k-Space lab in Python

This is an interactive Jupyter Notebook. The notebook consists of markdown cells (like this) and code cells. To edit a code cell, you just click in it and start typing. If you want to edit a markdown cell, you double click it. This reveals the markdown code that is used to format the content of the cell. If you want to know everything about markdown cells, look here.

To execute code (markdown or otherwise) you can either click the Run button, or use the keyboard shortcut Shift + Enter.

This lab is making use of the NumPy (tutorial) and PyPlot (tutorial) packages.

A few tips:

- I have defined lambda functions for "centered" Fourier transforms, feel free to use them
- Use `plt.imshow` to display images, and use the option `cmap='gray'` to display in black and white.
- When the question calls for multiple plots, you can use `plt.subplot`, e.g. `plt.subplot(2,2,1)` to plot in the first position of a 4 by 4 grid
- Arrays can be sliced using the `[start:stop:step]` notation, e.g. `kspace_res512_FOV48[1::2]` for every second row.
- If you at any point want to initialize a zero matrix with `np.zeros`, you need to specify `dtype=np.complex64`
- The command `np.log` does not work with arrays containing zeros, you can use `np.log(x+1)` for those instances.
- `ij` or `0+ij` can be used to represent complex numbers. `np.pi` can be used for pi.
- Finally, if things get messed up and want to reset. Run the first cell again!

In [ ]:

```python
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import h5py

with h5py.File('data.h5', 'r') as F:
    kspace_res512_FOV24 = np.array(F['kspace_res512_FOV24'])
    kspace_res512_FOV48 = np.array(F['kspace_res512_FOV48'])

fftc = lambda f: np.fft.ifftshift(np.fft.fftshift(np.fft.fft(f)))
ifftc = lambda F: np.fft.fftshift(np.fft.ifft(np.fft.ifftshift(F)))
fft2c = lambda f: np.fft.ifftshift(np.fft.fftshift(np.fft.fft2(f)))
ifft2c = lambda F: np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(F)))
```

Task 1

Use `kspace_res512_FOV24`

1. Show the magnitude, phase, real, and imaginary part of the k-space. To better visualize k-space, you should log the matrix using the log command.

In [ ]:

1. Do an inverse FT of the k-space data in the frequency encoding direction (along columns).
   - Display the magnitude and phase of the result.
   - Describe what's happened, which domain is the data in – spatial or frequency?
   - What are the unit on the x and y axis?

In [ ]:

1. Do an inverse FT of the k-space data in both the frequency and phase encoding direction.
   - Display the magnitude and phase of the result.
   - Which domain is the data in- spatial or frequency?

In [ ]:

Task 2

Use `kspace_res512_FOV48`

1. Decrease the FOV in the spatial domain to 24x24 cm, and 16x16 cm by removing parts of k-space.

In [ ]:

Task 3

Use `kspace_res512_FOV24`

1. Decrease the spatial resolution by replacing k-space data with zeros before performing the ifft. The new images should have a 512x256, 256x256, 64x64, 512x64 and 64x512 resolution. Display the magnitude of k-space and the magnitude image in the spatial domain.

In [ ]:

1. What is the difference between the 512x64 and the 64x512? Describe the artifact that is appearing at low resolutions.

Task 4

Use `kspace_res512_FOV24`

1. Set the k-space element (kx,ky) to 200000 for these elements: (100,100), (200,200), (250,250), (255,255), (257,257). Display the spatial image for each k-space adjustment. Describe the artifact.

In [ ]:

Task 5

Use `kspace_res512_FOV24`

Change the k-space phase without altering the magnitude.Remember that complex numbers can be written as `z = mag * np.exp(1j*angle)`

1. Increase the phase by 2 radians in the following k-space rows: 1–20, 201–220, 237–256 and 247–266. Display the magnitude image in the spatial domain for each k-space adjustment

In [ ]:

Task 6

Use `kspace_res512_FOV24cm`

1. Filter the spatial image with a boxcar filter.
2. Use two kernel sizes: 7x7 and 4x4.
3. Do an inverse FT and display the magnitude of k-space. Describe the results.

To define a boxcar filter, you can use `np.ones((n,n))`, 2d convolution is a feature of the `scipy` package `signal`. Use `from scipy import signal` to import it, and then `sp.signal.convolve2d(image,filter)` to perform a 2D convolution.

In [ ]: