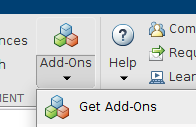
# Excitations Data Analysis Training Course – Solutions 6

## SpinW Introduction

[SpinW](https://spinw.org/) is a Matlab program to calculate spin wave dispersion and inelastic neutron scattering spectra using linear spin wave theory.

You can install SpinW using the “Add-Ons” option:

Then search for SpinW and “Add” it.

On the IDAaaS system, you should use the

>> horace\_on

command instead.

SpinW is implemented as a Matlab class. To run a calculation you first have to create an object (instance) of this class:

>> object = spinw();

then set properties, such as the lattice parameters and atom positions, by calling methods of the class, e.g.:

>> object.genlattice('lat\_const',[4,4,4],'angled',[90,90,90],'spgr','P 4');

## SpinW I – Excercise 1: 1D chain

As an alternative to setting up a spin wave model from scratch, SpinW also has a function to set up some basic models called sw\_model. Type:

>> help sw\_model

to see how to use this. Generally all the SpinW commands have help text with the syntax of the command at the start of the help text. (As an alternative to “help <command>”, you can also use “doc <command>” which will display the help text in a separate window.)

1. Using sw\_model set up a model of an antiferromagnetic spin chain with nearest neighbour interaction *J*=1 meV. (Remember that positive values indicate antiferromagnetic exchange in the convention used by SpinW)
2. Plot the model – which crystallographic direction is the spin chain along? In which direction in Q would you expect the spin waves to be dispersive?

Hint: You can use “plot(object)” to plot a 3D model of the SpinW object.

Answer: The chain is along the *a* direction ([100]) so the spin waves should disperse along the *h*-direction (H00).

1. Use the spinwave method to calculate the spin wave dispersion *along the chain*, and plot the result.

Hint: Type “help object.spinwave” or “help spinw/spinwave” to get the syntax for the spinwave method. You can use sw\_plotspec to plot the spectra object returned by spinwave. [N.B. a “method” is a function belonging to a class]

1. Use the sw\_egrid function on the calculated spectrum object to create an intensity spectrum and plot it with sw\_plotspec with the 'dE' option.
2. Now create a *ferromagnetic* chain using sw\_model and calculate and plot its neutron intensity spectrum. (Use the figure command to create a new figure before plotting the new spectrum otherwise the old [antiferromagnetic chain] spectrum will be erased and overplotted).
3. Aside from the difference in the dispersion, what is the difference between the neutron intensity between the ferromagnetic and antiferromagnetic chain? Why do you think this difference arises?

Hint: Remember that neutron scattering function is proportional to the spin-spin correlation function.

Answer: The intensity of the FM chain is constant throughout the BZ, whereas the intensity of the AFM chain is peaked at (0.5,0,0) – the AFM order wavevector. This is because for the FM chain, the spin-spin correlation is the same along the chain. In the AFM chain, at ***q***=(000), effectively the nearest neighbour spins are coupled and they are antiferromagnetically aligned and so are not well correlated. At ***q***=(½00), it’s the next-nearest neighbours which are coupled and they are aligned so are well correlated.

1. Use the powspec method to calculate a powder spectrum of the FM chain between 0 and 3Å-1 in 100 steps with 1000 random sampling points per Q bin. What feature of the dispersion correspond to what feature of the powder spectrum?

Answer: The cosine dispersion near the Bragg peak is still clear, and the spectrum cuts off abruptly at the top of the band.

1. Repeat the powder spectrum calculation for the AFM chain – what differences do you notice?

Answer: The dispersion is now coming out of the AFM Bragg peak around 1Å-1 instead and shows a V-shaped (|sin|) dispersion.

