ASSIGNMENT 12: SOLUTION

E. ARNOLD, M-S. LIU, R. PRABHU & C.S. RICHARDS
THE UNIVERSITY OF CAMBRIDGE

Written in XƎLATEX

SOLENOIDS

INTRODUCTION

In a helium spin-echo experiment, a beam of spin-polarised ^{3}He atoms passes through a magnetic field before being scattered off the sample. The scattered beam then goes through another field before striking the detector. In this tutorial, we will simulate an idealised version of these precession magnetic fields.

PRECESSION FIELD

Generating the magnetic field of produced by a realistic solenoid is a somewhat involved task, and is therefore out of the scope of this tutorial. Instead, we will focus on how to set up an idealised magnetic field without having to consider the geometry of any particular solenoid.

PARAMETERS OF THE SOLENOID

The only properties of our solenoid that we will explicitly define will be its dimensions and its position. For simplicity, we will consider a rectangular solenoid whose symmetry axes align with our coordinate axes, and we will place the centre of the solenoid at the origin. In the tasks we will construct a class the contain properties described. The methods within the class will include the functions that generate our precession magentic fields.

TASK

1. Construct a class (which we will call `solenoid`) with properties described above.
2. Write a function that generates a magnetic field with the following features:
   - It should point along the z-axis
   - It should have a uniform magnitude within the solenoid
   - It should have a magnitude equal to 0 outside of the solenoid

3. Write a function that generates a magnetic field with the following features:
   - It should point along the z-axis
   - It should have a uniform magnitude within the solenoid
   - There should be an exponential decay in the magnitude that starts from the two ends of the solenoid
4. Write a short script that calls the two functions that you have written, and shows that they work as desired

TIPS

1. In addition to the parameters of the solenoid, the `solenoid` class should have as one (or more) of its properties the set of points at which you wish to evaluate the magnetic field. There are many ways in which you may wish to allow a user to pass these points to the class, but if you are stuck consider how you may use the `meshgrid` function to help you
2. Consider giving your functions the following general structure:
   - Initialise a matrix that will contain the values of the magnetic field at every point that the user is interested in.
   - Use a `for` loop to run through each point in the matrix and use one or more `if` statements to check whether each point is inside or outside the solenoid, assigning a value to each point when necessary

SUMMARY

- Helium atoms pass through a magnetic "precession" field before and after striking a sample.
- The nature of the precession fields, particularly their transition time, determines the spin phase the atoms have after leaving the fields
- The analytic magnetic fields for realistic solenoids are difficult to find, and so this tutorial focusses on generating simple, idealised versions.

SPIN PRECESSION

PREREQUISITES

The chapter *Spin Echo* from the *Theory Handbook*.

INTRODUCTION

In previous tutorials, we have considered the scattering of helium-3 atoms from surfaces. These scattered atoms can be detected; thus we have previously discussed how we may interpret the scattering using helium-3 spin-echo spectroscopy (^{3}HeSE). Alongside these tutorials, we have produced models for the diffusion of adsorbates on a surface - which can too be measured with ^{3}HeSE. However, there is a very clear area of the research we have not directed any attention towards: simulating the inner workings of the ^{3}HeSE machine. In this tutorial, we will focus on simulating the spin precession of the atoms in the helium beam while travelling through magnetic fields.

SPIN

Motivation

In the UK, many second year courses cover the so-called "spin" of a particle in good detail. However, many do not. In addition to this, we aim for our package to be suitable for students having completed just one year of university education, and so must introduce the relevent concepts of spin physics here; we hope that this eases the difficulty that may be faced when attempting this tutorial, without overburdening you with information.

Formalism

How do we mathematically describe spin? Suppose that the general state of an object can be written as the state $|\psi\rangle$, which can be decomposed into two parts as

$$|\psi\rangle = |\mathbf{r}, t\rangle|S\rangle.$$

In this expression, $|\mathbf{r}, t\rangle$ is a ket representing the wavefunction of the object, and $|S\rangle$ is a ket representing the spin state of the object. What do we do with this new way of writing the state of the particle? Introducing an operator $\hat{\mathbf{S}}$ to represent the spin observable, then some basic comments can be made. These can be seen by investigating the behaviour of the spin operator on the state of the system; thus writing the action of the spin as:

$$\hat{\mathbf{S}}|\psi\rangle = \hat{\mathbf{S}}(|\mathbf{r}, t\rangle|S\rangle),$$

our first comment can be made. The spin of a particle does not depend on the position of the particle; this is no surprise - a fair comparison would be to assert that the charge of a particle does not changed when the position of the particle changes. Therefore the spin of a particle depends on different degrees of freedom: those intrinsic to the particle itself! How is this related to the expression we have written above? If the spin does not depend on spatial degrees of freedom, then the spin operator $|S\rangle$ does not act on the wavefunction's ket. Thus the action of the operator on the state of the object is

$$\hat{\mathbf{S}}|\psi\rangle = |\mathbf{r}, t\rangle\hat{\mathbf{S}}|S\rangle$$

(*i.e.* the spin operator acts on the spin state only). The second comment we wish to make is that the spin operator is a vector; it can be written as:

$$\hat{\mathbf{S}} = \hat{S}_x\mathbf{x} + \hat{S}_y\mathbf{y} + \hat{S}_z\mathbf{z},$$

which signals the spin state has spatial components! Note that this is distinct to the spin not having spatial dependence.

Our discussion so far has deviated very little from the most general abstractions we can make. The next steps we take are to make some assumptions about the spin of a particle; assuming that spin has the same behaviour as the angular momentum of a particle, we suggest that:

1. Denoting the operator returning the square magnitude of the spin as $\hat{S}^2$, the operators $\hat{S}^2$ and $\hat{S}_j$ are compatible, where j is any component.
2. If the spin is measured along z, then the eigenvalues of the operator $\hat{S}_z$ are $m_s\hbar$, where m_s is a number from a discrete set of values.
3. m_s takes one of two series of values: one is based on integer values in the range $-s \le m_s \le s$; the other is based on half integer values in the range $-s \le m_s \le s$. In both cases, s is a constant that represents the maximum eigenvalue of a measured spin component (divided by $\hbar$).
4. Using this same constant s, the eigenvalues of the square magnitude spin operator $\hat{S}^2$ are $s(s+1)\hbar^2$.

Following on from the assumptions listed above, the majority of the remainder of the theory is manipulating the mathematics that arises from these assumptions. For the remainder of the discussion in this section, we consider either a helium-3 atom, or an electron. For both of these particles, the value of s is $1/2$. Therefore the permitted values of m_s for both particles are $+1/2$ and $-1/2$. Note that this has a profound consequence: as the magnitude of the spin along z is always the same in magnitude when observed, and the overall magnitude of the spin is constant, we can assert that the radial component of the spin is also constant in magnitude. Recalling the eigenvalues of the square of the magnitude of the overall spin to be $s(s+1)\hbar^2$, it is evident that for our particles, the magnitude of the overall spin is $\sqrt{3}\hbar/2$. What can we infer from this? The spin of a particle always points at 55° to the z axis.

A Generic Spin State

Consider again that the spin of a particle, when measured along z, always points at 55° to the z axis. Noting that the magnitude is constant, this leads to two packets of particles: one where the number m_s is $1/2$, and one where the number m_s is $-1/2$. These are assigned the labels $|\uparrow\rangle$ and $|\downarrow\rangle$ respectively. If these states form a complete orthonormal basis set, then it is possible to write the spin state $|\chi(t)\rangle$ at a generic time t in the form:

$$|\chi(t)\rangle = \alpha|\uparrow\rangle + \beta|\downarrow\rangle,$$

where the coefficients α and β are parameters that may evolve with time. For this spin state to be normalised, the inner product with itself must be unity. This implies that:

$$\begin{aligned}\langle\chi(t)|\chi(t)\rangle &= (\alpha^*\langle\uparrow| + \beta^*\langle\downarrow|)(\alpha|\uparrow\rangle + \beta|\downarrow\rangle);\\ &= |\alpha|^2 + |\beta|^2;\\ &= 1.\end{aligned}$$

Moving from the first to the second line is trivial once it is realised that given the spin states are distinguishable, they are orthogonal. Orthonormality is then assumed from the definition, as the kets represent physical states.

