Pulsed Nuclear Magnetic Resonance (NMR)

Keenan McConkey 21337167

Toolbox Dependencies

Note that sections of the notebook will not run without the Curve Fitting Toolbox!

Run This Cell First!

Make sure to run this cell to cd into directory, all data extraction code assumes you are in a directory already!

Assuming you are already in the working directory of this file...

```
%cd 'Task 2' % Already change to a folder so future 'cd ..' will put us in home directory
set(groot, 'DefaultAxesFontSize', 14);
set(groot, 'DefaultLineLineWidth', 1.2);
set(0, 'DefaultTextInterpreter', 'latex')
set(0, 'DefaultLegendInterpreter', 'latex')
set(0, 'DefaultAxesTickLabelInterpreter', 'latex')
```

Task 1v1 - Precession of a Single Proton

Goal

This is my first attempt at simulating the free precession of a magnetic moment in a permanent external magnetic field.

```
clearvars;      % Clean environment
```

Define Physical Parameters

Using a classical approach, we derive that the motion of a magnetic moment in an applied field in thermal equilibrium is described by the following ordinary differential equation (ODE):

$$\frac{\overrightarrow{\mathrm{dM}}}{\mathrm{dt}} = \gamma \cdot \vec{M} \times \vec{B} \qquad (1)$$

Where $\vec{M}$ is the magnetic moment, $?$ is the gyromagnetic ratio, and $\vec{B}$ is the applied magnetic field. Thermal equlibrium implies there is no magnetization in the xy plane.

In this simulation, we simplify things by only considering a single hydrogen nucleus, i.e. proton, and its corresponding magnetic moment from its spin (sometimes $?$ is used to denote this single proton moment but we just use M here). The external magnetic field is assumed to be constant and applied in only the positive $\hat{z}$ direction, i.e.

$$\vec{B} = B_0 \cdot \hat{z}$$

These physical parameters with corresponding variable names and physical units are listed:

- `gam`: Gyromagnetic ratio [rad / s * T]
- `Bo`: Applied magnetic field strength (scalar) [T]

The numerical values of these parameters:

```
gamma = 2.675e8;        % Gyromagnetic ratio of a proton [rad/s*T]
Bo    = 0.88;           % Applied magnetic field strength [T]
```

We then simply define our $\vec{B}$ vector as a 3 dimensional column vector, with $B_z = B_0$:

```
B = [0.0; 0.0; Bo];     % Applied magnetic field vector [T] (3D column vector)
```

We can also calculate an expected precession frequency of the magnetic moment, ω_0, from the resonant frequency of the proton's magnetic states (this can be worked out using an energy-based approach). We find that ω_0 is simply given by:

$$\omega_0 = \gamma B_0$$

We call this resonant frequency `omega0` ([rad/s]), and solve for it:

```
omega0 = gamma * Bo     % Resonance frequency, i.e. expected precession frequency [rad/s]
```

```
omega0 = 235400000
```

And find $\omega_0 = 2.354 \times 10^8 \ \frac{\mathrm{rad}}{s} = 4.034 \times 10^7 \ \frac{\mathrm{rad}}{s}$

Define Numerical Simulation Parameters

Since the expected precession frequency is on the order of $10^7 \ \mathrm{Hz}$ (i.e a period on the order of $10^{-7} \ s$), we use a time scale on the order of $10^{-7} \ s$, giving us the following numerical simulation parameters:

```
dt = 1e-9;                  % Numerical time resolution [s]
Dt = 2e-7;                  % Numerical time window [s]
tspan = (0.0:dt:Dt);        % Time scale
```

Define the Initial Conditions of the ODE

Since we can't assume that the proton has any initial fixed magnetization when it enters the B field, we have some flexibility when choosing its initial magnetization.

For simplicity we will use Bohr magneton units, which allow us to operate on the order of 10^0. We use the following vector initial conditions for $M_i = \vec{M}(0)$ to produce similar results to Figure 1.9 (a) in the instruction manual.

```
Mi = [0.35; 0.55; 0.75];     % Magnetic moment initial conditions as a vector [Bohr magnetons]
```

Solve the ODE

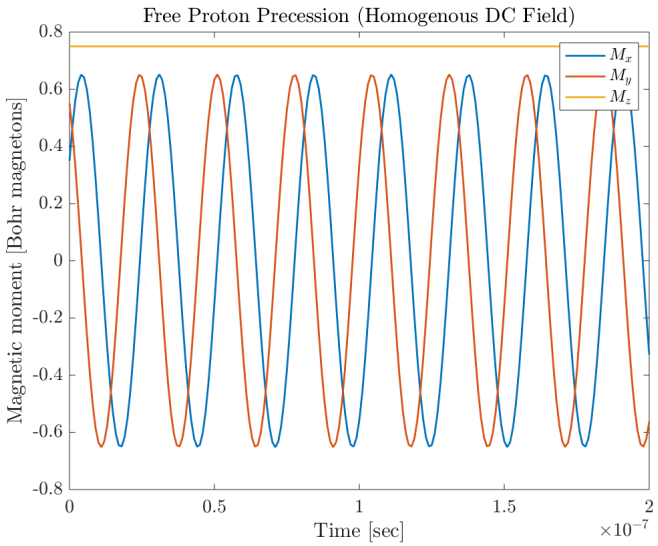
We can now solve the differential equation (1) numerically using an ODE solver. To begin we define the function handle for the ODE, `nmr_ode1` as a function of symbolic values time t and magnetic moment M:

```
nmr_ode1 = @(t, M) gamma * cross(M, B')';   % NMR ODE equation (1)
```

Note that t is not actually used in this function handle but we'll leave it in for consistency with the more complicated ODE function handles we will use in the future.

We are now ready to numerically integrate the ODE with `ode45`, which takes the ODE function `nmr_ode1`, time scale `tspan`, and initial conditions for magnetic moment `Mi` as inputs, and plot our results:

```
% Integrate nmr_ode1 using ODE 45
[tout, Mout] = ode45(@(t,M) gamma * cross(M,B')', tspan, Mi);

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
title('Free Proton Precession (Homogenous DC Field)')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('$M_x$', '$M_y$', '$M_z$');
```

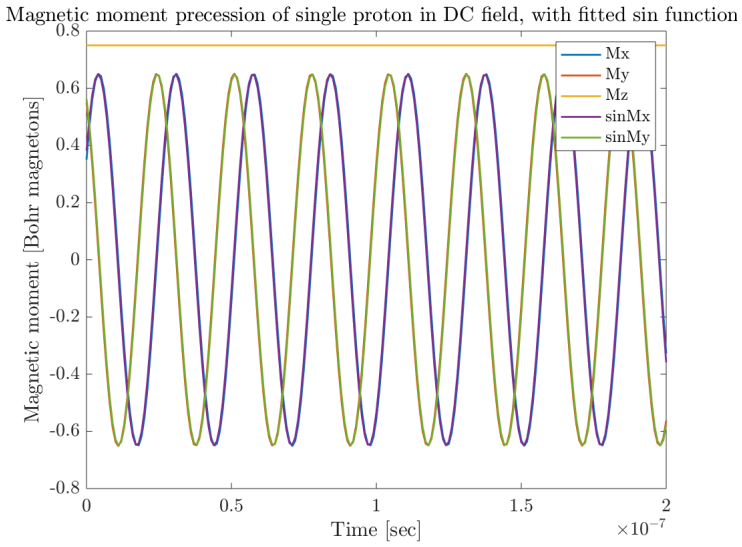


This looks quite close to Figure 1.9 (a) in the instruction manual. We see that the M_z component (along the direction of the field) of the magnetic moment is clearly preserved at a value of 0.75 Bohr magnetons. We now just need to fit a $\sin$ function to each of the oscillating M_x and M_y components to extract their frequency of procession, confirming that they are oscillating at the expected frequency of ω_0.

We introduce the following two fitted $\sin$ function handles, where the amplitude and phase values have been determine by trial and erorr, and plot them over the previous graph:

```
% Fitted sin functions for Mx and My
% Amplitude and phase determined by trial and error
sinMx = @(t) 0.65 * sin(omega0 * t + pi/5);
sinMy = @(t) 0.65 * sin(omega0 * t + 2*pi/3);

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
plot(tout, sinMx(tout));
hold on;
```

```
plot(tout, sinMy(tout));
hold on;
title('Magnetic moment precession of single proton in DC field, with fitted sin functions')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz', 'sinMx', 'sinMy');
```



Conclusion

We see that the fitted sin functions fit nearly exactly over the M_x and M_y precession curves once we have calibrated their ampltude and phase, confirming that they oscillate at as expected at resonance frequency ω_0. We determined M_x and M_y to have to have a phase components of $\frac{\pi}{5}$ and $\frac{2\pi}{3}$ radians respectively, giving a phase difference between the two of $\frac{7}{15}\pi$ radians We also conclude that the M_z component remains undisturbed in the field when we do not include spin-lattice interaction in our model.

Task 1v2 - Introducing Spin-Latttice and Spin-Spin Decays

Goal

Here I begin to experiment with including the T_1 and T_2 decays in solving the magnetic moment differential equations. Here I only want to show that the I can modify the equations to include both decays and still get a result that makes sense.

```
clearvars;              % Clean environment
```

Define Physical Parameters

We now have to modify our ODE equation (1) from **Task 1v1** to include both the spin-lattice relaxation time T_1 and the spin-spin relaxation time T_2. Equation (1) is restated here:

$$\frac{\overrightarrow{\mathrm{dM}}}{\mathrm{dt}} = \gamma \cdot \overrightarrow{M} \times \overrightarrow{B} \qquad (1)$$

We begin by carrying over the physical parameters from **Task 1v1**, listed below. For now, we carry over the stipulation that $\overrightarrow{B}$ only acts in the $\hat{z}$ direction.

- `gam`: Gyromagnetic ratio [rad / s * T]
- `Bo`: Applied magnetic field strength [T] (scalar)
- `B`: Applied magnetic field vector [T] (3D column vector)

The numerical values of these parameters:

```
gamma = 2.675e8;        % Gyromagnetic ratio of a proton [rad/s*T]
Bo    = 0.88;           % Applied magnetic field strength [T]
B = [0.0; 0.0; Bo];     % Applied magnetic field vector [T] (3D column vector)
```

Define Numerical Simulation Parameters

Since our ω_0 is the same as in Task 1v1 and we are just showing that the integration result makes sense, we'll leave numerical parameters the same for now. They are defined again below:

```
dt = 1e-9;                  % Numerical time resolution [s]
Dt = 2e-7;                  % Numerical time window [s]
tspan = (0.0:dt:Dt);        % Time scale
```

Define the Initial Conditions of the ODE

Again we can't assume the proton has any fixed magnetization when it enters the magnetic field. The initial conditions we used in Task 1v1 worked well, and since we are only showing that our integration result makes sense, we can reuse them here. Again we use units of Bohr magnetons.

```
Mi = [0.35; 0.55; 0.75];     % Magnetic moment initial conditions as a vector [Bohr magnetons]
```

Define Parameters of the New ODE

Spin-lattice relaxation time T_1 governs the exponential growth of M_z towards equilbirum when placed in a magnetic field or displaced from equilibrium. To model this growth we modify equation (1) to include another term for the time derivative of M_z:

$$\frac{dM_z(t)}{dt} = \frac{M_o - M_z(t)}{T_1}$$

This is equation requires introducing a new parameter M_o, the equilbrium value of M_z. We can choose this somewhat arbitrarily, but for now we will make it the same as the initial condition of M_z, i.e. $M_z(0)$.

We define M_o as a vector $\vec{M}_o$, which captures the subtraction of M_o from M_z:

$$\overrightarrow{M_0} = \begin{bmatrix} 0 \\ 0 \\ M_o \end{bmatrix}$$

Numerically, we rename $\vec{M}_o$ as `Mo` [Bohr magnetons] and define it:

```
Mo = [0; 0; 0.75];      % Vector capturing subraction of equilbrium value from moment in z [Bohr magnetons]
```

Meanwhile, spin-spin relxation time T_2 models the situation where we have magnetic moment rotating in phase in the xy plane. We will see why we need to model T_2 when we use $\pi/2$ to flip the magnetic moment into the xy plane. Since the situation is a combination of spin-up and spin-up states, T_2 models the exponential decay as the spin states interact and cancel out over time.

$$\frac{dM_{xy}}{dt} = -\frac{M_{xy}(t)}{T_2}$$

These two new parameters are summarized here:

- `T1`: Spin-lattice relaxation time [s]
- `T2`: Spin-spin relaxation time [s]

We are given test values of $T_1 \approx 5\,\mu s$ and $T_2 \approx 1\mu s$ for these two parameters, representing an imaginary material. Note that these values are different what we expect with numerical data.

```
T1 = 5e-6;      % Spin-lattice relaxation time [s]
T2 = 1e-6;      % Spin-spin relaxation time [s]
```

Putting this all together, he components of the new ODE equations including time constants T_1 and T_2 are:

$$\frac{\mathrm{dM}_x(t)}{\mathrm{dt}} = \left(\gamma\vec{M} \times \vec{B}\right)_x - \frac{M_x(t)}{T_2}, \qquad \frac{\mathrm{dM}_y(t)}{\mathrm{dt}} = \left(\gamma\vec{M} \times \vec{B}\right)_y - \frac{M_y(t)}{T_2}, \qquad \frac{\mathrm{dM}_z(t)}{\mathrm{dt}} = \left(\gamma\vec{M} \times \vec{B}\right)_z - \frac{M_z(t) - M_o}{T_1}$$

We introduce a 3x3 matrix T_{inv} to represent the multiplication of each term of $\vec{M}$ by its corresponding T_1 and T_2 terms:

$$T_{\mathrm{inv}} = \begin{bmatrix} T_2^{-1} & 0 & 0 \\ 0 & T_2^{-1} & 0 \\ 0 & 0 & T_1^{-1} \end{bmatrix}$$

Numerically, `Tinv` is:

```
Tinv = [1/T2 0 0; 0 1/T2 0 ; 0 0 1/T1]; % Multiplies terms of M by repsective inverse time constant
```

Solve the New ODE

We can now reduce the 3 equations for time derivative of M in each direction to a single ODE using a matrix product and subtraction:

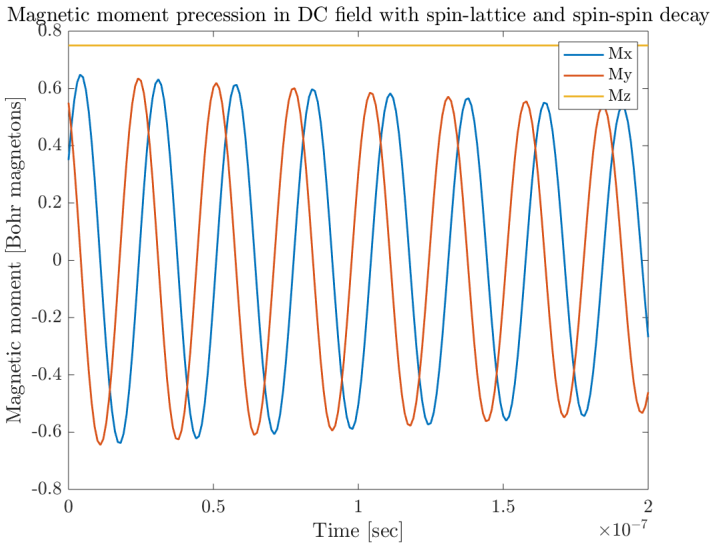
$$\frac{d\vec{M}(t)}{\mathrm{dt}} = \gamma\,\vec{M} \times \vec{B} - T_{\mathrm{inv}}\left(\vec{M} - \overrightarrow{M_0}\right) \qquad (2)$$

Putting this into function handle form:

```
nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo); % NMR ODE equation (2)
```

We are now ready to integrate `nmr_ode2` using `ode45` and plot:

```
% Integrate nmr_ode2 using ODE 45
[tout, Mout] = ode45(nmr_ode2, tspan, Mi);

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
title('Magnetic moment precession in DC field with spin-lattice and spin-spin decay')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz');
```



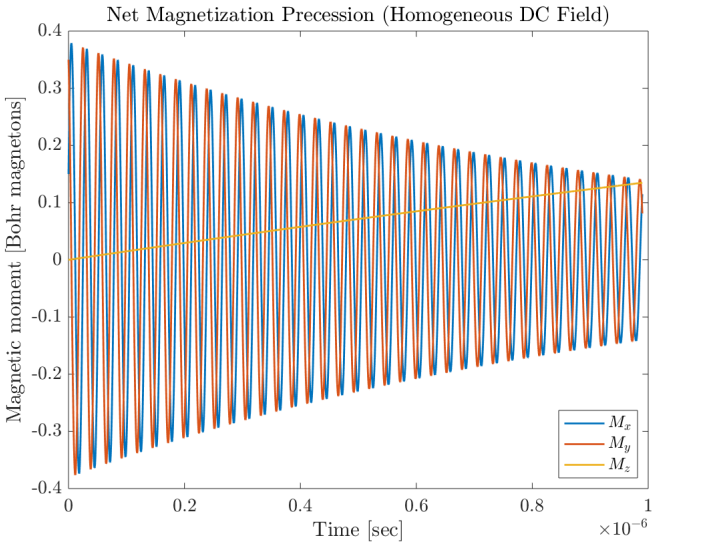
Showing M_z Returns to Equilibrium

We should also show that the M_z component will grow exponentially towards its equilibrium values. To do this, we set the intial value of $M_z(0)$ (i.e. the third vector value of $\vec{M}_i$ to 0.0 and increase the time window by two orders of magnitude to show that M_z eventually reaches the equilibrium value of M_o, which we set to 0.75 Bohr magnetons:

```
Mi = [0.15; 0.35; 0];     % Magnetic moment initial conditions with Mz(0)=0 [Bohr magnetons]
dt = 1.99e-10;                  % Numerical time resolution [s]
Dt = 0.99e-6;                 % Numerical time window [s]
tspan = (0.0:dt:Dt);          % Time scale
```

We now redo the integration:

```
% Integrate nmr_ode2 using ODE 45 with the new initial conditions and time
[tout, Mout] = ode45(nmr_ode2, tspan, Mi);

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
title('Net Magnetization Precession (Homogeneous DC Field)')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('$M_x$', '$M_y$', '$M_z$', 'Location', 'southeast');
```



Conclusion

The results of both integrations make sense. When we include spin-lattice and spin-spin decays T_1 and T_2 in our magnetic moment ODE, both M_x and M_y components now oscillate and decay exponentially over time to due spin-spin interaction modelling. We also saw that if the initial value of M_z component is the same as its equilbrium value M_o, it remains constant, while If M_z is not at equilbrium, it decays exponentially. We expect the time constant of this decay to be T_1.

Task 1v3 Introducing the $\pi \longrightarrow \frac{\pi}{2}$ Pulse

Goal

We now introduce the $\pi \longrightarrow \frac{\pi}{2}$ pulse to first flip the magnetic field and then rotate it into the xy plane.

```
clearvars;              % Clean environment
```

Define Physical Parameters

We use the same physical parameters from **Task 1v2**, listed below.

- `gam`: Gyromagnetic ratio [rad / s * T]
- `Bo`: Applied magnetic field strength [T]
- `B`: Applied magnetic field vector [T] (3x1 column vector)
- Mo: Vector capturing subtraction of equilbrium value from moment in z [Bohr magnetons] (3x1 column vector)
- `omega0`: Resonance frequency, i.e. expected precession frequency [rad/s]
- `T1`: Spin-lattice relaxation time [s]
- `T2`: Spin-spin relaxation time [s]
- `Tinv`: Array to multiply terms of M by respective inverse time constant [1/s] (3x3 array)

The numerical values of these parameters:

```
gamma = 2.675e8;        % Gyromagnetic ratio of a proton [rad/s*T]
Bo    = 0.88;           % Applied magnetic field strength [T]
B = [0.0; 0.0; Bo];     % Applied magnetic field vector [T] (3D column vector)
omega0 = gamma * Bo;     % Resonance frequency, i.e. expected precession frequency [rad/s]
Mo = [0; 0; 1.0];       % Vector capturing subtraction of equilbrium value from moment in z [Bohr magnetons] (3D column vector
T1 = 5e-6;              % Spin-lattice relaxation time [s]
T2 = 1e-6;              % Spin-spin relaxation time [s]
Tinv = [1/T2 0 0; 0 1/T2 0 ; 0 0 1/T1]; % Array to multiply terms of M by respective inverse time constant [1/s] (3x3 array)
```

We now introduce the concept of using a radio frequency (RF) magnetic field to rotate the moment M as desired. If we add our original DC field to a rotating (circularly polarized) magnetic field B_1, our total field in the rotating coordinates of the field is given by:

$$\vec{B}^*_{eff} = B_1\hat{x}^* + \left(B_o - \frac{\omega}{\gamma}\right)\hat{z}^*$$

If the added field is rotating at ω_0 (i.e. the RF magnetic field has frequency ω_0) then the second term goes to zero and:

$$\vec{B}^*_{eff} = B_1\hat{x}^*$$

We define the following new parameters for this new magnetic field:

- `B1`: Applied magnetic field strength [T]
- `Beff`: Effective magnetic field vector in rotating coordinates [T] (3D column vector)

```
B1   = 0.88;            % Rotating magnetic field strength [T]
Beff  = [B1; 0.0; 0.0];  % Effective magnetic field vector in rotating coordinates [T](3D column vector)
```

From equation (1) the new ODE is simply:

$$\frac{d\vec{M}}{dt} = \gamma\vec{M} \times \vec{B}^*_{eff} = \gamma\vec{M} \times \vec{B}_1 \quad (3)$$

The angle of rotation caused by B_1 is determined by the pulse length it is on for. If we turn off B_1 as soon as M aligns with xy plane, we have a coherent magnetization entirely in the xy plane M_{xy}. If we leave it on twice as long, M is flipped all the way around into $-\hat{z}$. We define t_{pulse} [s] as the pulse length required to rotate M by $?$ radians into $-\hat{z}$. Clearly then the pulse length required to put it into xy is just $t_{pulse}/2$. This t_{pulse} was determined by trial and error in the simulation for the given B_o until M was flipped exactly into $-\hat{z}$:

```
tpulse = 0.01333e-6;   % Pulse length required to rotate by radians [s], determined by trial and error
```

Define Numerical Simulation Parameters

We increase the numerical time resolution `dt` and time window `Dt` since we needer a larger window to see full effects

```
dt = 1e-10;             % Numerical time resolution [s]
Dt = 1.0e-6;            % Numerical time window [s]
```

We also introduce two new times to capture the dynamics of the new pulses, listed below:

- `tpi`: Starting time of the $?$ pulse, i.e. the first pulse [s]
- `tau`: Time after the $?$ pulse to start the $\pi/2$ pulse [s]

These can be varied to produce different results. For now define them numerically as:

```
tpi = 0.1e-6;          % Starting point of pi pulse, i.e the first pulse [s]
tau = 0.4e-6;          % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
```

We now have to introduce **5** different time windows to capture the 5 different regions of pulse dynamics:

- `tspan1`: Time window before any pulse dynamics
- `tspan2`: Time window during which the $?$ pulse RF field is turned on
- `tspan3`: Time window after the $?$pulse and before the $\pi/2$ pulse
- `tspan4`: Time window during which the $\pi/2$ pulse RF field is turned on
- `tpsan5`: Time window after both pulses

```
tspan1 = (0 : dt : tpi);                          % Time window before any pulse dynamics
tspan2 = (tpi : dt : (tpi+tpulse));               % Time window during which the  pulse RF field is turned on
tspan3 = ((tpi+tpulse) : dt : (tpi+tau));         % Time window after the pulse and before the  pulse
tspan4 = ((tpi+tau) : dt : (tpi+tau+tpulse/2));   % Time window during which the  pulse RF field is turned on
tspan5 = ((tpi+tau+tpulse/2) : dt : Dt);          % Time window after both pulses
```

Initial Conditions of the ODE

To model a magnetic moment that is initially entirely in the $\hat{z}$ direction, we simply define initial magnetization M_i as:

```
Mi = [0.0; 0.0; 1.0];  % Magnetic moment initial conditions with M entirely in z [Bohr magnetons]
```

Integrating the New ODE

We define the ODE equation (3) symbolically:

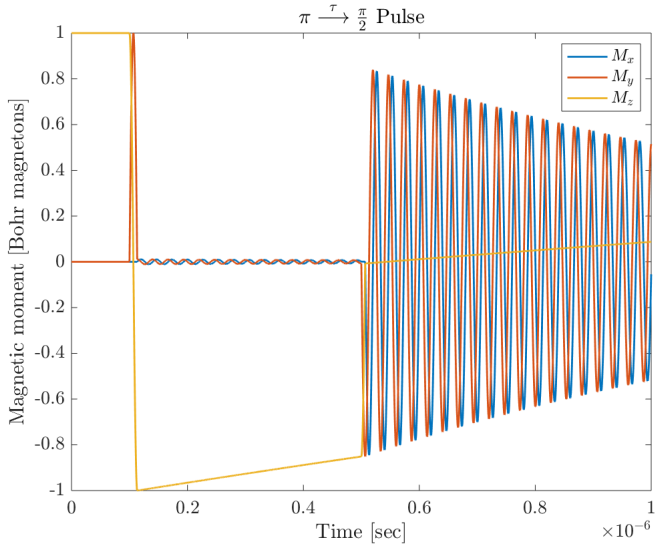
```
nmr_ode3 = @(t, M) gamma * cross(M, Beff')'; % NMR ODE equation (3)
```

We also need to use the ODE equation (2) from **Task 1v2**:

```
nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo); % NMR ODE equation (2)
```

Based on the pulse dynamics described, we integrate equation (2) in time windows 1, 3 and 5 and equation (3) in time windows 2, and 4. At each new integration, we use the end results of the last integration as the initial conditions, and then combine results.

```
% Integrate results in each time window
[t1, M1] = ode45(nmr_ode2, tspan1, Mi);
[t2, M2] = ode45(nmr_ode3, tspan2, M1(end, :));
[t3, M3] = ode45(nmr_ode2, tspan3, M2(end, :));
[t4, M4] = ode45(nmr_ode3, tspan4, M3(end, :));
[t5, M5] = ode45(nmr_ode2, tspan5, M4(end, :));

% Combine results of all time windows
tout = [t1; t2(2:end); t3(2:end); t4(2:end); t5(2:end)];
Mout = [M1; M2(2:end, :); M3(2:end, :); M4(2:end, :); M5(2:end, :)];

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
title('$\pi \stackrel{\tau}{\longrightarrow} \frac{\pi}{2}$ Pulse')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('$M_x$', '$M_y$', '$M_z$');
```



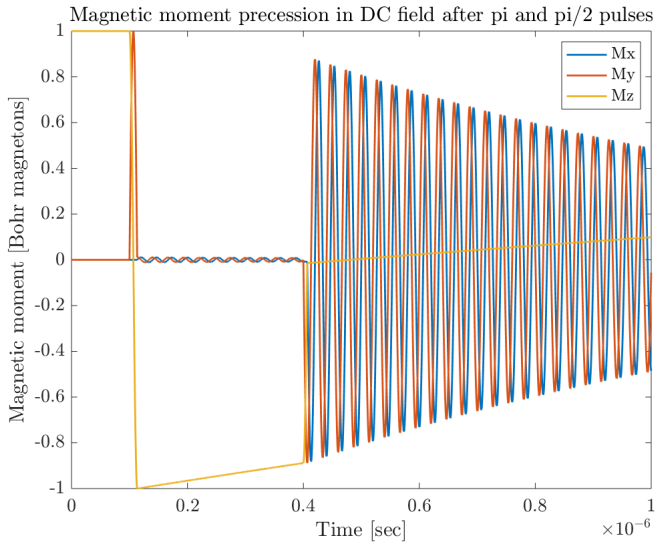
Reovering T_1 from M_{xy}

With the simulation looking good, we now want to confirm we can recover T_1 numerically. To do this, we vary *?* to affect the initial (i.e. max) amplitude of M_{xy}. We can just choose two values of *?* and extract the time decay from the different in amplitude.