1. Finally, you can also calculate the magnon density of states (DOS) by randomly sampling in the first Brillouin zone and summing the results over Q:

Qvec = rand(3, 1e5); % Generate 10000 random Q points between [0,1]

dosspec = swobj.spinwave(Qvec); % Calculates the dispersion

% Now take the trace of the Sab correlation matrix instead Sperp

dosspec = sw\_egrid(dosspec, 'component', 'Sxx+Syy+Szz');

% Now broaden the spectrum with a Gaussian of width 0.1meV

dosspec = sw\_instrument(dosspec, 'dE', 0.1);

magnon\_dos = sum(dosspec.swConv, 2);

% The energy vector is stored as bin boundaries in the spectrum

en = (dosspec.Evect(1:end-1) + dosspec.Evect(2:end))/2

figure; plot(en, magnon\_dos);

1. How does the DOS compare to summing the powder spectrum over Q? Why did we use the 'Sxx+Syy+Szz' component instead of the neutron 'Sperp' component which is the default used in powspec?

Answer: The 'Sperp' component takes the component of the spin-spin correlation tensor perpendicular to ***Q*** which even though it is powder averaged it is still biased whereas taking the trace of the spin-spin correlation tensor with 'Sxx+Syy+Szz' avoids this bias. This can be seen in the slight differences between the powder average and the DOS calculation – but in practice is it minimal so you can also sum a powder spectrum to get the magnon DOS.

Solution script:

% Set up an AFM chain

chain = sw\_model('chain', 1);

% Plot it

plot(chain)

% Calculate the dispersion along the chain

spec = chain.spinwave({[0 0 0] [1 0 0]})

figure

sw\_plotspec(spec)

% Plot the intensity spectrum

spec = sw\_egrid(spec)

sw\_plotspec(spec, 'dE', 0.1)

%%

% Create a FM chain

fmchain = sw\_model('chain', -1);

spec = fmchain.spinwave({[0 0 0] [1 0 0]});

spec = sw\_egrid(spec);

figure

sw\_plotspec(spec, 'dE', 0.1)

% Create a powder spectrum

powspecfm = fmchain.powspec(linspace(0,3,100), 'Evect', linspace(0, 5, 100), 'nRand', 1000)

figure;

sw\_plotspec(powspecfm)

%%

% AFM chain powder spectrum

powspec\_afm = chain.powspec(linspace(0,3,100), 'Evect', linspace(0, 3, 100), 'nRand', 1000)

figure;

sw\_plotspec(powspec\_afm)

%%

% Calculates the DOS for the AFM chain

Qvec = rand(3, 1e5); % Generate 10000 random Q points between [0,1]

dosspec = chain.spinwave(Qvec); % Calculates the dispersion

% Now take the trace of the Sab correlation matrix instead Sperp

% also make sure we use the same energy vector as for the powder calculation to compare

dosspec = sw\_egrid(dosspec, 'component', 'Sxx+Syy+Szz', 'Evect', linspace(0,3,100));

% Now broaden the spectrum with a Gaussian of width 0.01meV

dosspec = sw\_instrument(dosspec, 'dE', 0.01);

magnon\_dos = sum(dosspec.swConv, 2);

% The energy vector is stored as bin boundaries in the spectrum

en = (dosspec.Evect(1:end-1) + dosspec.Evect(2:end))/2

figure; plot(en, magnon\_dos);

% Compare to summing the powder spectrum

hold all;

enp = (powspec\_afm.Evect(1:end-1) + powspec\_afm.Evect(2:end))/2;

powsum = sum(powspec\_afm.swConv, 2);

plot(enp, powsum./max(powsum).\*max(magnon\_dos))

%%

% Calculates the DOS for the FM chain

Qvec = rand(3, 1e5); % Generate 10000 random Q points between [0,1]

dosspec = fmchain.spinwave(Qvec); % Calculates the dispersion

% Now take the trace of the Sab correlation matrix instead Sperp

% also make sure we use the same energy vector as for the powder calculation to compare

dosspec = sw\_egrid(dosspec, 'component', 'Sxx+Syy+Szz', 'Evect', linspace(0,5,100));

% Now broaden the spectrum with a Gaussian of width 0.01meV

dosspec = sw\_instrument(dosspec, 'dE', 0.01);

magnon\_dos = sum(dosspec.swConv, 2);

% The energy vector is stored as bin boundaries in the spectrum

en = (dosspec.Evect(1:end-1) + dosspec.Evect(2:end))/2

figure; plot(en, magnon\_dos);

