Coursework 2

University College London

Medical Physics & Biomedical Engineering department

MRes - Coursework 2

COMP0121 - Computational MRI

AUTUMN Term

14th December 2020

Author:
Mr. Imraj Singh (SN: 20164771)

Module lead:
Dr. Gary Zhang
Graduate Teaching Assistant:
Mr. Sean Epstein

This report along with the zip file containing animations make up the coursework 2 submission.

Contents

All problems are simulated in the rotating frame of reference (FoR) with a Cartesian coordinate system x', y' and y', with the underlying coordinates where isochromats are placed being x, y and y. The spins are at thermal equilibrium with a common magnetisation $M_0 = [0, 0, 1]'$ at the beginning of each experiment, and all experiments start with a 90° hard radio frequency (RF) excitation pulse along the x' axis over 0.32 ms.

Comments on the signal obtain: To calculate the signal a plethora of parameters are needed, with the main underlying physical principle being that the signal precipitates directly from an integral of the transverse magnetisation of all isochromats within a sample. In this report the normalised sum of transverse magnetisation of all isochromats of interest is used as a synonym of signal. This normalised sum of transverse magnetisations is measured in the rotating FoR, thus the components are denoted $M_{x'}$ and $M_{y'}$.

1 Problem 1: Frequency Encoding

A 1-D length uniform spin density was immersed in a homogeneous static magnetic field B_0, which is aligned parallel to the z axis.

Comments on isochromat spatial frequency, k_{iso}: As there are initially 20 isochromats ($N_{iso} = 20$) along a length of 20mm (L), we can say that there is 1 isochromat every mm. In-essence the spatial frequency of the isochromats, we denote as k_{iso} is 1 1/mm, where $k_{iso} = \frac{N_{iso}}{L} = 1$ 1/mm. Applying a constant gradient in the x direction to the sample, G_x, will cause the isochromats to dephase. This dephasing will not be random, but due to the isochromat make-up. In this experiment the segment of isochromats has a uniform distribution with a given k_{iso}. The gradient acts to select specific spatial frequencies (k_x) of features within the imaging domain and obtain their signal, the longer the gradient is on the wider range of spatial frequencies sampled, i.e. more k-space traversed. Initially, the features within the domain of this experiment have $k_{iso} = 1$ 1/mm, therefore a $k_x = 1$ recovers the full signal (with some loss due to relaxation), and so does $k_x = 2, 3, 4, 5....$ Thus we can say that the total signal is recoverable given our prescribed isochromat make-up at $k_x = n \cdot k_{iso}$. where n is a natural number.

Comments on isochromat spatial phase, Φ_{iso}: The isochromat spatial phase will give an indication of when $k_x = n \cdot k_{iso}$, what the corresponding signal will be. This is seems to be quite complicated and outside of the learning thus far, but from experimentation two cases were explored. The case when there is a isochromat at $x = 0$ and the case when isochromats are at $x = \pm\frac{1}{2k_{iso}} = \pm\frac{\Delta x}{2}$, which we denote as in-phase and out-phase respectively. More formally:

$$\Phi_{iso} = \begin{cases} \text{in-phase, when there are isochromats at } x = 0 \\ \text{out-phase, when there are isochromats at } x = \pm\frac{\Delta x}{2} = \pm\frac{1}{2k_{iso}} \end{cases}$$

At $k_x = n \cdot k_{iso}$ it was found that when in-phase positive values of $M_{y'}$ were always recovered, and when out-phase negative values of $M_{y'}$ were recovered if n was odd, and positive values when n was positive. Please see animations 'In_phase' and 'Out_phase' where the $k_{iso} = \frac{1}{3}$, and the former is in-phase and latter is out-phase.

Comments on the objective of the simulation: The simulation represents unit impulses (or dirac delta functions) of spin density at given k_{iso} and Φ_{iso}. This is because the unit impulses are representing spin density and in this simulation the spin density can be represented as a single isochromat, where the magnitude of the isochromats are irrelevant as a uniform spin density is prescribed so that all isochromats must have the same magnitude of magnetisation.

The spatial discretisation spacing of the simulation is equivalent to the spatial frequency of $\Delta x = 1/k_{iso}$. This means that although unit impulses of spin density are prescribed, the unit impulses are actually representing piece-wise constant approximations of the spin density. Where the spin density is uniform. We are not wanting to simulate unit impulses of spin density, but a uniform distribution of

density which is discretised at spatial frequency k_{iso}. The figure 1 shows that numerical discretisation actually represents an area along the x-axis by a single isochromat.

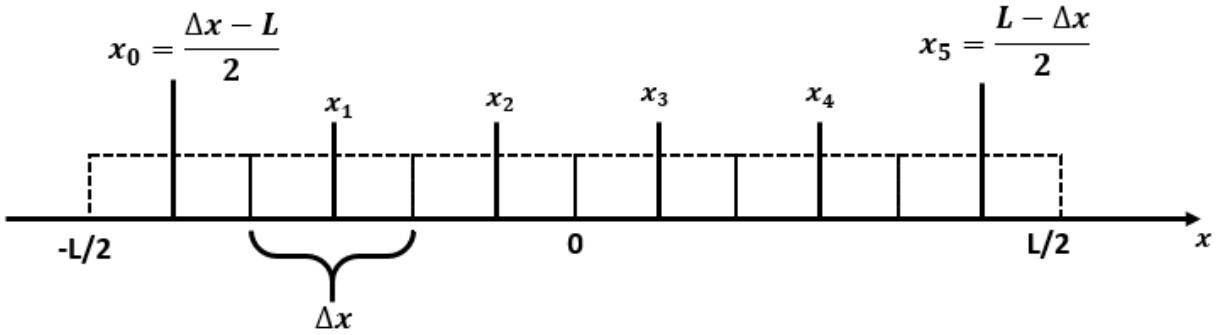

Figure 1: 1D discretisation of a segment of uniform spin density. Where x_0 to x_5 represent where the individual isochromats are placed. L is the length of the segment and Δx is the spatial spacing. It should be noted that the initial point x_0 and final point x_N must be within the length of the segment. If $x_0 = -L/2$ and/or $x_0 = L/2$ then an extended segment is actually being simulated.

