2023–2024 Spring Semester

Term Project Report

EEE474: Foundations of Magnetic Resonance Imaging

Measurement of the Main Field (B_0) Inhomogeneity in the 3T Siemens Magnetom MRI Scanner

Supervised by: Prof. Ergin Atalar

Implemented by: Mohammed Abed

Introduction

In the domain of advanced medical imaging, precise calibration and maintenance of equipment play a crucial role in ensuring diagnostic accuracy and image quality. A major step in the calibration process is to define the quantities of interest in order to measure them and correct for them through different procedures. In a magnetic resonance imaging (MRI) scanner, one of the primary components is the main magnet formed by superconducting coils. It generates the main magnetic field (B_0) that aligns the nuclear spins within the object of interest, and establishes the basic condition of the nuclear magnetic resonance as the field strength determines the famous Larmor frequency at which the nuclei resonate. Off-resonance is a common phenomena in MRI, which is potentially caused by the B_0 field inhomogeneity, the chemical shift, or the susceptibility-induced field variations [1]; such disturbances of the off resonance can degrade the quality of imaging or provide additional information that can be useful in certain contexts [2]. This project focuses primarily on the measurement of the B_0 field inhomogeneity in the 3T Magnetom scanner. Accurate measurement and correction of these inhomogeneities are essential for improving image clarity, ensuring the diagnostic validity, and optimizing the performance of high-field MRI systems. The method used to quantify the field inhomogeneity involves analyzing the phase evolution between different echo times (TE) accumulated by spins. This study aims to reconstruct the phase and magnitude data acquired from the scanner after imaging a testing phantom using a labeled gradient echo pulse sequence with three different TE's. Once the phase was reconstructed, phase unwrapping algorithms were used to make the distribution of the phase continuous, which gave rise to constructing the B_0 field map and observing the field inhomogeneity in parts per million (ppm) over the diameter of the spherical volume (DSV). The results showed that 3T Siemens Magnetom scanner field has a homogeneity of < 1.5 ppm over the 40 cm DSV.

Methodology

1. Pulse Sequence Selection and Design

While Spin Echo (SE) sequences are generated through the use of paired radiofrequency (RF) pulses, a gradient echo (GRE) is created using a single RF pulse accompanied by a gradient reversal. This property of GRE sequences makes them superior to SE sequences in terms of the acquisition time; they are technically faster in acquiring scans and the time is significantly reduced when compared to the SE as the echo is recorded shortly after the single RF pulse is applied. More importantly, the reversal gradient applied in the GRE refocusses only the dephased spins due to the gradient itself [3]. In other words, the phase shift due to the main field inhomogeneity is not canceled as in the case of SE [4]. As a result, the phase shift between echos can be recorded for each pixel of the image, and using the appropriate relation between the phase difference and echo time difference, the field inhomogeneity can be calculated for each pixel.

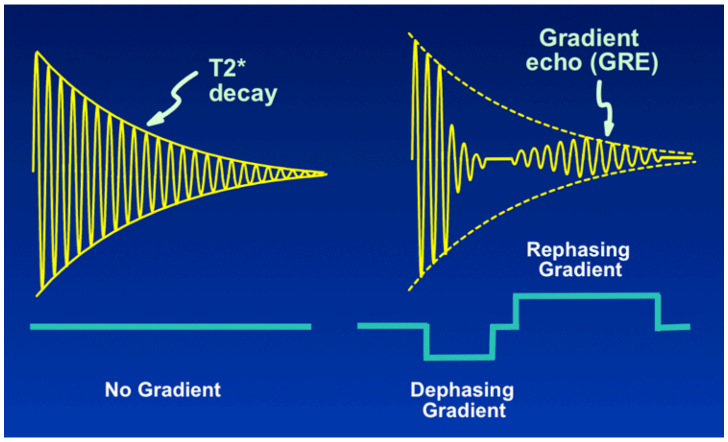

Fig. 1: From Free Induction Decay (FID) to Gradient Echo (GRE)

The phantom was scanned using a labeled GRE sequence 3 times with different echo times:

$TE_1 = 5$ ms $TE_2 = 12$ ms $TE_3 = 20$ ms

The other parameters specified in the sequence were picked as follows:

