Computational MR imaging
Laboratory 4: Reconstruction of non-Cartesian k-space data

Report is due on Wednesday the week after the lab session at 23:59. Send your report by email to Bruno Riemenschneider (bruno.riemenschneider@fau.de) and Florian Knoll (florian.knoll@fau.de).

Learning objectives

- Reconstruct non-Cartesian MRI data using gridding and NUFFT toolbox
- Apply gridding operations: density compensation, oversampling and deapodization
- Learn to use the NUFFT toolbox

1. **Radial sampling pattern:** Load radial_data.mat (variable) and plot the acquired k-space. Each column corresponds to the readout dimension for each radial line. This data was acquired with a radial acquisition using a golden angle increment (111.246117975°). Generate a sampling trajectory that corresponds to this data for the reconstruction. **Figure 1** shows a plot of the first 10 spokes of such a trajectory for reference. If the matrix size for Cartesian imaging is 384x384, what is the number of radial lines corresponding to the Nyquist rate?

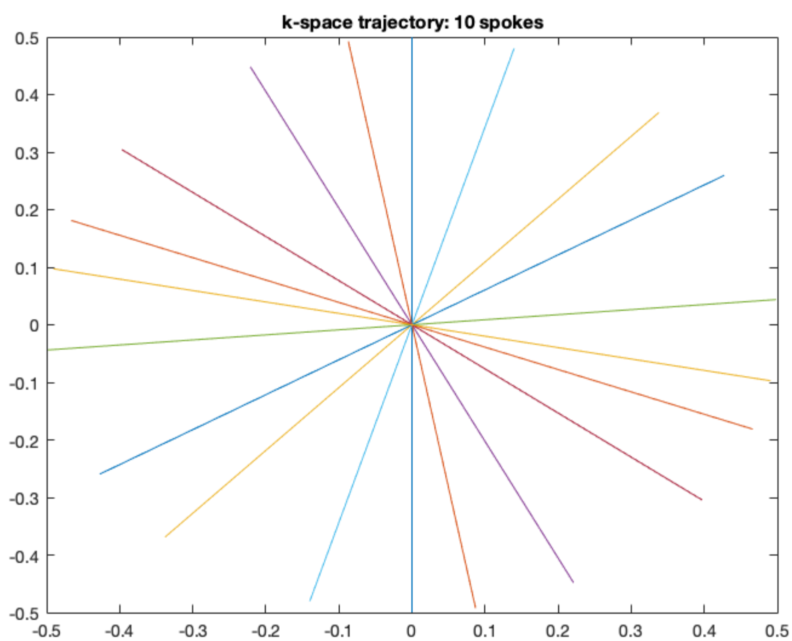

Figure 1: Radial trajectory with golden angle increment. Note that the first angle is at pi/2.

2. **Basic gridding reconstruction:** Reconstruct this dataset using the provided grid1 function that grids 2D non-Cartesian k-space data to Cartesian k-space data using triangular gridding kernel of width 2:

   Matlab version:
   function m = grid1(d,k,n)

```
% Input:
% d: non-Cartesian k-space data
% k: non-Cartesian k-space trajectory, each k(i) is a complex variable
% where the real part is the kx coordinate and the imaginary part is the
% ky coordinate.
% n: Cartesian grid size, e.g. [128x128]
% Output:
% m: gridded k-space data [nxn]
```

Python version:

```
def grid(d, k, n):
  """Grid non-cartesian kspace data to a cartesion grid
  Keyword Arguments:
    d - 2D numpy array, non-cartesian kspace
    k - 2D numpy array, kspace trajectory, scaled -0.5 to 0.5
    n - int, grid size
  Returns:
    2D numpy array (n, n)
```

Use a 384x384 Cartesian grid. Comment on the artifacts. Can you guess what organ was imaged?

3. **Density compensation**: Reconstruct the radial dataset from part 2 using a ramp filter. Plot your results. Do you need to employ oversampling and de-apodization on this dataset? Explain your answer.

4. **Oversampling**: Grid the decimated data from part 3 using oversampling factors of 1.5 and 2, apply inverse FFT and crop in the image domain.

5. **De-apodization**: Compute the de-apodization function in the image domain and apply to the gridded image with oversampling of 2.

6. **NUFFT toolbox**: Reconstruct the radial dataset using a widely used NUFFT toolbox from the research community. Plot your reconstructions and compare them with gridding reconstruction using the triangular kernel.
    a. Matlab users: You will use the NUFFT toolbox from Jeffrey Fessler (University of Michigan). The toolbox is included as a zip file in the lab folder. You will have to include the folder in the Matlab path and then build a NUFFT object:
    FT = NUFFT(data, density_compensation, 1, 0, grid_size, 2);
    This constructor initializes a NUFFT with a Kaiser-Bessel kernel.
    You can then access the forward and adjoint NUFFT by calling FT and FT'.
    b. Python users: You will use the Torch KB-NUFFT toolbox from Matt Muckley (Facebook AI research). This toolbox is modeled after Jeffrey Fessler's toolbox. You can get it from github, and documentation in provided there: https://github.com/mmuckley/torchkbnufft