# Excitations Data Analysis Training Course – Solutions 10

## Fitting the crystal field parameters of Yb2Ti2O7

Yb2Ti2O7 is famous as a “quantum spin ice” material – the “quantum” part comes from the fact that at low temperatures it behaves as an effective S=1/2 system with strong anisotropy which is needed to stabilise a quantum spin liquid. This is because although the magnetic Yb3+ ions actually have a full moment of J=7/2, the crystal field splits this into 4 doublets and the splitting is such that the ground state doublet has a large energy gap to the first excited state. We will be looking at fitting the crystal field parameters of Yb2Ti2O7 in today’s practical.

The data file is available at:

/mnt/ceph/auxiliary/excitations/edatc/powder\_data/yb2ti2o7\_Ei200\_100K.nxspe

1. Load the data using MSlice and make two cuts along energy, the first integrating from Q=0 to Q=5 Å-1 and the second from Q=12 to Q=17 Å-1. Then make another cut along |Q| integrating from 10 to 50 meV in energy. From this last cut determine a scaling factor to scale the high-Q energy cut to low-Q to be used as a background.

Hint: You can either just inspect the scale factor by eye or fit a Q^2 dependence to the Q-cut and evaluate the fitted function at 2.5 Å-1 and 14.5 Å-1 and take the ratio.

1. Export the three cuts to the Mantid Workbench using the “MD Histo” tab. In the workbench script editor, type the following code to make the background subtraction:

cf\_cut = mtd[<name\_of\_lowQ\_cut>] - scaling\_factor\*mtd[<name\_of\_highQ\_cut>]

where <name> is the name of the workspaces you exported (you can drag and drop the workspace name from the Workbench workspaces list to the script window), and scaling\_factor is the factor you determined in step 1.

1. Create a resolution model for the measurement with the following code:

from CrystalField import ResolutionModel

from PyChop import PyChop2

mari = PyChop2('MARI', 'S', 600)

mari.setEi(200)

resmod = ResolutionModel(mari.getResolution, xstart=-200, xend=199)

1. Set up a CrystalField model with ion Yb, in 'D3d' symmetry at 100K with the ResolutionModel defined above and random parameters here:

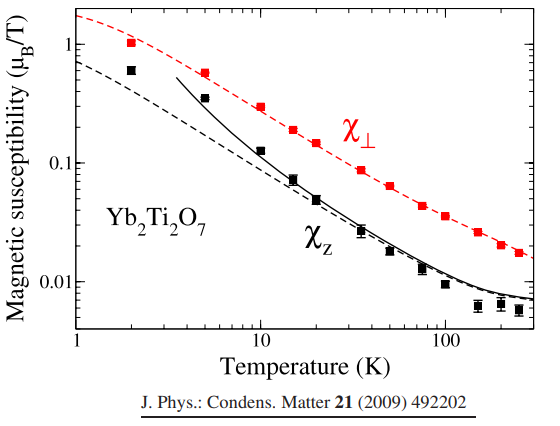
nonzero\_parameters = ['B20', 'B40', 'B60', 'B43', 'B63', 'B66']

Blm = {}

for pname in nonzero\_parameters:

Blm[pname] = np.random.rand()\*2-1

1. Set up a CrystalFieldFit object with the Model specified in step 4 and InputWorkspace as the cf\_cut subtracted cut created in step 2. Run the estimate\_parameters function of the fit object with EnergySplitting set to 100, and Parameters set to nonzero\_parameters. You can also set NSamples to 1000 (or so) and set Seed=int(np.random.rand()\*1e5). Select the best sample (using .select\_estimated\_parameters(1) – note that I was wrong in the lecture, index 1 is the best sample) and then run the fit against the data. Plot the resulting fit vs the data – is it any good? (You’ll probably have to run this step several times to get a good-ish set of initial parameters for the fit).
2. Calculate the susceptibility along the x- and z-direction and compare to this graph:

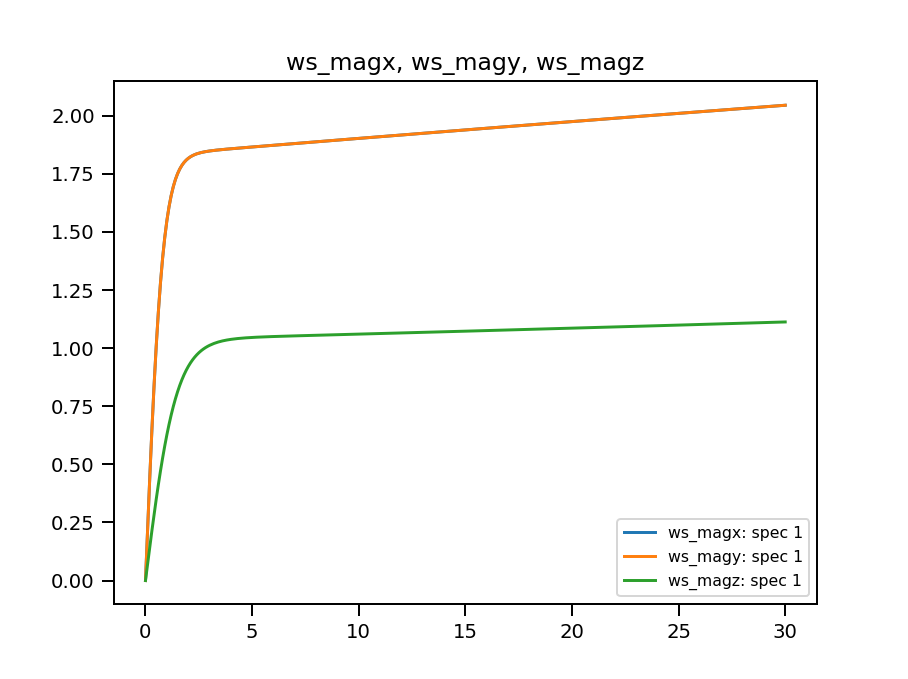
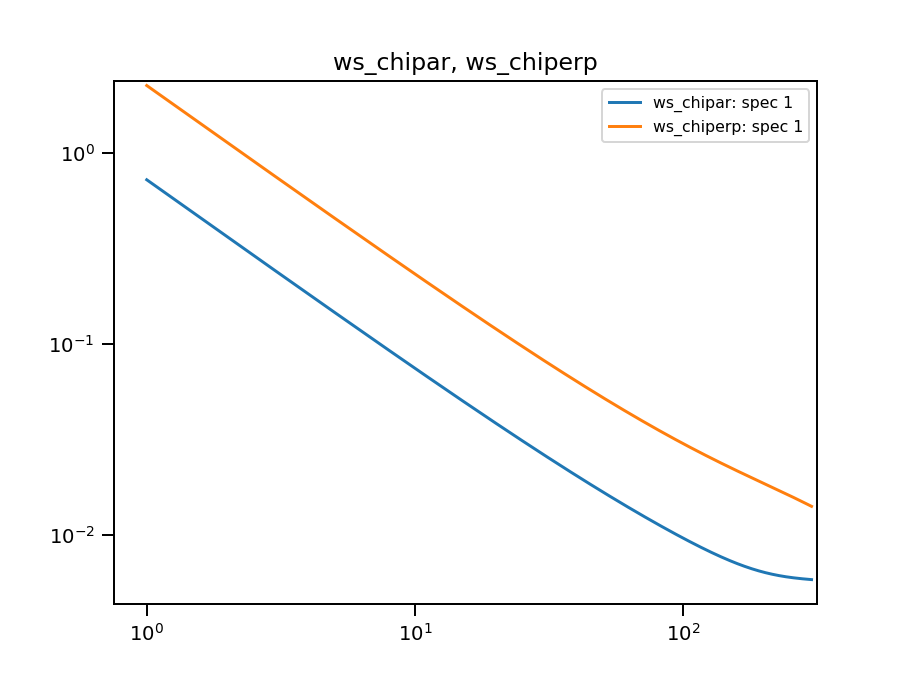


Where the x-direction is the perpendicular direction – does your graph match the data? (Is the x-axis susceptibility higher than the z-axis susceptibility? If not, try step 5 again.

1. Print out the eigenvalues of the Hamiltonian and check the energy from the ground state to the first excited state is large. Print the eigenvectors – what is the ground state wavefunction in terms of the |mJ> states? Do you think the system will have easy-axis or easy-plane anisotropy?

Hint: You can plot the magnetisation to work it out.