1.1 Task 1

For this task the length of segment L is set to 20mm. There are spin isochromats set along the segment with a spatial spacing $\Delta x = 1$ mm, therefore the spatial frequency of the isochromats k_{iso} is 1 1/mm. Furthermore as is illustrated in figure 1, the discretisation does not prescribe an isochromat at the spatial location $x = 0$, but instead at $x = \pm\frac{\Delta}{2}$. Therefore it can be said that the Φ_{iso} is out-phase, and has implications as previously mentioned. The values $T_s = 5.12$ ms, G_x magnitude is 4.6 and N are all as specified in the hand out. Additionally it was decided that it was necessary to sample at $k_x = 0$. This has implications for experiment 2 and 3.

Experiment 1 is the most basic experiment and there are no frequency encoding gradients, G_x. The experiment consists of the RF pulse and free precession back to M_0. With this experiment it was important to visualise the time-scales that relaxation occurs on. The values of T_1 and T_2 for brain tissues were roughly approximated from[1] to be 1 s and 0.1 s respectively. From animation 'P1_E1' we can see that over the time signal is recorded T_s the relaxation is negligible as the time scales of recording is much smaller than T_1 and T_2. It can also be seen that the timescale of T_1 is a lot larger than T_2, therefore $T_s << T_1 << T_2$.

The effect of relaxation is negligible for recording when the time between the RF pulse and end of ADC (analogue-to-digital converter) recording is $\approx \mathcal{O}(ms)$. Thus the magnitude of the transverse magnetisation ($M_\perp$) is approximately constant over the ADC recording. Knowing that, it can be said that the import aspect of transverse magnetisation to visualise is the direction. As this is what determines the signal. The $M_\perp$ direction can be most easily visualised with a scalar value, the angle α. This is found by converting $M_\perp$ into polar coordinates and discarding the magnitude as it is approximately constant, such that $\alpha = \text{atan2}\left(\frac{M_{y'}}{M_{x'}}\right)$, with $\alpha = [0, 360)$. This alpha is then used to specify a circular colour map. Figure 2 is the vector $M_\perp$ with unit magnitude pointed in different directions (corresponding to different α) to show how the colour varies with direction.

Experiment 2 includes a gradient $G_x = -4.6 \times 10^{-3}$ T/m that de-phases the isochromats along the x axis. The prescribed k_{iso} is 1 1/mm, we can work out the maximum $k_{x,max}$ value from $k_{x,max} = \frac{\gamma}{2\pi} G_x T_s = 1.0046$ 1/mm. We take the real values of gyromagnetic ratio of a hydrogen proton as $\gamma = 2.68 \times 10^8$ rad/s/T, and gradient and sample time are as previously stated. We can predict that at the end of the sampling time T_s the total signal will be recovered. As the Φ_{iso} is out-phase when $k_{iso} = k_x$ the $M_{y'}$ is negative. This can be seen in animation 'P1_E2'.

[1] J.P. Wansapura and others. *Relaxation times in the human brain at 3.0 tesla.* Journal Magnetic Resonance Imaging. 1999 Apr;9(4):531-8. doi: 10.1002\(sici)1522-2586(199904)9:4¡531::aid-jmri4¿3.0.co;2-l. PMID: 10232510.

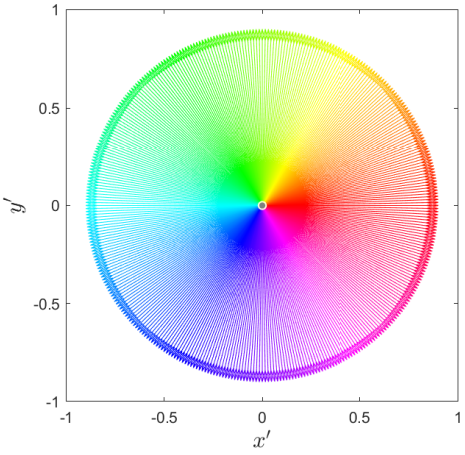

Figure 2: Circular colour map of the direction of transverse magnetisation corresponding to different colours, with unit magnitude of the transverse magnetisation.

As can be seen in the animation the gradient has the effect of increasing the relative Larmor frequency of the isochromats on the left $x < 0$, while decreasing the Larmor frequency of the isochromats on the right $x > 0$. Therefore a graph of a Larmor frequency vs spatial position, would have a negative gradient of γG_x where G_x is prescribed as negative. Additionally $x = 0$ would have a Larmor frequency of γB_0, where B_0 is the static magnetic field. The effect of demodulation removes this high frequency γB_0. Therefore only the effect of the gradient on the isochromats is obtained by the signal, this is done as this effect is important in spatially isolating the underlying spin density function. The animation accurately shows isochromats rotating faster at the extremities of the domain with positive demodulated Larmor frequencies corresponding to clockwise rotations and negative demodulated Larmor frequencies corresponding to anti-clockwise rotations. The $M_{x'}$ channel never reads any non-zero values. This is due to the gradient being applied along the x-axis, and that the distribution of isochromats is symmetric across $x = 0$.

In the background of the top left subplot a finer resolution of isochromats is visualised with the colour map given in fig. 2. This shows very clearly how traversing k-space acts to pick out specific spatial frequencies. This background colour map will be used in all subsequent simulations.

Experiment 3 includes a gradient $G_x = -4.6 \times 10^{-3}$ T/m that de-phases the isochromats for $T_s/2$, then a opposite gradient $-G_x$ is turned on for T_s. Again, the prescribed k_{iso} is 1 1/mm, we can work out the maximum $k_{x,max}$ value from $k_{x,max} = \frac{\gamma}{4\pi} G_x T_s = 0.5023$ 1/mm. Therefore the point $k_x = k_{iso}$ does not occur, and the total signal is only recovered at $k_x = 0$. The gradient echo is achieved at T_s from the RF pulse, or $T_s/2$ after the dephasing lobe. We recall that we want to simulate a uniform spin density. The Fourier transform of a constant signal is a unit impulse. From the animation 'P1_E3' the period over which a recording occurs it can be seen that there is a distinct sinc function over time. This is spurious and due to the field of view (FOV) of the imaging domain. There is an interaction between the FOV, x spacing (Δx), k-space spacing (Δk), and size of k-space ($k_{x,max}$). This will be explored.

