CS 463/516
Medical Imaging

Lecture 3

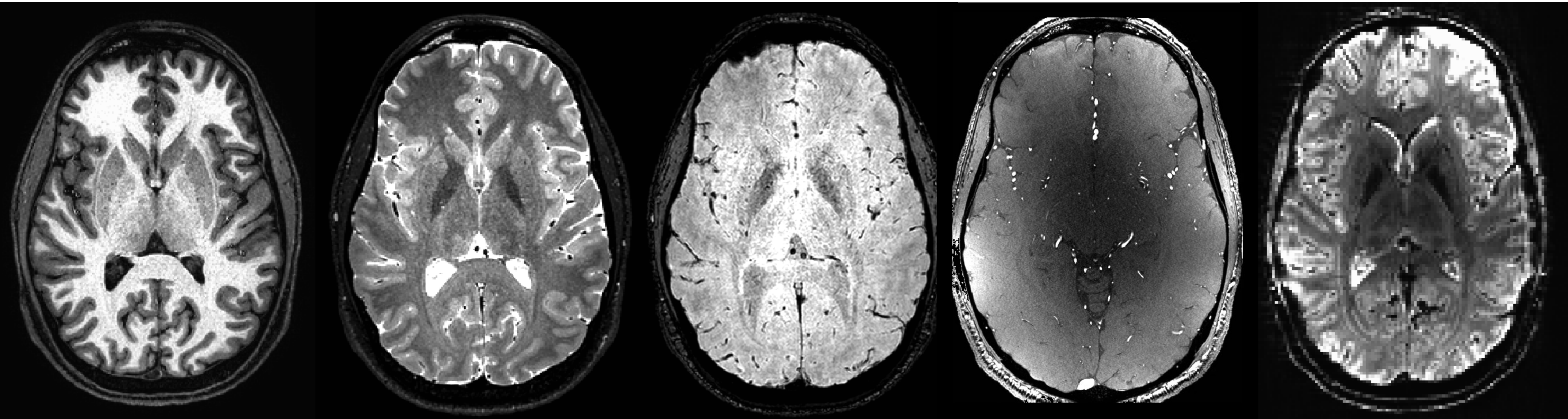
Magnetic Resonance Imaging (MRI)

3 tesla MRI

‘tesla’ is a unit of magnetic field strength

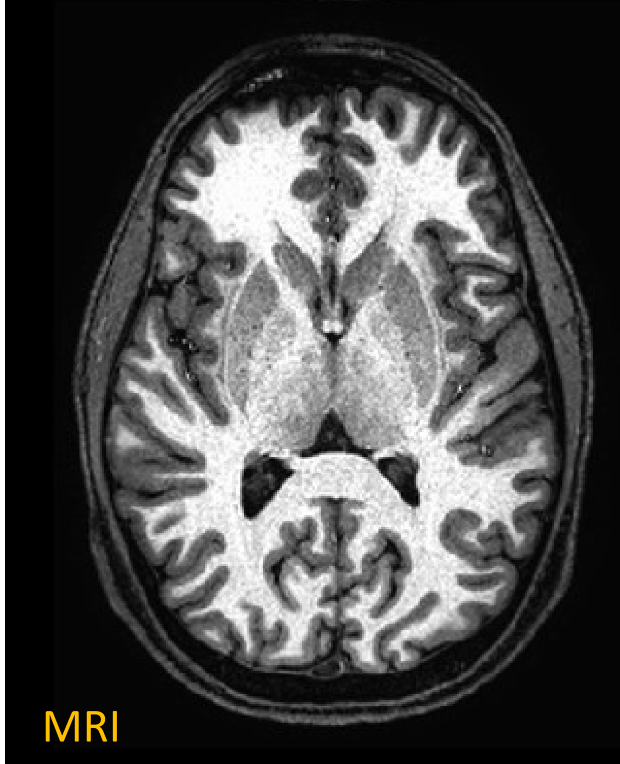
- MRI has best soft tissue contrast of all imaging modalities
  - can capture a wide range of different tissue types. More flexible than CT.
- Example: 5 different images all acquired from same person using MRI
  - We will see how MRI *pulse sequences* can be designed to generate a range of contrasts

T1 T2 SWI TOF BOLD



MRI vs CT

- **CT** voxel values measured in *attenuation coefficients*: μ
  - Brighter = higher attenuation (skull absorbs most X-rays)
  - Hounsfield Units $HU = 1000 \cdot \frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}}$
- **MRI** is (usually) not quantitative
  - Voxel values are just numbers with no physical units
  - However, ongoing research on quantitative MRI:
    - https://www.sciencedirect.com/science/article/abs/pii/S2468451117300247
- **MRI better** than CT because:
  - MRI doesn't use X-rays (CT will give you cancer if you do too many scans)
  - MRI has better soft tissue contrast (can visualize more structures)
  - MRI better for functional imaging, flow imaging, microstructure imaging
- **CT better** than MRI because:
  - Faster – CT image acquisition <1 minute, MRI scan takes up to 30 minutes
  - More patient-friendly environment
    - No loud noises, no strong magnetic fields, less claustrophobia risk

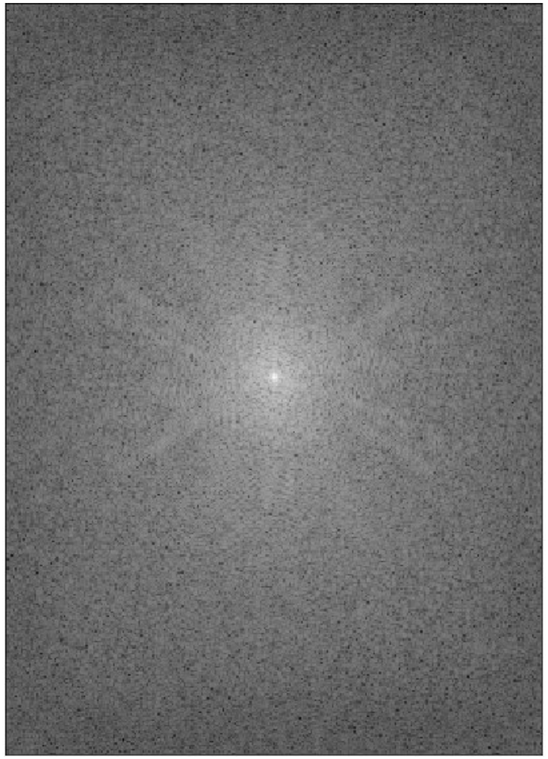


MRI acquisition

- MRI acquisition very different from CT or any other method
- MRI acquires images in ‘frequency space’ also known as ‘k-space’
- To understand MRI, **we must first understand the *Fourier transform***