% Compare to summing the powder spectrum

hold all;

enp = (powspecfm.Evect(1:end-1) + powspecfm.Evect(2:end))/2;

powsum = sum(powspecfm.swConv, 2);

plot(enp, powsum./max(powsum).\*max(magnon\_dos))

## SpinW I – Excercise 2: Triangular Lattice Antiferromagnet

1. Create an empty spinw object with:

>> swobj = spinw();

1. Use the genlattice method of this object to create a hexagonal lattice with parameters *a*=*b*=3 Å and c=5 Å (α=β=90˚, γ=120˚).
2. Use the addatom method to add a Ni2+ ion with S=1 at **r**=(0,0,0).
3. Use the gencoupling method to generate the bonds up to 5 Å distance.
4. Use the addmatrix method to add an exchange tensor (3x3 matrix) representing an antiferromagnetic Heisenberg exchange interaction of magnitude 1 meV, then use the addcoupling method to define the first (nearest neighbour bond) using this matrix (tensor).
5. Plot your model and see that it represents a triangular lattice antiferromagnet (you should be able to click on the atoms and bonds to get information on them).
6. Use the genmagstr method with the 'mode', 'helical' option, **k**=[⅓,⅓,0], **S**=[1;0;0] and **n**=[0,0,1] to create a 120˚ magnetic structure in the incommensurate representation.
7. Calculate and plot the inelastic neutron spin wave spectrum.
8. Add an easy-axis anisotropy in the *a-b* plane (perpendicular to [0,0,1]) with magnitude 0.1 meV using the addmatrix and addaniso methods and recalculate and re-plot the INS spectrum. What is the difference between this and the previous spectrum you plotted? From this can you infer what fluctuations are associated with the different magnon modes in the spectrum? (How many modes are there in the spectrum?)

Answer: There are three modes in the spectrum in both cases. Without the SIA, the modes all go to zero energy at the Bragg peak. With the SIA one of the mode is gapped – its lowest energy is about 1meV at the ordering wavevector of (⅓⅓0). Because we have an easy-plane anisotropy, fluctuations in the plane should not cost more energy due to the SIA. Fluctuations out of the *a-b* plane, however, does cost energy – we can thus infer that the gapped mode involves fluctuations out of the plane.

1. Make a copy of swobj (the spinw object you just created – spinw objects are “handle” objects so with swobj2 = swobj, swobj2 is just another handle to the original object – operations on swobj2 actually operate on the same object as swobj. In Horace most objects are “value” objects where the = (equals) operator creates a new copy). Using genmagstr on the copied object with the option 'nExt', 0.1 and the same parameters as before, create the same 120˚ magnetic structure but in the super-cell representation instead of the rotating frame (incommensurate) representation. Plot the model to check that it still looks like the 120˚ structure.
2. Calculate and plot the INS spectrum – what’s the difference with the previous (incommensurate) case? (How many modes are plotted now?) Why is this?

Answer: There are now a lot more modes (6 visible modes, but actually 9 in total because there are 6 doubly degenerate modes). This is because for magnons there are always at least as many modes at positive energies as there are magnetic ions. Using the supercell method, we have to have a 3x3 supercell so have 9 atoms in the magnetic cell – yielding 9 modes. In the incommensurate calculations previously we only had one atom in the original cell. But we used a property of the rotating frame which gives additional modes at ***Q***+***k*** and ***Q***-***k*** as well as at ***Q*** where ***Q*** is the momentum transfer and ***k*** is the propagation vector – so have 3 modes. We can see though, that the neutron spectrum is the same in the two calculations, just that the super-cell calculation has more “ghost” (or silent) modes.

1. Add an easy-axis anisotropy along the *a* axis with magnitude +1 meV. Recalculate the INS spectrum – you should get an error message. Why does this occur?

Hint: Think about what the axial anistropy term you just added does to the magnetic ground state vs what the current magnetic ground state (spin structure) is set to.

Answer: The error message says that the “magnetic structure is wrong” – what this means is that the magnetic structure does not agree with the exchange and SIA parameters so that the Hamiltonian is not Hermitian and has imaginary eigenvalues (magnon energies). This is because we have 120° magnetic structure but a very strong axial anisotropy which makes the system more Ising like. The 120° structure thus doesn’t agree with the SIA and makes the Hamiltonian non-Hermitian.

