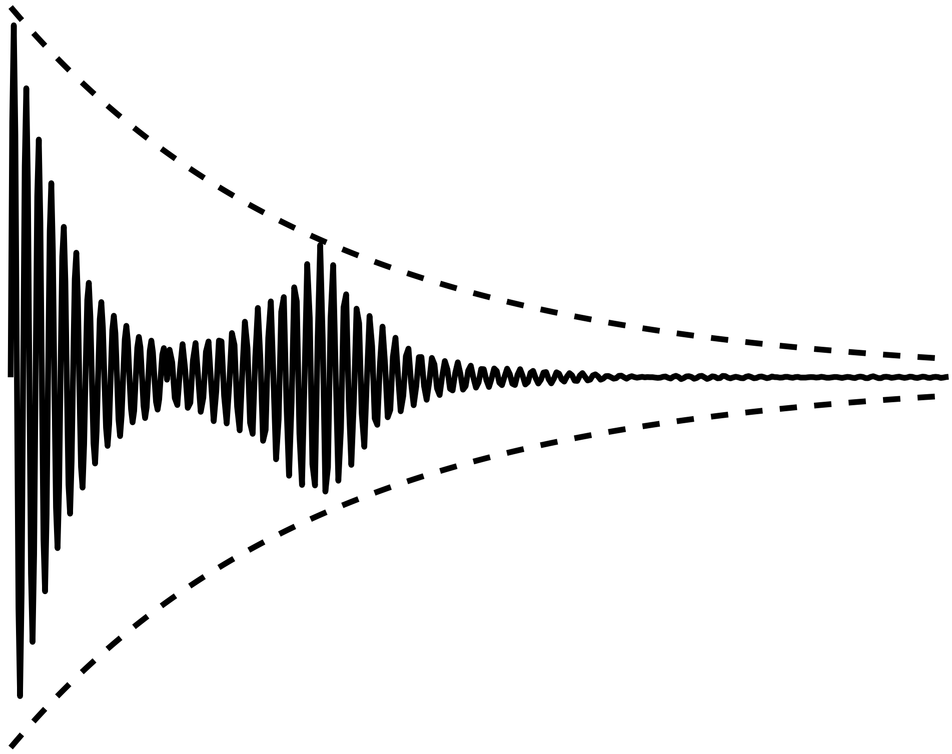
Coursework 1



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

1 Spin Excess

Spin excess is given by the following equation:

$$\text{Spin excess} \approx N\frac{\hbar\omega_o}{2kT} \tag{1}$$

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 2π, it is a universal constant; ω_o is the Larmor frequency and is dependent on the gyromagnetic ratio γ 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 γ 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 θ, 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:

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

To verify Eqn. (2) consider the unit vectors $\hat{x}$, $\hat{y}$ and $\hat{z}$ defined previously. The unit vectors are rotated by θ clockwise around the z-axis to produce $\hat{x}_{rot}$, $\hat{y}_{rot}$ and $\hat{z}_{rot}$. The rotated unit vectors are calculated from $\hat{x}_{rot} = R_z(\theta)\cdot\hat{x}$, $\hat{y}_{rot} = R_z(\theta)\cdot\hat{y}$ and $\hat{z}_{rot} = R_z(\theta)\cdot\hat{z}$. 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 θ 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).

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

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

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

2.4 Forced procession.. 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: $\Delta\theta = 90°$ 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: $B_1 = \frac{\Delta\theta}{\gamma\tau} = 5.9\mu T$.

In the rotating FoR the flip of the spin is a 90° 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 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 scanner 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 2.68×10^8 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 procession will include no z component. This is seen in animation '3_1' where there is a clockwise procession 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 precessing 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 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 precessing slower than Larmor frequency. This is 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 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 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°), using a RF pulse, to the transverse plane.

2. Free procession 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 π radians (180°), using a RF pulse, to the $-B_0$ direction.

2. Free procession with relaxation for spin from the $-B_0$ direction to B_0 direction.

3. Flip the spins by $\pi/2$ radians (90°), 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 procession 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 π), 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 50π 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°), using a RF pulse along the x' axis, to the y' axis.

2. Free procession 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 π radians (180°), 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, $\mid M_\perp \mid$. 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 $\mid \frac{1}{N}\sum M_\perp \mid$. 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 π 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 π 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 τ 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 Δ and ω_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 Δ increase the amount of dephasing, effectively T_2' increases with δ. At very high values of Δ 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 ω_0 has no effect on the amount of dephasing. Changing values of ω_0 does not have an effect when viewing the magnitude of the average transverse components. Using the magnitude obscures the effect of ω_0. As ω_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 ω_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 π 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:

```
function SE = calcSE(T,B, Gamma)
%Function to calculate the approximate spin excess
%   Inputs
%   T = Temperature of the medium
%   B = Magnetic field strength
%   Gamma = Gyromagnetic ratio of medium
%
%   Output
%   SE = Spin excess

% Parameters

% Number of spins calculated from typical voxel volume 2*2*5mm = 0.02
% Avogadro's number = 6.02*10^23
% One voxel has 2 x 6.02 x10^23 x 0.02 / 18 protons
N = 1.338 * 10^21;

% Reduced planks constant
hbar = (6.62607015*10^(-34))/(2*pi);

% Boltzman constant
k = 1.38064852 * 10^(-23);

SE = N*(hbar*B*Gamma)/(2*k*T);
end
```

B Problem 2 code:

B.1 Rotation matrix z-axis:

```
function [rotated] = rotateZ(vector,theta)
%Rotate around z-axis
%   vector - the vector you want rotated, must have three dimensions
%   theta - angle in radians to rotate the vector around the z-axis
%   clockwise
%

rotated(1) = vector(1)*cos(theta) + vector(2)*sin(theta);
rotated(2) = vector(2)*cos(theta) - vector(1)*sin(theta);
rotated(3) = vector(3);
end
```

B.2 Rotation matrix x-axis:

```
function [rotated] = rotateX(vector,theta)
%Rotate around x-axis
%   vector - the vector you want rotated, must have three dimensions
%   theta - angle in radians to rotate the vector around the x-axis
%   clockwise
%

rotated(1) = vector(1);
rotated(2) = vector(2)*cos(theta) + vector(3)*sin(theta);
rotated(3) = vector(3)*cos(theta) - vector(2)*sin(theta);
end
```

B.3 Rotation matrix y-axis:

```
function [rotated] = rotateY(vector,theta)
%Rotate around y-axis
%   vector - the vector you want rotated, must have three dimensions
%   theta - angle in radians to rotate the vector around the y-axis
%   clockwise
%

rotated(1) = vector(1)*cos(theta) - vector(3)*sin(theta);
rotated(2) = vector(2);
rotated(3) = vector(3)*cos(theta) + vector(1)*sin(theta);
end
```

