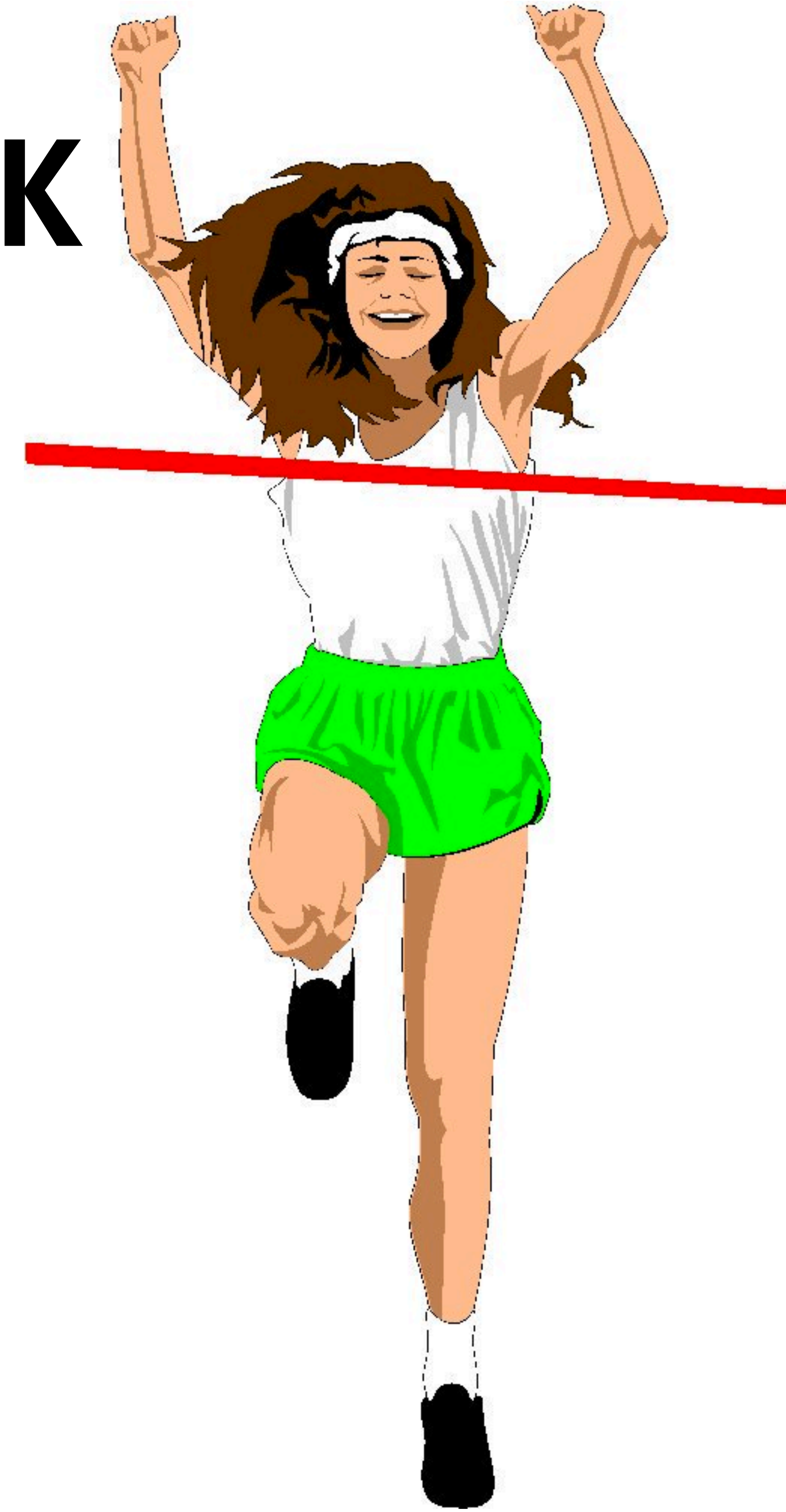
TIMESERIES:
THE FINAL CHAPTER?

10.30.2020

🎃

HOMEWORK



* Problem Set 4 was due today!

* Problem Set 5 will be posted this evening, due in 3 weeks

RECAP

* power spectrum / psd

* spectrogram

* filtering

FILTERING

* **scipy.signal** is a module in scipy that contains lots of useful functions for filter design

* **scipy.signal.firwin** creates “finite impulse response” filters with desired properties

ANALYZING A FILTER

* **scipy.signal.freqz** is a great function that tells you what the *frequency response* of your filter looks like

  * i.e. it tells you what the filter is going to do to your signal

RECALL:
FOURIER ANALYSIS

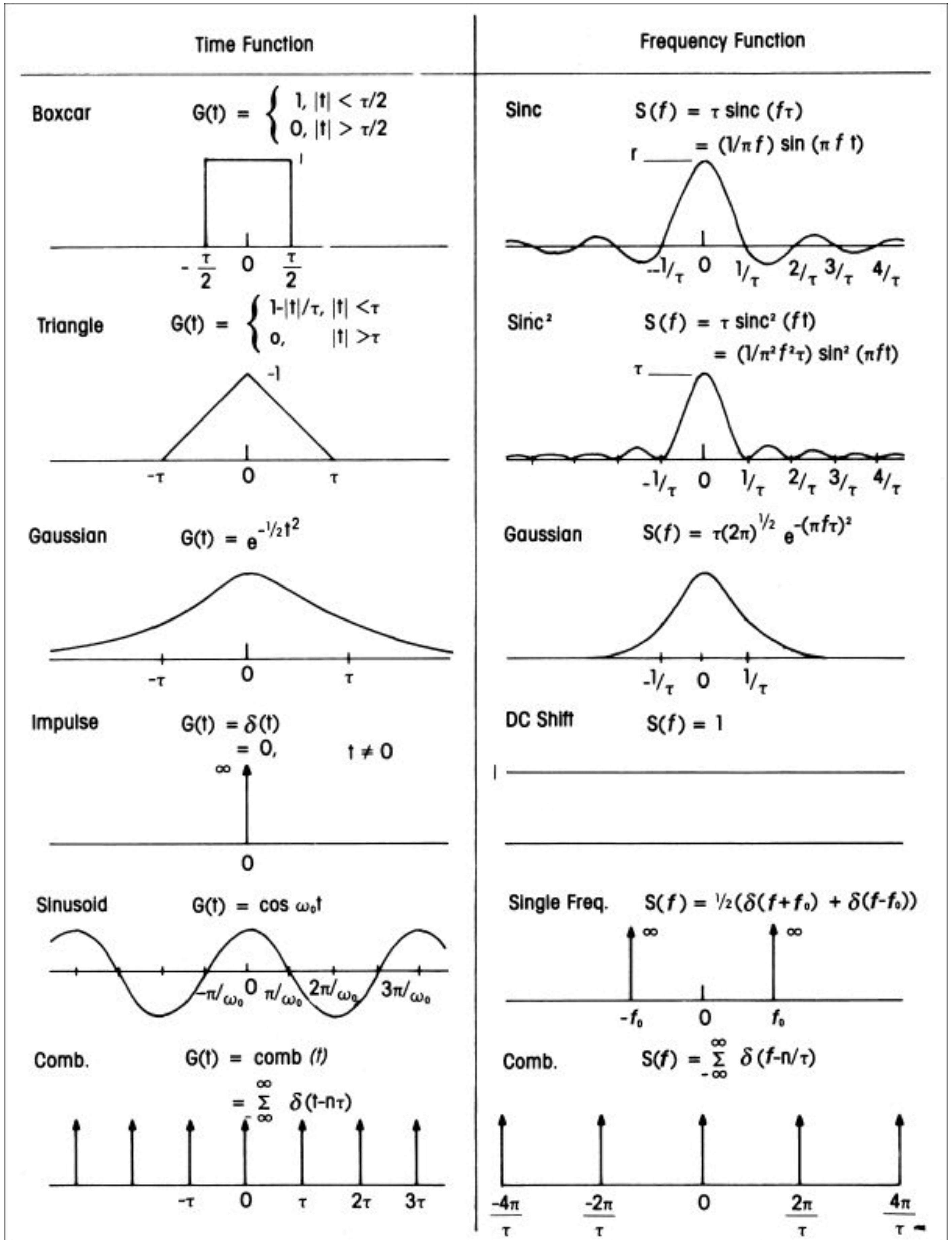
* fourier transforms have an interesting property related to convolution:

* given two timeseries, f and g, the fourier transform of their convolution = the element-wise product of their fourier transforms

$$\mathrm{FT}(f \star g) = F \cdot G$$

* the reverse is also true:

$$F \star G = \mathrm{FT}(f \cdot g)$$



Time Function
Frequency Function
Boxcar
G(t) = { 1, |t| < τ/2 ; 0, |t| > τ/2
-τ/2 0 τ/2
Sinc
S(f) = τ sinc (fτ)
= (1/πf) sin (π f t)
-1/τ 0 1/τ 2/τ 3/τ 4/τ
Triangle
G(t) = { 1-|t|/τ, |t| <τ ; 0, |t| >τ
-τ 0 τ
Sinc²
S(f) = τ sinc² (ft)
= (1/π²f²τ) sin² (πft)
-1/τ 0 1/τ 2/τ 3/τ 4/τ
Gaussian
G(t) = e^(-½t²)
-τ 0 τ
Gaussian
S(f) = τ(2π)^½ e^(-(πfτ)²)
-1/τ 0 1/τ
Impulse
G(t) = δ(t)
= 0, t ≠ 0
∞
0
DC Shift
S(f) = 1
Sinusoid
G(t) = cos ω₀t
-π/ω₀ 0 π/ω₀ 2π/ω₀ 3π/ω₀
Single Freq.
S(f) = ½(δ(f+f₀) + δ(f-f₀))
∞ ∞
-f₀ 0 f₀
Comb.
G(t) = comb (t)
= Σ_{-∞}^{∞} δ(t-nτ)
-τ 0 τ 2τ 3τ
Comb.
S(f) = Σ_{-∞}^{∞} δ(f-n/τ)
-4π/τ -2π/τ 0 2π/τ 4π/τ

NYQUIST FREQUENCY

* all of the timeseries we work with are *discrete* or *digital*, meaning that they are made up of **samples** separated by some even spacing in time

  * (note that **sample** is used in a different sense here than in statistics)

NYQUIST FREQUENCY

* the number of samples taken per unit time is called the **sampling rate**
    * e.g. in fMRI our sampling rate is typically 0.5 Hz (1 sample every 2 seconds)
    * in electrophysiology it could be as high as 25 kHz (25,000 samples per second)

NYQUIST FREQUENCY

* the sampling rate limits the frequencies that can be represented in a timeseries

* the highest frequency that a timeseries can represent is called the **Nyquist frequency**, and it is exactly half the sampling rate

Harry Nyquist

NYQUIST FREQUENCY

* for example if our fMRI data is sampled at 0.5 Hz, then the Nyquist frequency is 0.25 Hz

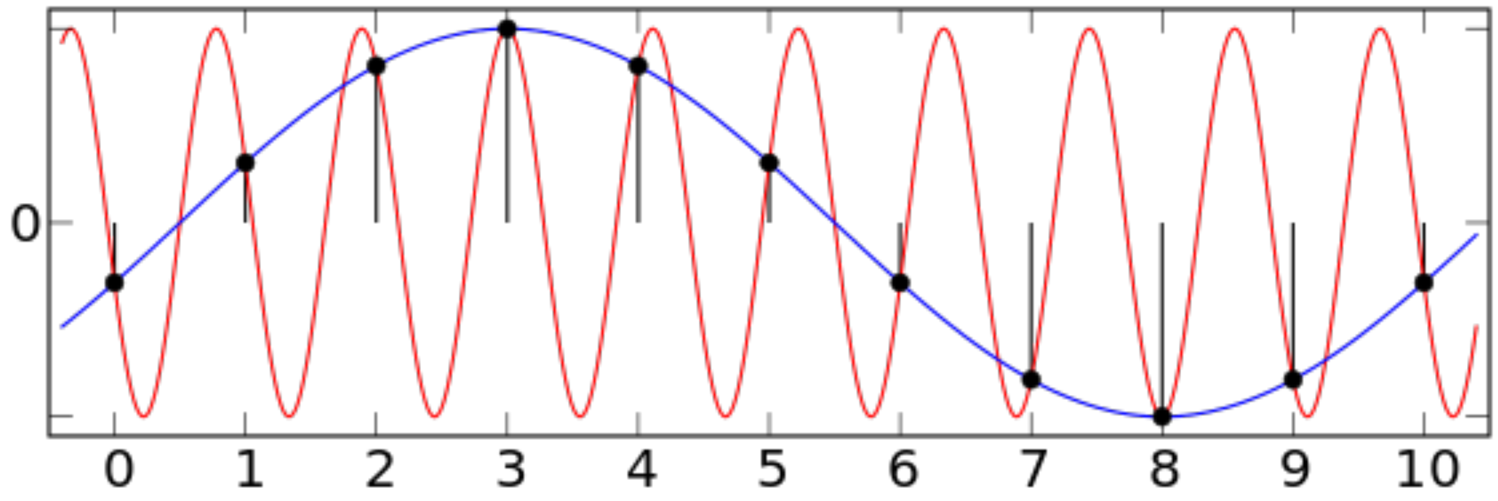
NYQUIST FREQUENCY

* why is this? why can’t higher frequency signals be represented?

