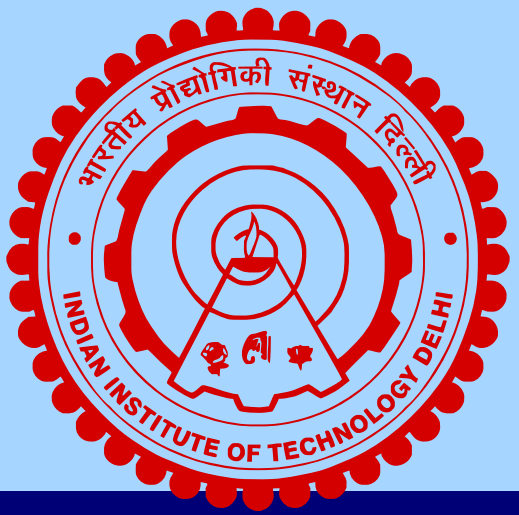


Generate fragment if pixel center is inside triangle

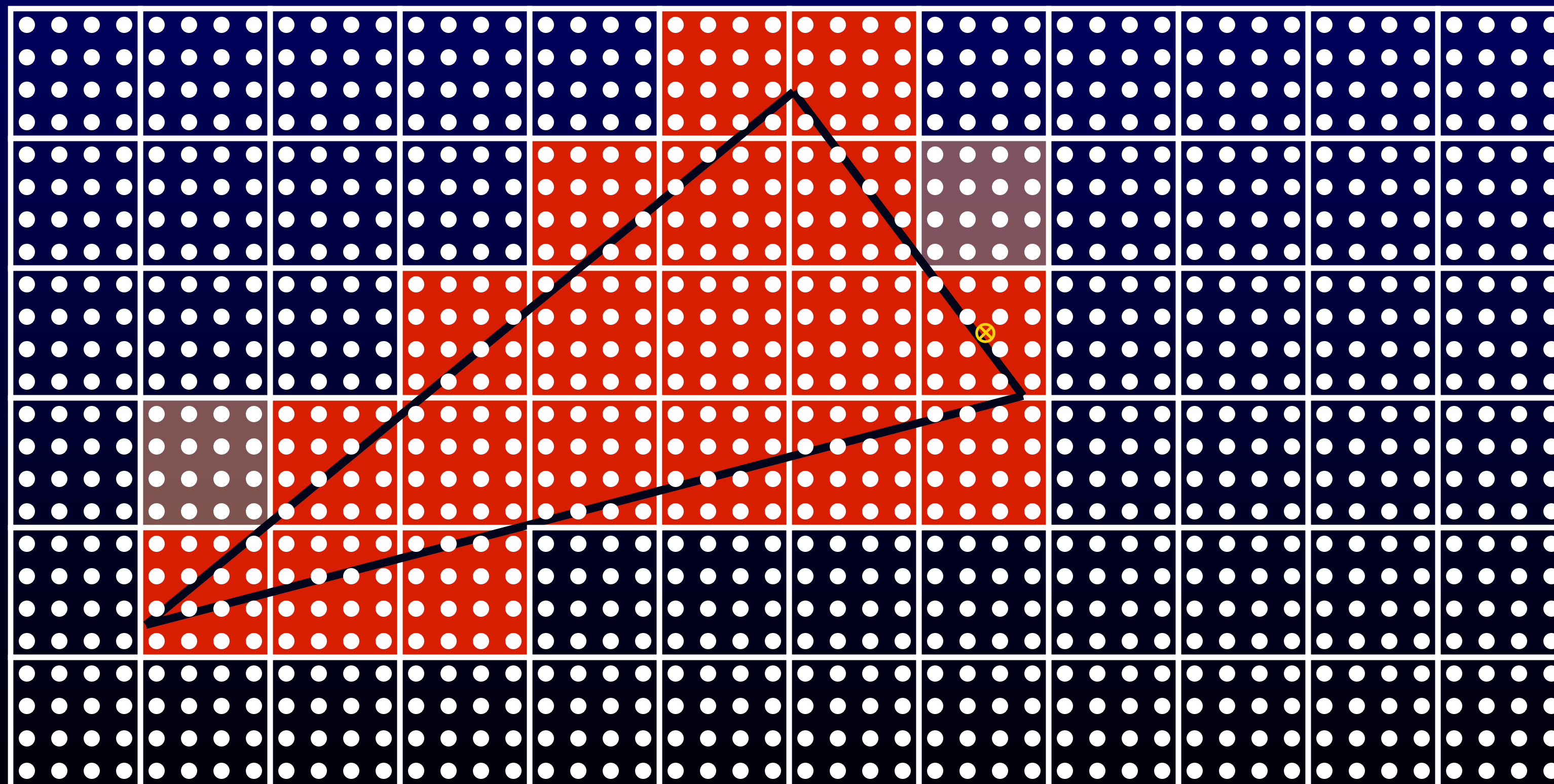


9 fragments generated

Multi-sample Fragment Selection

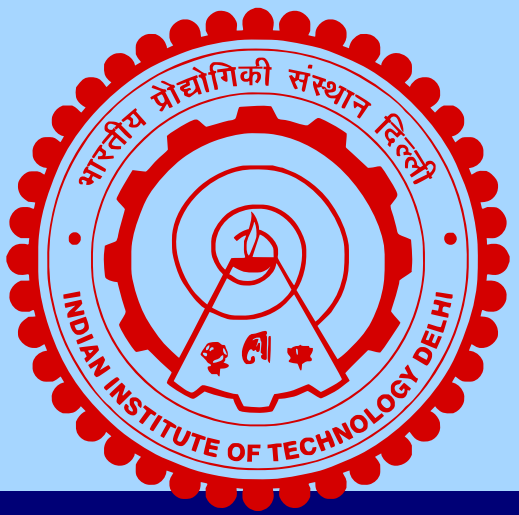


Generate fragment if any sample inside triangle

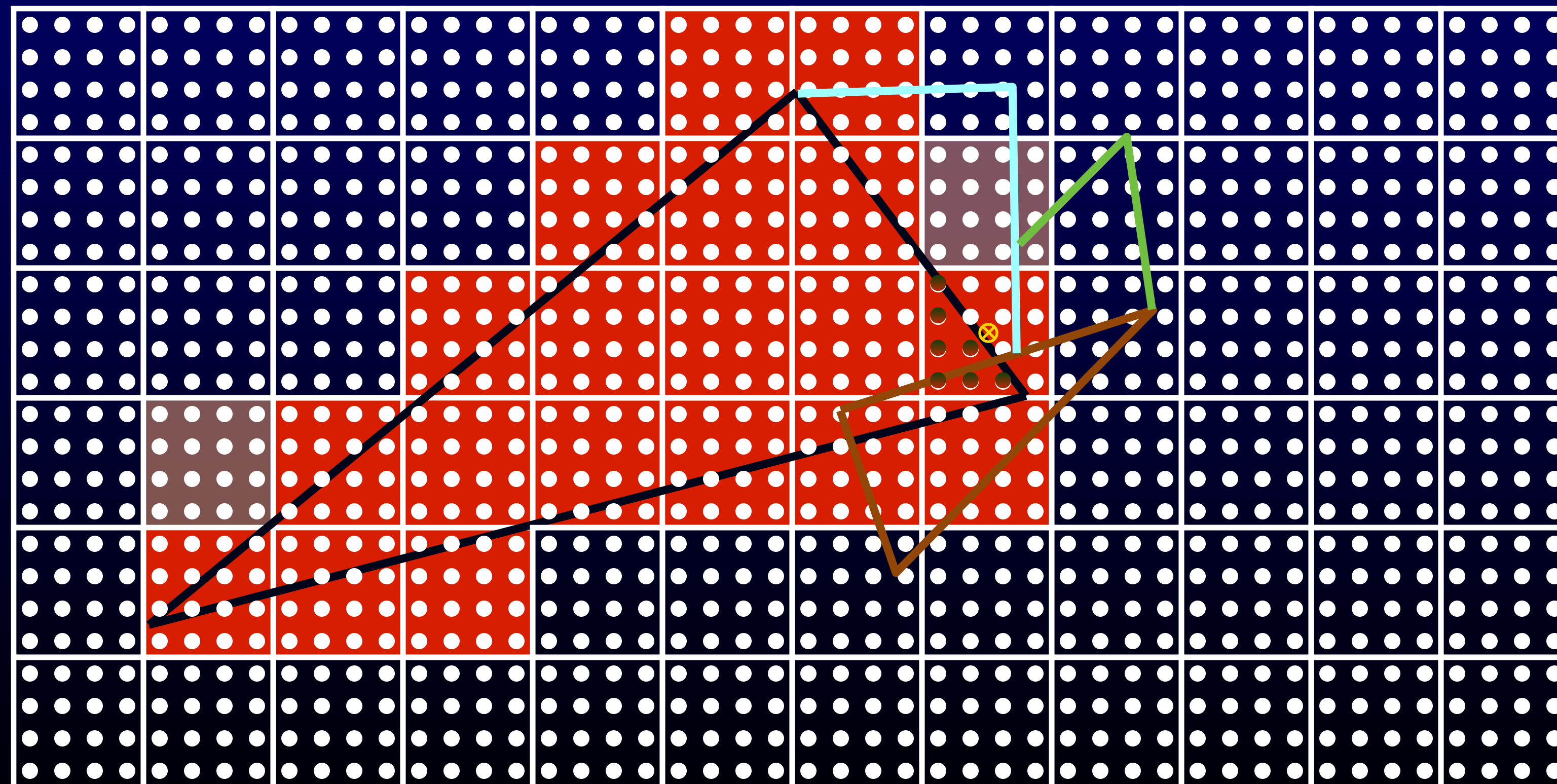


19 fragments generated

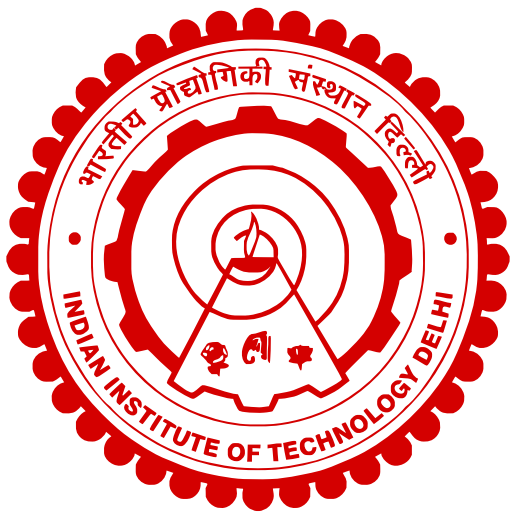
Multi-sample Fragment Selection



Generate fragment if any sample inside triangle



19 fragments generated



Multisampling

- Super-sampling: Shade each sample, then filter to get final pixel color
 - Multisampling = approximate super-sampling
 - ➡ Super-sampling sized depth and color buffers (store per sample data)
 - ➡ Single invocation of fragment shader per pixel (not per sample)
 - ➡ Rasterizer generates a coverage mask (for each fragment)
 - ▶ "1" for those samples within the pixel that are covered by the current primitive
 - ▶ Different primitives may cover different samples of a pixel
 - ➡ Fragment shader output is written to all covered samples that pass the depth test
 - ➡ The final pixel color by filtering all sample colors
- Alternative: 2nd pass blur**



Summary

- Anti-aliasing is hard
- Supersampling does not eliminate aliasing
 - ➔ It only reduces it
- Multisampling = approximate supersampling
 - ➔ Single fragment per pixel
- Handles occlusions correctly
- In OpenGL:

```
glfwWindowHint(GLFW_SAMPLES, 8);  
glEnable(GL_MULTISAMPLE); Also see: glEnable(GL_SAMPLE_SHADING);  
// render as usual, almost everything works
```

In-built GLSL variables

`gl_SampleID`

`gl_NumSamples`

`gl_SamplePosition: (x,y)`

`gl_SampleMask: Output mask to AND with rasterizer-computed coverage`