Faculty of Engineering,
Cairo University

SBE 311

MRI

Bloch Equation Simulation

Authors:
Ali Gamal Ahmed
Khaled Maher
Nada Ashraf
Sarah Mohammad

1 Bulk Magnetization Vector

1.1 Rotation Function

```
T1 = 0.5;  % relaxation time for Mz
T2 = 0.5;  % decay time for Mx and My
f = 20 ;  % larmor frequency
t = linspace(0,3*T1,1000); % determine time steps for plotting

Mz = (1-exp(-t/T1)); % Eq of Mz
Mx = exp(-t/T2).*(sin(2*pi*f.*t)); % Eq of Mx
My = exp(-t/T2).*(cos(2*pi*f.*t)); % Eq of My

figure,
view([3,3,2]); % set the view point of plot

axis([-max(Mx) max(Mx) -max(My) max(My) 0 max(Mz)]); % set axis limits
curve = animatedline('LineWidth' , 0.002 , 'Color','c'); % define
    animated curve to plot the route

hold on
grid on

for i=1:length(t)
addpoints(curve ,Mx(i),My(i),Mz(i)); % add point to the curve
h2 = plot3([0,Mx(i)],[0,My(i)],[0,Mz(i)],'r','LineWidth',2); % plot
    magnetization vector
h3 = plot3([0,Mx(i)],[0,My(i)],[0,0],'b','LineWidth',2); % plot xy
    component (transverse component) of magnetization vector
h4 = plot3([0,0],[0,0],[0,Mz(i)],'k','LineWidth',2); % plot z component
    (longtudinal component) of magnetization vector
drawnow % add frames to update figure
pause(0.01)
delete(h2) % deletion for update with new values
delete(h3)
delete(h4)
end
```

Included within the project folder in the Bloch Equation Simulation.m file. Simply run this in any MATLAB environment.

1.2 Bulk Magnetization Trajectory

Incase your pdf reader does not support video and other multimedia, the video will also be included within the project folder.

2 Fourier transform & K-space domain plots

2.1 Running The Program

To run the GUI that enables interaction with our tool, simply open the Fourier-TransformTool folder included within the project and run main.py

Dependencies needed to run main.py:

- pyqt5
- scipy
- matplotlib
- cv2
- numpy

2.2 Python Code Snippets

```
def load_image(self):
   self.load_img =QtWidgets.QFileDialog.getOpenFileName(None, "Open File")
   self.image=cv.cvtColor(cv.imread(self.load_img[0]), cv.COLOR_BGR2GRAY)


   if self.load_img[0]:
      if self.loadCheck == 0:
         self.widget.show()
         self.widget.setImage(((self.image)).T)
         self.comboBox.setEnabled(True)
         self.loadCheck=0
         self.dft = np.fft.fft2(self.image)
         self.real = np.real(self.dft)
         self.imaginary =1j*np.imag(self.dft)
         self.magnitude = np.abs(self.dft)
         self.phase = np.angle(self.dft)
```

Image loading function opens a file browser and allows for loading of images, then starts calculating the fourier transform of the loaded picture after which it calculates it's real, imaginary, magnitude and phase components.

```
   def select_parameter(self,currentIndex):
      self.dft = np.fft.fft2(self.image)
      self.real =np.real(self.dft)
      self.imaginary =1j*np.imag(self.dft)
      self.magnitude = np.abs(self.dft)
      self.phase = np.angle(self.dft)

      self.image_Data=[(20*np.log(np.fft.fftshift(self.magnitude))),((self.phase)),(20*np.
      self.widget_2.show()
      self.widget_2.setImage((self.image_Data[self.comboBox.currentIndex()]).T)
```

Parameter selection function allows us to define a certain component of the image to select and display on our image widget.

The complete code is included within the FourierTransformTool folder.

