Nanoparticles as the origin of contrast in magnetic resonance imaging

Dominic Askew
Student ID: 4252677

School of Physics and Astronomy

University of Nottingham

Superparamagnetic nanoparticles are used in medical applications of magnetic resonance imaging to artificially create a contrast. The increased contrast of an image produced in an MRI procedure results in an improved resolution thus a better diagnosis can be done. In this research report, we will focus on the physics underpinning the contrast produced from a nanoparticle in an external magnetic field by writing a program which can simulate the signal decay of hydrogen ions and finding the corresponding T_2 *relaxation times. We will also be analysing how* T_2 *varies upon variations to the nanoparticles magnetic moment, and its concentration. From our simulation, we found that an increase in magnetic moment, lead to a decrease in* T_2*. We also found that an increase in nanoparticle concentration, lead to a decrease in* T_2*.*

<u>Contents:</u>

Introduction:

Magnetic resonance imaging techniques study the interactions between an external magnetic field and nuclear spin. Analysing the behaviour of the nuclear spin following current MRI procedures can tell us about how signal decay of spin states leads to the origin of contrast in imaging. The aim of this research project is to develop a deeper understanding of nuclear magnetic resonance in to produce a close-formed model simulating the signal decay due to a superparamagnetic nanoparticle following a process called transverse relaxation. From this simulation, we would like to assess how this artificial contrast varies upon adjustment to fundamental variables such as the magnetic moment of the nanoparticle. To complete these aims we must develop a 3-dimensional Monte Carlo simulation to model the random motion of the surrounding self-diffusing water molecules as well as apply relevant theory about an MRI procedure to implement a nanoparticle whose local magnetic fields produce a contrast. We must also verify the model against experimental and analytical results to ensure our simulation can produce data similar to that of an NMR spectrometer.

Although most of the underlying physics in magnetic resonance imaging had be described prior to its discovery, it was Dr. Lawrence Bennett and Dr. Irwin Weisman in the early 1970s who published their findings about MRI in the journal science on how NMR being used to differentiate between normal mouse tail tissue and malignant transplanted melanoma, S91, located on the tail.[1] However, more notably, physicists Dr. Peter Mansfield and Dr. Paul Lauterbur developed more advanced techniques such as the echo-planar imaging technique optimizing scan times and producing clear images.[2] This ultimately lead to them being awarded the 2003 Nobel Prize in physiology or medicine for their "discoveries concerning magnetic resonance imaging". Since then, advancements in MRI techniques have been developed to increase the imaging resolution to improve medical diagnoses and the quality of treatment. Numerous scientists and institutions are credited with development of MRI techniques.[3][4]

Nuclear magnetic resonance is extensively used in medicine in the form of MRI, however, there is an abundance of other applications of NMR such as in the fields of computer science and chemistry. Its use in spectroscopy allows for the structure of many compounds to be determined since observed nuclear precession frequencies are specific to individual compounds; almost like a finger print, where comparison to known trends can determine the composition and structure of any compound.[5] Another use of NMR is its application in the petroleum industry. It is used as an analysis method of rocks. More specifically, instruments of low field NMR are able to determine rock porosity, and the surrounding fluids, such as water, oil and gas since different materials log different measurements of T_2 decay[6].

Applying our research project to the fields of medicine; We aim to describe how variations in fundamental properties of superparamagnetic nanoparticles effect the magnitude of the T_2 relaxation time, as well as being able to determine T_2 providing input variables of any system following our simulation. This way, nanoparticles can be artificially modified for any medium to allow for a large contrast in magnetic resonant images.

Theory:

Within water molecules, the hydrogen nucleus, ^1H, has spin ½ and a large magnetic susceptibility making it a perfect natural choice for imaging since it is so abundant in the human body specifically.
To effectively simulate the random motion of water molecules within the body we need to consider the self-diffusion of water as well as making assumptions for the surrounding medium. In a physical system, the self-diffusion of water will tend to spread from regions of high concentration to regions of lower concentration therefore, if each particle has no preferred direction of oscillation we can consider the diffusion of water as a macroscopic manifestation of Brownian motion on the microscopic level. For a particle diffusing in 1-D, the relative distance a particle can move is described by:

$$\Delta x = \sqrt{2D\Delta t} \tag*{\{1\}}$$

Where the time, t, required to diffuse over a distance, Δx, is proportional to the square distance, x^2. The constant of proportionality is defined as the diffusivity, D, in units $\mathrm{m^2s^{-1}}$.
Simulating a random walk in 1-D for an ensemble of N particles following the motion described in equation {1}, the final positions of the ensemble should be distributed according to the theoretical equation derived by Crank's 'Mathematics of Diffusion':[7]

$$C = \frac{N_p}{2\sqrt{\pi D t}} e^{\left(\frac{-x^2}{4Dt}\right)} \tag*{\{2\}}$$

Where N_p is the number of particles. Crank also tells us that for a simulation of N particles confined in a set region *-h > x > +h*, then the theoretical distribution of particles can be expressed as:

$$C = \frac{1}{2} C_o \left\{ erf\left(\frac{h-x}{2\sqrt{Dt}}\right) + erf\left(\frac{h+x}{2\sqrt{Dt}}\right) \right\} \tag*{\{3\}}$$

Where *erf*() represents the gaussian error function used in Crank's derivation as a consequence of summing the effect of a series of line sources, each yielding an exponential type distribution.

In a relaxed system, with no external magnetic field applied, a macroscopic collection of hydrogen atoms, such as tissue saturated in water, has no net magnetic moment since the nuclear spin states are randomly orientated. The orientation of spin states however slightly favours the lower energy state meaning there are slightly more spin states in the lower energy bound than there are in the higher energy bound however, this number is so insignificantly small that overall, the net magnetic moment is zero. Introducing a static field, B_o, the spin states will precess about axis parallel to the static field with Larmor frequency:

$$\omega_o = \frac{\gamma B_o}{2\pi} \tag{4}$$

Where γ is the gyromagnetic ratio, units $s^{-1}T^{-1}$ (By convention, B_o is labelled in the *z*-axis).

Due to the static field, there is now a spin imbalance since conventionally, more spin states will align with B_o than the opposing direction; This ratio is proportional to $e^{-\frac{\Delta E}{KT}}$. The Zeeman energy, ΔE, is the energy splitting between the spin up and spin down states and is proportional to B_o:

$$\Delta E = h\omega_o = \frac{h\gamma B_o}{2\pi} \tag{5}$$

Applying a radio frequency (RF) magnetic pulse, spin states are excited out of equilibrium, causing the net magnetization, M, to rotate towards the transverse (*x-y)* plane. The net magnetization can be made to lie entirely on the transverse plane, perpendicular to B_o if the pulse is sufficiently long and high in amplitude; more commonly known as a $\frac{\pi}{2}$ pulse.

