Fourier Transforms

CS 482, Prof. Stein

Lecture 03

Motivation: What do we mean when we say high-frequency and low-frequency?

We use these terms a lot to talk about manipulating images:

Motivation: What do we mean when we say high-frequency and low-frequency?

We use these terms a lot to talk about manipulating images:

- Edges are high-frequency regions in the image

Motivation: What do we mean when we say high-frequency and low-frequency?

We use these terms a lot to talk about manipulating images:

- Edges are high-frequency regions in the image
- Noise is usually high-frequency

Motivation: What do we mean when we say high-frequency and low-frequency?

We use these terms a lot to talk about manipulating images:

- Edges are high-frequency regions in the image
- Noise is usually high-frequency
- Blurring promotes low-frequency components of the image, while smoothing out high-frequency components

We want to be able to talk about this more precisely, and Fourier Transforms give us a language for doing that.

Motivation: Fourier Transforms will help us better understand our tools

We won't be using Fourier Transforms directly very much during the rest of the course, but we will constantly be talking about frequency space representations of images as we learn about more techniques for image manipulation.

Reading & Slide Credits

Readings:

- Szeliski 3.4 Fourier Transforms

Slide Credits (from which many of these slides are either directly taken or adapted)

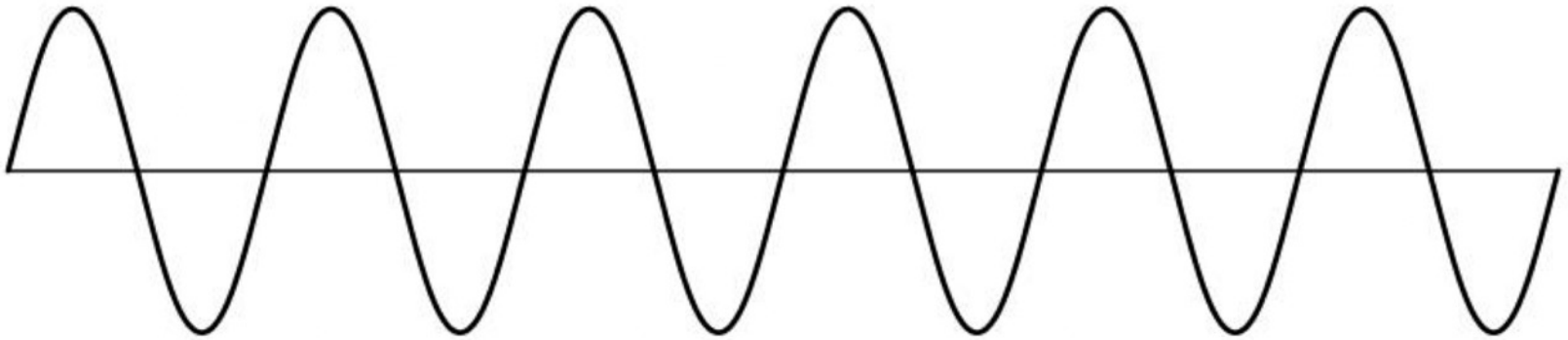
- CMU Computer Vision Course (Yannis Gkioulekas)
- Cornell Tech CV Course

Summing Sine Waves

A basic sine wave:

$$A\sin(\omega x + \phi)$$

where A is the amplitude, ω is the frequency, and ϕ is the phase.



Summing Sine Waves

A basic sine wave:

$$A\sin(\omega x + \phi)$$

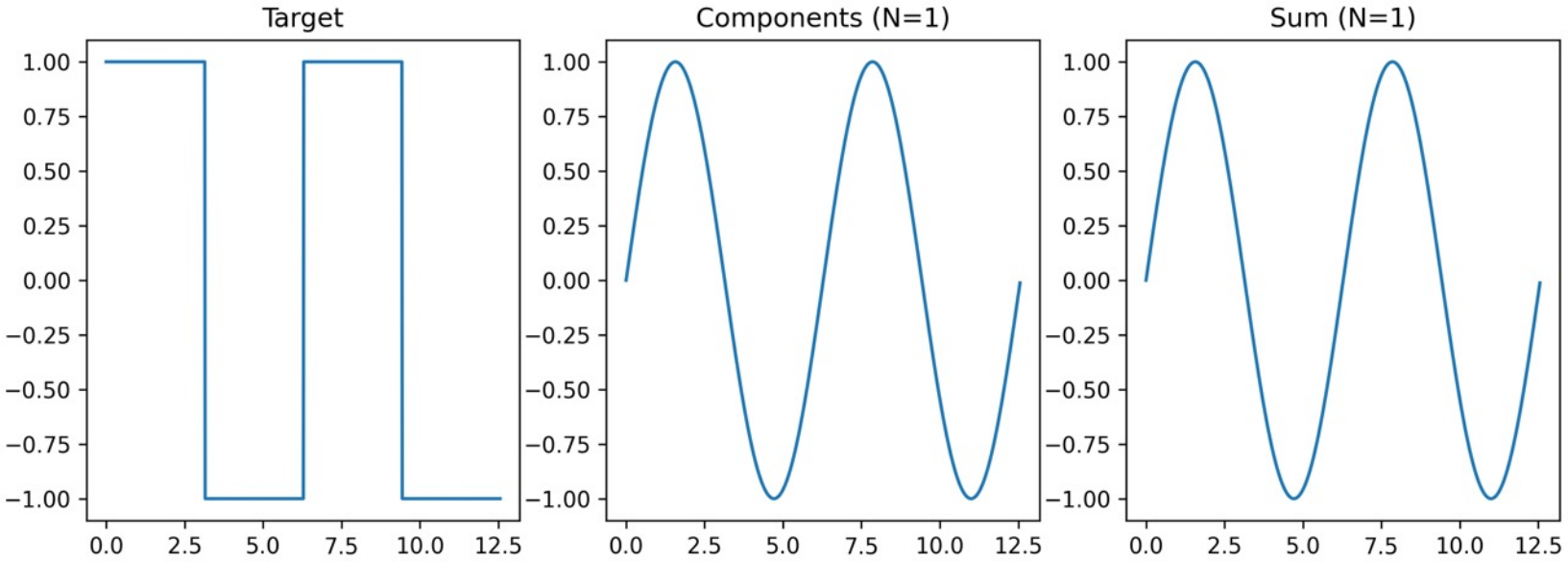
where A is the amplitude, ω is the frequency, and ϕ is the phase.

Fourier claimed that we could reproduce any signal by adding up enough sine waves of different frequencies.

We now know (hundreds of years later) that it's not quite "any" signal, but for us we won't have to worry about the exceptions.

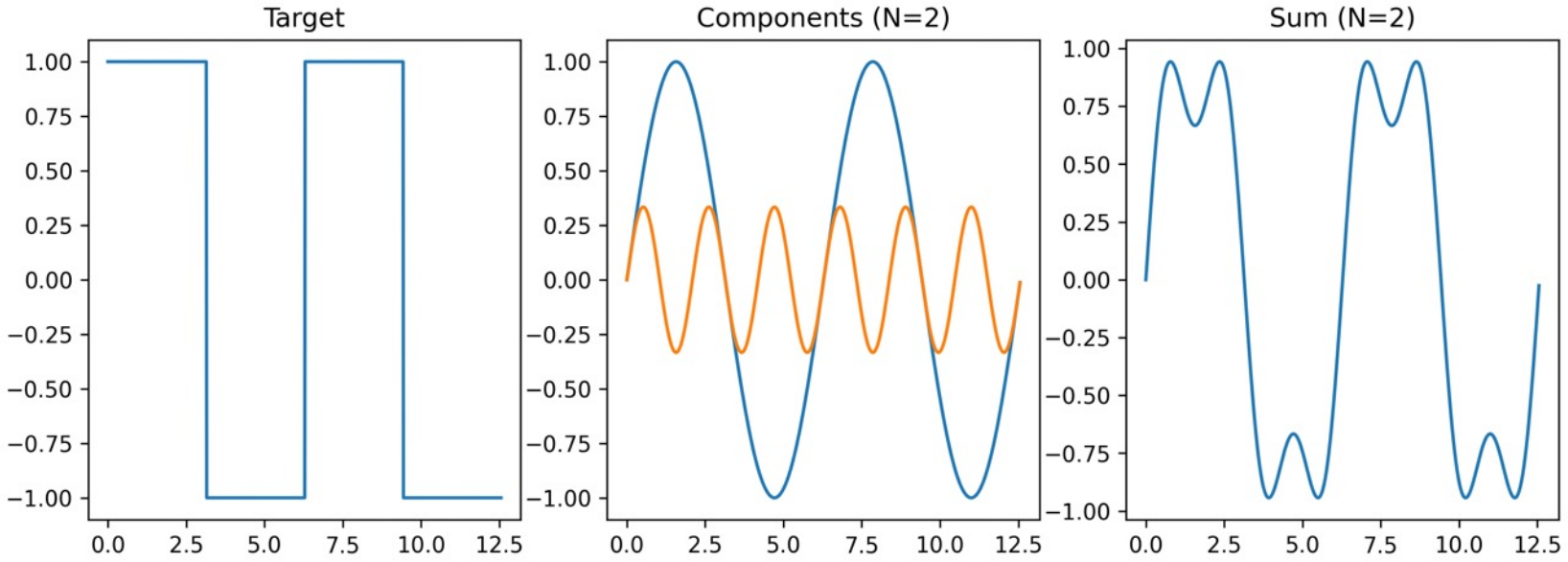
Example: Building a square wave

Let's say you wanted to build a square wave:



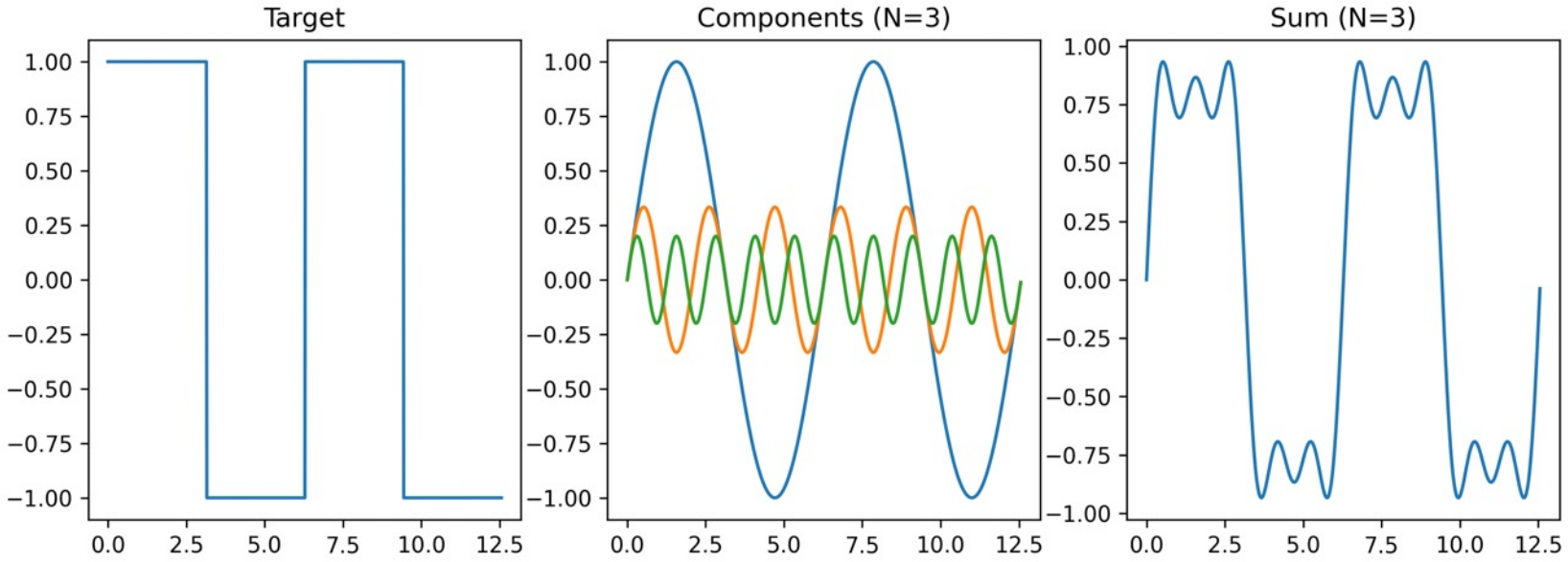
Example: Building a square wave

Let's say you wanted to build a square wave:



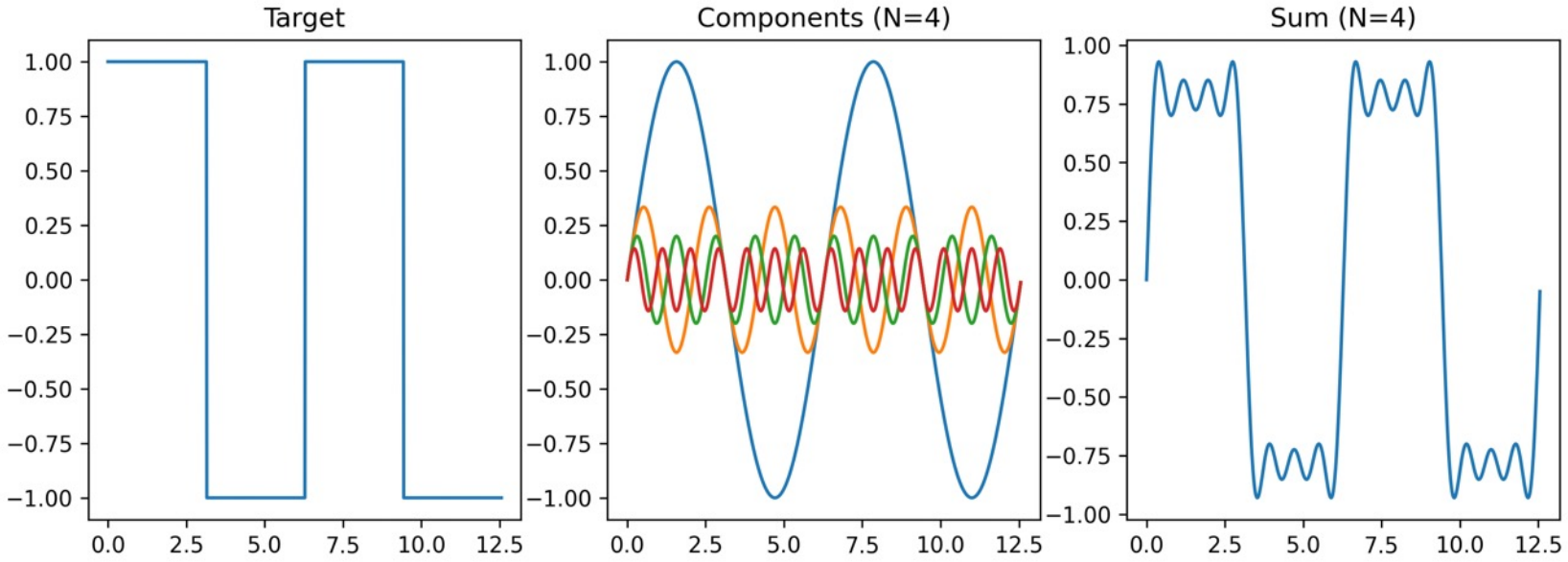
Example: Building a square wave

Let's say you wanted to build a square wave:



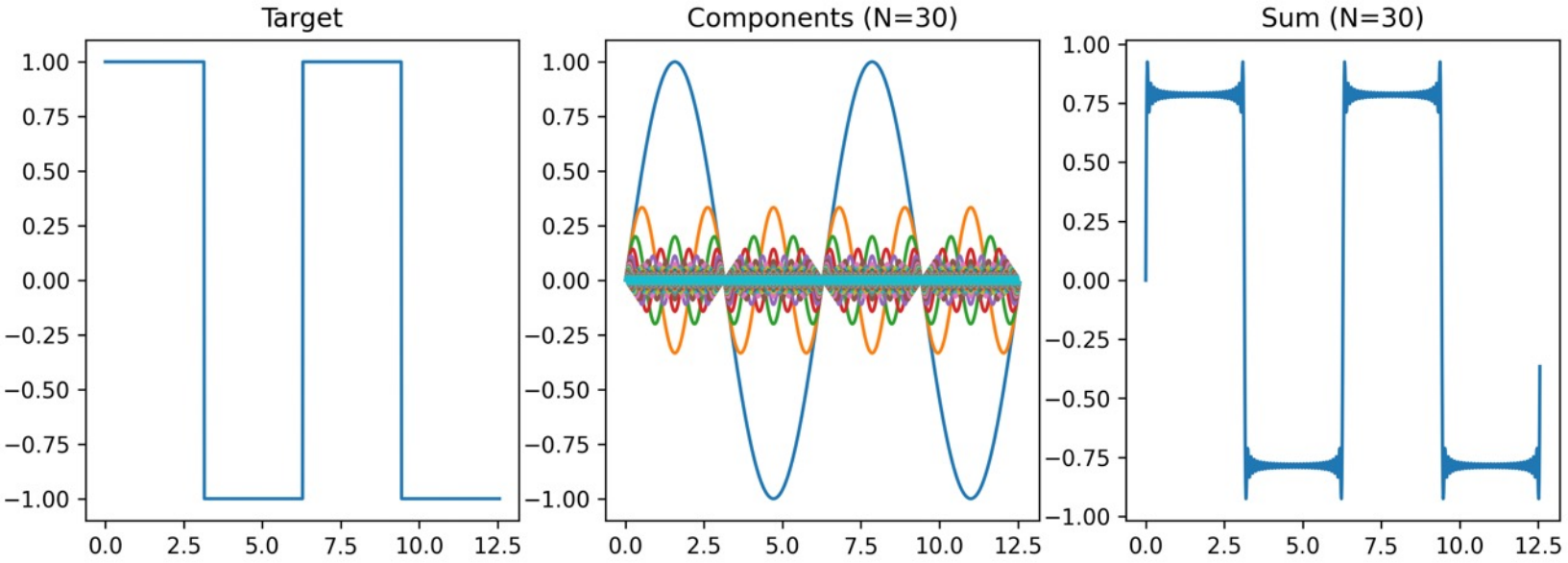
Example: Building a square wave

Let's say you wanted to build a square wave:



Example: Building a square wave

Let's say you wanted to build a square wave:



Example: Building a Square Wave

The series decomposition for a square wave might look like this:

$$f_{\text{square}}(x) = \sin(x) + \frac{1}{3}\sin(3x) + \cdots = \sum_{i=0}^{\infty} \frac{\sin((2i+1)x)}{2i+1}$$

Example: Building a Square Wave

The series decomposition for a square wave might look like this:

$$f_{\text{square}}(x) = \sin(x) + \frac{1}{3}\sin(3x) + \cdots = \sum_{i=0}^{\infty} \frac{\sin((2i+1)x)}{2i+1}$$

What does this mean? Why should we care?

Example: Building a Square Wave

The series decomposition for a square wave might look like this:

$$f_{\text{square}}(x) = \sin(x) + \frac{1}{3}\sin(3x) + \cdots = \sum_{i=0}^{\infty} \frac{\sin((2i+1)x)}{2i+1}$$

What does this mean? Why should we care?

It's just another way of representing this data.

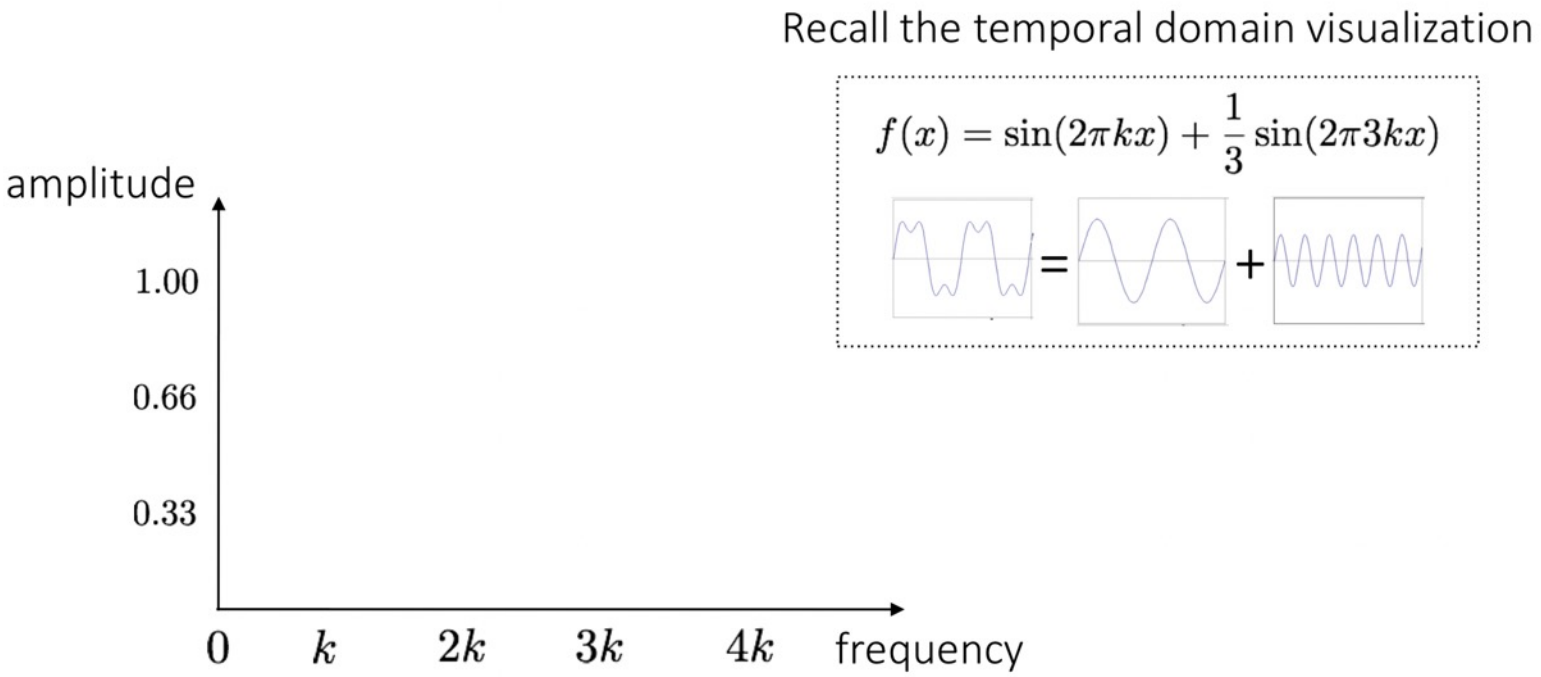
The square wave can be alternatively represented as a sum of all these sinusoids. This new representation allows us to ask different questions like *how important are the higher-frequency (or lower frequency) components to making up this signal?*

Visualizing Frequency Space

We have a signal in terms of sine waves. Can we visualize the signal in frequency space?

Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

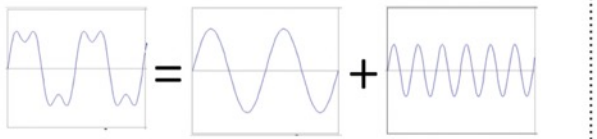


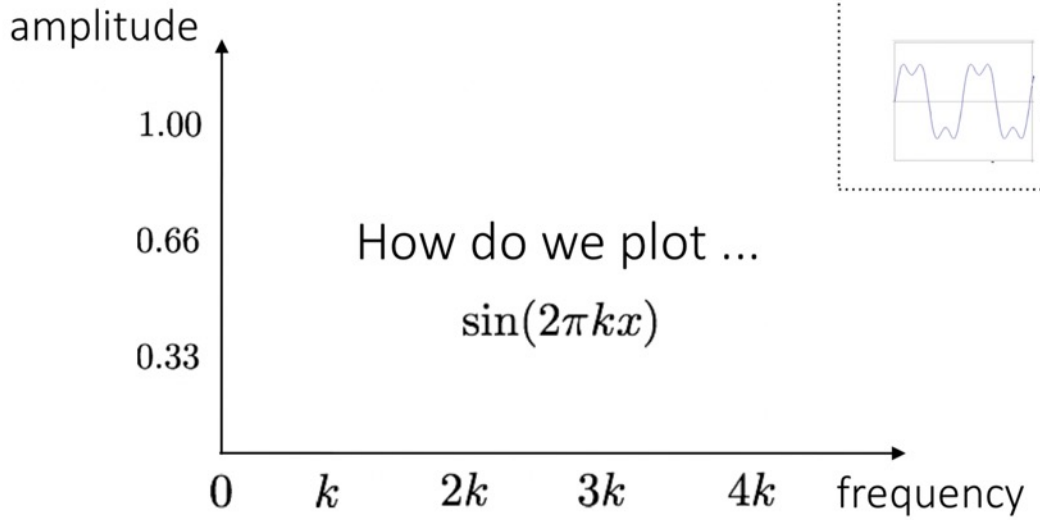
Visualizing Frequency Space

We have a signal in terms of sine waves. Can we visualize the signal in frequency space?

Recall the temporal domain visualization

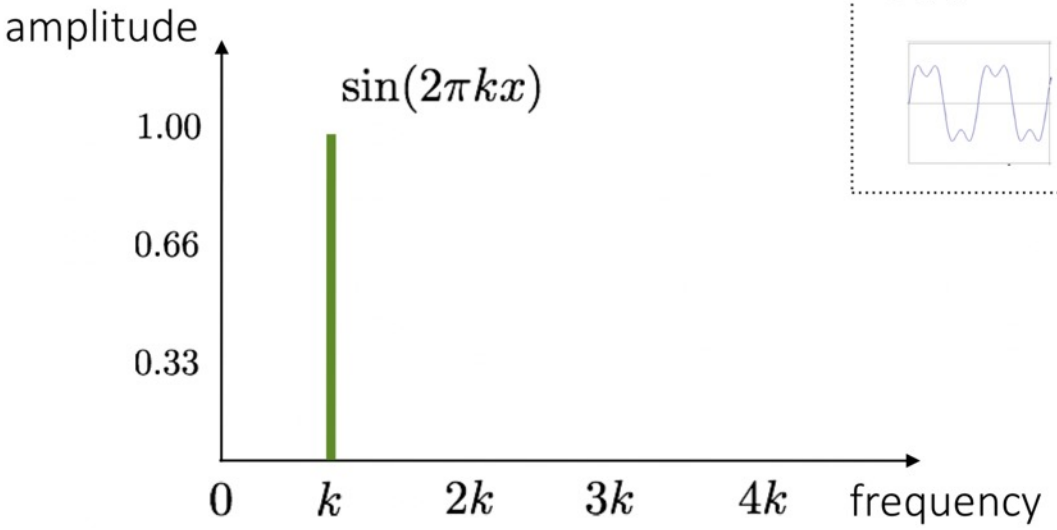
$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$





Visualizing Frequency Space

We have a signal in terms of sine waves. Can we visualize the signal in frequency space?



Recall the temporal domain visualization

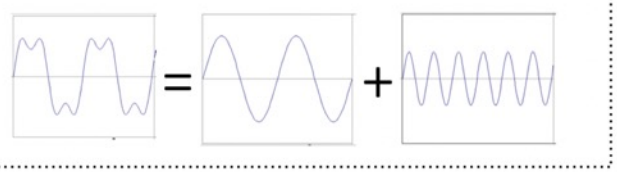
$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

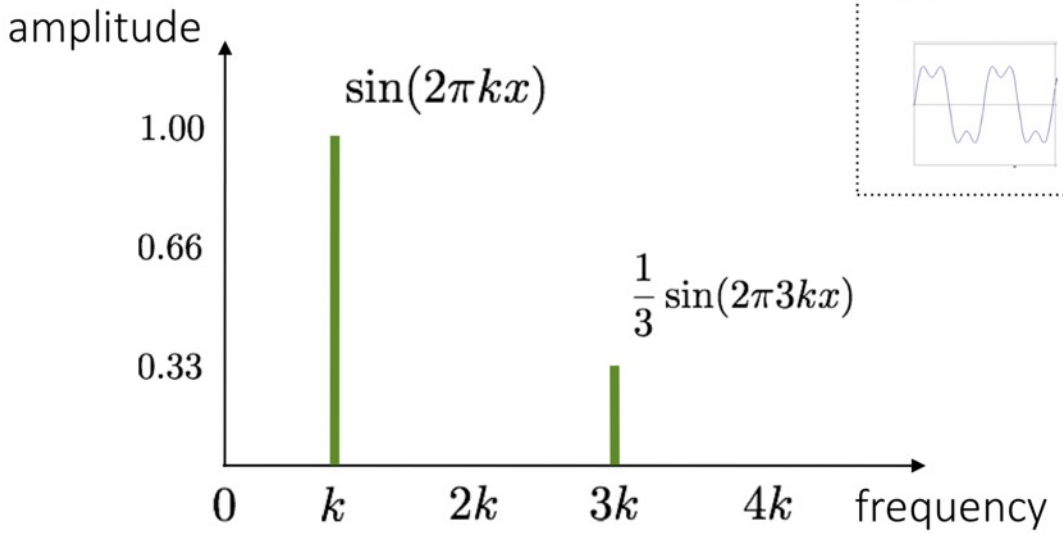
Visualizing Frequency Space

We have a signal in terms of sine waves. Can we visualize the signal in frequency space?

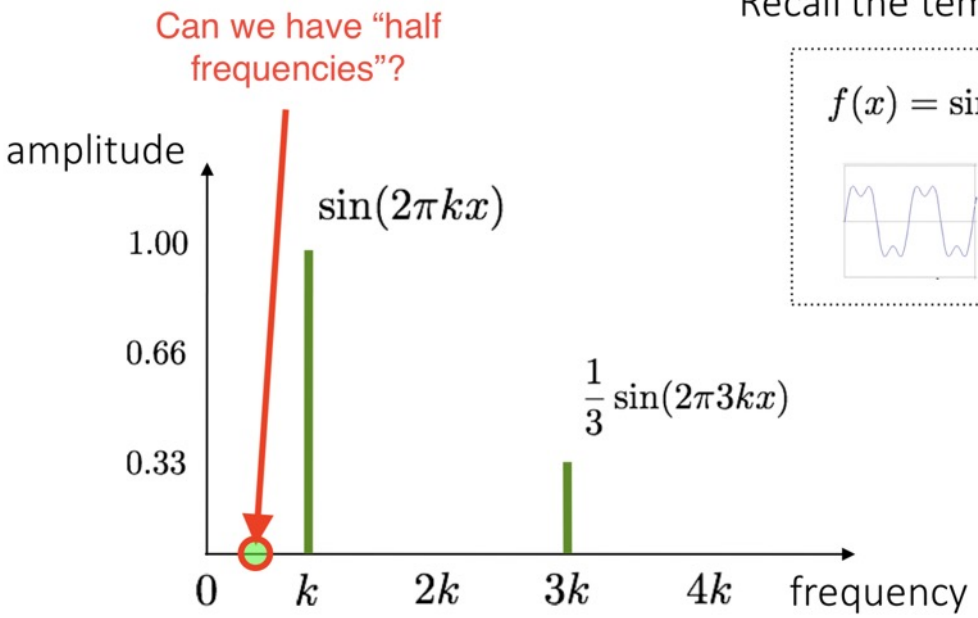
Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$





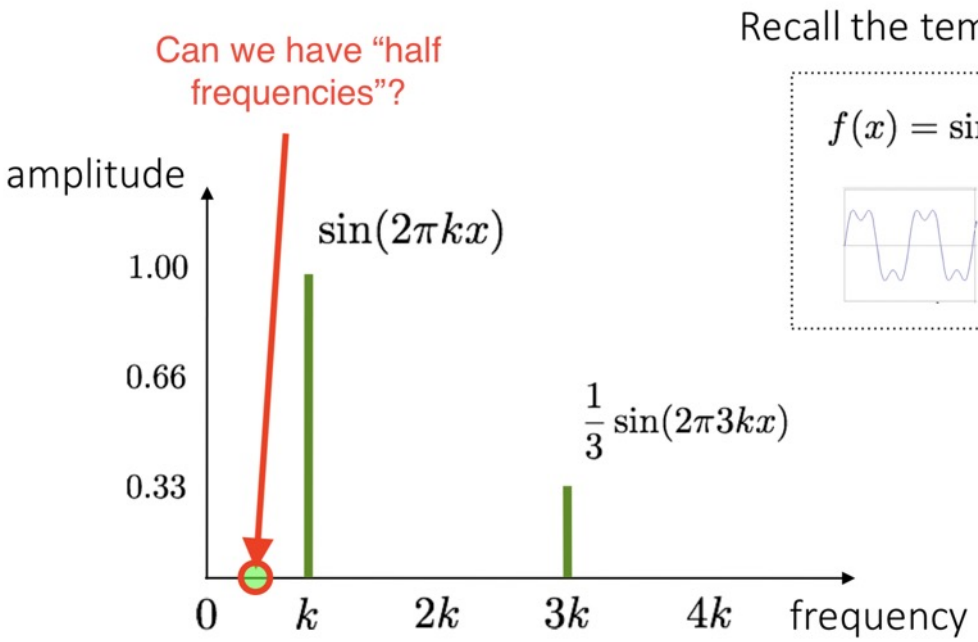
Visualizing Frequency Space



Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

Visualizing Frequency Space



Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

Not really… In a periodic window, sine waves that don't match up, can be built by summing up other sine waves (like the square wave).

Limits of Discrete Fourier Transforms

In continuous Fourier Transforms there is no maximum frequency (it's infinite); this is not true for discrete Fourier Transforms.

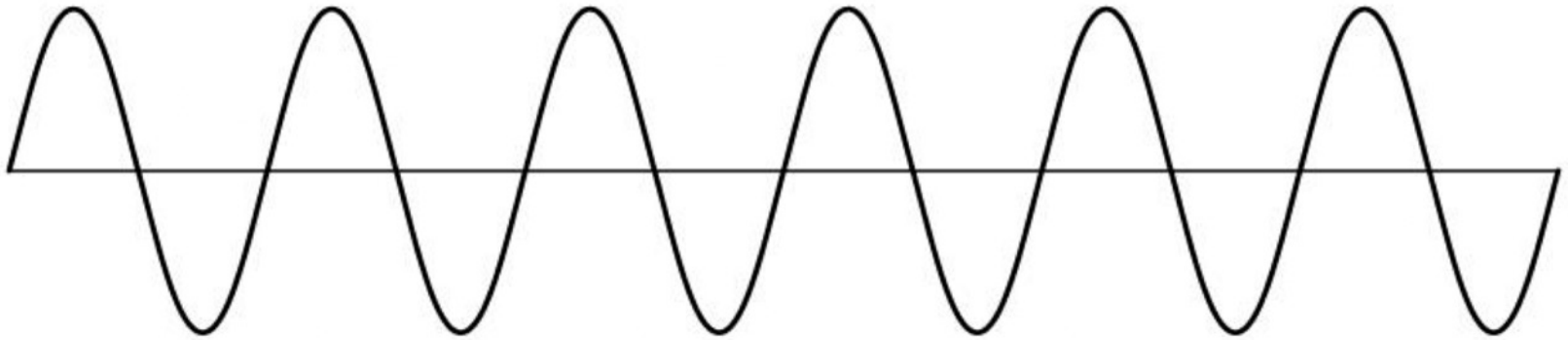
Limits of Discrete Fourier Transforms

In continuous Fourier Transforms there is no maximum frequency (it's infinite); this is not true for discrete Fourier Transforms.

How do continuous and discrete signals relate to one another?

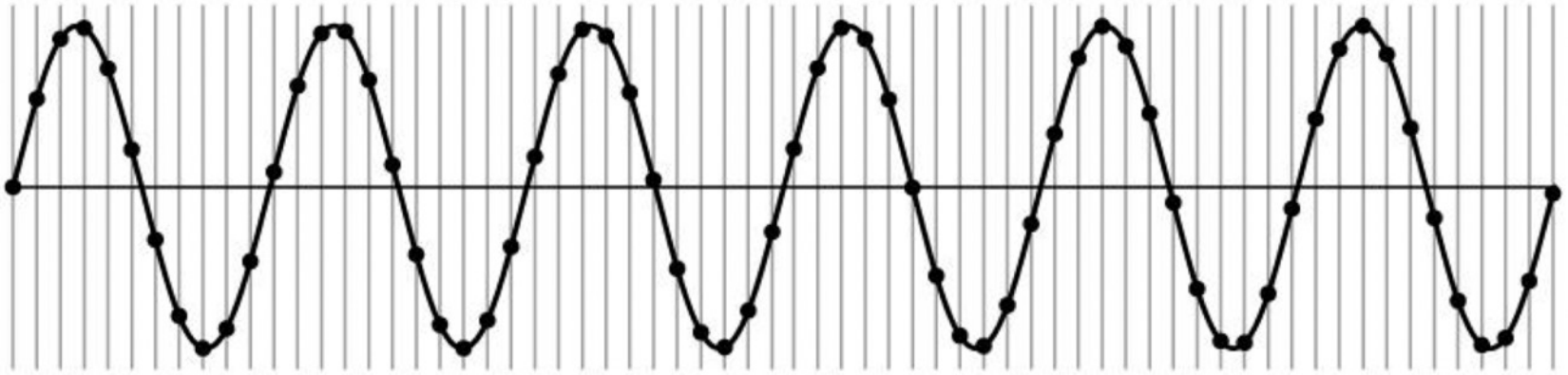
Sampling: How would you discretize this?

A sine wave:



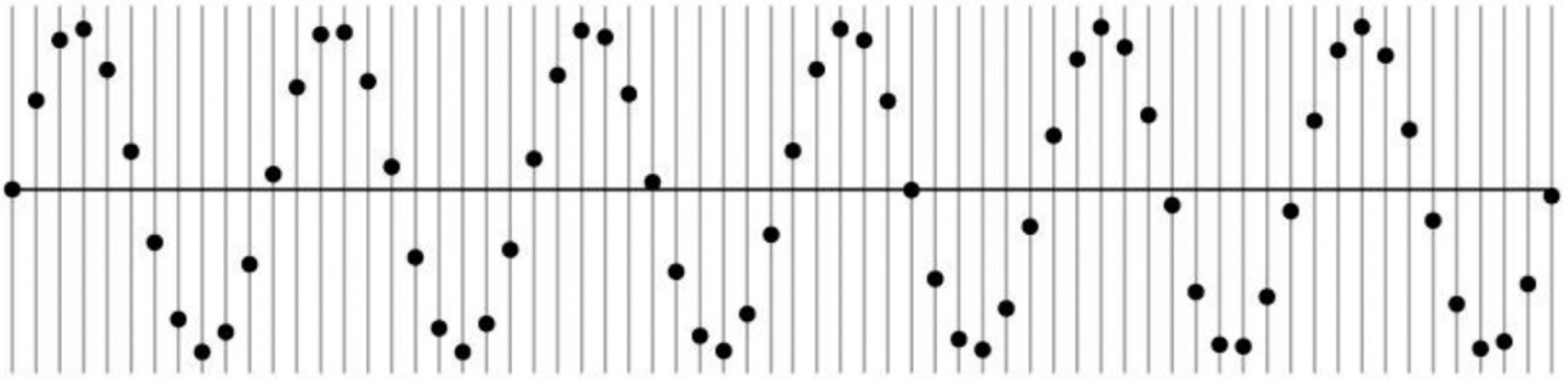
Sampling: How would you discretize this?

A sine wave:



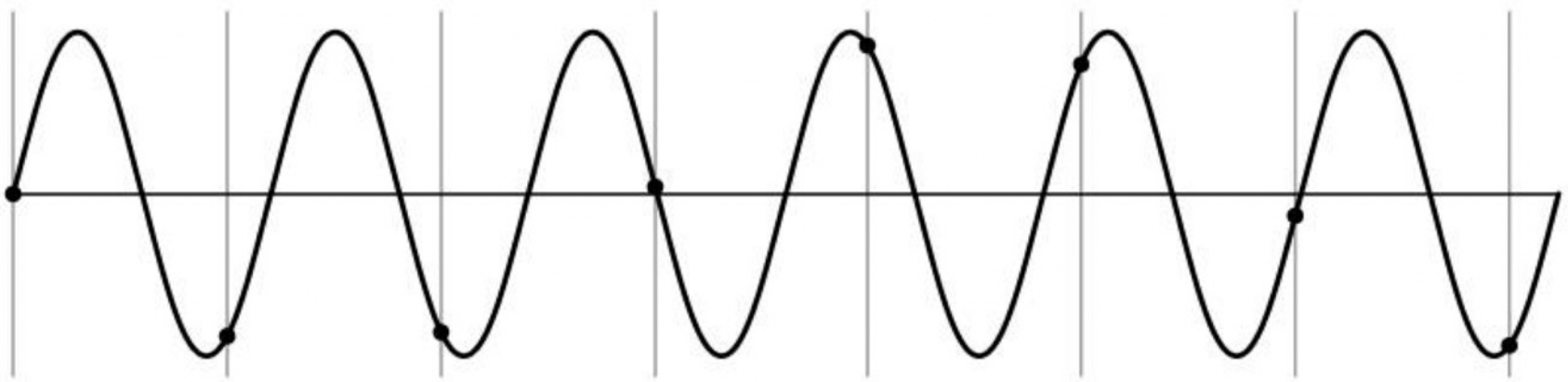
Sampling: How would you discretize this?

A sine wave:



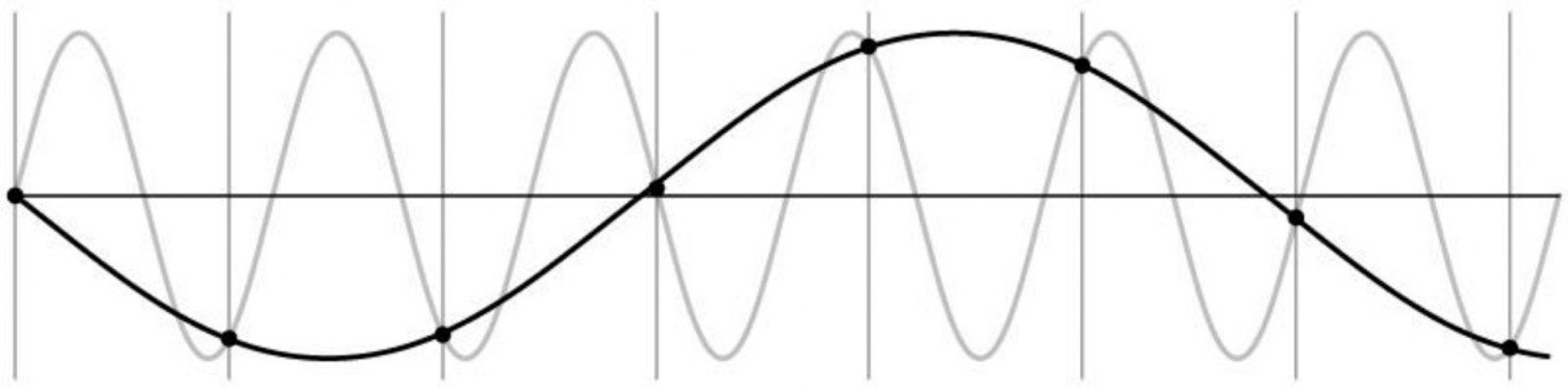
Sampling: How would you discretize this?

A sine wave:



Sampling: How would you discretize this?

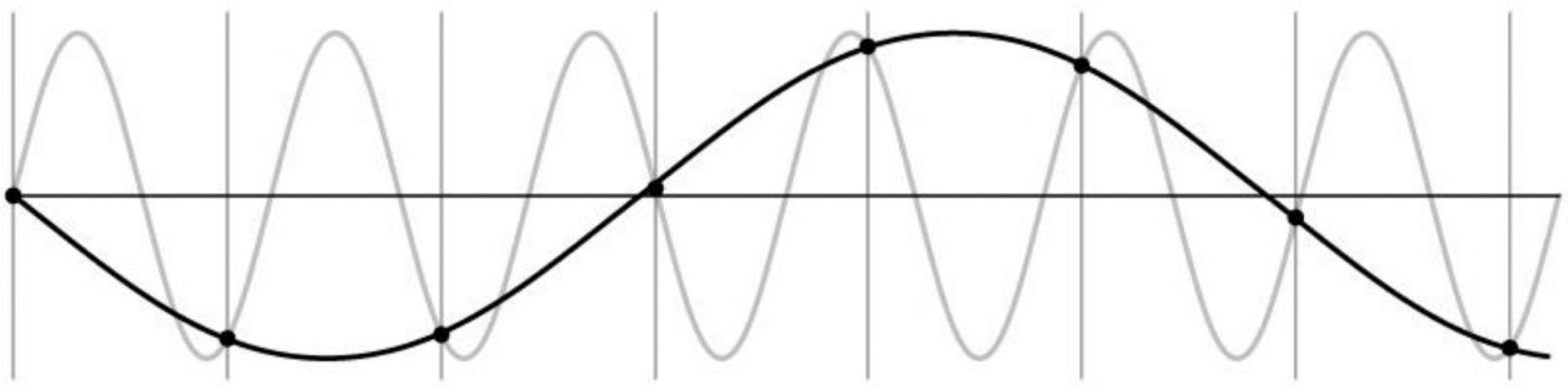
A sine wave:



This sampling rate is too low to capture the underlying wave. The darker signal would have given the same samples.