B.4 Animations of rotations:

```
% housekeeping
clc
clear

%% Rotation around the z-axis - visualisation
% Author - Imraj 11/11/2020

% initialise the video
video = VideoWriter(['2_1', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = 100;
video.set('FrameRate', frameRate);

% open video
video.open();

% initialise the figure
h = figure;

% define unit vectors
X = [1,0,0]';
Y = [0,1,0]';
Z = [0,0,1]';

% define angle of rotation 1 degree
theta = 2*pi/360;

% specify animation captures each degree of rotation
for i=0:360
    quiver3(0,0,0,X(1),X(2),X(3),'linewidth',2,'LineStyle','-','Color','k')
    hold on
    quiver3(0,0,0,Y(1),Y(2),Y(3),'linewidth',2,'LineStyle','--','Color','b')
    quiver3(0,0,0,Z(1),Z(2),Z(3),'linewidth',2,'LineStyle','-.','Color','r')

    % format title, legend, labels, limits, grid and box
    title(['z-axis rotation of unit vectors, $\theta =$', num2str(theta*i
        *360/(2*pi)),'$^\circ$'], interpreter, latex, fontsize, 15)
    legend('$\hat{x}$','$\hat{y}$','$\hat{z}$', interpreter, latex, fontsize, 15)
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
```

```
    ylim([-1 1]);
    zlim([-1 1]);
    hold off

    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);

    % rotate by theta
    X = rotateZ(X,theta);
    Y = rotateZ(Y,theta);
    Z = rotateZ(Z,theta);
end

video.close();

%% Rotation around the x-axis - visualisation
% Author - Imraj

% initialise the video
video = VideoWriter(['2_2', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = 100;
video.set('FrameRate', frameRate);

% open video
video.open();

% initialise the figure
h = figure;

% define unit vectors
X = [1,0,0]';
Y = [0,1,0]';
Z = [0,0,1]';

% define angle of rotation 1 degree
theta = 2*pi/360;

% specify animation captures each degree of rotation
for i=0:360
    quiver3(0,0,0,X(1),X(2),X(3),'linewidth',2,'LineStyle','-','Color','k')
    hold on
    quiver3(0,0,0,Y(1),Y(2),Y(3),'linewidth',2,'LineStyle','--','Color','b')
    quiver3(0,0,0,Z(1),Z(2),Z(3),'linewidth',2,'LineStyle','-.','Color','r')

    % format title, legend, labels, limits, grid and box
    title(['x-axis rotation of unit vectors, $\theta =$', num2str(theta*i
        *360/(2*pi)),'$^\circ$'], interpreter, latex, fontsize, 15)
    legend('$\hat{x}$','$\hat{y}$','$\hat{z}$', interpreter, latex, fontsize, 15)
    xlabel(M_x, interpreter, latex, fontsize, 15)
    ylabel(M_y, interpreter, latex, fontsize, 15)
    zlabel(M_z, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    hold off

```

```
    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);

    % rotate by theta
    X = rotateX(X,theta);
    Y = rotateX(Y,theta);
    Z = rotateX(Z,theta);
end

video.close();

%% Rotation around the y-axis - visualisation
% Author - Imraj

% initialise the video
video = VideoWriter(['2_3', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = 100;
video.set('FrameRate', frameRate);

% open video
video.open();

% initialise the figure
h = figure;

% define unit vectors
X = [1,0,0]';
Y = [0,1,0]';
Z = [0,0,1]';

% define angle of rotation 1 degree
theta = 2*pi/360;

% specify animation captures each degree of rotation
for i=0:360
    quiver3(0,0,0,X(1),X(2),X(3),'linewidth',2,'LineStyle','-','Color','k')
    hold on
    quiver3(0,0,0,Y(1),Y(2),Y(3),'linewidth',2,'LineStyle','--','Color','b')
    quiver3(0,0,0,Z(1),Z(2),Z(3),'linewidth',2,'LineStyle','-.','Color','r')

    % format title, legend, labels, limits, grid and box
    title(['y-axis rotation of unit vectors, $\theta =$', num2str(theta*i
        *360/(2*pi)),'$^\circ$'], interpreter, latex, fontsize, 15)
    legend('$\hat{x}$','$\hat{y}$','$\hat{z}$', interpreter, latex, fontsize, 15)
    xlabel(M_x, interpreter, latex, fontsize, 15)
    ylabel(M_y, interpreter, latex, fontsize, 15)
    zlabel(M_z, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    hold off

    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);

```

```
    % rotate by theta
    X = rotateY(X,theta);
    Y = rotateY(Y,theta);
    Z = rotateY(Z,theta);
end

video.close();
```

B.5 Problem 2.4:

```
% housekeeping
clc
clear
%% Forced procession ignoring relaxation
% Author: Imraj Singh 03/11/2020

% Given parameters

% Magnitisation aligned along z-axis initially
M = [0, 0, 1]';

% Duration of the RF pulse
t = 1*10^(-3);

% Flip angle
dtheta = 90*pi/180;

% Gyromagnetic ratio for proton in hydrogen
gamma = 2.68*10^8;

% Calculate neccessary parameters

% The magnetic field strength caused by RF
B1 = dtheta/(gamma*t);

% Precession frequency
omega1 = gamma*B1;

% Define modelling parameters

% Duration and timestep of model
time = linspace(0,t,101);

Msoln = zeros(101,3);

Msoln(:,1) = M(1);
Msoln(:,2) = M(2)*cos(omega1.*time) + M(3)*sin(omega1.*time);
Msoln(:,3) = M(3)*cos(omega1.*time) - M(2)*sin(omega1.*time);

%% Animation

% initialise the video
video = VideoWriter(['2_4', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = 10;
video.set('FrameRate', frameRate);

video.open();

```

```
% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,0,1,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')

    % format title, legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off


    subplot(2,2,2)
    quiver(0,0,0,1,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot(Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')

    % format title, legend, labels, limits, grid and box
    xlabel($M_{y'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([0 1]);
    hold off

    subplot(2,2,3:4)
    plot(time(1:i)*1000,Msoln(1:i,2),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i)*1000,Msoln(1:i,3),'linewidth',2,'LineStyle','-.','Color','b');

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)*1000),' ms'], interpreter, latex, fontsize, 15)
    xlabel(Time (ms), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend($M_{y'}$,$M_{z'}$, interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time*1000)]);
    ylim([0 1]);
    hold off

    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end
```

```
video.close();
```

C Problem 3 code:

C.1 Problem 3.1:

```
% housekeeping
clc
clear
%% Free procession ignoring relaxation
% Author: Imraj Singh 11/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Larmor frequency
omega0 = 2 * 2 * pi;

% End time
t = 4;
resolution = 100;

% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);



Msoln(:,1) = M(1)*cos(omega0.*time) + M(2)*sin(omega0.*time);
Msoln(:,2) = M(2)*cos(omega0.*time) - M(1)*sin(omega0.*time);
Msoln(:,3) = M(3);


%% Animation module
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['3_1', '.mp4'], 'MPEG-4');

% set the frame rate quarter of real speed
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
```

```
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(2)
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend($M_x$,$M_y$, interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time)]);
    ylim([-1 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

C.2 Problem 3.2:

```
% housekeeping
clc
clear
%% Free procession with relaxation
% Author: Imraj Singh 03/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Larmor frequency
omega0 = 2 * 2 * pi;

% End time
t = 6;
resolution = 50;

% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);

Msoln(:,1) = exp(-time./T2).*(M(1)*cos(omega0.*time) + M(2)*sin(omega0.*time));
Msoln(:,2) = exp(-time./T2).*(M(2)*cos(omega0.*time) - M(1)*sin(omega0.*time));
Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));


%% Animation module
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['3_2', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
```

```
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(2)
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend($M_x$,$M_y$, $M_z$,interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time)]);
    ylim([-1 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();

%% Animation module - just the 3D
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['3_2_3D', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution;
```

```
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off
    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

C.3 Problem 3.3:

```
% housekeeping
clc
clear
%% Free procession with relaxation rotating FoR
% Author: Imraj Singh 03/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% End time
t = 6;
resolution = 50;

```

```
% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);


Msoln(:,1) = exp(-time./T2).*M(1);
Msoln(:,2) = exp(-time./T2).*M(2);
Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));


%% Animation

% initialise the video
video = VideoWriter(['3_3', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')
    hold on
    plot(Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver(0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_{y'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([0 1]);
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
```

```
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend($M_{x'}$,$M_{y'}$, $M_{z'}$,interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time)]);
    ylim([0 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

C.4 Problem 3.4:

```
% housekeeping
clc
clear
%% Free procession with relaxation rotating FoR at Larmor
% Faster than Larmor by omega0/10
% Author: Imraj Singh 03/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% End time
t = 6;
resolution = 50;

% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);



Msoln(:,1) = exp(-time./T2).*(M(1)*cos(omega0/10.*time) + M(2)*sin(omega0/10.*
    time));
Msoln(:,2) = exp(-time./T2).*(M(2)*cos(omega0/10.*time) - M(1)*sin(omega0/10.*
    time));
Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));
```