Following the approach of transverse magnetization in the case where a magnetic field gradient is present, a method originally used by Carr and Purcell in their paper on 'Spin echoes', the only motion of concern is that along the field gradient axis labelled, *x*.

For linear field gradient, G, with positional dependence, *x*, the local Larmor frequency in each case is:

$$\omega(x) = \frac{\gamma(B_o + G)}{2\pi} \tag{6}$$

then, the phase angle after a time, Δt, is:

$$\varphi(\Delta t) = \omega(x)\Delta t \tag{7}$$

A cumulative sum of the phase angle over a total time, t, describes how individual spin states precess over time in a linear field gradient. To allow for a simplified calculation to determine how the net magnetization decays over time, t, we observe the spin states in the rotating frame, ω_o. Observing the system in the rotating frame allows us to break down the phase angle, φ, into individual *x* and *y* components:

$$\begin{aligned} x_\varphi &= \cos\varphi, \\ y_\varphi &= \sin\varphi. \end{aligned} \tag{8}$$

Then, taking the mean over an ensemble of N particles, the net magnetization is described as:

$$M = \sqrt{(\overline{x_\varphi})^2 + (\overline{y_\varphi})^2} \tag{9}$$

Callaghan's theoretical representation of Magnetization decay can be used to verify the methods we used in our simulation in both 1-D and 3-D. Callaghan's approach was to use Monte Carlo distributions, similar to our approach, where he derived the demagnetization of spin states following an initial $\frac{\pi}{2}$ pulse. From this derivation, the signal attenuation due to diffusion is expressed by:[8]

$$\overline{e^{\iota\Delta\varphi}} = e^{-\frac{1}{3}\gamma^2 G^2 D t^3} \tag{10}$$

With this equation, we may determine the net magnetization at any time, t, given the diffusivity, D, and a magnetic field gradient, G. Callaghan also derived an equation which can determine the theoretical amplitude of the spin echo produced following a Carr-Purcell-Meiboom-Gill (CPMG) pulse-echo sequence. With pulsed magnetic field gradients of duration, δ, placed symmetrically before and after the π pulse, the recorded signal S(TE, G) can be expressed according to:

$$S(TE, G) = S(0,0) \cdot e^{\frac{-TE}{T2}} \cdot \left[1 - 2e^{-\frac{\left(TR - \frac{TE}{2}\right)}{T1}} + e^{-\frac{TR}{T1}}\right] \cdot e^{-\gamma^2 \delta^2 G^2 \left(\Delta - \frac{\delta}{3}\right) D} \tag{11}$$

Where S(0, 0) is the signal at TE=0, TR=∞, and TE=echo time, T1=longitudinal relaxation time, T2=transverse relaxation time. Rearranging this equation yields the theoretical echo amplitude, cancelling expressions for the T1 and T2 relaxation processes:

$$\frac{S(TE, G)}{S(TE, 0)} = e^{-\gamma^2 \delta^2 G^2 \left(\Delta - \frac{\delta}{3}\right) D} \tag{12}$$

If we consider a spherical nanoparticle with radius r_x and magnetic dipole moment $\vec{m}$ pinned in the z-axis by an external static field B_o, which is sufficiently strong so we can assume that self-interaction between nanoparticles is negligible, the *z*-component of the local magnetic field generated by each nanoparticle is:

$$B_z(r, \theta) = \frac{\mu_o m}{4\pi} \left(\frac{3\cos^2(\theta) - 1}{r^3} \right) \tag{13}$$

Where r is the position relative to the centre of the nanoparticle, μ_o is the permeability of free space and θ is the angle to the z-axis. The magnetic dipole moment $\vec{m}$ of a nanoparticle is defined as the torque it experiences in an external magnetic field. We can express this in terms of the external magnetic field B_o and the nanoparticles volume, V:

$$m = \chi_v \mathrm{B_o} V \tag{14}$$

Where the constant of proportionality χ_v is known as the volume magnetic susceptibility, dimensionless. The Larmor frequency due to the local magnetic field produced by the nanoparticle also becomes:

$$\omega(x) = \frac{\gamma(\mathrm{B_o} + B_z(r,\theta))}{2\pi} \qquad \{15\}$$

The contrast in images produced from Magnetic resonance are due to the relaxation times of the signal produced from a CPMG pulse-echo sequence. This time is referred to as the $\mathrm{T_2}$ relaxation time and can be manipulated by adjusting variables contributing to its magnitude. Following the spin echo sequence we have simulated, the $\mathrm{T_2}$ relaxation time can be calculated from the net magnetization and the echo time, TE:

$$\mathrm{M}_{xy}(t) = \mathrm{M}_{xy}(0)e^{-\frac{TE}{T_2}} \qquad \{16\}$$

From the boundary conditions used to confine the water molecules in a set region, the molar concentration (Molarity) of the nanoparticle can be expressed in terms of the volume of the sample size, V_b:

$$\mathrm{Molarity} = \frac{moles}{V_b} \qquad \{17\}$$

Where the number of moles can be expressed in terms of the mass of the nanoparticle, m, and the relative molecular mass (RMM) of the nanoparticle; Since we are using the SPIO nanoparticle of composition $\mathrm{Fe_3O_4}$, therefore its RMM is approximately 232 $\mathrm{gmol^{-1}}$:

$$moles = \frac{\mathrm{m}}{\mathrm{RMM}} = \frac{\rho V_p}{\mathrm{RMM}} \qquad \{18\}$$

Where ρ is the density of the SPIO nanoparticle, and V_p is the volume of the nanoparticle.

Method:

Our initial approach to this simulation, was to firstly figure out how to best code for a 1-D random walk to represent self-diffusing water molecules. We identified two approaches; A Monte Carlo method where we would use the inbuild 'rand(n)' function in MATLAB to randomly 'generate' a number in the interval (0,1) implying conditions within a 'IF' expression differentiating whether the particle will move in 1-D, left or right. This expression was set so that if rand(n) > 0.5, then the particles position, *x* becomes +1 otherwise, it is -1. The other approach was to use the 'randn(n)' function rather than 'rand(n)' since this will pull numbers from a gaussian distribution. Using this along with background theory about Brownian motion, we computed a function which selected a random number, which was scaled relative to a 'step size' given by equation {1}. This allowed a more realistic simulation of the self-diffusing water molecules since this step size is relative to the diffusivity. We selected this approach for our random walk.

