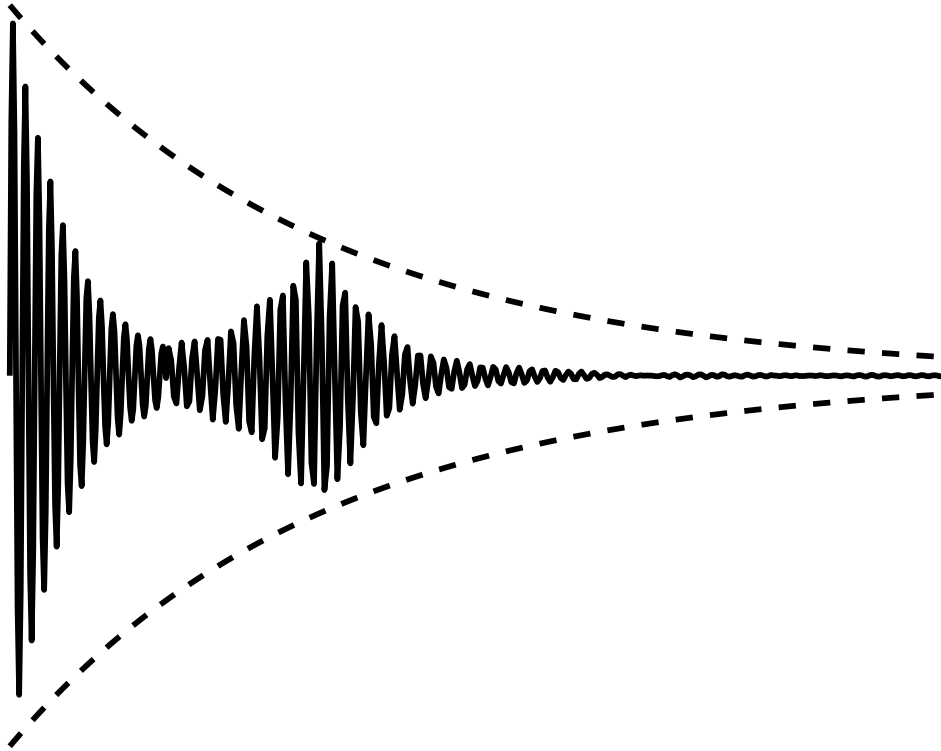


---

# Coursework 1

---



UNIVERSITY COLLEGE LONDON

MEDICAL PHYSICS & BIOMEDICAL ENGINEERING DEPARTMENT

MRES - COURSEWORK 1

COMP0121 - COMPUTATIONAL MRI

AUTUMN TERM

18TH APRIL 2021

*Author:*

Mr. Imraj SINGH (SN: 20164771)

*Module lead:*

Dr. Gary ZHANG

*This report along with the zip file containing animations make up the coursework 1 submission.*

## Contents

---

<b>1 Spin Excess</b>	<b>1</b>
<b>2 Forced precession with an on-resonance RF field</b>	<b>1</b>
<b>3 Free precession in the main static magnetic field.</b>	<b>2</b>
<b>4 Free induction decay and Inversion Recovery.</b>	<b>3</b>
<b>5 Spin echo</b>	<b>4</b>
<b>A Problem 1 code:</b>	<b>6</b>
<b>B Problem 2 code:</b>	<b>6</b>
B.1 Rotation matrix z-axis: . . . . .	6
B.2 Rotation matrix x-axis: . . . . .	6
B.3 Rotation matrix y-axis: . . . . .	6
B.4 Animations of rotations: . . . . .	7
B.5 Problem 2.4: . . . . .	10
<b>C Problem 3 code:</b>	<b>12</b>
C.1 Problem 3.1: . . . . .	12
C.2 Problem 3.2: . . . . .	14
C.3 Problem 3.3: . . . . .	16
C.4 Problem 3.4: . . . . .	18
C.5 Problem 3.4: . . . . .	20
C.6 Problem 3.5: . . . . .	22
C.7 Comparison between 3.4 and 3.5: . . . . .	24
<b>D Problem 4 code:</b>	<b>25</b>
D.1 Problem 4.1: . . . . .	25
D.2 Problem 4.2: . . . . .	28
<b>E Problem 5 code:</b>	<b>32</b>
E.1 Problem 5.1: . . . . .	32
E.2 Problem 5.2: . . . . .	36
E.3 Problem 5.3: . . . . .	40
E.4 Problem 5.3 components plot: . . . . .	46
E.5 Problem 5.4: . . . . .	49

## 1 Spin Excess

Spin excess is given by the following equation:

$$\text{Spin excess} \approx N \frac{\hbar \omega_o}{2kT} \quad (1)$$

The factors involved in the calculation of the spin excess are:  $\hbar$  is reduced Planck's constant which is Planck's constant ( $h$ ) divided by  $2\pi$ , it is a universal constant;  $\omega_o$  is the Larmor frequency and is dependent on the gyromagnetic ratio  $\gamma$  and external static magnetic field  $B_o$  ( $\omega_o = \gamma B_o$ ); gyromagnetic ratio is dependent on the proton of interest, this is typically a hydrogen proton;  $k$  is the Boltzmann's constant;  $T$  is the absolute temperature;  $N$  is the total number of spins in a sample.

Eqn. (1) can be thought of as a ratio between the quantum energy difference ( $\hbar \omega_o$ ) to the thermal energy ( $kT$ ), or spin excess is proportional to quantum energy difference and inversely proportional to thermal energy. Explicitly, the quantum energy difference is the energy difference between the parallel alignment to  $B_o$  versus anti-parallel alignment to  $B_o$ , where the parallel state is the lower energy state. The thermal energy is directly proportional to  $T$ . The quantum energy difference is directly proportional to  $B_o$ . Therefore, given a measurement of the same protons in the same sample (to ensure  $N$  and  $\gamma$  are constant), increasing  $B_o$  and/or decreasing  $T$  will increase the spin excess.

## 2 Forced precession with an on-resonance RF field

The following problem is answered by first defining the three dimensions  $\mathbb{R}^3$  in the vector form  $\mathbf{x} = [x, y, z]'$  with the associated unit vectors  $\hat{x} = [1, 0, 0]'$ ,  $\hat{y} = [0, 1, 0]'$  and  $\hat{z} = [0, 0, 1]'$ . The purpose of the rotation vectors are to rotate a vector define in  $\mathbf{x}$  about each of the axes.

The rotation around any one of the axes corresponds to a positive clockwise rotation, by  $\theta$ , on the orthogonal plane when viewing the plane from the negative direction of the axis. Intuitively, using your left-hand closing it into a fist with your thumb pointing out, the thumb represents the direction of the axis and the curl of your fingers represent a positive clockwise rotation on the orthogonal plane. Visualisations for the  $z$ ,  $x$  and  $y$  axis rotations are given by the animations '2\_1', '2\_2' and '2\_2', respectively.

**2.1 Rotation around the z-axis.** The rotation matrix around the  $z$ -axis is defined as:

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

To verify Eqn. (2) consider the unit vectors  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  defined previously. The unit vectors are rotated by  $\theta$  clockwise around the  $z$ -axis to produce  $\hat{x}_{rot}$ ,  $\hat{y}_{rot}$  and  $\hat{z}_{rot}$ . The rotated unit vectors are calculated from  $\hat{x}_{rot} = \mathbf{R}_z(\theta) \cdot \hat{x}$ ,  $\hat{y}_{rot} = \mathbf{R}_z(\theta) \cdot \hat{y}$  and  $\hat{z}_{rot} = \mathbf{R}_z(\theta) \cdot \hat{z}$ . Multiplying the rotation matrix by the unit vectors we find:  $\hat{x}_{rot} = [\cos \theta, -\sin \theta, 0]'$ ,  $\hat{y}_{rot} = [\sin \theta, \cos \theta, 0]'$   $\hat{z}_{rot} = [0, 0, 1]'$ . These results show that rotating the vector preserves the magnitude whilst changing the  $x$ - $y$  components,  $\|\hat{x}_{rot}\| = 1$ ,  $\|\hat{y}_{rot}\| = 1$  and  $\|\hat{z}_{rot}\| = 1$ . There is a reciprocal relationship between the  $x$ - $y$  components of the vector when rotated. Furthermore, a positive  $\theta$  results in a clockwise rotation on the  $x$ - $y$  plane;  $\hat{x}_{rot}|\theta=90^\circ = [0, -1, 0]'$  and  $\hat{y}_{rot}|\theta=90^\circ = [1, 0, 0]'$ .

**2.2 & 2.3 Rotation around the  $x$ -axis and  $y$ -axis.** The following matrices correspond to rotations around the  $x$ -axis, Eqn. (3), and  $y$ -axis, Eqn. (4).

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad (3) \quad \mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (4)$$

Similar to problem 2.1 rotating around the  $x$  and  $y$  axes preserve the magnitude of the vector being rotated, whilst ensuring a positive  $\theta$  corresponds to a clockwise rotation around the axis of rotation.

**2.4 Forced precession..** The effects of  $T_1$  and  $T_2$  relaxation are ignored as the RF pulse is on a much smaller timescale than relaxation. Parameters given in the question are:  $\Delta\theta = 90^\circ$  flip angle, that the pulse is on-resonance and along the  $\hat{x}'$  axis in the rotating frame of reference (FoR), and the duration of the pulse  $\tau = 1$  ms. This information gives an insight into the dynamics. The on-resonance condition dictates that the applied RF frequency matches the Larmor frequency, meaning the resulting magnetic field  $B_1$  is maximally synchronized to tip the spin around the  $x'$ -axis. The duration of pulse and flip angle dictate the  $B_1$  strength as:  $B_1 = \frac{\Delta\theta}{\gamma\tau} = 5.9\mu\text{T}$ .

In the rotating FoR the flip of the spin is a  $90^\circ$  clockwise rotation around the  $x$ -axis. This means that the magnetisation vector is confined to the  $y$ - $z$  plane. This is visualised by animation 2\_4.

### 3 Free precession in the main static magnetic field.

In order to visualise the precession of the spins as they relax it was necessary to set a fictitious Larmor frequency much lower than reality. In reality typical MRI scanners have a static magnetic field strength of the order  $B_0 \approx \mathcal{O}(10^1)$  T, most clinical scanners are 1.5 or 3 T. The gyromagnetic ratio of protons are of order  $\gamma \approx \mathcal{O}(10^8)$  rad/(sT), hydrogen protons are  $2.68 \times 10^8$  rad/(sT). As  $\omega_0 = \gamma B_0$ , therefore  $\omega_0 \approx \mathcal{O}(10^9)$ . This has the consequence of many, many precessions per second meaning visualisation would be saturated by these spins. This is further compounded as the time scale for relaxation of the spin toward  $B_0$  is over a lot longer time scales than the Larmor frequency.  $T_1$  and  $T_2$  relaxations are the longitudinal and transverse relaxations, respectively. Where the effect of relaxation is included, the values are  $T_1 = 2$  s and  $T_2 = 1$  s for all problems. These values were chosen as they were of similar order to reality (seconds to milliseconds depending on the material), additionally  $T_1$  is typically longer than  $T_2$ . It was found that a Larmor frequency of  $\omega_0 = 4\pi$  gave good visualisation, this corresponded to two rotations a second. The initial magnetization all of problem 3 was  $M(0) = [0, 1, 0]'$ . These dynamics are prescribed by the fact these problems pertain to only free precession (no RF pulse), and there is only a main static magnetic field (only  $B_0$  no  $B_1$ ).

The Bloch equations are the set of ordinary differential equations (ODE) that govern this system. This ODE has analytic solutions that will be used when solving the following problems. The solution consists of a rotation around the  $z$ -axis and exponential decays.

The transverse magnetisation is given as  $M_\perp$  and consists of the  $x$  and  $y$  components of the magnetisation. The parallel magnetisation is given by  $M_\parallel$  and is solely the  $z$  component of magnetisation.

**3.1 Free precession with no relaxation..** As the initial magnetization vector is on the  $x$ - $y$  plane and there is no relaxation, the free precession will include no  $z$  component. This is seen in animation '3.1' where there is a clockwise precession on the  $x$ - $y$  plane, with two rotations per second, and no  $z$  component, as expected. This is because without relaxation the governing Bloch equations collapse to simply a reciprocal relationship between the  $x$  and  $y$  components that corresponds to a rotation of the magnetisation vector around the  $z$ -axis. The solution is  $M(t) = \mathbf{R}_z(\omega_0 t) \cdot M(0)$ . This is within the laboratory FoR.

**3.2 Free precession with relaxation..** Including the relaxation provides more interesting, and more realistic dynamics. As the analytical solutions to the Bloch equations dictate that the transverse magnetisation exponentially decays to zero given  $\exp(-\frac{t}{T_2})$ . Furthermore, over time the parallel magnetisation decays toward one, toward the direction of the static magnetic field. The result is a precessing spin at the Larmor frequency, that moves toward  $M(t = \infty) = [0, 0, 1]'$ , as the transverse components decay and parallel component increase to align with the static magnetic field. This is visualised in animation '3.2'. Additionally, a stand-alone visualisation of the 3D magnetisation vector trajectory is given in '3.2.3D'.

**3.3 Rotating FoR.** In the rotating FoR the laboratory FoR axes rotate around the  $z$ -axis at the Larmor frequency. In the rotating FoR the circular precession of the spin at the Larmor frequency in the  $x$ - $y$  plane is effectively halted, as these dynamics are captured by the fact that the FoR itself is rotating. Hence, problem 3.1 in the rotating frame corresponds to  $M' = [0, 1, 0]'$ , the magnetisation

vector is static in the rotating FoR. As this is a trivial solution it is not visualised. The solution to problem 3.2 in the rotating FoR is restricted to the  $y'$ - $z'$  plane as, again, the rotation at the Larmor frequency is captured by the rotating FoR. This is visualised in animation '3\_3'.

**3.4 Isochromat precession faster than Larmor frequency.** If the precession is faster than the Larmor frequency ( $\omega_0 + \Delta\omega$ ) then in the rotating FoR the rotation is slower than the precession. This means that even though the rotating FoR is rotating at a similar frequency, the simulated spin will spin at the difference between the rotating FoR and the actual spin of the isochromat. Therefore the solution in the rotating FoR will be the same as visualised in '3\_3' but with a  $x$ - $y$  component due to a rotation of  $\mathbf{R}_z(\Delta\omega t)$ . This is a clockwise rotation as  $\Delta\omega$  is positive. This is seen in animation '3\_4'. In the laboratory FoR the solution will be similar to '3\_2' but with slightly more rotation around the  $z$ -axis at any given time..

**3.5 Isochromat precession slower than Larmor frequency.** This solution is very similar to the previous, the difference being that the precession is at a frequency of  $\omega_0 - \Delta\omega$ . Again  $\Delta\omega$  is positive, so the solution is identical in the rotating FoR although the rotation around the  $z$ -axis is anti-clockwise. In the laboratory FoR the solution is similar but with slightly less rotation around the  $z$ -axis at any given time. This is visualised in animation '3\_5'.

A comparison of problem 3.4 and 3.5 in the laboratory frame of reference is given in animation '3\_45\_comp'.

