Signal Processing

COS 323

Digital "Signals"

- 1D: functions of space or time (e.g., sound)
- 2D: often functions of 2 spatial dimensions (e.g. images)
- 3D: functions of 3 spatial dimensions (CAT, MRI scans) or 2 space, 1 time (video)

Digital Signal Processing

1. Understand analogues of *filters*
2. Understand nature of *sampling*

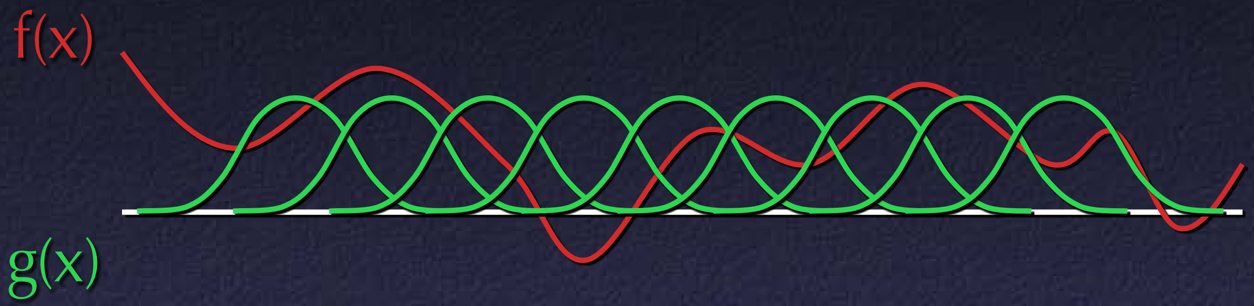
Filtering

- Consider a noisy 1D signal f(x)
- Basic operation: smooth the signal
  - Output = new function h(x)
  - Want properties: linearity, shift invariance
- Linear Shift-Invariant Filters
  - If you double input, double output
  - If you shift input, shift output

Convolution

- Output signal at each point = weighted average of local region of input signal
  - Depends on input signal, pattern of weights
  - “Filter” $g(x)$ = function of weights for linear combination
  - Basic operation = move filter to some position x, add up f times g

Convolution



$$f(x) * g(x) = \int_{-\infty}^{\infty} f(t)\, g(x-t)\, dt$$

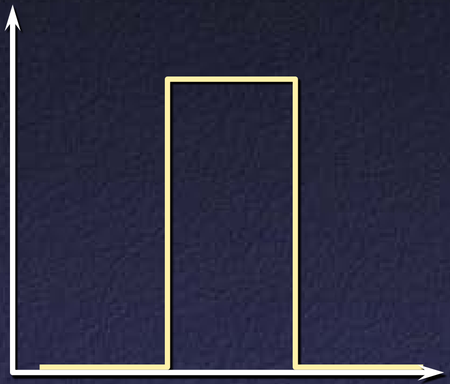
Convolution

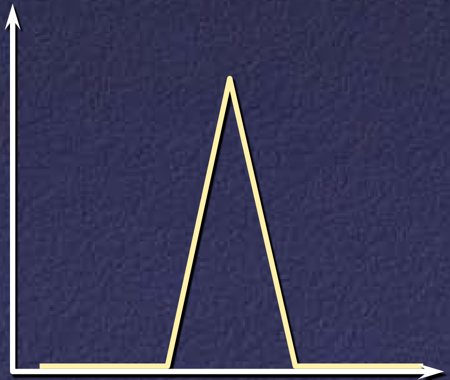
- f is called “signal” and g is “filter” or “kernel”, but the operation is symmetric
- Usually desirable to leave a constant signal unchanged: choose g such that

