

ASSIGNMENT 10: GRADIENT FIELD, FAT NULLING AND MATLABGRADIENT FIELD VS MAIN FIELD B_0

1. Verify that if the gradient field \mathbf{G} is much smaller than the main field \mathbf{B}_0 , then we can ignore the components of the gradient field which do not point in the z -direction. From geometric considerations, it is obvious that the gradient will not have a significant effect on the direction in which the overall field points; the only major effect is that it strengthens or weakens the \mathbf{B}_0 field. So the problem can be restated as follows:

Suppose that $\mathbf{G}_\lambda(x, y, z) = \lambda(\ell_x, \ell_y, \ell_z)$ and $\mathbf{B}_0 = (0, 0, b_0)$. Define

$$\mathbf{B}_\lambda = \mathbf{B}_0 + \mathbf{G}_\lambda(x, y, z).$$

Use the Taylor expansion of $\|\mathbf{B}_\lambda\|$ to show that, as $\lambda \rightarrow 0$, the terms involving ℓ_x and ℓ_y go to zero much faster than the term involving ℓ_z .

FAT NULLING

2. An inversion-recovery sequence provides images with high T_1 contrast. Another use of inversion recovery sequence is to null a particular T_1 species by timing the 90° readout to occur at the moment when M_z crosses the transversal (or $M_z = 0$) axis. This is done by setting

$$1 - 2e^{-TI/T_1} + e^{-TR/T_1} = 0, \quad (1)$$

(this is step (7) in the sequence we studied in class, so we are making $M_z = 0$). Recall that TI is the time between 180° pulse and 90° , and TR is the repetition time (e.g., between two consecutive 180° or between two consecutive 90° pulses).

a. Solve for TI .

b. Next, suppose that $TR \gg T_1$, simplify the expression for TI .

c. Since fat is often quite bright in conventional saturation-recovery images due to its relatively short T_1 , inversion-recovery ‘nulling’ (1) has been used for fat suppression in images. Suppose $\mathbf{B}_0 = 1.5T$. From the table for T_1 , find the value of T_1 for fat (at this strength of the main magnetic field). Let the repetition time $TR = 800ms$. Find what TI should be to cause the M_z -component of fat to be zero at the moment of the 90° pulse (step (7) in the inversion recovery sequence) - that’s when ideal readout should be happening. Give the answer in ms.

MATLAB

Displaying images. The best option for displaying images is “`imshow`”. If your image is in the variable “`image`”, then to display it, type

```
>> imshow(abs(image), [ ])
```

This command automatically scaled the image from the image minimum to maximum. If one wants to specify the range explicitly, then insert

```
[win_min win_max]
```

into the brackets. This accentuates the part of the image that is of interest.

Don't use "image" function, it does a poor job of displaying images.

When displaying images, always right titles and if possible give specifics on the axis, or ranges used. (For more on this, see the first MATLAB assignment).

Orientation. In displaying medical images, there are conventions about the orientations the images should be presented.

- For axial slices, the image is oriented as it would appear if you were looking at the subject from the feet: the subject's left is on the right side of the image, and their front is up.

- For a sagittal slice, you are viewing the subject from their left side, so up is up, the subject's front is to the left, and back is to the right.

- For coronal slices, you are viewing the subject from the front, and up is up, and left is right. When displaying MR images, it is important to get the orientation right: the radiologist, or, say a neurosurgeon, will get quite confused if the image is flipped or rotated.

To flip an image, there are several choices. First one is to transpose, i.e., flip an image about the diagonal:

```
>> image_trans = image.');
```

For an image of 256×256 size, one can flip the image left-right and up-down by indexing the image in the reverse order:

```
>> image_tb_flip=image(256:-1:1,:);
```

```
>> image_lr_flip=image(:,256:-1:1);
```

To rotate an image, one can use the function "imrotate" - it rotates an image in the counter-clockwise direction ('ang' is in degrees):

```
>> image_rot = imrotate(image, ang);
```

To save an image, you can for example save it in pdf format:

```
>> print -dpdf filename.pdf
```

This will produce a file in the current directory. You can also create an eps file the same way.

Centering MR data. In medical image reconstruction, the origin for both the k -space data and the reconstructed image is the center of the image or the data array. The usual convention for the 'fft' is that the origin is at the beginning of the array, or the corner of a 2d array. To do a centered 'fft', do an 'fftshift' both before and after the 'fft':

```
>> image = fftshift(fft2(fftshift(d)));
```

here, 'd' is the data either loaded from a data file.mat or read from a binary file, see more on this below.