1. Use the 'hermit', false option to spinwave to force the calculation and plot the result – what’s the maximum energy of imaginary modes? If the maximum energy of the imaginary mode is small (~0.01-0.02 [a few percent] of the maximum energy of all the modes) then you can ignore the warning and the calculate spectrum is probably accurate enough. But if it is much larger then you should not trust the calculation.

Answer: 1.5meV – you can’t use this calculation!

1. Use the optmagsteep method to refine the magnetic structure, and recalculate and replot the INS spectrum. Are the imaginary modes still there? What are the main differences you see now, compared to previously? Can you explain what the modes are now?

Answer: optmagsteep causes the spins where were not along [100] to cant towards that direction. The INS spectrum is now composed of three clear modes well separated from each other. One mode is still a Goldstone mode, with zero energy at the ordering wavevector – it involves fluctuations of the spins which are aligned along [100]. The other two modes involve fluctuations in plane but away from [100] and fluctuations out of plane.

Solution script:

% Set up a triangular lattice AFM

tri = spinw();

tri.genlattice('lat\_const', [3,3,5], 'angled', [90 90 120]);

tri.addatom('r', [0,0,0], 'S', 1, 'label', 'MNi2');

tri.gencoupling('maxDistance', 5)

tri.addmatrix('label', 'J', 'value', 1);

tri.addcoupling('mat', 'J', 'bond', 1);

plot(tri)

% Generates a 120 deg structure

tri.genmagstr('mode', 'helical', 'k', [1/3,1/3,0], 'S', [1;0;0], 'n', [0,0,1]);

plot(tri)

% Plots the spin wave spectrum

spec = tri.spinwave({[0 0 0] [1 1 1] 200});

spec = sw\_egrid(spec)

figure

sw\_plotspec(spec, 'dE', 0.1)

%%

% Add easy-plane SIA and recalculate spectrum

tri.addmatrix('label', 'K', 'value', diag([0 0 0.1]));

tri.addaniso('K');

spec = tri.spinwave({[0 0 0] [1 1 1] 200});

spec = sw\_egrid(spec)

figure

sw\_plotspec(spec, 'dE', 0.1)

%%

% Make a copy of the spinw object and create a super-cell 120 deg structure

tri2 = copy(tri);

tri2.genmagstr('mode', 'helical', 'k', [1/3,1/3,0], 'S', [1;0;0], 'n', [0,0,1], 'nExt', 0.1);

plot(tri2)

% Recalculate the spin wave dispersion

spec = tri2.spinwave({[0 0 0] [1 1 1] 200});

spec = sw\_egrid(spec)

figure

sw\_plotspec(spec, 'dE', 0.1)

%%

% Add an easy-axis SIA

tri2.genmagstr('mode', 'helical', 'k', [1/3,1/3,0], 'S', [1;0;0], 'n', [0,0,1], 'nExt', 0.1);

tri2.addmatrix('label', 'K2', 'value', diag([-1 0 0]));

tri2.addaniso('K2');

% Recalculate the spectrum

%spec = tri2.spinwave({[0 0 0] [1 1 1] 200});

% Error!

% Force it to ignore the non-Hermiticity

spec = tri2.spinwave({[0 0 0] [1 1 1] 200}, 'hermit', false);

figure

sw\_plotspec(spec)

% Run optmagsteep to refine the structure and recalculate

tri2.optmagsteep

plot(tri2)

spec = tri2.spinwave({[0 0 0] [1 1 1] 200}, 'hermit', false);

spec = sw\_egrid(spec);

figure

sw\_plotspec(spec, 'dE', 0.1)

## SpinW I – Exercise 3: BiFeO3

In this exercise we’re going to look at BiFeO3 a distorted G-type (cubic) antiferromagnet, which has a long period (small ***k***) cycloid magnetic structure as the result of the Dzyaloshinskii-Moriya interaction. The spin Hamiltonian and INS data is described in this paper:

<https://web.ornl.gov/~okapon/manybodytheory/publication/pdf/PhysRevLett_109_067205.pdf>

1. Construct a spinw object from a CIF description of the structure of BiFeO3 using:

>> bfo = spinw('BiFeO3.cif');

The CIF can be found in here:

<https://github.com/pace-neutrons/edatc/matlab_scripts/BiFeO3.cif>

1. Plot the Fe-O octahedra using swplot.plotchem. Type bfo.unit\_cell first to check what the atom labels are. (They should be 'Fe Fe' and 'O O' for the iron and oxygen atoms). Can you see how this structure relates to the cubic perovskite structure?