NYQUIST FREQUENCY

* the problem is that any frequency above Nyquist would appear identical to some frequency below Nyquist

* this is called *aliasing*



SUBSAMPLING

* suppose we have a 20 kHz timeseries and want to **downsample** it to 2 kHz

SUBSAMPLING

* one idea: just take every 10th sample!

  * (this is called **subsampling**)

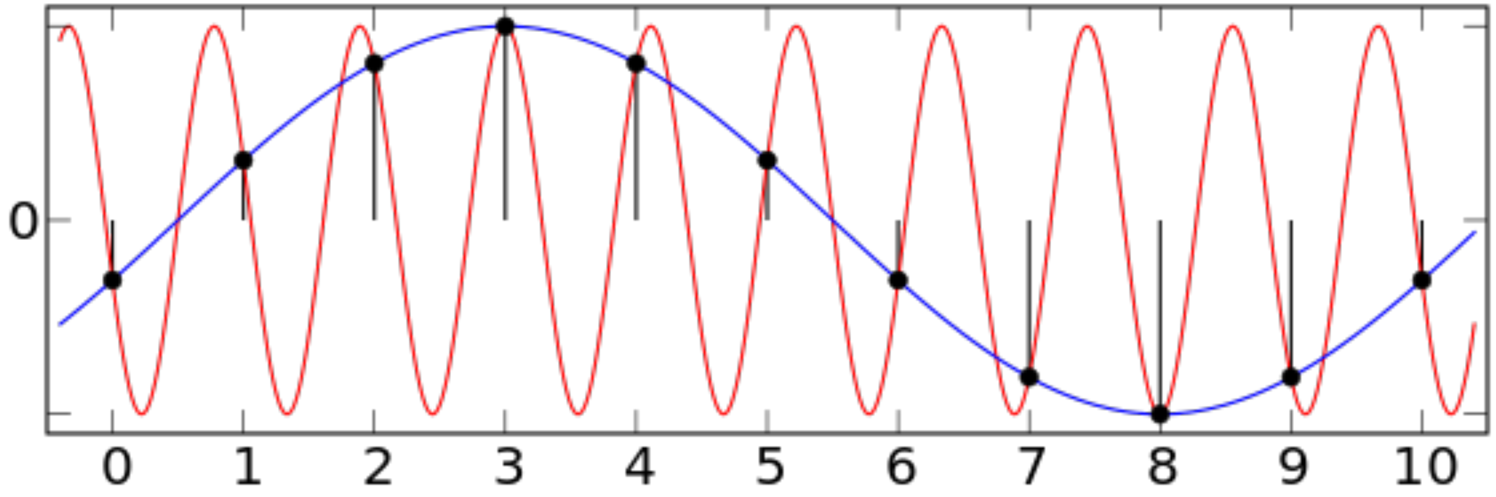
* taking every 10th sample is ***LITERALLY THE WORST IDEA***

  * (let’s see an example)