```


%% Animation module
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['3_4', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(2)
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
```

```
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend($M_{x'}$,$M_{y'}$, $M_{z'}$,interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time)]);
    ylim([-1 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

C.5 Problem 3.4:

```
% housekeeping
clc
clear
%% Free procession with relaxation rotating FoR at Larmor
% Faster than Larmor by omega0/10
% Author: Imraj Singh 03/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% End time
t = 6;
resolution = 50;

% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);



Msoln(:,1) = exp(-time./T2).*(M(1)*cos(omega0/10.*time) + M(2)*sin(omega0/10.*
    time));
Msoln(:,2) = exp(-time./T2).*(M(2)*cos(omega0/10.*time) - M(1)*sin(omega0/10.*
    time));
Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));


```

```
%% Animation module
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['3_4', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(2)
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

```

```
    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend($M_{x'}$,$M_{y'}$, $M_{z'}$,interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time)]);
    ylim([-1 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

C.6 Problem 3.5:

```
% housekeeping
clc
clear
%% Free procession with relaxation rotating FoR at Larmor
% Faster than Larmor by omega0/10
% Author: Imraj Singh 03/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% End time
t = 6;
resolution = 50;

% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);



Msoln(:,1) = exp(-time./T2).*(M(1)*cos(-omega0/10.*time) + M(2)*sin(-omega0/10.*
    time));
Msoln(:,2) = exp(-time./T2).*(M(2)*cos(-omega0/10.*time) - M(1)*sin(-omega0/10.*
    time));
Msoln(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));


%% Animation module
% Author: Imraj Singh 03/11/2020
```

```
% initialise the video
video = VideoWriter(['3_4', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(2)
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
```

```
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend(M_{x'},M_{y'}, M_{z'},interpreter, latex, fontsize, 10)
    grid on
    box on
    xlim([0 max(time)]);
    ylim([-1 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

C.7 Comparison between 3.4 and 3.5:

```
% housekeeping
clc
clear
%% Free procession with relaxation rotating FoR at Larmor
% Faster than Larmor by omega0/10
% Author: Imraj Singh 03/11/2020

% Given parameters
% Magnitisation aligned along y-axis initially
M = [0, 1, 0]';
M0 = [0, 0, 1]';

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% End time
t = 6;
resolution = 50;

% Duration and timestep of model
time = linspace(0, t, t * resolution + 1);

Msolnf(:,1) = exp(-time./T2).*(M(1)*cos(omega0*11/10.*time) + M(2)*sin(omega0
    *11/10.*time));
Msolnf(:,2) = exp(-time./T2).*(M(2)*cos(omega0*11/10.*time) - M(1)*sin(omega0
    *11/10.*time));
Msolnf(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));

Msolns(:,1) = exp(-time./T2).*(M(1)*cos(omega0*9/10.*time) + M(2)*sin(omega0
    *9/10.*time));
Msolns(:,2) = exp(-time./T2).*(M(2)*cos(omega0*9/10.*time) - M(1)*sin(omega0
    *9/10.*time));
Msolns(:,3) = M(3)*exp(-time./T1) + M0(3)*(1-exp(-time./T1));


```

```matlab


%% Animation module - just the 3D
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['3_45_comp', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = resolution/4;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    quiver3(0,0,0,Msolns(i,1),Msolns(i,2),Msolns(i,3),'linewidth',4,'LineStyle','-','Color','r')
    hold on
    quiver3(0,0,0,Msolnf(i,1),Msolnf(i,2),Msolnf(i,3),'linewidth',4,'LineStyle','-','Color','b')
    quiver3(0,0,0,0,1,0,'linewidth',2,'LineStyle','--','Color','k')
    plot3(Msolns(1:i,1),Msolns(1:i,2),Msolns(1:i,3),'r--','linewidth',2)
    plot3(Msolnf(1:i,1),Msolnf(1:i,2),Msolnf(1:i,3),'b--','linewidth',2)

    % format title, legend, labels, limits, grid and box
    title(['Time: ', num2str(time(i)),' s'], interpreter, latex, fontsize, 15)
    legend('Slower','Faster', interpreter, latex, fontsize, 10)
    xlabel($M_x$, interpreter, latex, fontsize, 15)
    ylabel($M_y$, interpreter, latex, fontsize, 15)
    zlabel($M_z$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off
    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

D Problem 4 code:

D.1 Problem 4.1:

```matlab
clc
clear

%% Problem 4.1 FID sequence magnetisation vector visualisation
% Author: Imraj Singh 11/11/2020

% Given parameters

```

```
% Magnitisation aligned along z−axis initially
M = [0, 0, 1]';

% Number of sample in each part of sequence
N = 101;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% Flip time
INIT = 0;
tINIT = linspace(0, 0, N);

% Flip time
FT = .001;
tFT = linspace(0, FT, N);

% Relaxation time
RT = 8;
tRT = linspace(FT, RT + FT, N);

time = [tINIT, tFT, tRT];

% Indices
% Initial time
INITi = N;

% Flip time
FTi = INITi + N;

% Relaxation time
RTi = FTi + N;

Msoln = zeros(length(time),3);

% Initial time
Msoln(1:INITi,1) = M(1);
Msoln(1:INITi,2) = M(2);
Msoln(1:INITi,3) = M(3);

% Flip time
Vis90 = pi/2/FT;
M = Msoln(INITi,:);
Msoln(INITi:FTi,1) = M(1);
Msoln(INITi:FTi,2) = M(2)*cos(Vis90.*time(INITi:FTi)) + M(3)*sin(Vis90.*time(
    INITi:FTi));
Msoln(INITi:FTi,3) = M(3)*cos(Vis90.*time(INITi:FTi)) − M(2)*sin(Vis90.*time(
    INITi:FTi));

% Flip time 2
M = Msoln(FTi,:);
Msoln(FTi:RTi,1) = M(1).*exp(−time(FTi:RTi)./T2);
Msoln(FTi:RTi,2) = M(2).*exp(−time(FTi:RTi)./T2);
Msoln(FTi:RTi,3) = M(3)*exp(−time(FTi:RTi)./T1) + (1−exp(−time(FTi:RTi)./T1));

```

```
%% Animation module
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['4_1_rot', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = (N-1)/10;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,0,1,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-'
        ,'Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([0 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')
    hold on
    plot(Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver(0,0,0,1,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_{y'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([0 1]);
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

    % format title, legend, labels, limits, grid and box

    if i > 0 && i <= INITi
```

```
        title('Magnetisation vector along $B_0$ initially', interpreter, latex,
            fontsize, 15)
    xlim([0 1]);
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    elseif i > INITi && i <= FTi
        title(['RF pulse $\pi / 2$, time: ',num2str(round(1000*time(i), 3)),' ms
            '], interpreter, latex, fontsize, 15)
    xlim([0 FT]);
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    elseif i > FTi && i <= RTi
        title(['Time from $\pi / 2$ flip: ', num2str(round(time(i) - INIT - FT,
            3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)
    xlim([0 RT]);
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    end
    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend(M_{x'},M_{y'}, M_{z'},interpreter, latex, fontsize, 10)
    grid on
    box on
    ylim([0 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