## 4 Free induction decay and Inversion Recovery.

The sequences are best simulated and visualised using the magnetisation vector and transverse components. It is important to note that the transverse components are proportional to the signal, such that no transverse magnetisation will result in no signal. The reason for this is that the magnetisation of each individual spin is extremely small. In the receive coils a changing magnetic field induces a current which can be measured, the Larmor frequency produces a very quickly changing magnetic field in the transverse plane, and the fact that it changes so quickly aids its detection. Additionally there are many spins in a given volume and this aids detection too.

For this problem the Larmor frequency for visualisation is a lot lower than reality,  $\omega_0 = 50\pi$ . This Larmor frequency will allow for the visualisation of the magnetisation vector and signal for the rest of the problems. For this problem we are only concerned with one isochromat, that the spins held within all precess at the same Larmor frequency.

**4.1 Free induction decay (FID) sequence,** The sequence consists of two parts:

1. Flip the spins from alignment with  $B_0$  direction by  $\pi/2$  radian ( $90^\circ$ ), using a RF pulse, to the transverse plane.
2. Free precession with relaxation for spin from the transverse plane to  $B_0$  direction.

The time over which the flip occurs and the time over which relaxation occurs are orders of magnitudes different. This makes visualisation difficult. In order to capture the dynamics correctly the simulation was carried out in the rotating FoR. If simulation was done in the laboratory FoR either two Larmor frequencies (one for each part of the sequence) would have to be used or the time over which the flip takes place would have to be a lot longer. If not, then as the fictitious Larmor frequency used is so low it would look like there was little to no precession around the  $z$ -axis during a flip. In reality there would be many many rotations around the  $z$ -axis, therefore the dynamics are not captured correctly. Using a higher Larmor frequency for the flip part of the sequence is not realistic as there should only be one Larmor frequency, also a longer flip time would mean that relaxation effects could not be ignored. Therefore a rotating FoR is used, and it should be noted that in the laboratory FoR the spins are rotating around the  $z$ -axis at the Larmor frequency. The visualisation can be seen in '4.1', the FID gives a signal as transverse magnetisation occurs.

**4.2 Inversion recovery sequence.** The sequence consists of four parts:

1. Flip the spins from alignment with  $B_0$  direction by  $\pi$  radians ( $180^\circ$ ), using a RF pulse, to the  $-B_0$  direction.
2. Free precession with relaxation for spin from the  $-B_0$  direction to  $B_0$  direction.
3. Flip the spins by  $\pi/2$  radians ( $90^\circ$ ), using a RF pulse. At the time of the flip the magnetisation vector have relaxed to either remain in the  $-B_0$  direction, have zero magnetisation, or be in the  $B_0$  direction. This will result in three possibilities when flipped by a  $\pi/2$  RF pulse along the  $x'$ -axis. The time over which this relaxation occurs is  $T_I$ .
  - (a) If in the  $-B_0$  direction, the RF pulse will flip the spin onto the  $-y'$ -axis.
  - (b) If the magnetisation is at zero, the RF pulse have no effect on the magnetisation, as the the is no parallel component to flip onto the transverse plane.
  - (c) If in the  $B_0$  direction, the RF pulse will flip the spin onto the  $y'$ -axis.
4. Where there is a transverse component, these spins will have a free precession with relaxation for spin from the transverse plane to  $B_0$  direction. If no transverse component the spin will just continue to relax along the  $z$ -axis toward the  $B_0$  direction,

Using the same reasoning as problem 4.1, the sequence was simulated in the rotating FoR. This sequence is commonly used to measure  $T_1$ . How this is done is understanding the dynamics. If  $T_I$  is such that when  $M_z(t = T_I) = 0$  no signal will be obtained by the receiver coils. This  $T_I$  can be used to find  $T_1$  by analysing the  $z$ -component of the Bloch equations:  $M_z(t) = M_z(0)e^{-t/T_1} + M_0(1 - e^{-t/T_1})$ , using the conditions that  $t = T_I$ ,  $M_z(t) = M_z(t = T_I) = 0$  (hence no signal when flipped  $\pi/2$ ),  $M_z(0) = M_z(t = 0) = -1$  (just been flipped  $\pi$ ), and that  $M_{z0} = 1$ , it can be found that  $T_I = T_1 \ln(2)$ . Three separate animations are done in order to visualise this effect of varying  $T_I$ : '4.2\_highTI' has  $T_I > T_1 \ln(2)$ ; '4.2\_lowTI' has  $T_I < T_1 \ln(2)$ ; and '4.2\_opt' has  $T_I = T_1 \ln(2)$ .

## 5 Spin echo

To simulate this sequence we deal with the rotating FoR, additionally a Larmor frequency of  $50\pi$  was used. The time scales for the RF pulses are on the order of milliseconds and are on-resonance, and therefore can be thought of approximately instantaneous with no relaxation effects. This sequence consists of five parts:

1. Flip the collection of isochromats from alignment with  $B_0$  direction by  $\pi/2$  radians ( $90^\circ$ ), using a RF pulse along the  $x'$  axis, to the  $y'$  axis.
2. Free precession with relaxation of the isochromats from the transverse plane to  $B_0$  direction. The transverse components dephase due to the collection of isochromats being of different Larmor frequencies.
3. Flip the spins by  $\pi$  radians ( $180^\circ$ ), using a RF pulse along the  $y'$  axis. This reverses the directions of the  $z$  and  $x$  components of magnetisation vectors. This happens at  $t = \tau$  where  $t$  is defined from after the initial flip.
4. The transverse components rephase and a echo is acquired,  $t = 2\tau = T_E$ .
5. The transverse components begin to dephase again,  $t = 2\tau = T_E$ .

For both problems 5.1 and 5.2, the visualisation of the transverse components of the isochromats was prioritised. Additionally, relaxation effects were only taken into account for the normalised signal. Where, as previously discussed, the signal is proportional to the magnitude of the transverse components of magnetisation,  $|M_\perp|$ . Furthermore, as the collections of spins are not all at the same frequency they will dephase. Dephasing will cause the average transverse magnetisation across all isochromats to decay faster than purely thermodynamic effects from  $T_2$ . This additional dephasing decay can

be denoted  $T_2'$ , and both effects are described by a decay time  $T_2^*$ . To account for the dephasing in the magnetisation, the average  $\frac{1}{N} \sum M_\perp$  is used to as the transverse component of magnetisation, where  $N$  is the number of isochromats in the collection and the sum is over  $N$ . Therefore the signal is proportional to  $|\frac{1}{N} \sum M_\perp|$ . This is what will be plotted to give an indication of the signal from the collection of isochromats. The number of isochromats used was 1001, but only 11 were visualised. An echo is produced at  $t = 2\tau = T_E = 1s$ , where  $t = \tau = 0.5s$  is the time between  $\pi/2$  and  $\pi$  RF pulses. The signal at  $T_E$  is not effected by dephasing and follows the  $T_2$  envelope.

**5.1 Spin echo with uniform distribution.** See animation '5.1' for visualisation of the Spin Echo sequence with uniform isochromat distribution. It can be seen that with a uniform distribution of isochromats gives rise to beats (approximately 0.25, 0.5 and 0.75 seconds), where there are momentary increases in signal due to certain isochromat transverse magnetisation components rephasing. As time goes on the effect of beats is diminished as less isochromats rephase, additionally the effect of  $T_2$  decreases beats. From the animation it is easy to see  $t = \tau$  when the  $\pi$  RF pulse occurs, and the subsequent echo at  $T_E$ .

**5.2 Spin echo with Cauchy distribution.** See animation '5.2' for visualisation of the Spin Echo sequence with Cauchy isochromat distribution. It can be seen that with a uniform distribution of isochromats does not give rise to regular beat pattern which is found with a uniform distribution. This due to the random isochromat values (which follow a Cauchy distribution) the components of the transverse magnetisation do not momentarily rephase in a uniform fashion. Instead there are random points of rephasing due to the random nature of the distribution. Again it is easy to see  $\tau$  and  $T_E$ .

**5.3 Right choice of isochromat make-up.** To choose an appropriate make up of isochromats a brief sensitivity analysis was undertaken. Where a range of  $\delta\omega$  for the uniform distribution, and a range of  $\Delta$  and  $\omega_0$  for the Cauchy distributions, were used to investigate the right choice of for isochromat make-up. These can be respectively seen in visualisations '5.3\_dOmega', '5.3\_Delta' and '5.3\_omega'.

From 5.3\_dOmega it can be seen that increasing  $\delta\omega$  decreases the amplitude of the beats, increases the frequency of beats and increases the amount of dephasing, effectively  $T_2'$  increases with  $\delta\omega$ . A value of  $\delta\omega = 16$  was deemed sufficient for visualisation of  $T_2^*$ .

From 5.3\_Delta it can be seen that increasing  $\Delta$  increase the amount of dephasing, effectively  $T_2'$  increases with  $\delta$ . At very high values of  $\Delta$  random oscillations can be seen. This is due to the nature of taking a continuous probability function and discretising it, random rephasing will occur. An infinite amount of isochromats should only decay. A value of  $\Delta = 8$  was deemed sufficient for visualisation.

From 5.3\_omega it can be seen that increasing  $\omega_0$  has no effect on the amount of dephasing. Changing values of  $\omega_0$  does not have an effect when viewing the magnitude of the average transverse components. Using the magnitude obscures the effect of  $\omega_0$ . As  $\omega_0$  is equivalent to changing the mean Larmor frequency. In 5.3\_omega\_comp the  $M_\perp = [M_{x'}, M_{y'}]'$  components are plotted and it can be seen that varying  $\omega_0$  causes the components to oscillate in the rotating FoR, see animation '5.3\_omega\_comp'. As the mean value of the distribution should be the rate at which the FoR rotates, it was chosen to keep  $\omega_0 = 0$ .

**5.4 Hahn echo.** The Hahn echo is visualised in 5.4. A Cauchy distribution of isochromats was used with  $\omega_0 = 0$  and  $\Delta = 8$ . The first two steps are the same as the Spin echo sequence. In the third step the  $\pi$  RF pulse is replaced with a  $\pi/2$  RF pulse along the  $x'$  axis. The second  $\pi/2$  pulse rotates the magnetisation vector onto the  $x'-z'$  plane. This eliminates the  $y'$  components of  $M_\perp$ , and causes  $\frac{1}{N} \sum M_\perp \approx 0$ . The spins then begin to rephase again due to there still being a  $x'$  component. At the echo time, as the  $y'$  components were eliminated, only half of the  $T_2$  relaxed signal is recovered. See animation '5.4' for visualisation of the Hahn echo sequence with Cauchy isochromat distribution. The second pulse occurs at  $t = \tau = 0.5s$ , with a half intensity echo at  $t = T_E = 1s$ .



## A Problem 1 code:

```

1 function SE = calcSE(T,B, Gamma)
2 %Function to calculate the approximate spin excess
3 % Inputs
4 % T = Temperature of the medium
5 % B = Magnetic field strength
6 % Gamma = Gyromagnetic ratio of medium
7 %
8 % Output
9 % SE = Spin excess
10
11 % Parameters
12
13 % Number of spins calculated from typical voxel volume 2*2*5mm = 0.02
14 % Avogadro's number = 6.02*10^23
15 % One voxel has 2 x 6.02 x10^23 x 0.02 / 18 protons
16 N = 1.338 * 10^21;
17
18 % Reduced planks constant
19 hbar = (6.62607015*10^(-34))/(2*pi);
20
21 % Boltzman constant
22 k = 1.38064852 * 10^(-23);
23
24 SE = N*(hbar*B*Gamma)/(2*k*T);
25 end

```

## B Problem 2 code:

### B.1 Rotation matrix z-axis:

```

1 function [rotated] = rotateZ(vector,theta)
2 %Rotate around z-axis
3 % vector - the vector you want rotated, must have three dimensions
4 % theta - angle in radians to rotate the vector around the z-axis
5 % clockwise
6 %
7
8 rotated(1) = vector(1)*cos(theta) + vector(2)*sin(theta);
9 rotated(2) = vector(2)*cos(theta) - vector(1)*sin(theta);
10 rotated(3) = vector(3);
11 end

```

### B.2 Rotation matrix x-axis:

```

1 function [rotated] = rotateX(vector,theta)
2 %Rotate around x-axis
3 % vector - the vector you want rotated, must have three dimensions
4 % theta - angle in radians to rotate the vector around the x-axis
5 % clockwise
6 %
7
8 rotated(1) = vector(1);
9 rotated(2) = vector(2)*cos(theta) + vector(3)*sin(theta);
10 rotated(3) = vector(3)*cos(theta) - vector(2)*sin(theta);
11 end

```

### B.3 Rotation matrix y-axis:



```

1 function [rotated] = rotateY(vector,theta)
2 %Rotate around y-axis
3 %   vector — the vector you want rotated, must have three dimensions
4 %   theta — angle in radians to rotate the vector around the y-axis
5 %   clockwise
6 %
7
8 rotated(1) = vector(1)*cos(theta) - vector(3)*sin(theta);
9 rotated(2) = vector(2);
10 rotated(3) = vector(3)*cos(theta) + vector(1)*sin(theta);
11 end

```

#### B.4 Animations of rotations:

```

1 % housekeeping
2 clc
3 clear
4
5 %% Rotation around the z-axis — visualisation
6 % Author — Imraj 11/11/2020
7
8 % initialise the video
9 video = VideoWriter(['2_1', '.mp4'], 'MPEG-4');
10
11 % set the frame rate
12 frameRate = 100;
13 video.set('FrameRate', frameRate);
14
15 % open video
16 video.open();
17
18 % initialise the figure
19 h = figure;
20
21 % define unit vectors
22 X = [1,0,0]';
23 Y = [0,1,0]';
24 Z = [0,0,1]';
25
26 % define angle of rotation 1 degree
27 theta = 2*pi/360;
28
29 % specify animation captures each degree of rotation
30 for i=0:360
31     quiver3(0,0,0,X(1),X(2),X(3), 'linewidth',2,'LineStyle','-', 'Color','k')
32     hold on
33     quiver3(0,0,0,Y(1),Y(2),Y(3), 'linewidth',2,'LineStyle','—', 'Color','b')
34     quiver3(0,0,0,Z(1),Z(2),Z(3), 'linewidth',2,'LineStyle','-.', 'Color','r')
35
36     % format title, legend, labels, limits, grid and box
37     title(['z-axis rotation of unit vectors, $\theta =$', num2str(theta*i
38           *360/(2*pi)), '$^\circ$'], interpreter, latex, fontsize, 15)
39     legend('$\hat{x}$', '$\hat{y}$', '$\hat{z}$', interpreter, latex, fontsize, 15)
40     xlabel(M_x, interpreter, latex, fontsize, 15)
41     ylabel(M_y, interpreter, latex, fontsize, 15)
42     zlabel(M_z, interpreter, latex, fontsize, 15)
43     grid on
44     box on
45     xlim([-1 1]);

```

```

45     ylim([-1 1]);
46     zlim([-1 1]);
47     hold off
48
49     % assign frame and write it to the video
50     frame = getframe(h);
51     video.writeVideo(frame);
52
53     % rotate by theta
54     X = rotateZ(X,theta);
55     Y = rotateZ(Y,theta);
56     Z = rotateZ(Z,theta);
57 end
58
59 video.close();
60
61 %% Rotation around the x-axis — visualisation
62 % Author — Imraj
63
64 % initialise the video
65 video = VideoWriter(['2.2', '.mp4'], 'MPEG-4');
66
67 % set the frame rate
68 frameRate = 100;
69 video.set('FrameRate', frameRate);
70
71 % open video
72 video.open();
73
74 % initialise the figure
75 h = figure;
76
77 % define unit vectors
78 X = [1,0,0]';
79 Y = [0,1,0]';
80 Z = [0,0,1]';
81
82 % define angle of rotation 1 degree
83 theta = 2*pi/360;
84
85 % specify animation captures each degree of rotation
86 for i=0:360
87     quiver3(0,0,0,X(1),X(2),X(3), 'linewidth',2, 'LineStyle', '-', 'Color', 'k')
88     hold on
89     quiver3(0,0,0,Y(1),Y(2),Y(3), 'linewidth',2, 'LineStyle', '—', 'Color', 'b')
90     quiver3(0,0,0,Z(1),Z(2),Z(3), 'linewidth',2, 'LineStyle', '-.', 'Color', 'r')
91
92     % format title, legend, labels, limits, grid and box
93     title(['x-axis rotation of unit vectors,  $\theta =$ ', num2str(theta*i
94           *360/(2*pi)), ' $^\circ$ '], interpreter, latex, fontsize, 15)
95     legend('$\hat{x}$', '$\hat{y}$', '$\hat{z}$', interpreter, latex, fontsize, 15)
96     xlabel( $M_x$ , interpreter, latex, fontsize, 15)
97     ylabel( $M_y$ , interpreter, latex, fontsize, 15)
98     zlabel( $M_z$ , interpreter, latex, fontsize, 15)
99     grid on
100    box on
101    xlim([-1 1]);
102    ylim([-1 1]);
103    zlim([-1 1]);
104    hold off

```

```

105 % assign frame and write it to the video
106 frame = getframe(h);
107 video.writeVideo(frame);
108
109 % rotate by theta
110 X = rotateX(X,theta);
111 Y = rotateX(Y,theta);
112 Z = rotateX(Z,theta);
113 end
114
115 video.close();
116
117 %% Rotation around the y-axis – visualisation
118 % Author – Imraj
119
120 % initialise the video
121 video = VideoWriter(['2.3', '.mp4'], 'MPEG-4');
122
123 % set the frame rate
124 frameRate = 100;
125 video.set('FrameRate', frameRate);
126
127 % open video
128 video.open();
129
130 % initialise the figure
131 h = figure;
132
133 % define unit vectors
134 X = [1,0,0]';
135 Y = [0,1,0]';
136 Z = [0,0,1]';
137
138 % define angle of rotation 1 degree
139 theta = 2*pi/360;
140
141 % specify animation captures each degree of rotation
142 for i=0:360
143     quiver3(0,0,0,X(1),X(2),X(3),'linewidth',2,'LineStyle','-', 'Color','k')
144     hold on
145     quiver3(0,0,0,Y(1),Y(2),Y(3),'linewidth',2,'LineStyle','—', 'Color','b')
146     quiver3(0,0,0,Z(1),Z(2),Z(3),'linewidth',2,'LineStyle','-.', 'Color','r')
147
148 % format title, legend, labels, limits, grid and box
149 title(['y-axis rotation of unit vectors', '$\theta =$', num2str(theta*i
        *360/(2*pi)), '$^\circ$'], interpreter, latex, fontsize, 15)
150 legend('$\hat{x}$', '$\hat{y}$', '$\hat{z}$', interpreter, latex, fontsize, 15)
151 xlabel(M_x, interpreter, latex, fontsize, 15)
152 ylabel(M_y, interpreter, latex, fontsize, 15)
153 zlabel(M_z, interpreter, latex, fontsize, 15)
154 grid on
155 box on
156 xlim([-1 1]);
157 ylim([-1 1]);
158 zlim([-1 1]);
159 hold off
160
161 % assign frame and write it to the video
162 frame = getframe(h);
163 video.writeVideo(frame);
164

```

```

165     % rotate by theta
166     X = rotateY(X,theta);
167     Y = rotateY(Y,theta);
168     Z = rotateY(Z,theta);
169 end
170
171 video.close();

```

## B.5 Problem 2.4:

```

1 % housekeeping
2 clc
3 clear
4 %% Forced precession ignoring relaxation
5 % Author: Imraj Singh 03/11/2020
6
7 % Given parameters
8
9 % Magnetisation aligned along z-axis initially
10 M = [0, 0, 1]';
11
12 % Duration of the RF pulse
13 t = 1*10^(-3);
14
15 % Flip angle
16 dtheta = 90*pi/180;
17
18 % Gyromagnetic ratio for proton in hydrogen
19 gamma = 2.68*10^8;
20
21 % Calculate necessary parameters
22
23 % The magnetic field strength caused by RF
24 B1 = dtheta/(gamma*t);
25
26 % Precession frequency
27 omega1 = gamma*B1;
28
29 % Define modelling parameters
30
31 % Duration and timestep of model
32 time = linspace(0,t,101);
33
34 Msoln = zeros(101,3);
35
36 Msoln(:,1) = M(1);
37 Msoln(:,2) = M(2)*cos(omega1.*time) + M(3)*sin(omega1.*time);
38 Msoln(:,3) = M(3)*cos(omega1.*time) - M(2)*sin(omega1.*time);
39
40 %% Animation
41
42 % initialise the video
43 video = VideoWriter(['2_4', '.mp4'], 'MPEG-4');
44
45 % set the frame rate
46 frameRate = 10;
47 video.set('FrameRate', frameRate);
48
49 video.open();
50

```

```

51 % initialise the figure
52 h = figure;
53
54 % specify animation captures each degree of rotation
55 for i=1:length(time)
56     subplot(2,2,1)
57     quiver3(0,0,0,0,0,1,'linewidth',2,'LineStyle','—','Color','k')
58     hold on
59     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
60     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
        'Color','r')
61
62 % format title, legend, labels, limits, grid and box
63 xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
64 ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
65 zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
66 grid on
67 box on
68 xlim([-1 1]);
69 ylim([-1 1]);
70 zlim([0 1]);
71 view(3)
72 hold off
73
74
75 subplot(2,2,2)
76 quiver(0,0,0,1,'linewidth',2,'LineStyle','—','Color','k')
77 hold on
78 plot(Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
79 quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—','Color','r')
80
81 % format title, legend, labels, limits, grid and box
82 xlabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
83 ylabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
84 grid on
85 box on
86 xlim([-1 1]);
87 ylim([0 1]);
88 hold off
89
90 subplot(2,2,3:4)
91 plot(time(1:i)*1000,Msoln(1:i,2),'linewidth',2,'LineStyle','—','Color','r');
92 hold on
93 plot(time(1:i)*1000,Msoln(1:i,3),'linewidth',2,'LineStyle','—','Color','b')
94 ;
95
96 % format title, legend, labels, limits, grid and box
97 title(['Time: ', num2str(time(i)*1000), ' ms'], interpreter, latex, fontsize, 15)
98 xlabel(Time (ms), interpreter, latex, fontsize, 15)
99 ylabel(Magnetisation, interpreter, latex, fontsize, 15)
100 legend( $M_{y'}$ ,  $M_{z'}$ , interpreter, latex, fontsize, 10)
101 grid on
102 box on
103 xlim([0 max(time*1000)]);
104 ylim([0 1]);
105 hold off
106
107 % assign frame and write it to the video
108 frame = getframe(h);
109 video.writeVideo(frame);
110 end

```

```

110
111 video.close();

```

## C Problem 3 code:

### C.1 Problem 3.1:

```

1 % housekeeping
2 clc
3 clear
4 %% Free precession ignoring relaxation
5 % Author: Imraj Singh 11/11/2020
6
7 % Given parameters
8 % Magnetisation aligned along y-axis initially
9 M = [0, 1, 0]';
10 M0 = [0, 0, 1]';
11
12 % Prescribe T1
13 T2 = 1;
14 T1 = T2 * 2;
15
16 % Calculate necessary parameters
17
18 % Larmor frequency
19 omega0 = 2 * 2 * pi;
20
21 % End time
22 t = 4;
23 resolution = 100;
24
25 % Duration and timestep of model
26 time = linspace(0, t, t * resolution + 1);
27
28
29
30 Msoln(:,1) = M(1)*cos(omega0.*time) + M(2)*sin(omega0.*time);
31 Msoln(:,2) = M(2)*cos(omega0.*time) - M(1)*sin(omega0.*time);
32 Msoln(:,3) = M(3);
33
34
35 %% Animation module
36 % Author: Imraj Singh 03/11/2020
37
38 % initialise the video
39 video = VideoWriter(['3_1', '.mp4'], 'MPEG-4');
40
41 % set the frame rate quarter of real speed
42 frameRate = resolution/4;
43 video.set('FrameRate', frameRate);
44
45 video.open();
46
47 % initialise the figure
48 h = figure;
49
50 % specify animation captures each degree of rotation
51 for i=1:length(time)
52     subplot(2,2,1)
53     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')

```

```

54     hold on
55     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
56     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
57           , 'Color','r')
58
59     % format legend, labels, limits, grid and box
60     xlabel( $M_x$ , interpreter, latex, fontsize, 15)
61     ylabel( $M_y$ , interpreter, latex, fontsize, 15)
62     zlabel( $M_z$ , interpreter, latex, fontsize, 15)
63     grid on
64     box on
65     xlim([-1 1]);
66     ylim([-1 1]);
67     zlim([0 1]);
68     view(3)
69     hold off
70
71     subplot(2,2,2)
72     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
73           , 'Color','r')
74     hold on
75     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
76     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
77
78     % format legend, labels, limits, grid and box
79     xlabel( $M_x$ , interpreter, latex, fontsize, 15)
80     ylabel( $M_y$ , interpreter, latex, fontsize, 15)
81     zlabel( $M_z$ , interpreter, latex, fontsize, 15)
82     grid on
83     box on
84     xlim([-1 1]);
85     ylim([-1 1]);
86     zlim([0 1]);
87     view(2)
88     hold off
89
90     subplot(2,2,3:4)
91     plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','—','Color','r');
92     hold on
93     plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','—','Color','b');
94
95     % format title, legend, labels, limits, grid and box
96     title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
97     xlabel(Time (s), interpreter, latex, fontsize, 15)
98     ylabel(Magnetisation, interpreter, latex, fontsize, 15)
99     legend( $M_x$ ,  $M_y$ , interpreter, latex, fontsize, 10)
100    grid on
101    box on
102    xlim([0 max(time)]);
103    ylim([-1 1]);
104    hold off
105
106    % assign frame and write it to the video
107    frame = getframe(h);
108    video.writeVideo(frame);
109
110 end
111 video.close();

```



## C.2 Problem 3.2:

```

1 % housekeeping
2 clc
3 clear
4 %% Free procession with relaxation
5 % Author: Imraj Singh 03/11/2020
6
7 % Given parameters
8 % Magnetisation aligned along y-axis initially
9 M = [0, 1, 0]';
10 M0 = [0, 0, 1]';
11
12 % Prescribe T1
13 T2 = 1;
14 T1 = T2 * 2;
15
16 % Calculate necessary parameters
17
18 % Larmor frequency
19 omega0 = 2 * 2 * pi;
20
21 % End time
22 t = 6;
23 resolution = 50;
24
25 % Duration and timestep of model
26 time = linspace(0, t, t * resolution + 1);
27
28 Msoln(:,1) = exp(-time./T2).*(M(1)*cos(omega0.*time) + M(2)*sin(omega0.*time));
29 Msoln(:,2) = exp(-time./T2).*(M(2)*cos(omega0.*time) - M(1)*sin(omega0.*time));
30 Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
31
32
33 %% Animation module
34 % Author: Imraj Singh 03/11/2020
35
36 % initialise the video
37 video = VideoWriter(['3_2', '.mp4'], 'MPEG-4');
38
39 % set the frame rate
40 frameRate = resolution/4;
41 video.set('FrameRate', frameRate);
42
43 video.open();
44
45 % initialise the figure
46 h = figure;
47
48 % specify animation captures each degree of rotation
49 for i=1:length(time)
50     subplot(2,2,1)
51     quiver3(0,0,0,0,1,0, 'linewidth',2, 'LineStyle','—', 'Color','k')
52     hold on
53     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3), 'k—', 'linewidth',2)
54     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3), 'linewidth',4, 'LineStyle','—',
55           , 'Color','r')
56
57 % format legend, labels, limits, grid and box
58 xlabel(M_x, interpreter, latex, fontsize, 15)
59 ylabel(M_y, interpreter, latex, fontsize, 15)

```

```

59     xlabel( $M_z$ , interpreter, latex, fontsize, 15)
60     grid on
61     box on
62     xlim([-1 1]);
63     ylim([-1 1]);
64     zlim([0 1]);
65     view(3)
66     hold off
67
68     subplot(2,2,2)
69     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')
70     hold on
71     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
72     quiver3(0,0,0,1,0,0,'linewidth',2,'LineStyle','—','Color','k')
73
74     % format legend, labels, limits, grid and box
75     xlabel( $M_x$ , interpreter, latex, fontsize, 15)
76     ylabel( $M_y$ , interpreter, latex, fontsize, 15)
77     xlabel( $M_z$ , interpreter, latex, fontsize, 15)
78     grid on
79     box on
80     xlim([-1 1]);
81     ylim([-1 1]);
82     zlim([0 1]);
83     view(2)
84     hold off
85
86     subplot(2,2,3:4)
87     plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
88     hold on
89     plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-','Color','b');
90     plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','—','Color','k');
91
92     % format title, legend, labels, limits, grid and box
93     title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
94     xlabel(Time (s), interpreter, latex, fontsize, 15)
95     ylabel(Magnetisation, interpreter, latex, fontsize, 15)
96     legend( $M_x$ ,  $M_y$ ,  $M_z$ , interpreter, latex, fontsize, 10)
97     grid on
98     box on
99     xlim([0 max(time)]);
100    ylim([-1 1]);
101    hold off
102
103
104    % assign frame and write it to the video
105    frame = getframe(h);
106    video.writeVideo(frame);
107 end
108
109 video.close();
110
111 %% Animation module — just the 3D
112 % Author: Imraj Singh 03/11/2020
113
114 % initialise the video
115 video = VideoWriter(['3_2-3D', '.mp4'], 'MPEG-4');
116
117 % set the frame rate
118 frameRate = resolution;

```

```

119 video.set('FrameRate', frameRate);
120
121 video.open();
122
123 % initialise the figure
124 h = figure;
125
126 % specify animation captures each degree of rotation
127 for i=1:length(time)
128     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
129     hold on
130     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
131     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
        'Color','r')
132
133     % format title, legend, labels, limits, grid and box
134     title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
135     xlabel(M_x, interpreter, latex, fontsize, 15)
136     ylabel(M_y, interpreter, latex, fontsize, 15)
137     zlabel(M_z, interpreter, latex, fontsize, 15)
138     grid on
139     box on
140     xlim([-1 1]);
141     ylim([-1 1]);
142     zlim([0 1]);
143     view(3)
144     hold off
145     % assign frame and write it to the video
146     frame = getframe(h);
147     video.writeVideo(frame);
148 end
149
150 video.close();

```

### C.3 Problem 3.3:

```

1 % housekeeping
2 clc
3 clear
4 %% Free precession with relaxation rotating FoR
5 % Author: Imraj Singh 03/11/2020
6
7 % Given parameters
8 % Magnetisation aligned along y-axis initially
9 M = [0, 1, 0]';
10 M0 = [0, 0, 1]';
11
12 % Prescribe T1
13 T2 = 1;
14 T1 = T2 * 2;
15
16 % Calculate necessary parameters
17
18 % Precession frequency
19 omega0 = 2 * 2 * pi;
20
21 % End time
22 t = 6;
23 resolution = 50;
24

```

```

25 % Duration and timestep of model
26 time = linspace(0, t, t * resolution + 1);
27
28
29 Msoln(:,1) = exp(-time./T2).*M(1);
30 Msoln(:,2) = exp(-time./T2).*M(2);
31 Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
32
33
34 %% Animation
35
36 % initialise the video
37 video = VideoWriter(['3_3', '.mp4'], 'MPEG-4');
38
39 % set the frame rate
40 frameRate = resolution/4;
41 video.set('FrameRate', frameRate);
42
43 video.open();
44
45 % initialise the figure
46 h = figure;
47
48 % specify animation captures each degree of rotation
49 for i=1:length(time)
50     subplot(2,2,1)
51     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
52     hold on
53     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
54     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
55             'Color','r')
56
57     % format legend, labels, limits, grid and box
58     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
59     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
60     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
61     grid on
62     box on
63     xlim([-1 1]);
64     ylim([-1 1]);
65     zlim([0 1]);
66     view(3)
67     hold off
68
69     subplot(2,2,2)
70     quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—','Color','r')
71     hold on
72     plot(Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
73     quiver(0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
74
75     % format legend, labels, limits, grid and box
76     xlabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
77     ylabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
78     grid on
79     box on
80     xlim([-1 1]);
81     ylim([0 1]);
82     hold off
83
84     subplot(2,2,3:4)
85     plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','—','Color','r');

```

```

85     hold on
86     plot(time(1:i), Msoln(1:i,2), 'linewidth',2, 'LineStyle','-.', 'Color','b');
87     plot(time(1:i), Msoln(1:i,3), 'linewidth',2, 'LineStyle','—', 'Color','k');
88
89     % format title, legend, labels, limits, grid and box
90     title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
91     xlabel(Time(s), interpreter, latex, fontsize, 15)
92     ylabel(Magnetisation, interpreter, latex, fontsize, 15)
93     legend(Mx', My', Mz', interpreter, latex, fontsize, 10)
94     grid on
95     box on
96     xlim([0 max(time)]);
97     ylim([0 1]);
98     hold off
99
100
101     % assign frame and write it to the video
102     frame = getframe(h);
103     video.writeVideo(frame);
104 end
105
106 video.close();

```

#### C.4 Problem 3.4:

```

1 % housekeeping
2 clc
3 clear
4 %% Free precession with relaxation rotating FoR at Larmor
5 % Faster than Larmor by omega0/10
6 % Author: Imraj Singh 03/11/2020
7
8 % Given parameters
9 % Magnitisation aligned along y-axis initially
10 M = [0, 1, 0]';
11 M0 = [0, 0, 1]';
12
13 % Prescribe T1
14 T2 = 1;
15 T1 = T2 * 2;
16
17 % Calculate neccessary parameters
18
19 % Precession frequency
20 omega0 = 2 * 2 * pi;
21
22 % End time
23 t = 6;
24 resolution = 50;
25
26 % Duration and timestep of model
27 time = linspace(0, t, t * resolution + 1);
28
29
30
31 Msoln(:,1) = exp(-time./T2).*(M(1)*cos(omega0/10.*time) + M(2)*sin(omega0/10.*
    time));
32 Msoln(:,2) = exp(-time./T2).*(M(2)*cos(omega0/10.*time) - M(1)*sin(omega0/10.*
    time));
33 Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));

```

```

34
35
36 %% Animation module
37 % Author: Imraj Singh 03/11/2020
38
39 % initialise the video
40 video = VideoWriter(['3_4', '.mp4'], 'MPEG-4');
41
42 % set the frame rate
43 frameRate = resolution/4;
44 video.set('FrameRate', frameRate);
45
46 video.open();
47
48 % initialise the figure
49 h = figure;
50
51 % specify animation captures each degree of rotation
52 for i=1:length(time)
53     subplot(2,2,1)
54     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
55     hold on
56     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
57     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
58         , 'Color','r')
59
60     % format legend, labels, limits, grid and box
61     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
62     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
63     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
64     grid on
65     box on
66     xlim([-1 1]);
67     ylim([-1 1]);
68     zlim([0 1]);
69     view(3)
70     hold off
71
72     subplot(2,2,2)
73     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
74         , 'Color','r')
75     hold on
76     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
77     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
78
79     % format legend, labels, limits, grid and box
80     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
81     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
82     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
83     grid on
84     box on
85     xlim([-1 1]);
86     ylim([-1 1]);
87     zlim([0 1]);
88     view(2)
89     hold off
90
91     subplot(2,2,3:4)
92     plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','—','Color','r');
93     hold on
94     plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','—','Color','b');

```

```

93     plot(time(1:i), Msoln(1:i,3), 'linewidth',2, 'LineStyle', '—', 'Color', 'k');
94
95     % format title, legend, labels, limits, grid and box
96     title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
97     xlabel(Time(s), interpreter, latex, fontsize, 15)
98     ylabel(Magnetisation, interpreter, latex, fontsize, 15)
99     legend(Mx', My', Mz', interpreter, latex, fontsize, 10)
100    grid on
101    box on
102    xlim([0 max(time)]);
103    ylim([-1 1]);
104    hold off
105
106
107    % assign frame and write it to the video
108    frame = getframe(h);
109    video.writeVideo(frame);
110 end
111
112 video.close();

```

### C.5 Problem 3.4:

```

1 % housekeeping
2 clc
3 clear
4 %% Free precession with relaxation rotating FoR at Larmor
5 % Faster than Larmor by omega0/10
6 % Author: Imraj Singh 03/11/2020
7
8 % Given parameters
9 % Magnitisation aligned along y-axis initially
10 M = [0, 1, 0]';
11 M0 = [0, 0, 1]';
12
13 % Prescribe T1
14 T2 = 1;
15 T1 = T2 * 2;
16
17 % Calculate neccessary parameters
18
19 % Precession frequency
20 omega0 = 2 * 2 * pi;
21
22 % End time
23 t = 6;
24 resolution = 50;
25
26 % Duration and timestep of model
27 time = linspace(0, t, t * resolution + 1);
28
29
30
31 Msoln(:,1) = exp(-time./T2).*(M(1)*cos(omega0/10.*time) + M(2)*sin(omega0/10.*
    time));
32 Msoln(:,2) = exp(-time./T2).*(M(2)*cos(omega0/10.*time) - M(1)*sin(omega0/10.*
    time));
33 Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
34
35

```



```

36 %% Animation module
37 % Author: Imraj Singh 03/11/2020
38
39 % initialise the video
40 video = VideoWriter(['3_4', '.mp4'], 'MPEG-4');
41
42 % set the frame rate
43 frameRate = resolution/4;
44 video.set('FrameRate', frameRate);
45
46 video.open();
47
48 % initialise the figure
49 h = figure;
50
51 % specify animation captures each degree of rotation
52 for i=1:length(time)
53     subplot(2,2,1)
54     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
55     hold on
56     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
57     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
58         , 'Color','r')
59
60     % format legend, labels, limits, grid and box
61     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
62     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
63     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
64     grid on
65     box on
66     xlim([-1 1]);
67     ylim([-1 1]);
68     zlim([0 1]);
69     view(3)
70     hold off
71
72     subplot(2,2,2)
73     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
74         , 'Color','r')
75     hold on
76     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
77     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
78
79     % format legend, labels, limits, grid and box
80     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
81     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
82     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
83     grid on
84     box on
85     xlim([-1 1]);
86     ylim([-1 1]);
87     zlim([0 1]);
88     view(2)
89     hold off
90
91     subplot(2,2,3:4)
92     plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','—','Color','r');
93     hold on
94     plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','—','Color','b');
95     plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','—','Color','k');

```

```

95 % format title , legend , labels , limits , grid and box
96 title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
97 xlabel(Time(s), interpreter, latex, fontsize, 15)
98 ylabel(Magnetisation, interpreter, latex, fontsize, 15)
99 legend( $M_{x'}$ ,  $M_{y'}$ ,  $M_{z'}$ , interpreter, latex, fontsize, 10)
100 grid on
101 box on
102 xlim([0 max(time)]);
103 ylim([-1 1]);
104 hold off
105
106
107 % assign frame and write it to the video
108 frame = getframe(h);
109 video.writeVideo(frame);
110 end
111
112 video.close();

```

### C.6 Problem 3.5:

```

1 % housekeeping
2 clc
3 clear
4 %% Free precession with relaxation rotating FoR at Larmor
5 % Faster than Larmor by omega0/10
6 % Author: Imraj Singh 03/11/2020
7
8 % Given parameters
9 % Magnitisation aligned along y-axis initially
10 M = [0, 1, 0]';
11 M0 = [0, 0, 1]';
12
13 % Prescribe T1
14 T2 = 1;
15 T1 = T2 * 2;
16
17 % Calculate neccessary parameters
18
19 % Precession frequency
20 omega0 = 2 * 2 * pi;
21
22 % End time
23 t = 6;
24 resolution = 50;
25
26 % Duration and timestep of model
27 time = linspace(0, t, t * resolution + 1);
28
29
30
31 Msoln(:,1) = exp(-time./T2).*(M(1)*cos(-omega0/10.*time) + M(2)*sin(-omega0/10.*
    time));
32 Msoln(:,2) = exp(-time./T2).*(M(2)*cos(-omega0/10.*time) - M(1)*sin(-omega0/10.*
    time));
33 Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
34
35
36 %% Animation module
37 % Author: Imraj Singh 03/11/2020

```

```

38
39 % initialise the video
40 video = VideoWriter(['3_4', '.mp4'], 'MPEG-4');
41
42 % set the frame rate
43 frameRate = resolution/4;
44 video.set('FrameRate', frameRate);
45
46 video.open();
47
48 % initialise the figure
49 h = figure;
50
51 % specify animation captures each degree of rotation
52 for i=1:length(time)
53     subplot(2,2,1)
54     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
55     hold on
56     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
57     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
58         , 'Color','r')
59
60 % format legend, labels, limits, grid and box
61 xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
62 ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
63 zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
64 grid on
65 box on
66 xlim([-1 1]);
67 ylim([-1 1]);
68 zlim([0 1]);
69 view(3)
70 hold off
71
72 subplot(2,2,2)
73 quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
74     , 'Color','r')
75 hold on
76 plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
77 quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
78
79 % format legend, labels, limits, grid and box
80 xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
81 ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
82 zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
83 grid on
84 box on
85 xlim([-1 1]);
86 ylim([-1 1]);
87 zlim([0 1]);
88 view(2)
89 hold off
90
91 subplot(2,2,3:4)
92 plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','—','Color','r');
93 hold on
94 plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','—','Color','b');
95 plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','—','Color','k');
96
97 % format title, legend, labels, limits, grid and box
98 title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)

```

```

97 xlabel(Time(s), interpreter, latex, fontsize, 15)
98 ylabel(Magnetisation, interpreter, latex, fontsize, 15)
99 legend( $M_{x'}$ ,  $M_{y'}$ ,  $M_{z'}$ , interpreter, latex, fontsize, 10)
100 grid on
101 box on
102 xlim([0 max(time)]);
103 ylim([-1 1]);
104 hold off
105
106
107 % assign frame and write it to the video
108 frame = getframe(h);
109 video.writeVideo(frame);
110 end
111
112 video.close();

```

### C.7 Comparison between 3.4 and 3.5:

```

1 % housekeeping
2 clc
3 clear
4 %% Free precession with relaxation rotating FoR at Larmor
5 % Faster than Larmor by omega0/10
6 % Author: Imraj Singh 03/11/2020
7
8 % Given parameters
9 % Magnetisation aligned along y-axis initially
10 M = [0, 1, 0]';
11 M0 = [0, 0, 1]';
12
13 % Prescribe T1
14 T2 = 1;
15 T1 = T2 * 2;
16
17 % Calculate necessary parameters
18
19 % Precession frequency
20 omega0 = 2 * 2 * pi;
21
22 % End time
23 t = 6;
24 resolution = 50;
25
26 % Duration and timestep of model
27 time = linspace(0, t, t * resolution + 1);
28
29 Msolnf(:,1) = exp(-time./T2).*(M(1)*cos(omega0*11/10.*time) + M(2)*sin(omega0
    *11/10.*time));
30 Msolnf(:,2) = exp(-time./T2).*(M(2)*cos(omega0*11/10.*time) - M(1)*sin(omega0
    *11/10.*time));
31 Msolnf(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
32
33 Msolns(:,1) = exp(-time./T2).*(M(1)*cos(omega0*9/10.*time) + M(2)*sin(omega0
    *9/10.*time));
34 Msolns(:,2) = exp(-time./T2).*(M(2)*cos(omega0*9/10.*time) - M(1)*sin(omega0
    *9/10.*time));
35 Msolns(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
36
37

```

```

38
39
40 %% Animation module – just the 3D
41 % Author: Imraj Singh 03/11/2020
42
43 % initialise the video
44 video = VideoWriter(['3_45_comp', '.mp4'], 'MPEG-4');
45
46 % set the frame rate
47 frameRate = resolution/4;
48 video.set('FrameRate', frameRate);
49
50 video.open();
51
52 % initialise the figure
53 h = figure;
54
55 % specify animation captures each degree of rotation
56 for i=1:length(time)
57     quiver3(0,0,0,Msolns(i,1),Msolns(i,2),Msolns(i,3),'linewidth',4,'LineStyle','-', 'Color','r')
58     hold on
59     quiver3(0,0,0,Msolnf(i,1),Msolnf(i,2),Msolnf(i,3),'linewidth',4,'LineStyle','-', 'Color','b')
60     quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','—','Color','k')
61     plot3(Msolns(1:i,1),Msolns(1:i,2),Msolns(1:i,3),'r—','linewidth',2)
62     plot3(Msolnf(1:i,1),Msolnf(1:i,2),Msolnf(1:i,3),'b—','linewidth',2)
63
64     % format title, legend, labels, limits, grid and box
65     title(['Time: ', num2str(time(i)), ' s'], interpreter, latex, fontsize, 15)
66     legend('Slower','Faster', interpreter, latex, fontsize, 10)
67     xlabel( $M_x$ , interpreter, latex, fontsize, 15)
68     ylabel( $M_y$ , interpreter, latex, fontsize, 15)
69     zlabel( $M_z$ , interpreter, latex, fontsize, 15)
70     grid on
71     box on
72     xlim([-1 1]);
73     ylim([-1 1]);
74     zlim([0 1]);
75     view(3)
76     hold off
77     % assign frame and write it to the video
78     frame = getframe(h);
79     video.writeVideo(frame);
80 end
81
82 video.close();