SUBSAMPLING

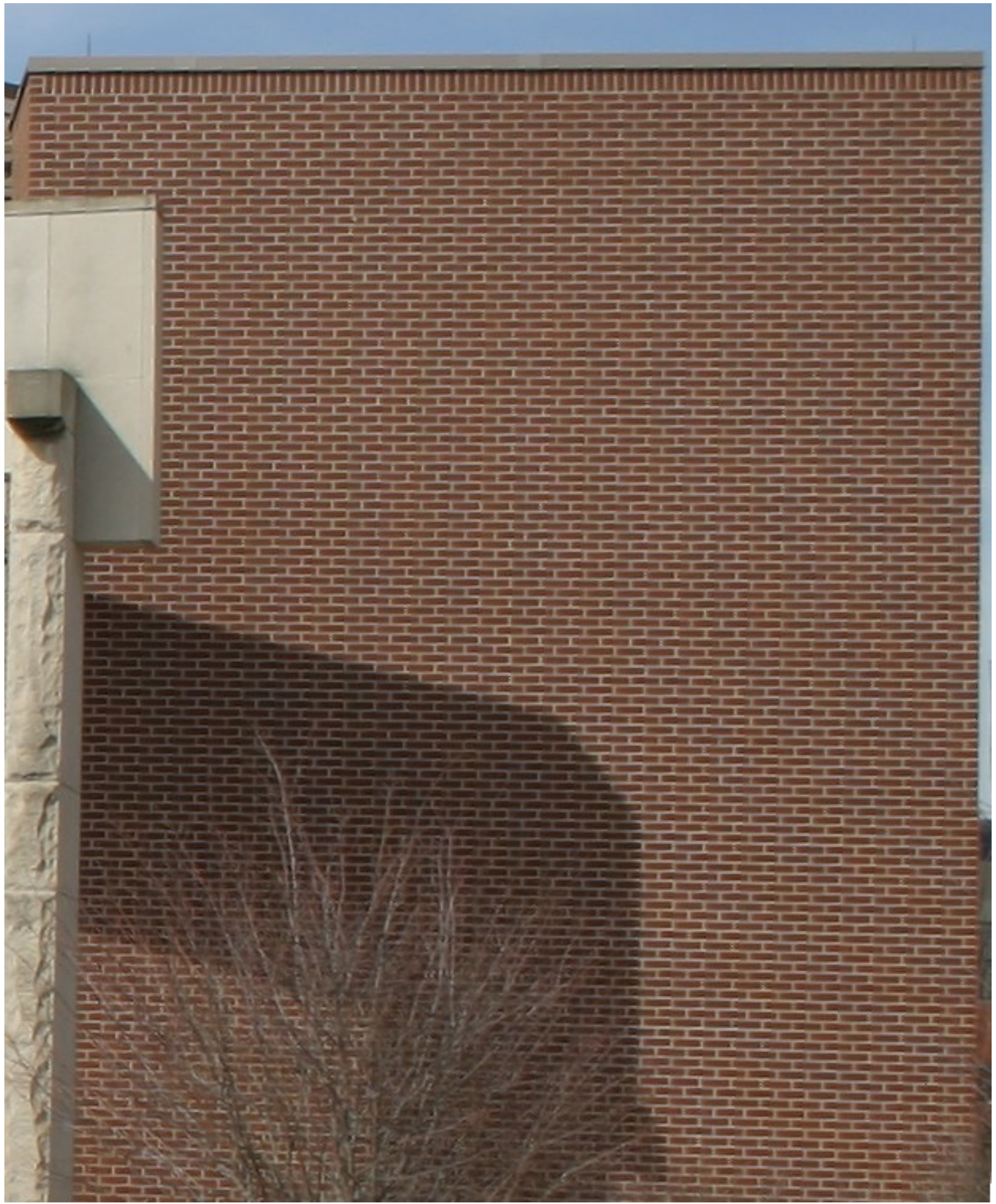
* subsampling doesn't remove high frequencies, it *turns them into low frequencies*