D.2 Problem 4.2:

```
clc
clear

%% Problem 4.1 IR sequence magnetisation vector visualisation
% Author: Imraj Singh 11/11/2020

% Given parameters

% Magnitisation aligned along z-axis initially
M = [0, 0, 1]';

% Number of sample in each part of sequence
N = 101;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 2 * 2 * pi;

% Flip time
INIT = 0;
tINIT = linspace(0, 0, N);

% Flip time 1
```

```
FT_1 = .002;
tFT_1 = linspace(0, FT_1, N);

% Relaxation time 1
RT_1 = .5;
tRT_1 = linspace(FT_1, RT_1 + FT_1, N);

% Flip time 2
FT_2 = .001;
tFT_2 = linspace(RT_1 + FT_1, RT_1 + FT_1 + FT_2, N);

% Relaxation time 2
RT_2 = 8;
tRT_2 = linspace(RT_1 + FT_1 + FT_2, + RT_1 + FT_1 + FT_2 + RT_2, N);

time = [tINIT, tFT_1, tRT_1, tFT_2, tRT_2];

% Indices
% Initial time
INITi_1 = N;

% Flip time 1
FTi_1 = INITi_1 + N;

% Relaxation time 1
RTi_1 = FTi_1 + N;

% Flip time 2
FTi_2 = RTi_1 + N;

% Relaxation time 2
RTi_2 = FTi_2 + N;

Msoln = zeros(length(time),3);

% Initial time
Msoln(1:INITi_1,1) = M(1);
Msoln(1:INITi_1,2) = M(2);
Msoln(1:INITi_1,3) = M(3);

% Flip time 1
tad = time;
Vis90 = pi/FT_1;
M = Msoln(INITi_1,:);
Msoln(INITi_1:FTi_1,1) = M(1);
Msoln(INITi_1:FTi_1,2) = M(2)*cos(Vis90.*tad(INITi_1:FTi_1)) + M(3)*sin(Vis90.*
    tad(INITi_1:FTi_1));
Msoln(INITi_1:FTi_1,3) = M(3)*cos(Vis90.*tad(INITi_1:FTi_1)) - M(2)*sin(Vis90.*
    tad(INITi_1:FTi_1));


% Relaxation time 1
tad = tad - FT_1;
M = Msoln(FTi_1,:);
Msoln(FTi_1:RTi_1,1) = M(1).*exp(-tad(FTi_1:RTi_1)./T2);
Msoln(FTi_1:RTi_1,2) = M(2).*exp(-tad(FTi_1:RTi_1)./T2);
Msoln(FTi_1:RTi_1,3) = M(3)*exp(-tad(FTi_1:RTi_1)./T1) + (1-exp(-tad(FTi_1:RTi_1
    )./T1));

% Flip time 2
tad = tad - RT_1;
```

```
Vis90 = pi/2/FT_2;
M = Msoln(RTi_1,:);
Msoln(RTi_1:FTi_2,1) = M(1);
Msoln(RTi_1:FTi_2,2) = M(2)*cos(Vis90.*tad(RTi_1:FTi_2)) + M(3)*sin(Vis90.*tad(RTi_1:FTi_2));
Msoln(RTi_1:FTi_2,3) = M(3)*cos(Vis90.*tad(RTi_1:FTi_2)) - M(2)*sin(Vis90.*tad(RTi_1:FTi_2));

% Relaxation time 2
tad = tad - FT_2;
M = Msoln(FTi_2,:);
Msoln(FTi_2:RTi_2,1) = M(1).*exp(-tad(FTi_2:RTi_2)./T2);
Msoln(FTi_2:RTi_2,2) = M(2).*exp(-tad(FTi_2:RTi_2)./T2);
Msoln(FTi_2:RTi_2,3) = M(3)*exp(-tad(FTi_2:RTi_2)./T1) + (1-exp(-tad(FTi_2:RTi_2)./T1));


%% Animation module
% Author: Imraj Singh 03/11/2020

% initialise the video
video = VideoWriter(['4_2_rot_opt', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = (N-1)/10;
video.set('FrameRate', frameRate);

video.open();

% initialise the figure
h = figure;

% specify animation captures each degree of rotation
for i=1:length(time)
    subplot(2,2,1)
    quiver3(0,0,0,0,0,1,'linewidth',2,'LineStyle','--','Color','k')
    hold on
    plot3(Msoln(1:i,1),Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver3(0,0,0,Msoln(i,1),Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')

    % format legend, labels, limits, grid and box
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    view(3)
    hold off

    subplot(2,2,2)
    quiver(0,0,Msoln(i,2),Msoln(i,3),'linewidth',4,'LineStyle','-','Color','r')
    hold on
    plot(Msoln(1:i,2),Msoln(1:i,3),'k--','linewidth',2)
    quiver(0,0,0,1,'linewidth',2,'LineStyle','--','Color','k')

    % format legend, labels, limits, grid and box
    xlabel($M_{y'}$, interpreter, latex, fontsize, 15)
```

```
    ylabel(M_z', interpreter, latex, fontsize, 15)
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    hold off

    subplot(2,2,3:4)
    plot(time(1:i),Msoln(1:i,1),'linewidth',2,'LineStyle','-','Color','r');
    hold on
    plot(time(1:i),Msoln(1:i,2),'linewidth',2,'LineStyle','-.','Color','b');
    plot(time(1:i),Msoln(1:i,3),'linewidth',2,'LineStyle','--','Color','k');

    % format title, legend, labels, limits, grid and box

    if i > 0 && i <= INITi_1
        title('Magnetisation vector along $B_0$ initially', interpreter, latex, fontsize, 15)
        xlim([0 1]);
        xlabel(Time (s), interpreter, latex, fontsize, 15)
    elseif i > INITi_1 && i <= FTi_1
        xlim([0 FT_1]);
        title(['RF pulse $\pi $, time: ',num2str(round(1000*time(i), 3)),' ms'], interpreter, latex, fontsize, 15)
        xlabel(Time (s), interpreter, latex, fontsize, 15)
    elseif i > FTi_1 && i <= RTi_1
        title(['Time from $\pi$ flip: ', num2str(round(time(i) - INIT - FT_1, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)
        xlim([0 FT_1+RT_1+FT_2+RT_2]);
        xlabel(Time (s), interpreter, latex, fontsize, 15)
    elseif i > RTi_1 && i <= FTi_2
        title(['RF pulse $\pi / 2$, time: ',num2str(round(1000*(time(i) - FT_1 - RT_1), 3)),' ms'], interpreter, latex, fontsize, 15)
        xlim([0 FT_1+RT_1+FT_2+RT_2]);
        xlabel(Time (s), interpreter, latex, fontsize, 15)
    elseif i > FTi_2 && i <= RTi_2
        title(['Time from $\pi / 2$ flip: ', num2str(round(time(i) - INIT - FT_1 - RT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)
        xlim([0 FT_1+RT_1+FT_2+RT_2]);
        xlabel(Time (s), interpreter, latex, fontsize, 15)
    end

    ylabel(Magnetisation, interpreter, latex, fontsize, 15)
    legend(M_x',M_y', M_z',interpreter, latex, fontsize, 10)
    grid on
    box on
    ylim([-1 1]);
    hold off


    % assign frame and write it to the video
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

E Problem 5 code:

E.1 Problem 5.1:

```
clc
clear

%% Free procession with Dephasing
% Author: Imraj Singh 03/11/2020

% Given parameters

% Number of sample in each part of sequence
N = 101;

% Number of isochromats
Num = 1001;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 50 * pi;
deltaomega = 16;
omegaovector = linspace(-deltaomega,deltaomega,Num);

% Inital time 1
INIT_1 = 0;
tINIT_1 = linspace(0, INIT_1, N);

% Flip time 1
FT_1 = .001;
tFT_1 = linspace(0, FT_1, N);

% Dephasing time 1
DP_1 = .5;
tDP_1 = linspace(FT_1, DP_1 + FT_1, N);

% Flip time 2
FT_2 = .001;
tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);

% Rephasing time
RP = DP_1;
tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);

% Inital time 2
INIT_2 = 0;
tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP,  DP_1 + FT_1 + FT_2 + RP, N);

% Dephasing time 2
DP_2 = DP_1;
tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);

time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];

% Indices
% Initial time
INITi_1 = N;
```

```
% Flip time 1
FTi_1 = INITi_1 + N;

% Dephasing time 1
DPi_1 = FTi_1 + N;

% Flip time 2
FTi_2 = DPi_1 + N;

% Rephasing time
RPi = FTi_2 + N;

% Initial time
INITi_2 = RPi + N;

% Dephasing time 2
DPi_2 = INITi_2 + N;