Sampling: How would you discretize this?

A sine wave:

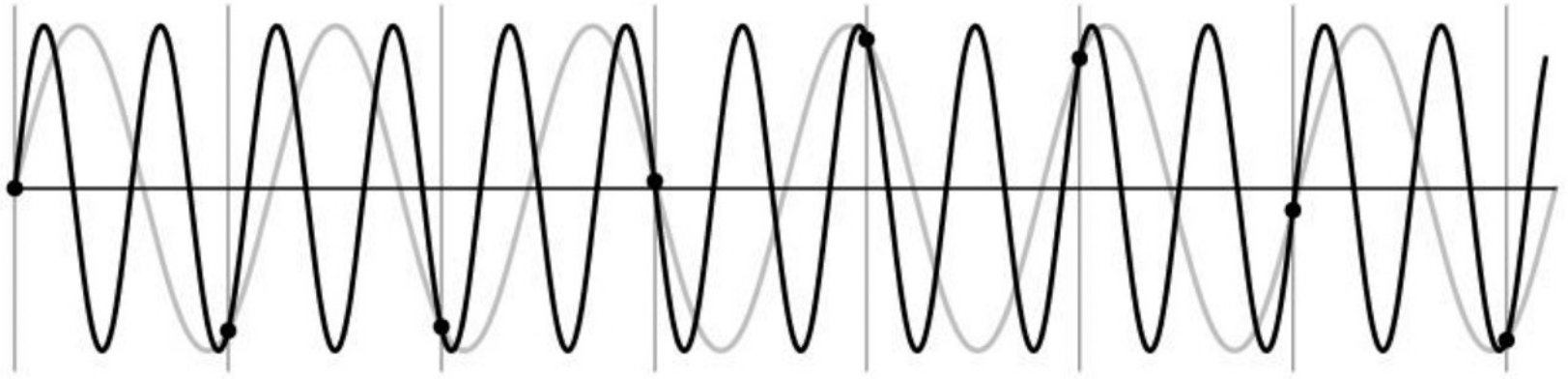


This sampling rate is too low to capture the underlying wave. The darker signal would have given the same samples.

This is known as *aliasing*: you mistake your signal for another.

Sampling: How would you discretize this?

A sine wave:

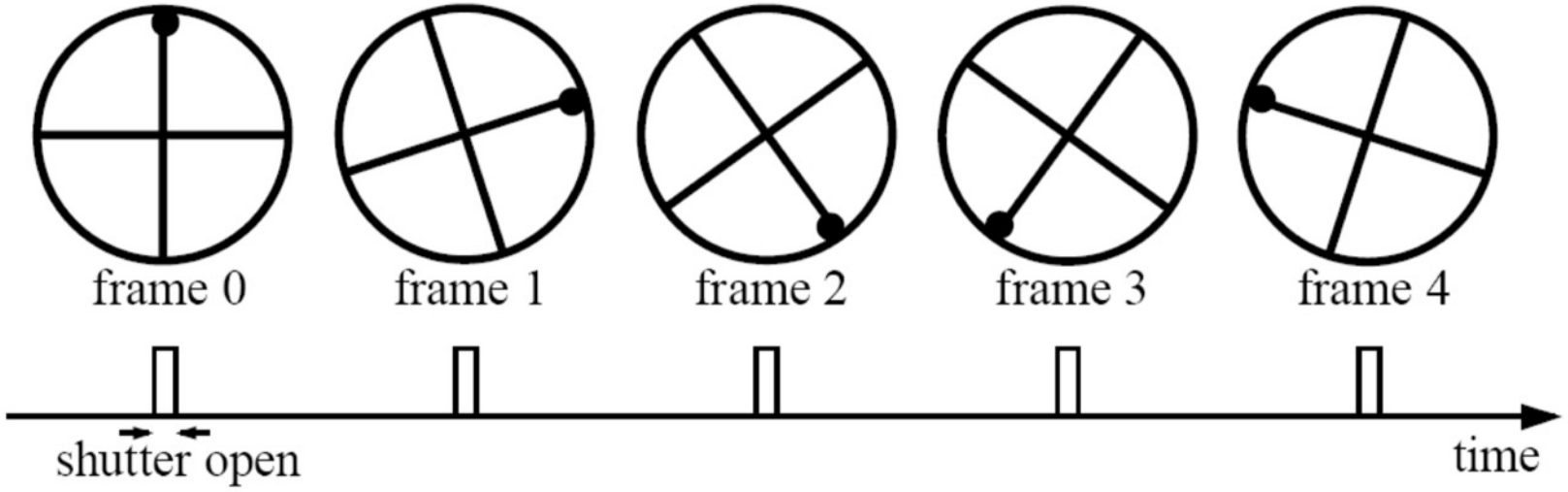


Alternatively, you could be mistaking a higher frequency signal for a lower frequency one… this is *always possible* for discrete signals.

The Wagon-Wheel Effect
Aliasing in time

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

The Nyquist Limit (Nyquist Rate)

To avoid aliasing:

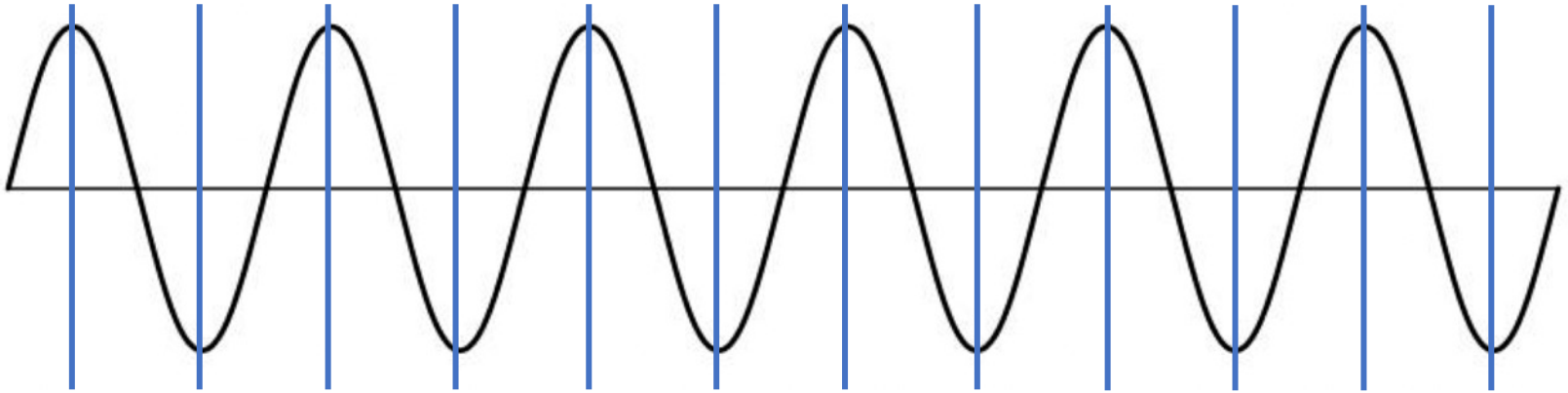
- Sampling rate ≥ 2 * max frequency in the image: in other words you must have ≥ 2 samples per cycle
- This minimum sampling rate is called the **Nyquist rate**

The Nyquist Limit (Nyquist Rate)

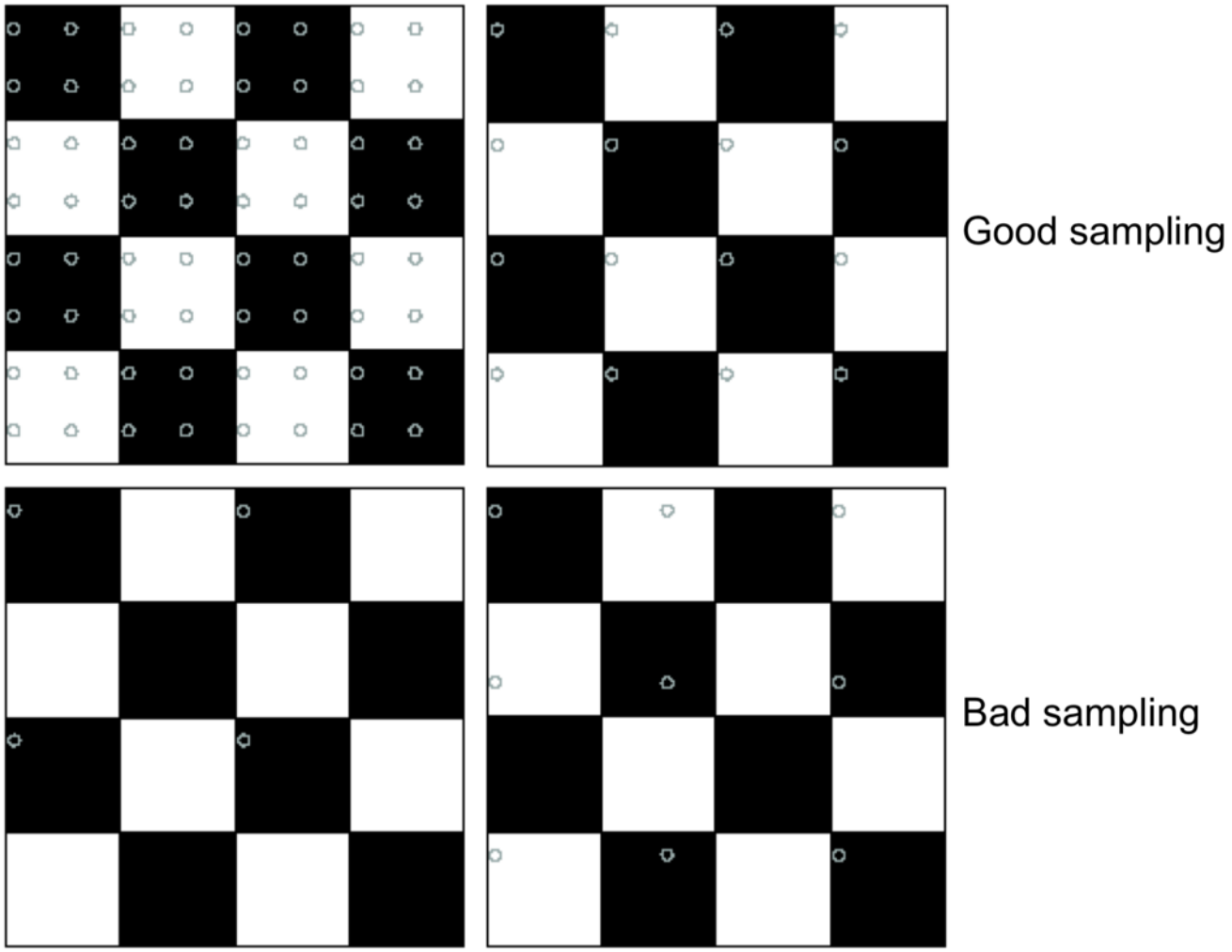
To avoid aliasing:

- Sampling rate ≥ 2 * max frequency in the image: in other words you must have ≥ 2 samples per cycle
- This minimum sampling rate is called the **Nyquist rate**

This makes intuitive sense: you need to have at least one point at the "top" of the cycle and one point at the "bottom of the cycle".



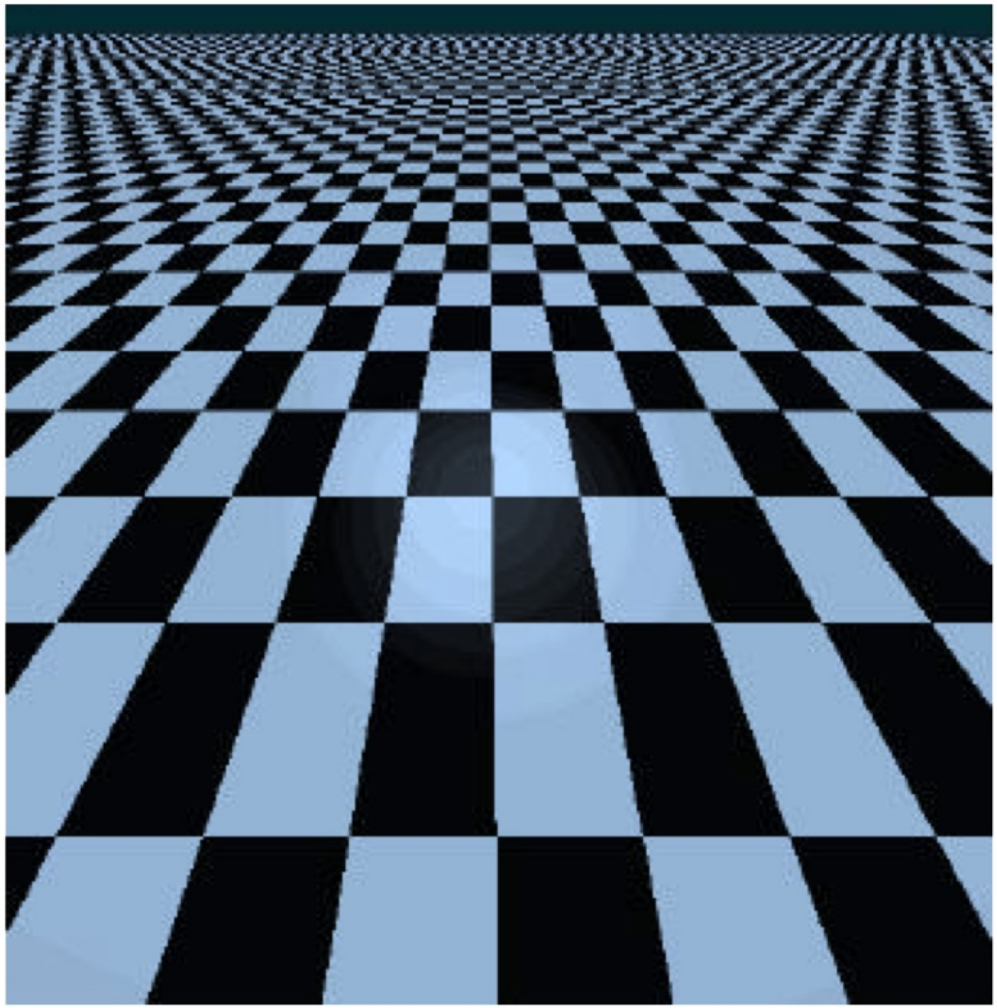
A 2D example of the Nyquist Limit



A 2D example of the Nyquist Limit

Simulated scenes can experience this to a great degree if one isn't careful.

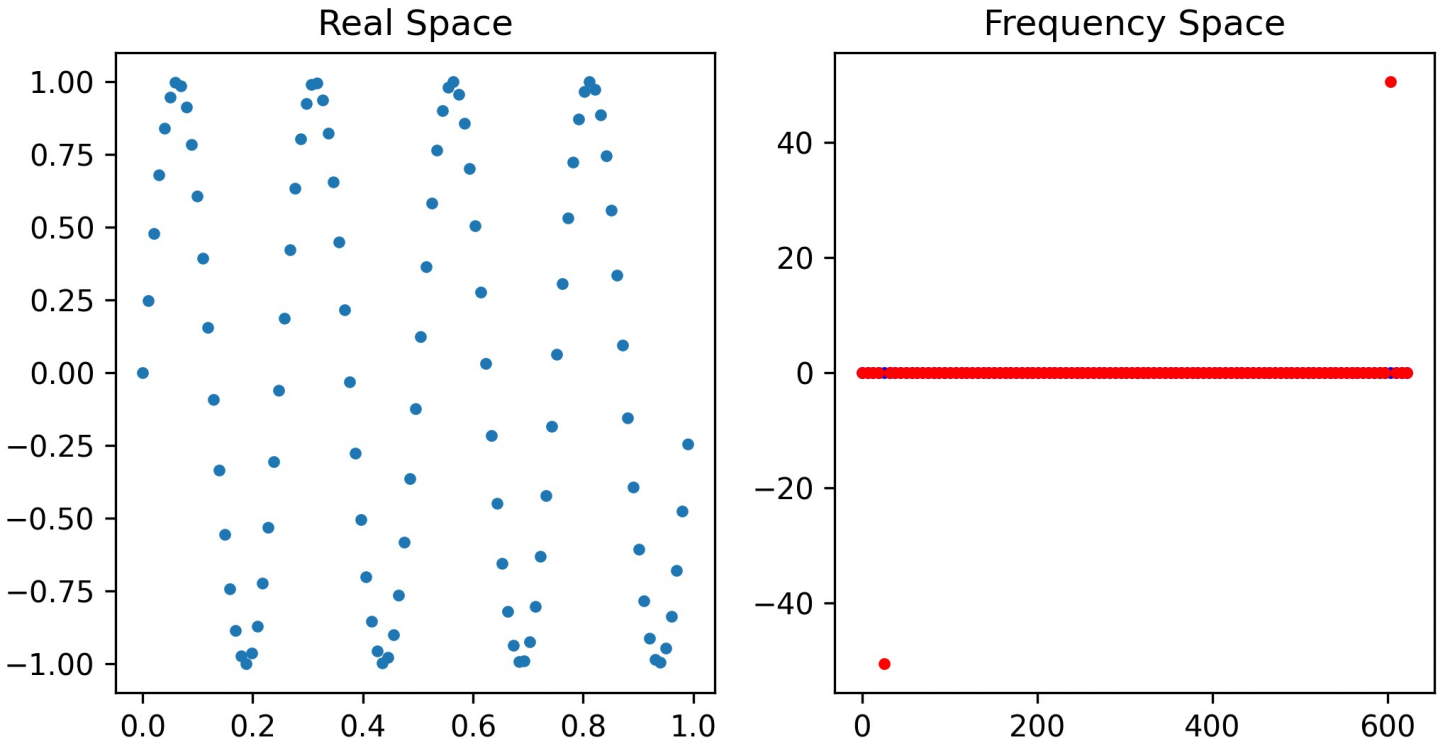
We discussed aliasing (in image space) last class.



A Fourier Transform Example (Code)

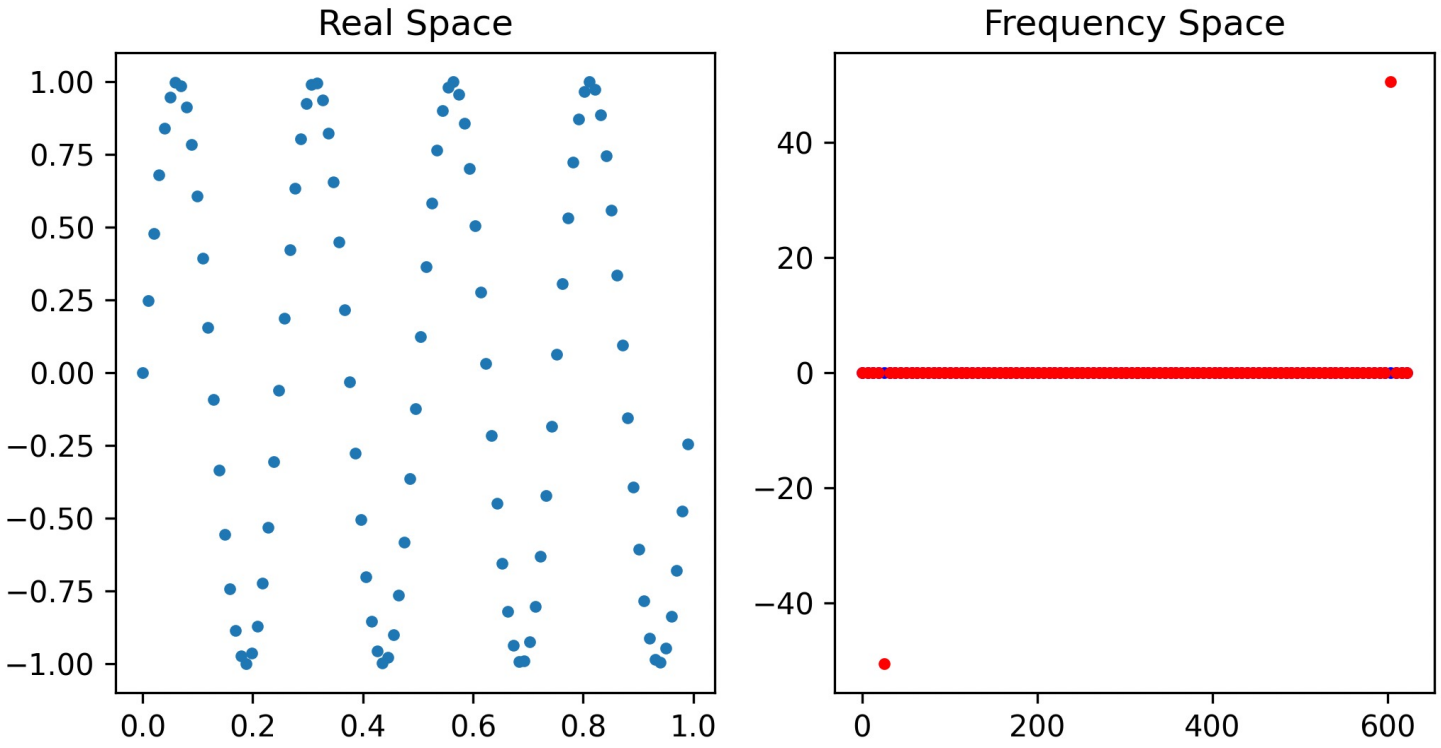
```
L = 1.0
N = 101
x = np.arange(0, L, L / N)
signal_x = np.sin(2 * math.pi * 4 * x)
signal_w = np.fft.fft(signal_x)
```

A Fourier Transform Example



Question: Why are there two non-zero points?