* this is also *aliasing*

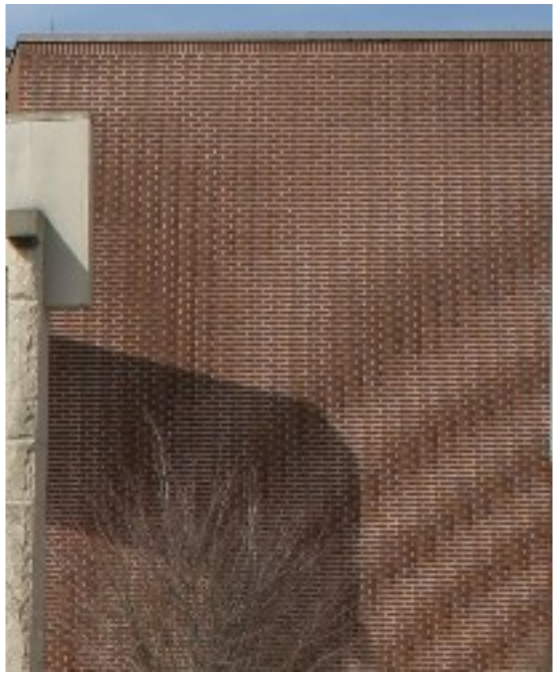


ALIASING IN IMAGES

Original Image



Subsampled

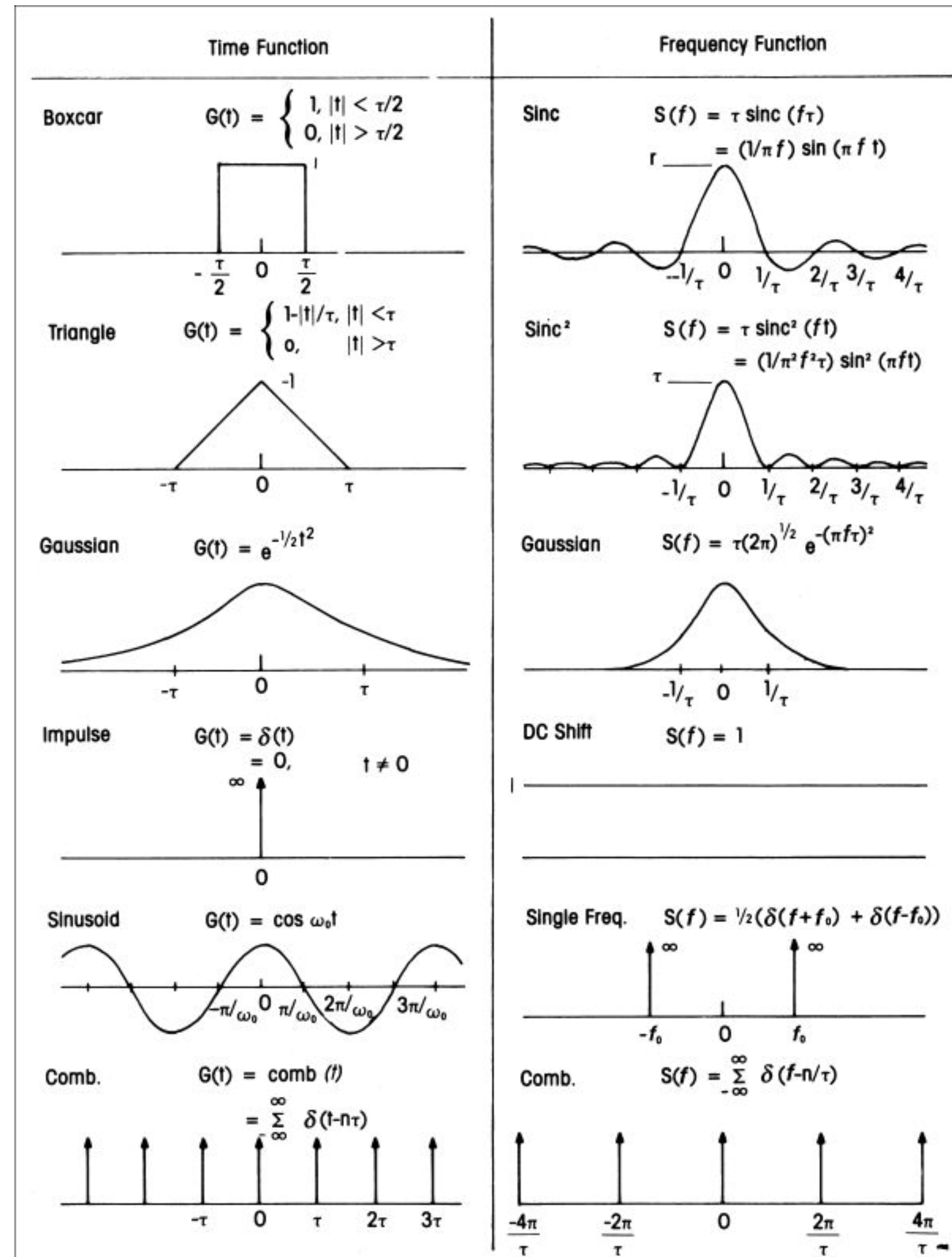


high frequency pattern (bricks) is aliased to low frequency “moiré pattern”

ALIASING

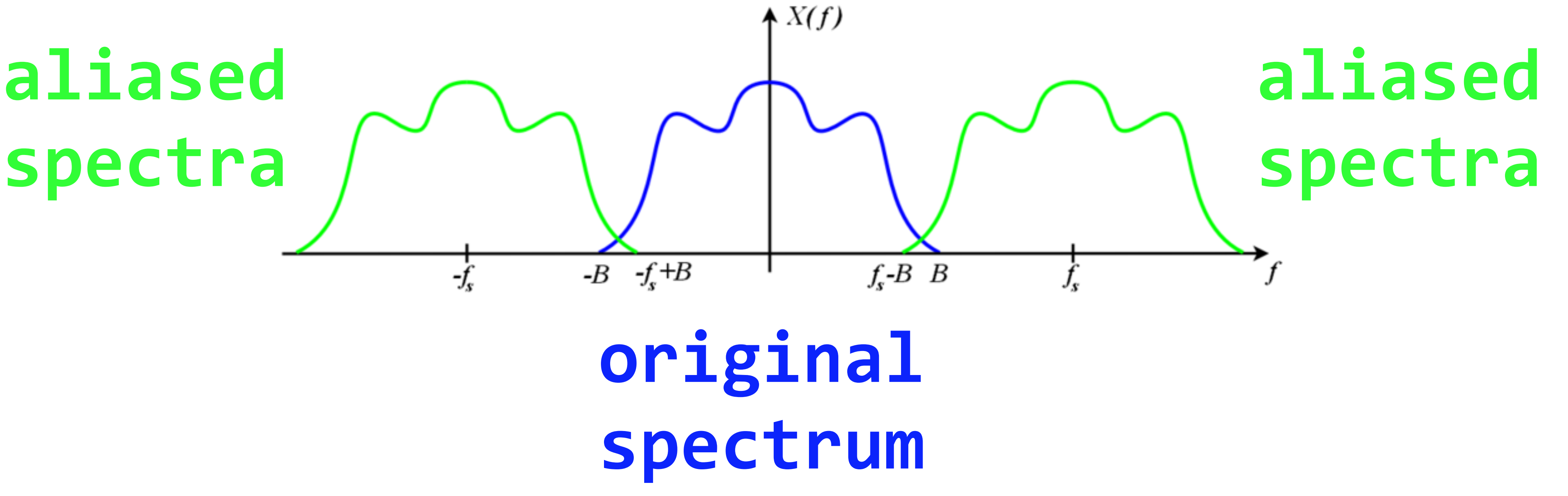
* sampling is like multiplying your timeseries by a “comb” function