Msoln = zeros(length(time),3, length(omegaovector));

for i = 1:length(omegaovector)
    % Initial time
    Msoln(1:INITi_1,1,i) = 0;
    Msoln(1:INITi_1,2,i) = 0;
    Msoln(1:INITi_1,3,i) = 1;

    % Flip time 1
    tad = time;
    Vis90 = pi/2/FT_1;
    M = Msoln(INITi_1,:,i);
    Msoln(INITi_1:FTi_1,1,i) = M(1);
    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));
    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));

    % Dephase time 1
    tad = tad - FT_1;
    M = Msoln(FTi_1,:,i);
    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)));
    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)));
    Msoln(FTi_1:DPi_1,3,i) = M(3);

    % Flip time 2
    tad = tad - DP_1;
    Vis90 = pi/FT_2;
    M = Msoln(DPi_1,:,i);
    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));
    Msoln(DPi_1:FTi_2,2,i) = M(2);
    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));

    % Rephase time
    tad = tad - FT_2;
    M = Msoln(FTi_2,:,i);
```

```
    Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(omegaovector(i).*tad(FTi_2:RPi));
    Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(omegaovector(i).*tad(FTi_2:RPi));
    Msoln(FTi_2:RPi,3,i) = M(3);

    % Echo time
    tad = tad - RP;
    M = Msoln(RPi,:,i);
    Msoln(RPi:INITi_2,1,i) = M(1);
    Msoln(RPi:INITi_2,2,i) = M(2);
    Msoln(RPi:INITi_2,3,i) = M(3);


    % Dephase time 2
    tad = tad - INIT_2;
    M = Msoln(INITi_2,:,i);
    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));
    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));
    Msoln(INITi_2:DPi_2,3,i) = M(3);
end
Mperp = zeros(length(time),1);
for i=1:length(time)
    Mperp(i) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/Num;
end

%% Animation module

video = VideoWriter(['5_1', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = N/8;
video.set('FrameRate', frameRate);

video.open();

h = figure;


for i=INITi_1:length(time)
    subplot(2,2,1)

    quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'LineStyle','-','Color','r')
    hold on
    for j=1:100:length(omegaovector)
        quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'LineStyle','-','Color','r')
    end
    grid on
    box on
    hold off
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)
```

```
    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
        LineStyle','-','Color','r')
    hold on
    for j=1:100:length(omegaovector)
        quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
            LineStyle','-','Color','r')
    end
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    hold off
    view(2)
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)


    subplot(2,2,3:4)

    plot(time(1:i),Mperp(1:i),'linewidth',2,'LineStyle','-','Color','k')
    hold on
    plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','--','Color','k')
    hold off
    grid on
    box on
    ylim([0 1]);
    xlim([0 max(time)]);
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Normalised signal, interpreter, latex, fontsize, 15)
    legend(|$\frac{1}{N}\sum M_\perp$|,$T_2$ envelope, interpreter, latex, fontsize, 10)

    % Initial
    if i > 0 && i <= INITi_1
        title('Magnetisation vector along $B_0$ initially', interpreter, latex,
            fontsize, 15)

        % First flip
    elseif i > INITi_1 && i <= FTi_1
        xlim([0 FT_1]);
        title('RF pulse $\pi /2$ ', interpreter, latex, fontsize, 15)

        % First Dephase
    elseif i > FTi_1 && i <= DPi_1
        title(['Isocromats dephasing, t = ', num2str(round(time(i) - INIT_1 -
            FT_1, 3)),' s'], interpreter, latex, fontsize, 15)

        % Second flip
    elseif i > DPi_1 && i <= FTi_2
        title(['RF pulse $\pi$, $t = \tau$ = ', num2str(round(DP_1, 3)),' s'],
            interpreter, latex, fontsize, 15)

        % Rephasing
    elseif i > FTi_2 && i <= RPi
        title(['Isocromats rephasing, $t =$ ', num2str(round(time(i) - INIT_1 -
            FT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)

        % Echo
```

```
    elseif i > RPi && i <= INITi_2
        title(['Echo time, $t = T_E = 2 \tau =$ ',num2str(round(DP_1 * 2, 3)),'
            s'], interpreter, latex, fontsize, 15)

        % Dephasing
    elseif i > INITi_2 && i <= DPi_2
        title(['Isocromats dephasing, $t =$ ', num2str(round(time(i) - INIT_1 -
            FT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)

    end

    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

E.2 Problem 5.2:

```
clc
clear

%% Free procession with Dephasing
% Author: Imraj Singh 03/11/2020

% Given parameters

% Number of sample in each part of sequence
N = 11;

% Number of isochromats
Num = 1001;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 50 * pi;
deltaomega = 0;
delta = 8;
omegaovector = deltaomega+delta*tan(pi*(rand(Num,1)-1/2));

% Inital time 1
INIT_1 = 0;
tINIT_1 = linspace(0, INIT_1, N);

% Flip time 1
FT_1 = .001;
tFT_1 = linspace(0, FT_1, N);

% Dephasing time 1
DP_1 = .5;
tDP_1 = linspace(FT_1, DP_1 + FT_1, N);

% Flip time 2
FT_2 = .001;
tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);
```

```

% Rephasing time
RP = DP_1;
tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);

% Inital time 2
INIT_2 = 0;
tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP,  DP_1 + FT_1 + FT_2 + RP, N);

% Dephasing time 2
DP_2 = DP_1;
tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);

time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];

% Indices
% Initial time
INITi_1 = N;

% Flip time 1
FTi_1 = INITi_1 + N;

% Dephasing time 1
DPi_1 = FTi_1 + N;

% Flip time 2
FTi_2 = DPi_1 + N;

% Rephasing time
RPi = FTi_2 + N;

% Initial time
INITi_2 = RPi + N;

% Dephasing time 2
DPi_2 = INITi_2 + N;


Msoln = zeros(length(time),3, length(omegaovector));

for i = 1:length(omegaovector)
    % Initial time
    Msoln(1:INITi_1,1,i) = 0;
    Msoln(1:INITi_1,2,i) = 0;
    Msoln(1:INITi_1,3,i) = 1;

    % Flip time 1
    tad = time;
    Vis90 = pi/2/FT_1;
    M = Msoln(INITi_1,:,i);
    Msoln(INITi_1:FTi_1,1,i) = M(1);
    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));
    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));

    % Dephase time 1
    tad = tad - FT_1;
    M = Msoln(FTi_1,:,i);
    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)));
```

```
    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)));
    Msoln(FTi_1:DPi_1,3,i) = M(3);

    % Flip time 2
    tad = tad - DP_1;
    Vis90 = pi/FT_2;
    M = Msoln(DPi_1,:,i);
    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));
    Msoln(DPi_1:FTi_2,2,i) = M(2);
    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));

    % Rephase time
    tad = tad - FT_2;
    M = Msoln(FTi_2,:,i);
    Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
    Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
    Msoln(FTi_2:RPi,3,i) = M(3);

    % Echo time
    tad = tad - RP;
    M = Msoln(RPi,:,i);
    Msoln(RPi:INITi_2,1,i) = M(1);
    Msoln(RPi:INITi_2,2,i) = M(2);
    Msoln(RPi:INITi_2,3,i) = M(3);


    % Dephase time 2
    tad = tad - INIT_2;
    M = Msoln(INITi_2,:,i);
    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));
    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));
    Msoln(INITi_2:DPi_2,3,i) = M(3);
end
Mperp = zeros(length(time),1);
for i=1:length(time)
    Mperp(i) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/
        Num;
end

%% Animation module

video = VideoWriter(['5_2', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = N/8;
video.set('FrameRate', frameRate);

video.open();

h = figure;


for i=INITi_1:length(time)
    subplot(2,2,1)
```

```

    quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
        LineStyle','-','Color','r')
    hold on
    for j=1:100:length(omegaovector)
        quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
            LineStyle','-','Color','r')
    end
    grid on
    box on
    hold off
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'
        LineStyle','-','Color','r')
    hold on
    for j=1:100:length(omegaovector)
        quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'
            LineStyle','-','Color','r')
    end
    grid on
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    hold off
    view(2)
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)


    subplot(2,2,3:4)

    plot(time(1:i),Mperp(1:i),'linewidth',2,'LineStyle','-','Color','k')
    hold on
    plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','--','Color','k')
    hold off
    grid on
    box on
    ylim([0 1]);
    xlim([0 max(time)]);
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Normalised signal, interpreter, latex, fontsize, 15)
    legend(|$\frac{1}{N}\sum M_\perp$|,$T_2$ envelope, interpreter, latex, fontsize, 10)

    % Initial
    if i > 0 && i <= INITi_1
        title('Magnetisation vector along $B_0$ initially', interpreter, latex,
            fontsize, 15)

        % First flip
    elseif i > INITi_1 && i <= FTi_1
        xlim([0 FT_1]);
        title('RF pulse $\pi /2$ ', interpreter, latex, fontsize, 15)
```

```

        % First Dephase
    elseif i > FTi_1 && i <= DPi_1
        title(['Isocromats dephasing, t = ', num2str(round(time(i) - INIT_1 -
            FT_1, 3)),' s'], interpreter, latex, fontsize, 15)

        % Second flip
    elseif i > DPi_1 && i <= FTi_2
        title(['RF pulse $\pi$, $t = \tau$ = ', num2str(round(DP_1, 3)),' s'],
            interpreter, latex, fontsize, 15)

        % Rephasing
    elseif i > FTi_2 && i <= RPi
        title(['Isocromats rephasing, $t =$ ', num2str(round(time(i) - INIT_1 -
            FT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)

        % Echo
    elseif i > RPi && i <= INITi_2
        title(['Echo time, $t = T_E = 2 \tau =$ ',num2str(round(DP_1 * 2, 3)),'
            s'], interpreter, latex, fontsize, 15)

        % Dephasing
    elseif i > INITi_2 && i <= DPi_2
        title(['Isocromats dephasing, $t =$ ', num2str(round(time(i) - INIT_1 -
            FT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)

    end

    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

E.3 Problem 5.3:

```
% house keeping
clc
clear
%% Sensitivity of Uniform Distribution to delta omega
% Author: Imraj Singh


% Given parameters

% Number of sample in each part of sequence
N = 101;

% Number of isochromats
Num = 10001;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Inital time 1
INIT_1 = 0;
tINIT_1 = linspace(0, INIT_1, N);
```

```

% Flip time 1
FT_1 = .001;
tFT_1 = linspace(0, FT_1, N);

% Dephasing time 1
DP_1 = .5;
tDP_1 = linspace(FT_1, DP_1 + FT_1, N);

% Flip time 2
FT_2 = .001;
tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);

% Rephasing time
RP = DP_1;
tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);

% Inital time 2
INIT_2 = 0;
tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP,  DP_1 + FT_1 + FT_2 + RP, N);

% Dephasing time 2
DP_2 = DP_1;
tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);

time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];

% Indices
% Initial time
INITi_1 = N;

% Flip time 1
FTi_1 = INITi_1 + N;

% Dephasing time 1
DPi_1 = FTi_1 + N;

% Flip time 2
FTi_2 = DPi_1 + N;

% Rephasing time
RPi = FTi_2 + N;

% Initial time
INITi_2 = RPi + N;

% Dephasing time 2
DPi_2 = INITi_2 + N;

%% Sensitivity of Uniform Distribution to delta omega
% Author: Imraj Singh

% Precession frequency
omega0 = 50 * pi;
delta = [1 2 4 8 16 32 64 128];

Mperp = zeros(length(time),length(delta));
for Di = 1:length(delta)
    omegaovector = linspace(-delta(Di),delta(Di),Num);
    Msoln = zeros(length(time),3, length(omegaovector));

```

```matlab
    for i = 1:length(omegaovector)
        % Initial time
        Msoln(1:INITi_1,1,i) = 0;
        Msoln(1:INITi_1,2,i) = 0;
        Msoln(1:INITi_1,3,i) = 1;

        % Flip time 1
        tad = time;
        Vis90 = pi/2/FT_1;
        M = Msoln(INITi_1,:,i);
        Msoln(INITi_1:FTi_1,1,i) = M(1);
        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));
        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));

        % Dephase time 1
        tad = tad - FT_1;
        M = Msoln(FTi_1,:,i);
        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)));
        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)));
        Msoln(FTi_1:DPi_1,3,i) = M(3);

        % Flip time 2
        tad = tad - DP_1;
        Vis90 = pi/FT_2;
        M = Msoln(DPi_1,:,i);
        Msoln(DPi_1:FTi_2,1,i) = M(1);
        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));
        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));

        % Rephase time
        tad = tad - FT_2;
        M = Msoln(FTi_2,:,i);
        Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,3,i) = M(3);

        % Echo time
        tad = tad - RP;
        M = Msoln(RPi,:,i);
        Msoln(RPi:INITi_2,1,i) = M(1);
        Msoln(RPi:INITi_2,2,i) = M(2);
        Msoln(RPi:INITi_2,3,i) = M(3);


        % Dephase time 2
        tad = tad - INIT_2;
        M = Msoln(INITi_2,:,i);
        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));
        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));
        Msoln(INITi_2:DPi_2,3,i) = M(3);
    end
```

```
    for i=1:length(time)
        Mperp(i,Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/Num;
    end
end

video = VideoWriter(['5_3_dOmega', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = .25;
video.set('FrameRate', frameRate);

video.open();

h = figure;
for i=1:length(delta)
    plot(time(:),Mperp(:,i),'linewidth',2,'LineStyle','--','Color','k')

    grid on
    box on
    xlim([0 max(time)]);
    ylim([0 1]);
    view(2)
    title(['$\delta \omega $ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(|\frac{1}{N}\sum M_\perp|, interpreter, latex, fontsize, 15)

    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();

%% Sensitivity of Cauchy to delta
% Author: Imraj Singh

% Precession frequency
omega0 = 50 * pi;
delta = [1 2 4 8 16 32 64 128];
deltaomega = 0;
Mperp = zeros(length(time),length(delta));
for Di = 1:length(delta)
    omegaovector = deltaomega+delta(Di)*tan(pi*(rand(Num,1)-1/2));
    Msoln = zeros(length(time),3, length(omegaovector));

    for i = 1:length(omegaovector)
        % Initial time
        Msoln(1:INITi_1,1,i) = 0;
        Msoln(1:INITi_1,2,i) = 0;
        Msoln(1:INITi_1,3,i) = 1;

        % Flip time 1
        tad = time;
        Vis90 = pi/2/FT_1;
        M = Msoln(INITi_1,:,i);
        Msoln(INITi_1:FTi_1,1,i) = M(1);
        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));
        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));

```

```
        % Dephase time 1
        tad = tad - FT_1;
        M = Msoln(FTi_1,:,i);
        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)));
        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)));
        Msoln(FTi_1:DPi_1,3,i) = M(3);

        % Flip time 2
        tad = tad - DP_1;
        Vis90 = pi/FT_2;
        M = Msoln(DPi_1,:,i);
        Msoln(DPi_1:FTi_2,1,i) = M(1);
        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));
        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));

        % Rephase time
        tad = tad - FT_2;
        M = Msoln(FTi_2,:,i);
        Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,3,i) = M(3);

        % Echo time
        tad = tad - RP;
        M = Msoln(RPi,:,i);
        Msoln(RPi:INITi_2,1,i) = M(1);
        Msoln(RPi:INITi_2,2,i) = M(2);
        Msoln(RPi:INITi_2,3,i) = M(3);


        % Dephase time 2
        tad = tad - INIT_2;
        M = Msoln(INITi_2,:,i);
        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));
        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));
        Msoln(INITi_2:DPi_2,3,i) = M(3);
    end
    for i=1:length(time)
        Mperp(i,Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/Num;
    end
end

h = figure;
for i=1:length(delta)
    plot(time(:),Mperp(:,i),'linewidth',2,'LineStyle','-','Color','k')

    grid on
    box on
    xlim([0 max(time)]);
    ylim([0 1]);
    view(2)
    title(['$\Delta $ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
```