```

## D Problem 4 code:

### D.1 Problem 4.1:

```

1 clc
2 clear
3
4 %% Problem 4.1 FID sequence magnetisation vector visualisation
5 % Author: Imraj Singh 11/11/2020
6
7 % Given parameters
8

```

```

9 % Magnitisation aligned along z-axis initially
10 M = [0, 0, 1]';
11
12 % Number of sample in each part of sequence
13 N = 101;
14
15 % Prescribe T1
16 T2 = 1;
17 T1 = T2 * 2;
18
19 % Calculate neccessary parameters
20
21 % Precession frequency
22 omega0 = 2 * 2 * pi;
23
24 % Flip time
25 INIT = 0;
26 tINIT = linspace(0, 0, N);
27
28 % Flip time
29 FT = .001;
30 tFT = linspace(0, FT, N);
31
32 % Relaxation time
33 RT = 8;
34 tRT = linspace(FT, RT + FT, N);
35
36 time = [tINIT, tFT, tRT];
37
38 % Indices
39 % Initial time
40 INITi = N;
41
42 % Flip time
43 FTi = INITi + N;
44
45 % Relaxation time
46 RTi = FTi + N;
47
48 Msoln = zeros(length(time),3);
49
50 % Initial time
51 Msoln(1:INITi,1) = M(1);
52 Msoln(1:INITi,2) = M(2);
53 Msoln(1:INITi,3) = M(3);
54
55 % Flip time
56 Vis90 = pi/2/FT;
57 M = Msoln(INITI,:);
58 Msoln(INITI:FTi,1) = M(1);
59 Msoln(INITI:FTi,2) = M(2)*cos(Vis90.*time(INITI:FTi)) + M(3)*sin(Vis90.*time(
    INITi:FTi));
60 Msoln(INITI:FTi,3) = M(3)*cos(Vis90.*time(INITI:FTi)) - M(2)*sin(Vis90.*time(
    INITi:FTi));
61
62 % Flip time 2
63 M = Msoln(FTi,:);
64 Msoln(FTi:RTi,1) = M(1).*exp(-time(FTi:RTi)./T2);
65 Msoln(FTi:RTi,2) = M(2).*exp(-time(FTi:RTi)./T2);
66 Msoln(FTi:RTi,3) = M(3)*exp(-time(FTi:RTi)./T1) + (1-exp(-time(FTi:RTi)./T1));
67

```

```

68
69 %% Animation module
70 % Author: Imraj Singh 03/11/2020
71
72 % initialise the video
73 video = VideoWriter(['4_1_rot', '.mp4'], 'MPEG-4');
74
75 % set the frame rate
76 frameRate = (N-1)/10;
77 video.set('FrameRate', frameRate);
78
79 video.open();
80
81 % initialise the figure
82 h = figure;
83
84 % specify animation captures each degree of rotation
85 for i=1:length(time)
86     subplot(2,2,1)
87     quiver3(0,0,0,0,0,1,'linewidth',2,'LineStyle','—','Color','k')
88     hold on
89     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
90     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
91             'Color','r')
92
93     % format legend, labels, limits, grid and box
94     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
95     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
96     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
97     grid on
98     box on
99     xlim([-1 1]);
100    ylim([-1 1]);
101    zlim([0 1]);
102    view(3)
103    hold off
104
105    subplot(2,2,2)
106    quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—','Color','r')
107    hold on
108    plot(Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
109    quiver(0,0,0,1,'linewidth',2,'LineStyle','—','Color','k')
110
111    % format legend, labels, limits, grid and box
112    xlabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
113    ylabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
114    grid on
115    box on
116    xlim([-1 1]);
117    ylim([0 1]);
118    hold off
119
120    subplot(2,2,3:4)
121    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','—','Color','r');
122    hold on
123    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','—','Color','b');
124    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','—','Color','k');
125
126    % format title, legend, labels, limits, grid and box
127    if i > 0 && i <= INITi

```



```

128         title('Magnetisation vector along $B_0$ initially', interpreter, latex,
129               fontsize, 15)
130     xlim([0 1]);
131     xlabel(Time(s), interpreter, latex, fontsize, 15)
132     elseif i > INITi && i <= FTi
133         title(['RF pulse $\pi / 2$, time: ', num2str(round(1000*time(i), 3)), ' ms
134               '], interpreter, latex, fontsize, 15)
135     xlim([0 FT]);
136     xlabel(Time(s), interpreter, latex, fontsize, 15)
137     elseif i > FTi && i <= RTi
138         title(['Time from $\pi / 2$ flip: ', num2str(round(time(i) - INIT - FT,
139               3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
140     xlim([0 RT]);
141     xlabel(Time(s), interpreter, latex, fontsize, 15)
142     end
143     ylabel(Magnetisation, interpreter, latex, fontsize, 15)
144     legend($M_x$, $M_y$, $M_z$, interpreter, latex, fontsize, 10)
145     grid on
146     box on
147     ylim([0 1]);
148     hold off
149
150     % assign frame and write it to the video
151     frame = getframe(h);
152     video.writeVideo(frame);
153 end
154
155 video.close();

```

## D.2 Problem 4.2:

```

1  clc
2  clear
3
4  %% Problem 4.1 IR sequence magnetisation vector visualisation
5  % Author: Imraj Singh 11/11/2020
6
7  % Given parameters
8
9  % Magnitisation aligned along z-axis initially
10 M = [0, 0, 1]';
11
12 % Number of sample in each part of sequence
13 N = 101;
14
15 % Prescribe T1
16 T2 = 1;
17 T1 = T2 * 2;
18
19 % Calculate neccessary parameters
20
21 % Precession frequency
22 omega0 = 2 * 2 * pi;
23
24 % Flip time
25 INIT = 0;
26 tINIT = linspace(0, 0, N);
27
28 % Flip time 1

```

```

29 FT_1 = .002;
30 tFT_1 = linspace(0, FT_1, N);
31
32 % Relaxation time 1
33 RT_1 = .5;
34 tRT_1 = linspace(FT_1, RT_1 + FT_1, N);
35
36 % Flip time 2
37 FT_2 = .001;
38 tFT_2 = linspace(RT_1 + FT_1, RT_1 + FT_1 + FT_2, N);
39
40 % Relaxation time 2
41 RT_2 = 8;
42 tRT_2 = linspace(RT_1 + FT_1 + FT_2, + RT_1 + FT_1 + FT_2 + RT_2, N);
43
44 time = [tINIT, tFT_1, tRT_1, tFT_2, tRT_2];
45
46 % Indices
47 % Initial time
48 INITi_1 = N;
49
50 % Flip time 1
51 FTi_1 = INITi_1 + N;
52
53 % Relaxation time 1
54 RTi_1 = FTi_1 + N;
55
56 % Flip time 2
57 FTi_2 = RTi_1 + N;
58
59 % Relaxation time 2
60 RTi_2 = FTi_2 + N;
61
62 Msoln = zeros(length(time),3);
63
64 % Initial time
65 Msoln(1:INITi_1,1) = M(1);
66 Msoln(1:INITi_1,2) = M(2);
67 Msoln(1:INITi_1,3) = M(3);
68
69 % Flip time 1
70 tad = time;
71 Vis90 = pi/FT_1;
72 M = Msoln(INITi_1,:);
73 Msoln(INITi_1:FTi_1,1) = M(1);
74 Msoln(INITi_1:FTi_1,2) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*sin(Vis90.*
    tad(INITi_1:FTi_1));
75 Msoln(INITi_1:FTi_1,3) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*sin(Vis90.*
    tad(INITi_1:FTi_1));
76
77
78 % Relaxation time 1
79 tad = tad - FT_1;
80 M = Msoln(FTi_1,:);
81 Msoln(FTi_1:RTi_1,1) = M(1).*exp(-tad(FTi_1:RTi_1)./T2);
82 Msoln(FTi_1:RTi_1,2) = M(2).*exp(-tad(FTi_1:RTi_1)./T2);
83 Msoln(FTi_1:RTi_1,3) = M(3)*exp(-tad(FTi_1:RTi_1)./T1) + (1-exp(-tad(FTi_1:RTi_1)
    )./T1));
84
85 % Flip time 2
86 tad = tad - RT_1;

```

```

87 Vis90 = pi/2/FT_2;
88 M = Msoln(RTi_1,:);
89 Msoln(RTi_1:FTi_2,1) = M(1);
90 Msoln(RTi_1:FTi_2,2) = M(2)*cos(Vis90.*tad(RTi_1:FTi_2)) + M(3)*sin(Vis90.*tad(
    RTi_1:FTi_2));
91 Msoln(RTi_1:FTi_2,3) = M(3)*cos(Vis90.*tad(RTi_1:FTi_2)) - M(2)*sin(Vis90.*tad(
    RTi_1:FTi_2));
92
93 % Relaxation time 2
94 tad = tad - FT_2;
95 M = Msoln(FTi_2,:);
96 Msoln(FTi_2:RTi_2,1) = M(1).*exp(-tad(FTi_2:RTi_2)./T2);
97 Msoln(FTi_2:RTi_2,2) = M(2).*exp(-tad(FTi_2:RTi_2)./T2);
98 Msoln(FTi_2:RTi_2,3) = M(3).*exp(-tad(FTi_2:RTi_2)./T1) + (1-exp(-tad(FTi_2:RTi_2
    )./T1));
99
100
101 %% Animation module
102 % Author: Imraj Singh 03/11/2020
103
104 % initialise the video
105 video = VideoWriter(['4_2_rot_opt', '.mp4'], 'MPEG-4');
106
107 % set the frame rate
108 frameRate = (N-1)/10;
109 video.set('FrameRate', frameRate);
110
111 video.open();
112
113 % initialise the figure
114 h = figure;
115
116 % specify animation captures each degree of rotation
117 for i=1:length(time)
118     subplot(2,2,1)
119     quiver3(0,0,0,0,0,1,'linewidth',2,'LineStyle','—','Color','k')
120     hold on
121     plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
122     quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—',
        'Color','r')
123
124     % format legend, labels, limits, grid and box
125     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
126     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
127     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
128     grid on
129     box on
130     xlim([-1 1]);
131     ylim([-1 1]);
132     zlim([-1 1]);
133     view(3)
134     hold off
135
136     subplot(2,2,2)
137     quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','—','Color','r')
138     hold on
139     plot(Msoln(1:i,2),Msoln(1:i,3),'k—','linewidth',2)
140     quiver(0,0,0,1,'linewidth',2,'LineStyle','—','Color','k')
141
142     % format legend, labels, limits, grid and box
143     xlabel( $M_{y'}$ , interpreter, latex, fontsize, 15)

```

```

144     ylabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
145     grid on
146     box on
147     xlim([-1 1]);
148     ylim([-1 1]);
149     hold off
150
151     subplot(2,2,3:4)
152     plot(time(1:i), Msoln(1:i,1), 'linewidth',2, 'LineStyle','-', 'Color','r');
153     hold on
154     plot(time(1:i), Msoln(1:i,2), 'linewidth',2, 'LineStyle','-.', 'Color','b');
155     plot(time(1:i), Msoln(1:i,3), 'linewidth',2, 'LineStyle','—', 'Color','k');
156
157     % format title, legend, labels, limits, grid and box
158
159     if i > 0 && i <= INITi_1
160         title('Magnetisation vector along  $B_0$  initially', interpreter, latex,
161             fontsize, 15)
162         xlim([0 1]);
163         xlabel(Time(s), interpreter, latex, fontsize, 15)
164     elseif i > INITi_1 && i <= FTi_1
165         xlim([0 FT_1]);
166         title(['RF pulse  $\pi$ , time: ', num2str(round(1000*time(i), 3)), ' ms'],
167             interpreter, latex, fontsize, 15)
168         xlabel(Time(s), interpreter, latex, fontsize, 15)
169     elseif i > FTi_1 && i <= RTi_1
170         title(['Time from  $\pi$  flip: ', num2str(round(time(i) - INIT - FT_1, 3)
171             ), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
172         xlim([0 FT_1+RT_1+FT_2+RT_2]);
173         xlabel(Time(s), interpreter, latex, fontsize, 15)
174     elseif i > RTi_1 && i <= FTi_2
175         title(['RF pulse  $\pi / 2$ , time: ', num2str(round(1000*(time(i) - FT_1 -
176             RT_1), 3)), ' ms'], interpreter, latex, fontsize, 15)
177         xlim([0 FT_1+RT_1+FT_2+RT_2]);
178         xlabel(Time(s), interpreter, latex, fontsize, 15)
179     elseif i > FTi_2 && i <= RTi_2
180         title(['Time from  $\pi / 2$  flip: ', num2str(round(time(i) - INIT - FT_1
181             - RT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
182         xlim([0 FT_1+RT_1+FT_2+RT_2]);
183         xlabel(Time(s), interpreter, latex, fontsize, 15)
184     end
185
186     ylabel(Magnetisation, interpreter, latex, fontsize, 15)
187     legend( $M_{x'}$ ,  $M_{y'}$ ,  $M_{z'}$ , interpreter, latex, fontsize, 10)
188     grid on
189     box on
190     ylim([-1 1]);
191     hold off
192
193     % assign frame and write it to the video
194     frame = getframe(h);
195     video.writeVideo(frame);
196 end
197
198 video.close();

```

## E Problem 5 code:

### E.1 Problem 5.1:

```

1  clc
2  clear
3
4  %% Free precession with Dephasing
5  % Author: Imraj Singh 03/11/2020
6
7  % Given parameters
8
9  % Number of sample in each part of sequence
10 N = 101;
11
12 % Number of isochromats
13 Num = 1001;
14
15 % Prescribe T1
16 T2 = 1;
17 T1 = T2 * 2;
18
19 % Calculate necessary parameters
20
21 % Precession frequency
22 omega0 = 50 * pi;
23 deltaomega = 16;
24 omegaovector = linspace(-deltaomega, deltaomega, Num);
25
26 % Inital time 1
27 INIT_1 = 0;
28 tINIT_1 = linspace(0, INIT_1, N);
29
30 % Flip time 1
31 FT_1 = .001;
32 tFT_1 = linspace(0, FT_1, N);
33
34 % Dephasing time 1
35 DP_1 = .5;
36 tDP_1 = linspace(FT_1, DP_1 + FT_1, N);
37
38 % Flip time 2
39 FT_2 = .001;
40 tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);
41
42 % Rephasing time
43 RP = DP_1;
44 tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);
45
46 % Inital time 2
47 INIT_2 = 0;
48 tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP, N);
49
50 % Dephasing time 2
51 DP_2 = DP_1;
52 tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP + DP_2, N);
53
54 time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];
55
56 % Indices
57 % Initial time
58 INITi_1 = N;