* … which is equivalent to convolving the fourier transform of your timeseries by a comb function



ALIASING

* which means that the fourier transform of the subsampled timeseries can have high frequencies “invading” lower frequencies



ANTIALIASING

* how do we solve this?

ANTIALIASING

* we can use an **antialiasing** filter

* e.g.: the original signal is sampled at 20 kHz, we want to downsample to 2 kHz

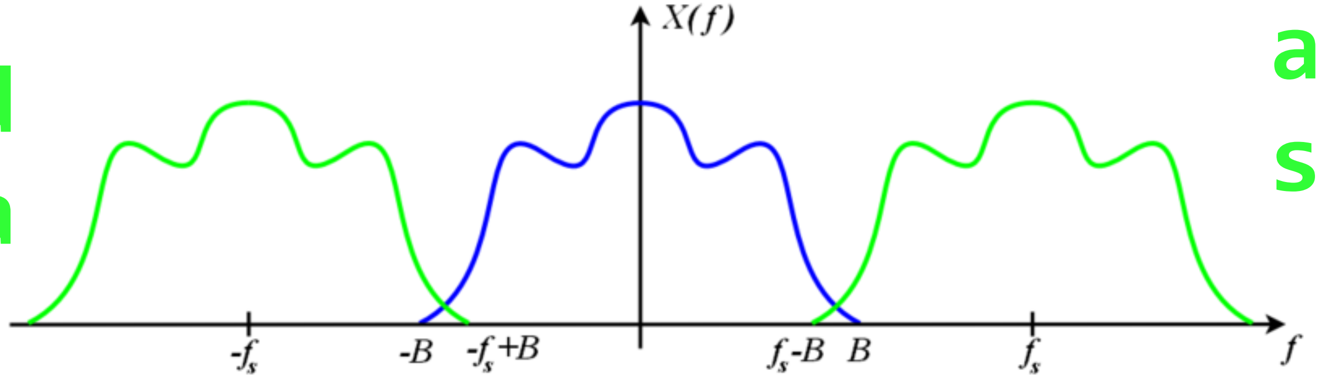
* the new 2 kHz shouldn’t contain any frequencies above Nyquist (1 kHz)