Simulating $\tau = 0.3$ *?*s

We redefine *?* as well as the relevant time windows.

```
tau1 = 0.3e-6;         % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
tspan3 = ((tpi+tpulse) : dt : (tpi+tau1));           % Time window after the pulse and before the  pulse
tspan4 = ((tpi+tau1) : dt : (tpi+tau1+tpulse/2));     % Time window during which the  pulse RF field is turned on
tspan5 = ((tpi+tau1+tpulse/2) : dt : Dt);            % Time window after both pulses

% Integrate results in each time window
[t1, M1] = ode45(nmr_ode2, tspan1, Mi);
[t2, M2] = ode45(nmr_ode3, tspan2, M1(end, :));
[t3, M3] = ode45(nmr_ode2, tspan3, M2(end, :));
[t4, M4] = ode45(nmr_ode3, tspan4, M3(end, :));
[t5, M5] = ode45(nmr_ode2, tspan5, M4(end, :));

% Combine results of all time windows
tout = [t1; t2(2:end); t3(2:end); t4(2:end); t5(2:end)];
Mout = [M1; M2(2:end, :); M3(2:end, :); M4(2:end, :); M5(2:end, :)];

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
title('Magnetic moment precession in DC field after pi and pi/2 pulses');
text(1.1e-6, -0.8, 'tau=0.3e-6 s');
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz');
```



ta

Max Amplitude when $\tau = 0.3$?s

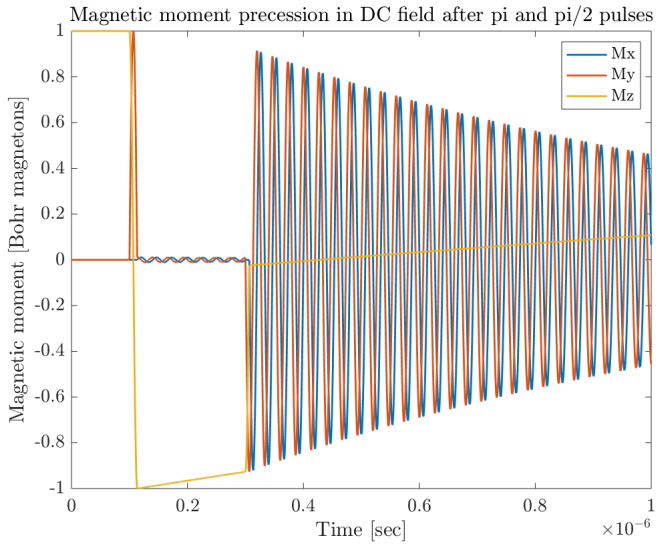
From the graph, max amplitude when ? = 0.3e-6 is given by the peak in $-M_x$ plus the equilibrium value of M_z, which is M_o:

```
Amax1 = 0.8766 + Mo(3);          % Max amplitude read off the graph when tau is 0.3e-6 ms [Bohr magnetons]
```

Simulating $\tau = 0.6$?s

We again redefine ? as well as the relevant time windows:

```
%tau2 = 0.6e-6;          % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
tau2 = 0.2e-6;          % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
tspan3 = ((tpi+tpulse) : dt : (tpi+tau2));           % Time window after the pulse and before the  pulse
tspan4 = ((tpi+tau2) : dt : (tpi+tau2+tpulse/2));     % Time window during which the  pulse RF field is turned on
tspan5 = ((tpi+tau2+tpulse/2) : dt : Dt);            % Time window after both pulses

% Integrate results in each time window
[t1, M1] = ode45(nmr_ode2, tspan1, Mi);
[t2, M2] = ode45(nmr_ode3, tspan2, M1(end, :));
[t3, M3] = ode45(nmr_ode2, tspan3, M2(end, :));
[t4, M4] = ode45(nmr_ode3, tspan4, M3(end, :));
[t5, M5] = ode45(nmr_ode2, tspan5, M4(end, :));

% Combine results of all time windows
tout = [t1; t2(2:end); t3(2:end); t4(2:end); t5(2:end)];
Mout = [M1; M2(2:end, :); M3(2:end, :); M4(2:end, :); M5(2:end, :)];

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
title('Magnetic moment precession in DC field after pi and pi/2 pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz');
```



Max Amplitude when $\tau = 0.6$?s

From the graph, max amplitude when ? = 0.6e-6 is given by the peak in $-M_x$ plus the equilibrium value of M_z, which is M_o:

```
Amax2 = 0.7674 + Mo(3);  % Max amplitude read off the graph when tau is 0.3e-6 ms [Bohr magnetons]
```

Estimating T_1 from Max Amplitude

We have the following relationship for the amplitude of an exponential at points (τ_1, A) and (τ_2, B):

$$\frac{B}{A} = e^{\frac{-(\tau_2 - \tau_1)}{T_1}}$$

$$T_{1,\text{est}} = \frac{(\tau_2 - \tau_1)}{\ln\left(\frac{A}{B}\right)}$$

We use this to estimate T_1 from the max amplitude results:

```
T1_est = (tau2 - tau1)/log(Amax1/Amax2);        % Estimated T1 from max amplitude results [s]
T1_est
```

```
T1_est = -1.6680e-06
```

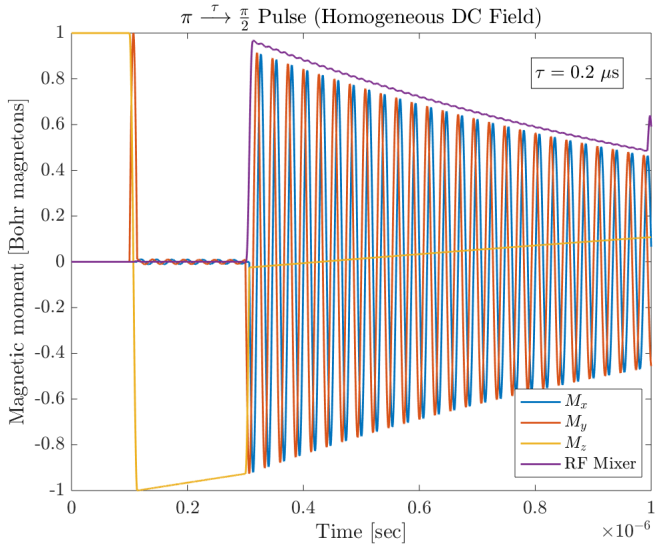
We confirm we can recover T_1 from the simulation, accounting for some slight error from reading off the graph.

Introducing the Mixer Signal

Finally, we want to recover T_2 from the simulation by introducing a mixer signal which is a rolling average of the multiplication of M_x and the RF oscillator (which is just a $\sin$ function at the resonance frequency assuming we have tuned the system to resonance when taking experimental measurements). It should result in a smoothly exponential decay curve. I had to adjust the size of the rolling average window to get a smooth curve. Any window that is a factor of 133 data points give smooth results, probably because this window perfectly coveres the sinusoid.

Define the mixer signal `mixer` of M_x and the RF oscillator as:

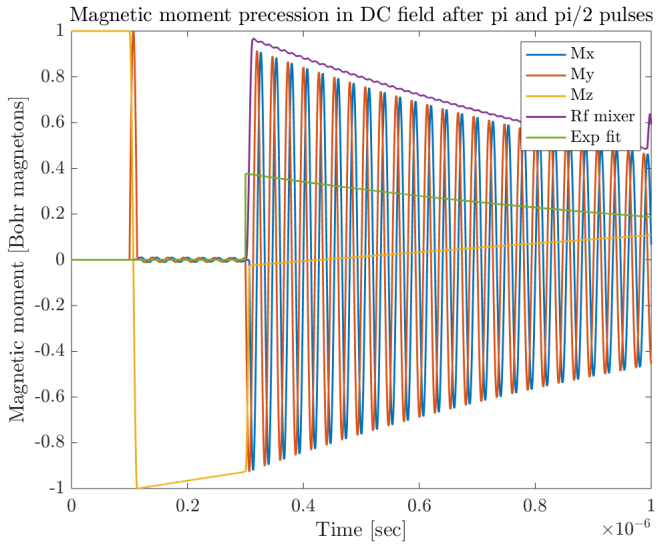
```
mixer = @(t) 2.1 * (movmean(sin(omega0 * t) .* Mout(:, 1), 133.333)); % Mixer signal of Mx

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
plot(tout, mixer(tout));
hold on;
title('$\pi \stackrel{\tau}{\longrightarrow} \frac{\pi}{2}$ Pulse (Homogeneous DC Field)')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
text(0.8e-6, 0.825,'$\tau=0.2\ \mu$s', 'FontSize', 14, "EdgeColor", 'black')
legend('$M_x$', '$M_y$', '$M_z$', 'RF Mixer', 'Location', "best")
```



Recovering T_2 from the Mixer Signal

We now simply recover T_2 from the mixer signal by fitting an exponential, which was determined trial and error.

```
% Define an exponential and fit it to the curve
expT2 = @(t) 0.3772*exp(-(t - (tpi+tau2))/T2) .* (t>=tpi+tau2);

clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
plot(tout, Mout(:, 3));
hold on;
plot(tout, mixer(tout));
hold on;
plot(tout, expT2(tout));
hold on;
title('Magnetic moment precession in DC field after pi and pi/2 pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz', 'Rf mixer', 'Exp fit');
```



From looking at the graph, the mixer signal and exponential fit clearly have the same time contstant. This confirms we can recover T_2 from the simulation.

Conclusion

Simulation

We numerically simulated the precession of a magnetic moment in a DC field after being hit with a $\pi \longrightarrow \pi/2$ pulse. As expected, after the $?$ pulse, M decays towards its equilbirium value with time constant $T_!$, until it is hit with $\pi/2$ pulse, which rotates it into xy plane where it precesses and decays with time constant T_2.

Recovering T_1

We confirmed we were able to recover T_1 from the peak amplitude of M_{xy}, which is the value of M_z right before it is flipped into M_{xy} by the $\pi/2$ pulse, by comparing values of M_{xy} for different simulated ?. It was important to make sure to account for the fact that M_z decays according to its difference from equilbrium, not its difference from zero, i.e. $(M_z - M_o)$.

Recovering T_2

We also confirmed we were able to recover T_2 from simulation by using a mixer signal, which shows the decay pattern of M_{xy} after the $\pi/2$ pulse. We don't actually expect to be able to recover an accurate result T_2 in the experiment due to field inhomogeneity. It worked here because we ignored magnet inhomogeneity.

Task 2v1 - Exploring Experimental Data

Goal

Load real experimental data and see that a plot of voltage vs time looks valid.

```
clearvars;
```

Loading Data Files

Load all the directories in the data folder:

```
cd ..
cd 'Task 2' % Go to data folder
files = dir('NMR*.DAT') % Load all data in Task 2 data folder
```

```
files = 40@1 struct
```

Fields	name	folder	date	bytes	isdir	datenum
1	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182037	0	7.3805e+05
2	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182037	0	7.3805e+05
3	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182036	0	7.3805e+05
4	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182036	0	7.3805e+05
5	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182038	0	7.3805e+05
6	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182038	0	7.3805e+05
7	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182037	0	7.3805e+05
8	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182037	0	7.3805e+05
9	'NMR1t2_0…	'/Users/kee…	'17-Sep-20…	182032	0	7.3805e+05
10	'NMR1t2_1…	'/Users/kee…	'17-Sep-20…	182032	0	7.3805e+05
11	'NMR1t2_1…	'/Users/kee…	'17-Sep-20…	182032	0	7.3805e+05
12	'NMR1t2_1…	'/Users/kee…	'17-Sep-20…	182031	0	7.3805e+05
13	'NMR1t2_1…	'/Users/kee…	'17-Sep-20…	182032	0	7.3805e+05
14	'NMR1t2_1…	'/Users/kee…	'17-Sep-20…	182031	0	7.3805e+05

?

```
% Confirm we can recover name from struct
files(1).name
```

```
ans = 'NMR1t2_01.DAT'
```

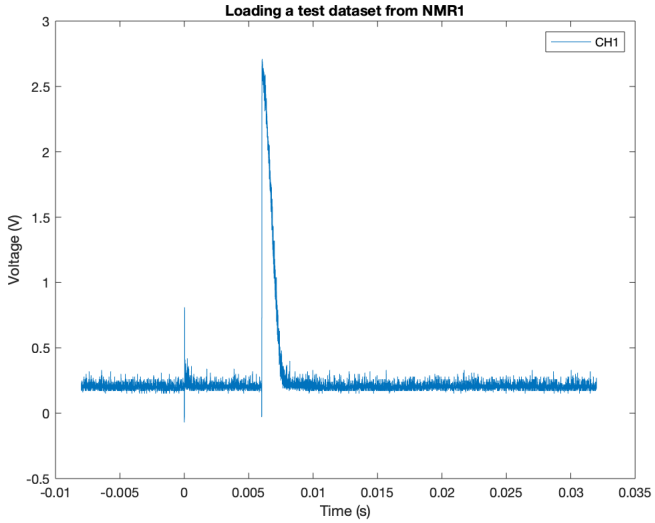
```
% Try loading a test file
filetest = importdata(files(1).name)
```

```
filetest = struct with fields:
          data: [10000@2 double]
      textdata: {'Time (S)'  'Ch1 Voltage (V)'}
    colheaders: {'Time (S)'  'Ch1 Voltage (V)'}
```

Plotting a Test Dataset

We extract time t of recording from the first data column, and voltage V from the second, and plot:

```
% Grab the data from this file and plot
ttest = filetest.data(:, 1);
vtest = filetest.data(:, 2);

% Plot
clf();
plot(ttest, vtest);
hold on;
title('Loading a test dataset from NMR1');
xlabel('Time (s)');
ylabel('Voltage (V)');
legend('CH1');
```



Conclusion

Looks like we have loaded everything correctly. Time to move on.

Task 2v2 - Preliminary T_1 Measurement

Goal

Determine T_1 by comparing maximum amplitudes at different $?$. We can use the $?$ values given in `descriptor.dat`:

```
clearvars;
```

Loading Data Files

```
cd '..'
cd 'Task 2' % Go to data folder
files = dir('NMR*.DAT'); % Load all data in Task 2 data folder
```

Define Physical Parameters

There seems to be a voltage offset present in all voltage data. We can remove this. From examining the graph, V_{off} (`v_off`) is:

```
voff = 0.25;
```

From `descriptor.dat` we have the values of $?$ in each dataset:

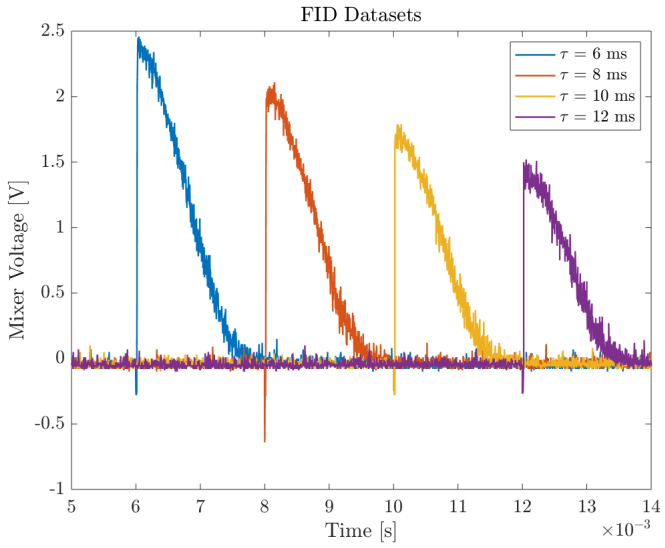
```
tau1 = 6e-3;    % Dataset 1 [ms]
tau2 = 8e-3;    % Dataset 2 [ms]
tau3 = 10e-3;   % Dataset 3 [ms]
tau4 = 12e-3;   % Dataset 3 [ms]
```

Plotting Data

Begin by plotting three oscilloscope curves for varying $?$ and grabbing maximum amplitudes at each $?$. Again t is the first column and V the second.

```
% Grab three test datasets and extract voltage/time
data1 = importdata(files(1).name);
t1 = data1.data(:, 1);
v1 = data1.data(:, 2) - voff;
data2 = importdata(files(2).name);
t2 = data2.data(:, 1);
v2 = data2.data(:, 2) - voff;
data3 = importdata(files(3).name);
t3 = data3.data(:, 1);
v3 = data3.data(:, 2) - voff;
data4 = importdata(files(4).name);
t4 = data4.data(:, 1);
v4 = data4.data(:, 2) - voff;

% Plot together for visualization
clf();
plot(t1, v1);
hold on;
plot(t2, v2);
hold on;
plot(t3, v3);
hold on;
plot(t4, v4);
hold on;
```

```
xlabel('Time [s]');
ylabel('Mixer Voltage [V]');
title('FID Datasets')
legend('$\tau=$ 6 ms', '$\tau=$ 8 ms', '$\tau=$ 10 ms', '$\tau=$ 12 ms');
xlim([5e-3 14e-3]);
```



Estimating T_1 from the Plots

From using the data point tool on the graphs, we have numerical estimates of voltage maximum:

```
vmax1 = 2.45;   % Maximum amplitude at tau 1 [V]
vmax2 = 2.08;   % Maximum amplitude at tau 2 [V]
vmax3 = 1.75;   % Maximum amplitude at tau 3 [V]
```

Reusing the formula from Task 1v3:

$$T_{1,\text{est}} = \frac{(\tau_2 - \tau_1)}{\ln\left(\frac{A}{B}\right)}$$

We compute the following estimates

```
T1_est1 = (tau2 - tau1)/log(vmax1/vmax2); % T1 estimated from tau1 and tau2
T1_est2 = (tau3 - tau1)/log(vmax1/vmax3); % T1 estimated from tau1 and tau3
T1_est3 = (tau3 - tau2)/log(vmax2/vmax3); % T1 estimated from tau2 and tau3
```

Taking the average of these gives

```
T1_est = (T1_est1 + T1_est2 + T1_est3) / 3 % Averaged estimate
```

```
T1_est = 0.0119
```

Determining Uncertainty

Determining uncertainty in this section is not worthwhile. We will find uncertainty we do a better estimate of uncertainty later in this task.

Conclusion

We estimate $T_1 \approx 12$ ms. This is on the order of 10^1 to 10^2 ms as expected. We have only looked at 3 of the 40 datasets though. Need to figure out a way to automate voltage max calcuation and fit a true value of T_1.

Task 2v3 - T_2 Measurement

Goal

To obtain an estimate of T_2 by matching an exponential to a few of the voltage vs time curves:

```
clearvars;
```

Loading Data Files

```
cd ..
cd 'Task 2' % Go to data folder
files = dir('*.DAT'); % Load all data in data Task 2 folder
```

Define Physical Parameters

Reuse voltage offset V_{off} (`v_off`) from **Task 2v3**:

```
voff = 0.22;
```

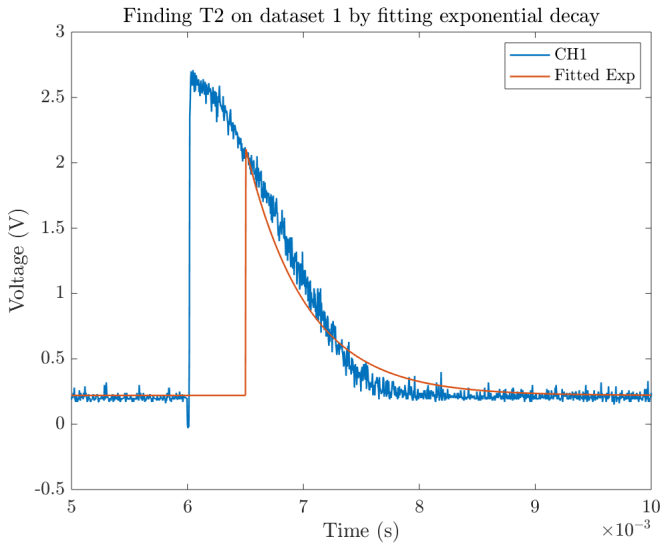
I also noticed that I have to offset the start of the exponential fit slightly in time to avoid some non-expoential data near the voltage peak. Call this t_{off} (`toff`), its about 0.5 ms.

```
toff = 0.5e-3;
```

Fitting T2 to Dataset 1

Now, find T_2 of dataset 1. Introduce the exponential function we will match to the curve and rerun this with modified parameters until it fits, then plot:

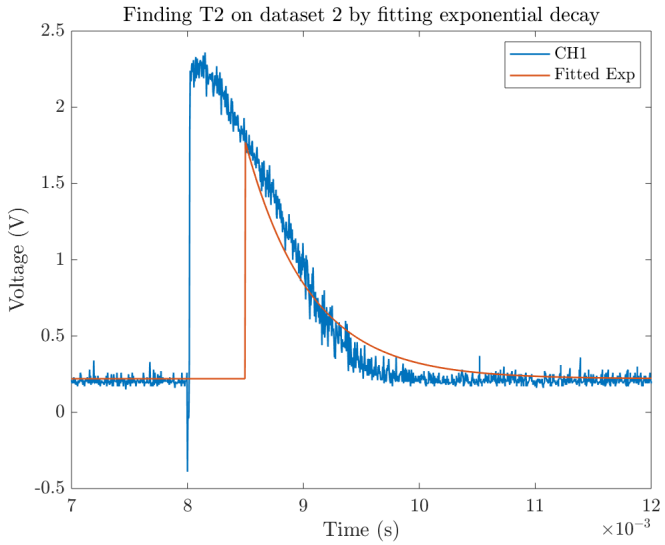
```
% Import dataset 1
data1 = importdata(files(1).name);
t1 = data1.data(:, 1);
v1 = data1.data(:, 2);

T21 = 0.52e-3; % Fitted T2 constant for dataset 1 [s]
tau1 = 6.0e-3 + toff; % Found in descriptor.dat (include time offset)
A1 = 1.9; % Fitted max amplitude of dataset 1 [V]
exp1 = @(t) voff .* (t < tau1) + (A1 * exp(-(t-tau1)/T21) + voff) .* (t >= tau1); % Fit dataset 1

% Plot
clf();
plot(t1, v1);
hold on;
plot(t1, exp1(t1));
hold on;
title('Finding T2 on dataset 1 by fitting exponential decay');
xlabel('Time (s)');
ylabel('Voltage (V)');
legend('CH1', 'Fitted Exp');
xlim([5e-3 10e-3]);
```



Fitting T2 to Dataset 2

Continue with dataset 2

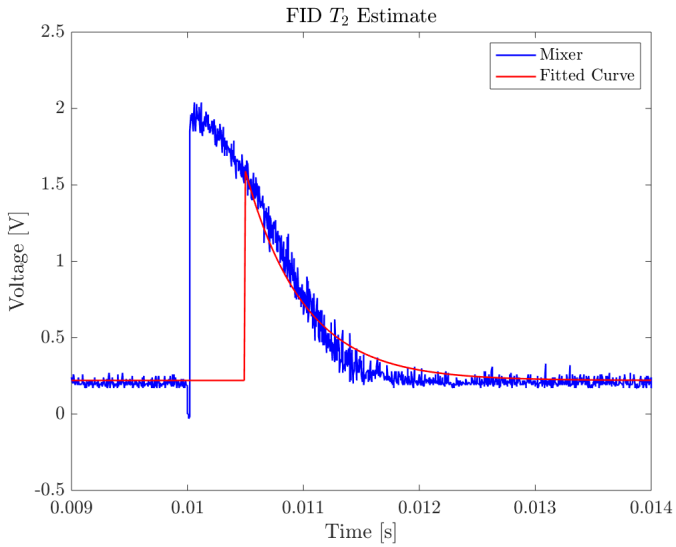
```
% Import dataset 2
data2 = importdata(files(2).name);
t2 = data2.data(:, 1);
v2 = data2.data(:, 2);

T22 = 0.55e-3; % Fitted T2 constant for dataset 2 [s]
tau2 = 8.0e-3  + toff; % Found in descriptor.dat (include time offset)
A2 = 1.55; % Fitted max amplitude of dataset 2 [V]
exp2 = @(t) voff .* (t < tau2) + (A2 * exp(-(t-tau2)/T22) + voff) .* (t >= tau2); % Fit dataset 2

% Plot
clf();
plot(t2, v2);
hold on;
plot(t2, exp2(t2));
hold on;
title('Finding T2 on dataset 2 by fitting exponential decay');
xlabel('Time (s)');
ylabel('Voltage (V)');
legend('CH1', 'Fitted Exp');
xlim([7e-3 12e-3]);
```



Fitting T2 to Dataset 3

Continue with dataset 3:

```
% Import dataset 3
data3 = importdata(files(3).name);
t3 = data3.data(:, 1);
v3 = data3.data(:, 2);

T23 = 0.5e-3; % Fitted T2 constant for dataset 3 [s]
tau3 = 10e-3 + toff; % Found in descriptor.dat (include time offset)
A3 = 1.37; % Fitted max amplitude of dataset 3 [V]
exp3 = @(t) voff .* (t < tau3) + (A3 * exp(-(t-tau3)/T23) + voff) .* (t >= tau3); % Fit dataset 3

% Plot
clf();
plot(t3, v3, 'b');
hold on;
plot(t3, exp3(t3), 'r');
hold on;
title('FID $T_2$ Estimate');
xlabel('Time [s]');
ylabel('Voltage [V]');
legend('Mixer', 'Fitted Curve');
%text(0.0115, 0.8,'$T_2=0.223$ ms', 'FontSize', 14, "EdgeColor", 'black')
xlim([9e-3 14e-3]);
```



Estimating T_2 from Fits

Average out the T_2 determined for each fit to estimate T_2 (`T2_est`):

```
T2_est = (T21 + T22 + T23) / 3;
T2_est
```

```
T2_est = 5.2333e-04
```

Determining Uncertainty

We can get an estimate for uncertainty by simply deciding how much variance I allowed in T_2 when fitting the exponentials by eye. I would say this was around $\pm 25\ \mu s$. Rounding this up, we have estimate for uncertainty in T_2 determined here:

$$\sigma(T_2) = \pm 30\ \mu s$$

Conclusion

My estimate for spin-spin decay with uncertainty is:

$$T_2 = 520 \pm 30\ \mu s$$

This is significantly lower than expected range of 10^1 to 10^2 ms.

This doesn't seem to be a good estimate of T_2. First off, there is significant leeway in the way I chose T_2 to fit each plot, since it is difficult what the best fit is by eye. We also don't expect this method to give us an accurate value of T_2 anyways. The inhomogeneity in the field seen by each proton due to spin-spin interaction means the decay seen isn't the true T_2. To obtain the true value of T_2 we need to use the spin-echo technique.

Task 2v4 - Improving T_1 Measurement

NOTE: REQUIRES CURVE FITTING TOOLBOX TO BE INSTALLED

Goal

Automate the collection of amplitude peak data to get a better estimate of T_1

```
clearvars;
```

Loading Data Files

```
cd ..
cd 'Task 2' % Go to data folder
files = dir('NMR*.DAT'); % Load all data in data folder
```

Extracting V_{max} from each Dataset

Get the maximum of each voltage dataset V_{max}, an put into an array containing each V_{max} (`vmax` a `1x40` array):

```
n = length(files);      % Number of datasets explored
vmax = zeros(1, n);     % Vmax of each dataset

for i = 1:n
    data = importdata(files(i).name);
    v = data.data(:, 2); % Voltage is second column
    vmax(i) = max(v); % Get voltage max minus voltage offset
end
```

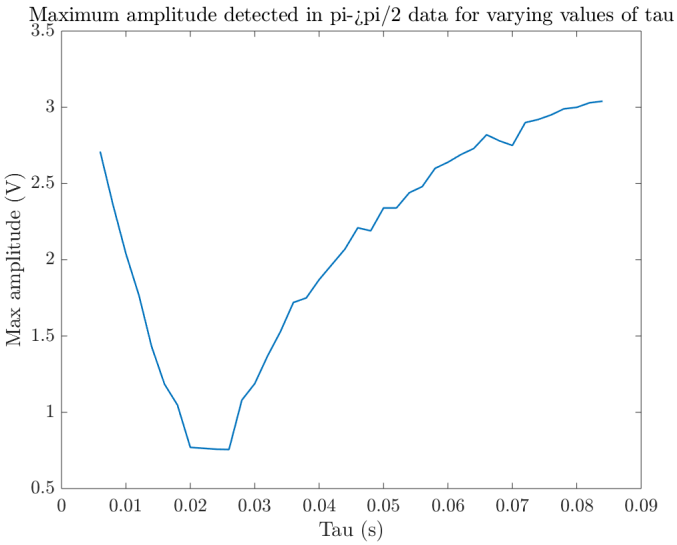
The $?$ for each dataset is given in the `descriptor.dat` file, we recreate the range as a `1x40` array `tau`:

```
tau = 6e-3:2e-3:84e-3;
```

Plotting Results

We can now plot V_{max} for varying values of $?$:

```
clf();
plot(tau, vmax);
hold on;
title('Maximum amplitude detected in pi->pi/2  data for varying values of tau');
xlabel('Tau (s)');
ylabel('Max amplitude (V)');
```



Immediately we notice that its not really an exponential. For my explanation of what's going on here see the **Conclusion** of this task.

Fitting an Exponential to Extract T_1

Lets try fitting an exponential to both the decaying and growing exponential sections.

Define Physical Parameters

At this point I realized we actually need to fit an exponential to the difference between M_z and the equilbrium moment M_o (equilibrium voltage in data). This value was determined by trial and error until the T_1 in the decaying region and growing region matched. Note that the we are actually plotting $-(M_{xy,peak} - M_o)$ in both regions to account for the fact that M_z is really negative before flip in the decaying region. This allows for a more natural exponential graph shape.

```
v_eqbm = 3.75;          % Equilibrim magnetic moment in voltage (V)
```

Using MATLAB's `fit` Function for Exponential Fit

Split the decay and growth regions, removing the middle values that don't make sense.

```
tau_decay  = tau(1:8)';                 % Decaying region of tau (1x8) array (s)
vmax_decay  = vmax(1:8)' + v_eqbm;      % Decaying region of vmax (1x8) array (V)
tau_grow   = tau(12:40)';               % Growing region of tau (1x28) array (s)
vmax_grow   = -vmax(12:40)' + v_eqbm;   % Growing region of vmax (1x28) array (V)
```

Use built-in `fit` function to fit an exponential to each region (by defauly using non-linear least squares regression):

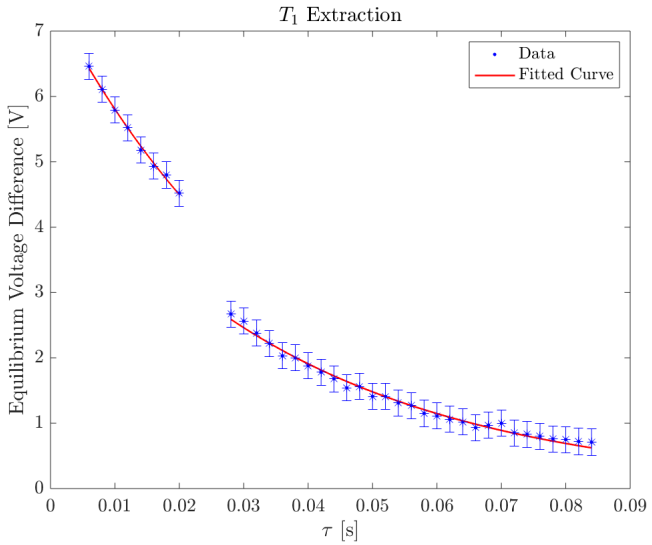
```
exp_decay = fit(tau_decay, vmax_decay, 'exp1')  % Use built-in MATLAB exponential fit
```

```
exp_decay =
     General model Exp1:
     exp_decay(x) = a*exp(b*x)
     Coefficients (with 95% confidence bounds):
       a =       7.492  (7.342, 7.642)
       b =      -25.52  (-27.09, -23.94)
```

```
exp_grow  = fit(tau_grow, vmax_grow, 'exp1')    % Use built-in MATLAB exponential fit
```

```
exp_grow =
     General model Exp1:
     exp_grow(x) = a*exp(b*x)
     Coefficients (with 95% confidence bounds):
       a =        5.28  (5.032, 5.528)
       b =      -25.44  (-26.47, -24.41)
```

```
% Plot
clf();
plot(exp_decay, tau_decay, vmax_decay);
hold on;
plot(exp_grow, tau_grow, vmax_grow);
hold on;
error_decay = 0.2 * ones(size(vmax_decay));
error_grow = 0.2 * ones(size(vmax_grow));
errorbar(tau_decay, vmax_decay, error_decay, 'b*');
hold on;
errorbar(tau_grow, vmax_grow, error_grow, 'b*');
hold on;
title('$T_1$ Extraction')
xlabel('$\tau$ [s]');
ylabel('Equilibrium Voltage Difference [V]');
legend('Data', 'Fitted Curve')
```



Extracting T_1 from the Fit

Extracting T_1 from the decay fit parameters:

```
coeff_decay = coeffvalues(exp_decay);
T1_decay = -1 / coeff_decay(2)
```

```
T1_decay = 0.0392
```

Extracting T_1 from the growth fit paramaters:

```
coeff_grow = coeffvalues(exp_grow);
T1_grow = -1 / coeff_grow(2)
```

```
T1_grow = 0.0393
```

Uncertainty Analysis for T_1

We can use the 95% confidence interval estimate given by the exponential `fit` function to estimate uncertainty in T_1. This corresponds to a k factor of $k \approx 2$ ($k = 1.96$ in reality), so we just take difference between the CI interval endpoints and divide by 2. Uncertainty estimates in both the decay and growing regions are:

$$\sigma(T_{1,\text{decay}}) = \frac{\frac{1}{23.94} - \frac{1}{27.09}}{2} = \pm 2.4 \text{ ms}$$

$$\sigma(T_{1,\text{grow}}) = \frac{\frac{1}{24.41} - \frac{1}{26.47}}{2} = \pm 1.6 \text{ ms}$$

Averaging out we get:

$$\sigma(T_1) = 2 \text{ ms}$$

Conclusion

After matching and extracted T_1 from both ranges of the data (minus the flat region neat the middle) by fitting an exponential, and we obtain a final estimate with uncertainties for T_1:

$$T_1 = 39 \pm 2 \text{ ms}$$

This is on the order of 10^1 to 10^2 ms as expected. This is a more accurate estimate than the one in **Task 2v2** because it accounts for a wider range of values and models the equlibrium value of M_z i.e. M_0.

This graph of maximum amplitude (2nd to last graph) makes sense. The $?$ pulse flips M_z into $-M_z$ and then after $?$ the $\pi/2$ pulse flips M_z into M_{xy}. Remember that M_{xy} amplitude peak is then just whatever M_z was before it was flipped. In the range $?$ is less than the time it takes for $-M_z$ to grow back to zero, the graph of max amplitude steadily decreases. In the range where $?$ is larger than this $M_z = 0$ time, the graph of max amplitude steadily increases. In the middle, values don't make sense, probably because the amplitude is too small to discern well.

N.B. This means that instead of fitting just the peak M_{xy} value, we instead need to fit the equilibrium difference of $M_{xy,peak} - M_0$. This was determined by trial and error until the T_1 values in the decaying and growing region matched up.

Improving Uncertainty

The result for uncertainty result seems reasonable. There are clearly some points that aren't completely along the fit line in the graph. We could reduce this by getting a better signal to noise ratio on the voltage reading. Since we just blindly pick the maximum voltage in each dataset, its possible some of the maximums are altered slightly by noise. Alternatively, we could use filtering on the voltage data (e.g. Butterworth).

Task 3v1 - Speeding Up Integration

Goal

During my checkup instructor meeting, it was suggested that I could speed up my simulation by manually performing the rotations of M (i.e. numerically forcing M into a direction as need be) intead of integrating over the pulse window. We already showed that the we could perform the rotations anyways so I may as well speed this up.

I want to show that performing integration this way speeds up integration while achieve nearly the same result

```
clearvars;              % Clean environment
```

Define Numerical Simulation Parameters

Since we're removing the integration over the pulse windows, we only need to integrate over three time windows now. We reuse the numerical parameters and from Task 2:

```
dt = 1e-10;            % Numerical time resolution [s]
Dt = 1.5e-6;           % Numerical time window [s]
tpi = 0.1e-6;          % Starting point of pi pulse, i.e the first pulse [s]
tau = 0.4e-6;          % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
```

We can now define the time windows as:

```
tspan1 = (0 : dt : tpi);        % The time window before any pulse dynamics
tspan2 = (tpi : dt : tpi+tau);  % The time window after the pi pulse
tspan3 = (tpi+tau : dt : Dt);   % The time window after the pi/2 pulse
```

Define Physical Paramaters

All of the physical parameters from **Task 2** are the same:

- `gam`: Gyromagnetic ratio [rad / s * T]
- `Bo`: Applied magnetic field strength [T]
- `B`: Applied magnetic field vector [T] (3x1 column vector)
- Mo: Vector capturing subtraction of equilbrium value from moment in z [Bohr magnetons] (3x1 column vector)
- `T1`: Spin-lattice relaxation time [s]
- `T2`: Spin-spin relaxation time [s]
- `Tinv`: Array to multiply terms of M by respective inverse time constant [1/s] (3x3 array)

```
gamma = 2.675e8;        % Gyromagnetic ratio of a proton [rad/s*T]
Bo    = 0.88;           % Applied magnetic field strength [T]
B = [0.0; 0.0; Bo];     % Applied magnetic field vector [T] (3D column vector)

Mo = [0; 0; 1.0];       % Vector capturing subraction of equilbrium value from moment in z [Bohr magnetons]
T1 = 5e-6;              % Spin-lattice relaxation time [s]
T2 = 1e-6;              % Spin-spin relaxation time [s]
Tinv = [1/T2 0 0; 0 1/T2 0; 0 0 1/T1]; % Multiplies terms of M by repsective inverse time constant
```

Define ODE Initial Conditions

Again, these are the same as **Task 2**

```
Mi = [0.0; 0.0; 1.0]; % Magnetic moment initial conditions as a vector [Bohr magnetons]
```

Integrate ODE

Since we don't have to worry about integrating in the combined RF and DC field mode, we only need ODE equation (2), restated here for convenience:

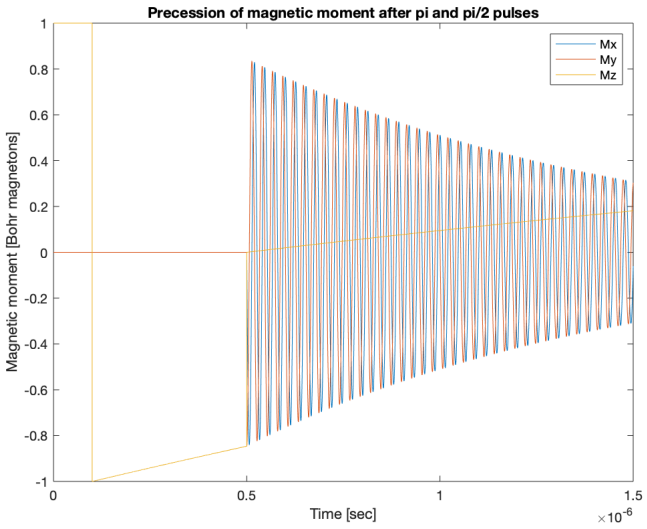
$$\frac{d\vec{M}(t)}{\mathrm{dt}} = \gamma \vec{M} \times \vec{B} - T_{\mathrm{inv}}\left(\vec{M} - \vec{M_0}\right) \qquad (2)$$

```
nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo); % NMR ODE equation (2)
```

We can now perform the integrations and flips in each time window and plot:

```
% Integrate up to pi pulse flip
[t1, M1] = ode45(nmr_ode2, tspan1, Mi);

% Flip Mz and integrate to pi/2 pulse at tau
Mi1 = [M1(end, 1); M1(end, 2); -M1(end, 3)];
[t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

% Flip Mz into y and integrate to end time
Mi2 = [M2(end, 1); M2(end, 3); 0.0];
[t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

% Put together output from all time windows
tout = [t1; t2(2:end); t3(2:end)];
Mout = [M1; M2(2:end, :); M3(2:end, :)];

% Plot
clf();
plot(tout, Mout(:, 1));
hold on;
plot(tout, Mout(:, 2));
hold on;
```

```
plot(tout, Mout(:, 3));
hold on;
title('Precession of magnetic moment after pi and pi/2 pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz');
```



Conclusion

This result looks good. There is a little less M_{xy} activity in the region before the $\pi/2$ pulse, but this is acceptable as it this region is not useful anyways. Runtime has noticeably sped up as we desired.

Task 3v2 - Introducing Inhomogeneity

WARNING: CODE IN THIS SECTION TAKES A FEW MINUTES TO RUN

Goal

Simulate the proton ensemble by introducing inhomogenous magnetic fields for each proton.

```
clearvars;          % Clean the environment
```

Define Physical Paramaters

We carry over the physical parameters from **Task 2v3**:

- `gam`: Gyromagnetic ratio [rad / s * T]
- `Bo`: Applied magnetic field strength [T]
- `B`: Applied magnetic field vector [T] (3x1 column vector)
- Mo: Vector capturing subtraction of equilbrium value from moment in z [Bohr magnetons] (3x1 column vector)
- `T1`: Spin-lattice relaxation time [s]
- `T2`: Spin-spin relaxation time [s]
- `Tinv`: Array to multiply terms of M by respective inverse time constant [1/s] (3x3 array)

```
gamma = 2.675e8;        % Gyromagnetic ratio of a proton [rad/s*T]
Bo    = 0.88;           % Applied magnetic field strength [T]

Mo = [0; 0; 1.0];       % Vector capturing subraction of equilbrium value from moment in z [Bohr magnetons]
T1 = 5e-6;              % Spin-lattice relaxation time [s]
T2 = 1e-6;              % Spin-spin relaxation time [s]
Tinv = [1/T2 0 0; 0 1/T2 0; 0 0 1/T1]; % Multiplies terms of M by repsective inverse time constant
```

Define the Inhomogenous Distribution

To simulate inhomogeneity, we saw that each of the protons feels slightly different magnetic field

$$B_k = B_0 + \delta B_k$$

For the simulation, we say δB_k is normally distributed. We can now introduce parameters for number of protons to simulate, n, and a $?$ parameter for the width of the distribution, which we will vary to investigate how it affects simulation results. We assume that mean $?$ is zero, i.e. the distribution fluctuates around B_0.

Nominally, we set n to 500, and use the standardized normal distribution, i.e. $\mu = 0$ and $\sigma = 1$. Numerically:

```
n = 500;                % Number of protons to simulate
sigma = 1.0;            % Variance parameter of normal distribution
mu = 0;                 % Mean parameter of normal distribution
```

Define the Time Parameters

```
dt = 1e-10;            % Numerical time resolution [s]
Dt = 0.99e-6;           % Numerical time window [s]
```

```
tpi = 0.1e-6;          % Starting point of pi pulse, i.e the first pulse [s]
tau = 0.3e-6;          % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
```

We can now define the time windows the same way as the previous task:

```
tspan1 = (0 : dt : tpi);        % The time window before any pulse dynamics
tspan2 = (tpi : dt : tpi+tau);  % The time window after the pi pulse
tspan3 = (tpi+tau : dt : Dt);   % The time window after the pi/2 pulse
tspan  = [tspan1 tspan2(2:end) tspan3(2:end)]; % The combined time windows
```

Define ODE Initial Conditions

Again, these are the same as **Task 2v3:**

```
Mi = [0.0; 0.0; 1.0];   % Magnetic moment initial conditions as a vector [Bohr magnetons]
```

Integrate the ODE for each Proton

We can now do an integration for each proton, where we sample δB_k from a normal distrubtion with the above paramaters and add it to B_0. This means we redefine ODE equation (2) for each proton. We will add the $\vec{M}$ of each proton to the net magnetic moment vector $\vec{M}_{net}$ as we compute. M_{net} is really the averaged field felt by all the protons.

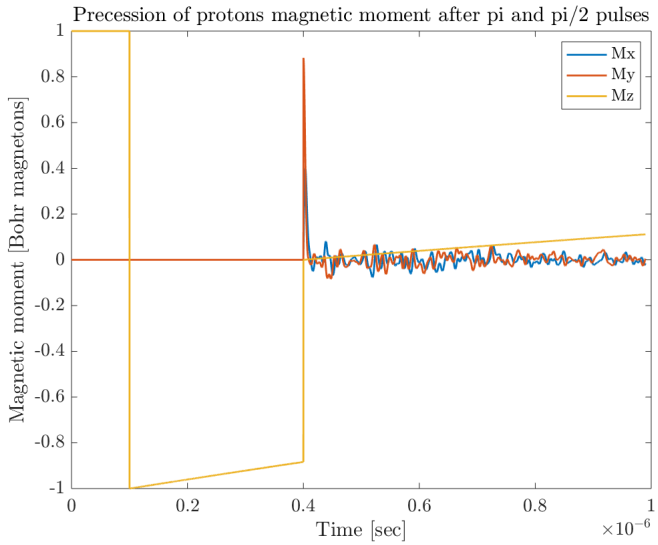
```
Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

We now perform the same integration from **Task 3v1** in a `for` loop:

```
for k = 1:n
    % New magnetic field sampled from distribution
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)];    % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi pulse flip
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz and integrate to Pi/2 pulse at tau
    Mi1 = [M1(end, 1); M1(end, 2); -M1(end, 3)];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip Mz into y and integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 3); 0.0];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];
```

And plot the results:

```
% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('Precession of protons magnetic moment after pi and pi/2 pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz');
```



This doesn't look great. Let's try modifiying the parameters of the distribution

Investigate the Distribution Parameters

My guess is the that $\sigma = 1$ is too high and causing too much variation in the B-field felt by each proton, which is why the net output just looks like noise. Lets try decreasing the original $?$ by an order of magnitude

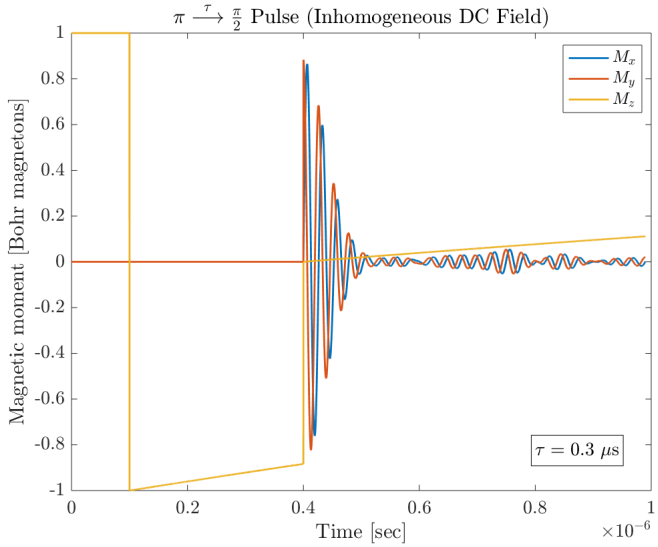
```
sigma = 0.1;              % Variance parameter of normal distribution
```

And make sure to reset M_{net}

```
Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

Rerunning the loop and plotting again:

```
for k = 1:n
    % New magnetic field
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)];    % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi pulse flip
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz and integrate to Pi/2 pulse at tau
    Mi1 = [M1(end, 1); M1(end, 2); -M1(end, 3)];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip Mz into y and integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 3); 0.0];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];

% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('$\pi \stackrel{\tau}{\longrightarrow} \frac{\pi}{2}$ Pulse (Inhomogeneous DC Field)')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
text(0.8e-6, -0.825,'$\tau=0.3\ \mu$s', 'FontSize', 14, "EdgeColor", 'black')
legend('$M_x$', '$M_y$', '$M_z$', 'Location', "best")
```



This looks looks recognizable as a decaying magnetic moment in the tranverse directions. The T_2 decay is much faster then the single magnetic moment simulation, around $0.1\ \mu s$. This is quite similar to the plot in Fig. 1.10(a) in the lab manual.

Let's try decreasing $?$ by a factor of 10 again to further investigate its effect.

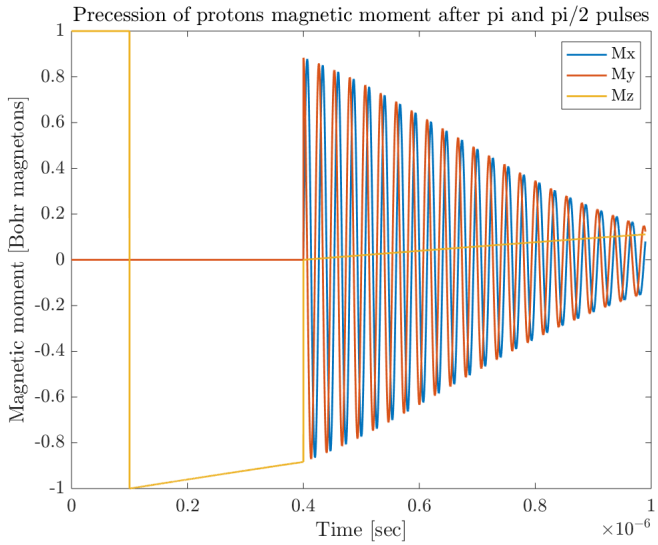
```
sigma = 0.01;               % Variance parameter of normal distribution
```

Don't forget to zero M_{net}:

```
Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

This loop and plot is repeated a final time:

```
for k = 1:n
    % New magnetic field
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)];    % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi pulse flip
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz and integrate to Pi/2 pulse at tau
    Mi1 = [M1(end, 1); M1(end, 2); -M1(end, 3)];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip Mz into y and integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 3); 0.0];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];

% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('Precession of protons magnetic moment after pi and pi/2 pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
legend('Mx', 'My', 'Mz');
```



Conclusion

We demonstrated the effects of a inhomogenous magnetic field distribution on T_2 by introducing a normally distributed fluctuation in the field and looking at the net magnetic moment of a ensemble of protons. A regime in which the apparent T_2 decay was much faster then the manually introduced T_2 was found.

By modifying the width of the fluctuation distributiod, we found that the higher $?$ resulted in faster T_2, while lower $?$ brought the precession closer to the case of single proton precession. This result makes sense, the lower $?$, the closer to having no fluctuation, which is the case of single proton precession. If $?$ is too high, the resulting B-field fluctuates too much and the resulting precession just looks like noise.

This also explains the results in **Task 2**. We expected T_2 to be on the order of magnitude of 10^1 to 10^2 ms, but it was on the order of 10^{-1}, so there must be some inhomogeneity shortening T_2, which makes sense because the experimental field is expected to be inhomogenous. To produce the accurate T_2, we need the spin echo.

The other fluctuation parameter n, the number of protons sampled, affected the amount of sinusoidal noise that appeared in between the two pulses. This noise appears because for low values of n, the fluctuation is not random enough to stop all sinusoidal modes from appearing. Choices of $n = \{100, 200, 500\}$ were tested. The amplitude of this noise was increased as n lowered. Overall $n = 200$ was chosen as a middle point for reduced sinusoidal noise without extremely long run times. We expect the spin echo to be independent of n parameter.

Task 3v3 - Modelling the Spin Echo

WARNING: CODE IN THIS SECTION TAKES A FEW MINUTES TO RUN

NOTE: REQUIRES CURVE FITTING TOOLBOX TO BE INSTALLED

Goal

Use a $\frac{\pi}{2} \xrightarrow{\tau} \pi$ pulse sequence to a avoid the dephasing of precessing magnetic moments and observe the appearance of a spin echo at $t = \tau$ (after the second pulse). Also investigate how $?$ affects the amplitude of the echo and how it compares to the value of T_2.