```

```

59
60 % Flip time 1
61 FTi_1 = INITi_1 + N;
62
63 % Dephasing time 1
64 DPi_1 = FTi_1 + N;
65
66 % Flip time 2
67 FTi_2 = DPi_1 + N;
68
69 % Rephasing time
70 RPi = FTi_2 + N;
71
72 % Initial time
73 INITi_2 = RPi + N;
74
75 % Dephasing time 2
76 DPi_2 = INITi_2 + N;
77
78
79 Msoln = zeros(length(time),3, length(omegaovector));
80
81 for i = 1:length(omegaovector)
82     % Initial time
83     Msoln(1:INITi_1,1,i) = 0;
84     Msoln(1:INITi_1,2,i) = 0;
85     Msoln(1:INITi_1,3,i) = 1;
86
87     % Flip time 1
88     tad = time;
89     Vis90 = pi/2/FT_1;
90     M = Msoln(INITi_1,:,i);
91     Msoln(INITi_1:FTi_1,1,i) = M(1);
92     Msoln(INITi_1:FTi_1,2,i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*sin(
        Vis90.*tad(INITi_1:FTi_1));
93     Msoln(INITi_1:FTi_1,3,i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*sin(
        Vis90.*tad(INITi_1:FTi_1));
94
95     % Dephase time 1
96     tad = tad - FT_1;
97     M = Msoln(FTi_1,:,i);
98     Msoln(FTi_1:DPi_1,1,i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) + M(2)
        *sin(omegaovector(i).*tad(FTi_1:DPi_1)));
99     Msoln(FTi_1:DPi_1,2,i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) - M(1)
        *sin(omegaovector(i).*tad(FTi_1:DPi_1)));
100    Msoln(FTi_1:DPi_1,3,i) = M(3);
101
102    % Flip time 2
103    tad = tad - DP_1;
104    Vis90 = pi/FT_2;
105    M = Msoln(DPi_1,:,i);
106    Msoln(DPi_1:FTi_2,1,i) = M(1)*cos(Vis90.*tad(DPi_1:FTi_2)) + M(3)*sin(Vis90
        .*tad(DPi_1:FTi_2));
107    Msoln(DPi_1:FTi_2,2,i) = M(2);
108    Msoln(DPi_1:FTi_2,3,i) = M(3)*cos(Vis90.*tad(DPi_1:FTi_2)) - M(2)*sin(Vis90
        .*tad(DPi_1:FTi_2));
109
110    % Rephase time
111    tad = tad - FT_2;
112    M = Msoln(FTi_2,:,i);

```

```

113     Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
114     Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
115     Msoln(FTi_2:RPi,3,i) = M(3);
116
117     % Echo time
118     tad = tad - RP;
119     M = Msoln(RPi,:,i);
120     Msoln(RPi:INITi_2,1,i) = M(1);
121     Msoln(RPi:INITi_2,2,i) = M(2);
122     Msoln(RPi:INITi_2,3,i) = M(3);
123
124
125     % Dephase time 2
126     tad = tad - INIT_2;
127     M = Msoln(INITi_2,:,i);
128     Msoln(INITi_2:DPi_2,1,i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPi_2)) + M
        (2)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
129     Msoln(INITi_2:DPi_2,2,i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPi_2)) - M
        (1)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
130     Msoln(INITi_2:DPi_2,3,i) = M(3);
131 end
132 Mperp = zeros(length(time),1);
133 for i=1:length(time)
134     Mperp(i) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/
        Num;
135 end
136
137 %% Animation module
138
139 video = VideoWriter(['5_1', '.mp4'], 'MPEG-4');
140
141 % set the frame rate
142 frameRate = N/8;
143 video.set('FrameRate', frameRate);
144
145 video.open();
146
147 h = figure;
148
149
150 for i=INITi_1:length(time)
151     subplot(2,2,1)
152
153     quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
        LineStyle','-','Color','r')
154     hold on
155     for j=1:100:length(omegaovector)
156         quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
            LineStyle','-','Color','r')
157     end
158     grid on
159     box on
160     hold off
161     xlim([-1 1]);
162     ylim([-1 1]);
163     zlim([-1 1]);
164     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
165     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
166     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)

```



```

167
168 subplot(2,2,2)
169 quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
    LineStyle','-','Color','r')
170 hold on
171 for j=1:100:length(omegaovector)
172     quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
        LineStyle','-','Color','r')
173 end
174 grid on
175 box on
176 xlim([-1 1]);
177 ylim([-1 1]);
178 zlim([-1 1]);
179 hold off
180 view(2)
181 xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
182 ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
183 zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
184
185
186 subplot(2,2,3:4)
187
188 plot(time(1:i),Mperp(1:i),'linewidth',2,'LineStyle','-','Color','k')
189 hold on
190 plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','—','Color','k')
191 hold off
192 grid on
193 box on
194 ylim([0 1]);
195 xlim([0 max(time)]);
196 xlabel(Time (s), interpreter, latex, fontsize, 15)
197 ylabel(Normalised signal, interpreter, latex, fontsize, 15)
198 legend( $|\frac{1}{N} \sum M_{\perp}|$ ,  $T_2$  envelope, interpreter, latex, fontsize, 10)
199
200 % Initial
201 if i > 0 && i <= INITi_1
202     title('Magnetisation vector along  $B_0$  initially', interpreter, latex,
        fontsize, 15)
203
204 % First flip
205 elseif i > INITi_1 && i <= FTi_1
206     xlim([0 FT_1]);
207     title('RF pulse  $\pi/2$ ', interpreter, latex, fontsize, 15)
208
209 % First Dephase
210 elseif i > FTi_1 && i <= DPi_1
211     title(['Isocromats dephasing,  $t =$ ', num2str(round(time(i) - INIT_1 -
        FT_1, 3)), ' s'], interpreter, latex, fontsize, 15)
212
213 % Second flip
214 elseif i > DPi_1 && i <= FTi_2
215     title(['RF pulse  $\pi$ ,  $t = \tau =$ ', num2str(round(DP_1, 3)), ' s'],
        interpreter, latex, fontsize, 15)
216
217 % Rephasing
218 elseif i > FTi_2 && i <= RPi
219     title(['Isocromats rephasing,  $t =$ ', num2str(round(time(i) - INIT_1 -
        FT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
220
221 % Echo

```

```

222     elseif i > RPi && i <= INITi_2
223         title(['Echo time, $t = T_E = 2 \tau = $ ', num2str(round(DP_1 * 2, 3)), '
                s'], interpreter, latex, fontsize, 15)
224
225         % Dephasing
226     elseif i > INITi_2 && i <= DPi_2
227         title(['Isocromats dephasing, $t = $ ', num2str(round(time(i) - INIT_1 -
                FT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
228
229     end
230
231     frame = getframe(h);
232     video.writeVideo(frame);
233 end
234
235 video.close();

```

## E.2 Problem 5.2:

```

1  clc
2  clear
3
4  %% Free procession with Dephasing
5  % Author: Imraj Singh 03/11/2020
6
7  % Given parameters
8
9  % Number of sample in each part of sequence
10 N = 11;
11
12 % Number of isochromats
13 Num = 1001;
14
15 % Prescribe T1
16 T2 = 1;
17 T1 = T2 * 2;
18
19 % Calculate necessary parameters
20
21 % Precession frequency
22 omega0 = 50 * pi;
23 deltaomega = 0;
24 delta = 8;
25 omegaovector = deltaomega + delta * tan(pi * (rand(Num, 1) - 1/2));
26
27 % Inital time 1
28 INIT_1 = 0;
29 tINIT_1 = linspace(0, INIT_1, N);
30
31 % Flip time 1
32 FT_1 = .001;
33 tFT_1 = linspace(0, FT_1, N);
34
35 % Dephasing time 1
36 DP_1 = .5;
37 tDP_1 = linspace(FT_1, DP_1 + FT_1, N);
38
39 % Flip time 2
40 FT_2 = .001;
41 tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);

```

```

42
43 % Rephasing time
44 RP = DP_1;
45 tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);
46
47 % Initial time 2
48 INIT_2 = 0;
49 tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP, N);
50
51 % Dephasing time 2
52 DP_2 = DP_1;
53 tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP + DP_2, N);
54
55 time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];
56
57 % Indices
58 % Initial time
59 INITi_1 = N;
60
61 % Flip time 1
62 FTi_1 = INITi_1 + N;
63
64 % Dephasing time 1
65 DPi_1 = FTi_1 + N;
66
67 % Flip time 2
68 FTi_2 = DPi_1 + N;
69
70 % Rephasing time
71 RPi = FTi_2 + N;
72
73 % Initial time
74 INITi_2 = RPi + N;
75
76 % Dephasing time 2
77 DPi_2 = INITi_2 + N;
78
79
80 Msoln = zeros(length(time), 3, length(omegaovector));
81
82 for i = 1:length(omegaovector)
83     % Initial time
84     Msoln(1:INITi_1, 1, i) = 0;
85     Msoln(1:INITi_1, 2, i) = 0;
86     Msoln(1:INITi_1, 3, i) = 1;
87
88     % Flip time 1
89     tad = time;
90     Vis90 = pi/2/FT_1;
91     M = Msoln(INITi_1, :, i);
92     Msoln(INITi_1:FTi_1, 1, i) = M(1);
93     Msoln(INITi_1:FTi_1, 2, i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*sin(
        Vis90.*tad(INITi_1:FTi_1));
94     Msoln(INITi_1:FTi_1, 3, i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*sin(
        Vis90.*tad(INITi_1:FTi_1));
95
96     % Dephase time 1
97     tad = tad - FT_1;
98     M = Msoln(FTi_1, :, i);
99     Msoln(FTi_1:DPi_1, 1, i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) + M(2)
        *sin(omegaovector(i).*tad(FTi_1:DPi_1)));

```

```

100     Msoln(FTi_1:DPI_1,2,i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPI_1)) - M(1)
      *sin(omegaovector(i).*tad(FTi_1:DPI_1)));
101     Msoln(FTi_1:DPI_1,3,i) = M(3);
102
103     % Flip time 2
104     tad = tad - DP_1;
105     Vis90 = pi/FT_2;
106     M = Msoln(DPI_1, :, i);
107     Msoln(DPI_1:FTi_2,1,i) = M(1)*cos(Vis90.*tad(DPI_1:FTi_2)) + M(3)*sin(Vis90
      .*tad(DPI_1:FTi_2));
108     Msoln(DPI_1:FTi_2,2,i) = M(2);
109     Msoln(DPI_1:FTi_2,3,i) = M(3)*cos(Vis90.*tad(DPI_1:FTi_2)) - M(2)*sin(Vis90
      .*tad(DPI_1:FTi_2));
110
111     % Rephase time
112     tad = tad - FT_2;
113     M = Msoln(FTi_2, :, i);
114     Msoln(FTi_2:RPI,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPI)) + M(2)*sin(
      omegaovector(i).*tad(FTi_2:RPI));
115     Msoln(FTi_2:RPI,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPI)) - M(1)*sin(
      omegaovector(i).*tad(FTi_2:RPI));
116     Msoln(FTi_2:RPI,3,i) = M(3);
117
118     % Echo time
119     tad = tad - RP;
120     M = Msoln(RPI, :, i);
121     Msoln(RPI:INITi_2,1,i) = M(1);
122     Msoln(RPI:INITi_2,2,i) = M(2);
123     Msoln(RPI:INITi_2,3,i) = M(3);
124
125
126     % Dephase time 2
127     tad = tad - INIT_2;
128     M = Msoln(INITi_2, :, i);
129     Msoln(INITi_2:DPI_2,1,i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPI_2)) + M
      (2)*sin(omegaovector(i).*tad(INITi_2:DPI_2));
130     Msoln(INITi_2:DPI_2,2,i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPI_2)) - M
      (1)*sin(omegaovector(i).*tad(INITi_2:DPI_2));
131     Msoln(INITi_2:DPI_2,3,i) = M(3);
132 end
133 Mperp = zeros(length(time),1);
134 for i=1:length(time)
135     Mperp(i) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/
      Num;
136 end
137
138 %% Animation module
139
140 video = VideoWriter(['5_2', '.mp4'], 'MPEG-4');
141
142 % set the frame rate
143 frameRate = N/8;
144 video.set('FrameRate', frameRate);
145
146 video.open();
147
148 h = figure;
149
150
151 for i=INITi_1:length(time)
152     subplot(2,2,1)

```

```

153
154     quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
     LineStyle','-','Color','r')
155     hold on
156     for j=1:100:length(omegaovector)
157         quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
     LineStyle','-','Color','r')
158     end
159     grid on
160     box on
161     hold off
162     xlim([-1 1]);
163     ylim([-1 1]);
164     zlim([-1 1]);
165     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
166     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
167     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
168
169     subplot(2,2,2)
170     quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
     LineStyle','-','Color','r')
171     hold on
172     for j=1:100:length(omegaovector)
173         quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
     LineStyle','-','Color','r')
174     end
175     grid on
176     box on
177     xlim([-1 1]);
178     ylim([-1 1]);
179     zlim([-1 1]);
180     hold off
181     view(2)
182     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
183     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
184     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
185
186
187     subplot(2,2,3:4)
188
189     plot(time(1:i),Mperp(1:i),'linewidth',2,'LineStyle','-','Color','k')
190     hold on
191     plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','—','Color','k')
192     hold off
193     grid on
194     box on
195     ylim([0 1]);
196     xlim([0 max(time)]);
197     xlabel(Time (s), interpreter, latex, fontsize, 15)
198     ylabel(Normalised signal, interpreter, latex, fontsize, 15)
199     legend( $|\frac{1}{N} \sum M_{\perp}|$ ,  $T_2$  envelope, interpreter, latex, fontsize, 10)
200
201     % Initial
202     if i > 0 && i <= INITi_1
203         title('Magnetisation vector along  $B_0$  initially', interpreter, latex,
      fontsize, 15)
204
205     % First flip
206     elseif i > INITi_1 && i <= FTi_1
207         xlim([0 FT_1]);
208         title('RF pulse  $\pi/2$ ', interpreter, latex, fontsize, 15)

```

```

209
210     % First Dephase
211     elseif i > FTi_1 && i <= DPi_1
212         title(['Isocromats dephasing, t = ', num2str(round(time(i) - INIT_1 -
213             FT_1, 3)), ' s'], interpreter, latex, fontsize, 15)
214
215     % Second flip
216     elseif i > DPi_1 && i <= FTi_2
217         title(['RF pulse $\pi$, $t = \tau$ = ', num2str(round(DP_1, 3)), ' s'],
218             interpreter, latex, fontsize, 15)
219
220     % Rephasing
221     elseif i > FTi_2 && i <= RPi
222         title(['Isocromats rephasing, $t = $ ', num2str(round(time(i) - INIT_1 -
223             FT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
224
225     % Echo
226     elseif i > RPi && i <= INITi_2
227         title(['Echo time, $t = T_E = 2 \tau = $ ', num2str(round(DP_1 * 2, 3)), '
228             s'], interpreter, latex, fontsize, 15)
229
230     % Dephasing
231     elseif i > INITi_2 && i <= DPi_2
232         title(['Isocromats dephasing, $t = $ ', num2str(round(time(i) - INIT_1 -
233             FT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
234
235     end
236
237     frame = getframe(h);
238     video.writeVideo(frame);
239 end
240
241 video.close();

```

### E.3 Problem 5.3:

```

1 % house keeping
2 clc
3 clear
4 %% Sensitivity of Uniform Distribution to delta omega
5 % Author: Imraj Singh
6
7
8 % Given parameters
9
10 % Number of sample in each part of sequence
11 N = 101;
12
13 % Number of isochromats
14 Num = 10001;
15
16 % Prescribe T1
17 T2 = 1;
18 T1 = T2 * 2;
19
20 % Calculate necessary parameters
21
22 % Initial time t
23 INIT_1 = 0;
24 tINIT_1 = linspace(0, INIT_1, N);