SPIN PRECESSION

Direction of Spin

Recall that the generic spin state is

$$|\chi(t)\rangle = \alpha|\uparrow\rangle + \beta|\downarrow\rangle.$$

In real cartesian space, does the spin point along a direction? Following through with the mathematics suggests that the answer to this question is yes. Assume that the unit vector representing the direction of the spin is **n**. Defining θ as the angle from the vertical and ϕ as the azimuthal angle, then it is clear that the spin points along the direction

$$\mathbf{n} = \sin\theta\cos\phi\mathbf{x} + \sin\theta\sin\phi\mathbf{y} + \cos\theta\mathbf{z}.$$

If α and β can be expressed in terms of these constants, then the direction the spin points in exists. We will now attempt to solve for the coefficients α and β. The solutions are:

$$\alpha = e^{-i\phi/2}\cos(\theta/2);$$
$$\beta = e^{i\phi/2}\sin(\theta/2).$$

Finding these requires the use of more algebra than we wish to use in this document. Therefore we list the results without proof.

Time Evolution Operator

While all of the discussion so far does not account for the origin of spin, it does allow us to start to consider the *time evolution operator* that is central to the understanding of spin precession. The time evolution of any quantum state $|\psi\rangle$ is described by the Schrodinger equation

$$i\hbar\frac{\partial}{\partial t}|\psi\rangle = \hat{H}|\psi\rangle.$$

The solution to this is:

$$|\psi(t)\rangle = e^{-i\hat{H}t/\hbar}|\psi(0)\rangle,$$

where $e^{-i\hat{H}t/\hbar}$ is known as the time evolution operator. How is this relevant to spin precession? If we know the Hamiltonian, then we can construct the time evolution operator. The Hamiltonian for a spin interacting with a magnetic field is:

$$\hat{H} = -\gamma\hat{\mathbf{S}}\cdot\mathbf{B},$$

where γ is a constant. For a field in the z direction, this gives the coefficients α and β to evolve with time as:

$$\alpha(t) = e^{-i(\phi+\omega_0 t)/2}\cos(\theta/2);$$
$$\beta(t) = e^{-i(\phi+\omega_0 t)/2}\sin(\theta/2).$$

This is best derived with Pauli spin matrices.

TASK

1. Before considering the HeSE experimental setup as a whole, it would be prudent to first write a function to model the precession of atomic spin through a single solenoid. It will be useful to consider not only the central axis but any set of points within the solenoid, as well as a range of atomic speeds. Such a function, called `precess()` is given in the appendices as a listing.
   (a) Read the function `precess()` and compare it to the process described in the theory handbook. Note in particular the way in which the lateral points `P` are used in this function. It is not in fact necessary to consider the location of these points - all that is relevant is the magnetic field `B` along the z-axis at each point modelled. Thus, the choice of coordinate system is irrelevant since the locations and number of points `P` are all encoded in the field `B`.
   (b) For all the fields generated in the previous assignment, simulate and plot the variation of the x, y and z spin-components of helium-3 atoms against distance through the solenoid. You may use the following parameters:
   - an initial spin of `Si = [0 1 0]'`,
   - an average speed of `V0`=768 $\mathrm{m\,s^{-1}}$,
   - a range of speeds `DV`=0.1 $\mathrm{m\,s^{-1}}$,
   - a number of considered speeds `nV`=10,
   - the field `B` defined only along the central axis,
   - a set of points `Z` as used in the construction of `B`.

   (c) Experiment with fields `B` defined along non-central axes in the solenoid.
2. Now we have built up some familiarity with the `precess()` function, we can move on to considering the arrangement of solenoids used in HeSE experiments. The three main stages of this process are as follows:
   (a) The atoms pass through the first solenoid and precess, acquiring different phases.
   (b) The atoms then collide with the surface and change direction. A 'spin rotator field' is used in this stage to compensate for this change in direction by rotating the atomic spins to lie in the new, rotated coordinate system.
   (c) The atoms pass through the second solenoid which has its field in the opposite direction and acquire an additional phase change that rotates the spin back to its original direction.

   If the collision involved any momentum/energy transfer with the surface, the spins would be altered and wouldn't rotate back to their original direction, producing a measurable change. We wish to model the elastic and specular case where no such transfer occurs. The code for this is given in the appendices. Try to understand how it functions before you attempt the next task.

3. We now have a complete code that simulates the precession of atomic spins across a range of speeds and transverse positions for a given magnetic field.
   (a) For a variety of fields, plot the variation in spin components as a function of distance through the solenoid along the central axis for different speeds (*i.e.* separately plot each of the three matrices `Spins(1,i0,:,:)`, `Spins(2,i0,:,:)` and `Spins(3,i0,:,:)` against the vector `Disps` where `i0` is the index in `B` corresponding to the central axis). Try this both with and without a spin rotator field. What is different about the range of output spins for each speed in each case? How does this vary if you assume the spin field provides uniform rotation for all speeds rather than a slight variance as it does in practice? Why is this variance not significant in HeSE experiments? Why might that not be the case if the entire wavelength intensity matrix is to be measured?
   (b) Repeat the above, but instead of plotting the curves for different speeds, try plotting them for the same speed along different axes.
4. Suppose that in a given experiment we choose to measure only outgoing atoms whose spins lie along the y-axis ([0 1 0]). In the previous assignment we generated two key magnetic fields of interest for the solenoids: a field with a fast transition and a field with a slow transition.
   By plotting bar charts of the y-component of the spin against both speed and the distance of axes from the centre, observe how the speed of the solenoid field transition affects the spread of spin components. Which is most suitable for the HeSE apparatus?

SUMMARY

In this tutorial, you should have learned how the spin of a helium-3 particle precesses in the magnetic fields of a helium-3 spin-echo spectrometer.

SOLUTIONS

SOLENOIDS: MAIN TASKS

Task 1: Question

Construct a class (which we will call `solenoid`) with properties described above.

Task 1: Solution

The `solenoid` class should be initialised in the usual way. The properties that we need to define are the three dimensions of the solenoid (denoted in our code `l`, `lx` and `ly`) and the points at which we wish to evaluate the magnetic field. There is no right or wrong format in which these points should be represented, but the way that we have done it here is as follows:

- The x and y coordinates are to be represented by a `meshgrid` that has been reshaped to form a row vector. Thus, if the vectors `X` and `Y` were to contain all of the x and y coordinates at which we wish to sample the field, then the object properties `x` and `y` would be given by `x=reshape(meshgrid(X, Y), [nP, 1])` and `y=reshape(meshgrid(X,Y), [nP, 1])`, where `nP = nummel(X)` is the number of grid points in either of the `meshgrids`.
- The z coordinates are to be represented by a row vector.