Field of view (FoV) = 224 mm

Resolution: $N_x = N_y = 256$

Flip angle (α) = 10°

Number of slices (N) = 1

Slice thickness = 3 mm

Repetition time (TR) = 100 ms

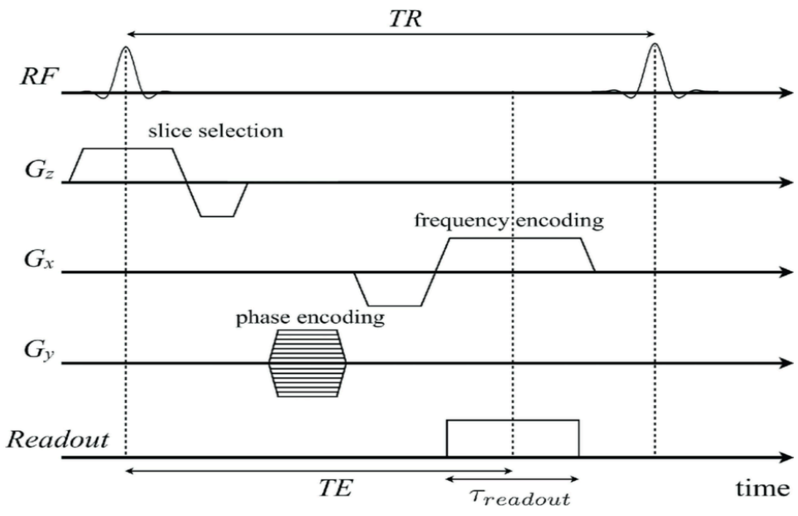

Fig. 2: Gradient Echo Timing Diagram

2. Magnitude and Phase Images Reconstruction

The phantom was scanned using the labeled GRE sequence and the Siemens 32-channel head coil was used to collect the data to increase the signal-to-noise (SNR) ratio [5]. The magnitude images were reconstructed for each acquisition using the sum of squares (SoS) method, and the phase images were reconstructed by averaging the phase images for each acquisition using the averaging method, but the resultant phase was wrapped between [-π,π]. In order to correct for this a phase unwrapping algorithm was used as will be explained in the next section.

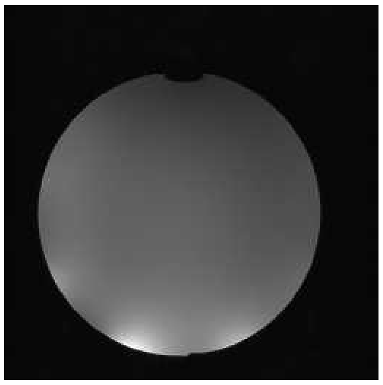

Fig. 3: Magnitude Image (TE_1 = 5 ms)

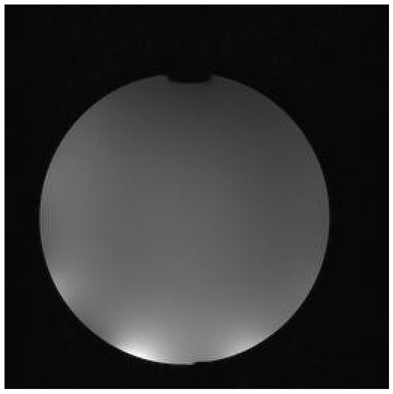

Fig. 4: Magnitude Image (TE_2 = 12 ms)

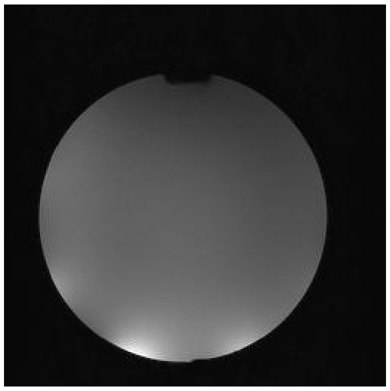

Fig. 5: Magnitude Image (TE_2 = 20 ms)

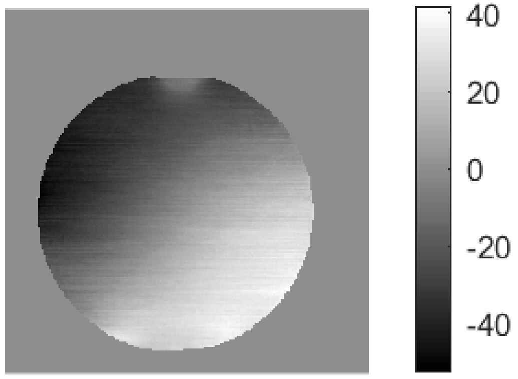

Fig. 5: Phase Image (TE_1 = 5 ms)

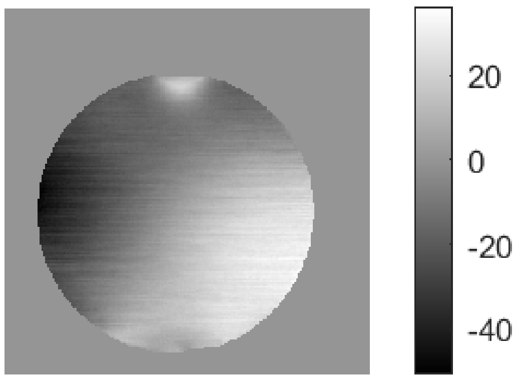

Fig. 7: Phase Image (TE_2 = 12 ms)

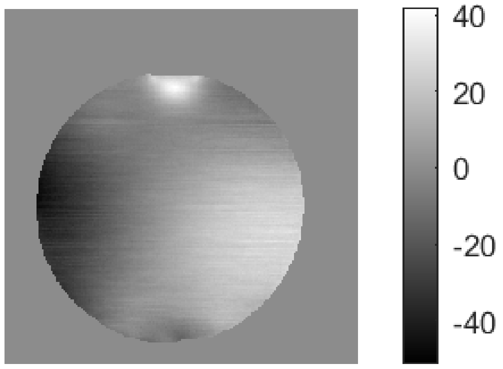

Fig. 8: Phase Image (TE_2 = 20 ms)

The timing diagrams and the k-space associated with the GRE sequences implemented on MATLAB using the Pulseq toolbox can be seen in appendix B.