```
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(|$\frac{1}{N}\sum M_{\perp}$|, interpreter, latex, fontsize, 15)

    frame(i) = getframe(h);
end
video = VideoWriter(['5_3_Delta', '.mp4'], 'MPEG-4');
frameRate = .25;
video.set('FrameRate', frameRate);
video.open();
video.writeVideo(frame);
video.close();

%% Sensitivity of Cauchy to omega
% Author: Imraj Singh

% Precession frequency
omega0 = 50 * pi;
delta1 = 4;
deltaomega = [-100 -50 -25 -10 -5 0 5 10 25 50 100];
Mperp = zeros(length(time),length(delta));
for Di = 1:length(delta)
    omegaovector = delta(Di)+delta1*tan(pi*(rand(Num,1)-1/2));
    Msoln = zeros(length(time),3, length(omegaovector));

    for i = 1:length(omegaovector)
        % Initial time
        Msoln(1:INITi_1,1,i) = 0;
        Msoln(1:INITi_1,2,i) = 0;
        Msoln(1:INITi_1,3,i) = 1;

        % Flip time 1
        tad = time;
        Vis90 = pi/2/FT_1;
        M = Msoln(INITi_1,:,i);
        Msoln(INITi_1:FTi_1,1,i) = M(1);
        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));
        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));

        % Dephase time 1
        tad = tad - FT_1;
        M = Msoln(FTi_1,:,i);
        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)));
        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)));
        Msoln(FTi_1:DPi_1,3,i) = M(3);

        % Flip time 2
        tad = tad - DP_1;
        Vis90 = pi/FT_2;
        M = Msoln(DPi_1,:,i);
        Msoln(DPi_1:FTi_2,1,i) = M(1);
        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));
        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));

        % Rephase time
        tad = tad - FT_2;
```

```
        M = Msoln(FTi_2,:,i);
        Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*
            sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*
            sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,3,i) = M(3);

        % Echo time
        tad = tad - RP;
        M = Msoln(RPi,:,i);
        Msoln(RPi:INITi_2,1,i) = M(1);
        Msoln(RPi:INITi_2,2,i) = M(2);
        Msoln(RPi:INITi_2,3,i) = M(3);


        % Dephase time 2
        tad = tad - INIT_2;
        M = Msoln(INITi_2,:,i);
        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));
        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));
        Msoln(INITi_2:DPi_2,3,i) = M(3);
    end
    for i=1:length(time)
        Mperp(i,Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i
            ,2,:))^2)/Num;
    end
end


% set the frame rate

h = figure;
for i=1:length(delta)
    plot(time(:),Mperp(:,i),'linewidth',2,'LineStyle','-','Color','k')

    grid on
    box on
    xlim([0 max(time)]);
    ylim([0 1]);
    view(2)
    title(['$\omega_0 $ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(|$\frac{1}{N}\sum M_\perp$|, interpreter, latex, fontsize, 15)
    frame(i) = getframe(gcf);
end
video = VideoWriter(['5_3_omega', '.mp4'], 'MPEG-4');
frameRate = .25;
video.set('FrameRate', frameRate);
video.open();
video.writeVideo(frame);
video.close();
```

E.4 Problem 5.3 components plot:

```
clc
clear

%% Sensitivity of Cauchy to omega_0
```

```
% Author: Imraj Singh

% Given parameters

% Number of sample in each part of sequence
N = 101;

% Number of isochromats
Num = 10001;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Inital time 1
INIT_1 = 0;
tINIT_1 = linspace(0, INIT_1, N);

% Flip time 1
FT_1 = .001;
tFT_1 = linspace(0, FT_1, N);

% Dephasing time 1
DP_1 = .5;
tDP_1 = linspace(FT_1, DP_1 + FT_1, N);

% Flip time 2
FT_2 = .001;
tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);

% Rephasing time
RP = DP_1;
tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);

% Inital time 2
INIT_2 = 0;
tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP,  DP_1 + FT_1 + FT_2 + RP, N);

% Dephasing time 2
DP_2 = DP_1;
tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);

time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];

% Indices
% Initial time
INITi_1 = N;

% Flip time 1
FTi_1 = INITi_1 + N;

% Dephasing time 1
DPi_1 = FTi_1 + N;

% Flip time 2
FTi_2 = DPi_1 + N;

% Rephasing time
RPi = FTi_2 + N;
```

```matlab

% Initial time
INITi_2 = RPi + N;

% Dephasing time 2
DPi_2 = INITi_2 + N;


% Precession frequency
omega0 = 50 * pi;
delta1 = 4;
delta = [0 5 10 25 50 100 200];
Mperp = zeros(length(time),length(delta));
for Di = 1:length(delta)
    omegaovector = delta(Di)+delta1*tan(pi*(rand(Num,1)-1/2));
    Msoln = zeros(length(time),3, length(omegaovector));

    for i = 1:length(omegaovector)
        % Initial time
        Msoln(1:INITi_1,1,i) = 0;
        Msoln(1:INITi_1,2,i) = 0;
        Msoln(1:INITi_1,3,i) = 1;

        % Flip time 1
        tad = time;
        Vis90 = pi/2/FT_1;
        M = Msoln(INITi_1,:,i);
        Msoln(INITi_1:FTi_1,1,i) = M(1);
        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));
        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));

        % Dephase time 1
        tad = tad - FT_1;
        M = Msoln(FTi_1,:,i);
        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)));
        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)));
        Msoln(FTi_1:DPi_1,3,i) = M(3);

        % Flip time 2
        tad = tad - DP_1;
        Vis90 = pi/FT_2;
        M = Msoln(DPi_1,:,i);
        Msoln(DPi_1:FTi_2,1,i) = M(1);
        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));
        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));

        % Rephase time
        tad = tad - FT_2;
        M = Msoln(FTi_2,:,i);
        Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(omegaovector(i).*tad(FTi_2:RPi));
        Msoln(FTi_2:RPi,3,i) = M(3);

```

```
        % Echo time
        tad = tad - RP;
        M = Msoln(RPi,:,i);
        Msoln(RPi:INITi_2,1,i) = M(1);
        Msoln(RPi:INITi_2,2,i) = M(2);
        Msoln(RPi:INITi_2,3,i) = M(3);


        % Dephase time 2
        tad = tad - INIT_2;
        M = Msoln(INITi_2,:,i);
        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));
        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));
        Msoln(INITi_2:DPi_2,3,i) = M(3);
    end
    for i=1:length(time)
        Mperp(i,1,Di) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i
            ,2,:))^2)/Num;
        Mperp(i,2,Di) = exp(-time(i)/T2)*sum(Msoln(i,1,:))/Num;
        Mperp(i,3,Di) = exp(-time(i)/T2)*sum(Msoln(i,2,:))/Num;
    end
end

h = figure;
for i=1:length(delta)
    plot(time(:),Mperp(:,2,i),'linewidth',2,'LineStyle','-','Color','r')
    hold on
    plot(time(:),Mperp(:,3,i),'linewidth',2,'LineStyle','-','Color','b')
    plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','--','Color','k')
    plot(time(:),-exp(-time(:)/T2),'linewidth',2,'LineStyle','--','Color','k')
    hold off
    grid on
    box on
    xlim([0 max(time)]);
    ylim([-1 1]);
    title(['$\Delta $ = ', num2str(delta(i))], interpreter, latex, fontsize, 15)
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Normalised average $M_\perp$, interpreter, latex, fontsize, 15)
    legend($\frac{1}{N}\sum M_{x'}$,$\frac{1}{N}\sum M_{y'}$,$T_2$ envelope, interpreter, latex, fontsize, 10)

    frame(i) = getframe(h);
end
video = VideoWriter(['5_3_omega_comp', '.mp4'], 'MPEG-4');
frameRate = .25;
video.set('FrameRate', frameRate);
video.open();
video.writeVideo(frame);
video.close();
```

E.5 Problem 5.4:

```
clc
clear

%% Hahn echo
% Author: Imraj Singh 03/11/2020

% Given parameters
```

```

% Number of sample in each part of sequence
N = 11;

% Number of isochromats
Num = 1001;

% Prescribe T1
T2 = 1;
T1 = T2 * 2;

% Calculate neccessary parameters

% Precession frequency
omega0 = 50 * pi;
deltaomega = 0;
delta = 2;
omegaovector = deltaomega+delta*tan(pi*(rand(Num,1)-1/2));

% Inital time 1
INIT_1 = 0;
tINIT_1 = linspace(0, INIT_1, N);

% Flip time 1
FT_1 = .001;
tFT_1 = linspace(0, FT_1, N);

% Dephasing time 1
DP_1 = .5;
tDP_1 = linspace(FT_1, DP_1 + FT_1, N);

% Flip time 2
FT_2 = .002;
tFT_2 = linspace(DP_1 + FT_1, DP_1 + FT_1 + FT_2, N);

% Rephasing time
RP = DP_1;
tRP = linspace(DP_1 + FT_1 + FT_2, DP_1 + FT_1 + FT_2 + RP, N);

% Inital time 2
INIT_2 = 0;
tINIT_2 = linspace(DP_1 + FT_1 + FT_2 + RP,  DP_1 + FT_1 + FT_2 + RP, N);

% Dephasing time 2
DP_2 = DP_1;
tDP_2 = linspace(DP_1 + FT_1 + FT_2 + RP, + DP_1 + FT_1 + FT_2 + RP + DP_2, N);

time = [tINIT_1, tFT_1, tDP_1, tFT_2, tRP, tINIT_2, tDP_2];

% Indices
% Initial time
INITi_1 = N;

% Flip time 1
FTi_1 = INITi_1 + N;

% Dephasing time 1
DPi_1 = FTi_1 + N;

% Flip time 2
FTi_2 = DPi_1 + N;
```

```

% Rephasing time
RPi = FTi_2 + N;

% Initial time
INITi_2 = RPi + N;

% Dephasing time 2
DPi_2 = INITi_2 + N;


Msoln = zeros(length(time),3, length(omegaovector));

for i = 1:length(omegaovector)
    % Initial time
    Msoln(1:INITi_1,1,i) = 0;
    Msoln(1:INITi_1,2,i) = 0;
    Msoln(1:INITi_1,3,i) = 1;

    % Flip time 1
    tad = time;
    Vis90 = pi/2/FT_1;
    M = Msoln(INITi_1,:,i);
    Msoln(INITi_1:FTi_1,1,i) = M(1);
    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));
    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));

    % Dephase time 1
    tad = tad - FT_1;
    M = Msoln(FTi_1,:,i);
    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)));
    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)));
    Msoln(FTi_1:DPi_1,3,i) = M(3);

    % Flip time 2
    tad = tad - DP_1;
    Vis90 = pi/2/FT_2;
    M = Msoln(DPi_1,:,i);
    Msoln(DPi_1:FTi_2,1,i) = M(1);
    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));
    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));

    % Rephase time
    tad = tad - FT_2;
    M = Msoln(FTi_2,:,i);
    Msoln(FTi_2:RPi,1,i) = M(1)*cos(omegaovector(i).*tad(FTi_2:RPi)) + M(2)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
    Msoln(FTi_2:RPi,2,i) = M(2)*cos(omegaovector(i).*tad(FTi_2:RPi)) - M(1)*sin(
        omegaovector(i).*tad(FTi_2:RPi));
    Msoln(FTi_2:RPi,3,i) = M(3);

    % Echo time
    tad = tad - RP;
    M = Msoln(RPi,:,i);
    Msoln(RPi:INITi_2,1,i) = M(1);
```

```
    Msoln(RPi:INITi_2,2,i) = M(2);
    Msoln(RPi:INITi_2,3,i) = M(3);


    % Dephase time 2
    tad = tad - INIT_2;
    M = Msoln(INITi_2,:,i);
    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));
    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));
    Msoln(INITi_2:DPi_2,3,i) = M(3);
end
Mperp = zeros(length(time),1);
for i=1:length(time)
    Mperp(i) = exp(-time(i)/T2)*sqrt(sum(Msoln(i,1,:))^2 + sum(Msoln(i,2,:))^2)/Num;
end

%% Animation module

video = VideoWriter(['5_4', '.mp4'], 'MPEG-4');

% set the frame rate
frameRate = N/8;
video.set('FrameRate', frameRate);

video.open();

h = figure;


for i=INITi_1:length(time)
    subplot(2,2,1)

    quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'LineStyle','-','Color','r')
    hold on
    for j=1:100:length(omegaovector)
        quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'LineStyle','-','Color','r')
    end
    grid on
    box on
    hold off
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)

    subplot(2,2,2)
    quiver3(0,0,0,Msoln(i,1,1),Msoln(i,2,1),Msoln(i,3,1),'linewidth',5,'LineStyle','-','Color','r')
    hold on
    for j=1:100:length(omegaovector)
        quiver3(0,0,0,Msoln(i,1,j),Msoln(i,2,j),Msoln(i,3,j),'linewidth',5,'LineStyle','-','Color','r')
    end
    grid on
```

```
    box on
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
    hold off
    view(2)
    xlabel($M_{x'}$, interpreter, latex, fontsize, 15)
    ylabel($M_{y'}$, interpreter, latex, fontsize, 15)
    zlabel($M_{z'}$, interpreter, latex, fontsize, 15)


    subplot(2,2,3:4)

    plot(time(1:i),Mperp(1:i),'linewidth',2,'LineStyle','-','Color','k')
    hold on
    plot(time(:),exp(-time(:)/T2),'linewidth',2,'LineStyle','--','Color','k')
    hold off
    grid on
    box on
    ylim([0 1]);
    xlim([0 max(time)]);
    xlabel(Time (s), interpreter, latex, fontsize, 15)
    ylabel(Normalised signal, interpreter, latex, fontsize, 15)
    legend($|\frac{1}{N}\sum M_\perp|$,$T_2$ envelope, interpreter, latex, fontsize, 10)

    % Initial
    if i > 0 && i <= INITi_1
        title('Magnetisation vector along $B_0$ initially', interpreter, latex,
            fontsize, 15)

        % First flip
    elseif i > INITi_1 && i <= FTi_1
        xlim([0 FT_1]);
        title('RF pulse $\pi /2$ ', interpreter, latex, fontsize, 15)

        % First Dephase
    elseif i > FTi_1 && i <= DPi_1
        title(['Isocromats dephasing, t = ', num2str(round(time(i) - INIT_1 -
            FT_1, 3)),' s'], interpreter, latex, fontsize, 15)

        % Second flip
    elseif i > DPi_1 && i <= FTi_2
        title(['RF pulse $\pi$, $t = \tau$ = ', num2str(round(DP_1, 3)),' s'],
            interpreter, latex, fontsize, 15)

        % Rephasing
    elseif i > FTi_2 && i <= RPi
        title(['Isocromats rephasing, $t =$ ', num2str(round(time(i) - INIT_1 -
            FT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)

        % Echo
    elseif i > RPi && i <= INITi_2
        title(['Echo time, $t = T_E = 2 \tau =$ ',num2str(round(DP_1 * 2, 3)),'
            s'], interpreter, latex, fontsize, 15)

        % Dephasing
    elseif i > INITi_2 && i <= DPi_2
        title(['Isocromats dephasing, $t =$ ', num2str(round(time(i) - INIT_1 -
            FT_1 - FT_2, 3)),' s (Relaxation)'], interpreter, latex, fontsize, 15)

    end
```

```
    frame = getframe(h);
    video.writeVideo(frame);
end

video.close();
```

Code for all the plots