k-space
acquisition

Image in frequency space



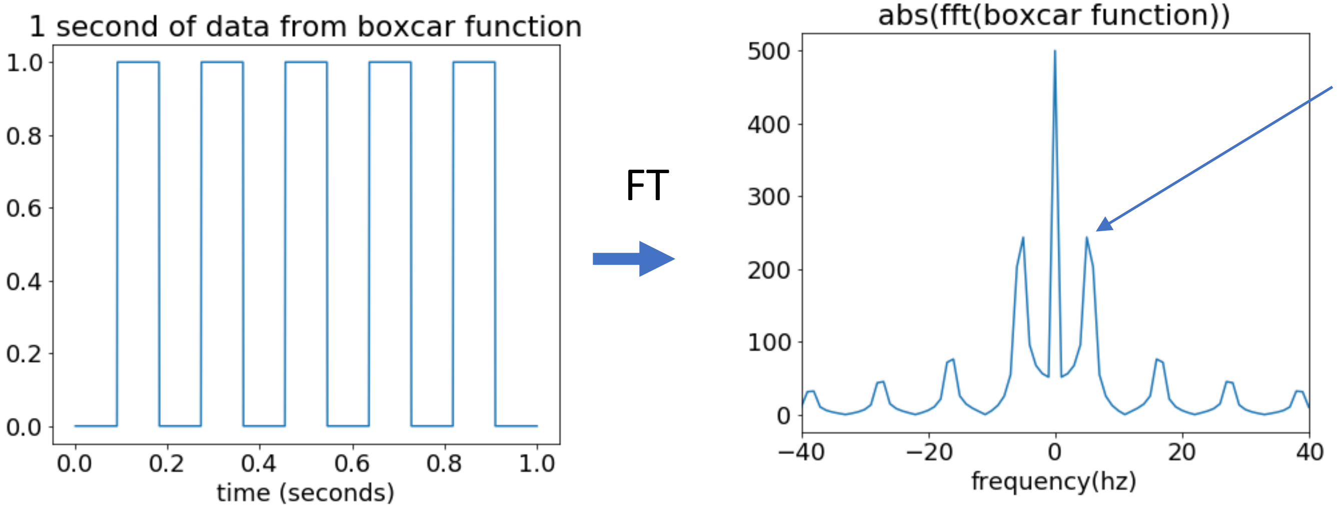
inverse
Fourier
transform

Final image

Fourier transform (FT) concept

- Fourier transform decomposes a function into its constituent *frequencies*
- FT is a linear transform that maps time (or space) domain to *frequency domain*

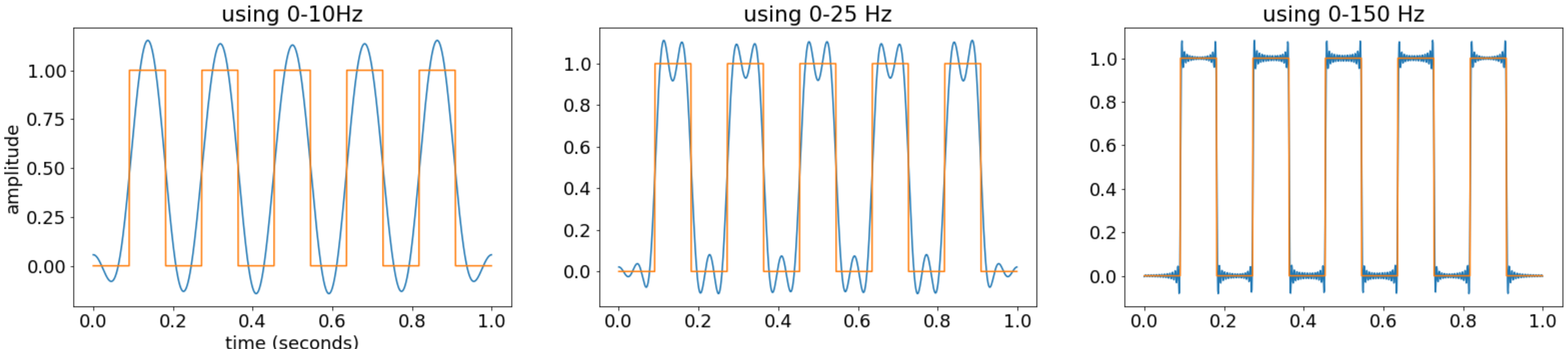
Joseph Fourier
1768-1830



Notice the peak at ~5 Hz (there are 5 boxcars cycles in one second, count them)
Hz = cycles/second