$$\int_{-\infty}^{\infty} g(t)\,dt = 1$$

Filter Choices

- Simple filters: box, triangle

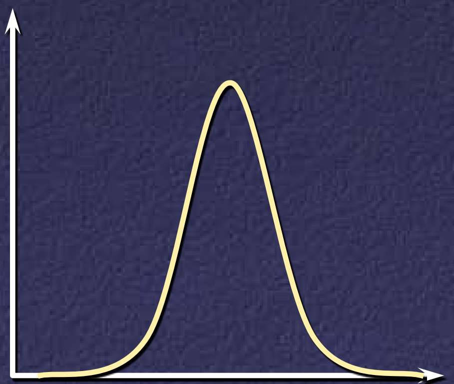




Gaussian Filter

- Very commonly used filter

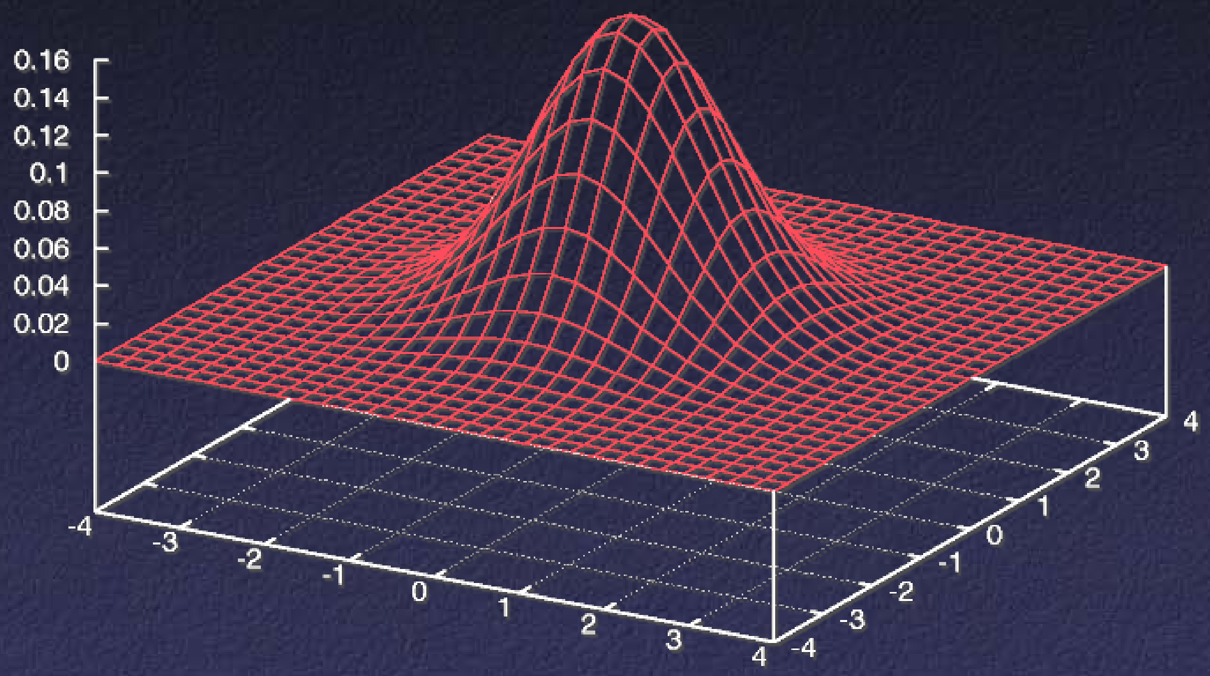
$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$



Gaussian Filters

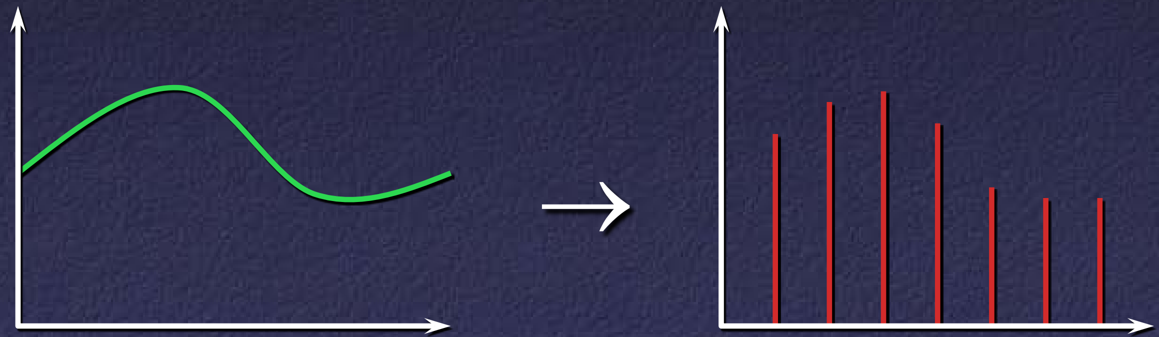
- Gaussians are used because:
  - Smooth (infinitely differentiable)
  - Decay to zero rapidly
  - Simple analytic formula
  - Separable: multidimensional Gaussian = product of Gaussians in each dimension
  - Convolution of 2 Gaussians = Gaussian
  - Limit of applying multiple filters (*) is Gaussian (Central limit theorem)

2D Gaussian Filter



Sampled Signals

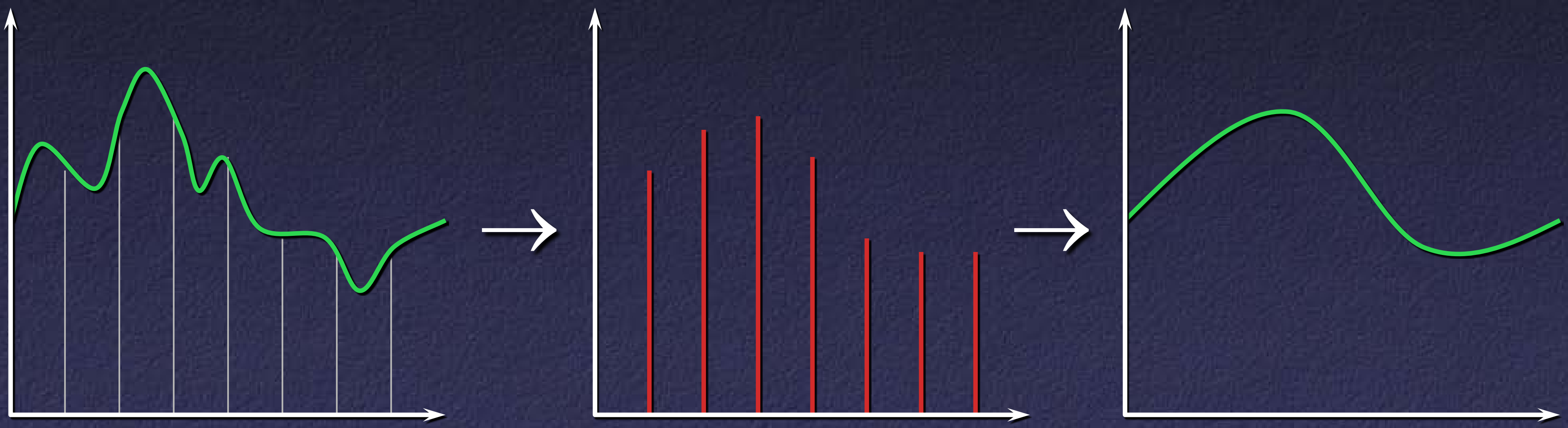
- Can't store continuous signal: instead store "samples"
  - Usually evenly sampled:
    $f_0 = f(x_0),\ f_1 = f(x_0 + \Delta x),\ f_2 = f(x_0 + 2\Delta x),\ f_3 = f(x_0 + 3\Delta x),\ \ldots$



- Instantaneous measurements of continuous signal
  - This can lead to problems

Aliasing

- Reconstructed signal might be very different from original: “aliasing”



- Solution: smooth the signal *before* sampling

Discrete Convolution

- Integral becomes sum over samples

$$f * g = \sum_i f_i \, g_{x-i}$$

- Normalization condition is

$$\sum_i g_i = 1$$

Computing Discrete Convolutions

$$f * g = \sum_i f_i \, g_{x-i}$$

- What happens near edges of signal?
  – Ignore (Output is smaller than input)
  – Pad with zeros (edges get dark)
  – Replicate edge samples
  – Wrap around
  – Reflect
  – Change filter

Computing Discrete Convolutions

$$f * g = \sum_i f_i \, g_{x-i}$$

- If f has *n* samples and g has *m* nonzero samples, straightforward computation takes time