If you are still unsure about the form that these properties should take, the code that is used to create a `solenoid` object is presented in the solution to task 4.

```
classdef Solenoid < handle
    %Producing a set of simple solenoid fields to be used for
        spin precession simulations


    %Defining the properties of the Solenoid object (whose
        values are to be defined in the main script)
    properties

        %Length of solenoid in z-direction
        l

        %Length of solenoid in x-direction
        lx

        %Length of solenoid in y−direction
        ly

        %x and y are meshgrids (containing the coordinates in
            the x−y plane at which to evaluate the field)
            reshaped as row vectors
        x
        y

        % Row vector of z−coords at which to evaluate the field
        z

    end
```

If we wish to use any of these properties as variables in the functions that we are about to write, then we also need to assign them to an object:

```
%Defining the properties of the class as variables that can be
    used by other functions

function obj = Solenoid(l, lx, ly, x, y, z)
    obj.l = l;
    obj.lx = lx;
    obj.ly = ly;

    obj.x = x;
    obj.y = y;
    obj.z = z;
end
```

Task 2: Question

Write a function that generates a magnetic field with the following features:

- It should point along the z-axis
- It should have a uniform magnitude within the solenoid
- It should have a magnitude equal to 0 outside of the solenoid

Task 2: Solution

As described in tip 2, we will first initialise the matrix that is to hold the values of the magnetic field at all of the points that the user is interested in. The dimensions of the matrix, which here is denoted `Bz_box`, are chosen such that each row contains the values of the field for a particular point in the x-y `meshgrid` and each column contains the values for a particular z-coordinate:

```
function [Bz_box] = box_z(obj)
    %Creating a Bz field that is equal to 1 inside a rectangular
         solenoid and 0 elsewhere

    %The centre of the solenoid is at x=y=z=0
```

```matlab


    %Creating a 2D matrix, Bz_box, where Bz_box(ij, k) contains
        the value of the field at (x(ij), y(ij), z(k))
    Bz_box = zeros([size(obj.x, 2) size(obj.z, 2)]);
```

We then loop over each row (tracked by the index `ij`) and each column (tracked by the index `k`) and check whether each point is within the solenoid; if it is, then it is given a value of 1 and otherwise, it is left with its default value of 0:

```matlab
    for ij = 1:size(obj.x,2)
        for k = 1:size(obj.z,2)
            if obj.x(ij) < (obj.lx/2) && obj.x(ij) > -(obj.lx/2)
                 && obj.y(ij) < (obj.ly/2) && obj.y(ij) > -(obj.
                ly/2) && obj.z(k) < (obj.l/2) && obj.z(k) > -(
                obj.l/2)
                Bz_box (ij,k) = 1;
            end
        end
    end
end
```

Task 3: Question

Write a function that generates a magnetic field with the following features:

- It should point along the z-axis
- It should have a uniform magnitude within the solenoid
- There should be a decay in the magnitude that starts from the two ends of the solenoid

Task 3: Solution

The first step is to copy over all of the code from the `box_z` function, because this produces a uniform field within the solenoid as required. Note, however, that when defining the function there needs to be an additional input parameter, `axial_decay_const`, that gives the exponential decay constant that we are going to put in later:

```matlab
function [Bz_axial_decay] = axial_decay(obj, axial_decay_const)
    %Creating a Bz field that is equal to 1 within the solenoid
        and then decays exponentially from the edges

    %The field is still 0 when you go outside of the solenoid in
         the x and y directions

    %The centre of the solenoid is at x=y=z=0


    %Creating a 2D matrix, Bz_axial_decay, where Bz_axial_decay(
        ij, k) contains the value of the field at (x(ij), y(ij),
         z(k))
    Bz_axial_decay = zeros([size(obj.x, 2) size(obj.z, 2)]);

    %Adding the uniform field inside the solenoid

    for ij = 1:size(obj.x,2)
        for k = 1:size(obj.z,2)
           if obj.x(ij) < (obj.lx/2) && obj.x(ij) > -(obj.lx/2)
               && obj.y(ij) < (obj.ly/2) && obj.y(ij) > -(obj.ly
               /2) && obj.z(k) < (obj.l/2) && obj.z(k) > -(obj.l
               /2)
                Bz_axial_decay (ij,k) = 1;

           end
        end
    end
```

We are then going to add some sort of decay that starts from the two ends of the solenoid. Here, we have chosen to use an exponential decay, but you could also use any other function that decays to zero.

We follow a similar process of using two nested `for` loops to loop over each point in the magnetic field matrix, and then we use two `if` functions, the first one to check if a point is outside of the solenoid in the positive z-axis and the second one to check if a point is outside of the solenoid in the negative z-axis. In either case, we assign an exponentially decaying magnetic field to those points that are outside of the solenoid on either end:

```
%Adding the exponential decay outside the solenoid

    for ij = 1:size(obj.x,2)
        for k = 1:size(obj.z,2)
           if obj.x(ij) < (obj.lx/2) && obj.x(ij) > -(obj.lx/2)
               && obj.y(ij) < (obj.ly/2) && obj.y(ij) > -(obj.ly
               /2) && obj.z(k) > obj.l/2
                Bz_axial_decay (ij,k) = exp(-abs(obj.z(k) - obj.
                    l/2)*axial_decay_const);

           end

           if obj.x(ij) < (obj.lx/2) && obj.x(ij) > -(obj.lx/2)
               && obj.y(ij) < (obj.ly/2) && obj.y(ij) > -(obj.ly
               /2) && obj.z(k) < -(obj.l/2)
                Bz_axial_decay (ij,k) = exp(-abs(obj.z(k) + obj.
                    l/2)*axial_decay_const);

           end
        end
    end


end
```

Task 4: Question

Write a short script that calls the two functions that you have written, and shows that they work as desired

Task 4: Solution

We start by defining the solenoid parameters

```
%Creating a Solenoid object and defining its properties (see
    Solenoid class)

%Defining the solenoid parameters

%Length
l = 5;

%x dimension
lx = 2;

%y dimension
ly = 3;
```

We then need to define the points at which we wish to find the value of the magnetic field. The precise format that this should take is given in the solution to task 1, and this code simply implements that:

```
%Defining three row vectors that give the x, y and z coordinates
     of the points where we wish to know the value of the
    magnetic field
X = linspace(-0.5, 0.5, 10);
Y = linspace (-0.5, 0.5, 20);
Z = linspace (-10, 10, 100);

%Making the x and y coordinates into a meshgrid
[x,y] = meshgrid(X,Y);

%The number of points in the meshgrids
nP = numel (x);

%Reshaping the meshgrids to make them into row vectors
x_linear = reshape (x, [1, nP]);
y_linear = reshape (y, [1, nP]);
```

We then create a `solenoid` object:

```
%Creating a solenoid object
s=Solenoid(l, lx, ly, x_linear, y_linear, Z);
```

In this example, we are going to generate a magnetic field that decays exponentially from the ends of the solenoid, using the `axial_decay` function produced in task 2. This function takes as one of its input parameters the decay constant, which we will set to 0.2.

```
%Generating a B field with an exponential decay
[Bz] = axial_decay(s,0.2);
```

To make sure that this has produced the desired result, we plot the magnetic field as a function of z, making sure to choose x and y coordinates that are within the solenoid:

```
%Plotting Bz field as function of z (for x and y coordinates
    within solenoid)
for i = 1:size(Bz,2)
    hold on
    plot (Z(i), Bz(2,i), 'linestyle', 'none', 'marker', 'o')
end
```

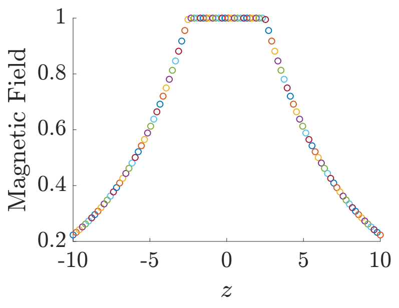

Figure 1 | A magnetic field that decays exponentially from the edges of a solenoid of length 5, plotted as a function of z

SPIN PRECESSION: MAIN TASK

Task 1: Questions

Before considering the HeSE experimental setup as a whole, it would be prudent to first write a function to model the precession of atomic spin through a single solenoid. It will be useful to consider not only the central axis but any set of points within the solenoid, as well as a range of atomic speeds. Such a function, called `precess()` is given in the appendices as a listing.

(a) Read the function `precess()` and compare it to the process described in the theory handbook. Note in particular the way in which the lateral points `P` are used in this function. It is not in fact necessary to consider the location of these points - all that is relevant is the magnetic field `B` along the z-axis at each point modelled. Thus, the choice of coordinate system is irrelevant since the locations and number of points `P` are all encoded in the field `B`.

(b) For both a fast and slow transition to a uniform field, simulate and plot the variation of the x, y and z spin-components of helium-3 atoms against distance through the solenoid. You may use the following parameters. Recall from the theoretical background that 'fast transition' implies that the rate of change of the longitudinal (z-component of) field strength is greater than the average speed of incident atoms.