1.2 Task 2

This task required generalisations of the code such that individual parameters could be easily changed and their effect investigated. The results of the investigation provide some interesting insights and key animations are provided.

Comments on length of of segment, L: Initially the isochromat spatial frequency was kept constant, whilst the length of the segment was increased. Holding k_{iso} constant meant that the number of isochromats making the segment increased with the increase in L. As previously described when a

$L = 20$ mm, as is the case for task 1 experiment 3, a sinc function signal is obtained when it should be a unit impulse. This is explained as the FOV is not big enough in the imaging domain. Thus it was found that increasing to $L = 100$ mm the signal resembles a unit impulses, see animations 'P1_T2_L100mm_constant_k'. The individual isochromats are not simulated and only the colour map is shown, this computationally cheaper.

A question to ask oneself is, what happens when this FOV is increased too much? We want each point in k-space to correspond to a new phase over the whole of the imaging FOV. From this we can say that $\Delta k = 1/FOV$. This is outside what is needed for the report but is explained succinctly here[2]. Furthermore we need a one full phase over the FOR to spatially differentiate an individual Δx. Therefore another condition to ensure resolution is $2 \cdot k_{x,max} = 1/\Delta x$.

Just varying L, not keeping k_{iso} =constant, has already been somewhat mentioned as simply $L \propto \frac{1}{k_{iso}}$. Thus by increasing L, k_{iso} decreases. If k_{iso} falls below $k_{x,max}$ then as previously described, the total signal is acquired. This is spurious as we are not trying to image the underlying discretisation, thus we the recovery of the total signal away from $k_x = 0$ will be henceforth know as an artefact, and we will ensure that $2 \cdot k_{x,max} = 1/\Delta x$. By lowering L you cannot get enough information for spatial differentiation, equivalently there is no full phase over the FOR for each k-space point. The results of changing $L = 1$ mm and $L = 100$ mm are animations 'P1_T2_L1mm' and 'P1_T2_L100mm' respectively.

Comments on frequency encoding gradient, G_x: By varying G_x you vary both $k_{x,max}$ and Δk, as $k_{x,max} \propto G_x$ and $\Delta k \propto G_x$. Therefore increasing G_x you increase the range of values k_x can be such that $k_x = k_{iso}$ can occur. Vice versa is true for decreasing G_x, but there is still the problem of spatial differentiation. The results of changing $G_x = 0.0046$ mT/m and $G_x = 46$ mT/m are animations 'P1_T2_Gx0046' and 'P1_T2_Gx46' respectively.

Comments on number of sample points, N: The number of sampling points is vitally important. There are several aspects to take into account.

1. The number of sample points taken should be a equal to 2^n. This is a requirement of the Fast Fourier Transform (FFT), which is a computationally efficient method of transforming k-space back into image space.
2. The above requirement requires an even amount of sample points to be used. An even number of sample points would preclude $k_x = 0$ from being recorded, this values is important to record particularly when frequency and phase encoding gradients. Therefore we choose to take one less point in negative k-space and capture $k = 0$.
3. Reducing N limits the spatial resolution attainable, ideally there should be the same number of k-space points as pixels in an image.

At present as only 20 isochromats are simulated, we are oversampling in k-space. Reducing the number of sample points to 10 shows the effect of undersampling. This can be seen in the animation 'P1_T2_undersampling'. Using the same the amount of points for the k-space than the number of isochromats is able to represent the signal exactly this can be seen in 'P1_T2_sparse'. This is a feature of Nyquist sampling criterion, as the frequency of the sinc function is Δk.

Comments on sample time, T_s: By varying T_s you vary both $k_{x,max}$ and Δk, as $k_{x,max} \propto T_s$ and $\Delta k \propto T_s$. This has the same effect as G_x. These animations are not shown for the sake of brevity. There is one caveat is that if the $T_s \approx \mathcal{O}(T_2)$ then the effect of relaxation needs to be considered. In a real-world applications relaxation would decrease the signal-to-noise ratio (SNR).

[2] http://mriquestions.com/field-of-view-fov.html

1.3 Task 3

The segment of isochromats is placed on the x-axis such that the coordinates of each isochromat is $[x, y, z]' = [x, 0, 0]'$. With the x values given as previously discussed, see Fig. 1. If the frequency encoding gradient is chosen along one of the other axes (y, z) then the isochromats are unaffected. See animation 'y_axis' so visualise this. If the segment is not placed along the x-axis, for example $[x, y, z]' = [x, k1, k2]'$ where k1 and k2 are constants. Then you could mitigate the effect of the G_y or G_z gradients by demodulating the signal by $\gamma(B_0 + k1G_y + k2G_z)$. Furthermore if there is a gradient at a orientation ϕ on the x-y plane such that $G = \text{Magnitude} \cdot [\cos\phi\hat{x}, \sin\phi\hat{y}]$. Then only the component along the x-axis ($\text{Magnitude} \cdot \cos\phi$) should be considered, this would have the effect of changing G_x and the effect of this has been discussed previously.

2 Problem 2: Phase Encoding

The extension from a 1D to 2D was trivial given that the software was developed with a translation to 2D in mind. The framework for discretisation was extended from Fig. 1 for 2D. The problem specifies that two identical non-overlapping squares should be placed in the 1st and 3rd quadrants of a plane with the centroids of each square being the same distance from the origin and lie on the line $y = x$. Making the assumption that we want the same discretisation in both the x and y directions. The problem is re-parameterised to depend on two parameters, d the distance between the squares and Num_{iso} the number of isochromats in along the edge of each square. Effectively there are only two degrees of freedom for spatial description of this simulation.

