Signal Processing

COS 323

Digital "Signals"

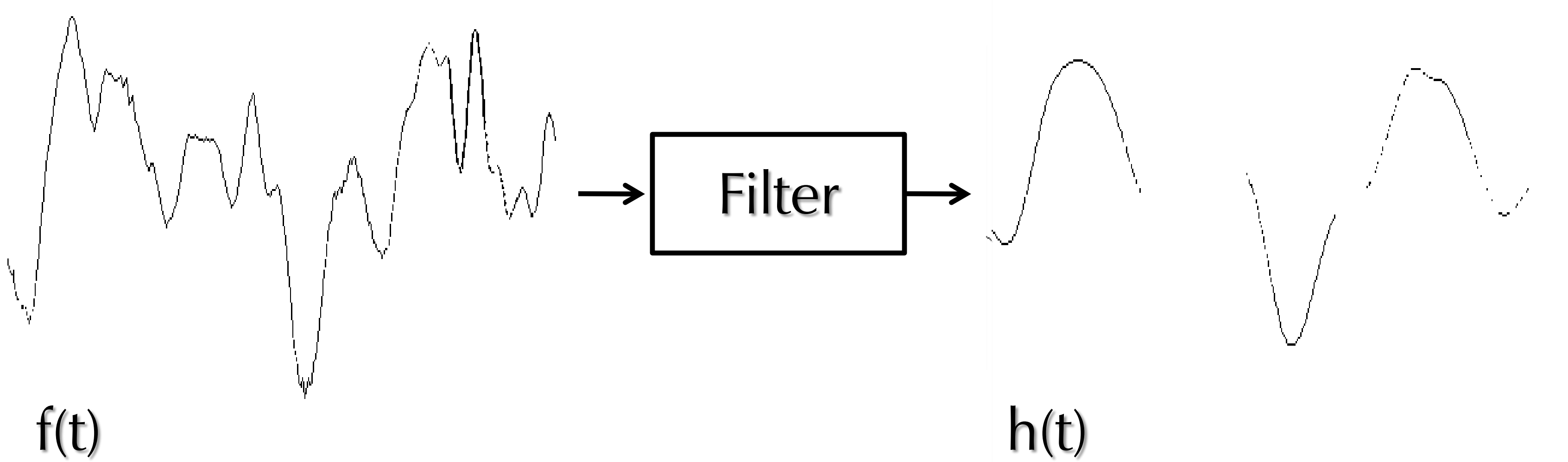
- 1D: functions of space or time (e.g., sound)
- 2D: often functions of 2 spatial dimensions (e.g. images)
- 3D: functions of 3 spatial dimensions (CAT, MRI scans) or 2 space, 1 time (video)

Digital Signal Processing

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

Filtering

- Consider a noisy 1D signal f(t)
- One basic operation: smooth the signal
  - Output = new function h(t)

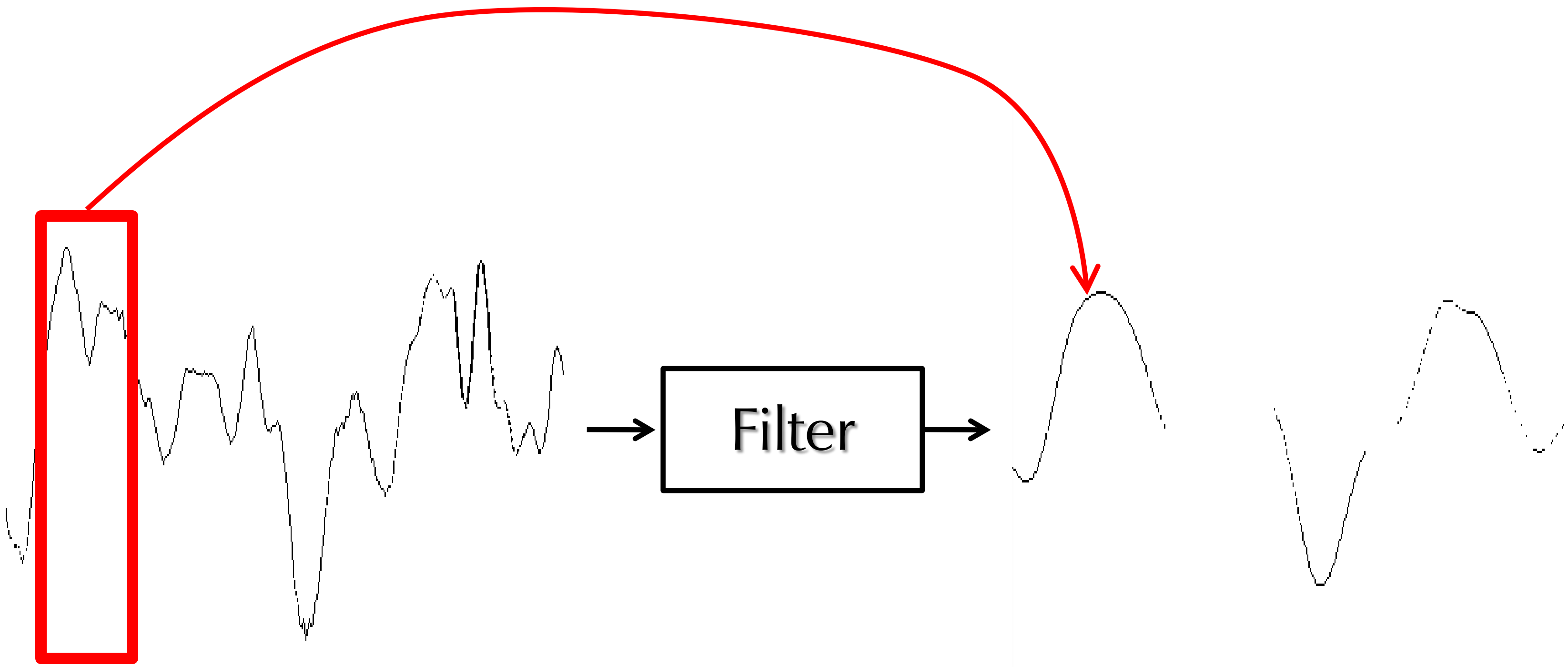


Filtering

- Consider a noisy 1D signal f(t)
- One basic operation: smooth the signal
  - Output = new function h(t)
- Linear Shift-Invariant Filters
  - If you double input, double output
  - If you shift input by Δt, shift output by Δt

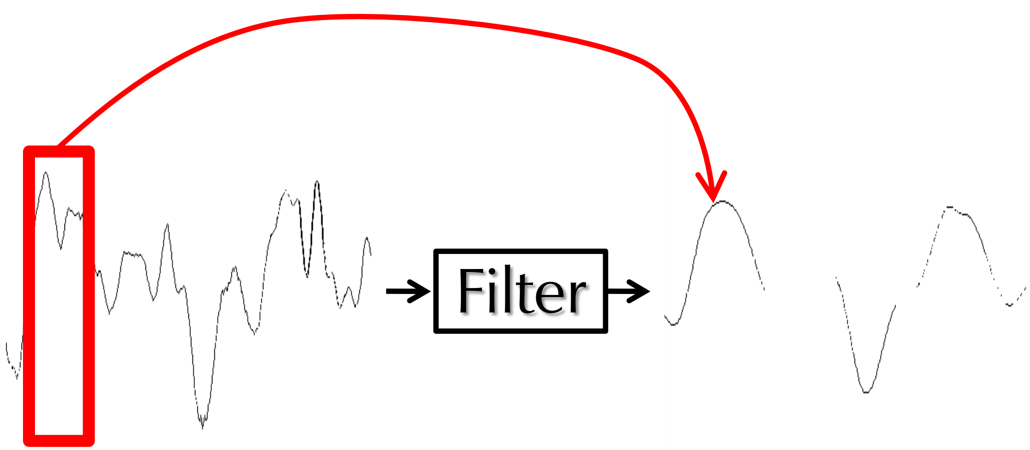
Simplest smoothing filter

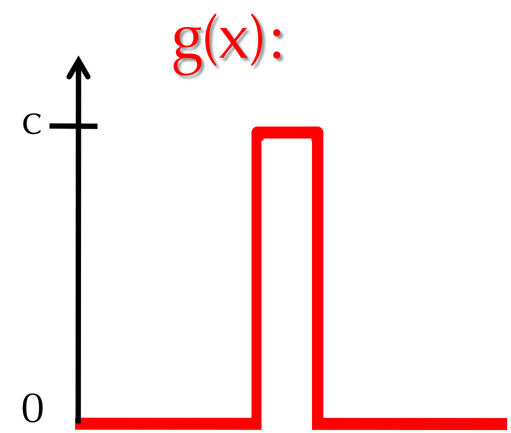
Take average of nearby points:



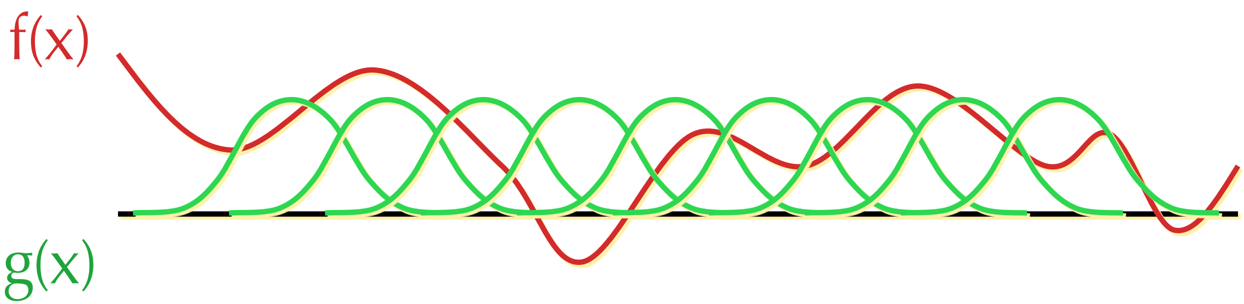
Convolution

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





Convolution



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

Try for yourself: http://jhu.edu/~signals/convolve/

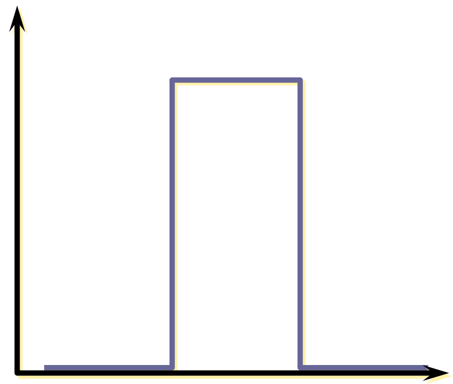
Convolution

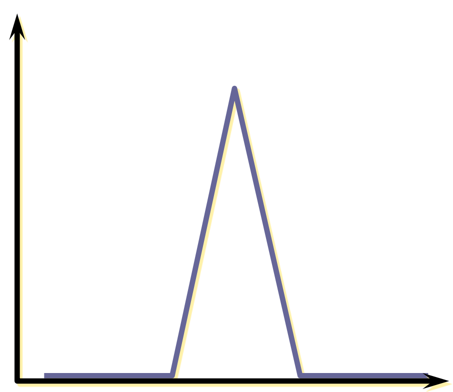
- f is called “signal” and g is “filter” or “kernel”, but the operation is symmetric *(for real functions)
- But: usually desirable to leave a constant signal unchanged: choose g such that

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

Filter Choices

- Simple filters: box, triangle

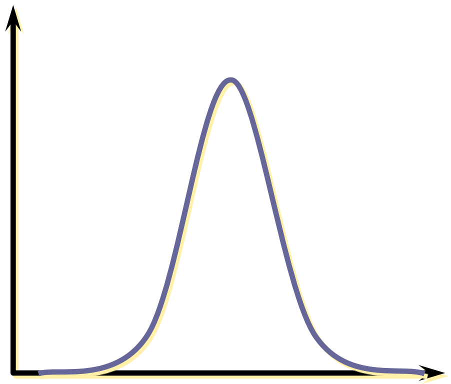




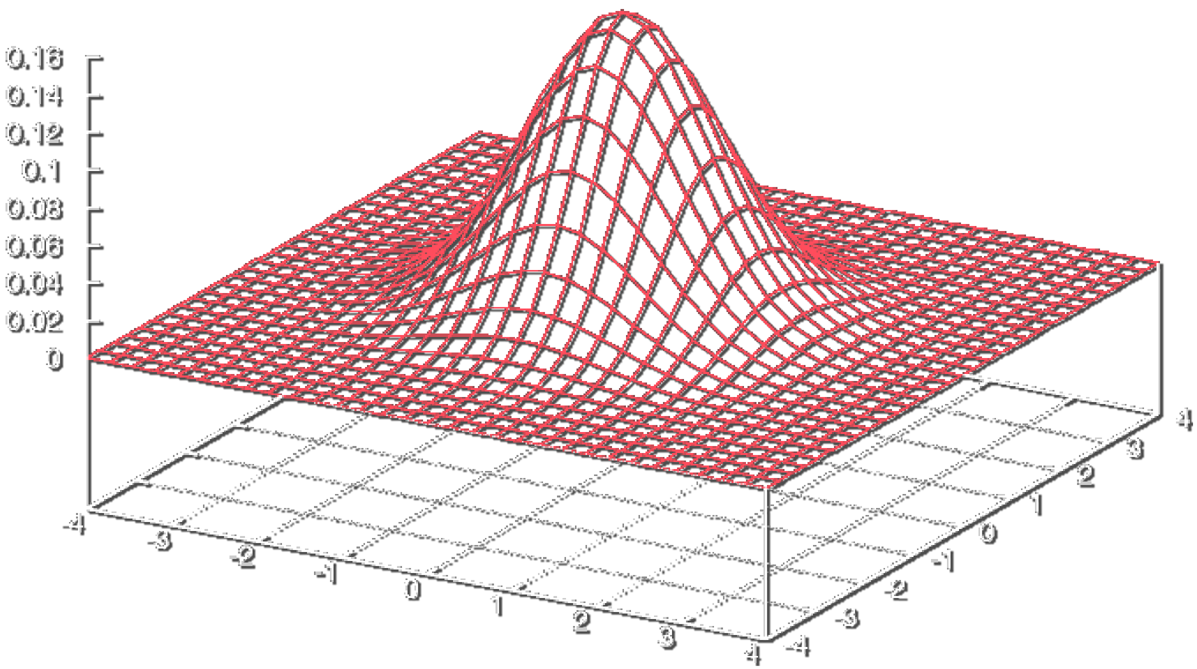
Gaussian Filter

- Commonly used filter

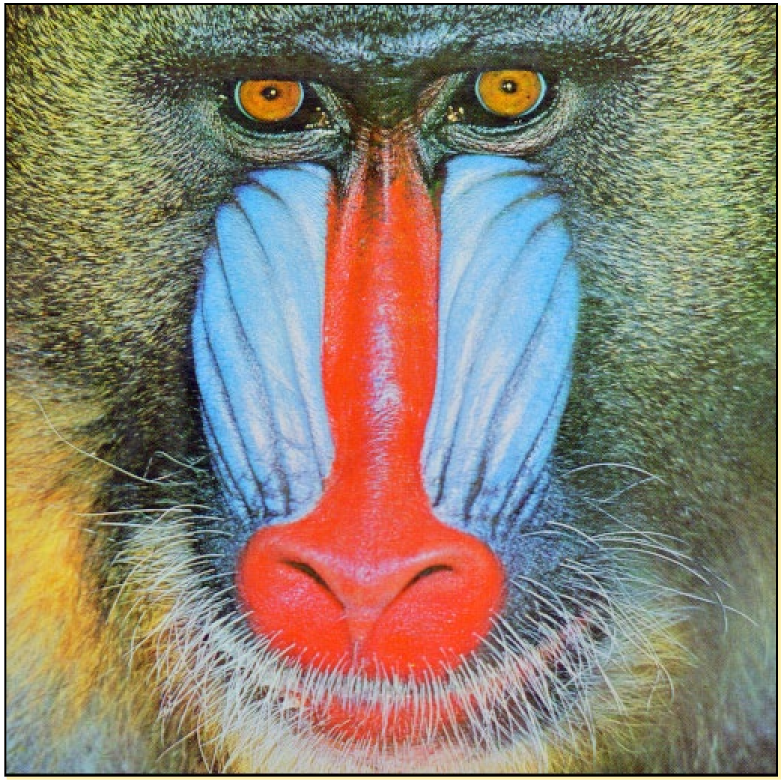
$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$