- an initial spin of `Si = [0 1 0]'`,
- an average speed of `V0`=768 m s^{-1},
- a range of speeds `DV`=0.1 m s^{-1},
- only a single considered speed `nV`=1,
- the field `B` defined only along the central axis,
- a set of points `Z` as used in the construction of `B`.

(c) Experiment with a range of speeds and precession along non-central axes in the solenoid.

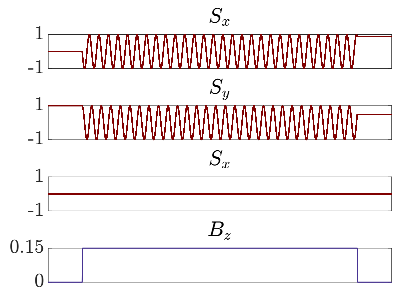

Figure 2 | Precession of all three spin components through an fast-transitioning uniform magnetic field

Task 1: Solutions

This question is intended as an exploratory exercise, however in general the precession inside the solenoid should be of the following form for the simple abrupt uniform field:

Note that for a slower transition to a uniform field, the precession starts earlier at a slower rate.

Other fields may also be of interest but we only give the above two examples as graphics. The code used to generate the above figures is given below:

We begin by initialising the parameters of atoms incident on the solenoid, as well as the geometric parameters of the solenoid:

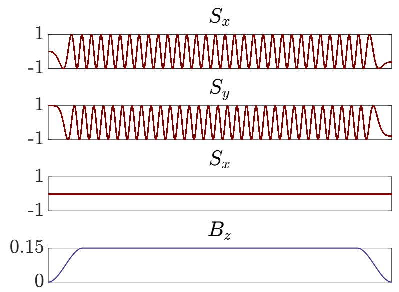

Figure 3 | Precession of all three spin components through a slowly-transitioning uniform magnetic field

```
%----------------------------------------------------------------
% initial atom parameters
%----------------------------------------------------------------

% average speed of He-3 atom through solenoid (along x-axis) in
    m/s
V0=768;

% maximum fractional difference of atom speeds from average
DV=0.1;

% number of velocities simulated at each point
nV=1;

% initialise vector of velocities
V = linspace(V0-(V0*DV),V0+(V0*DV), nV);

% initial spin polarisation (parallel to the y-axis)
spin=[0 1 0]'; Smag = sum(spin.^2);

%----------------------------------------------------------------
%----------------------------------------------------------------




%----------------------------------------------------------------
% solenoid parameters
%----------------------------------------------------------------

% length of solenoid /m
L = 0.005;

% width/height of solenoid /m
Lx = 0.05;
Ly = 0.05;

%----------------------------------------------------------------
%----------------------------------------------------------------
```

We then grid out a set of points along the central axis:

```
%----------------------------------------------------------------
% solenoid dimensions of interest
%----------------------------------------------------------------

% SETUP:

% length of the region of interest (symmetrical about solenoid)
Zmax = 5/4*L;

% number of Z-points to simulate
nZ = 750;



% CALCULATION:

% initialise vector of Z-points
Z = linspace(0,Zmax,nZ)';

% For central axis x=y=0 on the meshgrid (only one axis)
```

```
X = 0; Y = 0;

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

The parameters are then passed into the `Solenoid` class, allowing a field to be generated:

```
%-----------------------------------------------------------------
% solenoid B-field
%-----------------------------------------------------------------

s = SolenoidNew(L, Lx, Ly, X', Y', Z'-Zmax/2);

[Bx,By] = s.box_xy();
Bz = s.box_z();
% Bz = s.smoothstep_z((Z(end) - L)/2);

% intialise set of vectors for B-field at each point along the
    length
B=zeros([3 nP nZ]);
B(1,:,:) = Bx; B(2,:,:) = By; B(3,:,:) = Bz;


B = 0.15 .* B;

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

The initial spins can then be precessed through the solenoid:

```
%-----------------------------------------------------------------
% precess spins through B-field
%-----------------------------------------------------------------

% precess spin through solenoid
Si = repmat(spin, [1 nP nV]);
Sf1 = precess(Si, Z, V, B);

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

And a graph of the spin components against the distance through the solenoid plotted:

```
%-----------------------------------------------------------------
% plot spin components against z-distance for diff speeds
%-----------------------------------------------------------------
figure();
iP = 1;

% plot x-component of spin against distance through solenoid
subplot(4,1,1); plot(Z, squeeze(Sf1(1,iP,:,:)));
xlim([Z(1) Z(end)]); ylim([-Smag Smag]); ylabel('$S_x$');

% plot y-component of spin against distance through solenoid
subplot(4,1,2); plot(Z, squeeze(Sf1(2,iP,:,:)));
xlim([Z(1) Z(end)]); ylim([-Smag Smag]); ylabel('$S_y$');

% plot z-component of spin against distance through solenoid
subplot(4,1,3); plot(Z, squeeze(Sf1(3,iP,:,:)));
xlim([Z(1) Z(end)]); ylim([-Smag Smag]); ylabel('$S_z$');
```

```

% plot z-component of spin against distance through solenoid
subplot(4,1,4); plot(Z, Bz');
xlim([Z(1) Z(end)]); ylim([0 1]);  ylabel('$B_z$'); xlabel('$z$'
    );
%------------------------------------------------------------------
%------------------------------------------------------------------
```

This behaviour precisely matches that predicted in the theoretical background.

The trajectory for different speeds can easily be visualised (for a slow transition here) by setting `nV=4`:

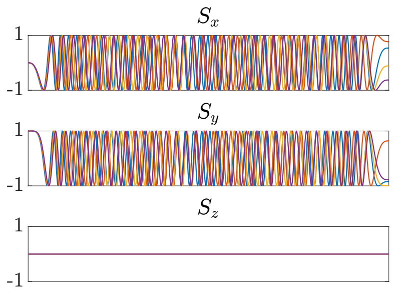

Figure 4 | Precession the spin of four atoms travelling along the central axis of a (slow transition) solenoid at different speeds.

It is of note that the output spin components are distributed in the range $[-1, 1]$.

In order for off-axis trajectories to behave differently, you will need to use a field which changes strength or direction depending on the lateral position. The most sensible lateral change in field is likely a radial Gaussian decay in the $z$-component.

A slow-transition Gaussian decay is given below. We first initialise a set of points of interest:

```
%------------------------------------------------------------------
% solenoid dimensions of interest
%------------------------------------------------------------------

% SETUP:

% length of the region of interest (symmetrical about solenoid)
Zmax = 5/4*L;

% number of Z-points to simulate
nZ = 750;


% maximum allowed beam radius
a = min(Lx, Ly)/2;

% number of radii to simulate
nr = 5;


% number of angles to simulate -- in range [0,2*pi)
ntheta = 6;


% CALCULATION:

% initialise vector of Z-points
Z = linspace(0,Zmax,nZ)';

% calculate spacing between Z-points (we wish to keep this
    consistent throughout the simulation
dZ = Z(2)-Z(1);


% set of radii around central beam in range [0,a)
r = linspace(0,a,nr+1); r(end) = [];

```

```

% set of angles around central beam in range [0,2*pi)
theta = linspace(0,2*pi,ntheta+1); theta(end) = [];



% meshgrid of all lateral points on the beamline/cylinder of
    atoms (the meshgrid here isn't actually important, just each
     meshgrid index corresponds to the field value in B at the
    same index (see below)
[R, Theta] = meshgrid(r, theta);

nP = numel(R); % number of points in meshgrid



% extract Cartesian grid for use with Solenoid class
X = reshape(R.*cos(Theta), [nP 1]); Y = reshape(R.*sin(Theta), [
    nP 1]);

