HL2011: Exercise Set 4
Parallel Imaging

Andreas Sigfridsson, Peter Nillius, Alexander Fyrdahl

1 Preparations

If you haven't done it already, download the package `seemri.zip` from Bilda and unzip it to a suitable directory. Start Matlab and add the path to seemri using the matlab command `addpath` <*path to seemri*>. In matlab, type `help seemri` and make sure you get the help page. You are now ready to start.

Define some useful constants:

```
gammabar = 42.58e6;
gamma = 2*pi*gammabar;
```

2 Brain Imaging

Use your developments in the previous exercise.

The function `brain_4mm_pixel` works as `seemri` but without the ImagingVolume and without options (i.e. no plotting), see `help brain_4mm_pixel`. The object is about 220 mm wide and the pixel size is 4 mm. **Q1. Make a pulse sequence to image the object and show the results.**

Image the object at a higher resolution, pixel size 2 mm, using the function `brain_2mm_pixel` or `brain_halfmm_pixel`. Remember to change your k-space sampling grid to fit the higher resolutions (i.e. 2 mm-pixels and 0.5 mm-pixels respectively). These simulations might take a long time depending on your Matlab version. **Q2. Show the results for the 2 mm resolution (0.5 mm optional).**

3 Parallel Imaging

Download the file `brain_4mm_pixel_coil.zip` from Canvas and extract the contents somewhere in your MATLAB path (for example the current working directory).

The function `brain_4mm_pixel_coil` works just like the `brain_4mm_pixel`, but will add a third dimension to S which represents two different receiver coils, and a new data structure C which contains the coil sensitivity maps for the local reception field of the receiver coils.

Modify the imaging sequence from the previous task to only acquire every second phase encoding step. Make sure the number of phase encoding steps is divisible by two! Note, the pixel size for this exercise is 4 mm, remember to adjust your parameters accordingly. Note that the frequency encoding direction should be fully sampled.

The multidimensional S can be reconstructed like this

```
[nx,ny,nc] = size(S);
for c = 1:nc
    img(:,:,c) = mrireconstruct(S(:,:,c), max(kmaxx,kmaxy), 'Plot', false);
end
```

Multi-coil images are often combined by squaring the signal from each coil, taking the sum over the coil dimension and finally taking the square-root of the image. Colloquially, this is called a "sum-of-squares" reconstruction. The command `sum(x,DIM)` is helpful here.

Q3: Display the sum-of-squares reconstruction of the coil images. Can you see the folding caused by the reduced field of view?

3.1 SENSE parallel imaging reconstruction

One of the most common methods for parallel imaging is called Sensitivity Encoding (SENSE). When the FOV is reduced, the image will fold in over itself. The purpose of SENSE is to unfold the image.

For every voxel in the folded coil images P, we can calculate the corresponding voxel positions, A and B, in the final (unfolded) image R. By considering the aliased voxels in the folded image and the receiver coils sensitivities S_1 and S_2 at positions A and B, we can construct a system of equations as

$$P_1 = R_A S_{1,A} + R_B S_{1,B} \tag{1}$$

$$P_2 = R_A S_{2,A} + R_B S_{2,B} \tag{2}$$

Since we have two equations and two unknowns, we can solve for R_A and R_B. By repeating this process for all voxels in the image, we arrive at the unfolded image.

Note: This problem is rather poorly conditioned, so it is best solved using the pseudo-inverse method provided. In this example we FFT-shift the image to make the calculation of A and B more straightforward.

```
img = fftshift(img,2);
sense_img = [];

for jj=1:nx
    for ii=1:ny
        AB =    % size(AB) = [1 2]
        S =     % size(S)  = [2 2]
        P =     % size(P)  = [2 1]
        sense_img(jj,AB) = pinv(S')*P;
    end
end
```

Q4: Present your unfolded image.

4 Report

Submit an individual report and source code (as separate PDF and .m files) on Canvas by April 30 23:59, or before 10:15 to count as attendance for the session.