A Fourier Transform Example



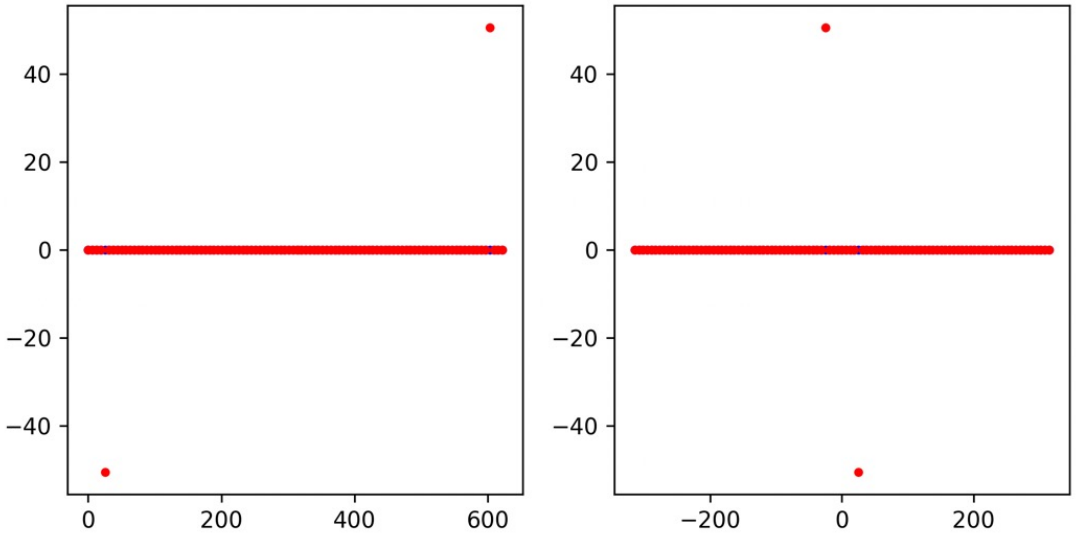
Question: Why are there two non-zero points? $\sin(-\theta) = -\sin(\theta)$

The signal is represented as a sum of the two possible sinusoids:

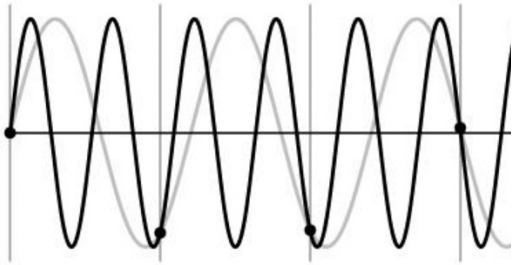
$$\sin(\theta) = 0.5 * [\sin(\theta) - \sin(-\theta)]$$

A Fourier Transform Example

Is there a difference between these two frequency-space signals?

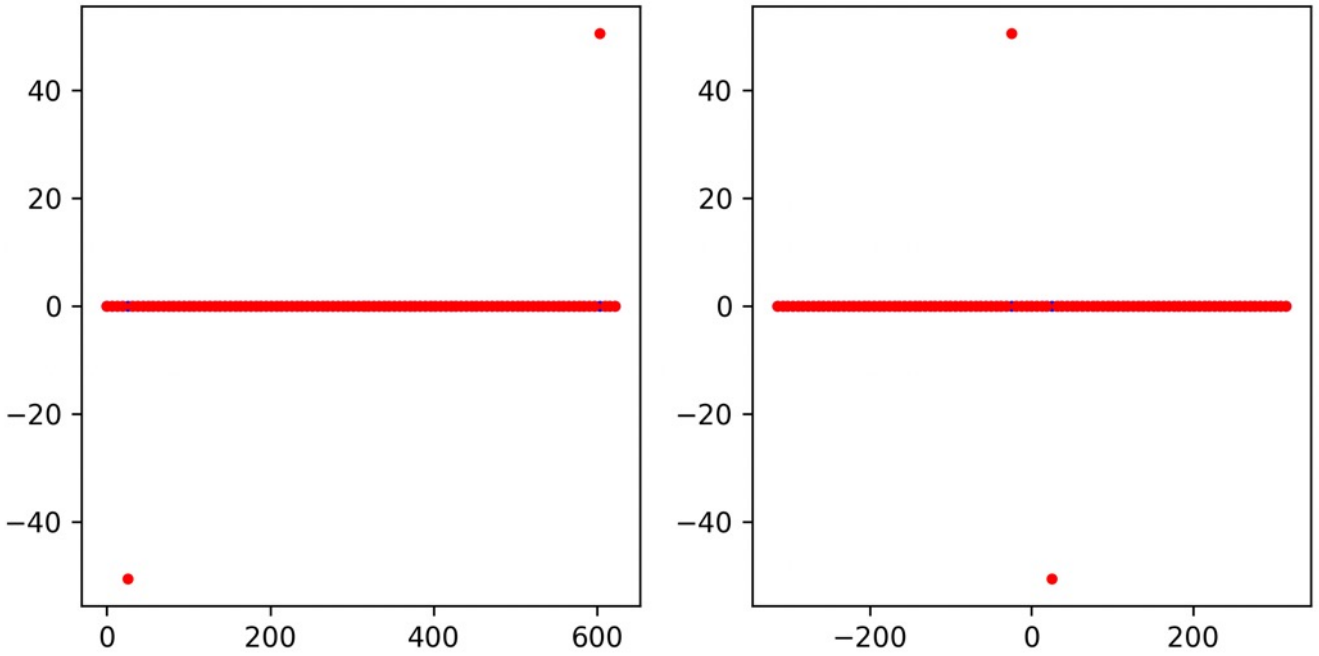


Remember:



A Fourier Transform Example

Is there a difference between these two frequency-space signals?



According to the Nyquist Limit: we can't tell the difference! However, practically, there is a difference.

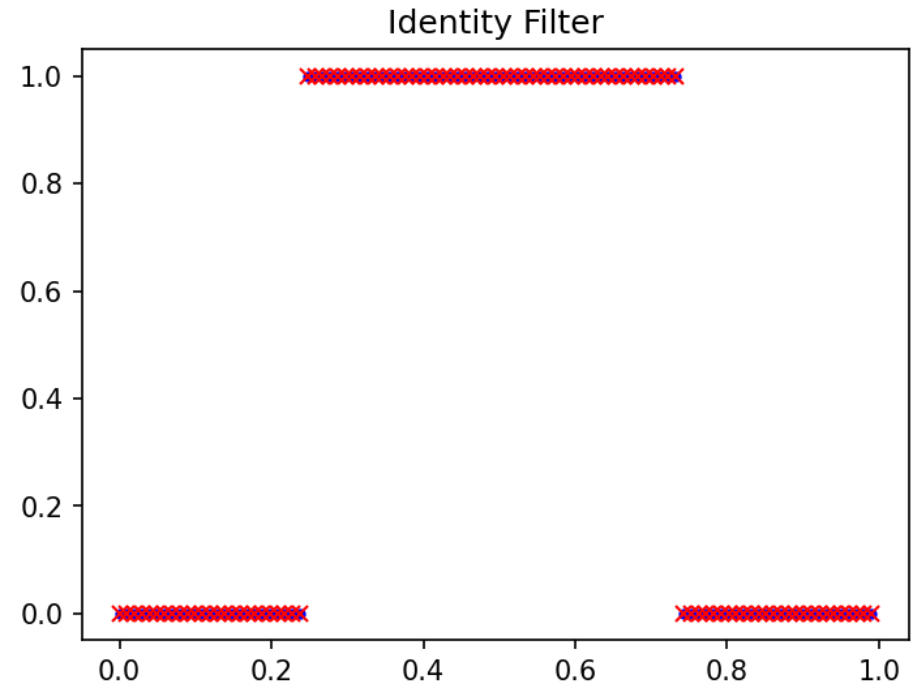
Breakout Session L03

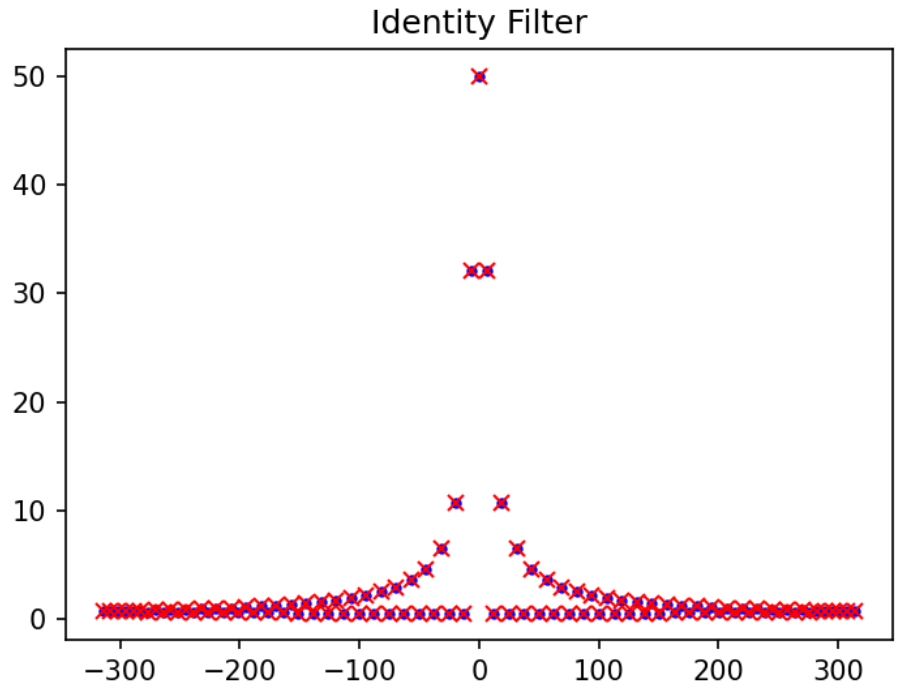
```
L = 1.0
N = 101
x = np.arange(0, L, L / N)
signal_x = np.sin(2 * math.pi * 4 * x)
signal_w = np.fft.fft(signal_x)
```

Let's run this example together

More Fourier Transform Examples

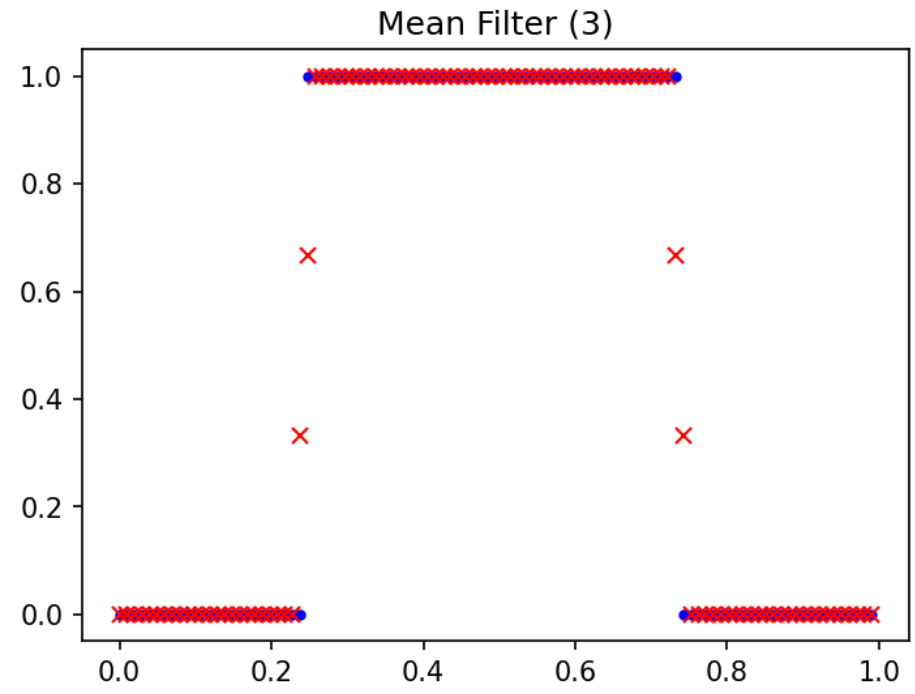
Let's generate a signal, and transform it into frequency space:

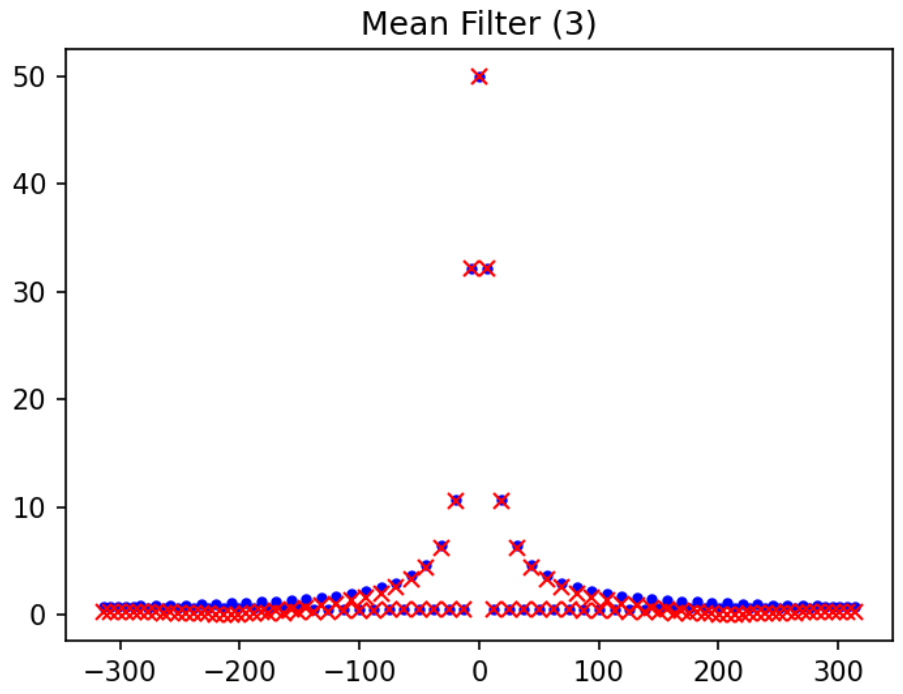




More Fourier Transform Examples

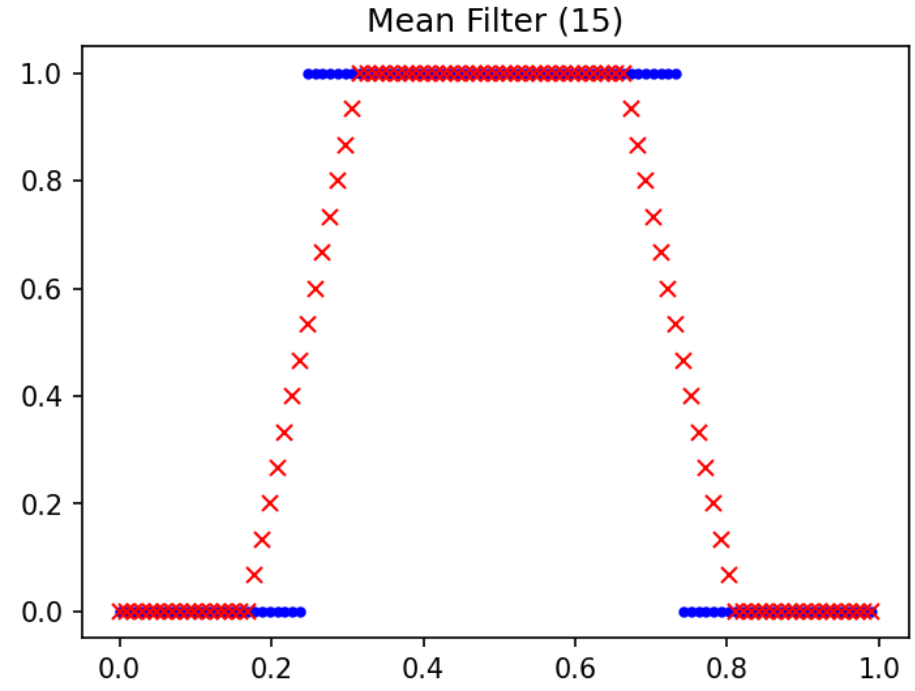
Let's generate a signal, and transform it into frequency space:

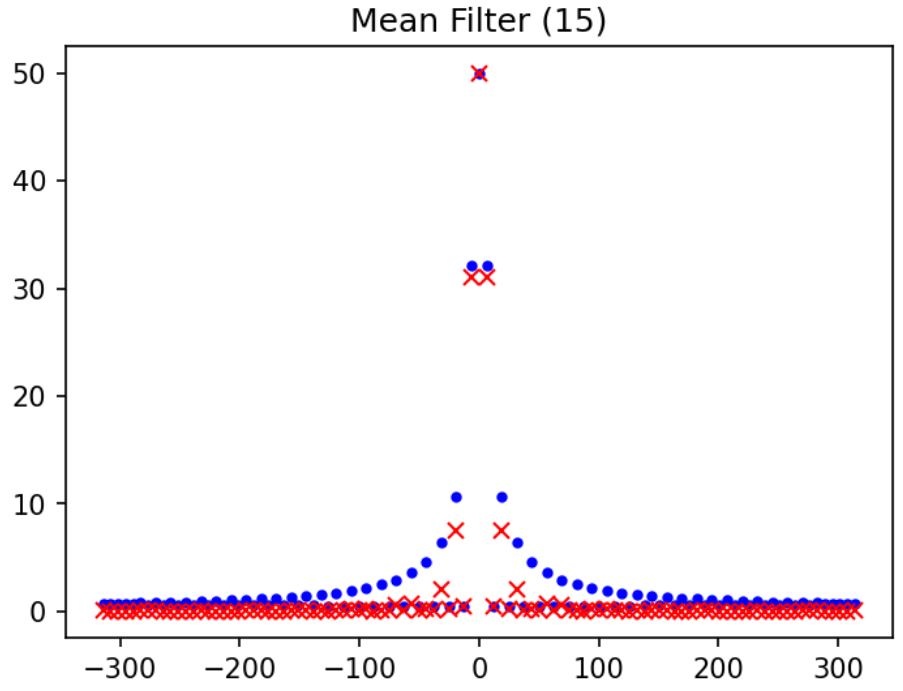




More Fourier Transform Examples

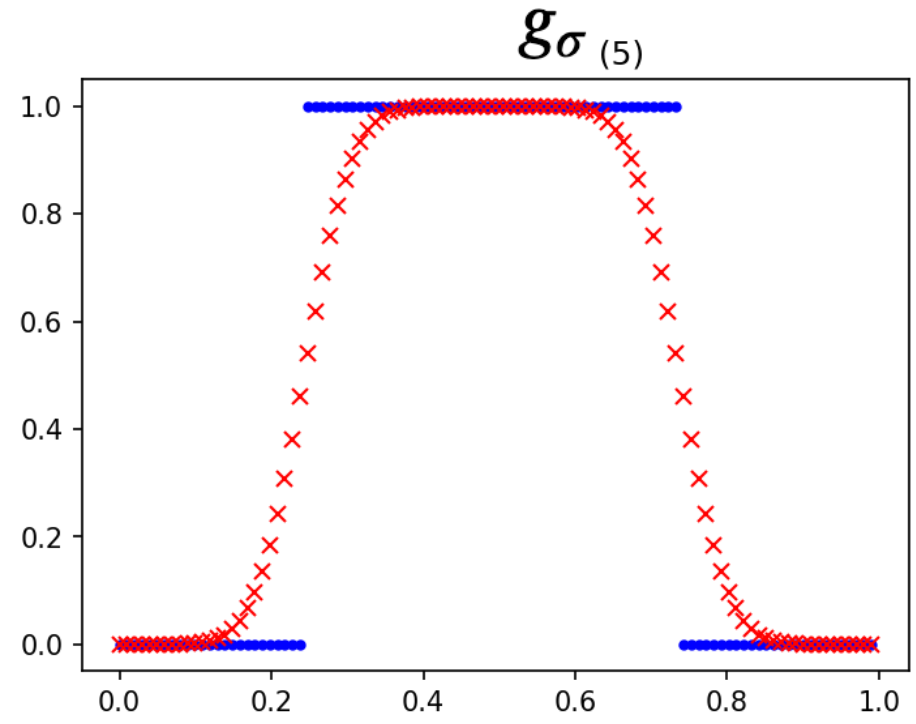
Let's generate a signal, and transform it into frequency space:

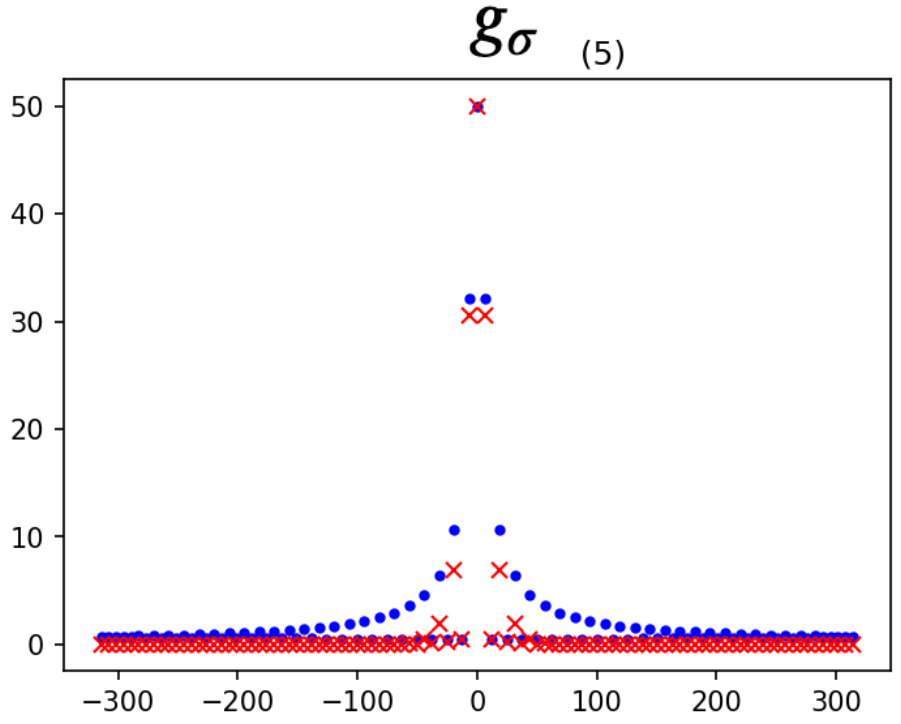




More Fourier Transform Examples

Let's generate a signal, and transform it into frequency space:

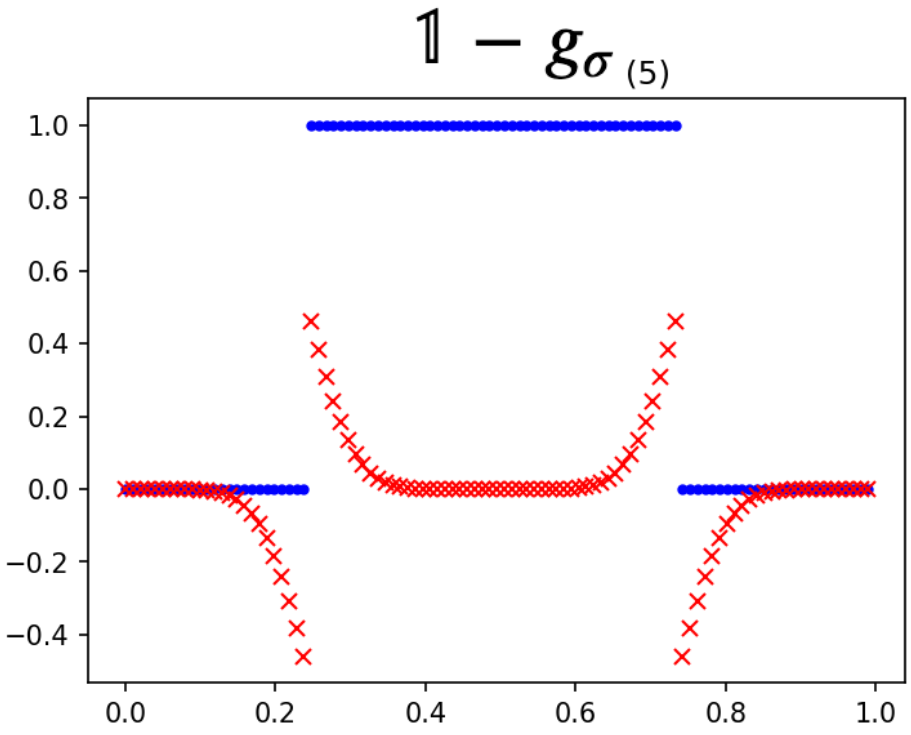


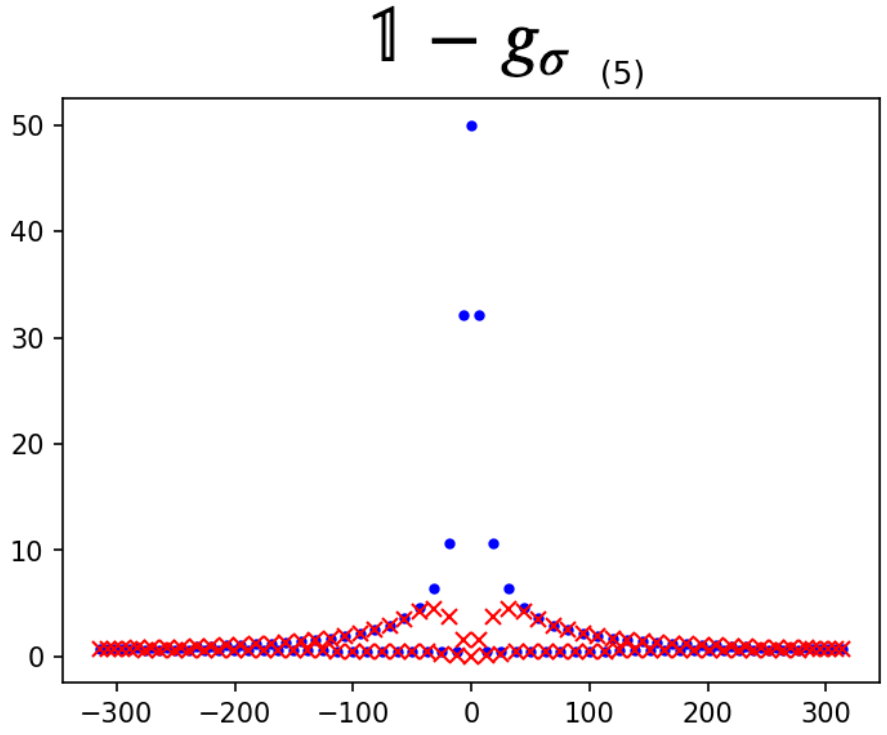


Gaussian filter example: we convolve the signal with a gaussian

More Fourier Transform Examples

Let's generate a signal, and transform it into frequency space:

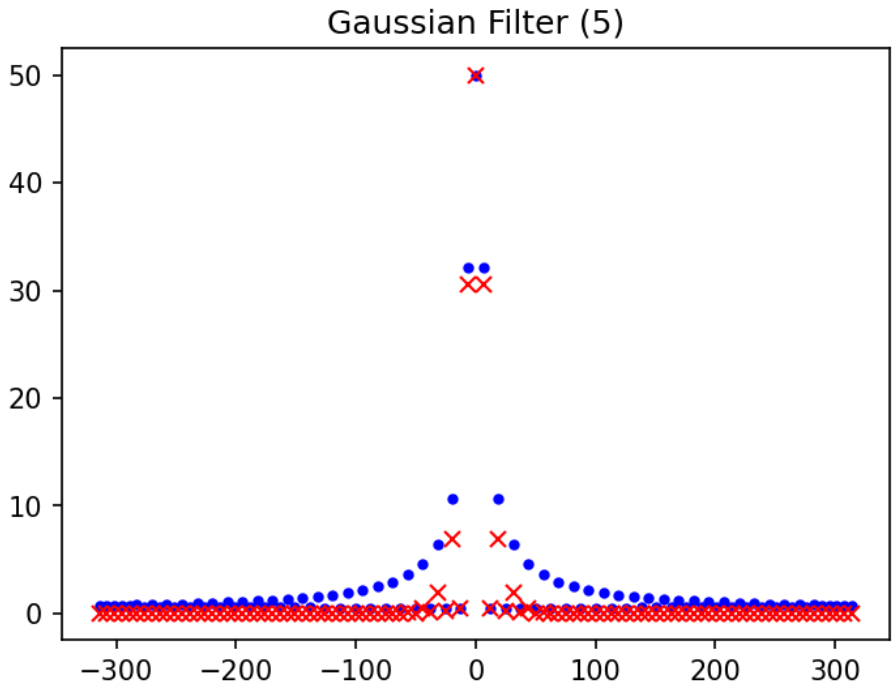


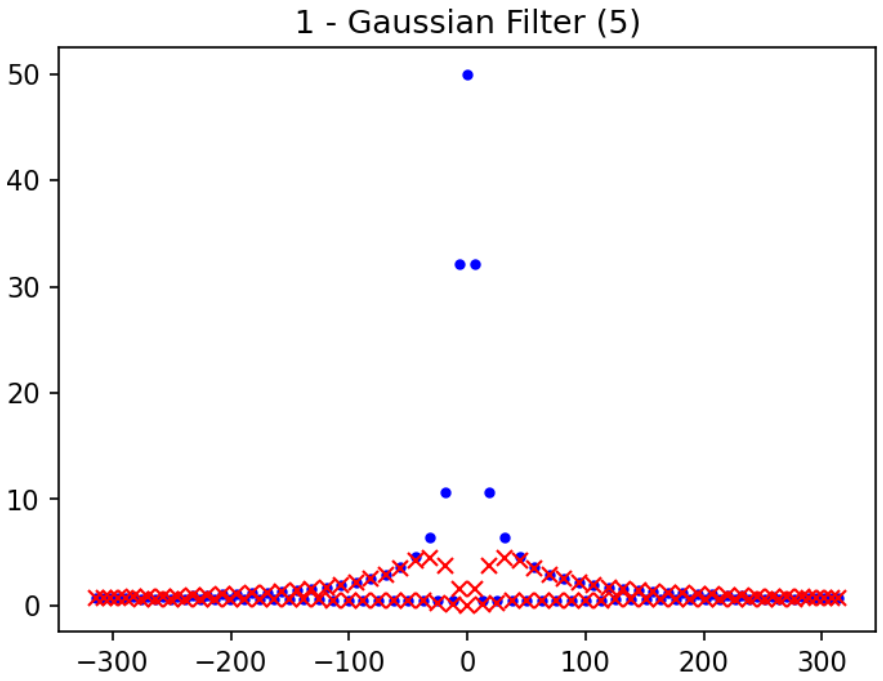


Original image minus Gaussian filtered image: we convolve the signal with a gaussian

More Fourier Transform Examples

Let's generate a signal, and transform it into frequency space:





Fourier Transforms and Convolutions

So why are we talking about Fourier Transforms at all…?

Fourier Transforms and Convolutions

So why are we talking about Fourier Transforms at all…?

There is a deep connection between the Fourier Transform and filtering. The Fourier Transform of a convolution has an interesting property:

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

i.e. the Fourier Transform of a convolution of two functions is equivalent to multiplying their frequency-space representations.

Fourier Transforms and Convolutions

So why are we talking about Fourier Transforms at all…?

There is a deep connection between the Fourier Transform and filtering. The Fourier Transform of a convolution has an interesting property:

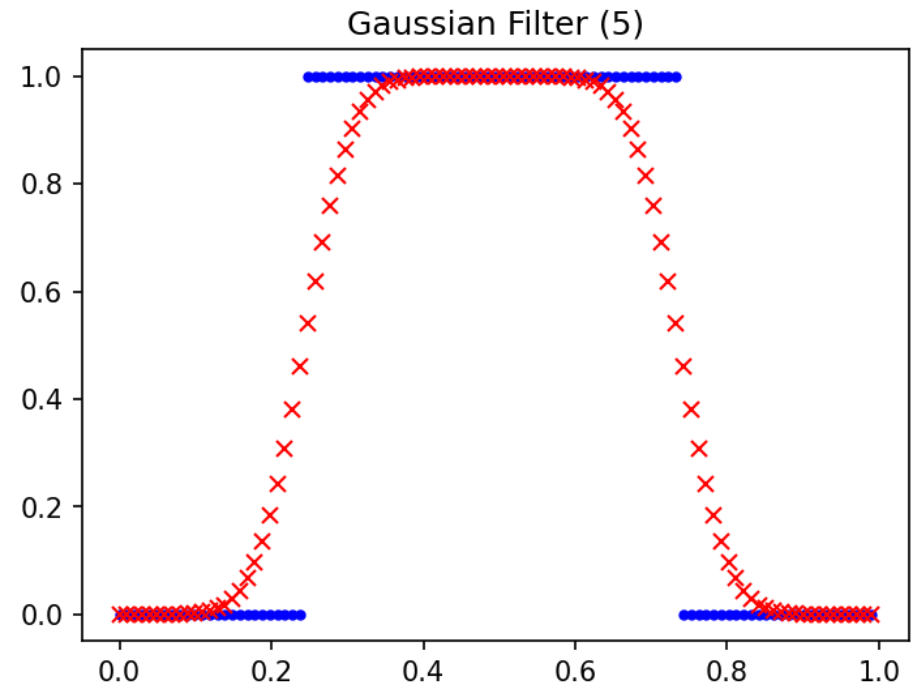
$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

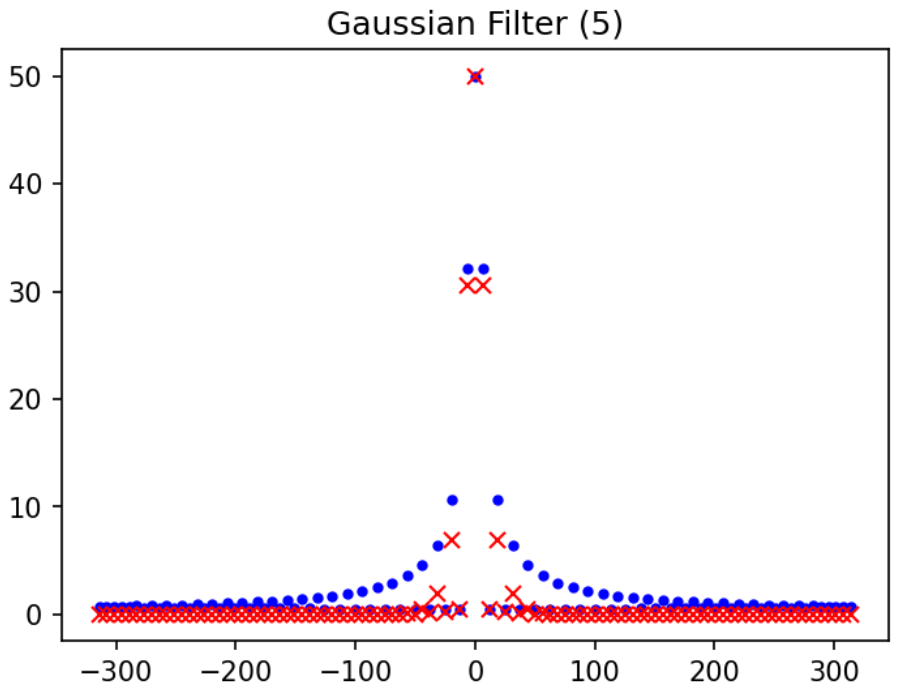
i.e. the Fourier Transform of a convolution of two functions is equivalent to multiplying their frequency-space representations.

We can use this to apply (and better understand) our filters.

More Fourier Transform Examples

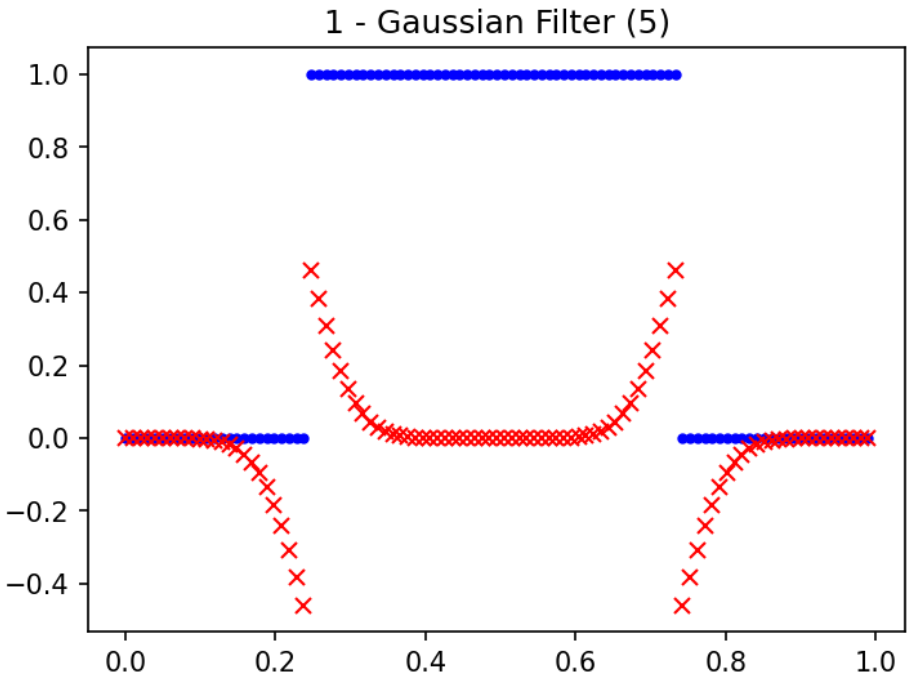
Let's generate a signal, and transform it into frequency space:

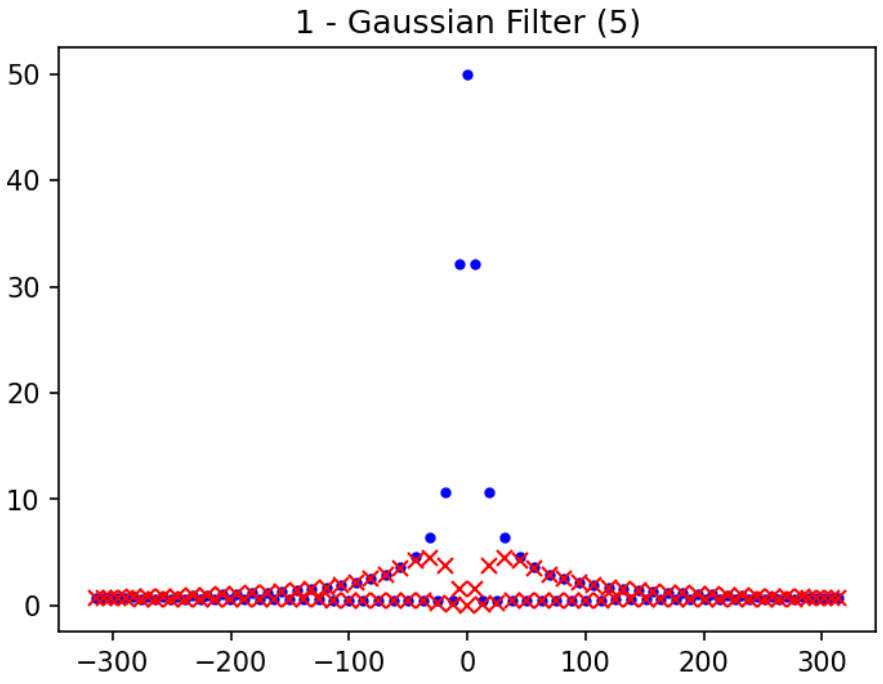




More Fourier Transform Examples

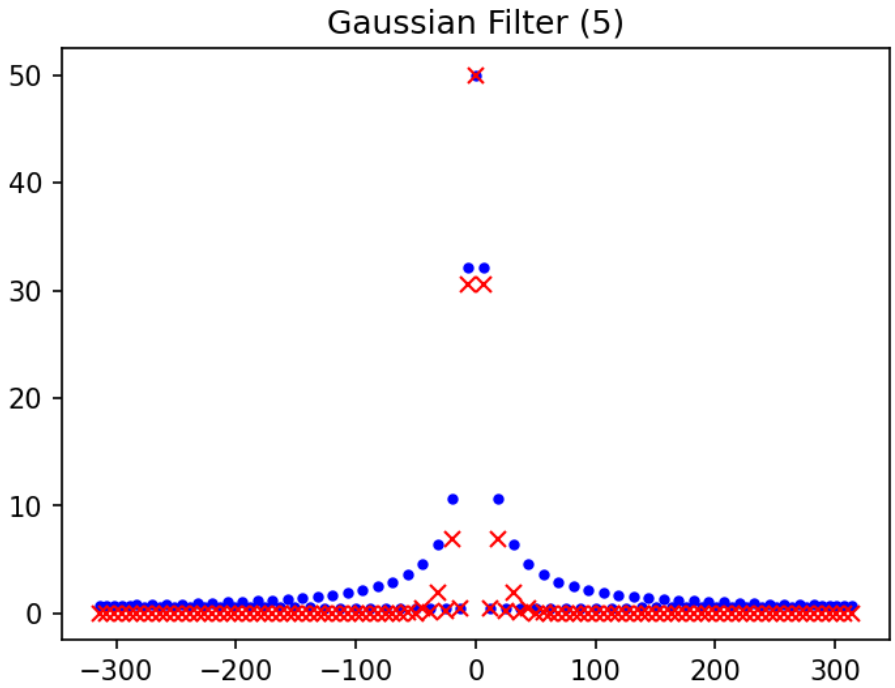
Let's generate a signal, and transform it into frequency space:

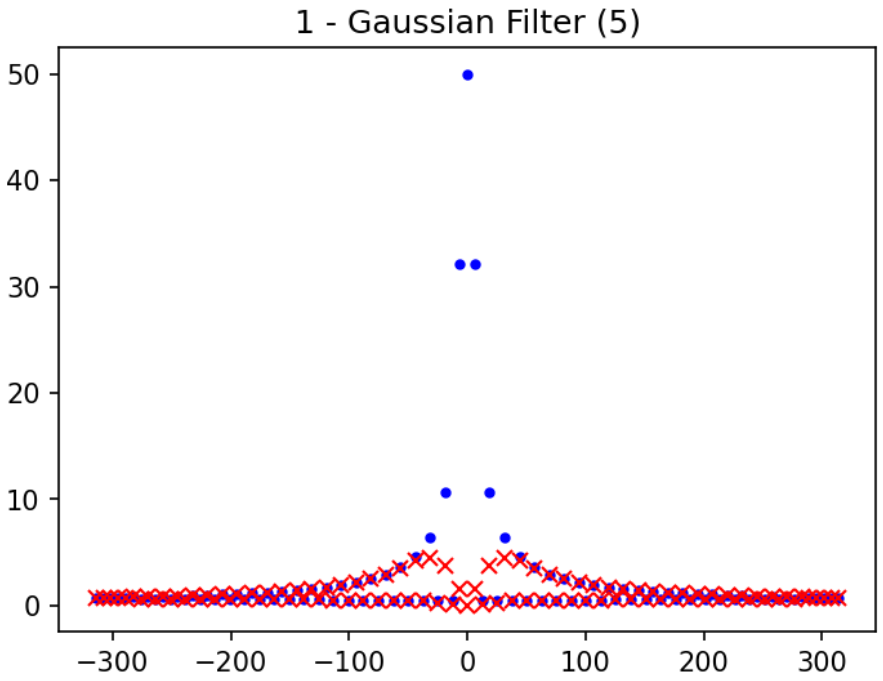




More Fourier Transform Examples

Let's generate a signal, and transform it into frequency space:





Fourier Transform of Some Common Functions

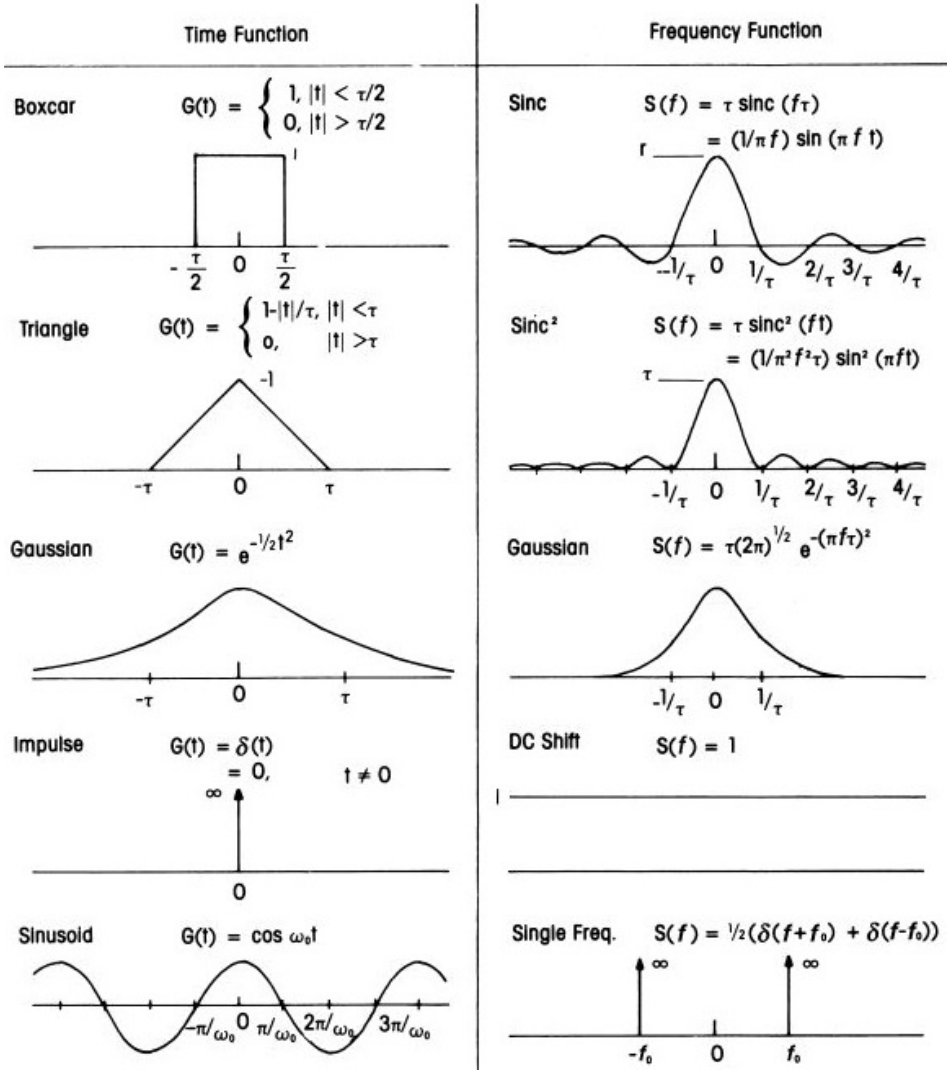
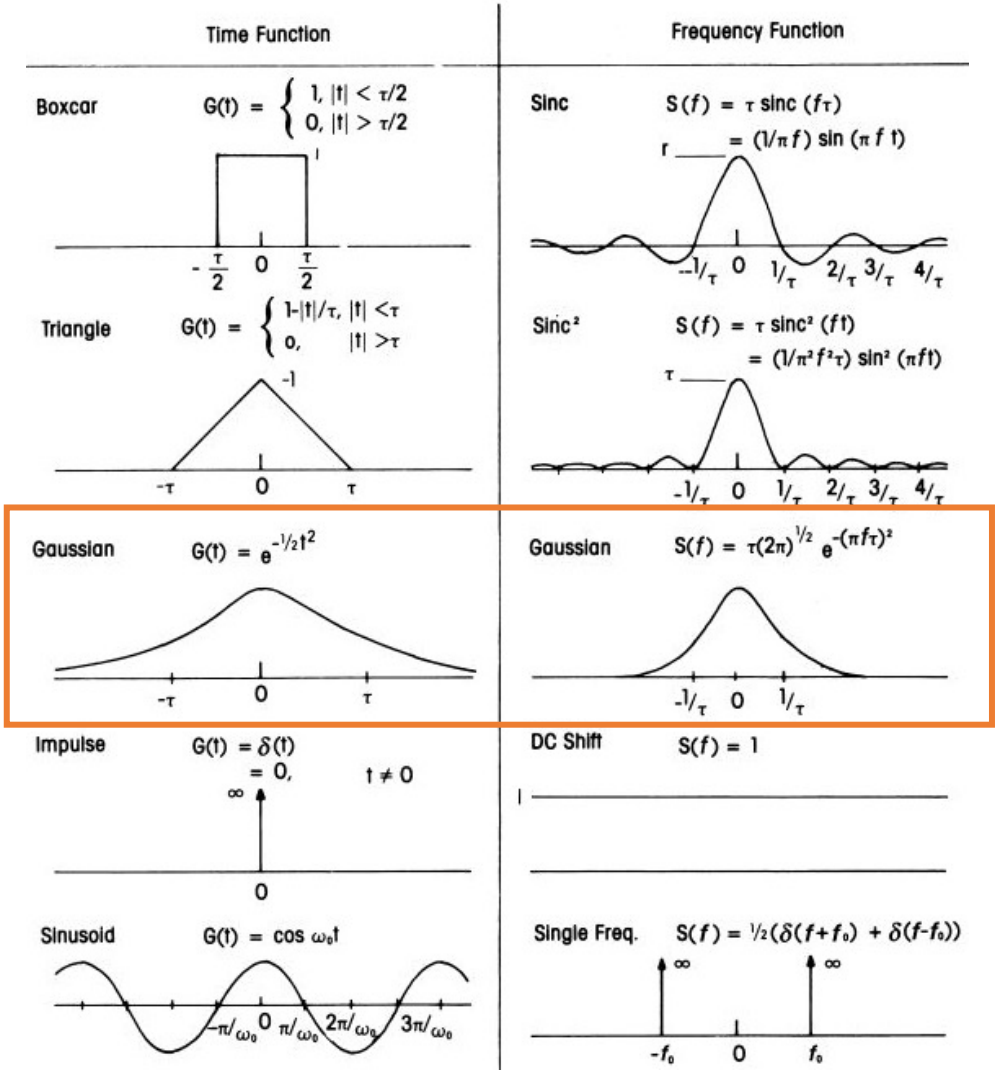


Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

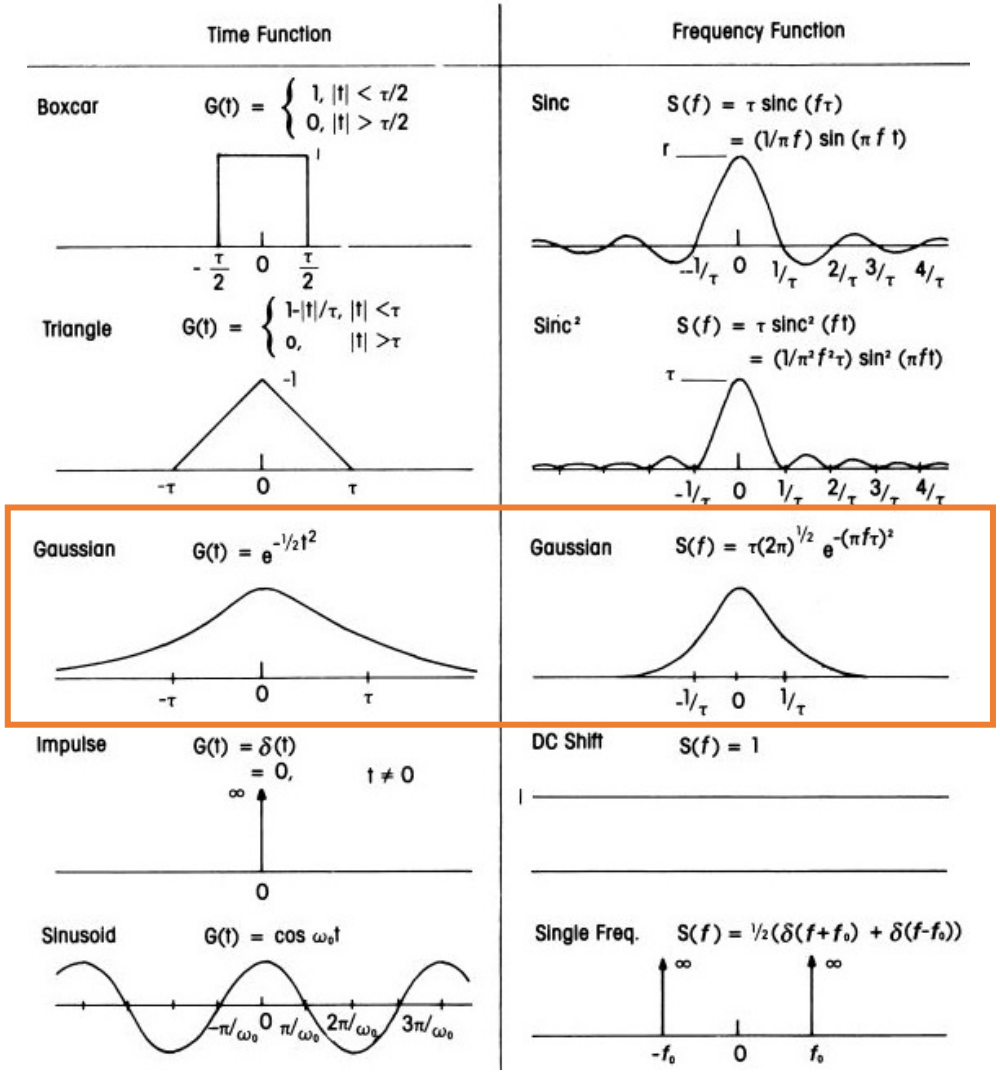
Fourier Transform of Some Common Functions



The transform of a Gaussian is another Gaussian.

Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

Fourier Transform of Some Common Functions

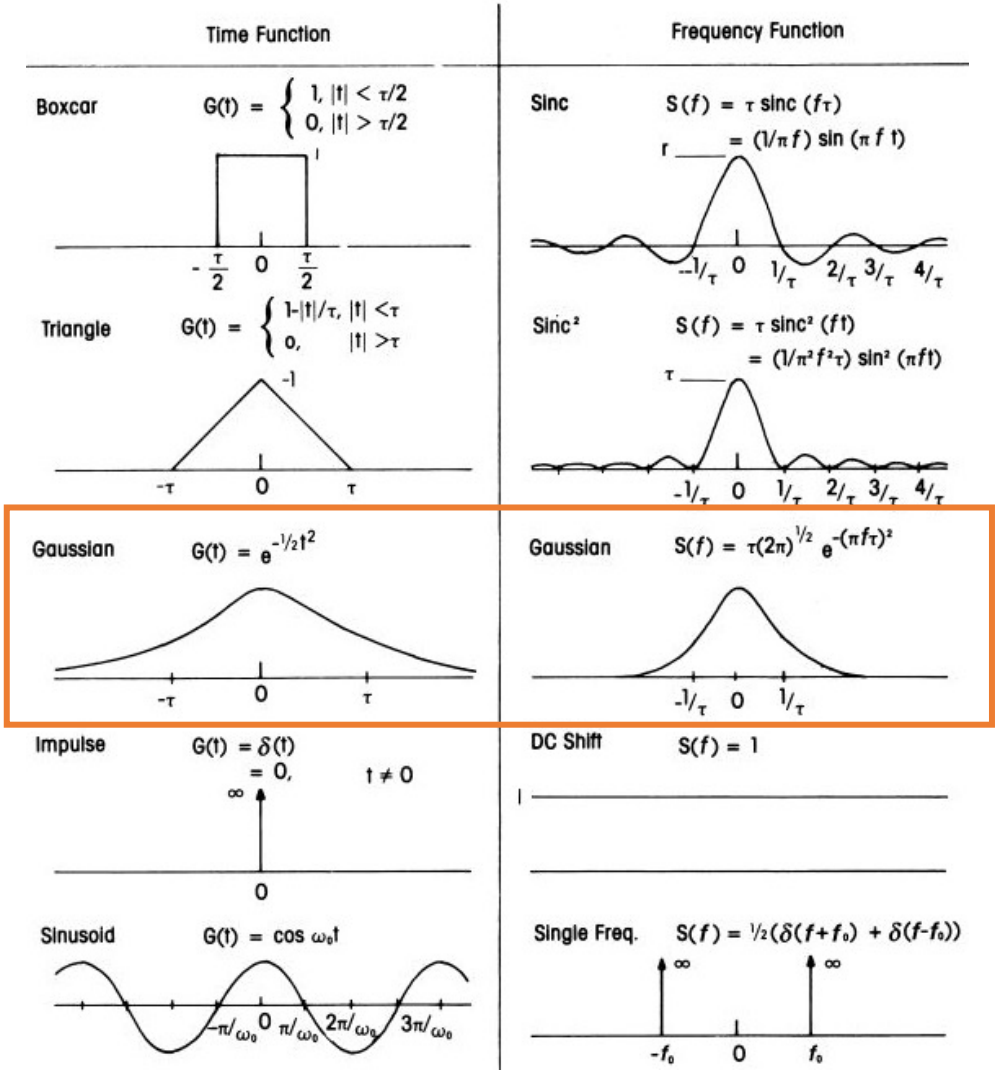


The transform of a Gaussian is another Gaussian.

The wider the gaussian in "normal space" (time or Cartesian) the narrower it is in frequency space.

Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

Fourier Transform of Some Common Functions



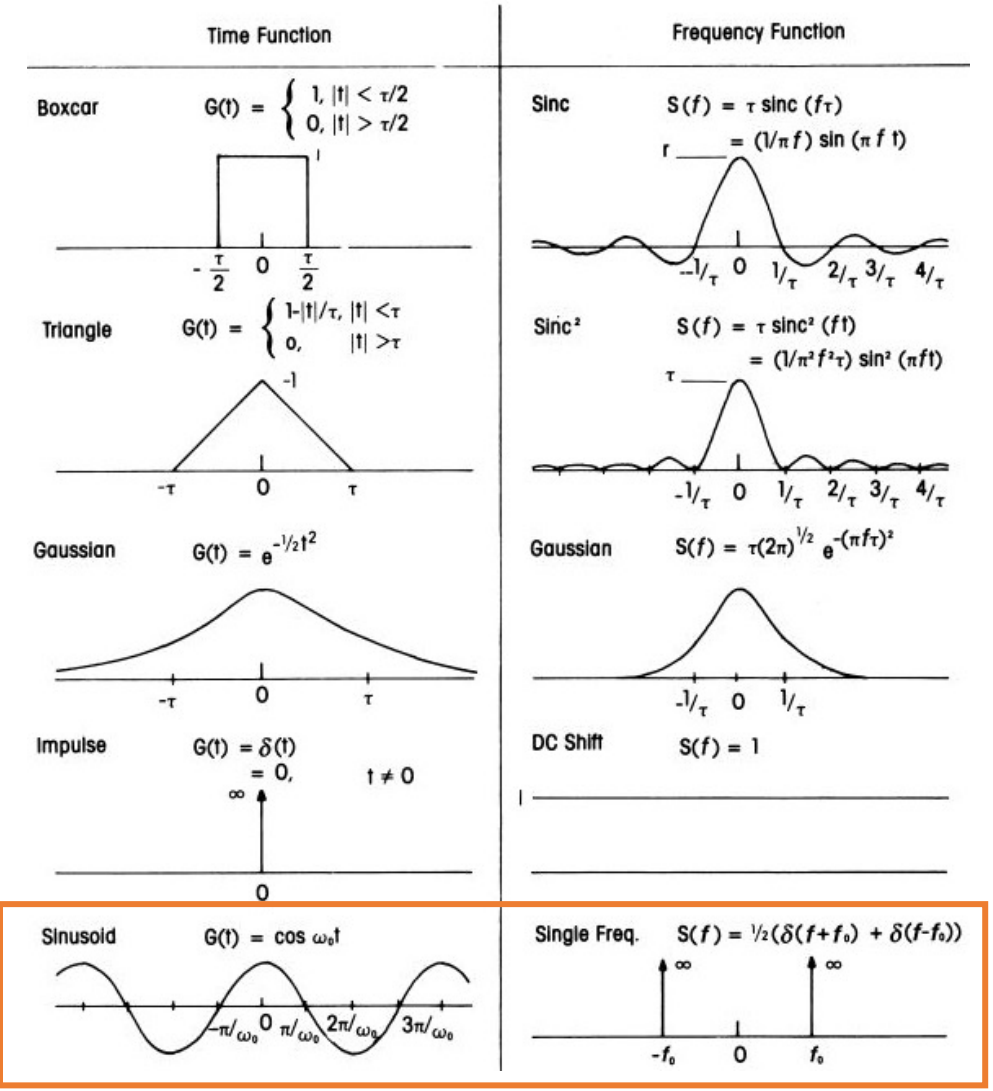
The transform of a Gaussian is another Gaussian.

The wider the gaussian in "normal space" (time or Cartesian) the narrower it is in frequency space.

More blur -> fewer high-frequency components.

Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

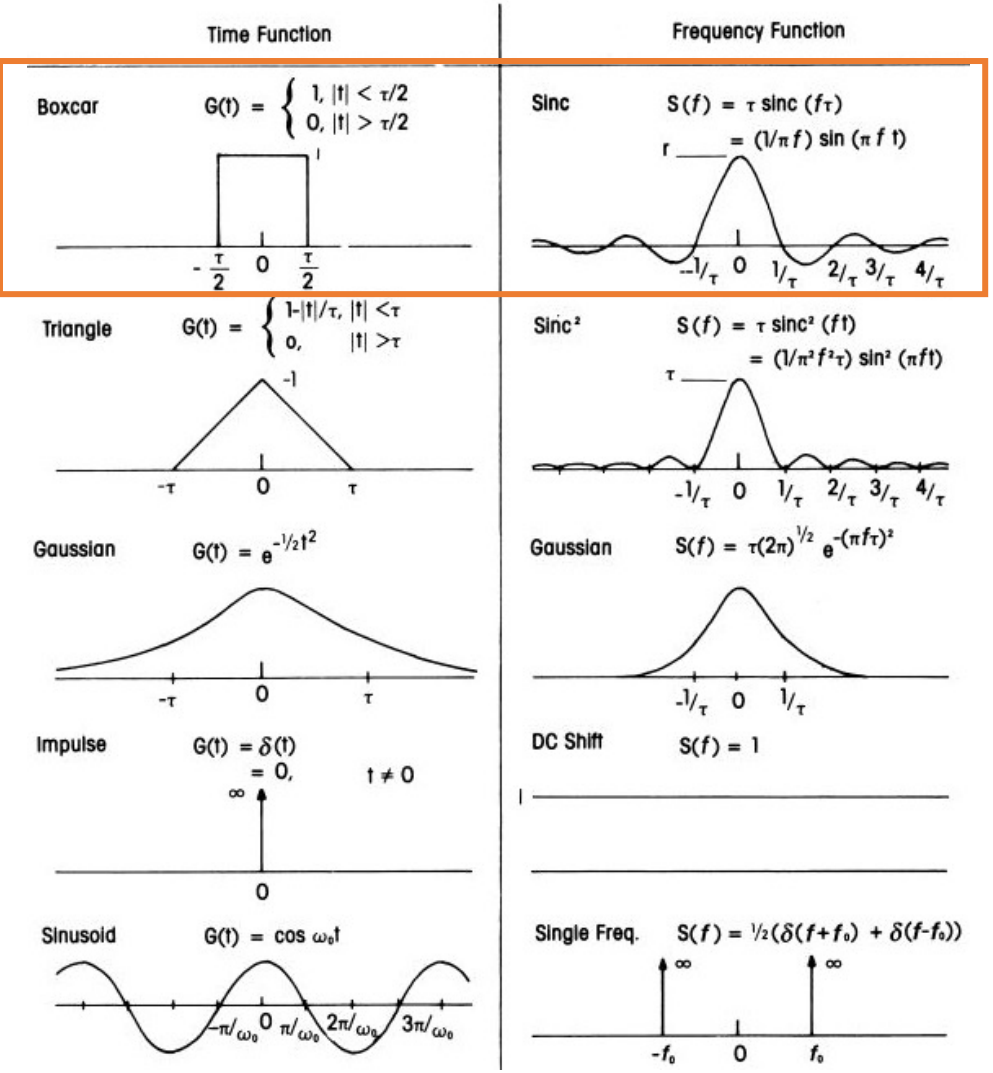
Fourier Transform of Some Common Functions



We've already talked about sine and cosine

Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

Fourier Transform of Some Common Functions



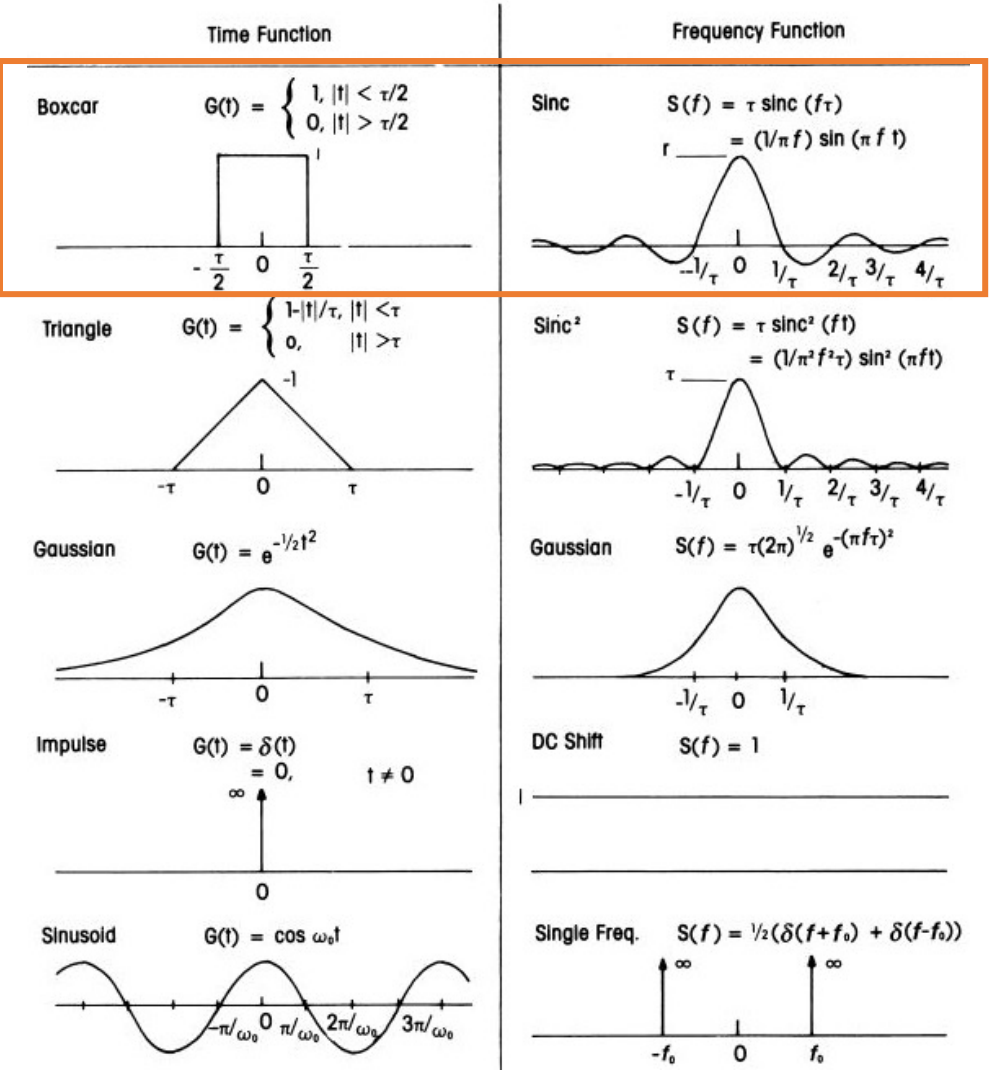
A "boxcar" becomes a "sinc" function:

$$\text{sinc}(x) = \frac{\sin(x)}{x}$$

(Note that sinc(0) = 1)

Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

Fourier Transform of Some Common Functions



A "boxcar" becomes a "sinc" function:

$$\text{sinc}(x) = \frac{\sin(x)}{x}$$

(Note that sinc(0) = 1)

Multiplying by the boxcar in time (or frequency) space is equivalent to a convolution in frequency (or time) space. What can we do with this?