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

We then use the `Solenoid` class to generate the smoothly-varying field then scale each point with a radially-varying Gaussian function

```
%-----------------------------------------------------------------
% solenoid B-field
%-----------------------------------------------------------------

s = Solenoid(L, Lx, Ly, X', Y', Z'-Zmax/2);

[Bx,By] = s.box_xy();
%Bz = s.box_z();

% the smoothing length is chosen to maximise smoothing over the
    considered region Z
Bz = s.smoothstep_z((Z(end) - L)/2);

% the standard deviation is chosen to be a third of the radius
    of the solenoid so that the field strength is close to zero
    at the edges
radial_decay_const = 3 * 1/(2*(min(Lx,Ly)/2)^2);

% scale the z-component of the field along the entire solenoid (
    and outside it)
for ij = 1:nP
    for k = 1:nZ
        Bz(ij,k) = Bz(ij,k) * exp(-(X(ij)^2)*radial_decay_const
            - (Y(ij)^2)*radial_decay_const);
    end
end

% intialise set of vectors for B-field at each point along the
    length
B=zeros([3 nP nZ]);
B(1,:,:) = Bx; B(2,:,:) = By; B(3,:,:) = Bz;

% scale field
B = 0.15 .* B;

%-----------------------------------------------------------------
```

```
%------------------------------------------------------------------
```

Plotting the spin components for a range of different trajectories (and plotting the z-component of the field as a `surf()` object across the x- and z- axes) gives the following figure:

The spread in the spin components of the trajectories is qualitatively very similar to the spread of different speeds in a uniform field, however this entirely depends on the field as well as the set of input trajectories (in the above case, the trajectories are uniformly distributed over a cylindrical beam).

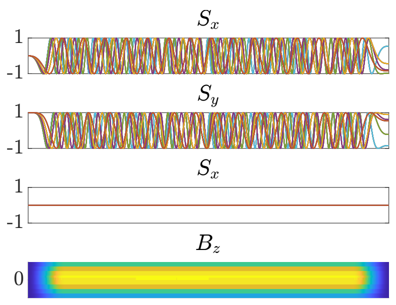

Figure 5 | Precession of atoms travelling along a set of axes distributed uniformly over a cylindrical path through a radially-decaying, slow-transitioning magnetic field (at the same speed).

Task 2: Questions

Now we have built up some familiarity with the `precess()` function, we can move on to considering the arrangement of solenoids used in HeSE experiments. The three main stages of this process are as follows:

1. The atoms pass through the first solenoid and precess, acquiring different phases.
2. The atoms then collide with the surface and change direction. A compensating 'spin rotator field' is used in this stage to compensate for this change in direction by rotating the atomic spins to lie in the new, rotated coordinate system.
3. The atoms pass through the second solenoid which has its field in the opposite direction and acquire an additional phase change that rotates the spin back to its original direction.

If the collision involved any momentum/energy transfer with the surface, the spins would be altered and wouldn't rotate back to their original direction, producing a measurable change. We wish to model this elastic and specular case where no such transfer occurs (*i.e.* the spin directions should be restored).

The code for this is given in the appendices. Try to understand how it functions before you attempt the next task.

Task 2: Solutions

Trivial.

Task 3: Questions

We now have a complete code that simulates the precession of atomic spins across a range of speeds and transverse positions for a given magnetic field.

(a) For a variety of fields, plot the variation in spin components as a function of distance through the solenoid along the central axis for different speeds (*i.e.* separately plot each of the three matrices

`Spins(1,i0,:,:)`, `Spins(2,i0,:,:)` and `Spins(3,i0,:,:)` against the vector `Disps` where `i0` is the index in `B` corresponding to the central axis). Try this both with and without a spin rotator field. What is different about the range of output spins for each speed in each case? How does this vary if you assume the spin field provides uniform rotation for all speeds rather than a slight variance as it does in practice? Why is this variance not significant in HeSE experiments?

(b) Repeat the above, but instead of plotting the curves for different speeds, try plotting them for the same speed along different axes. You will note that the trajectories all follow the same path in a uniform field regardless of the transition speed. Try plotting the same curves for a field which exhibits Gaussian decay in the z-component radially from the central axis.

Task 3: Solutions

Once again, this question is intended as an exploratory exercise. Figures are given for both the abrupt and smooth uniform fields.

For the fast transition fields we obtain the following figures:

The qualitative behaviour is the same for the slow transition as it is for the fast transition (see the next question for more quantitative analysis). Note in particular that without the SR field the spins of different speeds/trajectories don't converge back to their original direction. In the ideal case, they converge precisely back to the initial spin directions.

The uniform field causes the spins to converge back to their original direction on average, however there is spread due to both differences in speeds lateral positions. For a scattering angle of 45 | degree and a velocity spread of $\pm 10\,\mathrm{m\,s^{-1}}$, the variation in the precession angle due to a uniform SR field is around 4.5°. For projections of the rotated direction in the x and y directions this variation is largely negligible, hence the comparatively small spread in spin components seen above.

The qualitative behaviour is the same for different lateral positions as it is for different speeds. A non-uniform distribution of trajectories may generate an uneven distribution of spins throughout the motion, however.

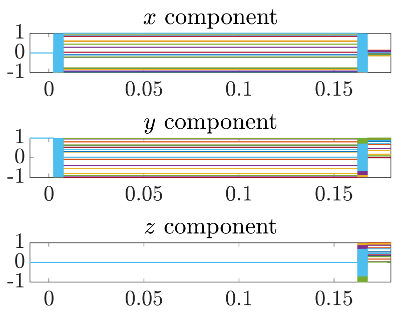

Figure 6 | Spin components for a fast solenoid transition with no SR field.

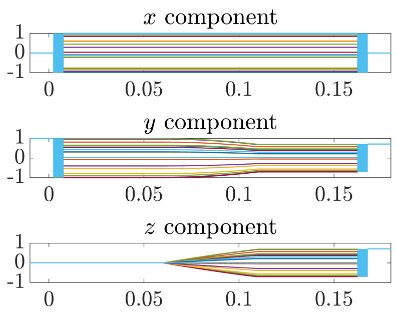

Figure 7 | Spin components for a fast solenoid transition with an idea SR field.

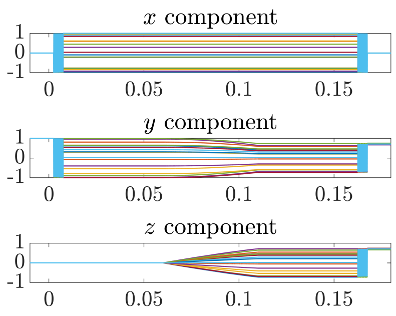

Figure 8 | Spin components for a fast solenoid transition with a uniform SR field.

Task 4: Questions

Suppose that in a given experiment we choose to measure only outgoing atoms whose spins lie along the y-axis ([0 1 0]). In this task you should consider two fields: a field with a fast transition and a field with a slow transition. The z-components of both fields should follow a Gaussian distribution for off-axis trajectories.

By plotting scatter graphs of the y-component of the spin against both speed and the x-coordinate of trajectories (this will be the same as the y-distribution due to symmetry), observe how the speed of the solenoid field transition affects the spread of spin components. Which is most suitable for the HeSE apparatus?

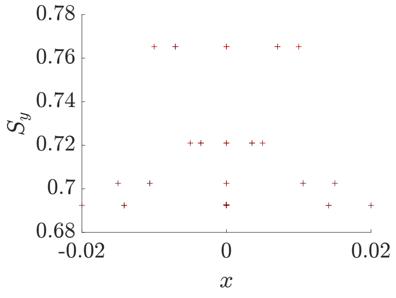

Figure 9 | Scatter graph of the y spin components against the x-components of their trajectories (at an arbitrary speed of 691.2 m s^{-1}) where the solenoid field transition is slow.

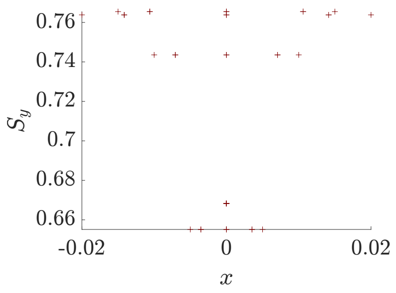

Figure 10 | Scatter graph of the y spin components against the x-components of their trajectories (at an arbitrary speed of 691.2 m s^{-1}) where the solenoid field transition is fast.