3. Phase Unwrapping

Due to the lack of two dimensional phase unwrapping functionality in MATLAB, third party algorithms were used to implement the unwrapping. The method used was the fast two-dimensional phase unwrapping algorithm based on sorting by reliability [6]. This method proved to be very powerful in handling MRI images due to its:

- Robustness: it manages to handle phase discontinuities better than traditional algorithms.
- Efficiency: it has fast processing time (around 0.5 seconds) for typical images.
- Effectiveness: it's proven through real-world tests and simulations to manage clean and noisy data.

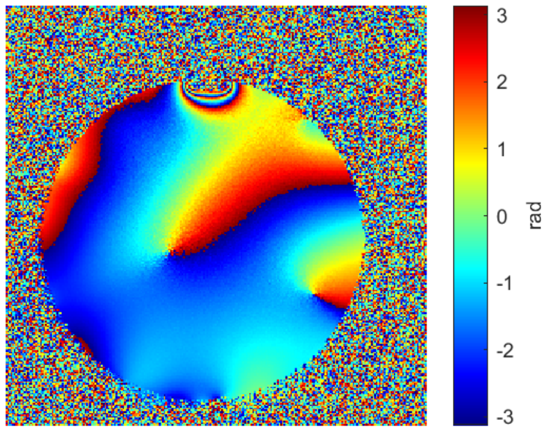

Fig. 9: Wrapped Phase Evolution

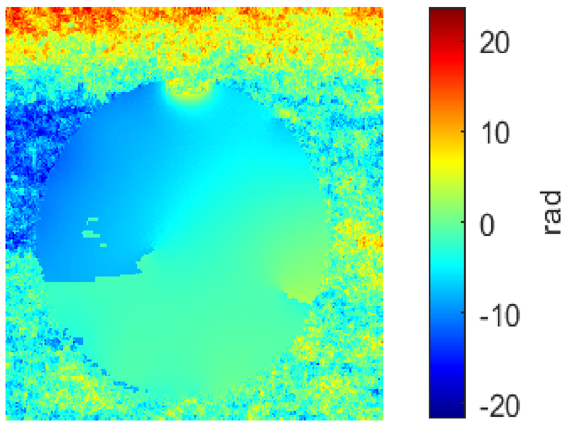

Fig. 10: Unwrapped Phase Evolution

The phase evolution between two echo times was reconstructed, and the difference between the wrapped and the unwrapped cases was observed to illustrate the importance of applying the unwrapping algorithm. As seen in Fig. 9, the phase distribution for the wrapped case is very discontinuous, especially around the top part of the phantom, and the background is extremely noisy. However, after applying the unwrapping algorithm, the phase looks more continuous as can be seen in Fig. 10. Moreover, the noise of the phase image background decreased significantly, and the actual phase shift values are reflected after the unwrapping instead of being restricted to the range of $[-\pi,\pi]$.

This step was essential in obtaining accurate B_0 field maps as will be illustrated in the following sections.

4. B_0 Field Map Generation

The first step to generate the B_0 field map was to apply the Hermitian inner product (HiP) over the complex signals I_j acquired from different channels to generate the phase evolution between echoes n and m, Θ^d [5]:

$$\Theta\left(\vec{r}, n, m\right) = \angle\left[\sum_j I_j\left(\vec{r}, TE_n\right) I_j\left(\vec{r}, TE_m\right)\right]$$

The field map is obtained using the unwrapped phase difference between different echo times and employing a weighting factor W^d:

$$W(\vec{r}, k) = \frac{M(\vec{r}, k)^2}{M(\vec{r}, k)^2 + M(\vec{r}, 1)^2}$$

Finally, the expression for the B_0 field map is obtained:

$$\Delta B(\vec{r}) = \frac{1}{2\pi\gamma} \frac{\sum_{k=2}^{K} \Theta(\vec{r}, n, m)\left(TE_k - TE_1\right) . W(\vec{r}, k)}{\sum_{k=2}^{K}\left(TE_k - TE_1\right)^2 . W(\vec{r}, k)}$$