2D Gaussian Filter



Example: Smoothing



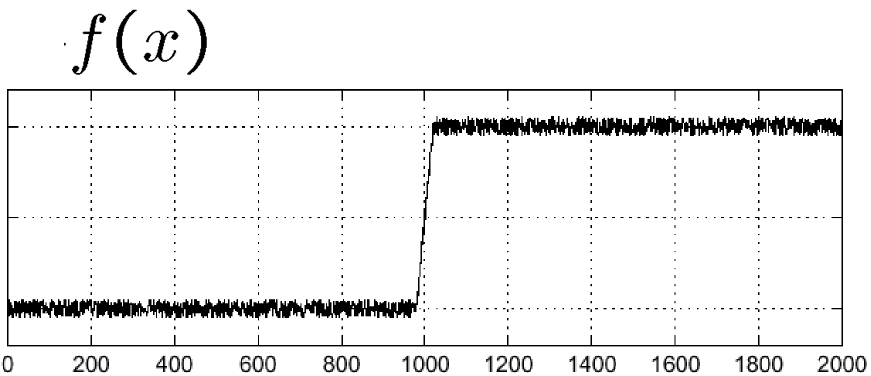
Original image

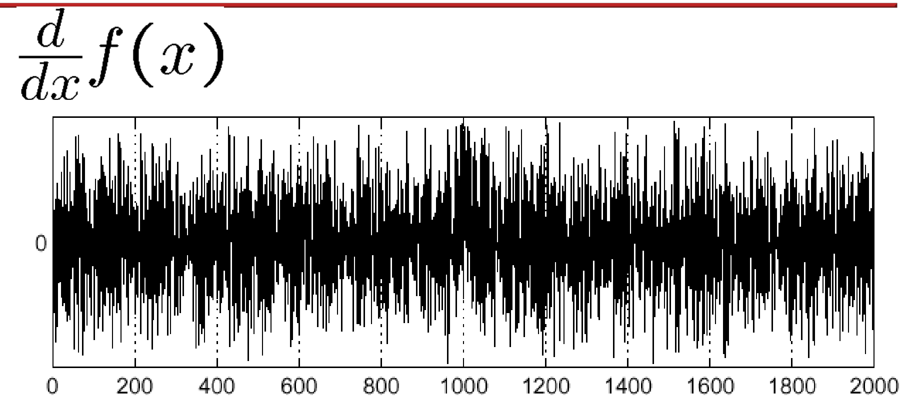
Smoothed with
2D Gaussian kernel

Example: Edge Detection

Consider magnitude of gradient (1st derivative)

Smoothed Derivative



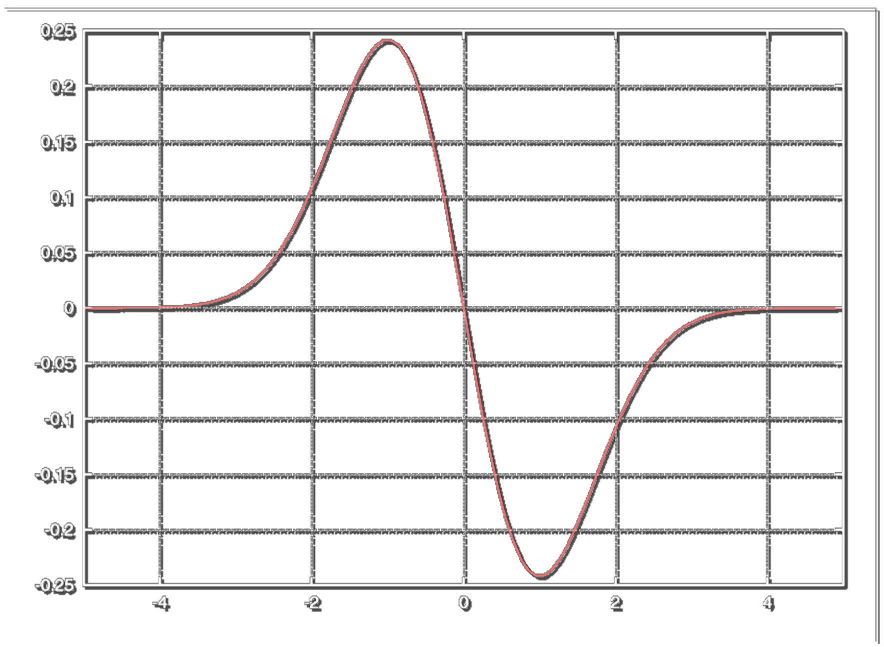


- Derivative of noisy signal = more noisy
- Solution: smooth with a Gaussian *before* taking derivative
- Differentiation and convolution both linear operators: they "commute"

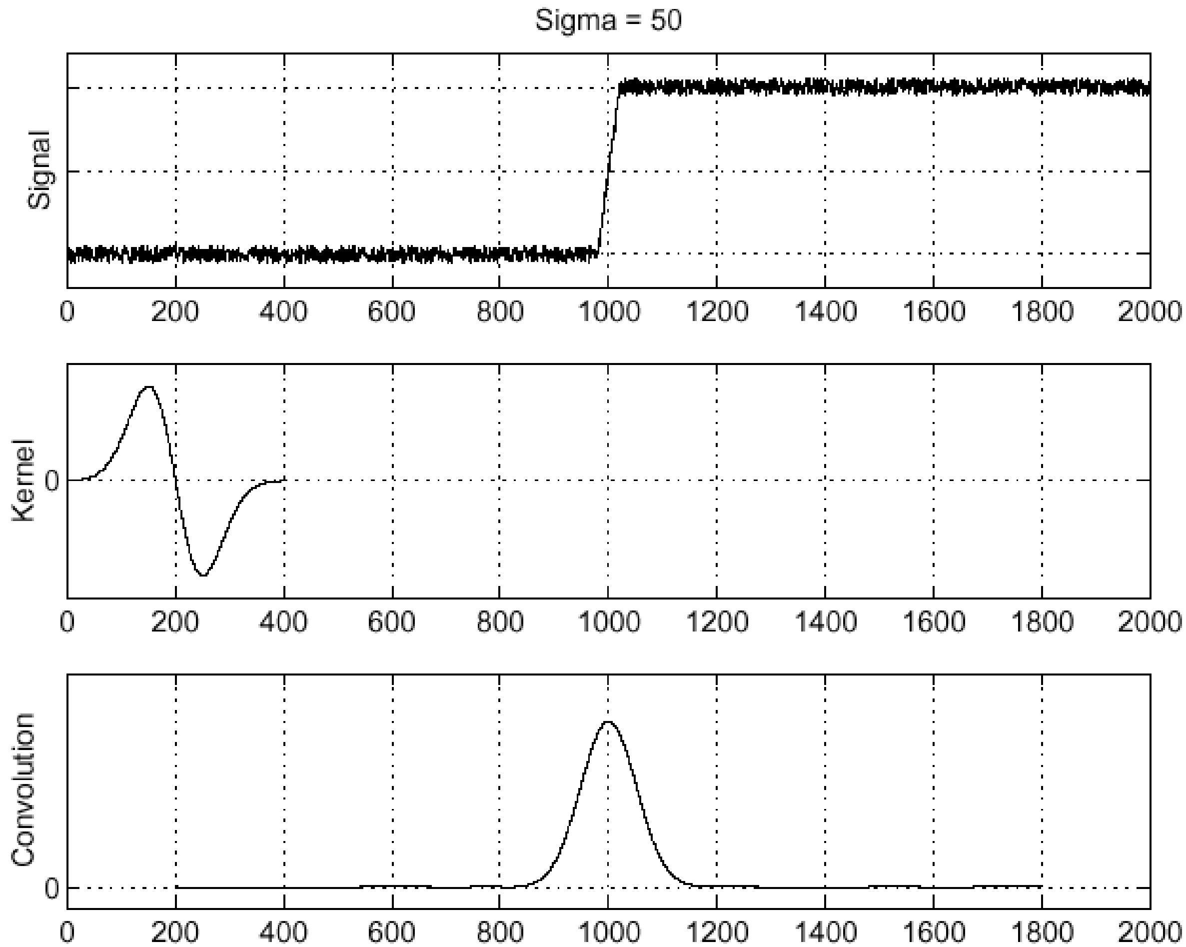
$$\frac{d}{dx}(f * g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

Smoothed Derivative

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



Results in 1D: Peak appears at edge



$$\frac{\partial}{\partial x}h$$

$$(\frac{\partial}{\partial x}h) \star f$$

Smoothed Derivative in 2D

- What is “derivative” in 2D? Gradient:

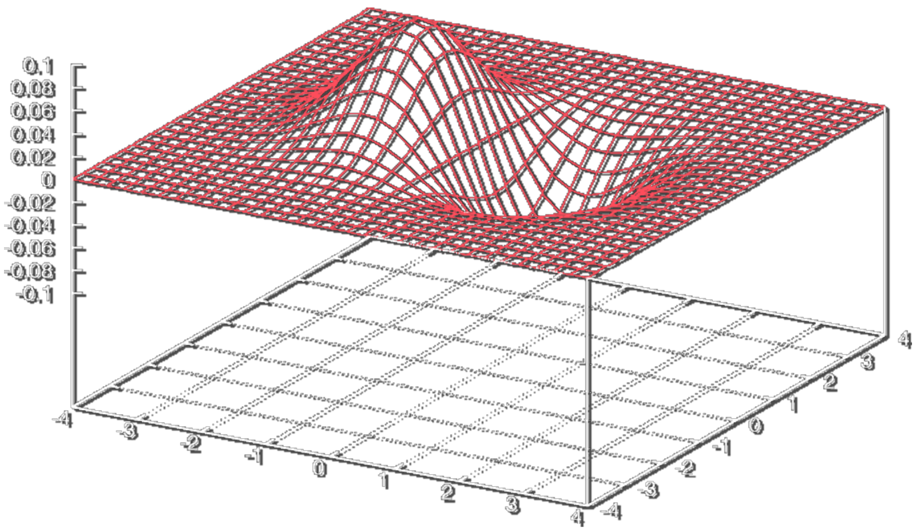
$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

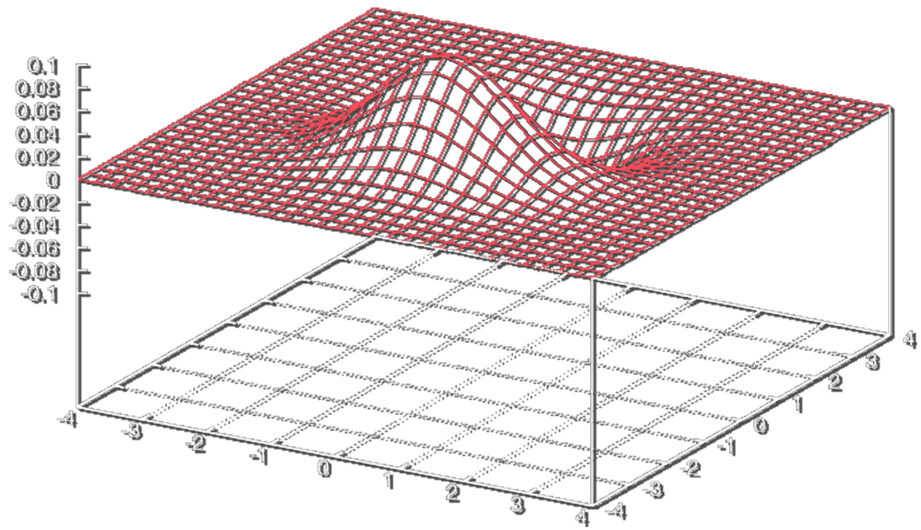
- Gaussian is separable! $G_2(x,y) = G_1(x)\,G_1(y)$