  $O(nm)$

- OK for small filter kernels, bad for large ones

Example: Smoothing

Original image

Smoothed with
2D Gaussian kernel

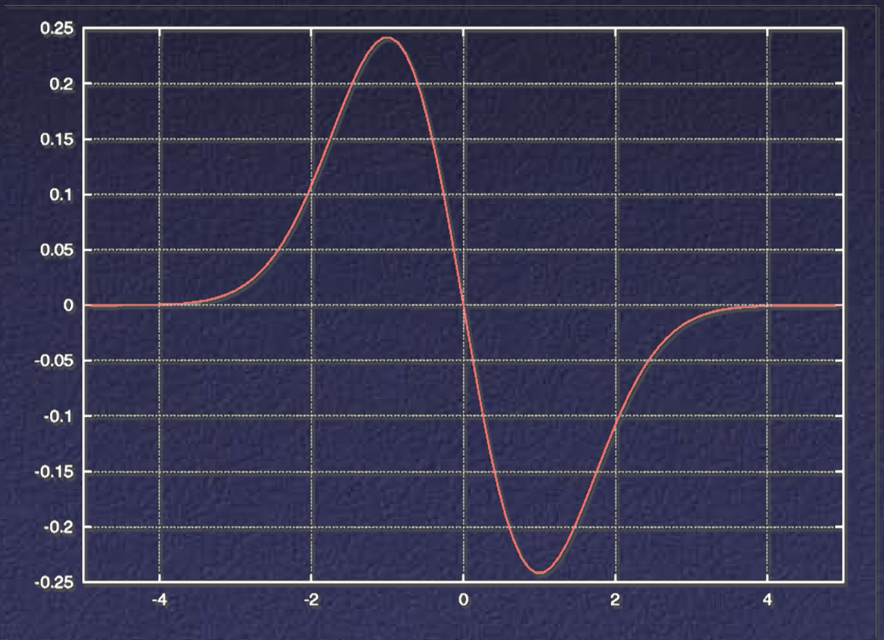
Example: Smoothed Derivative

- Derivative of noisy signal = more noisy
- Solution: smooth with a Gaussian *before* taking derivative
- Differentiation and convolution both linear operators: they "commute"

$$\frac{d}{dx}\left(f * g\right) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

Example: Smoothed Derivative

- Result: good way of finding derivative = convolution with derivative of Gaussian



Smoothed Derivative in 2D

- What is "derivative" in 2D? Gradient:

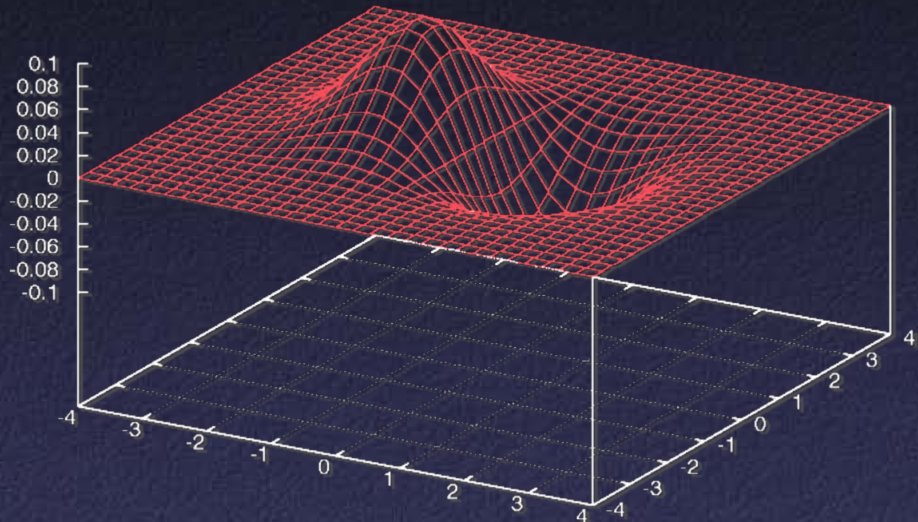
$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

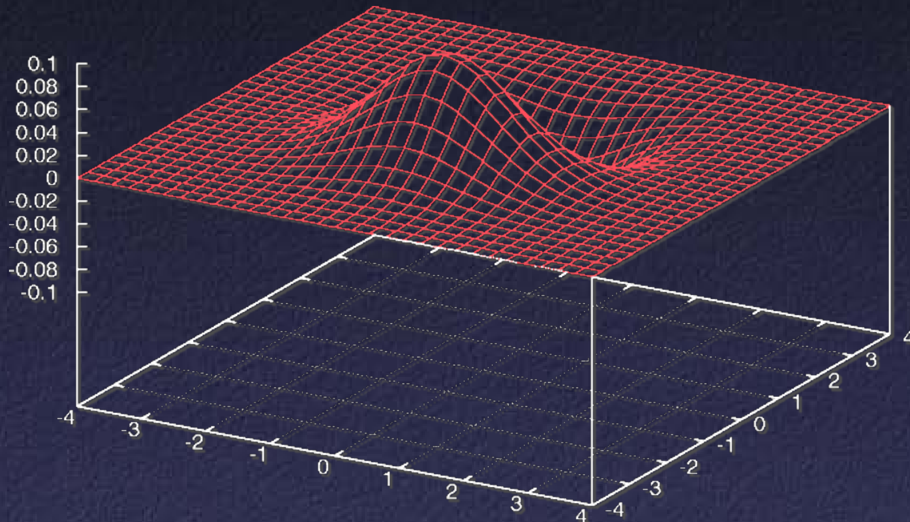
- Gaussian is separable! $G_2(x,y) = G_1(x)G_1(y)$

- Combine smoothing, differentiation:

$$\nabla\big(f(x,y) * G_2(x,y)\big) = \begin{bmatrix} f(x,y) * \big(G_1'(x)G_1(y)\big) \\ f(x,y) * \big(G_1(x)G_1'(y)\big) \end{bmatrix} = \begin{bmatrix} f(x,y) * G_1'(x) * G_1(y) \\ f(x,y) * G_1(x) * G_1'(y) \end{bmatrix}$$

Smoothed Derivative in 2D





$$\nabla\left(f(x,y) * G_2(x,y)\right) = \begin{bmatrix} f(x,y) * \left(G_1'(x) G_1(y)\right) \\ f(x,y) * \left(G_1(x) G_1'(y)\right) \end{bmatrix} = \begin{bmatrix} f(x,y) * G_1'(x) * G_1(y) \\ f(x,y) * G_1(x) * G_1'(y) \end{bmatrix}$$

Smoothed Derivative in 2D

Original Image

Smoothed Gradient Magnitude

Canny Edge Detector

- Smooth
- Find derivative
- Find maxima
- Threshold

Canny Edge Detector

Original Image

Edges

Fourier Transform

- Transform applied to function to analyze its "frequency" content
- Several versions
  - Fourier series:
    - input = continuous, bounded; output = discrete, unbounded
  - Fourier transform:
    - input = continuous, unbounded; output = continuous, unbounded
  - Discrete Fourier transform (DFT):
    - input = discrete, bounded; output = discrete, bounded

Fourier Series

- Periodic function f(x) defined over $[-\pi \, .. \, \pi]$

$$f(x) = \tfrac{1}{2} a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + b_n \sin(nx)$$

where

$$a_n = \tfrac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx) \, dx$$

$$b_n = \tfrac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx) \, dx$$

Fourier Series

- This works because sines, cosines are orthonormal over $[-\pi\,..\,\pi]$:

$$\frac{1}{\pi}\int_{-\pi}^{\pi}\cos(mx)\cos(nx)\,dx=\delta_{mn}$$

$$\frac{1}{\pi}\int_{-\pi}^{\pi}\sin(mx)\sin(nx)\,dx=\delta_{mn}$$

$$\frac{1}{\pi}\int_{-\pi}^{\pi}\sin(mx)\cos(nx)\,dx=0$$

- Kronecker delta:

$$\delta_{mn}=\begin{cases}1 \text{ if } m=n\\ 0 \text{ otherwise}\end{cases}$$

Fourier Transform

- Continuous Fourier transform:

$$\mathrm{F}(k) = \mathcal{F}\big(f(x)\big) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k x} dx$$

- Discrete Fourier transform:

$$\mathrm{F}_{\mathrm{k}} = \sum_{x=0}^{n-1} f_x e^{2\pi i \frac{k}{n} x}$$

- F is a function of frequency – describes how much of each frequency *f* contains
- Fourier transform is invertible

Fourier Transform and Convolution

- Fourier transform turns convolution into multiplication:

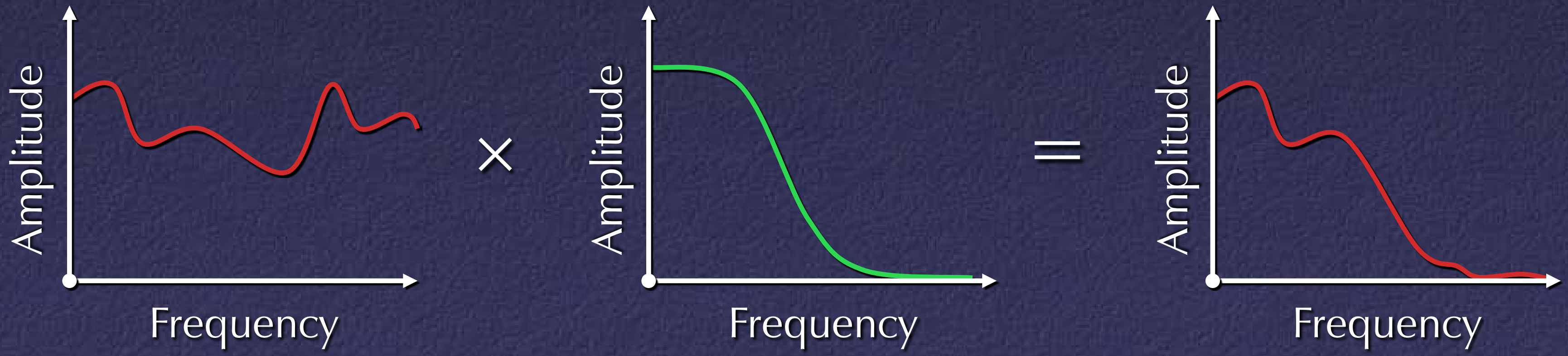
$$\mathcal{F}\big(f(x) * g(x)\big) = \mathcal{F}\big(f(x)\big)\,\mathcal{F}\big(g(x)\big)$$

(and vice versa):

$$\mathcal{F}\big(f(x)\,g(x)\big) = \mathcal{F}\big(f(x)\big) * \mathcal{F}\big(g(x)\big)$$

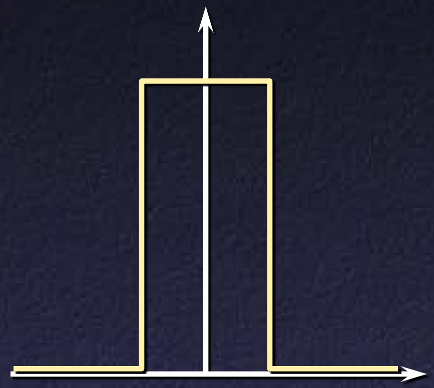
Fourier Transform and Convolution

- Useful application #1: Use frequency space to understand effects of filters
  - Example: Fourier transform of a Gaussian is a Gaussian
  - Thus: attenuates high frequencies

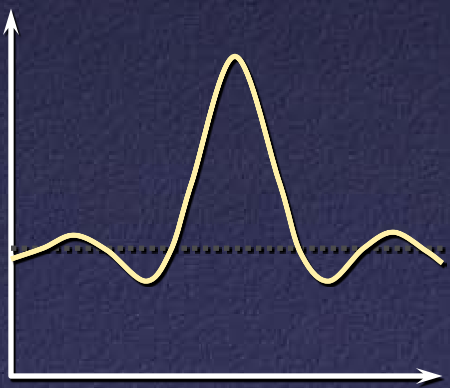


Fourier Transform and Convolution

- Box function?



- In frequency space: sinc function
  - $\text{sinc}(x) = \sin(x) / x$
  - Not as good at attenuating high frequencies



Fourier Transform and Convolution

- Fourier transform of derivative:

$$\mathcal{F}\left(\frac{d}{dx} f(x)\right) = 2\pi i k \, \mathcal{F}\big(f(x)\big)$$

- Blows up for high frequencies!
  – After Gaussian smoothing, doesn't blow up

Fourier Transform and Convolution

- Useful application #2: Efficient computation
  - Fast Fourier Transform (FFT) takes time

    $$O(n \log n)$$

  - Thus, convolution can be performed in time

    $$O(n \log n + m \log m)$$

  - Greatest efficiency gains for large filters