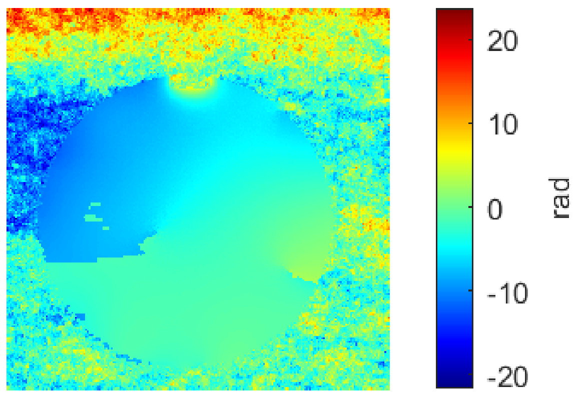

Fig. 11: Phase Evolution between TE_1 and TE_2

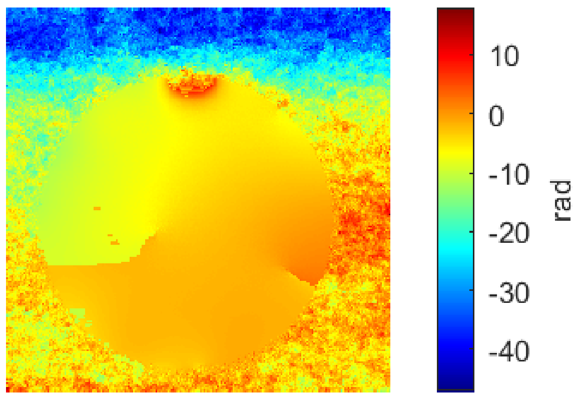

Fig. 12: Phase Evolution between TE_1 and TE_2

Fig. 11 and 12 show the phase unwrapped evolution between different echoes which were used together with the respective weighting factors to construct the B_0 field map that characterizes ΔB_0.

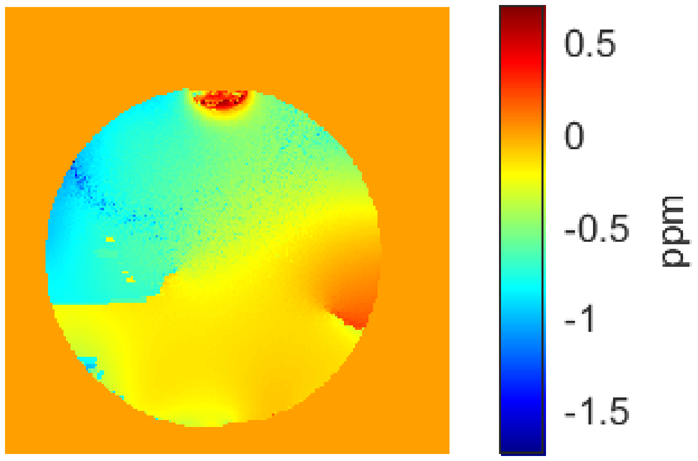

Fig. 13: B_0 Field Map (Masked)

Finally, the B_0 field map which characterizes the main field inhomogeneities was constructed, as seen in Fig. 13. The results show that the main field of the 3T scanner has an inhomogeneity of the range [-5,2]μT, and when converting that to the standard unit of measuring the homogeneity of the field, which is parts per million, the result is in the range of [-1.5, 0.5] ppm.

YouTubeVideo Link: https://youtu.be/CkeuGSBnlC0

Conclusion

In conclusion, this term project has successfully addressed the challenge of measuring the main magnetic field (B_0) inhomogeneity in the 3T Siemens Magnetom MRI scanner. Through a systematic approach involving gradient echo pulse sequences, phase image reconstruction and the application of advanced phase unwrapping algorithms, the B_0 field maps were generated and analyzed accurately. The project's finding confirmed that the field inhomogeneity of the scanner is within a precise range of [-1.5, 0.5] ppm over a 40cm diameter spherical volume, emphasizing the effectiveness of the methods carried out and the high degree of the homogeneity achieved by the scanner. Some of the improvements that could be adopted to enhance the accuracy of the measurement would be: acquiring the phase maps directly from the scanner, using the FSL toolbox to unwrap the phase, optimizing the choice of echo times to avoid phase unwrapping, and incorporating the coil sensitivity in the reconstruction algorithm. Under the guidance of Prof. Ergin Atalar, this study sets a foundational step towards optimizing MRI performance and reliability.

References

[1] D. G. Nishimura, *Principles of Magnetic Resonance Imaging*, 1.1st ed. Lulu, 2010, pp. 127-128.

[2] M. W. Haskelon, J.-F. Nielsen, and D. C. Noll, "Off-resonance artifact correction for MRI: A review," *NMR in Biomedicine*, vol. 36, 2023. doi: 10.1002/nbm.4867.

[3] “Importance of Field Homogeneity,” *Questions and Answers in MRI*. [Online] Available: https://mriquestions.com/why-homogeneity.html.

[4] “GRE v SE,” *Questions and Answers in MRI*. [Online] Available: https://www.mriquestions.com/gre-vs-se.html.

[5] S. D. Robinson, K. Bredies, et al., “An illustrated comparison of processing methods for MR phase imaging and QSM: combining array coil signals and phase unwrapping,” *NMR in Biomedicine*, vol. 30, 2016, doi: https://doi.org/10.1002/nbm.3601.

[6] M. A. Herráez, D. R. Burton, M. J. Lalor, and M. A. Gdeisat, “Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path,” *Applied Optics*, vol. 41, no. 35, p. 7437, Dec. 2002, doi: https://doi.org/10.1364/ao.41.007437.