- 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





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

Edge Detection using Derivative of Gaussian

Original Image

Smoothed Gradient Magnitude

Canny Edge Detector

- Smooth
- Find derivative
- Find maxima
- Threshold

Canny Edge Detector

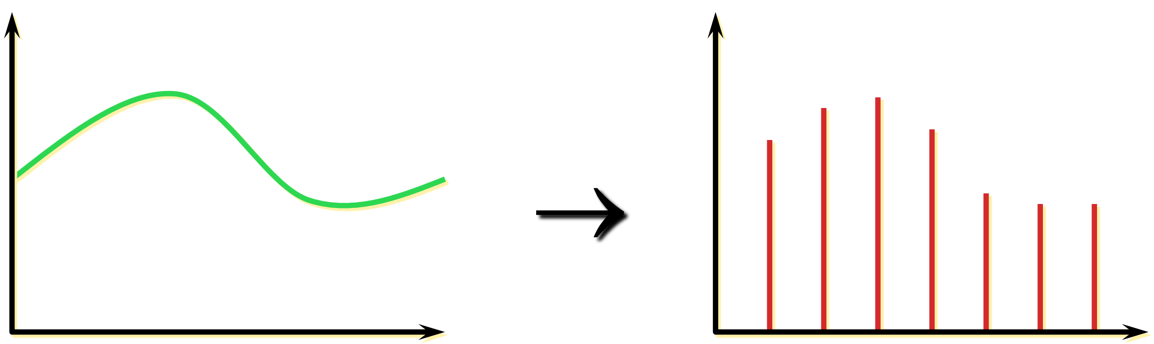
Original Image

Edges

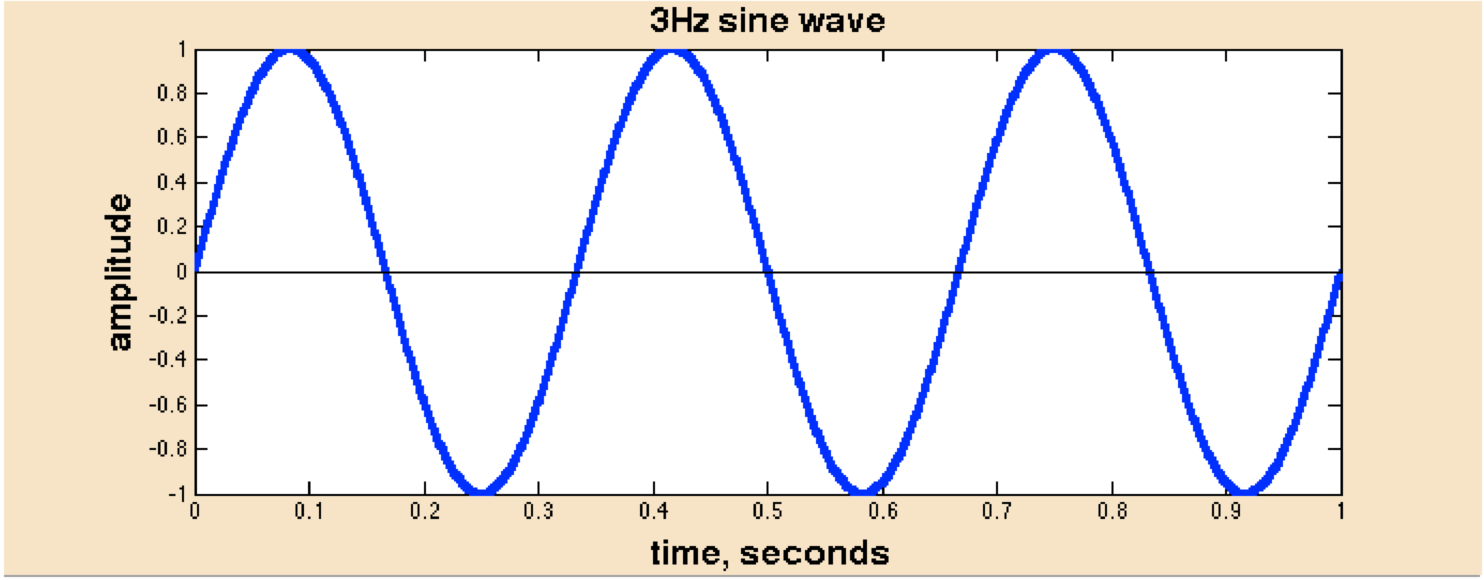
Sampling

Sampled Signals

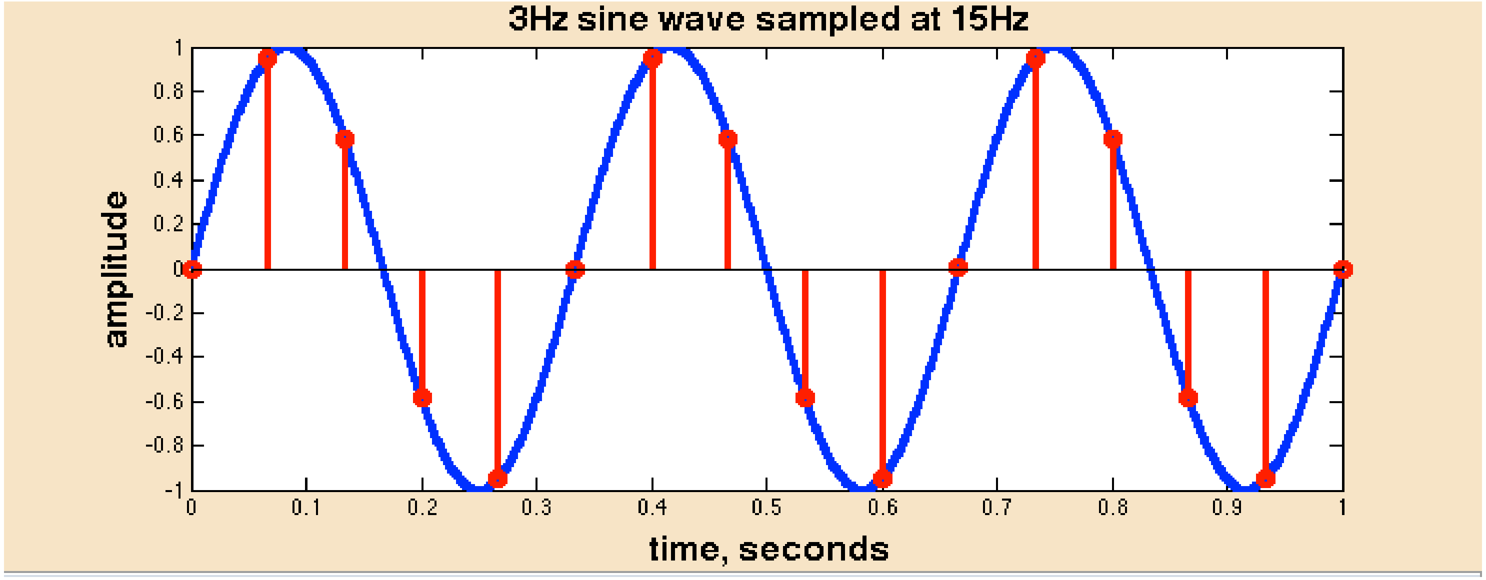
- Analog domain: Continuous signals
- Digital domain: Can't store continuous signal: instead store "samples"
  - Usually evenly sampled:
    $f_0=f(x_0)$, $f_1=f(x_0+\Delta x)$, $f_2=f(x_0+2\Delta x)$, $f_3=f(x_0+3\Delta x)$, ...