Answer: Easy-plane perpendicular to z. If you plot the magnetisation in the x, y and z direction, you should see that the x- and y- magnetisation are equal and higher than the z-magnetisation indicating that it is easier to magnetise along these directions (also even in the above graph you should see that the perpendicular susceptibility is higher indicating it is the easier direction). Also, you should get that the ground state wavefunction is mostly |mJ=±1/2> which indicates that the Jz component prefers to be minimised in the ground state indicating the spins prefer to align perpendicular to z. (In contrast if the wavefunction is mostly |mJ=±7/2> then the it prefers to align along z).



Solution script:

from mantid.simpleapi import \*

import matplotlib.pyplot as plt

import numpy as np

from CrystalField import CrystalField, CrystalFieldFit, ResolutionModel

from PyChop import PyChop2

# Make cuts in MSlice and save them to workbench first

cf\_cut = mtd['yb2ti2o7\_Ei200\_100K\_cut(0.000,5.000)'] – \

0.25\*mtd['yb2ti2o7\_Ei200\_100K\_cut(12.000,17.000)']

# Set up a resolution model for MARI

mari = PyChop2('MARI', 'S', 300)

mari.setEi(200)

resmod = ResolutionModel(mari.getResolution, xstart=-200, xend=199)

# Generates a random set of parameters

nonzero\_parameters = ['B20', 'B40', 'B60', 'B43', 'B63', 'B66']

Blm = {}

for pname in nonzero\_parameters:

Blm[pname] = (np.random.rand()\*2-1)/10

cf = CrystalField('Yb', 'D3d', Temperature=100, ResolutionModel=resmod, \*\*Blm)

if 'estimated\_parameters' in mtd:

DeleteWorkspace('estimated\_parameters')

# Runs the estimate\_parameters algorithm to find a decent set of initial parameters

cffit = CrystalFieldFit(cf, InputWorkspace='cf\_cut', Exclude=[-200,20], MaxIterations=0)

cffit.estimate\_parameters(EnergySplitting=120, Parameters=nonzero\_parameters, \

NSamples=1000, Seed=int(np.random.rand()\*1e5))

print('Returned', cffit.get\_number\_estimates(), 'sets of parameters.')

cffit.select\_estimated\_parameters(1)

print('Best guess parameters:')

for pname in nonzero\_parameters:

print('%s = %5.3g' % (pname, cf[pname]))

# Reruns the fit on these parameters

cffit.fit()

Blmfit = {pname:cf[pname] for pname in nonzero\_parameters}

# Calculates the resulting crystal field

cf = CrystalField('Yb', 'D3d', Temperature=100, ResolutionModel=resmod, \*\*Blmfit)

# Actual parameters (from PRB 97 224409 (2018))

# Uncomment to calculate physical properties for these.

#cf = CrystalField('Yb', 'D3d', Temperature=100, ResolutionModel=resmod,

# B20=1.1, B40=-0.0591, B43=0.3258, B60=0.00109, B63=0.0407, B66=0.00727)

# Calculates physical properties

ws\_fit = CreateWorkspace(\*cf.getSpectrum())

ws\_invchi = CreateWorkspace(\*cf.getSusceptibility(Temperature=np.linspace(1,30,300), \

Unit='cgs', Inverse=True, Hdir='powder'))

ws\_chipar = CreateWorkspace(\*cf.getSusceptibility(Temperature=np.linspace(1,300,300), \

Unit='bohr', Inverse=False, Hdir=[0,0,1]))

ws\_chiperp = CreateWorkspace(\*cf.getSusceptibility(Temperature=np.linspace(1,300,300), \

Unit='bohr', Inverse=False, Hdir=[1,0,0]))

ws\_magx = CreateWorkspace(\*cf.getMagneticMoment(Hmag=np.linspace(0,30,300), \

Unit='bohr', Inverse=False, Hdir=[1,0,0]))

ws\_magy = CreateWorkspace(\*cf.getMagneticMoment(Hmag=np.linspace(0,30,300), \

Unit='bohr', Inverse=False, Hdir=[0,1,0]))

ws\_magz = CreateWorkspace(\*cf.getMagneticMoment(Hmag=np.linspace(0,30,300), \

Unit='bohr', Inverse=False, Hdir=[0,0,1]))

np.set\_printoptions(precision=3, linewidth=150, suppress=True)

print(cf.getEigenvalues())

print(cf.getEigenvectors())