```
clearvars;          % Clean the environment
```

Define Physical Paramaters

Again we carry over the physical parameters from **Task 2v3**:

- `gam`: Gyromagnetic ratio [rad / s * T]
- `Bo`: Applied magnetic field strength [T]
- `B`: Applied magnetic field vector [T] (3x1 column vector)
- Mo: Vector capturing subtraction of equilbrium value from moment in z [Bohr magnetons] (3x1 column vector)
- `T1`: Spin-lattice relaxation time [s]
- `T2`: Spin-spin relaxation time [s]
- `Tinv`: Array to multiply terms of M by respective inverse time constant [1/s] (3x3 array)

```
gamma = 2.675e8;        % Gyromagnetic ratio of a proton [rad/s*T]
Bo    = 0.88;           % Applied magnetic field strength [T]

Mo = [0; 0; 1.0];       % Vector capturing subraction of equilbrium value from moment in z [Bohr magnetons]
T1 = 5e-6;              % Spin-lattice relaxation time [s]
T2 = 1e-6;              % Spin-spin relaxation time [s]
Tinv = [1/T2 0 0; 0 1/T2 0; 0 0 1/T1]; % Multiplies terms of M by repsective inverse time constant
```

Define the Time Parameters

We change `tpi` to `tpi2` to represent that we now start with th $\pi/2$ pulse, everything else is the same:

```
dt = 1e-10;            % Numerical time resolution [s]
Dt = 0.999e-6;           % Numerical time window [s]
tpi2 = 0.1e-6;         % Starting point of pi/2 pulse, i.e the first pulse [s]
```

```
tau = 0.4e-6;          % Vary to adjust time to hit with pi/2 pulse after pi pulse [s]
```

We can now define the time windows the same way as the previous task:

```
tspan1 = (0 : dt : tpi2);        % The time window before any pulse dynamics
tspan2 = (tpi2 : dt : tpi2+tau); % The time window after the pi pulse
tspan3 = (tpi2+tau : dt : Dt);   % The time window after the pi/2 pulse
tspan  = [tspan1 tspan2(2:end) tspan3(2:end)]; % The combined time windows
```

Define ODE Initial Conditions

Again, these are the same as **Task 2v3:**

```
Mi = [0.0; 0.0; 1.0];   % Magnetic moment initial conditions as a vector [Bohr magnetons]
```

Define the Inhomogenous Distribution

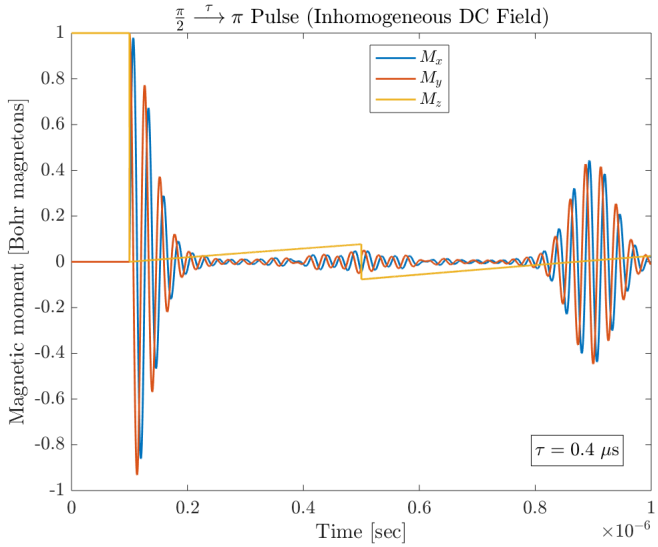
Again we simulate inhomogeneity using a distribution. In the last section $\sigma = 0.1$ and $n = 100$ gave good results so use them again:

```
n = 500;                % Number of protons to simulated
sigma = 0.1;            % Variance parameter of normal distribution
mu = 0;                 % Mean parameter of normal distribution
Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

Integrate the ODE

Perform the same loop integration as the previous section, but flip the order of the $?$ and $\pi/2$ pulses (i.e. change the order of the modification of initial conditions in the 2nd and third time window), and plot results:

```
for k = 1:n
    % New magnetic field
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)];    % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi/2 pulse at tpi2
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz into yand integrate to Pi pulse at tau
    Mi1 = [M1(end, 1); M1(end, 3); 0.0];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip the xy plane like a pancake integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 2); -M2(end, 3)];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];

% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('$\frac{\pi}{2} \stackrel{\tau}{\longrightarrow} \pi$ Pulse (Inhomogeneous DC Field)')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
text(0.8e-6, -0.825,'$\tau=0.4\ \mu$s', 'FontSize', 14, "EdgeColor", 'black')
legend('$M_x$', '$M_y$', '$M_z$', 'Location', "best")
```



Varying $?$ ($\tau = 0.2$e-6)

Lets now study the effect of changing $?$ on the amplitude of the spin echo and compare it to T_2. The plan here is to take four measurements of the maximum amplitude of the spin echo and fit an exponential to them.

Make sure to zero M_{net}:

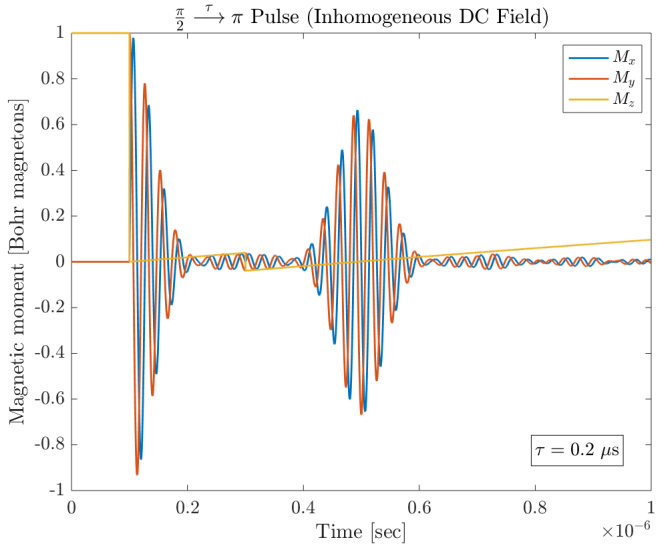
```
Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

Change the value of $?$ and recompute relevant time parameters:

```
tau = 0.2e-6;           % Vary to adjust time to hit with pi pulse after pi/2 pulse
tspan2 = (tpi2 : dt : tpi2+tau); % The time window after the pi pulse
tspan3 = (tpi2+tau : dt : Dt);   % The time window after the pi/2 pulse
```

Rerun the for loop integration and plot:

```
for k = 1:n
    % New magnetic field
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)];    % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi/2 pulse at tpi2
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz into yand integrate to Pi pulse at tau
    Mi1 = [M1(end, 1); M1(end, 3); 0.0];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip the xy plane like a pancake integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 2); -M2(end, 3)];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];

% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('$\frac{\pi}{2} \stackrel{\tau}{\longrightarrow} \pi$ Pulse (Inhomogeneous DC Field)')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
text(0.8e-6, -0.825,'$\tau=0.2\ \mu$s', 'FontSize', 14, "EdgeColor", 'black')
legend('$M_x$', '$M_y$', '$M_z$', 'Location', "best")
```



Varying τ ($\tau = 0.6$e-6)

Again recompute numerical parameters:

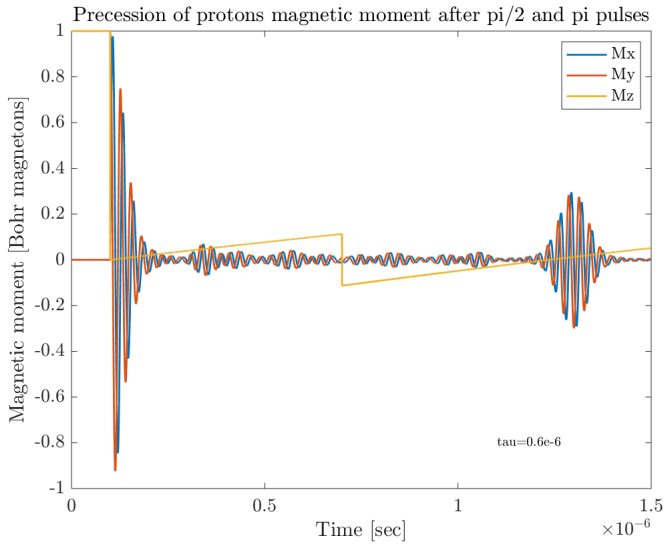
```
tau = 0.6e-6;           % Vary to adjust time to hit with pi/2 pulse after pi pulse
tspan2 = (tpi2 : dt : tpi2+tau); % The time window after the pi pulse
tspan3 = (tpi2+tau : dt : Dt);   % The time window after the pi/2 pulse
```

And again zero M_{net}:

```
Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

Rerun for the for loop:

```
for k = 1:n
    % New magnetic field
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)];    % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi/2 pulse at tpi2
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz into yand integrate to Pi pulse at tau
    Mi1 = [M1(end, 1); M1(end, 3); 0.0];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip the xy plane like a pancake integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 2); -M2(end, 3)];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];

% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('Precession of protons magnetic moment after pi/2 and pi pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
text(1.1e-6, -0.8, 'tau=0.6e-6');
legend('Mx', 'My', 'Mz');
```



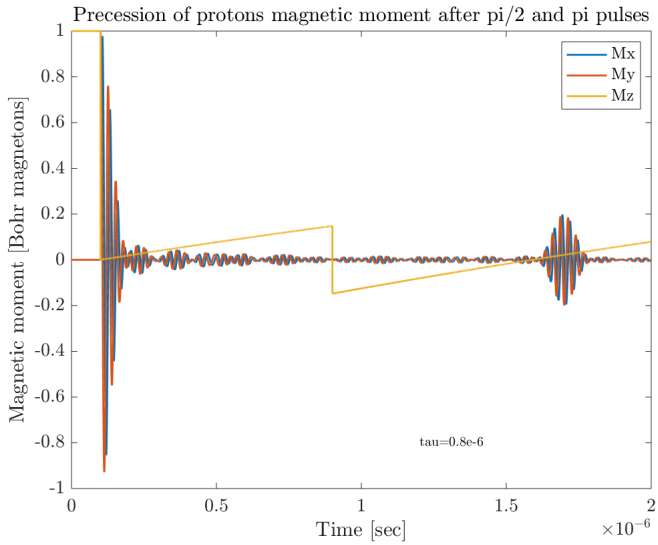
Varying τ ($\tau = 0.8$ e-6)

For a the final time, recompute numerical parameters and again zero M_{net}, after adjusting the time window end to be able to see the echo:

```
Dt = 2e-6;              % Maximum of time windows
tau = 0.8e-6;           % Vary to adjust time to hit with pi/2 pulse after pi pulse
tspan2 = (tpi2 : dt : tpi2+tau); % The time window after the pi pulse
tspan3 = (tpi2+tau : dt : Dt);   % The time window after the pi/2 pulse

Mnet = zeros(size(tspan, 1), 3); % Net magnetic moment vector, the averaged field by all the protons
```

Rerun for the for loop a final time:

```
for k = 1:n
    % New magnetic field
    B = [0.0; 0.0; Bo + normrnd(mu, sigma)]; % Applied magnetic field vector [T] (3D column vector)
    nmr_ode2 = @(t, M) gamma * cross(M, B')' - Tinv * (M - Mo);

    % Integrate up to Pi/2 pulse at tpi2
    [t1, M1] = ode45(nmr_ode2, tspan1, Mi);

    % Flip Mz into yand integrate to Pi pulse at tau
    Mi1 = [M1(end, 1); M1(end, 3); 0.0];
    [t2, M2] = ode45(nmr_ode2, tspan2, Mi1);

    % Flip the xy plane like a pancake integrate to end time
    Mi2 = [M2(end, 1); -M2(end, 2); -M2(end, 3)];
    [t3, M3] = ode45(nmr_ode2, tspan3, Mi2);

    % Magnetic moment over entire tspan
    Mout = [M1; M2(2:end, :); M3(2:end, :)];

    % Add to net magnetic moment
    Mnet = Mnet + Mout / n;
end

% This is the same in each loop so just compute at the end
tout = [t1; t2(2:end); t3(2:end)];

% Plot results
clf();
plot(tout, Mnet(:, 1));
hold on;
plot(tout, Mnet(:, 2));
hold on;
plot(tout, Mnet(:, 3));
hold on;
title('Precession of protons magnetic moment after pi/2 and pi pulses')
xlabel('Time [sec]');
ylabel('Magnetic moment [Bohr magnetons]');
text(1.2e-6, -0.8, 'tau=0.8e-6')
legend('Mx', 'My', 'Mz');
```



Fitting an Exponential to Max Spin Echo Amplitude

From the graph, determine following maximum spin echo amplitudes for each $?$:

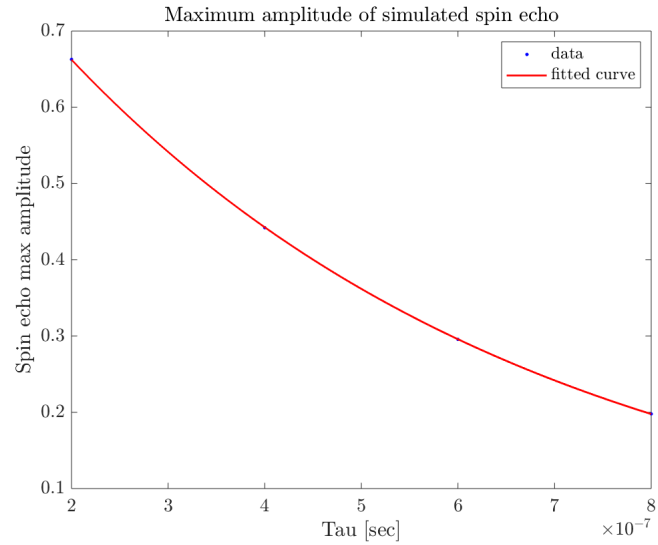
- $?$ = 0.2e-6 [s], Max Amplitude= 0.6629 [Bohr magnetons]
- $?$ = 0.4e-6 [s], Max Amplitude= 0.4421 [Bohr magnetons]
- $?$ = 0.6e-6 [s], Max Amplitude= 0.2956 [Bohr magnetons]
- $?$ = 0.8e-6 [s], Max Amplitude= 0.1979 [Bohr magnetons]

We put the results into 4x1 column vectors use the `fit` function to match an exponential and plot results:

```
tau = [0.2e-6, 0.4e-6, 0.6e-6, 0.8e-6]'; % Array of simulated tau (3x1) [s]
amp = [0.6629, 0.4421, 0.2956, 0.1979]'; % Array of maximum spin echo amplitude (3x1) [Bohr magnetons]
exp_fit = fit(tau, amp, 'exp1')
```

```
exp_fit =
     General model Exp1:
     exp_fit(x) = a*exp(b*x)
     Coefficients (with 95% confidence bounds):
       a =      0.9913  (0.9853, 0.9973)
       b =  -2.015e+06  (-2.031e+06, -1.999e+06)
```

```
% Plot results
clf();
plot(exp_fit, tau, amp);
hold on;
title('Maximum amplitude of simulated spin echo')
xlabel('Tau [sec]');
ylabel('Spin echo max amplitude');
```



Recovering T_2 from the Fit

Use the coefficients determined by the `fit` function to extract T_2:

```
coeff = coeffvalues(exp_fit);
T2_rec = -1 / coeff(2);
T2_rec
```

```
T2_rec = 4.9623e-07
```

Conclusion

We were able to numerically simulate a spin echo phenomenon using a $\pi/2 \longrightarrow \pi$ pulse sequence, and show that the spin echo shape could be changed by modifying pulse delay $?$. As expected, when we perform the pulses in this order, the $\pi/2$ pulse into xy causes dephasing due to magnetic inhomogeneity, which is then reversed by the $?$ sequence, allowing the spins to rephase after 2τ, which shows up as a spin echo.