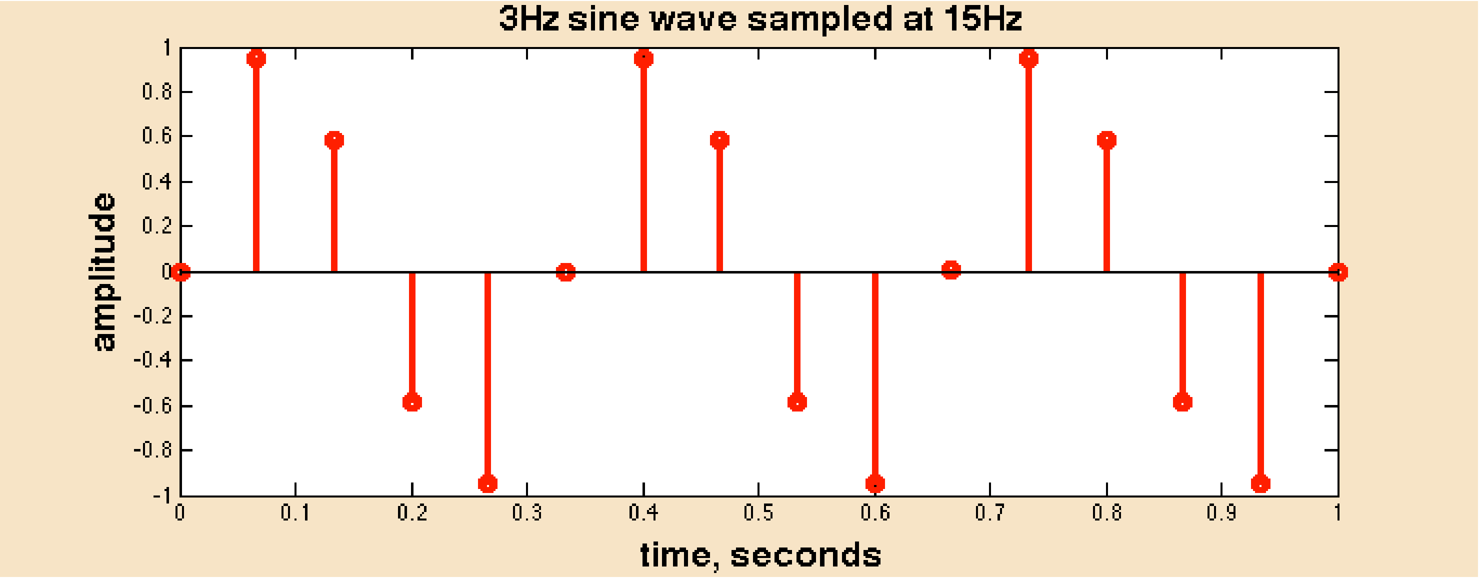
3 Hz sine wave



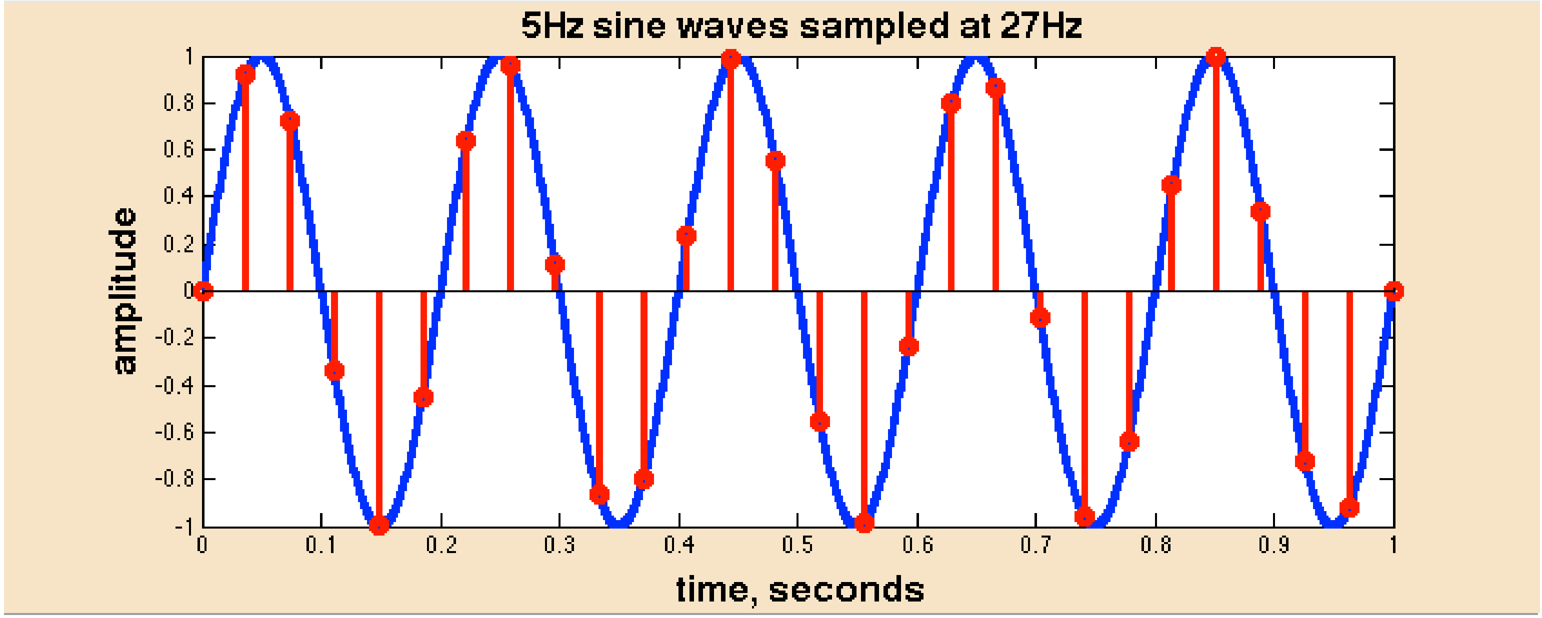
3Hz sine sampled at 15Hz



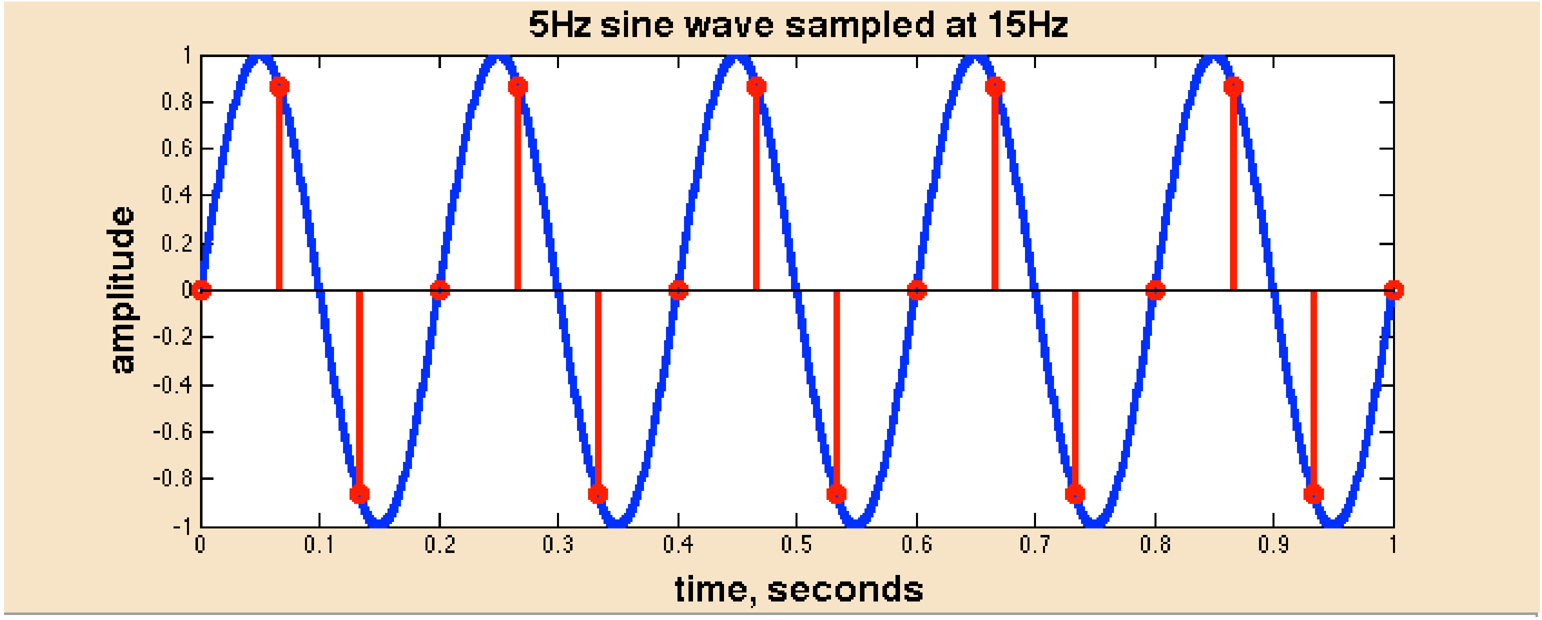
3Hz sine sampled at 15Hz



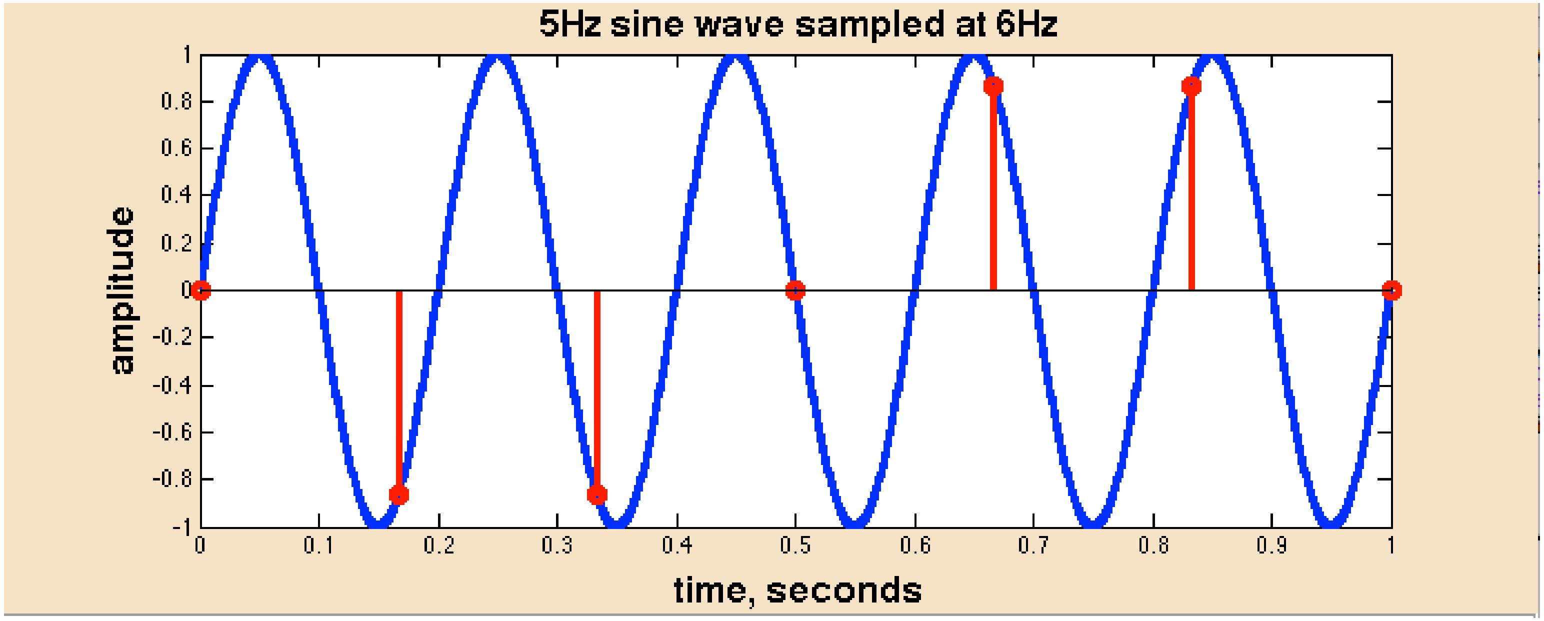
5Hz sine sampled at 27 Hz



5Hz sine sampled at 15Hz

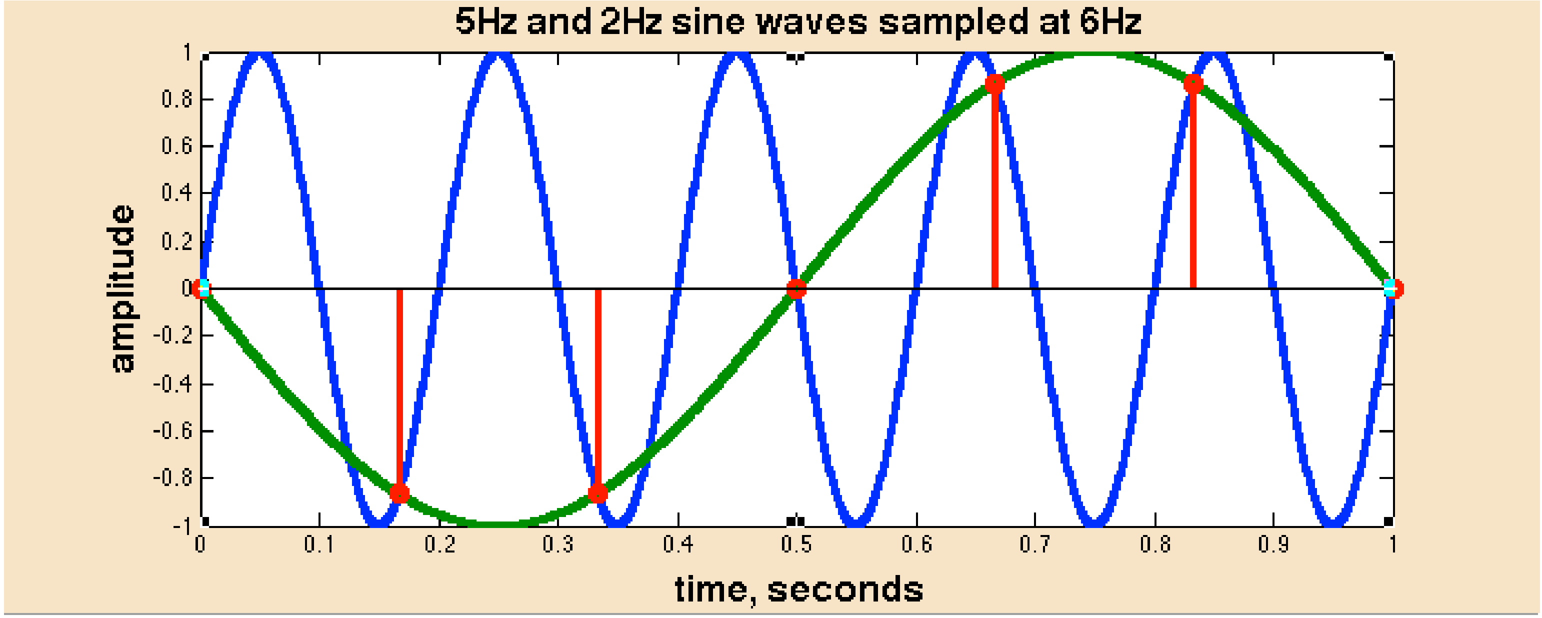


5Hz sine sampled at 6Hz



5Hz sine sampled at 6Hz

Aliasing!