$$\frac{5\ cycles}{1\ second} = 5\ Hz$$

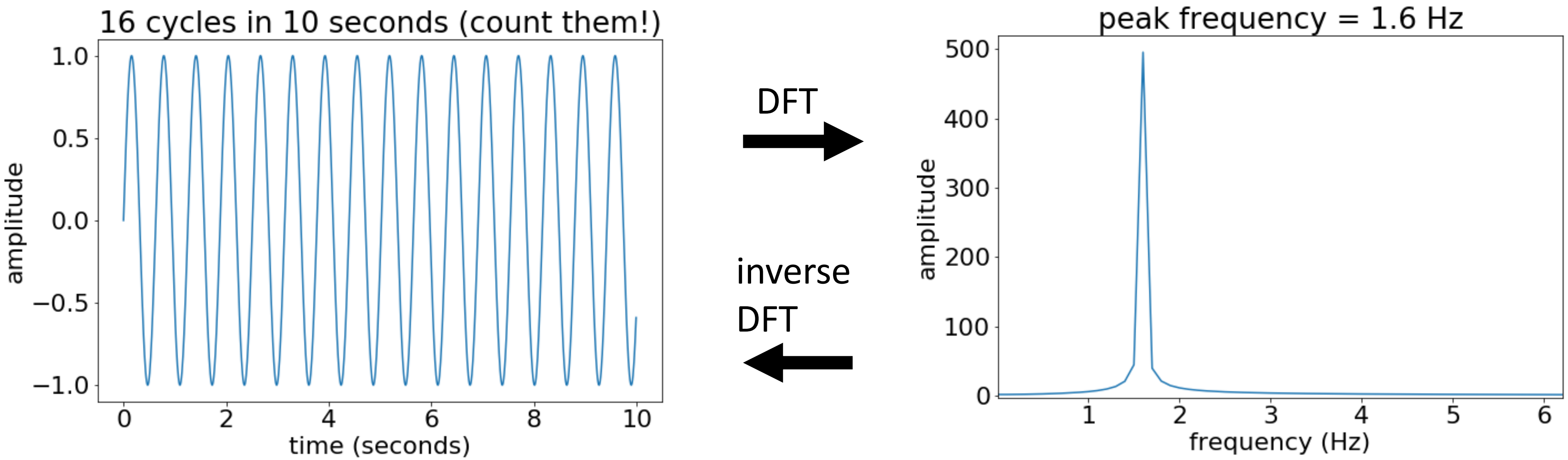
Approximating the boxcar using more and more frequencies:



approximation
original boxcar

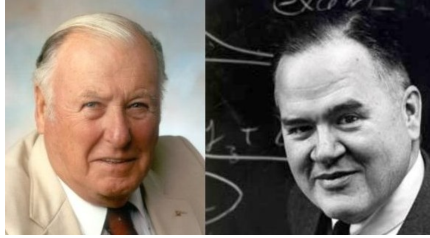
Discrete Fourier transform (DFT)

- DFT transforms sequence of N complex numbers $x_0, x_1, \ldots x_{N-1}$ into another sequence of complex numbers $X_0, X_1, \ldots X_{N-1}$:
- $X_k \;=\; \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right] \;=\; \sum_{n=0}^{N-1} x_n \cdot e^{\frac{-i2\pi}{N}kn}$ (Euler's formula)
- DFT is an invertible linear transform: $x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot e^{\frac{i2\pi kn}{N}}$
- $x_0, x_1, \ldots x_{N-1}$ is often a *time series*, and $X_0, X_1, \ldots X_{N-1}$ is a *frequency spectrum* :



Hz = cycles/second
So if there are 16 cycles in 10 seconds, 16/10 = 1.6 Hz

Cooley-Tukey FFT algorithm

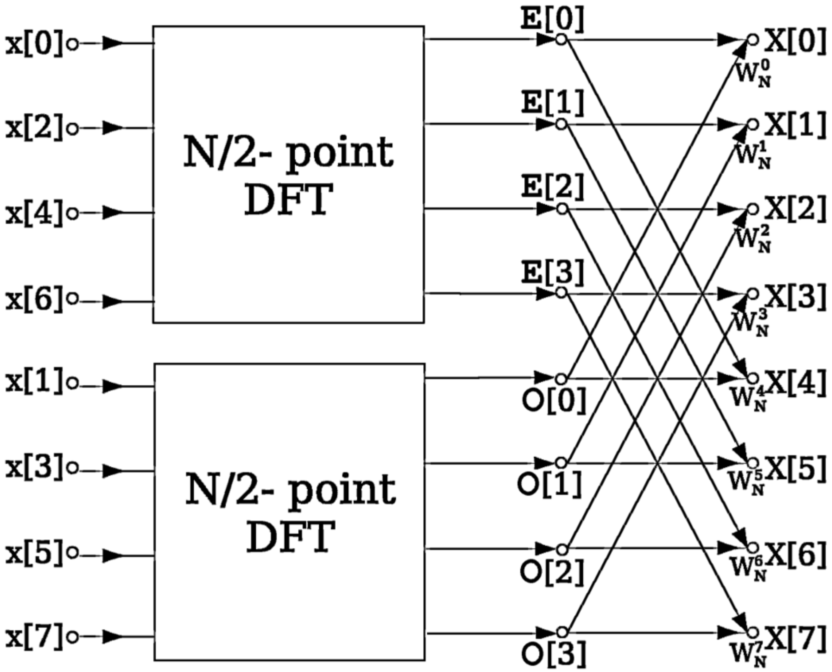


Cooley & Tukey

- How to compute the DFT and the inverse DFT on a digital computer?
- Computing $x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot e^{\frac{i2\pi kn}{N}}$ for $x_1 \dots x_N$ is too slow! $O(N^2)$ running time.
- *Cooley-Tukey algorithm* - divide and conquer, $O(NlogN)$ running time for DFT:

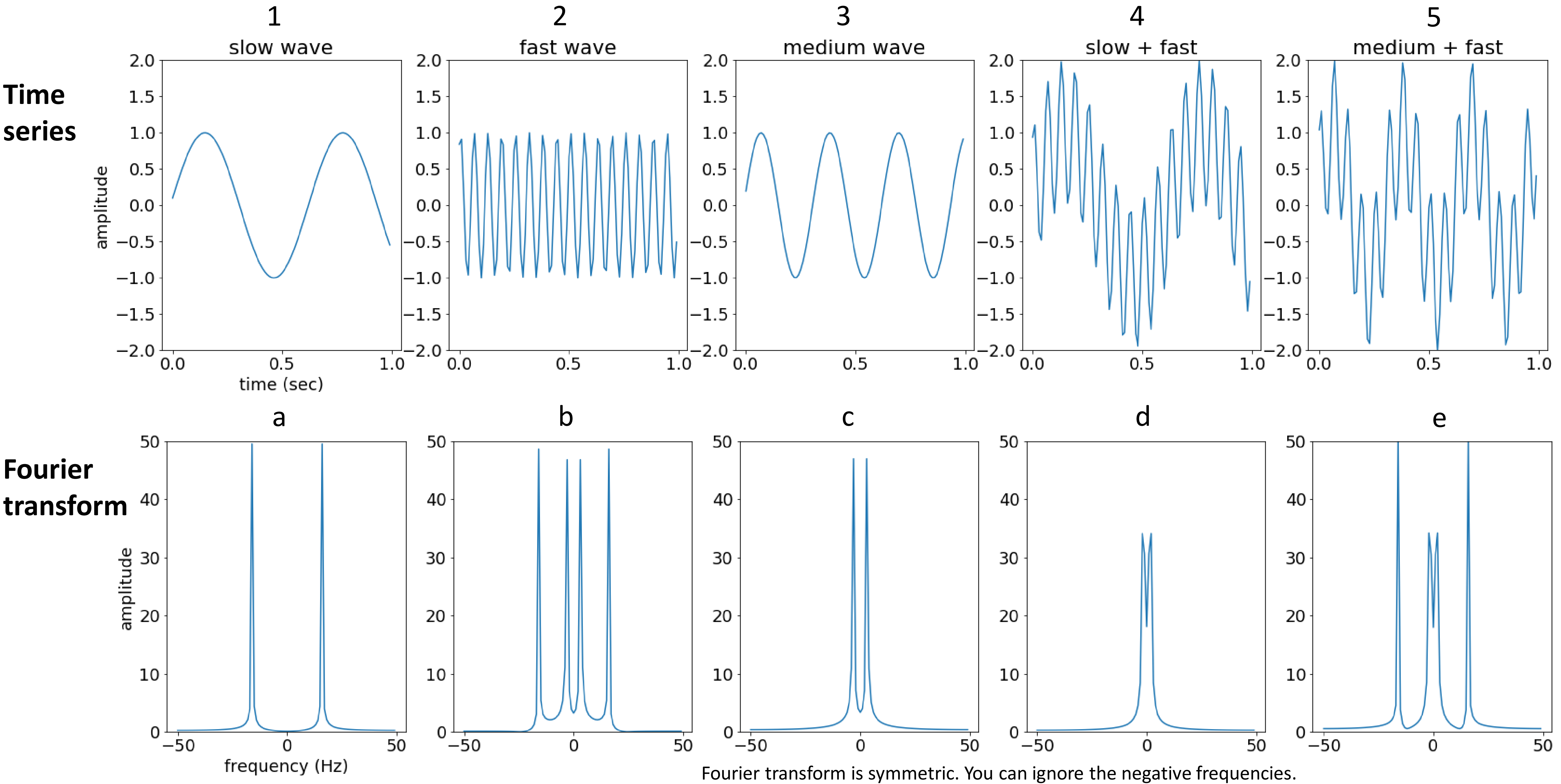
```
def fft(X):
    N = len(X)
    if N <= 1:
        return

    even = np.array(X[0:N:2])
    odd = np.array(X[1:N:2])

    fft(even)
    fft(odd)

    for k in range(0, N//2):
        t = np.exp(np.complex(0, -2 * np.pi * k / N)) * odd[k]
        X[k] = even[k] + t
        X[N//2 + k] = even[k] - t

    return X
```

Naïve python implementation.
Works, but not optimized.
Use the scipy version (link below)
https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.fft.html#scipy.fft.fft



Data flow diagram for *N*=8: a decimation-in-time radix-2 FFT breaks a length-*N* DFT into two length-*N*/2 DFTs followed by a combining stage

Quiz: match the Fourier transform to the time series



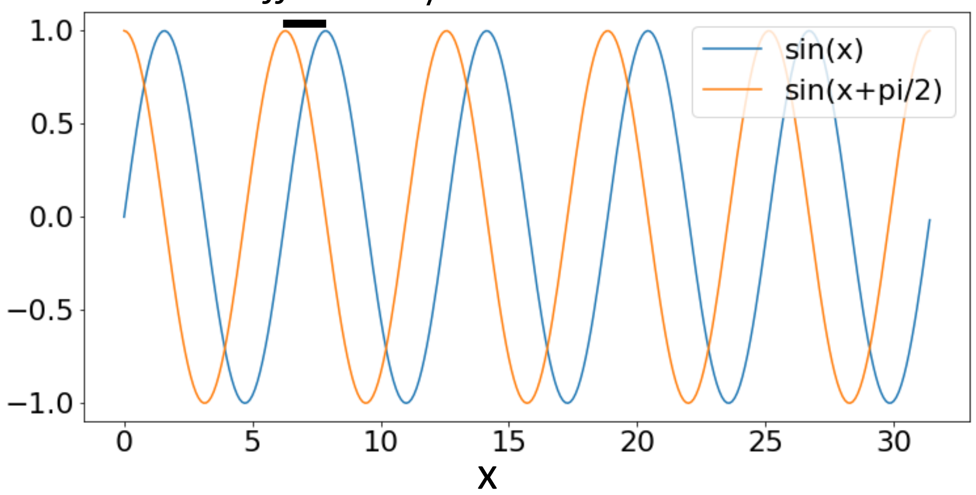
Fourier transform is symmetric. You can ignore the negative frequencies.

Solution to previous slide quiz:
$1 \rightarrow d, 2 \rightarrow a, 3 \rightarrow c, 4 \rightarrow e, 5 \rightarrow b$

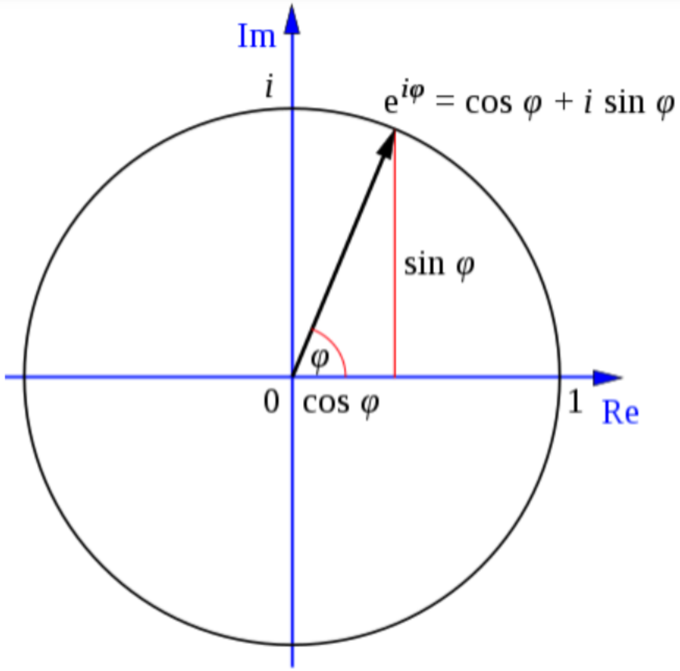
Magnitude and phase

- Fourier transform yields a complex function with two parts: *real* and *imaginary*
- Fourier transform plots we have seen so far have been *magnitude spectra*:
  - Magnitude of complex number $z = a + bi$ is $r = |z| = \sqrt{a^2 + b^2}$
  - Magnitude of Fourier coefficient represents amount of that frequency present in original signal
- The *angle* of a complex number: $z = a + bi$ is $\varphi = tan^{-1}(b/a)$ (also called *phase*)
  - Where a is real part (Re), b is imaginary part (Im)