2.3 Screenshots

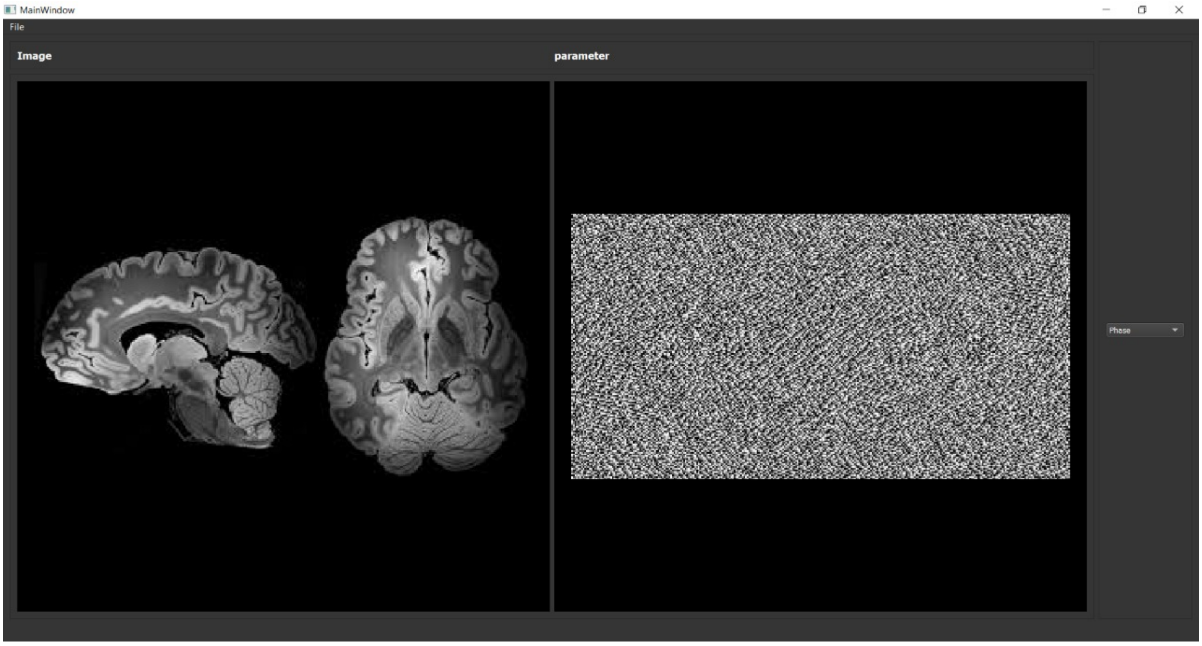

Figure 1: Phase of a selected image

Figure 2: Magnitude of a selected image

3 Uniformity Effect

3.1 MATLAB Function

```
z = (3-2.5).*rand(100,1) + 2; % create random values for B vector ranges
    from 2.5 to 3 tesla
x = (100-1).*rand(100,1) + 1; % creating random points of x
y = (100-1).*rand(100,1) + 1; % creating random points of y

xlin = linspace(min(x),max(x),33);
ylin = linspace(min(y),max(y),33);
[X,Y] = meshgrid(xlin,ylin);

f = scatteredInterpolant(x,y,z);
Z = f(X,Y);

figure
mesh(X,Y,Z) %interpolated
axis tight; hold on
stem3(x,y,z,'.','MarkerSize',15) %non uniform plot
```

Included within the project folder in the Uniformity Effect.m file. Run in any MATLAB environment.

3.2 Uniformity Effect Plot

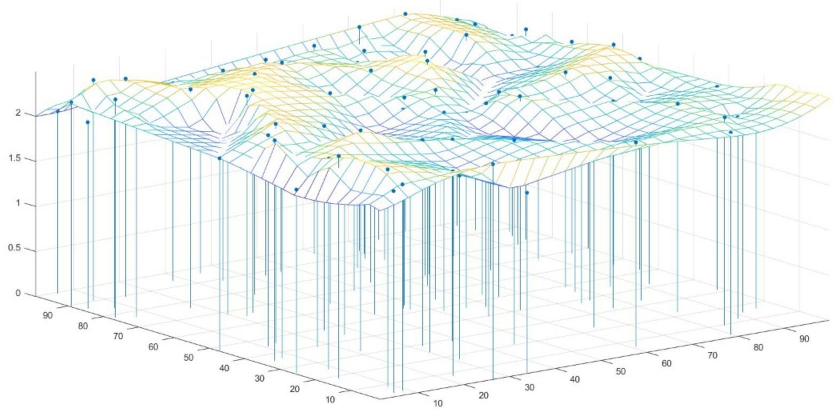

Figure 3: Uniformity Effect Plot