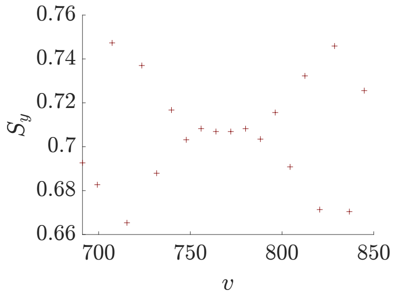

Figure 11 | Scatter graph of the y spin components against their corresponding speeds (for the central axis) where the solenoid field transition is slow.

Task 4: Solutions

For any measurement, there is variation in the spin components due to both different velocities and lateral positions of atoms, since it is impossible to create a perfectly narrow, monochromatic beam. Thus deviations will be introduced, either by inhomogeneity in the solenoid fields or the spread of precession angles in the SR field.

You will note that the spin components at different trajectories form `nr` discrete layers of `ntheta` points (some coincide). This clearly arises due to the choice of coordinate system and field. Since the field strength follows a symmetrical Gaussian distribution with respect to the radial distance from the central axis, all the points which are at a certain radius will precess in the same through the apparatus in the same way. The speed of transition will have no effect on the spread of trajectories since this is dependent only on the variation in field strength with lateral displacement, not with axial displacement.

Explaining the shape of the distribution for speeds is more complex. The points are clearly rotationally-symmetric about the average speed, but the shape of the curve can't really be determined from such a small set of samples. Using `nV=200` and `nr=ntheta=1` we can produce the figure in the margin. This is likely a very surprising result! Indeed, it approximately follows $S_y = 1/\sqrt{2} - v\cos v$. The same curves for the x and z spin components are $S_y = v\sin v$ and $S_z = 1/\sqrt{2} + v\cos v$ respectively. More information can be obtained by considering the curve without the spin rotator field or simply with the ideal field.

Note that the oscillation present that is present without the SR field is completely removed in the ideal case, leaving only the variation due to the solenoid fields. Thus we should only consider the plots with the ideal SR field when comparing fast/slow transitions. With the uniform field (plotted previously), there is no oscillation at the average speed since the SR field rotates atoms travelling at the average speed through the correct angle. However, for more extreme speeds the range of angles introduced by the uniform precession through the SR field becomes non-negligible and the oscillation returns.

The question remains as to the cause of the oscillation without the presence of the SR field. The cause is, unsurprisingly, the scattering geometry. Since the spins leaving the first solenoid have a range of different directions for different speeds, when they scatter from the sample surface, these spins are resolved differently into the new basis, thereby appearing to precess by different amounts than would be expected in the second solenoid. The difference in y-component varies sinusoidally with the incident angle of the spins (this can easily be verified diagramatically) in the y-z plane, and thus exhibits oscillatory behaviour for different speeds.

That discussion aside, we can now address the question of whether the solenoid fields should be fast or slow. Running the simulation for both fields (as given in the code in the appendices) with `nV=200` and `nr=ntheta=1` gives total ranges of $\pm 6.9500 \times 10^{-14}$ N m s and $\pm 4.9183 \times 10^{-14}$ N m s for the slow and fast transitions respectively. Thus, we can conclude that fast transitions are better for the solenoid fields. However, as has been demonstrated above, the dominating effect is the variation due to the spread of precession angles produced by the SR field (which was assumed to have a fast transition in our code).

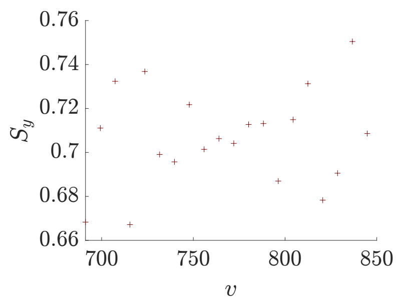

Figure 12 | Scatter graph of the y spin components against their corresponding speeds (for the central axis) where the solenoid field transition is fast.

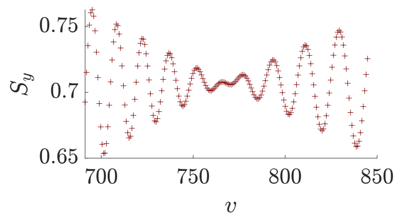

Figure 13 | Scatter graph of the y spin components against their corresponding speeds (for the central axis) where the solenoid field transition is fast. A uniform spin rotator field is applied.

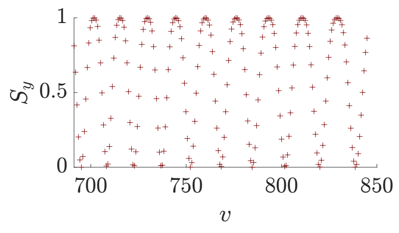

Figure 14 | Scatter graph of the y spin components against their corresponding speeds (for the central axis) where the solenoid field transition is slow. No spin rotator field is applied.

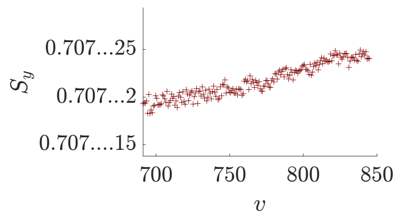

Figure 15 | Scatter graph of the y spin components against their corresponding speeds (for the central axis) where the solenoid field transition is fast. An ideal spin rotator field is applied.