* so we **low-pass filter** the original signal at 1 kHz, and then subsample

ANTIALIASING

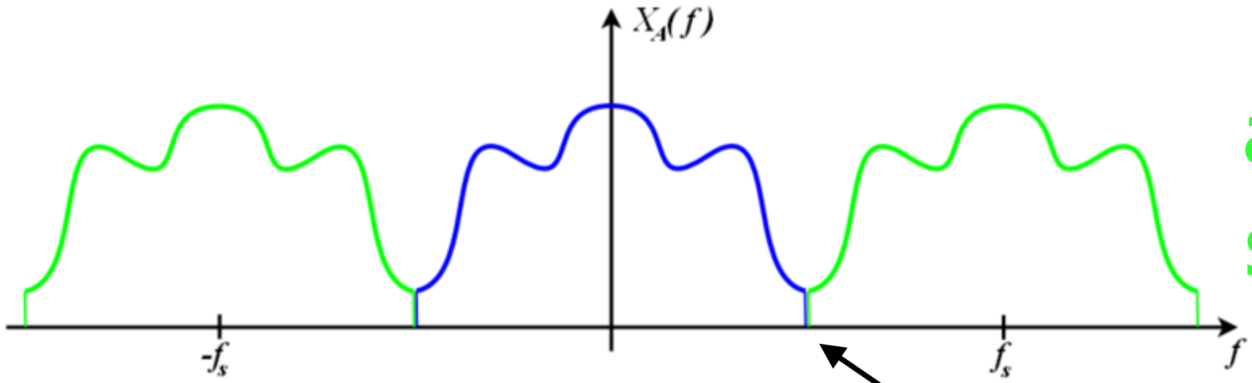
original spectrum

aliased spectra



aliased spectra

aliased spectra



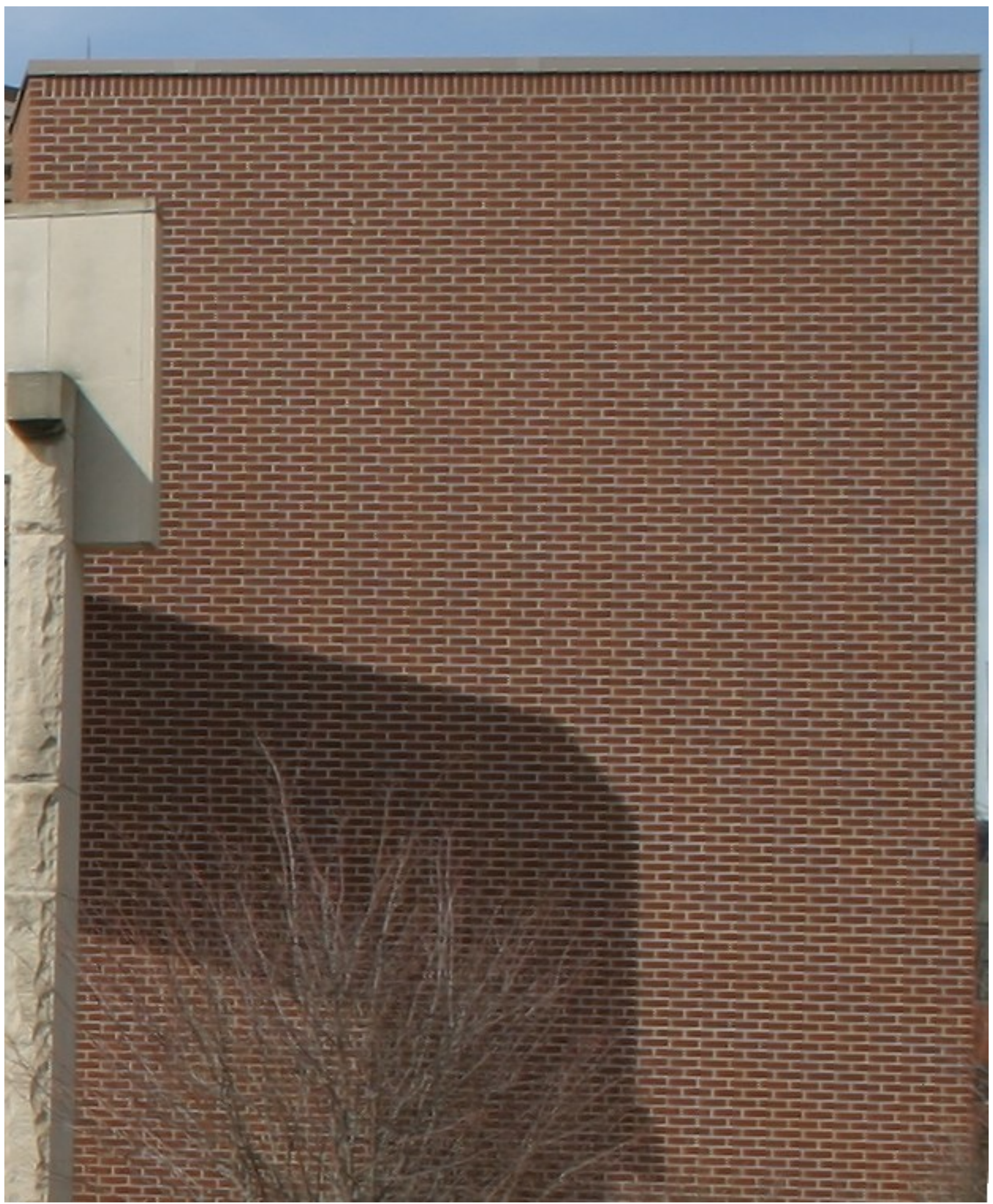
aliased spectra

filtered original spectrum

NO OVERLAP!

ANTIALIASING IN IMAGES

Original Image



Subsampled

Properly downsampled

ANTIALIASING

* there are functions in **scipy.signal** for doing good downsampling/resampling

    * **signal.decimate** is great for downsampling

    * **signal.resample** can do downsampling or upsampling

END