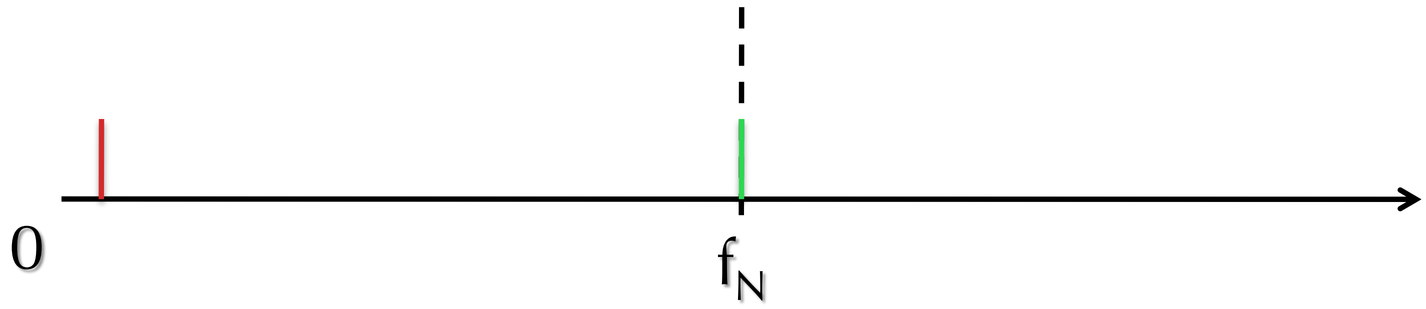


Aliasing

- Need to sample **at least twice per period** to capture the signal unambiguously
- Nyquist's theorem: Highest allowed signal frequency is half the sampling frequency = Nyquist frequency
- E.g., CD quality audio is 44,100 Hz
  - Highest frequency representable is 22,050 Hz
  - Limit of human hearing: ~17kHz to 20kHz

Aliasing in 1D

- Frequencies above Nyquist get "reflected" back below Nyquist



Aliasing strikes!

Preventing aliasing

- Use a sample rate high enough to capture frequencies of interest (i.e., > twice the highest frequency of interest)
- Apply a low-pass filter to remove frequencies above the Nyquist frequency, before sampling.

Discrete Convolution

- Integral becomes sum over samples

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

- Normalization condition is

$$\sum_i g_i = 1$$

Computing Discrete Convolutions

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

```
for x = 1 to n
    for i = 1 to m
        out(x) = out(x) + f(i) * g(x-i)
```

Does not handle boundaries!

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

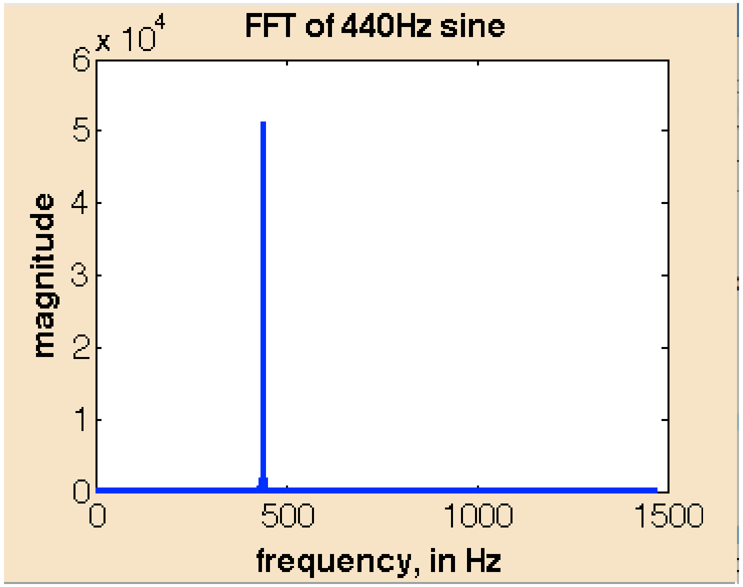
  O(*nm*)

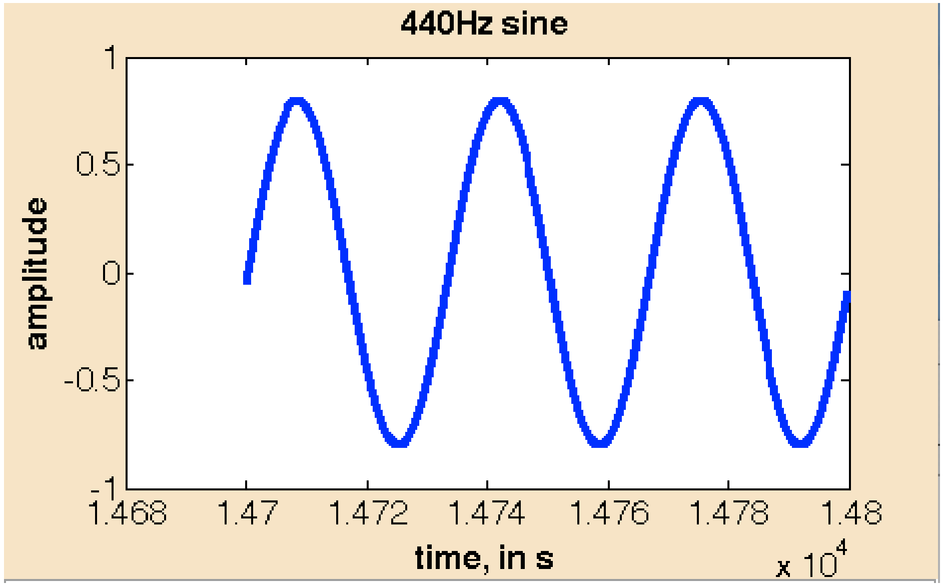
- OK for small filter kernels, bad for large ones