APPENDIX

PRECESS.M

```
function [Sf] = precess(Si, Z, V, B)
% precondition: Si = initial spin of all atoms at each of the na different speeds and different points P
%               Z = the set of points along the axis of the solenoid
%               B = matrix of the magnetic field at each point (P,Z) in the solenoid
%               V0 = average speed of incident atoms
%               DV = fractional spread in speeds of incident atoms
%               na = number of atoms considered
%
% postcondition: Sf = set of final spins for each speed and lateral point P
%

    % ----------------------------------------------------------------
    % set up dynamic variables of system
    % ----------------------------------------------------------------

    % number of Z-points within solenoid
    nZ = length(Z);
    nP = size(B,2);
    nV = length(V);

    % produce a linearly-varying set of speeds for each atom in the given range
    tmp(:,1,1) = V;
    % repeat this over the extent of the beam, i.e. constant speed since particles are uncharged
    V = repmat(tmp(:,1,1), [1 nP nZ]);

    % calculate the time taken for each atom to travel each considered distance => function t(d)
    t = zeros(size(V));
    t(:,1,:) = reshape(repmat(Z', [nV 1]), [nV 1 nZ])./V(:,1,:);
    % repeat this over the extent of the beam, i.e. constant speed since particles are uncharged
    t = repmat(t(:,1,:), [1 nP 1]);

    % differentiate t(Z) once wrt. Z to obtain the time interval
```

```
     dt taken to travel between each considered distance
    along the length
dt=diff(t,1,3);
dt(:,:,end+1)=dt(:,:,end);

% normalise spin to get initial orientation (unit vector):
% initial spins should all have the same magnitude...
Simag = sqrt(sum(Si(:,1).^2, 1));

% ...but possibly different directions for each speed
Sidir=Si./(sqrt(sum(Si.^2))+eps);

%
    ---------------------------------------------------------------

%
    ---------------------------------------------------------------







%
    ---------------------------------------------------------------

% calculate Larmor frequency
%
    ---------------------------------------------------------------


% gyromagnetic ratio for helium 3
gamma=2.0378e8; %rad/sec;

% calculate Larmor frequency for atoms of all speeds b/c
    independent of speed
w=gamma*sqrt(sum(B.^2,1));

% magnitude of B
Bmag=sqrt(sum((B.^2),1));

% normalise B since we only needed the magnitude for the
    Larmor frequency (eps rounds to the nearest floating-
    point number)
Bdir=B./(eps+repmat(Bmag, [3 1 1]));

%
    ---------------------------------------------------------------

%
    ---------------------------------------------------------------







%
```

```
    ----------------------------------------------------------------
% precess spin vectors of atoms along the length of the
    field
%
    ----------------------------------------------------------------

% rotation matrix for precessing spin at point in the
    solenoid
R=zeros(3,3,nP,nZ);

% final spin direction for each atom
spinvec1=zeros(3,nP,nZ,nV);

% matrix of *change* in precession angle in travelling
    between each considered distance, for each atom
dphimat=repmat(w, [nV 1 1]).*dt;

% for each velocity iV...
for iV=1:nV
    % change in angle phi in each step along the length for
        atom in1
    dphi = (dphimat(iV,:,:));

    % calculate the rotation matrix for atoms of speed V(iV)
         that acts on the spin at each point for every step
        along the solenoid

    R(1,1,:,:)=(Bdir(1,:,:).^2).*(1-cos(dphi))+cos(dphi);
    R(1,2,:,:)=(Bdir(1,:,:).*Bdir(2,:,:)).*(1-cos(dphi))-
        Bdir(3,:,:).*sin(dphi);
    R(1,3,:,:)=(Bdir(1,:,:).*Bdir(3,:,:)).*(1-cos(dphi))+
        Bdir(2,:,:).*sin(dphi);


    R(2,1,:,:)=(Bdir(1,:,:).*Bdir(2,:,:)).*(1-cos(dphi))+
        Bdir(3,:,:).*sin(dphi);
    R(2,2,:,:)=(Bdir(2,:,:).^2).*(1-cos(dphi))+cos(dphi);
    R(2,3,:,:)=(Bdir(2,:,:).*Bdir(3,:,:)).*(1-cos(dphi))-
        Bdir(1,:,:).*sin(dphi);


    R(3,1,:,:)=(Bdir(1,:,:).*Bdir(3,:,:)).*(1-cos(dphi))-
        Bdir(2,:,:).*sin(dphi);
    R(3,2,:,:)=(Bdir(2,:,:).*Bdir(3,:,:)).*(1-cos(dphi))+
        Bdir(1,:,:).*sin(dphi);
    R(3,3,:,:)=(Bdir(3,:,:).^2).*(1-cos(dphi))+cos(dphi);

    % initial spin for velocity iV at Z-point Z(iZ=1)=0
    spinvec1(:,:,1,iV)=Sidir(:,:,iV);

    % for points iP at each further step Z(iZ) along the
        length, apply the corresponding rotation operator
    for iP = 1:nP
        for iZ=2:nZ
            spinvec1(:,iP,iZ,iV)=R(:,:,iP,iZ)*spinvec1(:,iP,
                iZ-1,iV);
        end
    end
end
```

```

    % output spin, making sure dimensions are correct (so no
        squeezing of 1-dim axes)
    Sf = Simag .* reshape(spinvec1, [3 nP nZ nV]);
end
```

MAIN.M

The first step is to initialise the variable parameters of the incident atoms (3-He atoms are assumed so the gyromagnetic ratio is not initialised here)

```
%-----------------------------------------------------------------
% initial atom parameters
%-----------------------------------------------------------------

% average speed of He-3 atom through solenoid (along x-axis) in
    m/s
V0=768;

% maximum fractional difference of atom speeds from average
DV=0.1;

% number of velocities simulated at each point
nV=20;

% initialise vector of velocities
V = linspace(V0-(V0*DV),V0+(V0*DV), nV);

% initial spin polarisation (parallel to the y-axis)
spin=[0 1 0]'; Smag = sum(spin.^2);

% scattering angle at surface
sctAngle = pi/4;

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

We then turn to initialising the parameters of the apparatus, in particular the solenoids. We additionally define a meshgrid of lateral points inside the solenoid, with the understanding that incident atom trajectories will not colinear, rather they will be spread approximately uniformly over a circular area within the solenoid. In principle, this code allows for a meshgrid of any shape so long as the magnetic field for the solenoid is defined at each point on it.

```
%-----------------------------------------------------------------
% solenoid parameters
%-----------------------------------------------------------------

% length of solenoid /m
L = 0.005;

% width/height of rectangular solenoid /m
Lx = 0.05;
Ly = 0.05;

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

```matlab



%-----------------------------------------------------------------
% solenoid dimensions of interest
%-----------------------------------------------------------------

% SETUP:

% length of the region of interest (symmetrical about solenoid)
Zmax = 5/4*L;

% number of Z-points to simulate
nZ = 750;


% maximum allowed beam radius
a = min(Lx, Ly)/2;

% number of radii to simulate
nr = 5;


% number of angles to simulate -- in range [0,2*pi)
ntheta = 8;


% CALCULATION:

% initialise vector of Z-points
Z = linspace(0,Zmax,nZ)';

% calculate spacing between Z-points (we wish to keep this
    consistent throughout the simulation
dZ = Z(2)-Z(1);


% set of radii around central beam in range [0,a)
r = linspace(0,a,nr+1); r(end) = [];


% set of angles around central beam in range [0,2*pi)
theta = linspace(0,2*pi,ntheta+1); theta(end) = [];



% meshgrid of all lateral points on the beamline/cylinder of
    atoms (the meshgrid here isn't actually important, just each
     meshgrid index corresponds to the field value in B at the
    same index (see below)
[R, Theta] = meshgrid(r, theta);

nP = numel(R); % number of points in meshgrid



% extract Cartesian grid for use with Solenoid class
X = reshape(R.*cos(Theta), [nP 1]); Y = reshape(R.*sin(Theta), [
    nP 1]);

%-----------------------------------------------------------------
```

```
%-----------------------------------------------------------------
```

Having defined the solenoid parameters, we now either import a field from another file or create one. Below we define both a simple uniform field along the z-axis as well as a field directed along the central z-axis with its magnitude given by Gaussian disitribution across the distance from the centre. This should demonstrate the required format of the field variable `B`. Note that in this code there are several calls to functions in the `Solenoid` class which are commented-out for use in the questions in this exercise.

```
%-----------------------------------------------------------------
% solenoid B-field
%-----------------------------------------------------------------

s = Solenoid(L, Lx, Ly, X', Y', Z'-Zmax/2);

smoothLen = (Z(end) - L)/2;

[Bx,By] = s.box_xy();

% FAST Z-TRANSITION:
%Bz = s.box_z();

% SLOW Z-TRANSITION:
% the smoothing length is chosen to maximise smoothing over the
    considered region Z
Bz = s.smoothstep_z((Z(end) - L)/2);

% APPLY RADIAL DECAY OF FIELD STRENGTH:
% the standard deviation is chosen to be a third of the radius
    of the solenoid so that the field strength is close to zero
    at the edges
%radial_decay_const = 3 * 1/(2*(min(Lx,Ly)/2)^2);
%for ij = 1:nP
%    for k = 1:nZ
%        Bz(ij,k) = Bz(ij,k) * exp(-(X(ij)^2)*radial_decay_const
     - (Y(ij)^2)*radial_decay_const);
%    end
%end

% intialise set of vectors for B-field at each point along the
    length
B=zeros([3 nP nZ]);
B(1,:,:) = Bx; B(2,:,:) = By; B(3,:,:) = Bz;

B = 0.15 .* B;

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

The final step of initialisation is to initialise the parameters of the spin rotator field, as well as the distances between components of the apparatus. In the given code there are three options for the spin rotator field.

```
%-----------------------------------------------------------------
% spin rotator field parameters
```

```
%-------------------------------------------------------------

% this variable is: - 0 if there is no spin rotator field
%                   - 1 if the spin rotation is ideal (i.e.
    uniform rotation for all speeds)
%                   - 2 if the spin rotator field is (fast-
    transitioning) uniform (i.e. rotation depends on speed)
spinRotator = 2;

% distance from end of each solenoid to the spin rotator field
rotBDist= 0.05;

% length of the spin rotator field
rotBLen = 0.05;

%-------------------------------------------------------------
%-------------------------------------------------------------
```

Having defined the necessary parameters, we may now turn to simulating the apparatus. We begin by simulating the precession in the first solenoid. We store the spins as a function of distance travelled through the apparatus in the matrix `Spins` and store those distances in the vector `Disps`. Note also that we choose to record a short period of propagation before reaching the solenoid to later produce graphics that are easier to understand.