Appendix A

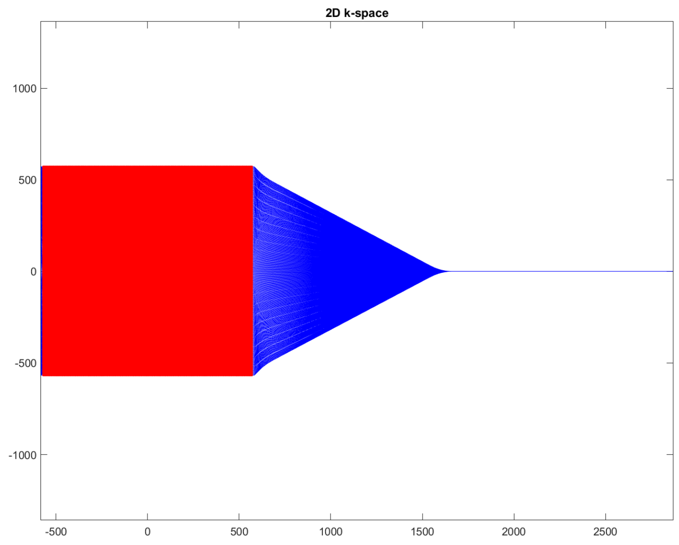

Fig. 1: 2D K-space of the GRE

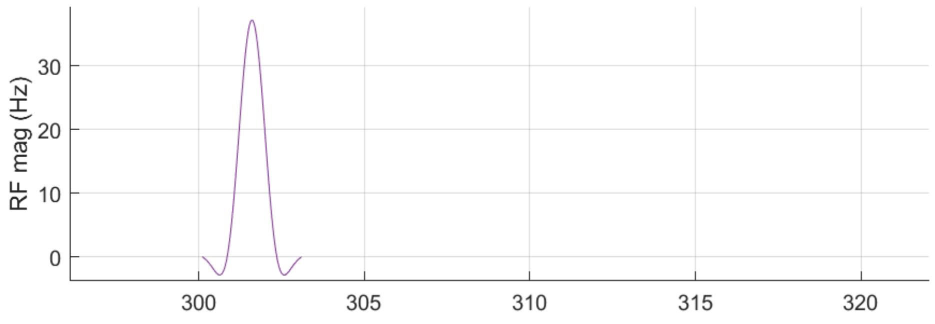

Fig. 2: Radio-frequency (RF) Pulse

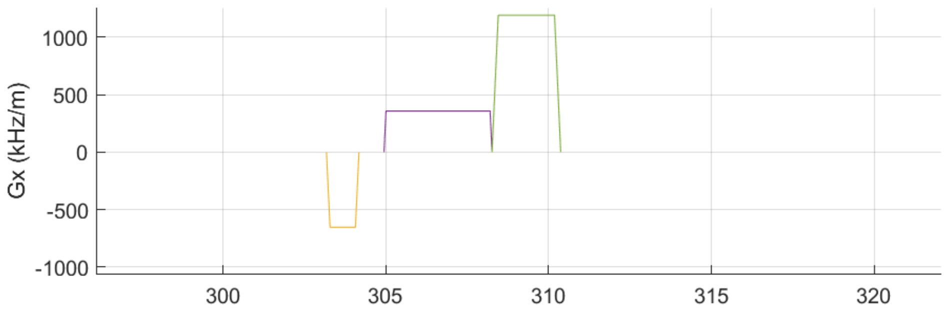

Fig. 3: Frequency Encoding Gradient (G_x)

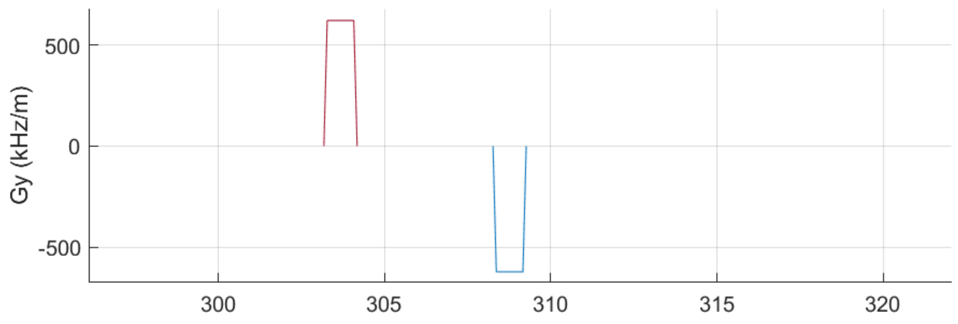

Fig. 4: Phase Encoding Gradient (G_y)

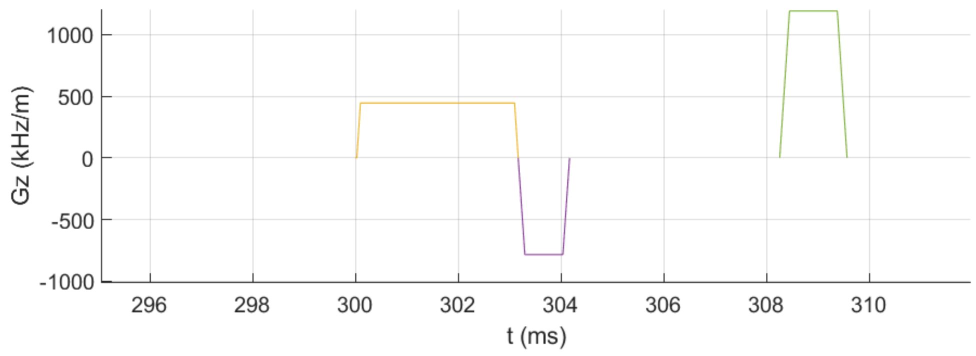

Fig. 6: Slice-Selection and Refocusing Gradient (G_z)