It is useful to define a MATLAB m-file (call it fft2c.m) that does this automatically:

```
function image=fft2c(d)
% image = fft2c(d)
%
% this function fft2c performs a centered fft2
%
image=fftshift(fft2(fftshift(d)));
```

The lines which start with % are comments that are printed when you ask for help

```
>> help fft2c
```

Binary data. To open a file with binary data use the command ‘fopen’ and then read the file into a variable, say ‘d’, or whatever is specified in the file description:

```
>> fid = fopen('filename','r');
>> d=fread(fid, [2 inf], 'float');
```

The last line reads the ‘fid’ data into 2 columns of “infinite” length (this means till the end of the file). You can also specify which format the data is in (in the above example it is ‘float’). Use help of ‘fread’ or ‘fopen’ to get more choices on reading formats and organizing data into a matrix or a string.

Binary data I have is written in pairs (real, imaginary), 8 bytes per pair. So if you read the data as above, then you need to organize it into a complex number, for example,

```
>> d1=d(1,:)+i*d(2,:);
```

Now you will have just one column array of complex data.

You will need to organize it into a matrix of given size. Let’s say that the image size is $N \times M$, then

```
>> for i=1:N
for k=1:M
data(i,k)=d1(M*(i-1)+k);
end
end
```

will produce a matrix with complex entries of the specified size.

Multiple coil data. A typical MR scan (of one slice) can be acquired by multiple coils surrounding the subject. In the data given there are 8 coils. The data for each coil is written one after the other. Therefore, you will need to create 8 matrices of data, for example, use the above procedure 8 times (but give different names to the matrices, for example, data1, data2, data3, ...). You might want to create an m-file for this too. Make sure you organize indexes properly, for example, for the 5th coil

```
>> data5(i,k)=d1(N*M*4+384*(i-1)+k);
```

Now you need to put this data together. Make sure you orient each one properly and center them. Now you can try adding or adding the absolute values of all (notice the difference). The best is to use the least squares (try it if you know it).

Assignment. There are 2 folders Data 1 and Data 2 after Assignment 10. The first folder contain data in mat files. You can just load these into MATLAB. All files are 256×256 matrices. When loading you will have a variable ‘d’ that contains the data in each file, unless specified separately:

```
se_t1_sag_data.mat
```

This is a sagittal T1-weighted midline slice of the head of a normal volunteer. White matter is brighter than gray matter. Fat is very bright due to its short T1. CSF (cerebrospinal fluid) is dark due to its very long T1. This is acquired by spin-echo pulse sequence.

```
fse_t1_ax_data.mat
```

This is an axial T1-weighted shot of the head at the level of the eyes. Same contrast as the previous shot. This is acquired by fast spin-echo pulse sequence.

`fse_t2_ax_data.mat`

This is an axial T2-weighted shot of the head at the level of the eyes. Gray matter is brighter than white matter. CSF is bright due to its long T2. This is acquired by fast spin-echo pulse sequence.

`gre_axial_head_ip.mat`

This is a gradient echo scan. The echo timing has been chosen so that water and fat are in phase. The data is in the variable `d_ip`.

`gre_axial_head_op.mat`

This is a gradient echo scan. The echo timing has been chosen so that water and fat are 180° out of phase. The data is in the variable `d_op`.

The second folder Data 2 contains raw (binary) data (it is large!) in file 't1flair_8coils'. It is the k -space data from 8 different coils for a single image. The k -space data is a 384x384 matrix, and is stored as mentioned before in (real,imaginary) pairs (8 bytes per pair). Each coil is written after the next. The total file size is: $384*384*8\text{coils}*8\text{bytes}/\text{complex number} = 9437184$ bytes. You need to open and read it as binary raw data described above.

For the MATLAB assignment:

- obtain an image from each data file
- compare with a similar one (T1 and T2 weighted, or e.g. water and fat in and out of phase)
- for the 8 coil data, reconstruct each coil separately and then the (total) image
- determine how sparse images are (from both folders): compress each image by 2d fft to 1%, 5%, 10%, 15% and 20% and compare with the 'ideal' image.

(Notice that if you plot the k -space data, it will be basically black except for the center area - that's where the most of the energy is.)

Write the MATLAB part in a pdf file (including the code and short discussion) and send that to me by e-mail.