Phase offset of $\pi/2$



For most of this course, whenever you see a plot of 'frequency spectrum' or 'frequency space' or 'Fourier domain', etc., it is typically a magnitude spectrum (meaning real and imaginary parts have been combined to yield $|z|$)

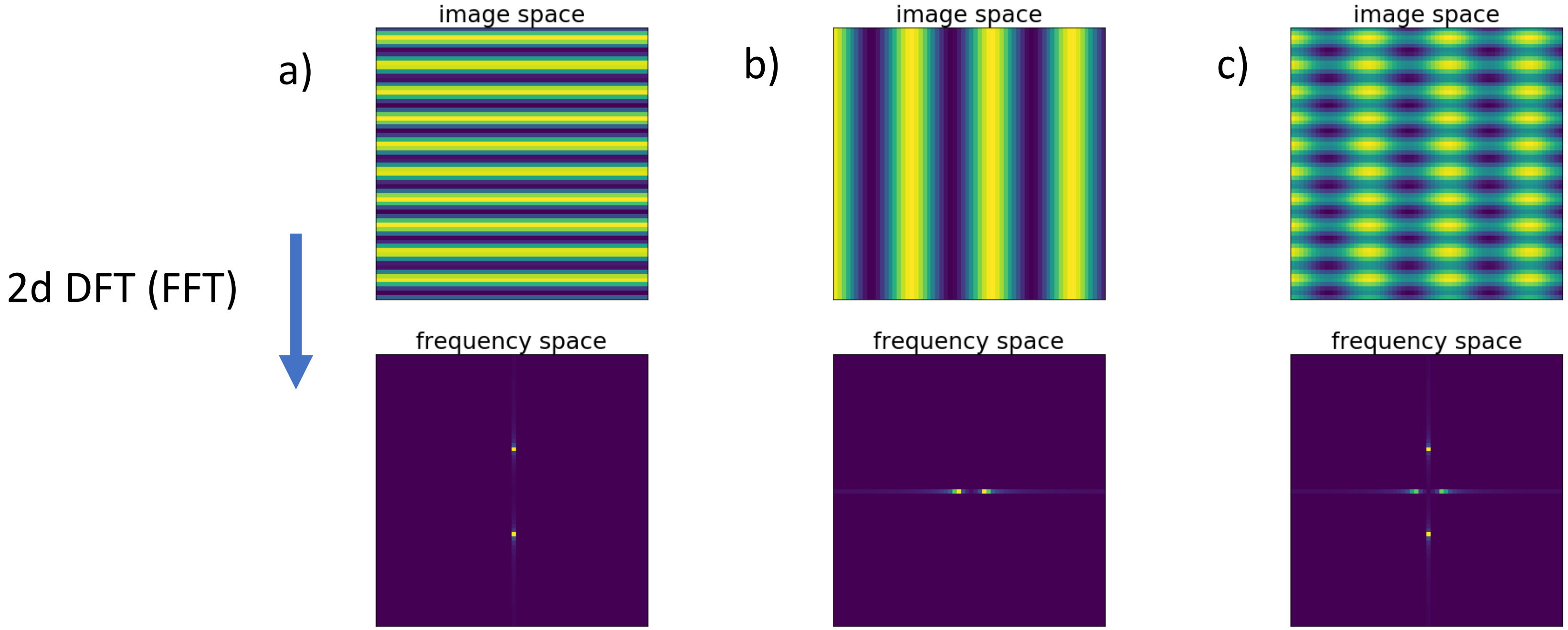


2d Discrete Fourier transform (2d DFT)

```
sz_x = 64
sz_y = 64
[X, Y] = np.mgrid[0:sz_x, 0:sz_y]
xpr = X - int(sz_x) // 2
ypr = (Y - int(sz_y) // 2)

freqs = fftfreq(64,1)

plt.subplot(2,3,1);
plt.imshow(np.sin(xpr));plt.yticks([], []); plt.xticks([], []);plt.title('image space')
plt.subplot(2,3,4)
plt.imshow((np.abs(fftshift(fft2(np.sin(xpr))))));plt.yticks([], []); plt.xticks([], [])
plt.title('frequency space')

plt.subplot(2,3,2);
plt.imshow(np.sin(ypr/3));plt.yticks([], []); plt.xticks([], []);plt.title('image space')
plt.subplot(2,3,5)
plt.imshow((np.abs(fftshift(fft2(np.sin(ypr/3))))));plt.yticks([], []); plt.xticks([], [])
plt.title('frequency space')

plt.subplot(2,3,3);
plt.imshow(np.sin(ypr/3)+np.sin(xpr));plt.yticks([], []); plt.xticks([], []);plt.title('image space')
plt.subplot(2,3,6)
plt.imshow((np.abs(fftshift(fft2(np.sin(ypr/3)+np.sin(xpr))))));plt.yticks([], []); plt.xticks([], [])
plt.title('frequency space')
```

Code to reproduce figure

- We have image $h(n,m)$ with N columns and M rows
- $\hat{h}(k,l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} e^{-i(\omega_k n+\omega_l m)} h(n,m)$ (2d DFT)
- $h(n,m) = \frac{1}{NM}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} e^{i(\omega_k n+\omega_l m)} \hat{h}(k,l)$ (inverse 2d DFT)
- Can create any image by summing 2d sinusoids:

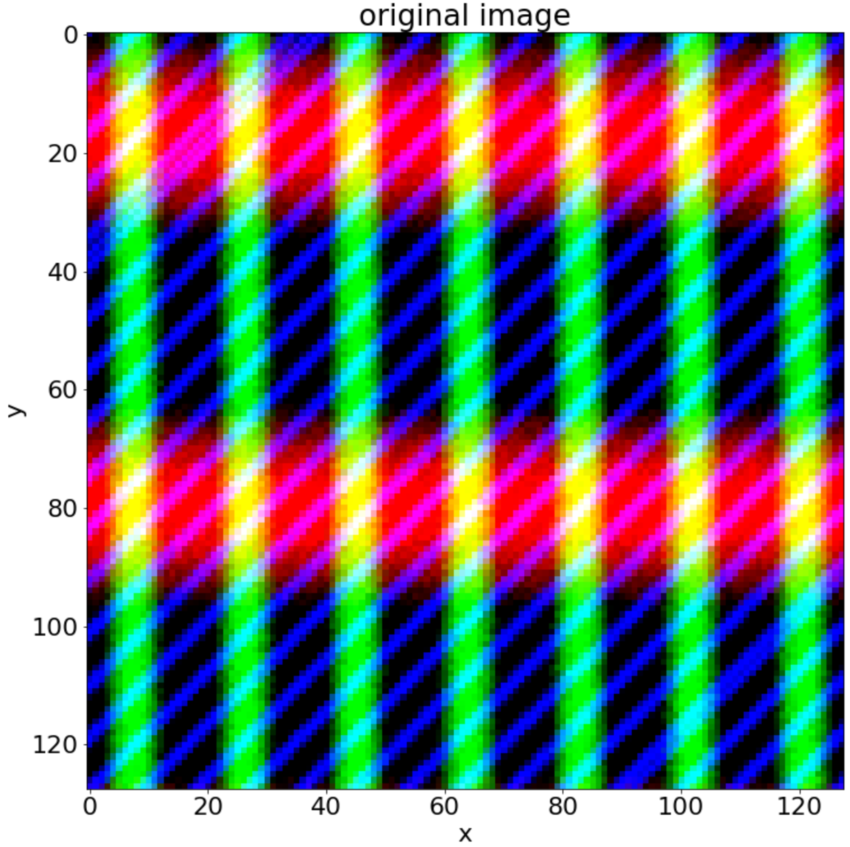


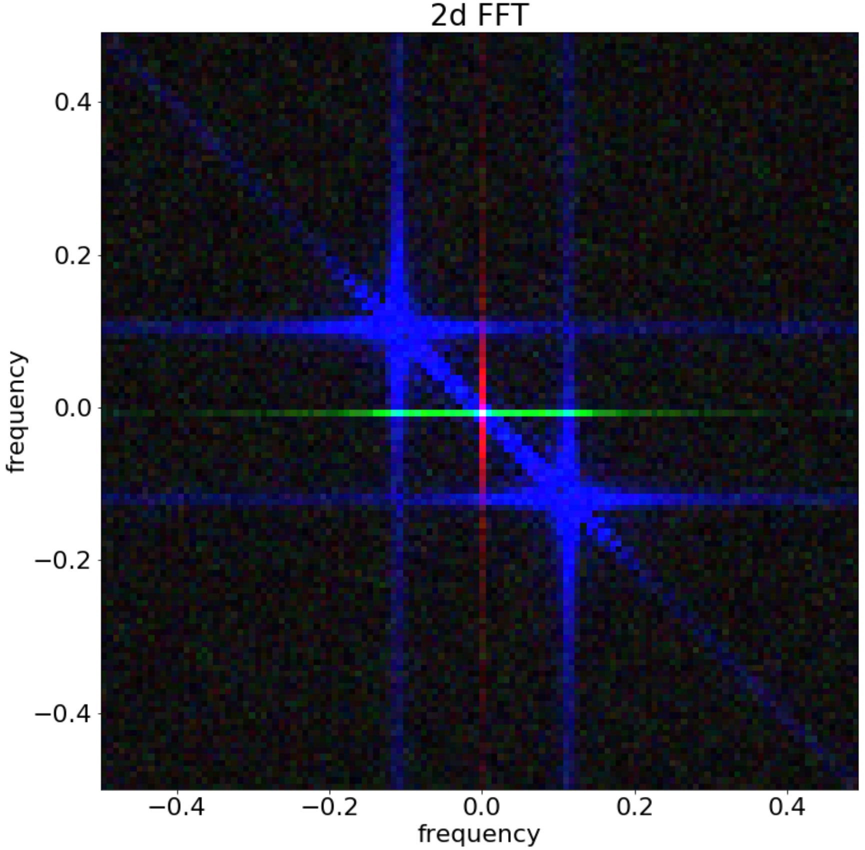
c) image created by summing the images from a,b. $c = a + b$

(c) is a simple image, created using only 2 waves, but any arbitrary image can be approximated by a sum of sin/cosine waves (as with 1d case)

2d DFT example

red = low frequency vertical wave
green = mid frequency horizontal wave
blue = high frequency 45° wave