Building on this, we needed to compute the walk for an ensemble of N particles. By defining a function 'diffusionwalk', we were able to optimise the time to calculate each random walk as well as keeping the script simple thus far. We defined new variables n, N and t being number of steps, number of particles and run time respectively. From these variables, we can compute a random walk over a set time t, with a time-dependent step size. For an ensemble of N particles, we used a 'For' loop where an array of dimensions (N,n); Used to store the output of our function 'diffusionwalk', was looped over N iterations corresponding to N random walks. To verify the random walk for an ensemble of N particles, we used equation {2}. Crank's equation theoretically represents the position distribution of N particles; If this plot encloses a histogram plot of the final positions of N particles, then our walk has been verified. The plot is shown in figure 1. Since both plots complement the each other the random walk holds and is theoretically verified.

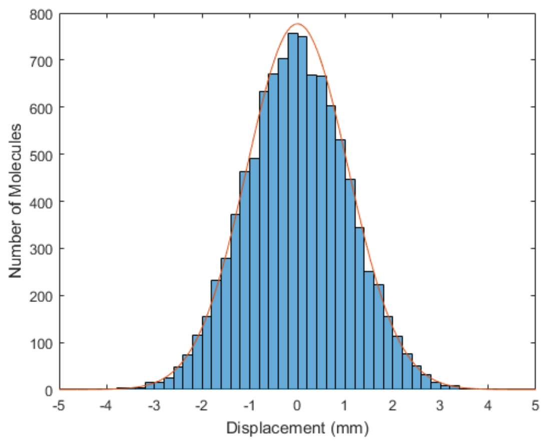

Fig.1: Plot of the distribution of final position for N particles enclosed in the theoretical distribution.

Since our random walk can run for any set time, t, we understood that eventually if the simulation was running for long enough, the particles will distribute themselves further and further away from the origin. To prevent this from happening, we introduced boundary conditions in the random walk. Again, there were two approaches we could've taken here; The first was, at a set boundary, *h*, the particles position would pass through the boundary and then appear at the opposite boundary, therefore its position will transition from, *h* to -*h*. The leftover 'step' would then be added / subtracted after this sign change to ensure the particle stays inside the boundary. The second approach was to introduce a reflection at the boundary. If the magnitude of the step size would cause the particles position to surpass the boundary, then the remaining 'step' could be added / subtracted from the boundary resulting in a reflection. Both approaches are very similar, however, we felt the best approach was a reflection at the boundary since this seemed more realistic as if the particle was enclosed in a box. Using equation {3} we can again verify the positional distribution of particles, but this time for a system enclosed in boundary conditions, $-h > x > h$.

Now we have written a script for the motion of N self-diffusing particles in 1-D, we want to observe how the net magnetization decays with time as well as producing a spin echo, following a CPMG pulse-echo sequence. We introduced a static field, B_o, which has linear field fluctuations, resulting in a field gradient, G, units Tm^{-1}. As explained in the theory, following a $\frac{\pi}{2}$ pulse, the spin states are aligned in the transverse plane, with initial magnetization, $M_o = 1$. We now needed to calculate the Larmor frequency, ω, and the phase angle for the progressive walk. Using equation {6} for the Larmor frequency, we can use values stored for the position of each random walk to calculate the Larmor frequency at every point. To simplify the calculations, as mentioned in the theory, we can assume the position of the rotating frame, ω_o thus we only have to consider contributions from the magnetic field gradient. Storing the Larmor frequencies in a separate (N,n) array, we calculated the relative phase angle at each position using equation {7}. Using MATLABs inbuild function 'cumsum()', over n steps, we calculated the progressive phase angle for each particle. Now we have an array which contains the phase angle over time for N-particles, we need to calculate the relative magnetization at each time step, Δt. Using equation {8}, we can split the phase angle into individual *x* and *y* components, and calculate the mean across N particles at each time interval. Using equation {9}, the net Magnetization can be found from $\overline{x_\varphi}$ and $\overline{y_\varphi}$ components, thus we can observe the magnetization decay by plotting this against time. Using equation {10}, we plotted the signal attenuation over our magnetization decay, to verify that our decay trend was correct theoretically. This is shown in figure (2). Here, we can see that the signal attenuation expression almost completely fits the decay produced from our simulation

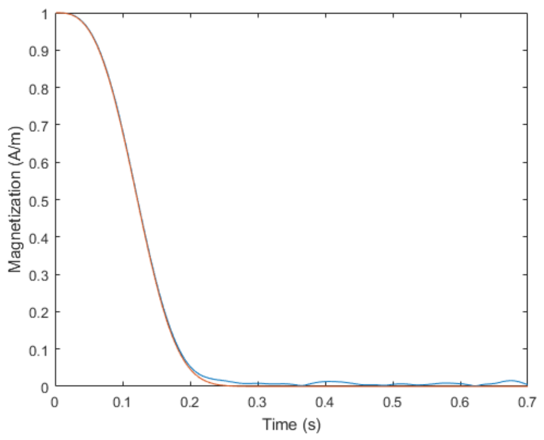

Fig.2: Plot of net magnetization against time to analyse the signal decay in comparison to the theoretical expression.

Since analysing how the magnetization decays, we can continue to follow the CPMG pulse-echo sequence to obtain a spin echo. (diagram) Following the initial $\frac{\pi}{2}$ pulse, causing spin states to lie in the transverse plane, we allow time, δ, for the particles to move in the magnetic field gradient, causing an initial decay in magnetization. After a later time, Δ, the system undergoes a π pulse, causing the spin states to flip 180° in the transverse plane. A time, δ, after this π-pulse, the particles are again moving randomly within the applied magnetic field gradient, when spin states realign, forming a spin echo. Figure (5) visually describes this sequence. Computing the magnetization during this process and plotting against time will allow us to analyse the amplitude of the spin echo as well at the echo time, TE.

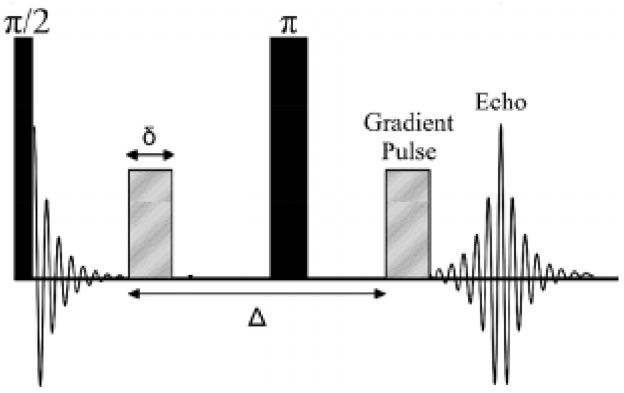

Fig.5: Diagram describing the pulse-echo sequence we followed where the labels: δ and Δ, are the times we are referring to in the

Using Callaghan's expression for the spin echo amplitude, equation {12}, and inputting variables used in our sequence, we can compare amplitude we obtained with the theoretical model. Figure (3) shows the spin echo amplitude for sequence times: δ = 0.1s and Δ = 0.4s, and field gradient, G = $1\,\mathrm{g/mm}$.
From this equation, we also investigated how the field gradient, G varied the magnitude of the spin echo and compared this with computed variations of G in our pulse-echo sequence. This is shown in Figure (4).