Answer: The hexagonal *c* axis is the cubic [111] axis. You can rotate the plot so that you look down the rows of Bi atoms and then it would look more like the cubic structure.

1. Set the spin magnitude and label of the Fe3+ ions to S=5/2 and 'MFe3':

bfo.unit\_cell.S(2) = 2.5; % Check that Fe is the 2nd atom.

bfo.unit\_cell.label{2} = 'MFe3'; % So SpinW knows the formfactor

1. Use gencoupling, addmatrix and addcoupling to define the nearest (J1=4.5 meV) and next nearest (J2=0.2 meV) neighbour couplings. (Note that the PRL paper uses the measured moment at 200K for the spin magnitude S value whilst we have used the theoretical S=5/2, and so there is a factor of 1.44 in the calculation as noted on page 4 of the PRL. Plot the structure again and check it looks like fig 1 of the PRL.
2. Use addmatrix to define a DM interaction (read its help text) and the use addcoupling to add an addition nearest neighbour DM interaction (SpinW can accommodate up to three difference exchange tensors per bond; the limit of 3 is just a hard-coded limit to save memory and can be extended (reprogrammed) if there is demand). The DM vector is along [1, -1, 0] in the hexagonal unit cell we are using, which is equivalent to the cubic [1,1,-2] direction which the PRL paper uses. Use the same magnitude DM as in the paper (D=0.1623 meV).
3. Use addmatrix and addaniso to add an easy-axis anisotropy along the (hexagonal) *c* axis of magnitude K=0.0068 meV (remember the SpinW convention of negative values for easy-axis and positive values for easy-plane anisotropy).
4. Use optmagk to optimise the incommensurate propagation vector *k*, both using the 'kbase' option and without it (the propogation vector is expected to be along [1;1;0]). Run the calculations a few times and compare the ground state energy E of the different approaches as well as the inferred *k* vector. What do you notice?

Hint: optmagk using a global minimisation algorithm, the particle-swarm optimizer to determine the *k*-vector which yields the smallest eigenvalue of the Fourier transform of the Hamiltonian, in a space which can have many small minima.

Answer: The algorithm sometimes gets stuck in local minima where the propagation vector is not along (δ δ 0) – but the ground state energy of these configurations are always higher than that with the propagation vector along (δ δ 0). Using the 'kbase' option forces the algorithm to always choose vectors along (δ δ 0) and hence avoid the local minima.

1. What value do you find for the propagation vector (δ δ 0)? Does this match the experimental value δ=0.0045 found in the PRL paper? Note that the propogation vector determined by SpinW here assumes a “harmonic” cycloid – that is that the ordered structure can be described by a single phase term exp(i*k*.*r*). In BiFeO3 when the SIA is included, the cycloid structure is no longer harmonic, but instead acquires additional modulations because the spins prefer to align along the easy-axis direction and so “bunches” up at those points. Given then, what value of D should you choose to get the experimental propagation vector?

Hint: Use figure 5 of the PRL paper.

Answer: You should get δ=0.0036. Using figure 5 of the paper, looking at the point where K=0 (e.g. for zero SIA and hence having a harmonic cycloid) the paper suggests that D=0.185 should be used. If you use this, then you get δ=0.0042, not quite right but this is probably from numerical effects – SpinW is not really the best program to calculate these quantities.

1. Calculate and plot the spin wave dispersion around the (111) Bragg peak (say from (0.98,0.98,1) to (1.02,1.02,1)) – what do you notice? What if you ran an optmagsteep optimisation? What changed and why?