```

```

25
26 % Flip time 1
27 FT_1 = .001;
28 tFT_1 = linspace(0, FT_1, N);
29
30 % Dephasing time 1
31 DP_1 = .5;
32 tDP_1 = linspace(FT_1, DP_1 + FT_1, N);
33
34 % Flip time 2
35 FT_2 = .001;
36 tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);
37
38 % Rephasing time
39 RP = DP_1;
40 tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);
41
42 % Initial time 2
43 INIT_2 = 0;
44 tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP, N);
45
46 % Dephasing time 2
47 DP_2 = DP_1;
48 tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP + DP_2, N);
49
50 time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];
51
52 % Indices
53 % Initial time
54 INITi_1 = N;
55
56 % Flip time 1
57 FTi_1 = INITi_1 + N;
58
59 % Dephasing time 1
60 DPi_1 = FTi_1 + N;
61
62 % Flip time 2
63 FTi_2 = DPi_1 + N;
64
65 % Rephasing time
66 RPi = FTi_2 + N;
67
68 % Initial time
69 INITi_2 = RPi + N;
70
71 % Dephasing time 2
72 DPi_2 = INITi_2 + N;
73
74 %% Sensitivity of Uniform Distribution to delta omega
75 % Author: Imraj Singh
76
77 % Precession frequency
78 omega0 = 50 * pi;
79 delta = [1 2 4 8 16 32 64 128];
80
81 Mperp = zeros(length(time), length(delta));
82 for Di = 1:length(delta)
83     omegaovector = linspace(-delta(Di), delta(Di), Num);
84     Msoln = zeros(length(time), 3, length(omegaovector));
85

```

```

86     for i = 1:length(omegaovector)
87         % Initial time
88         Msoln(1:INITi_1,1,i) = 0;
89         Msoln(1:INITi_1,2,i) = 0;
90         Msoln(1:INITi_1,3,i) = 1;
91
92         % Flip time 1
93         tad = time;
94         Vis90 = pi/2/FT_1;
95         M = Msoln(INITi_1, :, i);
96         Msoln(INITi_1:FTi_1,1,i) = M(1);
97         Msoln(INITi_1:FTi_1,2,i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*
            sin(Vis90.*tad(INITi_1:FTi_1));
98         Msoln(INITi_1:FTi_1,3,i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*
            sin(Vis90.*tad(INITi_1:FTi_1));
99
100        % Dephase time 1
101        tad = tad - FT_1;
102        M = Msoln(FTi_1, :, i);
103        Msoln(FTi_1:DPi_1,1,i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) +
            M(2)*sin(omegaovector(i).*tad(FTi_1:DPi_1)));
104        Msoln(FTi_1:DPi_1,2,i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) -
            M(1)*sin(omegaovector(i).*tad(FTi_1:DPi_1)));
105        Msoln(FTi_1:DPi_1,3,i) = M(3);
106
107        % Flip time 2
108        tad = tad - DP_1;
109        Vis90 = pi/FT_2;
110        M = Msoln(DPi_1, :, i);
111        Msoln(DPi_1:FTi_2,1,i) = M(1);
112        Msoln(DPi_1:FTi_2,2,i) = M(2)*cos(Vis90.*tad(DPi_1:FTi_2)) + M(3)*sin(
            Vis90.*tad(DPi_1:FTi_2));
113        Msoln(DPi_1:FTi_2,3,i) = M(3)*cos(Vis90.*tad(DPi_1:FTi_2)) - M(2)*sin(
            Vis90.*tad(DPi_1:FTi_2));
114
115        % Rephase time
116        tad = tad - FT_2;
117        M = Msoln(FTi_2, :, i);
118        Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*
            sin(omegaovector(i).*tad(FTi_2:RPi));
119        Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*
            sin(omegaovector(i).*tad(FTi_2:RPi));
120        Msoln(FTi_2:RPi,3,i) = M(3);
121
122        % Echo time
123        tad = tad - RP;
124        M = Msoln(RPi, :, i);
125        Msoln(RPi:INITi_2,1,i) = M(1);
126        Msoln(RPi:INITi_2,2,i) = M(2);
127        Msoln(RPi:INITi_2,3,i) = M(3);
128
129
130        % Dephase time 2
131        tad = tad - INIT_2;
132        M = Msoln(INITi_2, :, i);
133        Msoln(INITi_2:DPi_2,1,i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPi_2))
            + M(2)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
134        Msoln(INITi_2:DPi_2,2,i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPi_2))
            - M(1)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
135        Msoln(INITi_2:DPi_2,3,i) = M(3);
136     end

```



```

137     for i=1:length(time)
138         Mperp(i,Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i
            ,2,:))^2)/Num;
139     end
140 end
141
142 video = VideoWriter(['5_3_dOmega', '.mp4'], 'MPEG-4');
143
144 % set the frame rate
145 frameRate = .25;
146 video.set('FrameRate', frameRate);
147
148 video.open();
149
150 h = figure;
151 for i=1:length(delta)
152     plot(time(:),Mperp(:,i),'linewidth',2,'LineStyle','-','Color','k')
153
154     grid on
155     box on
156     xlim([0 max(time)]);
157     ylim([0 1]);
158     view(2)
159     title(['$\delta \omega $ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
160     xlabel(Time(s), interpreter, latex, fontsize, 15)
161     ylabel(| $\frac{1}{N} \sum M_{\perp}$ |, interpreter, latex, fontsize, 15)
162
163     frame = getframe(h);
164     video.writeVideo(frame);
165 end
166
167 video.close();
168
169 %% Sensitivity of Cauchy to delta
170 % Author: Imraj Singh
171
172 % Precession frequency
173 omega0 = 50 * pi;
174 delta = [1 2 4 8 16 32 64 128];
175 deltaomega = 0;
176 Mperp = zeros(length(time),length(delta));
177 for Di = 1:length(delta)
178     omegaovector = deltaomega+delta(Di)*tan(pi*(rand(Num,1)-1/2));
179     Msoln = zeros(length(time),3, length(omegaovector));
180
181     for i = 1:length(omegaovector)
182         % Initial time
183         Msoln(1:INITi_1,1,i) = 0;
184         Msoln(1:INITi_1,2,i) = 0;
185         Msoln(1:INITi_1,3,i) = 1;
186
187         % Flip time 1
188         tad = time;
189         Vis90 = pi/2/FTi_1;
190         M = Msoln(INITi_1:FTi_1,i);
191         Msoln(INITi_1:FTi_1,1,i) = M(1);
192         Msoln(INITi_1:FTi_1,2,i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*
            sin(Vis90.*tad(INITi_1:FTi_1));
193         Msoln(INITi_1:FTi_1,3,i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*
            sin(Vis90.*tad(INITi_1:FTi_1));
194

```

```

195 % Dephase time 1
196 tad = tad - FT_1;
197 M = Msoln(FTi_1, :, i);
198 Msoln(FTi_1:DPI_1, 1, i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPI_1)) +
199     M(2)*sin(omegaovector(i).*tad(FTi_1:DPI_1)));
200 Msoln(FTi_1:DPI_1, 2, i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPI_1)) -
201     M(1)*sin(omegaovector(i).*tad(FTi_1:DPI_1)));
202 Msoln(FTi_1:DPI_1, 3, i) = M(3);
203
204 % Flip time 2
205 tad = tad - DP_1;
206 Vis90 = pi/FT_2;
207 M = Msoln(DPi_1, :, i);
208 Msoln(DPi_1:FTi_2, 1, i) = M(1);
209 Msoln(DPi_1:FTi_2, 2, i) = M(2)*cos(Vis90.*tad(DPi_1:FTi_2)) + M(3)*sin(
210     Vis90.*tad(DPi_1:FTi_2));
211 Msoln(DPi_1:FTi_2, 3, i) = M(3)*cos(Vis90.*tad(DPi_1:FTi_2)) - M(2)*sin(
212     Vis90.*tad(DPi_1:FTi_2));
213
214 % Rephase time
215 tad = tad - FT_2;
216 M = Msoln(FTi_2, :, i);
217 Msoln(FTi_2:RPi, 1, i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*
218     sin(omegaovector(i).*tad(FTi_2:RPi));
219 Msoln(FTi_2:RPi, 2, i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*
220     sin(omegaovector(i).*tad(FTi_2:RPi));
221 Msoln(FTi_2:RPi, 3, i) = M(3);
222
223 % Echo time
224 tad = tad - RP;
225 M = Msoln(RPi, :, i);
226 Msoln(RPi:INITi_2, 1, i) = M(1);
227 Msoln(RPi:INITi_2, 2, i) = M(2);
228 Msoln(RPi:INITi_2, 3, i) = M(3);
229
230 % Dephase time 2
231 tad = tad - INIT_2;
232 M = Msoln(INITi_2, :, i);
233 Msoln(INITi_2:DPI_2, 1, i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPI_2))
234     + M(2)*sin(omegaovector(i).*tad(INITi_2:DPI_2));
235 Msoln(INITi_2:DPI_2, 2, i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPI_2))
236     - M(1)*sin(omegaovector(i).*tad(INITi_2:DPI_2));
237 Msoln(INITi_2:DPI_2, 3, i) = M(3);
238
239 end
240 for i=1:length(time)
241     Mperp(i, Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i, 1, :))^2 + sum(Msoln(i
242         , 2, :))^2)/Num;
243 end
244
245 end
246
247 h = figure;
248 for i=1:length(delta)
249     plot(time(:), Mperp(:, i), 'linewidth', 2, 'LineStyle', '-', 'Color', 'k')
250
251     grid on
252     box on
253     xlim([0 max(time)]);
254     ylim([0 1]);
255     view(2)
256     title(['$\Delta$ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)

```

```

247 xlabel(Time(s), interpreter, latex, fontsize, 15)
248 ylabel(| $\frac{1}{N} \sum M_{\perp}$ |, interpreter, latex, fontsize, 15)
249
250 frame(i) = getframe(h);
251 end
252 video = VideoWriter(['5_3_Delta', '.mp4'], 'MPEG-4');
253 frameRate = .25;
254 video.set('FrameRate', frameRate);
255 video.open();
256 video.writeVideo(frame);
257 video.close();
258
259 %% Sensitivity of Cauchy to omega
260 % Author: Imraj Singh
261
262 % Precession frequency
263 omega0 = 50 * pi;
264 delta1 = 4;
265 deltaomega = [-100 -50 -25 -10 -5 0 5 10 25 50 100];
266 Mperp = zeros(length(time), length(delta));
267 for Di = 1:length(delta)
268     omegaovector = delta(Di)+delta1*tan(pi*(rand(Num,1)-1/2));
269     Msoln = zeros(length(time),3, length(omegaovector));
270
271     for i = 1:length(omegaovector)
272         % Initial time
273         Msoln(1:INITi_1,1,i) = 0;
274         Msoln(1:INITi_1,2,i) = 0;
275         Msoln(1:INITi_1,3,i) = 1;
276
277         % Flip time 1
278         tad = time;
279         Vis90 = pi/2/FT_1;
280         M = Msoln(INITi_1, :, i);
281         Msoln(INITi_1:FTi_1,1,i) = M(1);
282         Msoln(INITi_1:FTi_1,2,i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*
            sin(Vis90.*tad(INITi_1:FTi_1));
283         Msoln(INITi_1:FTi_1,3,i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*
            sin(Vis90.*tad(INITi_1:FTi_1));
284
285         % Dephase time 1
286         tad = tad - FT_1;
287         M = Msoln(FTi_1, :, i);
288         Msoln(FTi_1:DPi_1,1,i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) +
            M(2)*sin(omegaovector(i).*tad(FTi_1:DPi_1)));
289         Msoln(FTi_1:DPi_1,2,i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) -
            M(1)*sin(omegaovector(i).*tad(FTi_1:DPi_1)));
290         Msoln(FTi_1:DPi_1,3,i) = M(3);
291
292         % Flip time 2
293         tad = tad - DP_1;
294         Vis90 = pi/FT_2;
295         M = Msoln(DPi_1, :, i);
296         Msoln(DPi_1:FTi_2,1,i) = M(1);
297         Msoln(DPi_1:FTi_2,2,i) = M(2)*cos(Vis90.*tad(DPi_1:FTi_2)) + M(3)*sin(
            Vis90.*tad(DPi_1:FTi_2));
298         Msoln(DPi_1:FTi_2,3,i) = M(3)*cos(Vis90.*tad(DPi_1:FTi_2)) - M(2)*sin(
            Vis90.*tad(DPi_1:FTi_2));
299
300         % Rephase time
301         tad = tad - FT_2;

```

```

302     M = Msoln(FTi_2, :, i);
303     Msoln(FTi_2:RPi, 1, i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*
        sin(omegaovector(i).*tad(FTi_2:RPi));
304     Msoln(FTi_2:RPi, 2, i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*
        sin(omegaovector(i).*tad(FTi_2:RPi));
305     Msoln(FTi_2:RPi, 3, i) = M(3);
306
307     % Echo time
308     tad = tad - RP;
309     M = Msoln(RPi, :, i);
310     Msoln(RPi:INITi_2, 1, i) = M(1);
311     Msoln(RPi:INITi_2, 2, i) = M(2);
312     Msoln(RPi:INITi_2, 3, i) = M(3);
313
314
315     % Dephase time 2
316     tad = tad - INIT_2;
317     M = Msoln(INITi_2, :, i);
318     Msoln(INITi_2:DPi_2, 1, i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPi_2))
        + M(2)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
319     Msoln(INITi_2:DPi_2, 2, i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPi_2))
        - M(1)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
320     Msoln(INITi_2:DPi_2, 3, i) = M(3);
321 end
322 for i=1:length(time)
323     Mperp(i, Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i, 1, :))^2 + sum(Msoln(i
        , 2, :))^2)/Num;
324 end
325 end
326
327
328 % set the frame rate
329
330 h = figure;
331 for i=1:length(delta)
332     plot(time(:), Mperp(:, i), 'linewidth', 2, 'LineStyle', '-', 'Color', 'k')
333
334     grid on
335     box on
336     xlim([0 max(time)]);
337     ylim([0 1]);
338     view(2)
339     title(['$\omega_0$ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
340     xlabel('Time (s)', interpreter, latex, fontsize, 15)
341     ylabel('|\frac{1}{N} \sum M_{\perp}|', interpreter, latex, fontsize, 15)
342     frame(i) = getframe(gcf);
343 end
344 video = VideoWriter(['5_3-omega', '.mp4'], 'MPEG-4');
345 frameRate = .25;
346 video.set('FrameRate', frameRate);
347 video.open();
348 video.writeVideo(frame);
349 video.close();