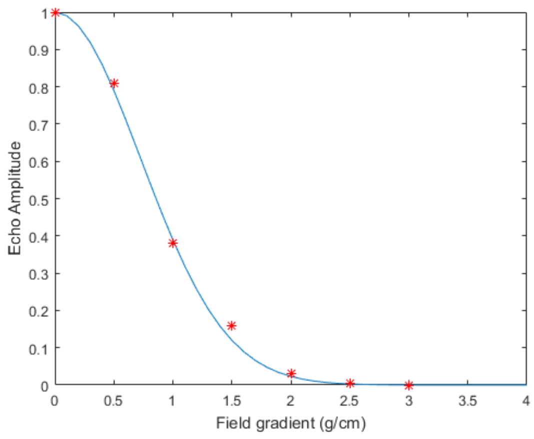

Fig.3: Plot of Echo amplitude against G to analyse how G varies the echo.

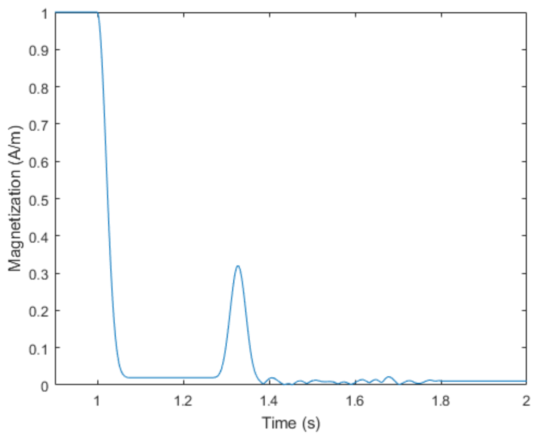

Fig.4: Plot of Magnetization against time obtaining a spin echo (1D).

Now we have computed and verified a 1-D simulation of self-diffusing water molecules exposed to a magnetic field gradient G, we can further the simulation to 3-D as well as implementing the nanoparticle. To expand the walk to 3-D we decided to produce three replica functions for individual *x*, *y* and *z* components like in 1-D, where the magnitude of the position, r, due to the origin was just the square sum of all components. Now implementing the nanoparticle; we modelled this to be pinned at the coordinate origin since particle positions can simply be taken as their coordinate position in space. From theory, an external magnetic field induces a local magnetic field on the nanoparticle given by equation {13}. The Larmor frequency can then be expressed as shown in equation {15}. Since the magnetic field $B_z(r,\theta)$ is inversely proportional to r^3, the water molecules need to be confined in tight boundary conditions for the local field to manipulate the phase angle like we saw with the linear gradient, G. The boundary conditions were set to

2.5μm to allow the nanoparticles to be of concentration 2.5mmolm^{-3} for sufficient field strength. Now following the pulse-echo sequence as we did in 1-D, we can complete a simulation to obtain a spin echo. This is shown in figure (6).

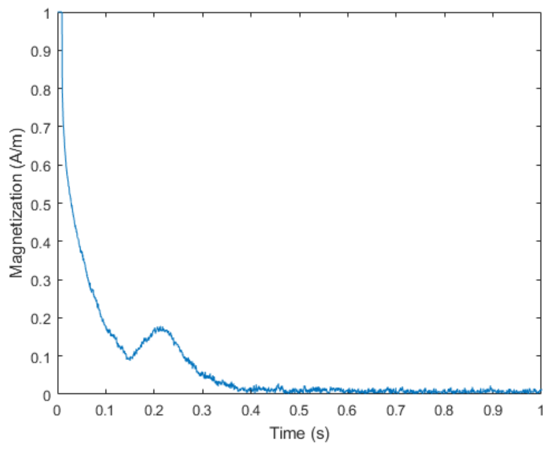

Fig.6: Plot of magnetization against time to observe the spin echo produced from the nanoparticle.

Analysing our simulation, we can produce a spin echo for any given medium given its properties are inputted into the scripts variables. From the echoes produced, we are able to calculate the T_2 relaxation time of the system. Using equation $\{16\}$, where $\mathrm{M}_{xy}(0)$ = 1, we can rearrange to calculate time, T_2. The relaxation time is a point of interest in MRI since is causes the contrast in the images produced. Understanding that T_2 time causes the contrast in MRI, we want to change variables used to produce the spin echo in our simulation and see how these effect the duration of the relaxation time. We decided to see how the variation in magnetic moment of the nanoparticle and how the concentration of the nanoparticle effected the T_2 relaxation time. Running simulations for both variables, we manually read off the values of TE and $\mathrm{M}_{xy}(t)$ and then plotted the corresponding relaxation time against each variable separately to analyse how they varied T_2.