Appendix B

Labeled Gradient Echo Sequence MATLAB Code:

```matlab
sys = mr.opts('MaxGrad', 28, 'GradUnit', 'mT/m', ...
  'MaxSlew', 150, 'SlewUnit', 'T/m/s', 'rfRingdownTime', 20e-6, ...
  'rfDeadTime', 100e-6, 'adcDeadTime', 10e-6);
seq=mr.Sequence(sys);         % aeq object
fov=224e-3; Nx=256; Ny=Nx; % field of view
alpha=10;                  % flip angle
thickness=3e-3;            % thickness of the slice
Nslices=1;
TR=100e-3;
TE=5e-3;
rfSpoilingInc=117;
roDuration=3.2e-3;
%G_z gradient (slice selection)
[rf, gz] = mr.makeSincPulse(alpha*pi/180,'Duration',3e-3,...
  'SliceThickness',thickness,'apodization',0.42,'timeBwProduct',4,'system',sys);
%other gradients (freq. encoding and phase encoding)
deltak=1/fov;
gx = mr.makeTrapezoid('x','FlatArea',Nx*deltak,'FlatTime',roDuration,'system',sys);
adc = mr.makeAdc(Nx,'Duration',gx.flatTime,'Delay',gx.riseTime,'system',sys);
gxPre = mr.makeTrapezoid('x','Area',-gx.area/2,'Duration',1e-3,'system',sys);
gzReph = mr.makeTrapezoid('z','Area',-gz.area/2,'Duration',1e-3,'system',sys);
phaseAreas = -((0:Ny-1)-Ny/2)*deltak; % phase area should be Kmax for clin=0 and -Kmax for
clin=Ny... strange
%spoiling
gxSpoil=mr.makeTrapezoid('x','Area',2*Nx*deltak,'system',sys);
gzSpoil=mr.makeTrapezoid('z','Area',4/thickness,'system',sys);
% sequence timing
delayTE=ceil((TE - mr.calcDuration(gxPre) - gz.fallTime - gz.flatTime/2 ...
  - mr.calcDuration(gx)/2)/seq.gradRasterTime)*seq.gradRasterTime;
```

```
delayTR=ceil((TR - mr.calcDuration(gz) - mr.calcDuration(gxPre) ...
  - mr.calcDuration(gx) - delayTE)/seq.gradRasterTime)*seq.gradRasterTime;
assert(all(delayTE>=0));
assert(all(delayTR>=mr.calcDuration(gxSpoil,gzSpoil)));
rf_phase=0;
rf_inc=0;
seq.addBlock(mr.makeLabel('SET','REV', 1));
%%
for s=1:Nslices
  rf.freqOffset=gz.amplitude*thickness*(s-1-(Nslices-1)/2);
  for i=1:Ny
    for c=1:length(TE)
      rf.phaseOffset=rf_phase/180*pi;
      adc.phaseOffset=rf_phase/180*pi;
      rf_inc=mod(rf_inc+rfSpoilingInc, 360.0);
      rf_phase=mod(rf_phase+rf_inc, 360.0);
      %
      seq.addBlock(rf,gz);
      gyPre =
mr.makeTrapezoid('y','Area',phaseAreas(i),'Duration',mr.calcDuration(gxPre),'system',sys);
      seq.addBlock(gxPre,gyPre,gzReph);
      seq.addBlock(mr.makeDelay(delayTE(c)));
      seq.addBlock(gx,adc);
      gyPre.amplitude=-gyPre.amplitude;
      spoilBlockContents={mr.makeDelay(delayTR(c)),gxSpoil,gyPre,gzSpoil}; % here we
demonstrate the technique to combine variable counter-dependent content into the same block
      if c~=length(TE)
        spoilBlockContents=[spoilBlockContents {mr.makeLabel('INC','ECO', 1)}];
      else
        if length(TE)>1
          spoilBlockContents=[spoilBlockContents {mr.makeLabel('SET','ECO', 0)}];
          end
          if i~=Ny
              spoilBlockContents=[spoilBlockContents {mr.makeLabel('INC','LIN', 1)}];
          else
              spoilBlockContents=[spoilBlockContents {mr.makeLabel('SET','LIN', 0),
mr.makeLabel('INC','SLC', 1)}];
          end
        end
        seq.addBlock(spoilBlockContents{:});
      end
    end
end
%% Timing Check
[ok, error_report]=seq.checkTiming;
if (ok)
  fprintf('Timing check passed successfully\n');
else
  fprintf('Timing check failed! Error listing follows:\n');
  fprintf([error_report{:}]);
  fprintf('\n');
end
%%
seq.setDefinition('FOV', [fov fov thickness*Nslices]);
seq.setDefinition('Name', 'gre_lbl');
seq.write('gre_lb_1.seq')
%%
seq.plot('timeRange', [0 32]*TR, 'TimeDisp', 'ms', 'Label', 'LIN,SLC');
[ktraj_adc, t_adc, ktraj, t_ktraj, t_excitation, t_refocusing] = seq.calculateKspacePP();
% k-spaces
figure; plot(t_ktraj, ktraj');
hold; plot(t_adc,ktraj_adc(1,:),'.');
```

```
title('k-trajectory with respect to time');
figure; plot(ktraj(1,:),ktraj(2,:),'b');
axis('equal');
hold;plot(ktraj_adc(1,:),ktraj_adc(2,:),'r.');
title('2D k-space');
```