Another comment on the objective of the simulation: The purpose of this simulation is to model the physical forward operator of MRI. Then from the obtained signal we could reconstruct the image. Therefore using literature on the image reconstruction we can gleam what we need to accurately obtain a signal from isochromats. In particular, the literature on FOV from mriquestions.com is useful, knowing that we want, in essence, an isochromat in the physical domain to represent a pixel in the image domain, thus the spatial spacing of isochromats and pixels have the same relationship with k-space. And we can say that $2 \cdot k_{x,max} = k_{iso}$, this constraint stops the production of the spurious total signal recovery which can be thought of as aliasing but in the Fourier domain. Additionally we should have equal numbers of isochromats and k-space values to ensure that just enough k-space values are collected to represent the signal efficiently, whilst maintaining resolution. We use $\Delta k = 1/FOV$ as a weak constraint as a signal is still recoverable with lower FOV, but suffers from a lack of discretisation to accurately represent the Fourier domain (quite like blurring in the Fourier domain), this can cause spurious signals that do not capture peaks and troughs correctly.

We prescribe $T_s = 5.12$ ms and $G_x = -4.6$ mT/m, thus $k_{max} = 0.5023$ 1/mm as described before. From this we can calculate the spatial frequency of the isochromats to be: $k_{iso} = 2 \cdot k_{x,max} \approx 1$ 1/mm. From this we can calculate the length of an edge of the square from $L = Num_{iso} \cdot k_{iso}$, and along with d we can fully specify the spatial representation of the squares in the physical domain, as they are highly constrained in the problem.

2.1 Task 1

The first task has a non-overlapping phase-encoding and dephasing read-out gradient (the read-out gradient is the frequency-encoding gradient G_x). These blocks of the sequence diagram each occur over $T_s/2 = 2.56$ ms. Thus the total time from the end of the RF pulse (flipping spins to y') to rephasing lobe of read-out gradient is T_s. To traverse k-space the phase-encoding gradient is varied in the range $G_y = (G_x, -G_x)$. The step change $\Delta k_x = \Delta k_y$ is used to ensure the same k-space coverage is achieved in both phase and frequency encoding directions.

As previously mentioned, it is important to record the value at $k = 0$, as $N = 2^n$ the number of k-space sample points is always an even number (n is a natural number) which helps when reconstructing an image with FFT (no padding needed). Therefore to include $k = 0$ and have a uniform spacing Δk, we sample one more negative k-space value in both frequency and phase encoding directions. Given

$k_{max} = 0.5023$ 1/mm, for 256 points we choose a $\Delta k = \frac{2 \cdot 0.5023}{256} = 3.92 \times 10^{-3}$ 1/mm, where the sampled points $k_x^i = k_y^i = -0.5023 + 3.92 \cdot i \times 10^{-3}$ 1/mm where $i = 0, 1, 2, 3, 4, ..., 254, 255$.

It is also worth noting that Φ_{iso} is out-phase, this can be changed to in-phase by increasing the d by $\frac{\sqrt{2}}{2k_{iso}}$. This was found to have no impact on the recovered signal, this can be somewhat attributed to the fact that $2 \cdot k_{x,max} = k_{iso}$ so the total signal is never recovered. Changing d arbitrarily did not seem to have an impact on the measured signal. This is better explored in task 2.

The animation for this task is denoted 'P2_T1'. It shows the inefficiency of non-overlapping phase-encoding and dephasing read-out gradients. And the time 'after initial dephase' lobe of the readout, does not include any time taken for subsequent relaxations and RF pulses. The simulation is done over has 16 spatial points in the x and y directions. Therefore 128 isochromats are simulated. The distance between centroids is set as $d = 2\sqrt{2}L$ where L is the length of the sides of one of the squares. If d is increased the FOV increases and we need to sample more of k-space to fully resolve the signal. For $d = 2\sqrt{2}L$ the same amount of points spatial points can be used in k-space to fully resolve the signal, i.e. 16 points were used for k_x and k_y. Albeit 256 represent k-space but only 128 isochromats represent physical domain. It was found that the majority of signal is held at the central portion of k-space.

By placing the squares in first and third quadrants of physical space with the corners touching we take advantage of the symmetry of k-space. This means that although only half of physical space has isochromats in it, the signal produced is equivalent to that from a uniform distribution of isochromats all over the FOV. Additionally the signal recovered along $k_x = 0$ or $k_y = 0$ is the same as a 1D example segment.

2.2 Task 2

The simulation of 256 points in k-space and 128 points in physical domain is given by 'P2_T2', with $d = 2\sqrt{2}L$. Thus the exact same features of k-space were found as in the previous task.

The pulse sequence in this task is more efficient than the previous. This is because the total time from the end of the RF pulse (flipping spins to y') to rephasing lobe of read-out gradient is $T_s/2$ not T_s. This is true when they overlap as much as possible. Therefore we tested the effect of changing d using this pulse sequence as it is more efficient. Initially we look at adding extra d such that Φ_{iso} is in-phase. This can be seen in animations 'P2_T2_in_phase', it was found that there was not too much difference. Furthermore we increase the number of points taken in k_x to ensure the full signal is correctly sampled, we also decrease the number of samples in k_y to ensure the task is computationally viable. These animations are shown in by 'P2_T2_in_phase_kx64' and 'P2_T2_out_phase_kx64', again we found not too much difference between the in-phase and out-phase solutions.

Increasing d increases the FOV and therefore the signal needs to be sampled with more points. Furthermore, if d is large enough, each square of isochromats can be though as rect-functions of spin density away from the origin, and increasing d further they can be though of unit impulses of spin density at the corners of the physical domain. Thus the Fourier representation of these squares with a large d value is multiple sinc functions interacting with one another. This can be visualised in animations 'P2_T2_in_phase_d10mm_kx256' and 'P2_T2_out_phase_d10mm_kx256', again there is not too much difference between the in-phase and out-phase solutions.

At the limit as d becomes very large the Fourier representation should tend towards a constant value as squares effectively become point sources, but this is not computationally feasible to simulate.

The effect of d can also be thought of introducing another spatial frequency into the features of the physical domain. For example, if $d = 2\sqrt{2}L \cdot n$ where $n > 1$ then specific sampling spatial frequencies k_x and k_y could pick out this introduced spatial frequency, which will be a very small.