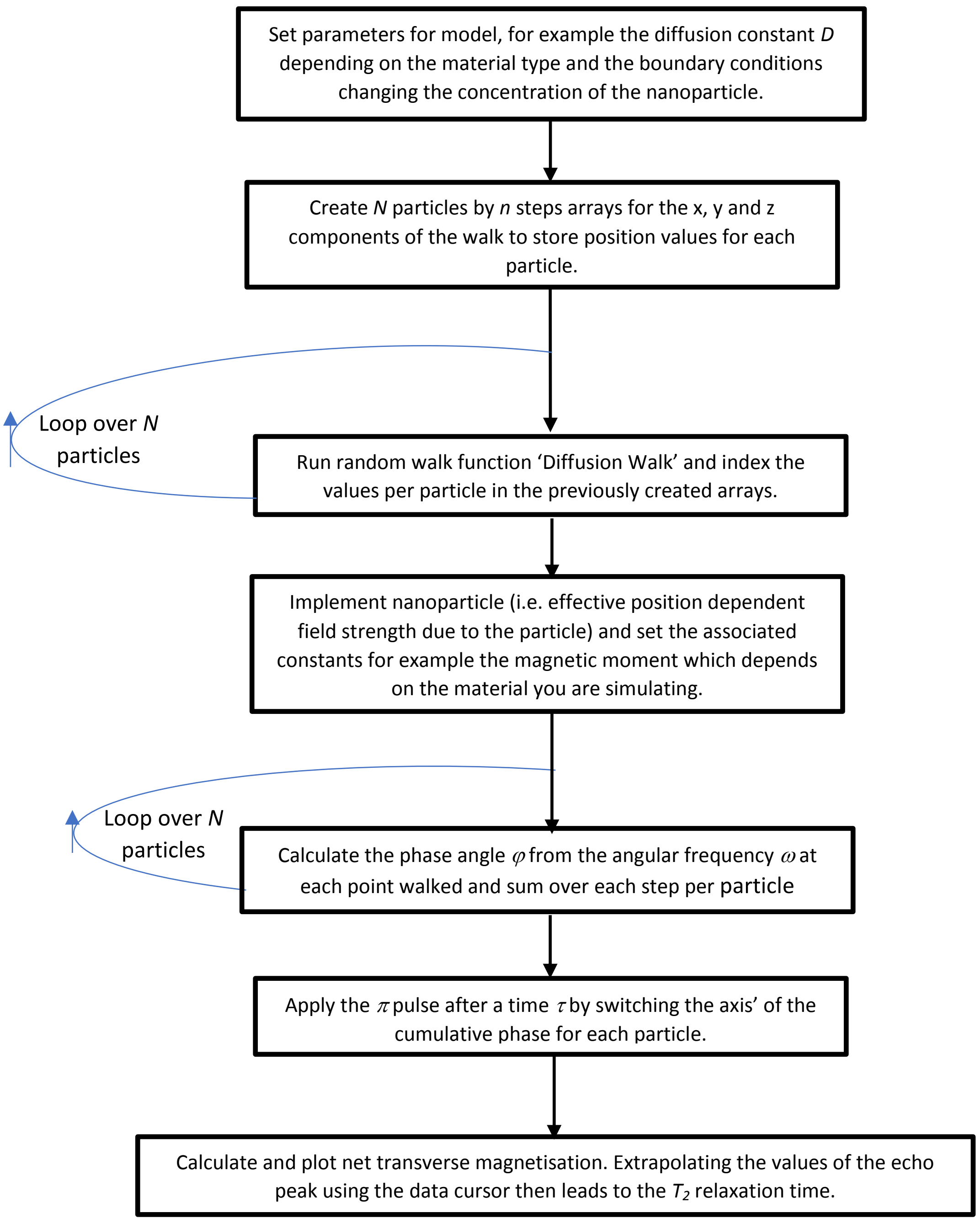
Set parameters for model, for example the diffusion constant *D* depending on the material type and the boundary conditions changing the concentration of the nanoparticle.
Create *N* particles by *n* steps arrays for the x, y and z components of the walk to store position values for each particle.
Loop over *N* particles
Run random walk function 'Diffusion Walk' and index the values per particle in the previously created arrays.
Implement nanoparticle (i.e. effective position dependent field strength due to the particle) and set the associated constants for example the magnetic moment which depends on the material you are simulating.
Loop over *N* particles
Calculate the phase angle φ from the angular frequency ω at each point walked and sum over each step per particle
Apply the π pulse after a time τ by switching the axis' of the cumulative phase for each particle.
Calculate and plot net transverse magnetisation. Extrapolating the values of the echo peak using the data cursor then leads to the T_2 relaxation time.

Fig.7: Flow diagram describing the logic used in our program.

Results:

Choosing to follow the CPMG pulse-echo sequence in our simulation allowed us to verify stages of our code to ensure that it was theoretically correct so it can be used as a model to compute relaxation times used for contrast images in MRI. All of our simulations were verified by analytical solutions defined in the theory.

Following variations in the magnetic moment and the concentration of the nanoparticles, we wanted to see how the T_2 relaxation time varied upon changes to these constants. We ranged the magnetic moment from $1\text{x}10^{-16}$ to $1\text{x}10^{-15}$ Am^2 and calculated the relaxation times using equation {16}. A plot of T_2 against magnetic moment is shown in figure (8). From the curve, we can see that increasing the magnetic moment causes a decrease in relaxation time.

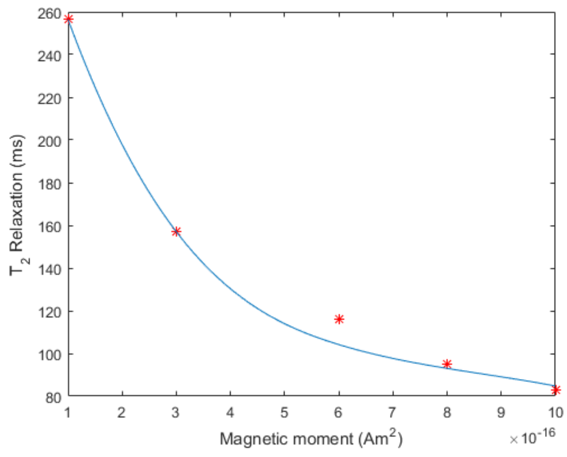

Fig.8: Plot of T_2 relaxation time against the magnetic moment, to observe how variations to this constant alter the contrast T_2.

We then decided to vary the concentration of nanoparticles within the simulation to see how this effects the contrast T_2. Varying the concentration from 1 – 7.5 mmolm^{-3} calculated using equations {17} and {18}, we observed the spin echo produced and obtained T_2 from equation {16}. The results are shown in figure (9). From the curve, we can see that increasing the concentration of the nanoparticle in the sample leads to a decrease in the relaxation time.

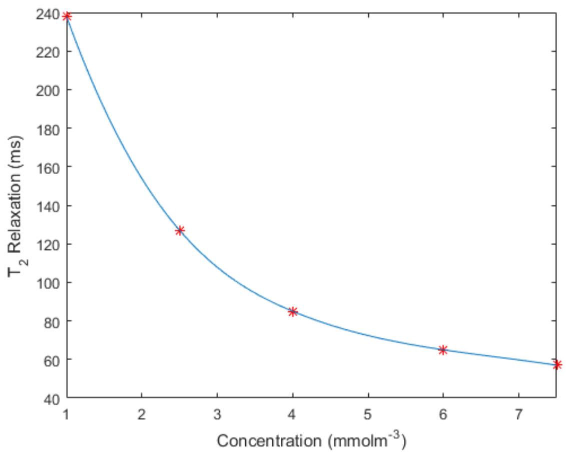

Fig.9: Plot of T_2 relaxation time against the concentration of the nanoparticle, to observe how variations to this constant alter the contrast T_2.