Reconstruction, Image Processing, and B_0 Field Map Generation MATLAB code:

```
clear all; close all; clc
% Load data
twix_obj_1 = mapVBVD('C:\Users\Mohammed
Karem\Desktop\Spring23-24\EEE474\Project\gre_scans\meas_MID83_GRE_lb_1_FID8266.dat'
);
twix_obj_2 = mapVBVD('C:\Users\Mohammed
Karem\Desktop\Spring23-24\EEE474\Project\gre_scans\meas_MID84_GRE_lb_2_FID8267.dat'
);
twix_obj_3 = mapVBVD('C:\Users\Mohammed
Karem\Desktop\Spring23-24\EEE474\Project\gre_scans\meas_MID85_GRE_lb_3_FID8268.dat'
);
%% Visualize coil phase images
do_unwrapping = false;
plot_all_channels = false;
data_1 = twix_obj_1.image.unsorted();
data_1 = permute(data_1, [1, 3, 2]);
nCoils = size(data_1, 3);
data_2 = twix_obj_2.image.unsorted();
data_2 = permute(data_2, [1, 3, 2]);
data_3 = twix_obj_3.image.unsorted();
data_3 = permute(data_3, [1, 3, 2]);
% to become compatible with the ICE reco the exmple sequence now uses an
% inverted phase encoding ordering, so a reversal of the dimension is needed
data_1 = flip(data_1, 2);
data_2 = flip(data_2, 2);
data_3 = flip(data_3, 2);
images_1 = zeros(size(data_1));
images_2 = zeros(size(data_2));
images_3 = zeros(size(data_3));
avg_1 = zeros(size(data_1, [1,2]));
```

```
avg_2 = zeros(size(avg_1));
avg_3 = zeros(size(avg_1));
if nCoils > 1 && plot_all_channels
   figure
end
%
https://www.mathworks.com/matlabcentral/fileexchange/45393-phase-unwrapping-using-recursi
ve-orthogonal-referring-puror
debug_PUROR = 0;
th4unwrap = 0.0;
th4supp = 8.0;
th4stack = 16.0;
for ii = 1:nCoils
   images_1(:,:,ii) = ifftshift(ifft2(data_1(:,:,ii))).';
   images_2(:,:,ii) = ifftshift(ifft2(data_2(:,:,ii))).';
   images_3(:,:,ii) = ifftshift(ifft2(data_3(:,:,ii))).';
   if nCoils > 1
       if do_unwrapping == false
           angle_1 = angle(images_1(:,:,ii));
           angle_2 = angle(images_2(:,:,ii));
           angle_3 = angle(images_3(:,:,ii));
       else
           th4noise_1 = NoiseEstimation(images_1(:,:,ii));
           [mask4unwrap_1, mask4supp_1, mask4stack_1] = mask_generation(images_1(:,:,ii),
th4noise_1, th4unwrap, th4supp, th4stack);
           angle_1 = PUROR2D(images_1(:,:,ii), mask4unwrap_1, mask4supp_1, mask4stack_1,
debug_PUROR );
           th4noise_2 = NoiseEstimation(images_2(:,:,ii));
           [mask4unwrap_2, mask4supp_2, mask4stack_2] = mask_generation(images_2(:,:,ii),
th4noise_2, th4unwrap, th4supp, th4stack);
```

```
            angle_2 = PUROR2D(images_2(:,:,ii), mask4unwrap_2, mask4supp_2, mask4stack_2,
debug_PUROR );
            th4noise_3 = NoiseEstimation(images_3(:,:,ii));
            [mask4unwrap_3, mask4supp_3, mask4stack_3] = mask_generation(images_3(:,:,ii),
th4noise_3, th4unwrap, th4supp, th4stack);
            angle_3 = PUROR2D(images_3(:,:,ii), mask4unwrap_3, mask4supp_3, mask4stack_3,
debug_PUROR );
        end
        avg_1 = avg_1 + 1/nCoils*angle_1;
        avg_2 = avg_2 + 1/nCoils*angle_2;
        avg_3 = avg_3 + 1/nCoils*angle_3;
        if plot_all_channels
            subplot(2,2,1)
            imshow(angle_1, []); colorbar; colormap('jet')
            title(['TE1 RF Coil ' num2str(ii)]);
            subplot(2,2,2)
            imshow(angle_2, []); colorbar; colormap('jet')
            title(['TE2 RF Coil ' num2str(ii)]);
            subplot(2,2,3)
            imshow(angle_3, []); colorbar; colormap('jet')
            title(['TE3 RF Coil ' num2str(ii)]);
            pause(0.05)
        end
        %         imwrite(tmp, ['img_coil_' num2str(ii) '.png'])
    end
end