We confirmed that spin echo amplitude decayed with changing $?$ by fitting an exponential function. The time constant T_2 was changed as compared to the value input into the experiment, with the spin echo T_2 being reduced by an approximate factor of 2, i.e.

$$T_{2,\,\text{echo}} \approx \frac{T_2}{2} \approx 0.5\,\mu s$$

This result was found to be robust in changes to in n and $?$, after rerunning this task for values of $\sigma = \{0.1, 0.01, 0.5\}$ and $n = \{100, 500\}$, so we extrapolate that the T_2 obtained by spin echo will always be reduced by a factor of 2. If they weren't independent of the fluctuation function, it would mean that we could not extrapolate results to experimental data where we have no idea what the fluctuation function is.

Task 4v1 - Exploring Spin Echo Data

Goal

Load spin-echo data and see that a plot of voltage vs time looks valid.

```
clearvars;
```

Loading Data Files

Load all the directories in the data folder:

```
cd ..
cd 'Task 4'
files = dir('NMR*.DAT') % Load all data in data folder
```

```
files = 30×1 struct
```

Fields	name	folder	date	bytes	isdir	datenum
1	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182127	0	7.3805e+05
2	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182132	0	7.3805e+05
3	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182083	0	7.3805e+05
4	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182105	0	7.3805e+05
5	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182190	0	7.3805e+05
6	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182036	0	7.3805e+05
7	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182035	0	7.3805e+05
8	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182037	0	7.3805e+05
9	'NMR1t4_0…	'/Users/kee…	'17-Sep-20…	182038	0	7.3805e+05
10	'NMR1t4_1…	'/Users/kee…	'17-Sep-20…	182031	0	7.3805e+05
11	'NMR1t4_1…	'/Users/kee…	'17-Sep-20…	182030	0	7.3805e+05
12	'NMR1t4_1…	'/Users/kee…	'17-Sep-20…	182031	0	7.3805e+05
13	'NMR1t4_1…	'/Users/kee…	'17-Sep-20…	182031	0	7.3805e+05
14	'NMR1t4_1…	'/Users/kee…	'17-Sep-20…	182032	0	7.3805e+05

?

```
% Confirm we can recover name from struct
files(1).name
```

```
ans = 'NMR1t4_01.DAT'
```

```
% Try loading a test file
filetest = importdata(files(4).name)
```

```
filetest = struct with fields:
          data: [100000×2 double]
      textdata: {'Time (S)'  'Ch1 Voltage (V)'}
    colheaders: {'Time (S)'  'Ch1 Voltage (V)'}
```

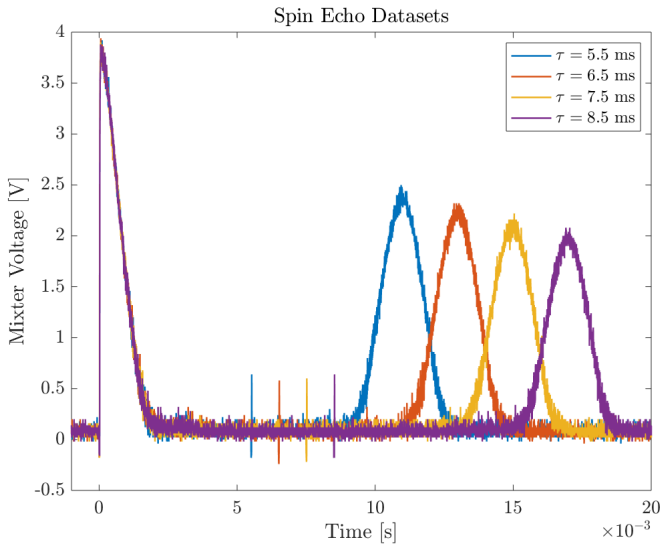
Plotting a Test Dataset

Time t of recording is the first data column, and voltage V is the second:

```
% Grab the data from this file and plot
ttest = filetest.data(:, 1);
vtest = filetest.data(:, 2);
plot(ttest, vtest);
hold on;
```

```
title('Spin Echo Datasets');
xlabel('Time [s]');
ylabel('Mixter Voltage [V]');
```

```
xlim([-0.001 0.02])
legend('$\tau=5.5$ ms', '$\tau=6.5$ ms', '$\tau=7.5$ ms', '$\tau=8.5$ ms');
```



Conclusion

Looks like correctly loaded spin-echo data. Clearly the second peak is the echo. Will need to figure out a way of removing the first peak if we want to simply detect max amplitude. Probably easiest to just clip the values before a certain time (approx 7 ms), since we know all peaks will be at a later time.

Task 4v2 - Improved T_2 Measurement

NOTE: REQUIRES CURVE FITTING TOOLBOX TO BE INSTALLED

Goal

Replicate the automated collection of amplitude peak data used in **Task 2v3** to obtain an estimate of T_2 from the spin echo:

```
clearvars;
```

Loading Data Files

```
cd ..
cd 'Task 4' % Go to data folder
files = dir('NMR*.DAT'); % Load all data in data folder
```

Extracting V_{max} from each Dataset

For each dataset: find where $t > 7$ ms and then find the amplitude peak V_{max} in this region (the spin echo max amplitude), then put it into an array containing each V_{max} (`vmax` is a `1x30` array):

```
n = length(files);      % Number of datasets explored
vmax = zeros(1, n);     % Vmax of each dataset

for i = 1:n
    data = importdata(files(i).name);
    t = data.data(:, 1);
    t_idx= find(t>7e-3); % Indexes where t > 7 ms
    v = data.data(t_idx, 2); % Voltage is second column
    vmax(i) = max(v); % Get voltage max minus voltage offset
end
```

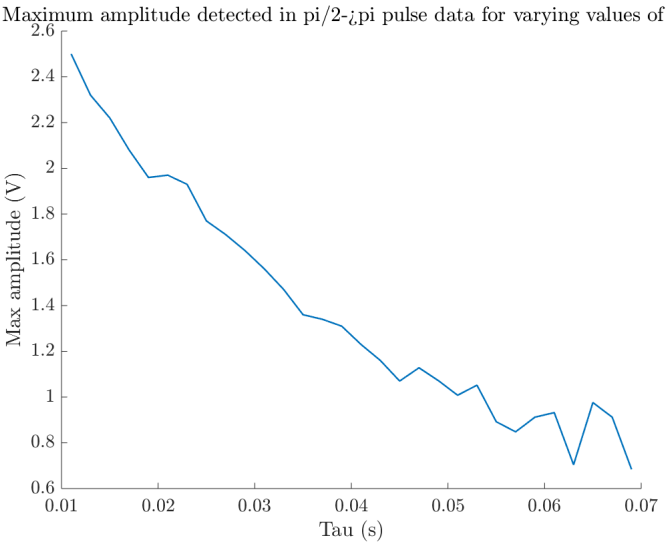
The $?$ for each dataset is given in the `descriptor.dat` file, we recreate the range as a `1x30` array `tau`:

```
tau = 5.5e-3:1e-3:34.5e-3;
tau = 2 .* tau;
```

Plotting Results

We can now plot V_{max} for varying values of $?$:

```
clf();
title('Maximum amplitude detected in pi/2->pi pulse data for varying values of tau');
xlabel('Tau (s)');
ylabel('Max amplitude (V)');
hold on;
plot(tau, vmax);
```



Fitting an Exponential to Extract T_2

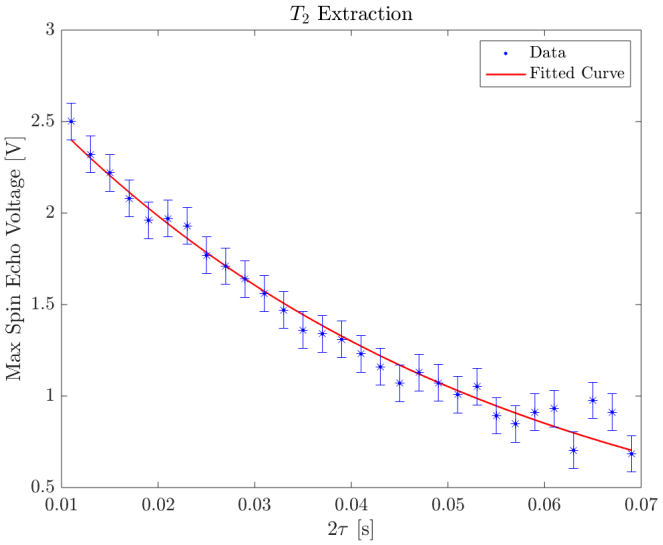
Use built-in `fit` function to fit an exponential (by defauly using non-linear least squares regression):

```
tau = tau'; % `fit` only works with column vectors
vmax = vmax';
exp_fit = fit(tau, vmax, 'exp1')  % Use built-in MATLAB exponential fit
```

```
exp_fit =
     General model Exp1:
     exp_fit(x) = a*exp(b*x)
     Coefficients (with 95% confidence bounds):
       a =       3.028  (2.905, 3.152)
       b =      -21.15  (-22.41, -19.88)
```

Plotting Fit

```
% Plot
clf();
plot(exp_fit, tau, vmax);
hold on;
error = 0.1 * ones(size(vmax));
errorbar(tau, vmax, error, 'b*');
hold on;
title('$T_2$ Extraction')
xlabel('$2\tau$ [s]');
ylabel('Max Spin Echo Voltage [V]');
legend('Data', 'Fitted Curve')
```



Extracting T_1 from the Fit

```
coeff = coeffvalues(exp_fit);
T2 = -1 / coeff(2)
```

```
T2 = 0.0473
```

Uncertainty Analysis for T_2

We can use the 95% confidence interval estimate given by the exponential `fit` function to estimate uncertainty in T_2, as in **Task 2v4**. This corresponds to a k factor of $k \approx 2$ ($k = 1.96$ in reality), so we just take difference between the CI interval endpoints and divide by 2, giving an uncertainty estimate of:

$$\sigma(T_1) = \frac{\frac{1}{19.88} - \frac{1}{22.41}}{2} = \pm 1.4 \text{ ms}$$

Conclusion

Using the exponential `fit` function, we deteremined the spin-echo value (with uncertainty) of spin-spin decay to be:

$$T_{2,\text{ spin}} = 24 \pm 1 \text{ ms}$$

Using the result from **Task 3v3** that $T_{2,spin}$ is the true T_2 reduced by a factor of 10, the true spin-spin decay is:

$$T_2 = 240 \pm 10 \text{ ms}$$

This result makes significantly more sense the result obtained from analysis of the $\pi \longrightarrow \pi/2$ pulse sequence in **Task 2**. T_2 is on the order of 10^1 to 10^2 ms as expected. Clearly inhomogeneity was affecting the T_2 result in **Task 2**, causing it to be signficantly lower. We can estimate how inhomogenous the field based on the by comparing the above result to the result from **Task 2** of $T_2 = 0.533$ ms. This gives an inhomogeneity reduction "ratio":

$$\text{Inhomogeneity ratio} = \frac{T_{2,\text{ spin}}}{T_{2,\text{ mixer}}} = \frac{240 \text{ ms}}{0.533 \text{ ms}} = 450\times$$

i.e. inhomogeneity reduced T_2 by a factor of 450. This echo results seen in **Task 3**, where we were able to significantly change T_2 by changing the variance $?$ of the magnetic field fluctation function we introduced. So if we were interested in determining a variance measure for the inhomogeneity of the field, we could go back to simulation and determine what value of $?$ gives a T_2 reduction of 450x. This phenomenon is most likely quantum mechanical in nature because it relates to nuclei interaction.

Improving Uncertainty

The result for uncertainty is reasonable. There is clearly some extracted data points that don't lie along the fitted exponential curve, especially at higher $?$. There is also uncertainty in the oscilloscope voltage reading from noise in the signal. Since voltage maximum is just read out from this noisy signal, we could definitely get results that include noise. One way to improve this would be to manually make sure the maximum picked is actually reasonable for each dataset, but this would take significant time. Another way would be to perform some filtering (e.g. Butterworth) to remove high-frequency voltage peaks before computing maximum of the dataset. One could also look at using different types of pulse sequences as mentioned in the manual.

END OF NOTEBOOK