Answer: You should see some imaginary modes – optmagk produces a cycloid but left the initial magnetic structure as moments parallel to the *c* axis. In addition to creating the cycloid, however, the DM interaction induces a very small canting which is not initially satisfied leaving a small non-hermicity. After running optmagsteep you should see this small canting and the imaginary modes go away. This canting is probably not physical but just a result of the calculation – it is like relaxing a crystal structure in a DFT phonon calculation; the relaxed structure will not be exactly like the experimental structure but you should still used the relaxed structure because it agrees better with the calculated forces (for DFT phonon calculations) or exchanges (for spin wave calculations). This is because your calculations may not include all the necessary interactions to reproduce the full experimental observations.

Solution script:

% SpinW BiFeO3 tutorial, Exercise 3, EDATC21

%

% The aim of this tutorial is to calculate the propagation vector

% of the cycloid of the multiferroic compound BiFeO3.

%

% An experimental description of the Hamiltonian may be found in

% the paper:

% Magnetic Dispersion and Anisotropy in Multiferroic BiFeO3,

% M. Matsuda et al., Phys. Rev. Lett. 109 067205 (2012)

% https://web.ornl.gov/~okapon/manybodytheory/publication/pdf/PhysRevLett\_109\_067205.pdf

%

% BiFeO3 is a famous room-temperature multiferroic. It is a slightly

% distorted perovskite structure. Its magnetic structure is a very

% long period cycloid which is stabilised by a Dzyaloshinskii-Moriya

% interaction. In the absence of the DM interaction, it would have

% a simple G-type AFM structure.

%

% We will apply the DM interaction and use SpinW to calculate the

% propagation vector k of the cycloid, using optmagk.

% First we load the structure from CIF and plot the Fe-O octahedra

bfo = spinw('BiFeO3.cif')

plot(bfo, 'range', [1 1 1])

% Prints out the unit cell to check what the labels are:

bfo.unit\_cell

swplot.plotchem('atom1', 'Fe Fe', 'atom2', 'O O', 'limit', 6);

%%

% SpinW is not able to infer which atom is magnetic so we manually specify

% here that the second atom is Fe3+ (changing its label to match the SpinW

% form factor library name)

bfo.unit\_cell.S(2) = 2.5;

bfo.unit\_cell.label{2} = 'MFe3'

% The PRL paper (and most spin wave models of BiFeO3) considers two

% exchange interactions, a DM interaction and an axial single-ion

% anisotropy. We will use the parameters in the PRL paper.

%

% Note that the spin length used by SpinW is g\*sqrt(S\*(S+1)) where S

% is defined in the addatom function. We have used the free-ion

% ("theoretical") value of S=5/2 for Fe3+ here, whereas Matsuda

% et al. have used the measured moment at 200K which is 4.1uB

% Compared to the theoretical spin length this is smaller by 1.44

% as the mention in page 4 of the paper. Thus the exchange parameters

% should be scaled down by this amount

J1 = 4.5; % == 6.48 / 1.44

J2 = 0.2; % == 0.29 / 1.44

D = 0.1623;

K = -0.0068; % negative to indicate axial SIA rather than planar

bfo.gencoupling('maxDistance', 20)

bfo.addmatrix('label', 'J1', 'value', J1, 'color', 'green')

bfo.addmatrix('label', 'J2', 'value', J2, 'color', 'white')

bfo.addcoupling('mat', 'J1', 'bond', 1)

bfo.addcoupling('mat', 'J2', 'bond', 2)

% Plots the unit cell - check it resembles fig 1 of paper

plot(bfo)

% Now we add the DM term. SpinW has a short cut where if you give it

% a 3-vector, it would interpret this as a DM vector and construct the

% correct 3x3 exchange tensor.

% In the case of BiFeO3, the DM vector is along [1 -1 0] and connects

% atoms along the spiral direction [1 1 0] (e.g. 2nd neighbours).