Comments on the code: There are 5 distinct parts of the code: spatial description, isochromat

initialisation, pulse sequence block description and initialisation, solving, and animations. If this code were to be developed further a graphical user interface could be used for spatial description and pulse sequence block description, and then initialised within code with error handling. Furthermore the solving part of the code can be made faster through parallelisation, each isochromat is solved individually for magnetisation over the pulse then added to the overall magnetisation and the values are then cleared to save memory. The isochromats could be solved in parallel. Finally, there is a B_0 field for each isochromat this could used to account for static field inhomogeneities. It was found that using *imagesc* instead of *quiver* greatly reduced run time, using images of the underlying phase of isochromats represented by colour results is quicker processing of animations.

A Work-horse functions:

Full scripts with individual modules can be found on Github, https://github.com/Imraj-Singh/COMP0121.

```
function iso = initial_dir(iso,Axis,N)
%Initialise the spin direction
%   User defined direction of the spin, can only be set to
%   along the positive axis directions

% Error handling
if (nargin ~= 3)
    error('The number of arguments is: %d there should be only one argument
        either x, y or z.', nargin)
end
if any((Axis~='x')&(Axis~='y')&(Axis~='z'))
    error('Needs be an axis of either x,y or z.')
end
% Initalise the direction
if (Axis=='x')
    iso.Mx = [iso.Mx; ones(N,1)];
    iso.My = [iso.My; zeros(N,1)];
    iso.Mz = [iso.Mz; zeros(N,1)];
elseif (Axis=='y')
    iso.Mx = [iso.Mx; zeros(N,1)];
    iso.My = [iso.My; ones(N,1)];
    iso.Mz = [iso.Mz; zeros(N,1)];
elseif (Axis=='z')
    iso.Mx = [iso.Mx; zeros(N,1)];
    iso.My = [iso.My; zeros(N,1)];
    iso.Mz = [iso.Mz; ones(N,1)];
end
end
```

```
function iso = initialise_B0(iso,B0)
%INITALISE_B0 by Imraj Singh
% Inputs
%   iso,B0
% Outputs
%   iso

% Set the B0 values
for i=1:length(iso(:,1))
    for j=1:length(iso(1,:))
        iso(i,j).B0 = B0;
    end
end
end
```

```
function iso = initialise_iso(x,y)
%INITALISE_ISO by Imraj Singh
% Inputs
%   x,y
% Outputs
%   iso

% Initialise the isochromat structure and set the x,y values
iso(1:length(x)) = struct();
for i=1:length(x)
        iso(i).x = x(i);
        iso(i).y = y(i);
```

```
        iso(i).Mx = [];
        iso(i).My = [];
        iso(i).Mz = [];
end
end
```

```
function iso = rf_pulse(iso, Axis, Angle, N, T)
%FREE_RELAXATION by Imraj Singh
% Inputs
%   iso, Axis, Angle, N, T
% Outputs
%   iso

% Error handling
if any((Axis~='x')&(Axis~='y')&(Axis~='z'))
    error('Needs be an axis of either x,y or z.')
end
if (~isa(Angle,'double'))
    error('Angle needs to be a double.')
end

% Create time vector
t = linspace(0,T,N)';

% Calculate
if (Axis=='x')
    M_x = iso.Mx(end)*ones(N,1);
    M_y = iso.My(end).*cos(Angle/T*t) + iso.Mz(end).*sin(Angle/T*t);
    M_z = iso.Mz(end).*cos(Angle/T*t) - iso.My(end).*sin(Angle/T*t);
elseif (Axis=='y')
    M_x = iso.Mx(end).*cos(Angle/T*t) - iso.Mz(end).*sin(Angle/T*t);
    M_y = iso.My(end).*ones(N,1);
    M_z = iso.Mz(end).*cos(Angle/T*t) + iso.Mx(end).*sin(Angle/T*t);
elseif (Axis=='z')
    M_x = iso.Mx(end).*cos(Angle/T*t) + iso.My(end).*sin(Angle/T*t);
    M_y = iso.My(end).*cos(Angle/T*t) - iso.Mx(end).*sin(Angle/T*t);
    M_z = iso.Mz(end)*ones(N,1);
end

% Add it appropriate fields
iso.Mx = [iso.Mx; M_x];
iso.My = [iso.My; M_y];
iso.Mz = [iso.Mz; M_z];
end
```

```
function [Gx_t, Gy_t, kx, ky] = unroll_plotting(N,Gx,Gy,resetk,gamma,Time)
%UNROLL_PLOTTING by Imraj Singh
% Inputs
%   N,Gx,Gy,resetk,gamma,Time
% Outputs
%   Gx_t, Gy_t, kx, ky

% Initalise the variable
Gx_t = [];
Gy_t = [];
kx = zeros(1,sum(N));
ky = zeros(1,sum(N));

% Create a vector of the gradient in time
```

```
for i=1:length(N)
    Gx_t = [Gx_t, ones(1,N(i))*Gx(i)];
    Gy_t = [Gy_t, ones(1,N(i))*Gy(i)];
end

% Create a vector of the k-space value in time
for i=2:length(N)
    % Reset the k value to 0,0 if instructed to
    if resetk(i)==1
        kx(sum(N(1:i-1)+1):sum(N(1:i))) = linspace(0,0,N(i));
        ky(sum(N(1:i-1)+1):sum(N(1:i))) = linspace(0,0,N(i));
    else
        % calculate the k-space value
        for j=sum(N(1:i-1))+1:sum(N(1:i))
            kx(j) = kx(j-1) + (Time(j)-Time(j-1))*gamma/(2*pi)*Gx(i);
            ky(j) = ky(j-1) + (Time(j)-Time(j-1))*gamma/(2*pi)*Gy(i);
        end
    end
end




end
```

```
function s_filter = unroll_signal(N, signal)
%UNROLL_SIGNAL by Imraj Singh
% Inputs
%   M, signal
% Outputs
%   s_filter

% Initialise array
s_filter = [];


for i=1:length(N)
    % Create the binary filter to pick out sample values
    s_filter = [s_filter, ones(1,N(i))*signal(i)];

    % Miss out the last element of the signal block so the k is kept
    % whilst ensuring 2^n points sampled for the fft
    if i>1
        s_filter(sum(N(1:i)))=0;
    end

end
end
```

```
function Time = unroll_time(N,t)
%UNROLL_TIME by Imraj Singh
% Inputs
%   N,t
% Outputs
%   Time

% Create the time vector
Time = linspace(0, 0, sum(N));
Time(1:N(1)) = linspace(0, t(1), N(1));
```

```
for i = 2:length(N)
    Time(sum(N(1:i-1))+1:sum(N(1:i))) = linspace(sum(t(1:(i-1))),sum(t(1:i)),N(i)
        );
end
end
```

B Example code:

This code was used for running the most complex simulation, problem 2 task 2.

```
%% Problem 1: Experiment 2 gradient relaxation
% Author: Imraj Singh 08/11/2020

clc
clear
close all

% Add the path that holds functions
addpath 'C:\Users\Imraj Singh\Documents\UCL\Comp_MRI\COMP0121\Coursework_2\
    Functions'

%% Define coordinate position of each isochromat

% Number of isochromats along one side of square
Num = 8;


% Spatial spacing
dxy = 1/(500*2);

% Length of segment
L = Num*dxy;

% Sampling time
Ts = 5.12/1000;

% Sampling time
Gradient = -4.6/1000;

% Number of isochromats x and y
Numxy = L/dxy/2;

% Distance between the two squares
d = 10*dxy + dxy*.5;

% Spatial coordinates of isochromats (m)
x1 = linspace(-L/2 + dxy/2 - d, 0 - dxy/2 - d,Numxy);
y1 = linspace(-L/2 + dxy/2 - d, 0 - dxy/2 - d,Numxy);

% Spatial coordinates of isochromats (m)
x2 = linspace(0 + dxy/2 + d, L/2 - dxy/2 + d,Numxy);
y2 = linspace(0 + dxy/2 + d, L/2 - dxy/2 + d,Numxy);

% Create mesh grid of x and y coordinates
[X1, Y1] = meshgrid(x1,y1);
% Create mesh grid of x and y coordinates
[X2, Y2] = meshgrid(x2,y2);

% Flatten the array into a vector to make manipulation easier
X = [X1(:);X2(:)];
```

```
Y = [Y1(:);Y2(:)];

clearvars X1 Y1 X2 Y2
%% Set NMR parameters and initialise isochromat structures

% Gyromagnetic ratio of hyrodrogen proton
gamma = 2.68*10^8;

% Initalise the isochromats, setting the X, Y cooridates and instantiating
% the Mx, My, Mz fields of the structures
iso = initialise_iso(X, Y);

% B0
B0 = 1.5;

% Initalise the B0 this is done such that arbitrary perturbation can be
% added to the B0 field to simulate field inhomogenities
iso = initialise_B0(iso, B0);

% Larmor frequency of rotating frame of reference
R_FoR = B0*gamma;

% Setting the brain tissure relaxation values. Approximation based off:
% Wansapura JP, Holland SK, Dunn RS, Ball WS Jr. NMR relaxation times in
% the human brain at 3.0 tesla. J Magn Reson Imaging. 1999 Apr;9(4):531-8.
% doi: 10.1002/(sici)1522-2586(199904)9:4<531::aid-jmri4>3.0.co;2-l.
% PMID: 10232510. Approximate average of white and grey matter.
T1 = 1;
T2 = .1;

clearvars B0

%% Define the experiment in terms of blocks

% Each experiment begins with initialising the direction of the spins then
% by flipping the spins with a pi/2 hard RF pulse along the x' axis over
% 0.32ms

NumGx = 256;
NumGy = 2;
Gradient_y = linspace(Gradient,-Gradient, NumGy + 1);

% [Set the number of points to be calculated minimum of 2 for each block
%blocks.N = [1 65 129 257];
blocks.N = [1 21 1+NumGx/2 1+NumGx];

% Time over which the block occurs
blocks.t = [0 .32/1000 Ts/2 Ts];

% The gradient of the field dB/dx for each block
blocks.Gx = [0 0 Gradient -Gradient];

% The gradient of the field dB/dz for each block
blocks.Gy = [0 0 Gradient_y(1) 0];

% The relaxation time only when spins are relaxation
blocks.relax = [0 0 Ts/2 Ts];

% Reset k-space
blocks.resetk = [1 0 0 0];

```

```
% Where the signal is collected
blocks.signal = [0 0 0 1];

for i=2:NumGy
    blocks.N = [blocks.N blocks.N(1) blocks.N(3) blocks.N(4)];
    % Time over which the block occurs
    blocks.t = [blocks.t 0 Ts/2 Ts];

    % The gradient of the field dB/dx for each block
    blocks.Gx = [blocks.Gx 0 Gradient -Gradient];

    % The gradient of the field dB/dz for each block
    blocks.Gy = [blocks.Gy 0 Gradient_y(i) 0];

    % The relaxation time only when spins are relaxation
    blocks.relax = [blocks.relax 0 Ts/2 Ts];

    % Reset k-space
    blocks.resetk = [blocks.resetk 1 0 0];

    % Where the signal is collected
    blocks.signal = [blocks.signal 0 0 1];
end

% Initialise the Mx, My, Mz values that stores the sum of all the spins
% magnetisation
Mx = zeros(sum(blocks.N),1);
My = zeros(sum(blocks.N),1);
Mz = zeros(sum(blocks.N),1);

% Subset definition
Sub_num = 2;
Sub_Set = [ min(x1) max(x1) min(x2) max(x2) ];%[(min(x1)-max(x1))/2 min(x1) max(
    x1) (max(x2)-min(x2))/2 min(x2) max(x2) ];

% Colour map ensuring that only isochromats have colour
%map = ones(Numxy*2,Numxy*2,sum(blocks.N))*-1;

z = 1;
Check_Dup = false;
% Calculate magnetisation for all the spins
for i=1:length(iso)
    % Initialise spin
    iso(i) = initial_dir(iso(i),'z',blocks.N(1));

    % RF pulse
    iso(i) = rf_pulse(iso(i), 'x', pi/2, blocks.N(2), blocks.t(2));

    % Free relaxation with phase encoding gradient
    iso(i) = free_relaxation(iso(i), blocks.N(3), blocks.t(3), ...
        blocks.Gx(3), blocks.Gy(3), blocks.relax(1:3), gamma, T1, T2, R_FoR);

    % Free relaxation with dephasing readout
    iso(i) = free_relaxation(iso(i), blocks.N(4), blocks.t(4), ...
        blocks.Gx(4), blocks.Gy(4), blocks.relax(1:4), gamma, T1, T2, R_FoR);


    for ii=2:NumGy
        index = ii-1+(ii-2)*2;
        % Initialise spin
        iso(i) = initial_dir(iso(i),'y',blocks.N(4+index));
```

```
171
172         % Free relaxation with phase encoding gradient
173         iso(i) = free_relaxation(iso(i), blocks.N(5 + index), blocks.t(5 + index
                ), ...
174             blocks.Gx(5+index), blocks.Gy(5+index), blocks.relax(1:3), gamma, T1
                    , T2, R_FoR);
175
176         % Free relaxation with dephasing readout
177         iso(i) = free_relaxation(iso(i), blocks.N(6 + index), blocks.t(6 + index
                ), ...
178             blocks.Gx(6+index), blocks.Gy(6+index), blocks.relax(1:3), gamma, T1
                    , T2, R_FoR);
179     end
180
181     % Calculate the index's for the map array dependent on spatial location
182     ix = round((iso(i).x + L/2 - dxy/2 + d)/(x1(2)-x1(1)) + 1);
183     iy = round((iso(i).y + L/2 - dxy/2 + d)/(y1(2)-y1(1)) + 1);
184
185     % Calculate the angles on the x-y plane the spin vector for each
186     % isochromat corresponding to a pixel, additionally we round up with
187     % each pixel as there are only 360 unique colours in the colour map,
188     % also we made sure that the calculated angles fall between 1->360
189     for k=1:sum(blocks.N)
190         map(iy,ix,k) = ceil(mod(atan2(iso(i).My(k), iso(i).Mx(k)),2*pi)/(pi)
                *180);
191     end
192
193
194     % Find the subset of isos that we specify that can select certain
195     % isochromats within the whole domain which we calculated for the
196     % imagesc for better visualisation
197     if  ismembertol(iso(i).x,Sub_Set) && ismembertol(iso(i).y,Sub_Set)
198         % Add to the plottable sub-set
199         Sub_Set_iso(z) = iso(i);
200
201         % Change the Check_Dup value once the first [0,0] isochromat is
202         % found this stops the duplicate values being saved
203         if iso(i).x == 0 && iso(i).y == 0
204             Check_Dup = true;
205         end
206
207         % Don't save the extra [0,0] isochromat as the [0,0] will never be
208         % at the end of the iso(i) array struct so the next value in the
209         % subset will overwrite it
210         if iso(i).x == 0 && iso(i).y == 0 && Check_Dup == false
211         else
212             z=z+1;
213         end
214
215     end
216
217     % Add the magnetisations to the total  magnetisation
218     Mx = Mx + iso(i).Mx;
219     My = My + iso(i).My;
220     Mz = Mz + iso(i).Mz;
221
222     % Clear the arrays as they take up a lot of memory
223     iso(i).Mx = [];
224     iso(i).My = [];
225     iso(i).Mz = [];
226 end
```

```
227
228 % Magnetisation now normalised across all the spins
229 Mx = Mx/(length(iso));
230 My = My/(length(iso));
231 Mz = Mz/(length(iso));
232
233 % Calculate the time vector
234 Time = unroll_time(blocks.N,blocks.t);
235
236 % Calculate the Gx, Gy, kx, ky vectors in time
237 [Gx_t, Gy_t, kx, ky] = unroll_plotting(blocks.N,blocks.Gx,blocks.Gy,blocks.
    resetk,gamma,Time);
238
239 s_filter = unroll_signal(blocks.N, blocks.signal);
240
241 clearvars iso z Check_Dup ix iy R_FoR T1 T2 Ts X Y x1 y1 gamma NumGx NumGy
    Gradient_y index ii i k
242 %% Animation module
243
244 % Normalisation constants used for plotting the quiver3s
245 norm = .5;
246 norm_scale = 2;
247
248 % Set the video writing name and location
249 video = VideoWriter(['P2_T2', '.mp4'], 'MPEG-4');
250
251 % Set the frame rate of video
252 frameRate = 5;
253 video.set('FrameRate', frameRate);
254
255 % Open the video
256 video.open();
257
258 % Open a figure that has frame recorded
259 h = figure;
260
261 for i=1:length(Time)
262
263     if Time(i)>=0 && Time(i)<=sum(blocks.t(1:1))
264         sgtitle(['Flipping the spins, time: ', num2str(Time(i)*1000), ' ms'],
              interpreter, latex, fontsize, 15)
265     elseif Time(i)>=sum(blocks.t(1:1)) && Time(i)<=sum(blocks.t(1:2))
266         sgtitle(['Flipping the spins, time: ', num2str(Time(i)*1000), ' ms'],
              interpreter, latex, fontsize, 15)
267     elseif Time(i)>sum(blocks.t(1:2)) && Time(i)<=sum(blocks.t(1:3))
268         sgtitle(['Phase encoding, time (after flip): ', num2str((Time(i)-sum(
              blocks.t(1:2)))*1000), ' ms'], interpreter, latex, fontsize, 15)
269     elseif Time(i)>sum(blocks.t(1:3)) && Time(i)<=sum(blocks.t(1:4))
270         sgtitle(['Dephasing lobe, time (after encode): ', num2str((Time(i)-sum(
              blocks.t(1:3)))*1000), ' ms'], interpreter, latex, fontsize, 15)
271     elseif Time(i)>sum(blocks.t(1:4)) && Time(i)<=sum(blocks.t(1:5))
272         sgtitle(['Recording signal, time (after dephase): ', num2str((Time(i)-
              sum(blocks.t(1:4)))*1000), ' ms'], interpreter, latex, fontsize, 15)
273     elseif Time(i)>sum(blocks.t(1:5)) && s_filter(i)==0
274         sgtitle(['Not recording, time (after initial dephase): ', num2str((Time(
              i)-sum(blocks.t(1:4)))*1000), ' ms'], interpreter, latex, fontsize, 15)
275     elseif Time(i)>sum(blocks.t(1:5)) && s_filter(i)==1
276         sgtitle(['Recording signal, time (after initial dephase): ', num2str((
              Time(i)-sum(blocks.t(1:4)))*1000), ' ms'], interpreter, latex, fontsize, 15)
277     end
278
```

```
    % First subplot - quiver of spin vectors in space
    subplot(2,2,1)
    colormap([1 1 1; hsv(360)]);

    if Time(i)>=0 && Time(i)<sum(blocks.t(1:2))
        imagesc(linspace(-L/2 + dxy/2 - d,L/2 - dxy/2 + d,100)*1000,linspace(-L
            /2 + dxy/2 - d,L/2 - dxy/2 + d,100)*1000,ones(length(map(:,:,1)))
            *-1,[0 360])
    elseif Time(i)>=sum(blocks.t(1:2))
        imagesc(linspace(-L/2 + dxy/2 - d,L/2 - dxy/2 + d,100)*1000,linspace(-L
            /2 + dxy/2 - d,L/2 - dxy/2 + d,100)*1000,map(:,:,i),[0 360])
    end
    hold on
    % First element of vector of iso structs
    quiver3(Sub_Set_iso(1).x*1000,Sub_Set_iso(1).y*1000,0,Sub_Set_iso(1).Mx(i)*
        norm,Sub_Set_iso(1).My(i)*norm,Sub_Set_iso(1).Mz(i)*norm,'linewidth',2,'
        LineStyle','-','Color','k');
    scatter(Sub_Set_iso(1).x*1000,Sub_Set_iso(1).y*1000, 10,'MarkerEdgeColor',[0
         0 0],'MarkerFaceColor',[1 1 1],'LineWidth',1);
    for z=2:length(Sub_Set_iso)
        % z'th element of vector of iso structs
        quiver3(Sub_Set_iso(z).x*1000,Sub_Set_iso(z).y*1000,0,Sub_Set_iso(z).Mx(
            i)*norm,Sub_Set_iso(z).My(i)*norm,Sub_Set_iso(z).Mz(i)*norm,'
            linewidth',2,'LineStyle','-','Color','k');
        scatter(Sub_Set_iso(z).x*1000,Sub_Set_iso(z).y*1000, 10,'MarkerEdgeColor
            ',[0 0 0],'MarkerFaceColor',[1 1 1],'LineWidth',1);
    end
    set(gca,'YDir','normal');
    grid on
    box on
    hold off
    % Set the axes labels
    xlabel(x', interpreter, latex, fontsize, 10)
    ylabel(y', interpreter, latex, fontsize, 10)


    % Second subplot - Magnetisation vs Time
    subplot(2,2,2);
    % Plot the full magnetisation
    plot(Time(1:i)*1000,Mx(1:i),'linewidth',1,'LineStyle','-','Color','r')
    hold on
    plot(Time(1:i)*1000,My(1:i),'linewidth',1,'LineStyle','-','Color','b')
    % plot the signal points
    if Time(i)>sum(blocks.t(1:3))
        plot(Time(find(s_filter(1:i)))*1000,Mx(find(s_filter(1:i))),'.','
            MarkerSize',8,'Color','r')
        plot(Time(find(s_filter(1:i)))*1000,My(find(s_filter(1:i))),'.','
            MarkerSize',8,'Color','b')
    end
    grid on
    box on
    hold off
    % Set the legend
    legend(M_x', M_y', interpreter, latex, fontsize, 10,'Location','southeast')
    % Set the axes limits
    if Time(i)>=0 && Time(i)<sum(blocks.t(1:4))
        xlim([0 sum(blocks.t(1:4))]*1000)
    elseif Time(i)>=sum(blocks.t(1:4))
        xlim([Time(i)-sum(blocks.t(1:4)) Time(i)]*1000)
    end
    ylim([-1 1]*1.2);
```

```
    % Set the axes labels
    xlabel(Time (ms), interpreter, latex, fontsize, 10)
    ylabel(Magnetisation, interpreter, latex, fontsize, 10)


    % Third subplot - K space values
    subplot(2,2,3);
    % plot the signal points
    if Time(i)>sum(blocks.t(1:3))
        plot(kx(find(s_filter(1:i)))/1000,ky(find(s_filter(1:i)))/1000,'.','
            MarkerSize',8,'Color','k')
        hold on
    end
    % plot the current location
    plot(kx(i)/1000,ky(i)/1000,'x','MarkerSize',5,'Color','k')
    grid on
    box on
    hold off
    % Set the axes limits
    xlim([-1.2 1.2]/2);
    ylim([-1.2 1.2]/2);
    % Set the axes labels
    xlabel($k_x$ (1/mm), interpreter, latex, fontsize, 10)
    ylabel($k_y$ (1/mm), interpreter, latex, fontsize, 10)


    % Fourth subplot - Time vs K space
    subplot(2,2,4);
    % X component
    plot(Time(1:i)*1000,kx(1:i)/1000,'linewidth',1,'LineStyle','-','Color','r')
    hold on
    % Y component
    plot(Time(1:i)*1000,ky(1:i)/1000,'linewidth',1,'LineStyle','-','Color','b')
    grid on
    box on
    hold off
    % Set the legend
    legend($k_x$,$k_y$, interpreter, latex, fontsize, 10,'Location','northwest')
    % Set the axes limits
    if Time(i)>=0 && Time(i)<sum(blocks.t(1:4))
        xlim([0 sum(blocks.t(1:4))]*1000)
    elseif Time(i)>=sum(blocks.t(1:4))
        xlim([Time(i)-sum(blocks.t(1:4)) Time(i)]*1000)
    end
    ylim([-.6 .6])
    % Set the axes labels
    xlabel(Time (ms), interpreter, latex, fontsize, 10)
    ylabel(Spatial frequency (1/mm), interpreter, latex, fontsize, 10)
    % Set frame to add to video
    frame = getframe(h);
    video.writeVideo(frame);
end

% Close the video
video.close();
```