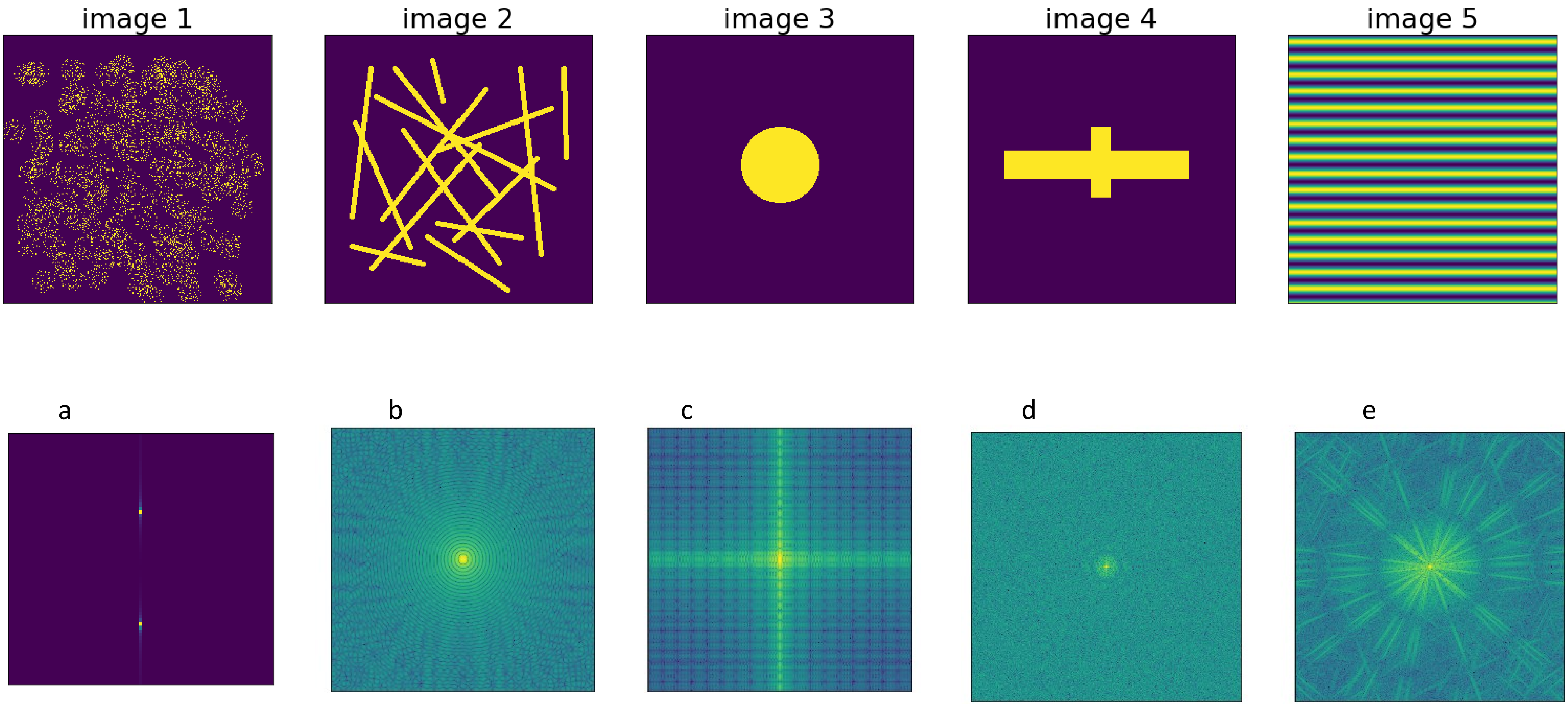


- Red wave represented in 2d FFT as higher values closer to the center (low frequencies)
- Green wave represented in 2d FFT as higher values in center and mid frequency range
- Blue wave represented as higher values further from center (higher frequencies)

Creating rgb image (above) in python:

```
sz = 128
rgb[:,:,0] =  np.sin(xpr/10) + np.random.rand(sz,sz)/5
rgb[:,:,1] =  np.cos(ypr/3) + np.random.rand(sz,sz)/5
rgb[:,:,2] =  rotate(np.cos(xpr),45,reshape=False,mode='reflect')  + np.random.rand(sz,sz)/5
```

Quiz: match the 2d Fourier transform to the image



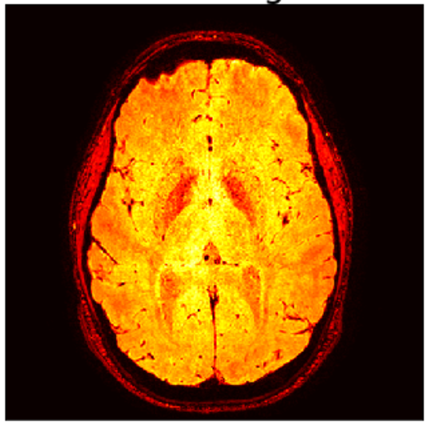
Solution to previous slide quiz:
$1 \rightarrow d, 2 \rightarrow e, 3 \rightarrow b, 4 \rightarrow c, 5 \rightarrow a$

Filtering in frequency space

- *Filters* are used in image/signal processing to reduce or accentuate certain frequencies
- Can use *low-pass* filter to increase the low frequency content of image (and vice versa)

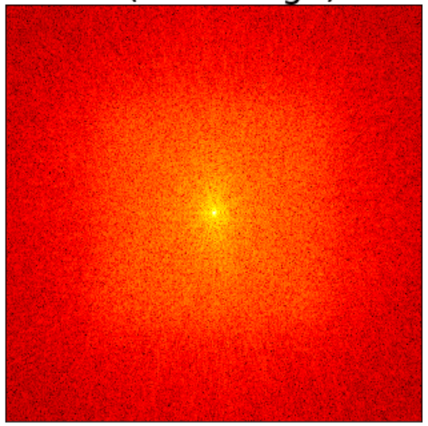
The '*' indicates a point-wise multiplication of the filter with the fft(image)

raw image



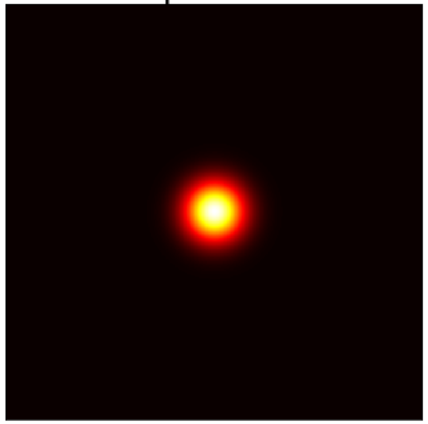
fft $\rightarrow$

fft(raw image)



