Computational MR imaging
Laboratory 2: k-space sampling and Fourier reconstruction

Code submission is due by 12:00 before the next Thursday lab section. Please upload your code to StudOn in a described format. Late submissions will not be accepted.

Learning objectives

- Reconstruct Cartesian MRI data
- Compute the point spread function
- Apply Square and Hamming filter to k-space
- Understand oversampling along the Readout direction

1. FFT reconstruction of Cartesian MRI data

- Kdata is already loaded. You will play around with this data.
- Reconstruct kdata using *iffc2c*
  - Implement methods *fft2c* and *ifft2c* to perform forward and inverse Fourier transforms for MRI data as described in class, see *numpy.fft* Python module
  - Keep in mind that you have to preserve signal energy between transforms
- Plot the magnitude and phase of the reconstructed image

2. Effects of the square filter on zero-padded k-space

- To simulate zero-padding of kdata, we will create square filters, which are the same size as kdata, outer with 0s and center with 1s in different central sizes of 64×64, 128×128, and 256×256.
  - Implement a method, *get_sqaure_filter.*
- Simulate zero-padded k-space by appying the square filters to the kdata.
  - Implement a method, *filtering* to apply the filter to kdata.
  - Repeate this for [64x64, 128x128, 256x256] filter sizes.
- Reconstruct all of zero-padded k-space and compare them with the reconstructed image from the Task 1.
- Discuss the effects of k-space truncation

3. Point spread function (PSF)

- Details about PSF will be explained in the exercise.
- Implement a method, *get_psf_1d_square*. To generate 1D PSF, you take the middle row of the 2D filter that you generate by *get_square_filter*.
  - Think about how to use *ifft2c* method for the 1D signal.
- Compute the (1D) PSF for each k-space truncation pattern in the Task 2
- Plot the different PSF's in the same figure using different colors
- Compute the full-width at half-maximum (FWHM) for each PSF
  - Implement a method *get_fwhm*.
- Discuss the differences in spatial resolution

4. Effects of the Hamming filter on zero-padded k-space

- In this task, you will apply Hamming filters to kdata. The Hamming filter is similar to the square filter but instead of 1s in center, center values follow bell shaped curves.
    - Implement a method, *get_hamming_filter*.
- Simulate zero-padded kdata with different sizes of hamming filters
- Reconstruct each zero-padded kdata and compare them with the reconstructed images from the Task 1 and Task 2.
- Compute the PSF for each window and compare it to the PSF from the Task 3, where a square window was implicitly used for the reconstruction
- Compute and compare the FWHM of the PSF of the Hamming window and again compare it to the square window from the Task 3
- Discuss advantages and disadvantages of the different windows with respect to each other, in particular the effects on resolution and Gibbs ringing. Hint: Take a look at the sidelobes of the PSF

5. Oversampling the readout dimension

Oversampling along the frequency-encoding dimension (readout) is usually employed to avoid wrap-around artifacts.

- kdata_os is an oversampled kdata. The oversampled kdata has dimensions of [168,336]. The frequency-encoding dimension (second dimension) was oversampled by a factor of 2. Remove the oversampling by cropping the frequency-encoding dimension. Reconstruct the oversampled and cropped data sets. Discuss the effect of oversampling.