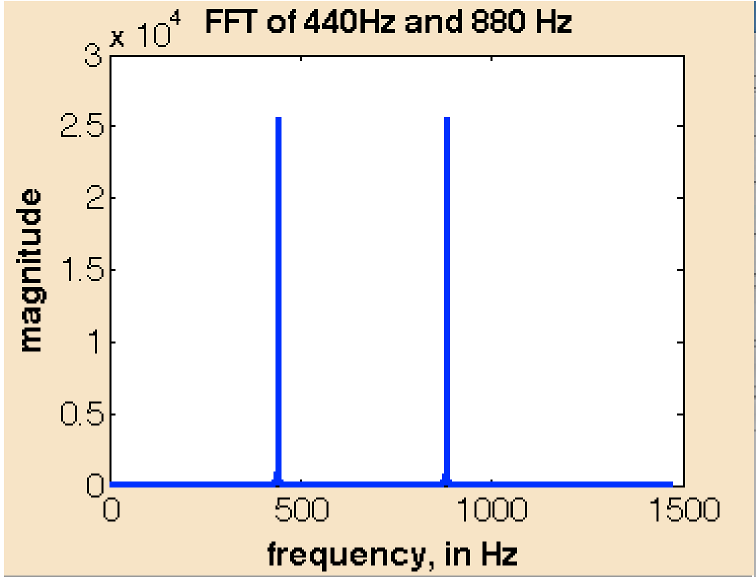
Fourier Analysis

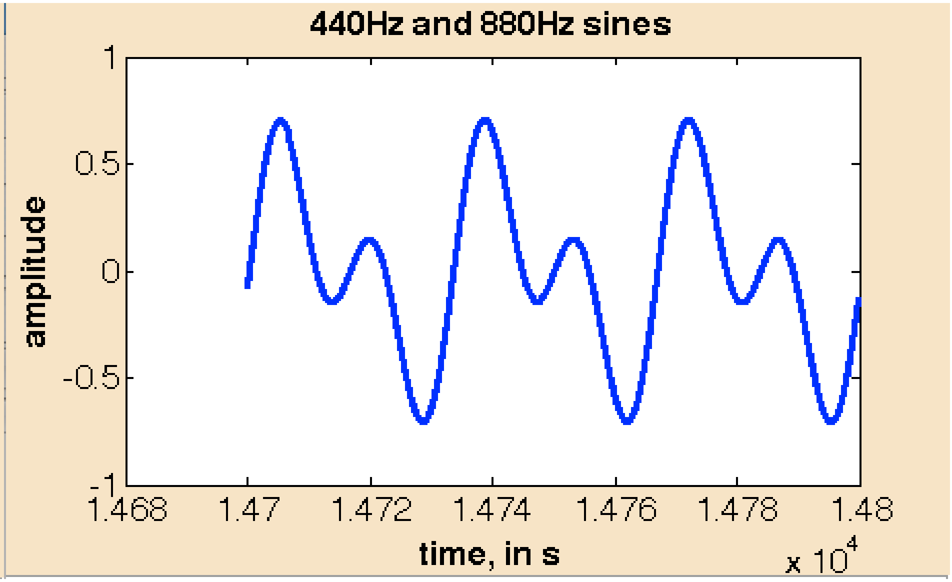
Frequency Domain

- Any signal can be represented as a sum of sinusoids at discrete **frequencies**, each with a given **magnitude** and **phase**


FFT of 440Hz sine
x 10^4
6
5
4
3
2
1
0
magnitude
0
500
1000
1500
frequency, in Hz


440Hz sine
1
0.5
0
-0.5
-1
amplitude
1.468
1.47
1.472
1.474
1.476
1.478
1.48
time, in s
x 10^4


FFT of 440Hz and 880 Hz
x 10^4
3
2.5
2
1.5
1
0.5
0
0
500
1000
1500
magnitude
frequency, in Hz


440Hz and 880Hz sines
1
0.5
0
-0.5
-1
1.468
1.47
1.472
1.474
1.476
1.478
1.48
amplitude
time, in s
x 10^4

Frequency content in audio

- Frequency related to pitch:
  - "A 440": 440 Hz
  - 880Hz: one octave above
- Frequency also related to timbre:
  - Real sounds contain many frequencies
  - Higher frequency content can make sounds "brighter"
- In speech, higher frequencies are related to vowels, consonants (independent of spoken/sung pitch)

Fourier Transform

- Transform applied to function to analyze a signal's frequency content
- Several versions:

	Continuous Time	**Discrete Time**
Aperiodic / unbounded time, continuous frequency	Fourier Transform	Discrete-time Fourier Transform (DTFT)
Periodic or bounded time, discrete frequency	Fourier Series	Discrete Fourier Transform (DFT) (FFT used here)

Fourier Series

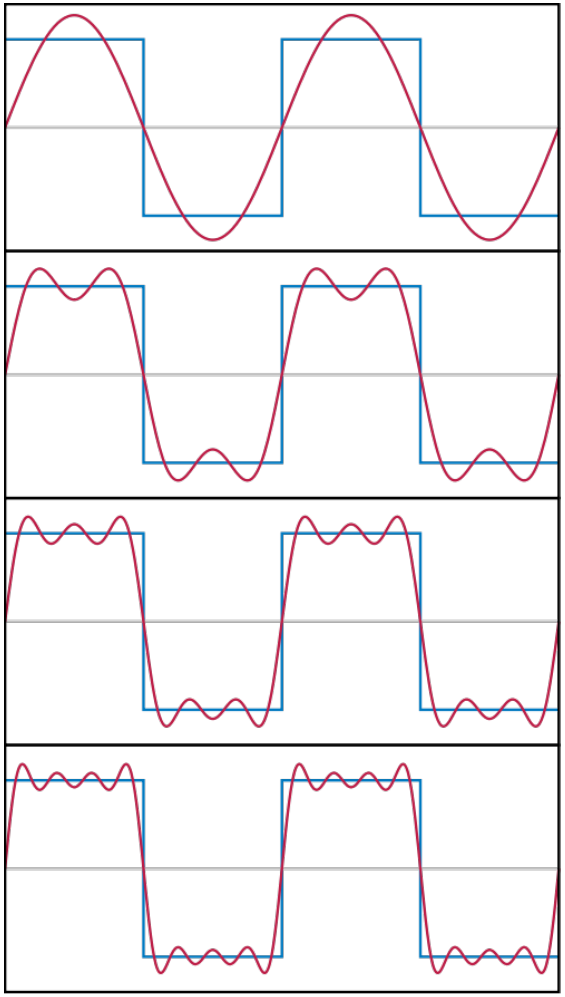
- Periodic function f(x) defined over $[-\pi \,..\, \pi]$

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

where

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

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



Applying Euler's Formula

- Euler's formula: $e^{ix} = \cos(x) + i\sin(x)$

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

becomes $f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$

where $c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx$

Fourier Transform

- [Continuous] Fourier transform:

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

- Discrete Fourier transform:

$$F_k = \sum_{x=0}^{n-1} f_x\, e^{-2\pi i \frac{k}{n} x}$$

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

The Convolution Theorem

- Fourier transform turns convolution into multiplication:

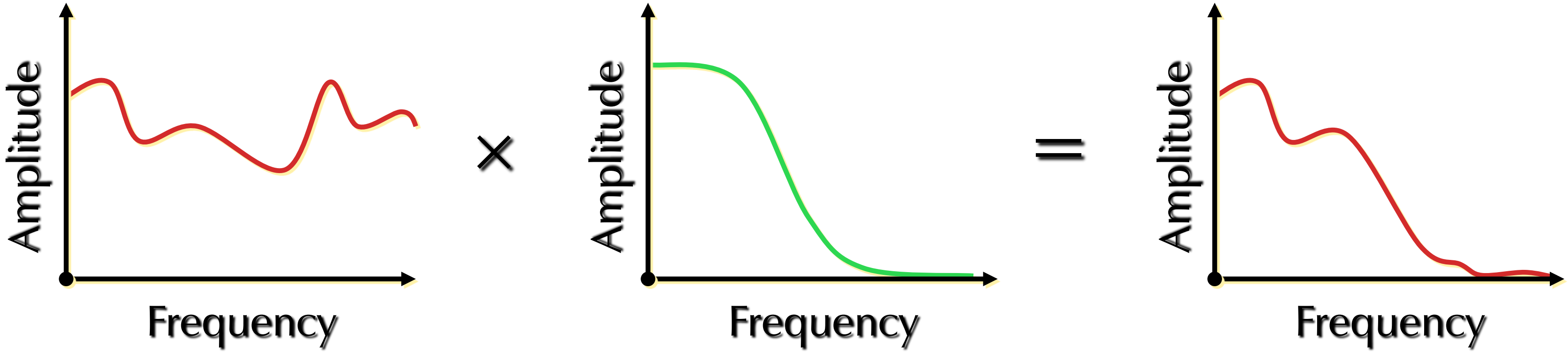
$$\mathcal{F}\big(f(x) * g(x)\big) = \mathcal{F}\big(f(x)\big)\ \mathcal{F}\big(g(x)\big)$$

(and vice versa):

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

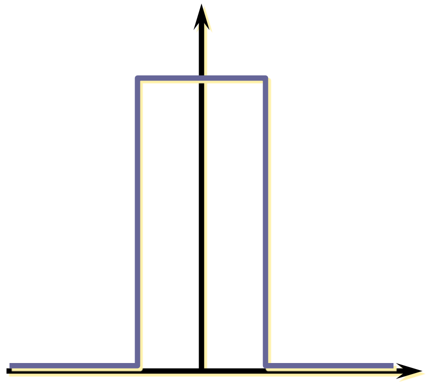
Fourier Transform and Convolution

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

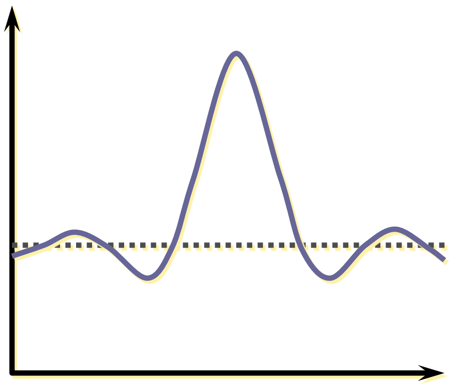


Fourier Transform and Convolution

- Box function?



- In frequency space: sinc function
  - 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