Image from https://wiki.seg.org/images/2/2c/Segf19.jpg

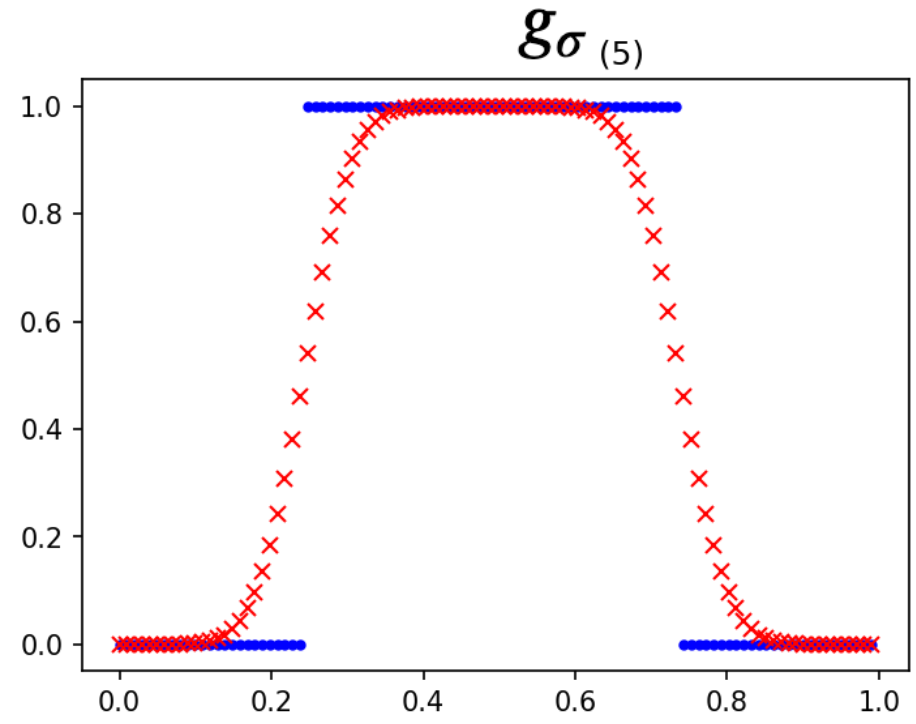
Low-Pass Filtering

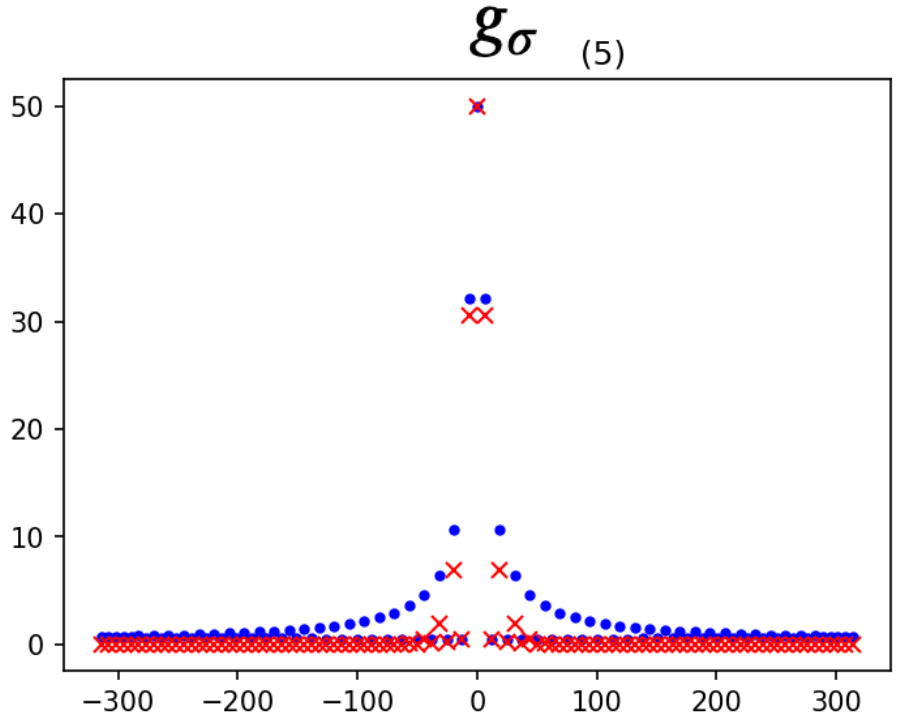
To low-pass filter (using Fourier Transforms), we can transform a signal to frequency space, multiply by the "boxcar" (rectangle) function, and convert back.

This is equivalent to convolving with the sinc function (a relationship you will be asked to think about in this week's quiz).

Low-Pass Filtering via Smoothing

Let's generate a signal, and transform it into frequency space:





Gaussian filter example: we convolve the signal with a gaussian
(An example of a low-pass filter. Why?)

So what's a high-pass filter?

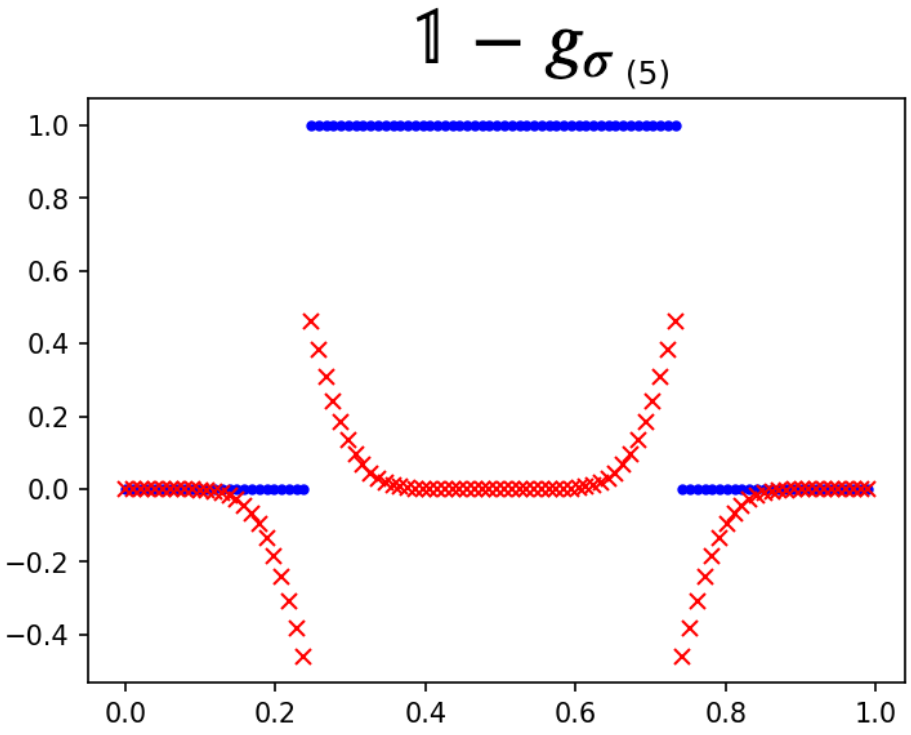
So what's a high-pass filter?

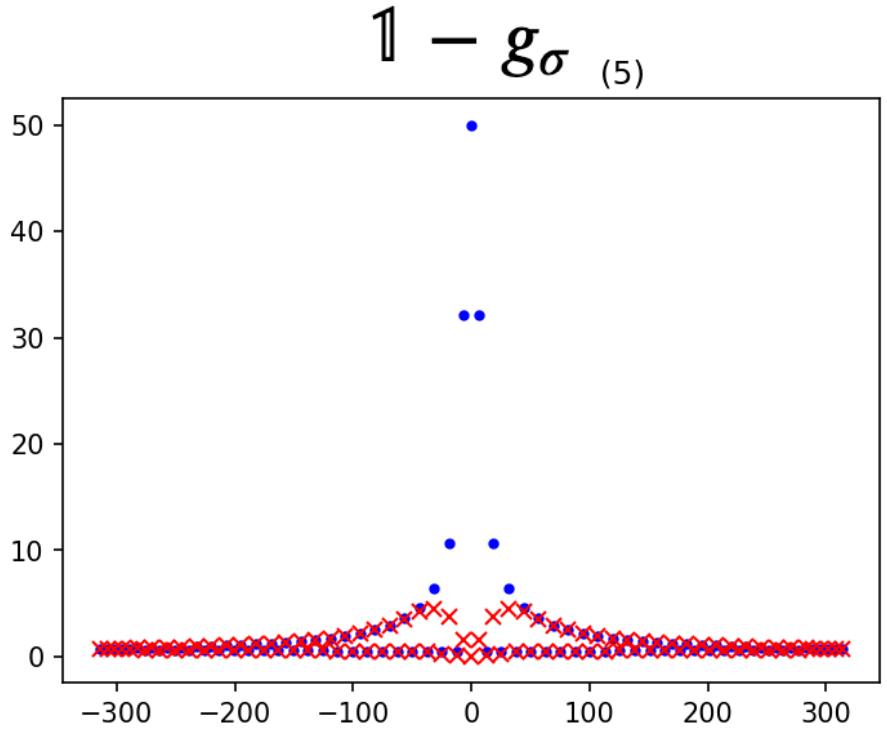
Remember: Fourier Transforms are linear.

```
high-pass = identity - low-pass
```

More Fourier Transform Examples

Let's generate a signal, and transform it into frequency space:





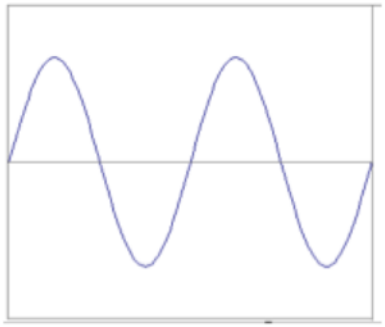
Original image minus Gaussian filtered image: we convolve the signal with a gaussian
(An example of a high-pass filter. Why?)

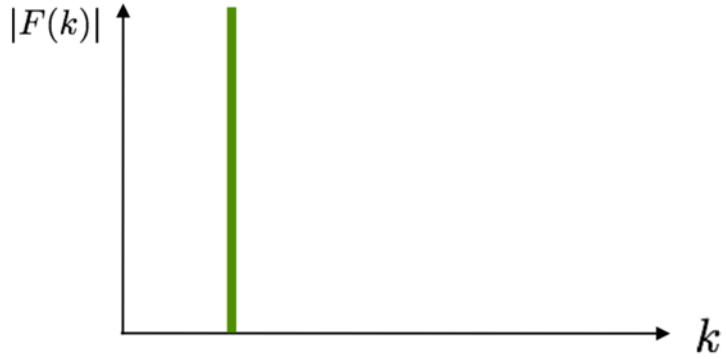
Finally, what do Fourier Transforms of images look like?

Spatial domain visualization

Frequency domain visualization

1D

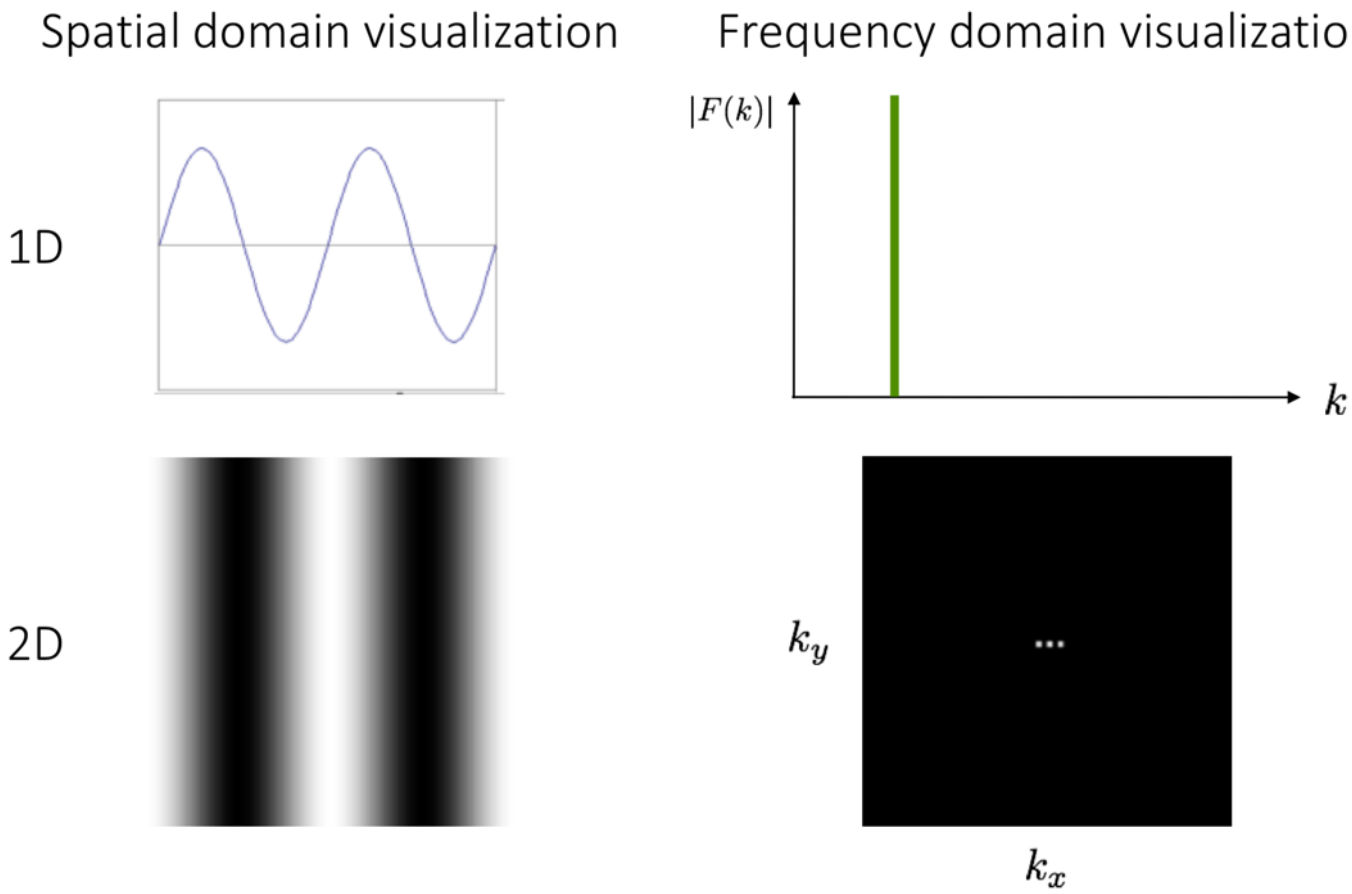




2D

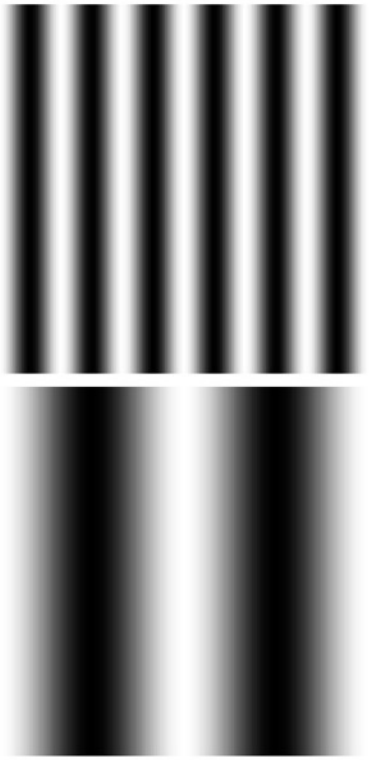
?

Finally, what do Fourier Transforms of images look like?



Finally, what do Fourier Transforms of images look like?

Spatial domain visualization

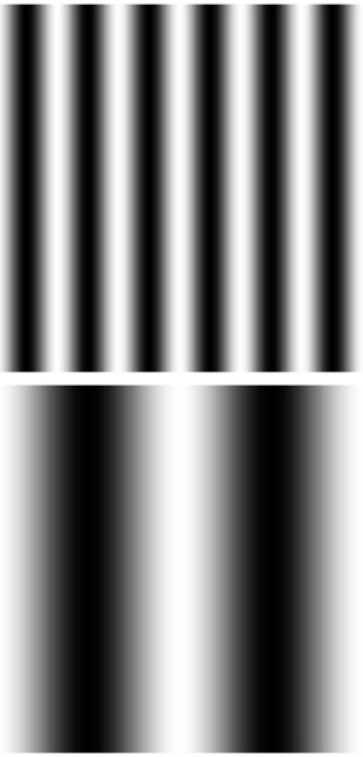


Frequency domain visualization

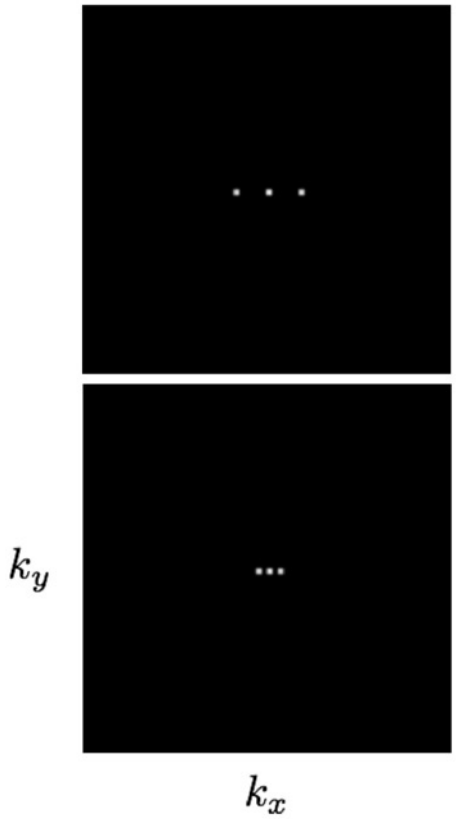
?

Finally, what do Fourier Transforms of images look like?

Spatial domain visualization



Frequency domain visualization



Hybrid Images

CS 482, Prof. Stein

Lecture 02.E

The Human Visual Perception system is quite fascinating

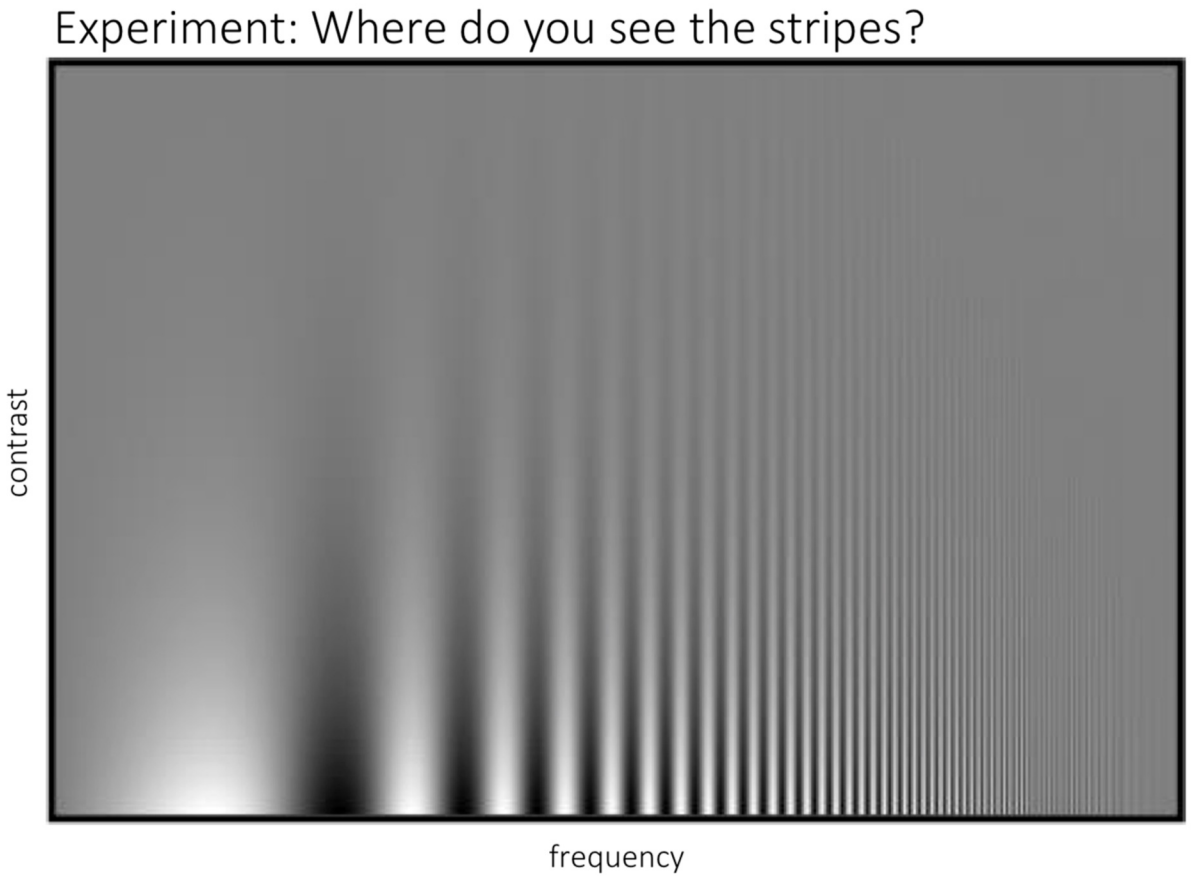
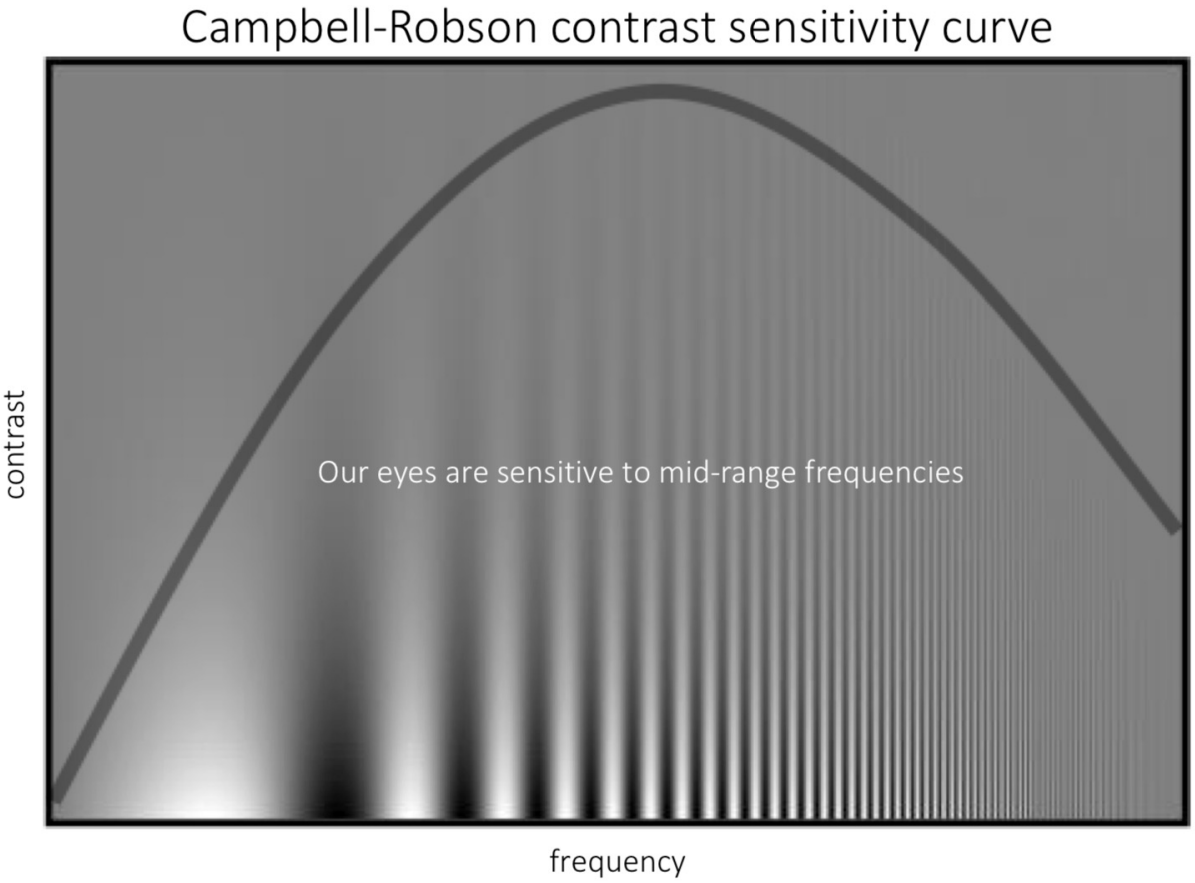