%

% Note that this is superficially different to what the paper has

% (it says the spiral is along [1 -1 0]) because the paper uses the

% pseudo-cubic (rhombohedral) unit cell, whereas in SpinW, 'R 3 c'

% defaults to a hexagonal cell. The principal axes in the two cells

% are: [1 1 0]hex = [1 -1 0]pc, and [0 0 1]hex = [1 1 1]pc.

bfo.addmatrix('label', 'D', 'value', [1 -1 0]\*D, 'color', [255 225 175])

bfo.addcoupling('mat', 'D', 'bond', 2)

% Print the exchange tensor for the DM interaction

id = find(~cellfun(@isempty, strfind(bfo.matrix.label, 'D')));

bfo.matrix.mat(:,:,id)

% Finally add the single-ion anisotropy

bfo.addmatrix('label', 'K', 'value', diag([0 0 K]), 'color', [185 135 0])

bfo.addaniso('K')

% Now optimise the propagation vector.

% We know that it should be qm = [delta delta 0] so we give optmagk

% a helping hand by setting this as a basis vector

res = bfo.optmagk('kbase', [1; 1; 0])

res.k

% What happens if we don't set the basis vector as [1; 1; 0]?

% E.g. if we used:

%res = bfo.optmagk() % without giving 'kbase' option

%res.k

% What about the energy of this ground state compared to the one with

% kbase=[1;1;0]? (Note, use:

%format long g

% To print more that 8 significant figures).

% What happens if we increase the value of D? (Or decrease J?)

%%

% You should find k = [0.0036 0.0036 0] or [0.9964 0.9964 0] because

% there is no energetic difference between k and 1-k.

% However, delta=0.0036 is quite different from delta=0.0045 that we

% were expecting.

%

% This is because in the harmonic approximation of treating the

% spin cycloid structure as S\_i = S0.exp(ik.r\_i), the propagation

% vector is independent of the single-ion anisotropy K

% (see "Origin of the long period magnetic ordering in BiFeO3",

% I. Sosnowska and AK. Zvezdin, J. Magn. Magn. Mat. 140-144

% (1995) 167-168, which also shows that for J2=0, the cycloid

% wavelength = 4J/D, or k = (D/J) / (2a) where a=5.58A

% is the lattice parameter, since cycloid wavelength = (a/2)/k.)

%

% The effect of the single-ion anisotropy is to make the cycloid

% anharmonic, so that S\_i = S0.exp(i phi\_i) where

% phi\_i = k.r\_i + psi\_i, and psi\_i is proportional to sin(2k.r\_i).

%

% This anharmonicity results in higher order satelite peaks in

% the neutron diffraction, and more importantly makes the magnon

% modes dipole active - that is, makes them electromagnons.

% However, including these effects in SpinW is not currently

% possible, except through the use of a large supercell

% (as in the Matsuda paper). The computational cost, however,

% for BiFeO3 is prohibitive due to the long wavelength of the

% cycloid (the calculations in the Matsuda paper used an optimised

% code on a cluster).

%

% Calculations with SpinW are thus restricted to the K=0 limit

% in figure 5, e.g.

J1 = 4.5; % == 6.48 / 1.44

J2 = 0.2; % == 0.29 / 1.44

D = 0.185;

K = 0;

% Substitute in new parameter values

bfo.matparser('mat', {'J1', 'J2', 'K(3,3)'}, 'param', [J1 J2 K]);

% For the DM vector we have to put in the matrix explicitly

dmmat = [0 0 1; 0 0 1; -1 -1 0];

bfo.matparser('mat', {'D'}, 'selector', dmmat, 'param', [D]);

% Optimise the propagation vector again

res = bfo.optmagk('kbase', [1; 1; 0])

res.k

plot(bfo)

%%

% Calculate the spin wave dispersion around the propagation vector

% with the ('hermit', false) option and see where the imaginary

% eigenvalues are (you will have to zoom in to below 2meV).

spec = bfo.spinwave({[0.98 0.98 1] [1.02 1.02 1] 200}, 'hermit', false);

sw\_plotspec(spec)

ylim([0 3])

% Why are there still imaginary energies?

% Print the magnetic moment directions after the optmagk run:

bfo.magstr.S

% Now run bfo.optmagsteep print the moments again.

% What has changed? Why?

bfo.optmagsteep()

bfo.magstr.S

% Now plot the spin wave dispersion again. Verify the imaginary

% energies are gone.

spec = bfo.spinwave({[0.98 0.98 1] [1.02 1.02 1] 200}, 'hermit', false);

figure

sw\_plotspec(spec)

ylim([0 3])