```
%-------------------------------------------------------------
% first solenoid
%-------------------------------------------------------------

% propagate short distance up to solenoid
Spins = repmat(spin, [1 nP 5 nV]);
Disps = [-0.01:0.002:-0.002]';

% initial spins are the same for all axes and speeds
Si = repmat(spin, [1 nP nV]);

% precess spin through first solenoid
Sf1 = precess(Si, Z, V, B);

% update stored spins/distances
Spins = cat(3, Spins, Sf1);
Disps = cat(1, Disps, Z);

% continue propagating up to SR field
numStepsTorotB = rotBDist/dZ - 1;
Disps = cat(1, Disps, linspace(Disps(end)+dZ, Disps(end)+
    rotBDist, numStepsTorotB)');
Spins = cat(3, Spins, repmat(Spins(:,:,end,:), [1 1
    numStepsTorotB 1]));

%-------------------------------------------------------------
%-------------------------------------------------------------
```

Having propagated through the first solenoid, we then apply the spin rotator (SR) field. In the case of `spinRotator==1` we assume an 'ideal' SR field that rotates atoms of all speeds uniformly through the total scattering angle (to compensate for the change in direction at scatter).

```
%--------------------------------------------------------------
% spin rotator field
%--------------------------------------------------------------
% Rotates spin about the x-axis through sctAngle rad without affecting trajectory (roughly the same for all speeds

Nsr = rotBLen/dZ - 1;
Zsr  = linspace(Disps(end)+dZ, Disps(end)+rotBLen, Nsr)';

if spinRotator == 1
    % for an ideal uniform rotation for all speeds...

    % define small rotation for each step along axis
    RotMsr = [1 0 0; 0 cos(sctAngle/Nsr) -sin(sctAngle/Nsr); 0 sin(sctAngle/Nsr) cos(sctAngle/Nsr)];

    % initialise temporary spin variable
    Sfsr = zeros([3 nP Nsr nV]);

    % first rotation
    for iP = 1:nP
        Sfsr(:,iP,1,:) = RotMsr * squeeze(Spins(:,iP,end,:));
    end

    % iteratively rotate through small angle until sctAngle rad in total
    for iV=2:Nsr
        for iP = 1:nP
            Sfsr(:,iP,iV,:) = RotMsr * squeeze(Sfsr(:,iP,iV-1,:));
        end
    end

    % update stored spins
    Spins = cat(3, Spins, Sfsr);
```

If instead `spinRotator==2`, we apply a uniform field along the x-axis which causes precession in the y-z plane as desired. However, as shown in the theoretical background this will cause atoms of different speeds to precess by different total amounts. The average precession angle is given by:

$$\langle \phi \rangle = \frac{\gamma B \ell}{\langle v \rangle},$$

as confirmed in the theoretical background. Thus, it is possible to choose a field strength that gives an average precession angle equal to the scattering angle. Whilst there will be a spread in the precession angle, for different atoms, this is rather small and becomes negligible when the intensity of a particular spin component is measured in a spin-echo experiment.

```
elseif spinRotator == 2
    % for a (fast-transitioning) uniform SR field with variance in rotation for diff speeds...

    % CREATE SR FIELD:
```

```
    % intialise set of vectors for B-field at each point along the length
    Bsr = zeros([3 nP length(Zsr)]);

    % for each point along the length...
    Bsr(1,:,:) = ones([1 nP, length(Zsr)]);

    % gyromagnetic ratio for helium 3
    gamma=2.0378e8; %rad/sec;

    % scale B-field for phi=sctAngle/Nsr -> <phi>=l*gamma*B/<v>
    Bsr = sctAngle*V0/(rotBLen*gamma) .* Bsr;

    % PRECESS SPIN THROUGH FIELD
    Sfsr = precess(Spins(:,:,end,:), Zsr, V, Bsr);

    % update stored spins
    Spins = cat(3, Spins, Sfsr);
```

Finally, if `spinPrecession` takes any other value then no SR field is applied at all:

```
else
    % for no SR field at all...

    % update stored spins
    Spins = cat(3, Spins, repmat(Spins(:,:,end,:), [1 1 Nsr 1]));
end

% update stored distances
Disps = cat(1, Disps, Zsr);

%---------------------------------------------------------------
%---------------------------------------------------------------
```

It is now necessary to resolve the spins of atoms into a new rotated basis since the direction of travel changes on scatter at the surface. This can be trivially achieved with a rotation matrix. (unfortunately, the the lack of support for higher-dimensional matrix products in Matlab demands the use of a cumbersome for-loop):

```
%---------------------------------------------------------------
% resolve Sf into new direction ready for second solenoid
%---------------------------------------------------------------

% rotation matrix between two bases
RotM = [1 0 0; 0 cos(sctAngle) sin(sctAngle); 0 -sin(sctAngle) cos(sctAngle)];

% rotate frame for final spins
Sfrot = zeros(size(Spins(:,:,end,:)));
for iP = 1:nP
    Sfrot(:,iP,:) = RotM * squeeze(Spins(:,iP,end,:));
end

% continue propagating up to next solenoid
Disps = cat(1, Disps, linspace(Disps(end)+dZ, Disps(end)+rotBDist, numStepsTorotB)');
Spins = cat(3, Spins, repmat(Spins(:,:,end,:), [1 1 numStepsTorotB 1]));
```

```

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

Having rotated into the new basis, we can simply apply the precession code for the second solenoid. Note that the magnetic field is in the opposite direction!

```
%-----------------------------------------------------------------
% second solenoid
%-----------------------------------------------------------------

% precess current spins through second solenoid (opposite field
    direction)
Sf2rot = precess(Sfrot, Z, V, -B);

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

For measurement, since the polarising filters are aligned with respect to the initial direction of travel, we must rotate the pecessed spins back to the original basis with an inveres rotation matrix:

```
%-----------------------------------------------------------------
% resolve Sf2 back
%-----------------------------------------------------------------

% rotate frame back to original basis
Sf2 = zeros(size(Sf2rot));
for iP=1:nP
    for iV=1:nV
        Sf2(:,iP,:,iV) = RotM \ squeeze(Sf2rot(:,iP,:,iV));
    end
end

% update stored spins/distances
Spins = cat(3, Spins, Sf2);
Disps = cat(1,Disps,Disps(end)+dZ+Z);

% propagate short distance after solenoid
Spins = cat(3, Spins, repmat(Spins(:,:,end,:), [1 1 5 1]));
Disps = cat(1, Disps, [Disps(end)+0.002:0.002:Disps(end)+0.01]')
    ;

%-----------------------------------------------------------------
%-----------------------------------------------------------------
```

It may then be desirable to plot curves of spin components against distance through the apparatus for different speeds/trajectories. An example is given below for plots of curves for different speeds, all along the central axis:

```
%-----------------------------------------------------------------
% plot graphs of spin for diff speeds on central axis
%-----------------------------------------------------------------

figure();

% lateral point we wish to plot components for (central axis)
iP = 1;
```

```

% plot x-component of spin against distance through solenoid
subplot(3,1,1);
plot(Disps, squeeze(Spins(1,iP,:,:)));
title('$x$ component'); xlim([Disps(1) Disps(end)]); ylim([-Smag
     Smag]);

% plot y-component of spin against distance through solenoid
subplot(3,1,2);
plot(Disps, squeeze(Spins(2,iP,:,:)));
title('$y$ component'); xlim([Disps(1) Disps(end)]); ylim([-Smag
     Smag]);

% plot z-component of spin against distance through solenoid
subplot(3,1,3);
plot(Disps, squeeze(Spins(3,iP,:,:)));
title('$z$ component'); xlim([Disps(1) Disps(end)]); ylim([-Smag
     Smag]);

%--------------------------------------------------------------
%--------------------------------------------------------------
```