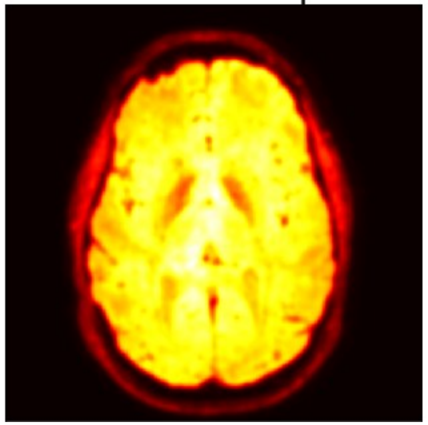
*

low-pass filter



inverse fft $\rightarrow$

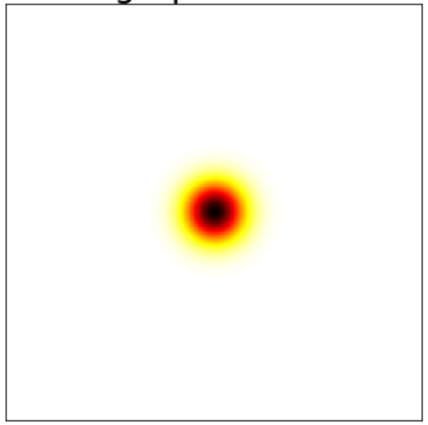
increased low frequencies



blurred

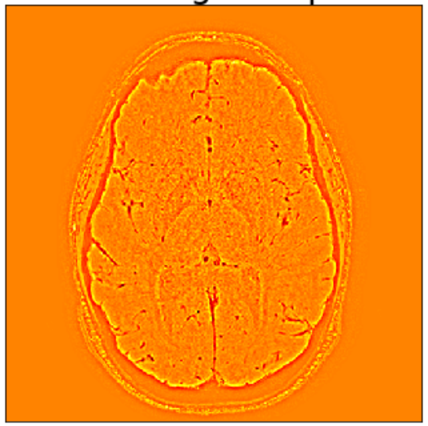
*

high-pass filter



inverse fft $\rightarrow$

increased high frequencies



sharpened

Steps to filter image in frequency domain:
1) take fft of raw image
2) multiply fft with a filter
3) invert the fft (after multiplying)

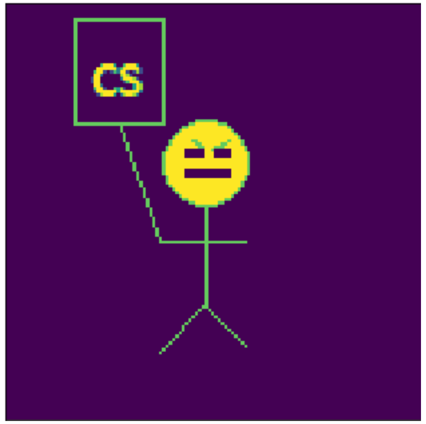
N.B. – this slide *does not* apply to MRI, it is for CT only. I'm just trying to show how the filtered backprojection method uses frequency space filtering to reconstruct the image.

Radon transform again

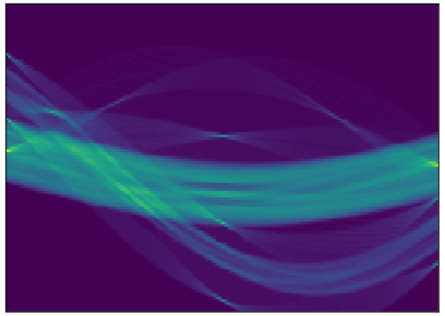
Filtered backprojection technique of iradon relies on filtering in frequency space. The ramp filter is a *high-pass* filter that reduces low frequency contribution

As you can see, filtered backprojection reduces the low frequency content of the reconstructed image

raw

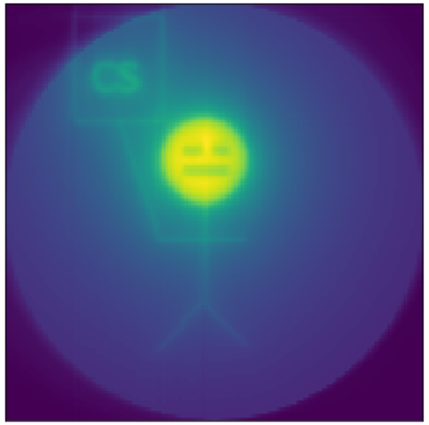


Radon(raw)

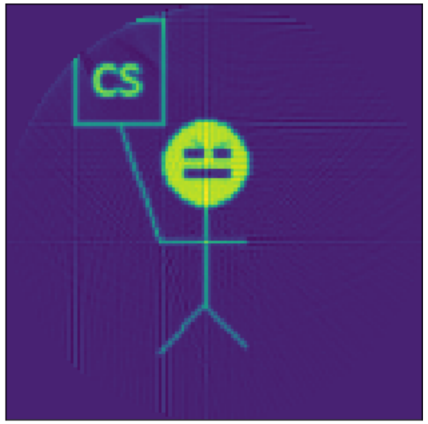


iradon (without filtering in frequency space)

backprojection image

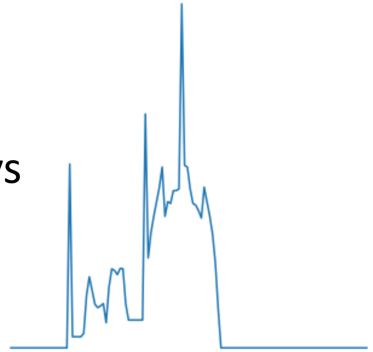


filtered backprojection



invert filtered FFT and do backprojection for all angles

X-rays

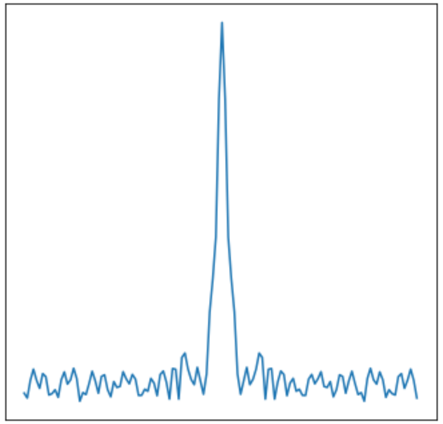


projection at $\theta = 0^{\circ}$

(one column from Radon(raw))

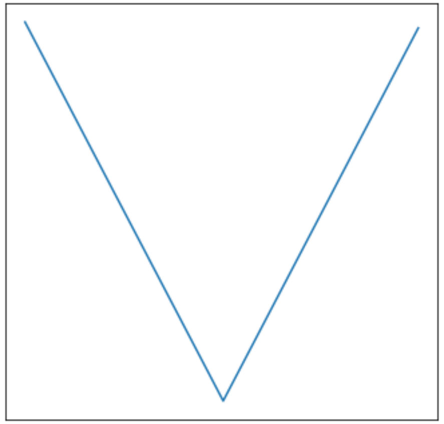
FFT

Magnitude spectrum of projection



*

Ramp filter



=

Magnitude spectrum after applying ramp filter