Image Credit

The Human Visual Perception system is quite fascinating



- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid frequencies dominate perception

Image Credit

Salvador Dalí
Skull of Zurbaran

https://dali.com/dalis-skull-zurbaran-just-might-perfect/

Salvador Dalí
Skull of Zurbaran

https://dali.com/dalis-skull-zurbaran-just-might-perfect/

Hybrid Images exploit our eye/brain sensitivity to certain frequencies

Paper Link: https://stanford.edu/class/ee367/reading/OlivaTorralb_Hybrid_Siggraph06.pdf

Hybrid images

Aude Oliva* MIT-BCS
Antonio Torralba† MIT-CSAIL
Philippe. G. Schyns‡ University of Glasgow

Figure 1: A hybrid image is a picture that combines the low-spatial frequencies of one picture with the high spatial frequencies of another picture producing an image with an interpretation that changes with viewing distance. In this figure, the people may appear sad, up close, but step back a few meters and look at the expressions again.

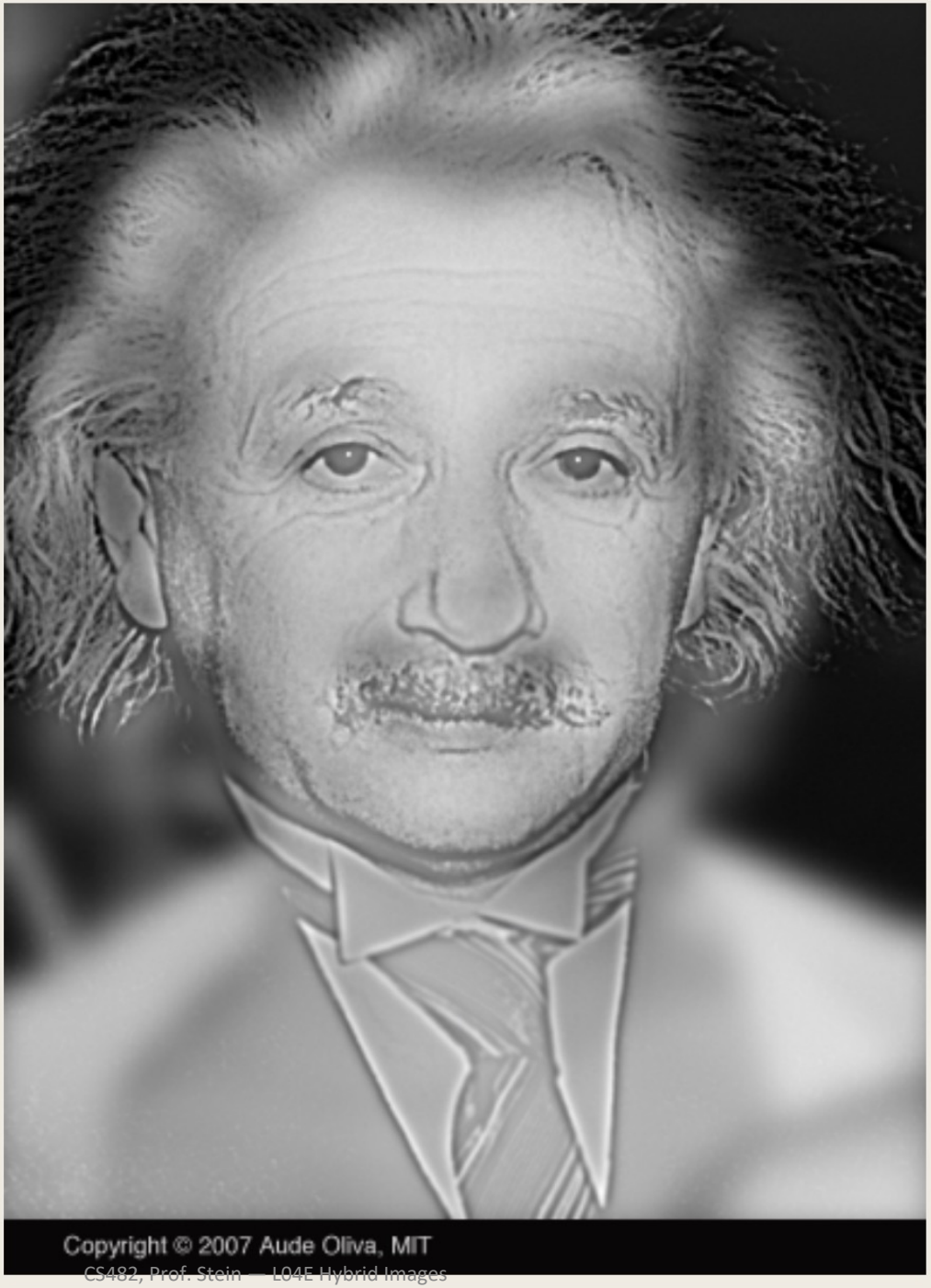
Abstract

in which the faces displayed different emotions. High spatial frequencies correspond to faces with "sad" expressions. Low spatial

Hybrid Images exploit our eye/brain sensitivity to certain frequencies

Paper Link: https://stanford.edu/class/ee367/reading/OlivaTorralb_Hybrid_Siggraph06.pdf

Marylin
Einstein



In your first Programming Assignment, you will make your own simplified Hybrid Image

The process is simple, but requires some experimentation.

1. Find some images you think will "work" (and align them if necessary in photo-editing software)
2. Take the high-pass version of one image and add that the the low-pass version of the other.

Note: you are free to use the "gaussian" version of the low- and high-pass filters. You do not need to use Fourier Transforms!

Other useful properties of Fourier Transforms

Upsampling can be done using Fourier Transforms if you'd like.

Procedure:

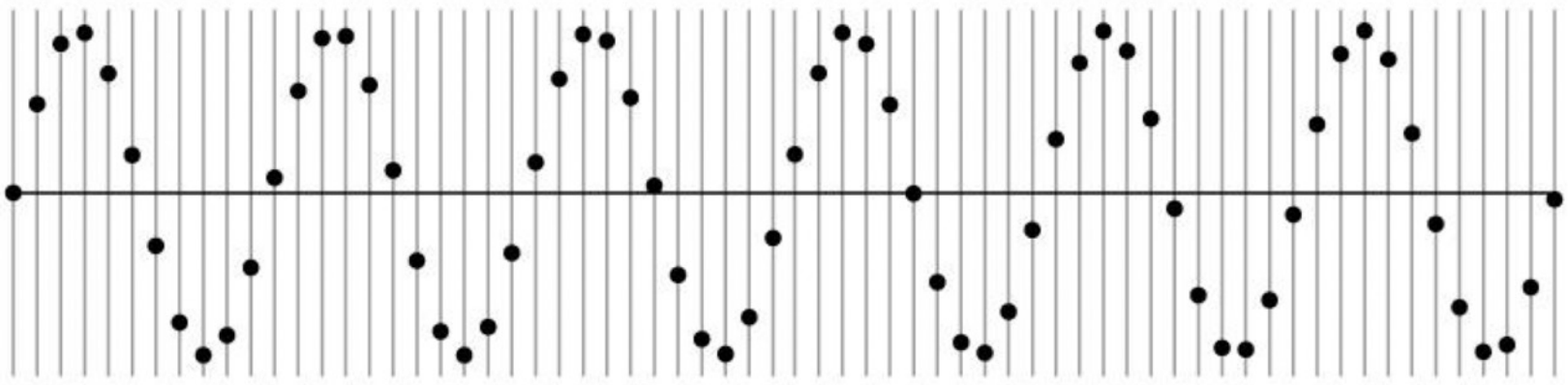
1. Convert the image to frequency space
2. ”Pad” the frequency-space image with zeros up to the desired size (be careful about normalization and the centering of the DFT frequency vector)
3. Transform back

In frequency space: it's as if we are adding higher frequencies

When we transform back we have more “spots to fill” in the time axis and must fill them in with the sine waves that made up the original signal.

Fourier Interpolation: a simple example

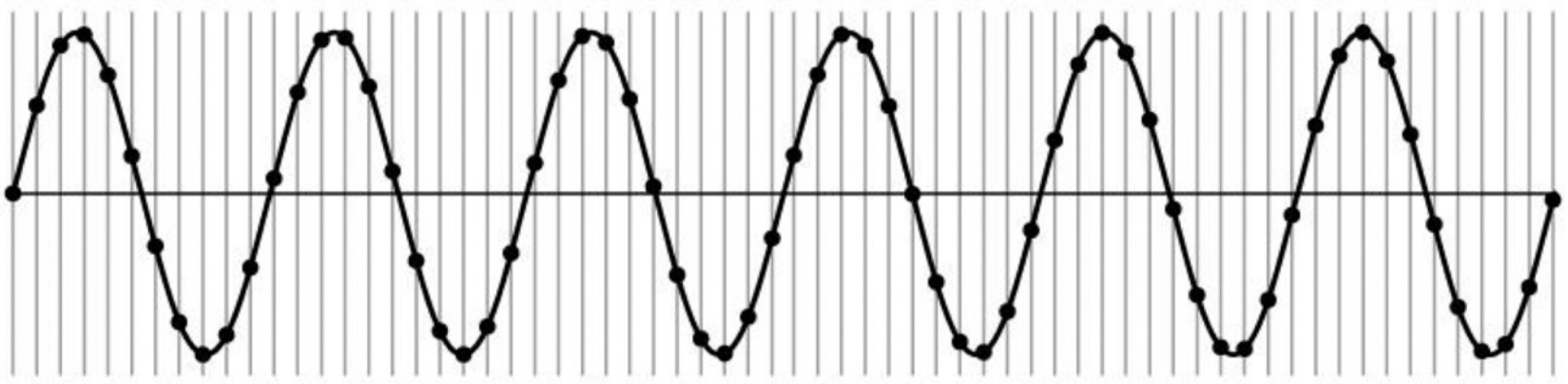
Earlier, we took some points and asked what the frequency-space representation would look like:



We notice that a single sine wave fits all the points perfectly, so we can use that sine wave to fill in the gaps.

Fourier Interpolation: a simple example

Earlier, we took some points and asked what the frequency-space representation would look like:



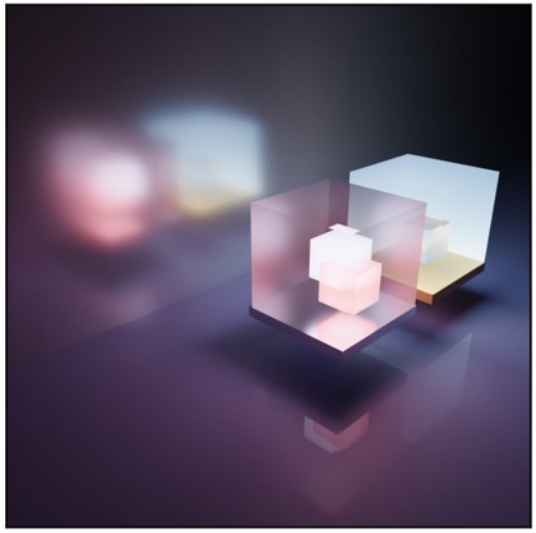
We notice that a single sine wave fits all the points perfectly, so we can use that sine wave to fill in the gaps.

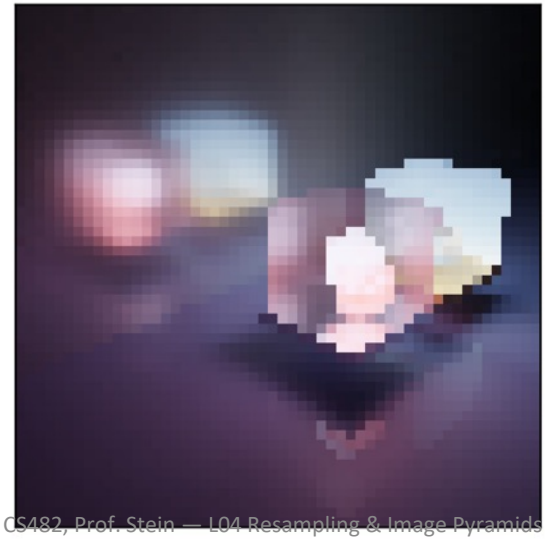
Fourier Transform interpolation generalizes this idea: we represent a signal by multiple sine waves and then use *those* to interpolate. The Fourier transform does all this for us.

Upsampling can be done using Fourier Transforms if you'd like.

Procedure:

1. Convert the image to frequency space
2. "Pad" the frequency-space image with zeros up to the desired size (be careful about normalization and the centering of the DFT frequency vector)
3. Transform back





We can also use Fourier Transforms to prove the Convolution Derivative Theorem

$$\mathcal{F}\left(\left(\frac{d}{dx}f\right)\star g\right)(x) = \left(\mathcal{F}\left(\frac{d}{dx}f\right)\right)\cdot(\mathcal{F}(g)(x)) = ix\mathcal{F}(f)(x)\cdot\mathcal{F}(g)(x).$$

Copied from MathOverflow, which has a few different ways of proving this theorem.

Fourier Transform Derivatives

Computing the derivative of a Fourier Transformed signal is relatively straightforward:

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{i2\pi kx/N}$$

This is another way to write the "frequency space" signal

Fourier Transform Derivatives

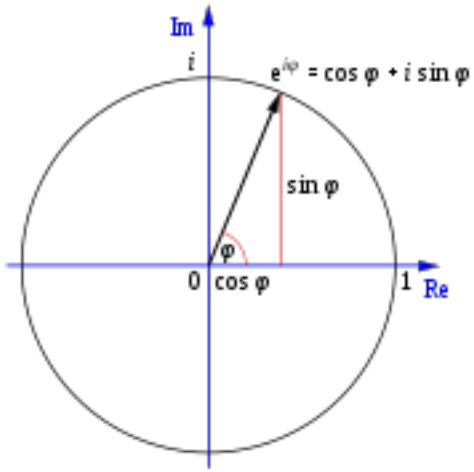
Computing the derivative of a Fourier Transformed signal is relatively straightforward:

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{i2\pi kx/N}$$

This is another way to write the "frequency space" signal

If you've never seen it, Euler's Identity relates exponentials and sinusoids:

$$\exp(i\theta) = \cos\theta + i\sin\theta$$



From Wikipedia: Euler's Identity

Fourier Transform Derivatives

Computing the derivative of a Fourier Transformed signal is relatively straightforward:

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{i2\pi kx/N}$$

$$\frac{d}{dx} f(x) = \sum_{k=0}^{N-1} \frac{i2\pi k}{N} F(k) e^{i2\pi kx/N}$$

Fourier Transform Derivatives

Computing the derivative of a Fourier Transformed signal is relatively straightforward:

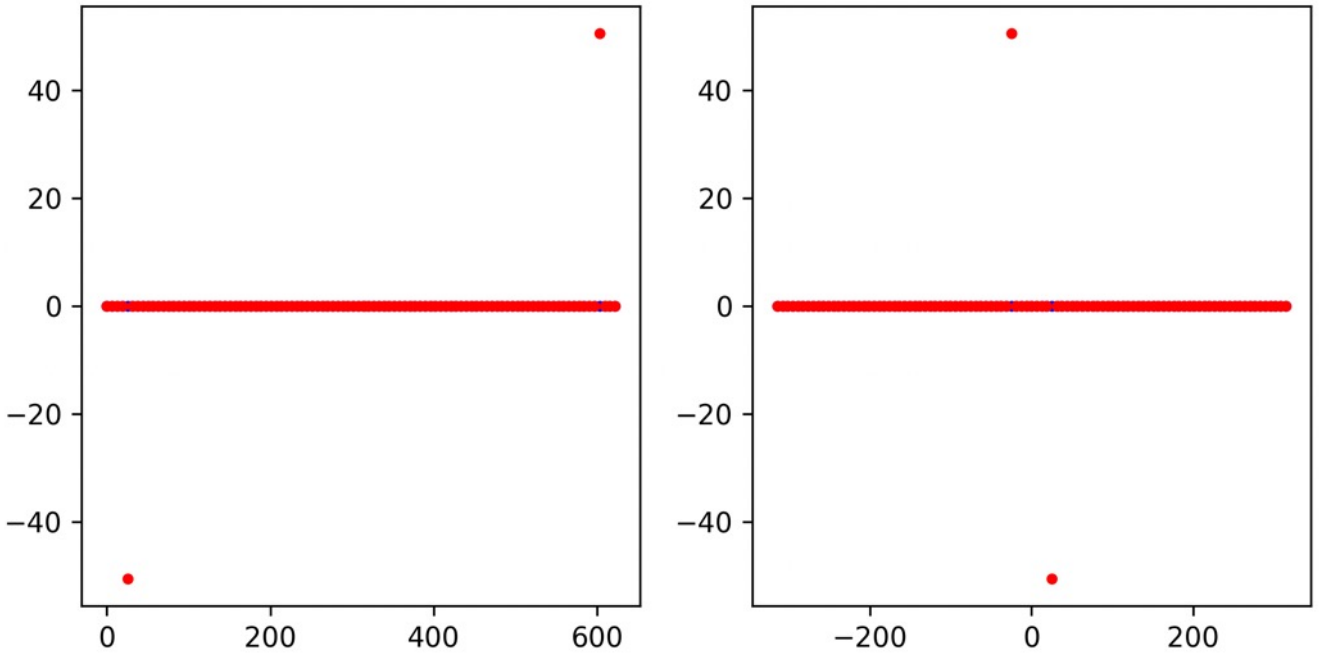
$$f(x) = \sum_{k=0}^{N-1} F(k) e^{i2\pi kx/N}$$

$$\frac{d}{dx} f(x) = \sum_{k=0}^{N-1} \frac{i2\pi k}{N} F(k) e^{i2\pi kx/N}$$

We can compute derivatives by multiplying in frequency space. This means that taking the Fourier Transform, multiplying by a frequency-dependent term, and taking the inverse Fourier Transform can be used to compute the derivative.

A Fourier Transform Example Revisited

Is there a difference between these two frequency-space signals?



According to the Nyquist Limit: we can't tell the difference! However, practically, there is a difference.

Fourier Transform Derivatives

Computing the derivative of a Fourier Transformed signal is relatively straightforward:

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{i2\pi kx/N}$$

$$\frac{d}{dx} f(x) = \sum_{k=0}^{N-1} \frac{i2\pi k}{N} F(k) e^{i2\pi kx/N}$$

You will be taking the derivative (using a Fourier Transform) in today's breakout session. It *seems* simple, but there are some idiosyncrasies in practice you will need to think about…for example: how do you define the 'k' vector?

Breakout Session L03
Computing Derivatives with Fourier Transforms

In class, we discussed computing the Fourier Transform of a signal, in which we could represent any periodic signal as a sum of sine and cosine functions (or complex exponentials):

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{i2\pi kx/N}$$

Computing the derivative of a Fourier Transformed signal is relatively straightforward:

$$\frac{d}{dx} f(x) = \sum_{k=0}^{N-1} \frac{i2\pi k}{N} F(k) e^{i2\pi kx/N}$$

We can compute derivatives by multiplying in frequency space. This means that taking the Fourier Transform, multiplying by a frequency-dependent term, and taking the inverse Fourier Transform can be used to compute the derivative.In practice, the process is still relatively simple, but there are some pitfals you should know to avoid.

Breakout Session L03 Solution

```python
# Solution

L = 1.0
N = 101
x = np.arange(0, L, L / N)
signal_x = np.sin(2 * np.pi * 4 * x / L)
d_signal_x = 2 * np.pi * 4 / L * np.cos(2 * np.pi * 4 * x / L)
w = np.fft.fftfreq(N) * N * 2 * np.pi / L
w = np.roll(np.arange(N), N//2) * 2 * np.pi / L
d_signal_x_fft = np.fft.ifft(1j * w * np.fft.fft(signal_x))

```