Discussion:

The 3-D simulation we have produced for a nanoparticle in a medium surrounded by self-diffusing water molecules allows us to successfully produce a spin echo, and allows us to calculate the T_2 relaxation time given all the constants of the system are inputted. Using this simulation, we can identify the ideal conditions of the system such as the concentration of nanoparticles and what type of nanoparticle to use, to produce a large contrast in imaging. Using analytical expressions to verify stages of our simulation allows for confidence in the results we obtain from the corresponding echo produced. However, since we haven't completely automated the simulation to output a single value of the T_2 relaxation time, we still must manually read off the values of TE and $M_{xy}(t)$ which can lead to errors in T_2. To reduce the error we obtain here, we could have implemented a line of code which differentiates the echo plot. Indexing this variable for a value equal to zero, we can locate the turning point, corresponding to the spin echo, then output values of TE and $M_{xy}(t)$ respectively. The error on T_2 would then be negligible and so, inputting these values into equation {16}, we can immediately obtain a value for T_2.

Using this simulation over experimental analysis would be much preferred since it would save time and money. Simply running a simulation rather than preparing samples and testing them for their respective T_2 values is very time saving and is much preferred when finding which best conditions to use for the best contrast in MRI. Optimization of our simulation would allow for faster T_2 findings and so our simulation would be far more efficient. To optimize our code, numerous measures were taken. One way was to reduce the number of logicals used. An example of this was calculating the Larmor frequency for N particles. We initially approached this running a 'For' loop over all N particles to find each relative Larmor frequency. Running this for a large number of particles, N, took a significant amount of time so instead, we decided to index the array so the frequencies can be calculated simultaneously, thus optimizing the calculation and reducing the run time.
Another way to reduce run time was to preallocate arrays before accessing them with functions or Logicals. An example is in our random walk. We initially define variables for the number of particles, N, and the number of steps, n, and so we preallocate an array of dimensions (N,n) using MATLABs 'zeros' function, allowing for it to be overwritten when calculating the positions of the random walks.

The results we obtained upon investigating how specific variables effect the T_2 relaxation time allows us to manipulate T_2 so we can produce images of large contrast. Both figures (7) and (8) indicate that increasing either the concentration of the nanoparticle or increasing the magnetic moment of the nanoparticle, leads to a shorter relaxation time. Therefore, in application, if we are looking to produce contrast images in a specific medium where the T_2 relaxation time is of a known value, we look to manipulate the relaxation time of the surrounding region by varying the properties of the nanoparticle and its concentration to increase the difference in T_2 times between the region targeted with nanoparticles and the surrounding regions. Increasing the difference in T_2 relaxation times will result in a larger contrast leading to better resolution images for improved diagnosis. To further verify our results, we compared the data plots we produced to data found experimentally.[9]

Improving the simulation, we would like to have increased the sample size of particles, N, which would allow for the simulation to be more realistic in an actual system since N=10000 particles is nowhere near the actual amount of water molecules in a biological system for the sample size we simulated. However, this value for N was sufficient for obtaining a spin echo but if we were to make the results for T_2 more reliable, an increase in sample size would definitely better the data displayed in our figures. If we were to extend the project, we would have liked to see how other factors varied the T_2 relaxation time, such as the time of the pulses used to manipulate the spin axis. We would also have liked to compare the data computed from our simulation with samples tested in a lab, to see how well our code determines T_2 and the relative error on it.

Conclusion:

Computing data from our simulation of nanoparticles in an external magnetic field, we managed to obtain T_2 relaxation times upon variation to properties of the nanoparticle and its concentration in the confined boundary. From the results, we identified both trends in T_2 relaxation time for a varied magnetic moment for the nanoparticle and a varied concentration. Both graphs tell us that an increase in magnetic moment or concentration, will correspond to a decreased T_2. For a higher resolution image, the contrast difference in relaxation times needs to be large, so depending on the region of interest, we can artificially produce nanoparticles which has a large contrast relative surrounding region, and calculate T_2 specifically from our simulation.

References:

[1] Weisman, I. D.; Bennett, L. H.; Maxwell, L. R.; Woods, M. W.; Burk, D. (1972-12-22). "Recognition of cancer in vivo by nuclear magnetic resonance". Science. **178** (4067): 1288–1290.

[2] Mansfield P; Grannell, P (1975). "Diffraction and microscopy in solids and liquids by NMR". Physical Review B. **12** (9): 3618–3634.

[3] Rinck PA (2014). "The history of MRI". Magnetic Resonance in Medicine (8th ed.).

[4] Lauterbur PC (1973). "Image Formation by Induced Local Interactions: Examples of Employing Nuclear Magnetic Resonance". Nature.

[5] Brian M. Tissue (1996). "Nuclear Magnetic Resonance (NMR) Spectroscopy". Technische Universitaet Braunschweig.

[6] Kleinberg, Robert L.; Jackson, Jasper A. (2001-01-01). "An introduction to the history of NMR well logging". Concepts in Magnetic Resonance. **13** (6): 340–342.

[7] Crank J. "The mathematics of diffusion", Clarendon press oxford.

[8] Callaghan, P. T. (1995). Principles of Nuclear Magnetic Resonance Microscopy, Oxford University Press.

[9] Bhattacharya K. "A report on the study of r1 and r2 relaxivities of magnetic nanostructures using a 9T NMR spectrometer". PHD placement report, University of Nottingham.

Appendices:

Matlab code for field gradient G:

```
clear all; clc;
N = input('Number of particles: ');% set 10000.
dT = input('Set time step frequency: ');% steps per second.
dt = 1/dT; % computing timestep.
t = input('Duration of walk: ');% How long you wish the walk to run for/s.
n =( t / dt)+1; % finding total steps for arrays.
d = input('Diffusion constant: '); % 0.282cm^2/s.
bound = input('Set boundary condition(+-): ');
m=1; % 1D
Bzero = 9.4; % Default external B field.
G = 1;% Field gradient G/cm.
gamma = 42.57;%(2pi)(MHzT-1)
%
% Random walk for set particles N, from diffusion function.
%
walk = zeros(N,n);
dum = 1;
for dum = 1:N
    walk(dum,:) = Diffusion1Dwalk(m,dt,d,t,bound);
end
histpoints = walk(:,n);
histogram(histpoints) %plotting histogram of walk.
hold on
% Proof using formula:
x = linspace(-5,5,1001);
c = ((2*N*((pi*d)^0.5))/(2*9.1*((pi*d*t)^0.5))).*(exp((-1.*(x.^2))/(4*d*t)));
plot(x,c)
xlabel('Displacement (mm)')
ylabel('Number of Molecules')
%
% end
%
%
%  Larmor frequency at each point in random walk.
%
figure
larmorfrequency = (-1.*gamma).*Bzero; % larmor frequency.
G = 1.5; % Field gradient set to 1 g/cm.
walklarmor =  gamma .* walk .* G; % Larmor Frequency @ each point - constant B field.
walkphase = (dt .* walklarmor); % relative phase at each point due to larmor frequency.
for L = 1:N % Loop to calculate progressive phase per particle.
    walkphase(L,:) = cumsum(walkphase(L,:));
end
XPHASE = cos(walkphase); % X component of phase.
YPHASE = sin(walkphase); % Y component of phase.
AVGX = mean(XPHASE);
AVGY = mean(YPHASE);
MAGXY = ( (AVGX.^2) + (AVGY.^2) ).^0.5; % Pythagoras to determine Magnetization.
timestep = [1:n] .* dt; % Time step array for plotting.
plot(timestep,MAGXY) % plot of Magnetization decay.
hold on
```

```
% Proof from Eq 3.50 of decay:
PHASEavg = exp((-1/3).*(gamma^2).*(G^2).*d.*(timestep.^3));
plot(timestep,PHASEavg)
xlabel('Time (s)')
ylabel('Magnetization (A/m)')
axis([0 0.7 0 1])
%
% end
%
%
% Spin echo
%
figure
Ninitial = 1.0 / dt; % initial positioning of particles.
noff = 1.07 / dt; % step number at which field is turned off.
non = 1.47 / dt; % step number at which field is turned back on after pi pulse.
npulse = ( non + noff )/2;
noff1 = 1.8 / dt; % step number at which field is turned off again.
walklarmor1 = zeros(N,n); %
XPHASE1 = zeros(N,n); % setup array of dimentions N,n.
YPHASE1 = zeros(N,n); %
walklarmor1(:,Ninitial:noff) = gamma .* walk(:,Ninitial:noff) .* G; % larmor frequency like
before, during the period of active B field.
% between noff and non, the field gradient g = 0 therefore larmor frequency
% is equal to 0.
walklarmor1(:,(non+1):noff1) =  gamma .* walk(:,(non+1):noff1) .* G; % larmor frequency when
G=1 again after pi pulse.
%after noff1, the field gradient G = 0 again therefore larmor freq = 0.
walkphase1 = (dt .* walklarmor1); % relative phase at each point.
for L1 = 1:N % Loop to find the progressive phase change per particle over time up to the pi
pulse.
    walkphase1(L1,1:npulse) = cumsum(walkphase1(L1,1:npulse));
end
XPHASE1(:,1:npulse) = cos(walkphase1(:,1:npulse));
YPHASE1(:,1:npulse) = sin(walkphase1(:,1:npulse));
walkphase1(:,npulse) = -1 * walkphase1(:,npulse); % Pi pulse - change sign.
for L2 = 1:N % Loop to find the progressive phase change per particle over time after pi
pulse.
    walkphase1(L2,npulse:n) = cumsum(walkphase1(L2,npulse:n));
end
XPHASE1(:,npulse:n) = cos(walkphase1(:,npulse:n));
YPHASE1(:,npulse:n) = sin(walkphase1(:,npulse:n));
AVGX1 = mean(XPHASE1);
AVGY1 = mean(YPHASE1);
MAGXY1 = ( (AVGX1.^2) + (AVGY1.^2) ).^0.5;

plot(timestep,MAGXY1) % plot of this new average phase against time.
axis([0.9 2.0 0 1])
xlabel('Time (s)')
ylabel('Magnetization (A/m)')
%
% end
%
% plot to see how field gradient effects the magnitude of the spin echo.
Lecho = linspace(0,4,41);
Techo = exp(-1*(gamma)^2 *([Lecho]).^2* ((noff-Ninitial)*dt)^2 * 0.282 * (((non-noff)*dt) -
((noff-Ninitial)*dt)/3 ));
plot(Lecho,Techo)
```

```
TechoSIM = [1,0.81,0.38,0.16,0.03,0.005,0];
Lecho1 = [0,0.5,1,1.5,2,2.5,3];
hold on
plot(Lecho1,TechoSIM,'r*')
xlabel('Field gradient (g/cm)')
ylabel('Echo Amplitude')
```

constants.

```
Hbar = 6.63e-34/(2*pi);
Gamma = 42.58; %MHz/T
Bzero = 9.4; %T
K = 1.38e-23; %J/K
Temperature = 310; %K
DeltaE = Hbar*Gamma*Bzero; %J
SpinR = exp(-DeltaE/(K*Temperature));
% SpinR = (DeltaE/(K*Temperature))*0.5*N;
```

Function code for random walk:

```
function x = Diffusionwalk1 (n, dt, D, Bound, WalkX1 )
%
%  Stepsize is normal.
%
  s = sqrt(2.0 * D * dt)  * randn ( 1, n - 1 );
%
%  Compute the individual steps.
%
  x = WalkX1;
%
%  Direction is random.
%
dx(1,1:n-1) = s(1:n-1);
%
%   Each position is the sum of the previous steps.
%
% x(1,2:n) = cumsum ( dx(1,1:n-1) ); %original code before
%
% Boundary condition.
%
    for dummy = 2:n
         x(1,dummy) = x(1,dummy-1) + dx(1,dummy-1);
       if (x(1,dummy)>Bound)
           x(1,dummy) = x(1,dummy)-(x(1,dummy)-Bound);
       elseif (x(1,dummy)<(-1*Bound))
           x(1,dummy) = x(1,dummy)-(x(1,dummy)+Bound);
       else
       end
    end
  return
```

3-D nanoparticle simulation (Final code):

```
%
% 3D Walk - NANO-particle present
%
clear all; clc; clf;
```

SET CONSTANTS:

```
N = 10000; % Number of particles.
F = 2000; % Frequency of steps /Hz.
dt = 1 / F; % Computing time step /s.
t = 3; % Duration of walk /s.
n =( t / dt)+1; % Calculating total steps.
D = 3.08e-14; % Diffusion constant /m^2/s.
Bound = 1.87e-6; % Boundary condition.
% boundary condition to be 1Mol/m^3 being volume of L = 2.5e-7m.
```