% imshow(abs(image1(:,:,ii)), []);
% image1 = ifftshift(ifft2(sos));
%% Average combination
figure
if do_unwrapping == false
```

```
    subplot(2, 2, 1)
    imshow(avg_1,[]); colorbar; colormap('hsv')
    title('TE1 Combined Phase')
    subplot(2, 2, 2)
    imshow(avg_2,[]); colorbar; colormap('hsv')
    title('TE2 Combined Phase')
    subplot(2, 2, 3)
    imshow(avg_3,[]); colorbar; colormap('hsv')
    title('TE3 Combined Phase')
else
    subplot(2, 2, 1)
    imshow(avg_1.*mask,[]);
    colorbar
    % colormap('jet')
    % title('Phase Image (TE')
    subplot(2, 2, 2)
    imshow(avg_2.*mask,[]);
    colorbar;
    % colormap('jet')
    %title('TE2 Combined Phase')
    subplot(2, 2, 3)
    imshow(avg_3.*mask,[]);
    colorbar;
    %colormap('jet')
    %title('TE3 Combined Phase')
end
% imwrite(sos, ['my_se3.png'])
%% Estimate Theta
theta = zeros(size(images_1, 1), size(images_1, 2), 3);
theta(:,:,2) = dot(images_1, conj(images_2), 3);
theta(:,:,3) = dot(images_1, conj(images_3), 3);
```

```
if do_unwrapping == false
  theta = angle(theta);
else
  th4noise = NoiseEstimation(theta);
  [mask4unwrap, mask4supp, mask4stack] = mask_generation(theta, th4noise, th4unwrap,
th4supp, th4stack);
  theta = PUROR2D(theta, mask4unwrap, mask4supp, mask4stack, debug_PUROR );
end
figure
subplot(2,1,1)
imshow(theta(:,:,2), []); cb=colorbar; colormap('jet');cb.Label.String  = 'rad';
%title('Theta n=1 m=2')
subplot(2,1,2)
imshow(theta(:,:,3), []); cb=colorbar; colormap('jet'); cb.Label.String  = 'rad';
%title('Theta n=1 m=3')
%% Sum of squares combination --> W
M = zeros(size(images_1, 1), size(images_1, 2), 3);
M(:,:,1) = abs(sqrt(sum((images_1).^2, 3)));
M(:,:,2) = abs(sqrt(sum((images_1 - images_2).^2, 3)));
M(:,:,3) = abs(sqrt(sum((images_1 - images_3).^2, 3)));
figure
subplot(2,2,1)
imshow(M(:,:,1), []); colorbar
title('Combined image for TE1')
subplot(2,2,2)
imshow(M(:,:,2), []); colorbar
title('Combined image difference between TE1 and TE2')
subplot(2,2,3)
imshow(M(:,:,3), []); colorbar
title('Combined image difference between TE1 and TE3')
% imwrite(sos, ['my_se3.png'])
% Estimate W
W = M.^2 ./ (M.^2 + M(:,:,1).^2);
%% Estimate Delta B
gamma = 42.58e6; %Hz/T
TE1 = 5e-3; %sec
TE2 = 12e-3; %sec
TE3 = 20e-3; %sec
B0 = 3; %T; Siemens
numerator = theta(:,:,2) .* (TE2 - TE1) .* W(:,:,2);
numerator = numerator + theta(:,:,3) .* (TE3 - TE1) .* W(:,:,3);
denomenator = (TE2 - TE1).^2 .* W(:,:,2);
denomenator = denomenator + (TE3 - TE1).^2 .* W(:,:,3);
DeltaB = 1/(2*pi*gamma) .* numerator ./ denomenator; %T
DeltaB = DeltaB / B0 * 1e6; %ppm
% Average all images to boost SNR
avg_sos = (abs(sqrt(sum((images_1).^2, 3))) + abs(sqrt(sum((images_2).^2, 3))) +
abs(sqrt(sum((images_3).^2, 3))))/3;
avg_sos = avg_sos ./ max(avg_sos(:)); % normalize for effective binarization
bin_sos = imbinarize(avg_sos, graythresh(avg_sos)/4); % Otsu threshold
% imshow(avg_sos, []); colorbar
% imshow(bin_sos, []); colorbar
% some image processing to make the mask "cleaner"
se = strel('disk', 50); % need this for imclose to work
mask = bwareaopen(bin_sos, 10); % remove small speckles
mask = imclose(mask, se); % remove holes
% imshow(mask, []); colorbar
figure
subplot(2,2,1)
imshow(DeltaB, []); cb = colorbar; colormap('jet'); cb.Label.String  = 'ppm';
%title('\Delta B')
subplot(2,2,4); colormap('gray')
```

```
imshow(mask, []); colorbar
title('Mask')
subplot(2,2,2)
imshow(DeltaB.*mask, []); cb = colorbar; colormap('jet'); cb.Label.String  = 'ppm';
%title('\Delta B (masked)')
```