```

#### E.4 Problem 5.3 components plot:

```

1 clc
2 clear
3
4 %% Sensitivity of Cauchy to omega_0

```

```

5 % Author: Imraj Singh
6
7 % Given parameters
8
9 % Number of sample in each part of sequence
10 N = 101;
11
12 % Number of isochromats
13 Num = 10001;
14
15 % Prescribe T1
16 T2 = 1;
17 T1 = T2 * 2;
18
19 % Calculate neccessary parameters
20
21 % Inital time 1
22 INIT_1 = 0;
23 tINIT_1 = linspace(0, INIT_1, N);
24
25 % Flip time 1
26 FT_1 = .001;
27 tFT_1 = linspace(0, FT_1, N);
28
29 % Dephasing time 1
30 DP_1 = .5;
31 tDP_1 = linspace(FT_1, DP_1 + FT_1, N);
32
33 % Flip time 2
34 FT_2 = .001;
35 tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);
36
37 % Rephasing time
38 RP = DP_1;
39 tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);
40
41 % Inital time 2
42 INIT_2 = 0;
43 tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP, N);
44
45 % Dephasing time 2
46 DP_2 = DP_1;
47 tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);
48
49 time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];
50
51 % Indices
52 % Initial time
53 INITi_1 = N;
54
55 % Flip time 1
56 FTi_1 = INITi_1 + N;
57
58 % Dephasing time 1
59 DPi_1 = FTi_1 + N;
60
61 % Flip time 2
62 FTi_2 = DPi_1 + N;
63
64 % Rephasing time
65 RPi = FTi_2 + N;

```

```

66
67 % Initial time
68 INITi_2 = RPi + N;
69
70 % Dephasing time 2
71 DPi_2 = INITi_2 + N;
72
73
74 % Precession frequency
75 omega0 = 50 * pi;
76 delta1 = 4;
77 delta = [0 5 10 25 50 100 200];
78 Mperp = zeros(length(time),length(delta));
79 for Di = 1:length(delta)
80     omegaovector = delta(Di)+delta1*tan(pi*(rand(Num,1)-1/2));
81     Msoln = zeros(length(time),3, length(omegaovector));
82
83     for i = 1:length(omegaovector)
84         % Initial time
85         Msoln(1:INITi_1,1,i) = 0;
86         Msoln(1:INITi_1,2,i) = 0;
87         Msoln(1:INITi_1,3,i) = 1;
88
89         % Flip time 1
90         tad = time;
91         Vis90 = pi/2/FT_1;
92         M = Msoln(INITi_1,:,i);
93         Msoln(INITi_1:FTi_1,1,i) = M(1);
94         Msoln(INITi_1:FTi_1,2,i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*
            sin(Vis90.*tad(INITi_1:FTi_1));
95         Msoln(INITi_1:FTi_1,3,i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*
            sin(Vis90.*tad(INITi_1:FTi_1));
96
97         % Dephase time 1
98         tad = tad - FT_1;
99         M = Msoln(FTi_1,:,i);
100        Msoln(FTi_1:DPi_1,1,i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) +
            M(2)*sin(omegaovector(i).*tad(FTi_1:DPi_1)));
101        Msoln(FTi_1:DPi_1,2,i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) -
            M(1)*sin(omegaovector(i).*tad(FTi_1:DPi_1)));
102        Msoln(FTi_1:DPi_1,3,i) = M(3);
103
104        % Flip time 2
105        tad = tad - DP_1;
106        Vis90 = pi/FT_2;
107        M = Msoln(DPi_1,:,i);
108        Msoln(DPi_1:FTi_2,1,i) = M(1);
109        Msoln(DPi_1:FTi_2,2,i) = M(2)*cos(Vis90.*tad(DPi_1:FTi_2)) + M(3)*sin(
            Vis90.*tad(DPi_1:FTi_2));
110        Msoln(DPi_1:FTi_2,3,i) = M(3)*cos(Vis90.*tad(DPi_1:FTi_2)) - M(2)*sin(
            Vis90.*tad(DPi_1:FTi_2));
111
112        % Rephase time
113        tad = tad - FT_2;
114        M = Msoln(FTi_2,:,i);
115        Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*
            sin(omegaovector(i).*tad(FTi_2:RPi));
116        Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*
            sin(omegaovector(i).*tad(FTi_2:RPi));
117        Msoln(FTi_2:RPi,3,i) = M(3);
118

```

```

119 % Echo time
120 tad = tad - RP;
121 M = Msoln(RPi,:,i);
122 Msoln(RPi:INITi_2,1,i) = M(1);
123 Msoln(RPi:INITi_2,2,i) = M(2);
124 Msoln(RPi:INITi_2,3,i) = M(3);
125
126
127 % Dephase time 2
128 tad = tad - INIT_2;
129 M = Msoln(INITi_2,:,i);
130 Msoln(INITi_2:DPi_2,1,i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPi_2))
    + M(2)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
131 Msoln(INITi_2:DPi_2,2,i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPi_2))
    - M(1)*sin(omegaovector(i).*tad(INITi_2:DPi_2));
132 Msoln(INITi_2:DPi_2,3,i) = M(3);
133 end
134 for i=1:length(time)
135     Mperp(i,1,Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i
    ,2,:))^2)/Num;
136     Mperp(i,2,Di) = exp(-time(i)/T2)*sum(Msoln(i,1,:))/Num;
137     Mperp(i,3,Di) = exp(-time(i)/T2)*sum(Msoln(i,2,:))/Num;
138 end
139 end
140
141 h = figure;
142 for i=1:length(delta)
143     plot(time(:),Mperp(:,2,i),'linewidth',2,'LineStyle','-','Color','r')
144     hold on
145     plot(time(:),Mperp(:,3,i),'linewidth',2,'LineStyle','-','Color','b')
146     plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','—','Color','k')
147     plot(time(:),-exp(-time(:)/T2),'linewidth',2,'LineStyle','—','Color','k')
148     hold off
149     grid on
150     box on
151     xlim([0 max(time)]);
152     ylim([-1 1]);
153     title(['$\Delta$ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
154     xlabel(Time (s), interpreter, latex, fontsize, 15)
155     ylabel(Normalised average  $M_{\perp}$ , interpreter, latex, fontsize, 15)
156     legend( $\frac{1}{N} \sum M_{x'}$ ,  $\frac{1}{N} \sum M_{y'}$ ,  $T_2$  envelope, interpreter, latex, fontsize, 10)
157
158     frame(i) = getframe(h);
159 end
160 video = VideoWriter(['5_3-omega-comp', '.mp4'], 'MPEG-4');
161 frameRate = .25;
162 video.set('FrameRate', frameRate);
163 video.open();
164 video.writeVideo(frame);
165 video.close();

```

## E.5 Problem 5.4:

```

1 clc
2 clear
3
4 %% Hahn echo
5 % Author: Imraj Singh 03/11/2020
6
7 % Given parameters

```

```

8
9 % Number of sample in each part of sequence
10 N = 11;
11
12 % Number of isochromats
13 Num = 1001;
14
15 % Prescribe T1
16 T2 = 1;
17 T1 = T2 * 2;
18
19 % Calculate neccessary parameters
20
21 % Precession frequency
22 omega0 = 50 * pi;
23 deltaomega = 0;
24 delta = 2;
25 omegaovector = deltaomega+delta*tan(pi*(rand(Num,1)-1/2));
26
27 % Inital time 1
28 INIT_1 = 0;
29 tINIT_1 = linspace(0, INIT_1, N);
30
31 % Flip time 1
32 FT_1 = .001;
33 tFT_1 = linspace(0, FT_1, N);
34
35 % Dephasing time 1
36 DP_1 = .5;
37 tDP_1 = linspace(FT_1, DP_1 + FT_1, N);
38
39 % Flip time 2
40 FT_2 = .002;
41 tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);
42
43 % Rephasing time
44 RP = DP_1;
45 tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);
46
47 % Inital time 2
48 INIT_2 = 0;
49 tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP, DP_1 + FT_1 + FT_2 + RP, N);
50
51 % Dephasing time 2
52 DP_2 = DP_1;
53 tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);
54
55 time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];
56
57 % Indices
58 % Initial time
59 INITi_1 = N;
60
61 % Flip time 1
62 FTi_1 = INITi_1 + N;
63
64 % Dephasing time 1
65 DPi_1 = FTi_1 + N;
66
67 % Flip time 2
68 FTi_2 = DPi_1 + N;

```



```

69
70 % Rephasing time
71 RPi = FTi_2 + N;
72
73 % Initial time
74 INITi_2 = RPi + N;
75
76 % Dephasing time 2
77 DPi_2 = INITi_2 + N;
78
79
80 Msoln = zeros(length(time),3, length(omegaovector));
81
82 for i = 1:length(omegaovector)
83     % Initial time
84     Msoln(1:INITi_1,1,i) = 0;
85     Msoln(1:INITi_1,2,i) = 0;
86     Msoln(1:INITi_1,3,i) = 1;
87
88     % Flip time 1
89     tad = time;
90     Vis90 = pi/2/FT_1;
91     M = Msoln(INITi_1,:,i);
92     Msoln(INITi_1:FTi_1,1,i) = M(1);
93     Msoln(INITi_1:FTi_1,2,i) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*sin(
        Vis90.*tad(INITi_1:FTi_1));
94     Msoln(INITi_1:FTi_1,3,i) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*sin(
        Vis90.*tad(INITi_1:FTi_1));
95
96     % Dephase time 1
97     tad = tad - FT_1;
98     M = Msoln(FTi_1,:,i);
99     Msoln(FTi_1:DPi_1,1,i) = (M(1)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) + M(2)
        *sin(omegaovector(i).*tad(FTi_1:DPi_1)));
100    Msoln(FTi_1:DPi_1,2,i) = (M(2)*cos(omegaovector(i).*tad(FTi_1:DPi_1)) - M(1)
        *sin(omegaovector(i).*tad(FTi_1:DPi_1)));
101    Msoln(FTi_1:DPi_1,3,i) = M(3);
102
103    % Flip time 2
104    tad = tad - DP_1;
105    Vis90 = pi/2/FT_2;
106    M = Msoln(DPi_1,:,i);
107    Msoln(DPi_1:FTi_2,1,i) = M(1);
108    Msoln(DPi_1:FTi_2,2,i) = M(2)*cos(Vis90.*tad(DPi_1:FTi_2)) + M(3)*sin(Vis90
        .*tad(DPi_1:FTi_2));
109    Msoln(DPi_1:FTi_2,3,i) = M(3)*cos(Vis90.*tad(DPi_1:FTi_2)) - M(2)*sin(Vis90
        .*tad(DPi_1:FTi_2));
110
111    % Rephase time
112    tad = tad - FT_2;
113    M = Msoln(FTi_2,:,i);
114    Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
115    Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
116    Msoln(FTi_2:RPi,3,i) = M(3);
117
118    % Echo time
119    tad = tad - RP;
120    M = Msoln(RPi,:,i);
121    Msoln(RPi:INITi_2,1,i) = M(1);

```

```

122     Msoln(RPi:INITi_2,2,i) = M(2);
123     Msoln(RPi:INITi_2,3,i) = M(3);
124
125
126     % Dephase time 2
127     tad = tad - INITi_2;
128     M = Msoln(INITi_2, :, i);
129     Msoln(INITi_2:DPI_2,1,i) = M(1)*cos(omegaovector(i).*tad(INITi_2:DPI_2)) + M
        (2)*sin(omegaovector(i).*tad(INITi_2:DPI_2));
130     Msoln(INITi_2:DPI_2,2,i) = M(2)*cos(omegaovector(i).*tad(INITi_2:DPI_2)) - M
        (1)*sin(omegaovector(i).*tad(INITi_2:DPI_2));
131     Msoln(INITi_2:DPI_2,3,i) = M(3);
132 end
133 Mperp = zeros(length(time),1);
134 for i=1:length(time)
135     Mperp(i) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/
        Num;
136 end
137
138 %% Animation module
139
140 video = VideoWriter(['5_4', '.mp4'], 'MPEG-4');
141
142 % set the frame rate
143 frameRate = N/8;
144 video.set('FrameRate', frameRate);
145
146 video.open();
147
148 h = figure;
149
150
151 for i=INITi_1:length(time)
152     subplot(2,2,1)
153
154     quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
        LineStyle','-','Color','r')
155     hold on
156     for j=1:100:length(omegaovector)
157         quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
            LineStyle','-','Color','r')
158     end
159     grid on
160     box on
161     hold off
162     xlim([-1 1]);
163     ylim([-1 1]);
164     zlim([-1 1]);
165     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
166     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
167     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
168
169     subplot(2,2,2)
170     quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
        LineStyle','-','Color','r')
171     hold on
172     for j=1:100:length(omegaovector)
173         quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
            LineStyle','-','Color','r')
174     end
175     grid on

```

```

176     box on
177     xlim([-1 1]);
178     ylim([-1 1]);
179     zlim([-1 1]);
180     hold off
181     view(2)
182     xlabel( $M_{x'}$ , interpreter, latex, fontsize, 15)
183     ylabel( $M_{y'}$ , interpreter, latex, fontsize, 15)
184     zlabel( $M_{z'}$ , interpreter, latex, fontsize, 15)
185
186
187     subplot(2,2,3:4)
188
189     plot(time(1:i), Mperp(1:i), 'linewidth',2, 'LineStyle','-', 'Color','k')
190     hold on
191     plot(time(:), exp(-time(:)/T2), 'linewidth',2, 'LineStyle','—', 'Color','k')
192     hold off
193     grid on
194     box on
195     ylim([0 1]);
196     xlim([0 max(time)]);
197     xlabel(Time (s), interpreter, latex, fontsize, 15)
198     ylabel(Normalised signal, interpreter, latex, fontsize, 15)
199     legend(|  $\frac{1}{N} \sum M_{\perp}$  |,  $T_2$  envelope, interpreter, latex, fontsize, 10)
200
201 % Initial
202 if i > 0 && i <= INITi_1
203     title('Magnetisation vector along  $B_0$  initially', interpreter, latex,
204           fontsize, 15)
205
206 % First flip
207 elseif i > INITi_1 && i <= FTi_1
208     xlim([0 FTi_1]);
209     title('RF pulse  $\pi/2$ ', interpreter, latex, fontsize, 15)
210
211 % First Dephase
212 elseif i > FTi_1 && i <= DPi_1
213     title(['Isocromats dephasing,  $t =$ ', num2str(round(time(i) - INIT_1 -
214           FT_1, 3)), ' s'], interpreter, latex, fontsize, 15)
215
216 % Second flip
217 elseif i > DPi_1 && i <= FTi_2
218     title(['RF pulse  $\pi$ ,  $t = \tau =$ ', num2str(round(DP_1, 3)), ' s'],
219           interpreter, latex, fontsize, 15)
220
221 % Rephasing
222 elseif i > FTi_2 && i <= RPi
223     title(['Isocromats rephasing,  $t =$ ', num2str(round(time(i) - INIT_1 -
224           FT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
225
226 % Echo
227 elseif i > RPi && i <= INITi_2
228     title(['Echo time,  $t = T_E = 2 \tau =$ ', num2str(round(DP_1 * 2, 3)), '
229           s'], interpreter, latex, fontsize, 15)
230
231 % Dephasing
232 elseif i > INITi_2 && i <= DPi_2
233     title(['Isocromats dephasing,  $t =$ ', num2str(round(time(i) - INIT_1 -
234           FT_1 - FT_2, 3)), ' s (Relaxation)'], interpreter, latex, fontsize, 15)
235
236 end

```

```
231  
232     frame = getframe(h);  
233     video.writeVideo(frame);  
234 end  
235  
236 video.close();
```

### Code for all the plots