RANDOM WALK 3D:

```
WalkX = zeros(N,n);
WalkX1 = zeros(N,n);
WalkX1(1:N,1) = linspace(Bound,-Bound,N);
WalkY = zeros(N,n);
WalkY1 = zeros(N,n);
WalkY1(1:N,1) = linspace(Bound,-Bound,N);
WalkZ = zeros(N,n);
WalkZ1 = zeros(N,n);
WalkZ1(1:N,1) = linspace(Bound,-Bound,N);
for I = 1:N % X - component walk.
    WalkX(I,:) = Diffusionwalk1(n, dt, D, Bound, WalkX1(I,:));
end
for J = 1:N % Y - component walk.
    WalkY(J,:) = Diffusionwalk2(n, dt, D, Bound, WalkY1(J,:));
end
for K = 1:N % Z - component walk.
    WalkZ(K,:) = Diffusionwalk3(n, dt, D, Bound, WalkZ1(K,:));
end
WalkR = sqrt( (WalkX.^2) + (WalkY.^2) + (WalkZ.^2) ); % Radial Component.
WalkT = acos(WalkZ ./ WalkR); % Angle to Z-axis /RAD.
```

NANOPARTICLE:

```
Mx = 6e6; % Magnetic Susceptibility.
H = 9.4; % External field strength /T.
ND = 30e-9; % Diameter of Nanoparticle /m.
MAG = Mx * H; % Magnetization /A/m.
Volume = 1/6 * pi * ND^3; % Volume of Nanoparticle /m^3.
% MM = MAG * Volume; % 'Induced' Magnetization is defined as the dipole moment per unit volume
/Am^2.
MM = 5e-16;
u0 = 1.257e-6; % Permeability of free space /mkg/s^2A^2.
Bz = (( u0 * MM )/( 4 * pi )) * ((( 3 .* (cos(WalkT).^2) ) - 1 )./( WalkR.^3 )); % Magnetic
```

```
field /T.
Gamma = 267.513e6; % Gyromagnetic ratio /RAD/sT.
```

NANOPARTICLE B FIELD:

```
[Y,Z] = meshgrid(linspace(-1e-5,1e-5,500));
R = sqrt( (Y.^2) + (Z.^2) );
Thetacos = Z ./ R;
Bz1 = (( u0 * MM )/( 4 * pi )) * ((( 3 .* (Thetacos.^2) ) - 1 )./( R.^3 ));
contour3(Y,Z,Bz1,230)
hold on
% axis([ -1.5e-7 1.5e-7 -1.8e-7 1.8e-7])

% Plot single walk
%stem3(WalkX(1,:),WalkY(1,:),WalkZ(1,:))
```

Spin Decay:

```
figure
Walklarmor = Gamma .* Bz;
Walkphase = (dt .* Walklarmor(:,1:n)); % relative phase at each point due to larmor frequency.
for L = 1:N % Loop to calculate progressive phase per particle.
    Walkphase(L,:) = cumsum(Walkphase(L,:));
end
XPHASE = cos(Walkphase); % X component of phase.
YPHASE = sin(Walkphase); % Y component of phase.
AVGX = mean(XPHASE);
AVGY = mean(YPHASE);
MAGXY = ( (AVGX.^2) + (AVGY.^2) ).^0.5; % Pythagoras to determine Magnetization.
timestep = [1:n] .* dt; % Time step array for plotting.
plot(timestep,MAGXY) % plot of Magnetization decay.
```

Spin Echo

```
figure
Ninitial = 0.01 / dt; % initial positioning of particles.
noff = 0.15 / dt; % step number at which field is turned off.
non = 0.15 / dt; % step number at which field is turned back on after pi pulse.
npulse = ( non + noff )/2;
noff1 = 1.50 / dt; % step number at which field is turned off again.
Walklarmor1 = zeros(N,n); %
XPHASE1 = zeros(N,n); % setup array of dimentions N,n.
YPHASE1 = zeros(N,n); %
Walklarmor1(:,Ninitial:noff) = Gamma .* Bz(:,Ninitial:noff); % larmor frequency like before,
during the period of active B field.
% between noff and non, the field gradient g = 0 therefore larmor frequency
% is equal to 0.
Walklarmor1(:,(non+1):noff1) =  Gamma .* Bz(:,(non+1):noff1); % larmor frequency when G=1
again after pi pulse.
%after noff1, the field gradient G = 0 again therefore larmor freq = 0.
Walkphase1 = (dt .* Walklarmor1); % relative phase at each point.
for L1 = 1:N % Loop to find the progressive phase change per particle over time up to the pi
pulse.
    Walkphase1(L1,1:npulse) = cumsum(Walkphase1(L1,1:npulse));
end
```

```
XPHASE1(:,1:npulse) = cos(Walkphase1(:,1:npulse));
YPHASE1(:,1:npulse) = sin(Walkphase1(:,1:npulse));
Walkphase1(:,npulse) = -1 * Walkphase1(:,npulse); % Pi pulse - change sign.
for L2 = 1:N % Loop to find the progressive phase change per particle over time after pi
pulse.
    Walkphase1(L2,npulse:n) = cumsum(Walkphase1(L2,npulse:n));
end
XPHASE1(:,npulse:n) = cos(Walkphase1(:,npulse:n));
YPHASE1(:,npulse:n) = sin(Walkphase1(:,npulse:n));
AVGX1 = mean(XPHASE1);
AVGY1 = mean(YPHASE1);
MAGXY1 = ( (AVGX1.^2) + (AVGY1.^2) ).^0.5;
timestep = [1:n] .* dt; % Time step array for plotting.

plot(timestep,MAGXY1) % plot of this new average phase against time.
xlabel('Time (s)')
ylabel('Magnetization (A/m)')
axis([0 1 0 1])
```

magnetic moment variation:

```
Mmoment = [1e-16, 3e-16, 6e-16, 8e-16, 1e-15];%values read from simulations.
trelax =1e3.* 10./[38.99, 63.69, 96.01, 107.52, 117.98];
trelax1 =1e3.* 10./[38.99, 63.69, 86.01, 105.52, 120.98];
plot(Mmoment, trelax1,'r*')
xlabel('Magnetic moment (Am^2)')
ylabel('T_2 Relaxation (ms)')
hold on
fitobject=linspace(min(Mmoment), max(Mmoment));
fit1 = interp1(Mmoment, trelax, fitobject, 'spline');
plot(fitobject, fit1)
```

Concentration variation:

```
ConcentrationT2 = [1, 2.5, 4, 6, 7.5]; %values read from simulations.
trelax2 = 1e3.*[0.238, 0.127, 0.085, 0.065, 0.057];
plot(ConcentrationT2,trelax2,'r*')
xlabel('Concentration (mmolm^-^3)')
ylabel('T_2 Relaxation (ms)')
hold on
fitobject2=linspace(min(ConcentrationT2), max(ConcentrationT2));
fit12 = interp1(ConcentrationT2, trelax2, fitobject2, 'spline');
plot(fitobject2, fit12)
axis([1 7.5 40 240])
```