PyCrystalField tutorial

Allen Scheie
(*Oak Ridge National Laboratory)*

Modeling and Fitting Crystal Fields With Neutron Scattering
Virtual Workshop
May 4, 2022

Outline

- Part 1: generate a point charge model from a .cif file.
- Part 2: fit a CEF Hamiltonian to neutron scattering data.
- Part 3: Calculate bulk magnetization and susceptibility.

Installation notes:

- PyCrystalField is designed to be used with Jupyter notebooks
  - Easiest way to install: Anaconda https://www.anaconda.com/
  - Anaconda also includes all required dependencies.

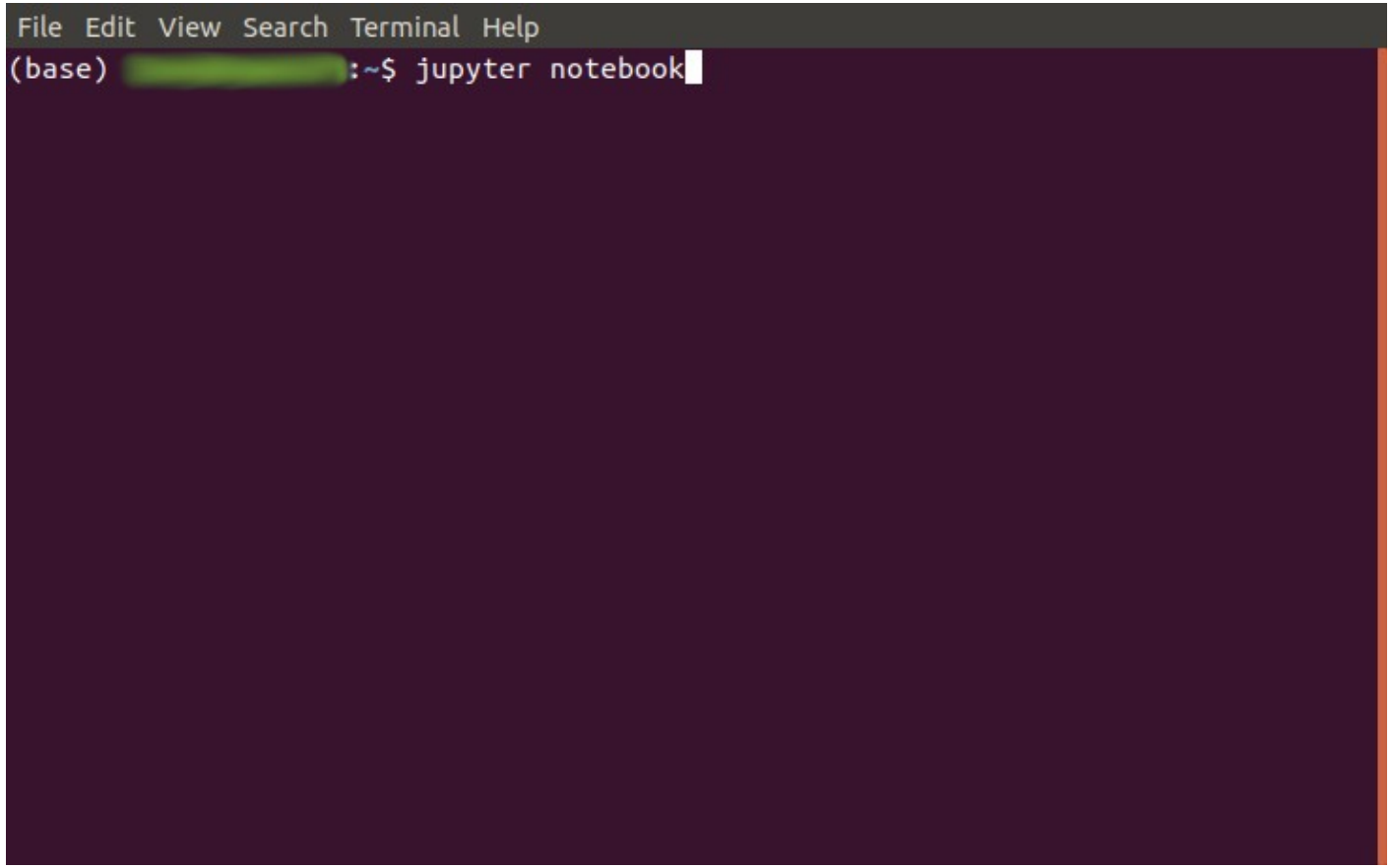
- Next, install using pip:

To download and use PyCrystalField, run the following command in a prompt/terminal window:

```
pip install git+https://github.com/asche1/PyCrystalField.git@master
```

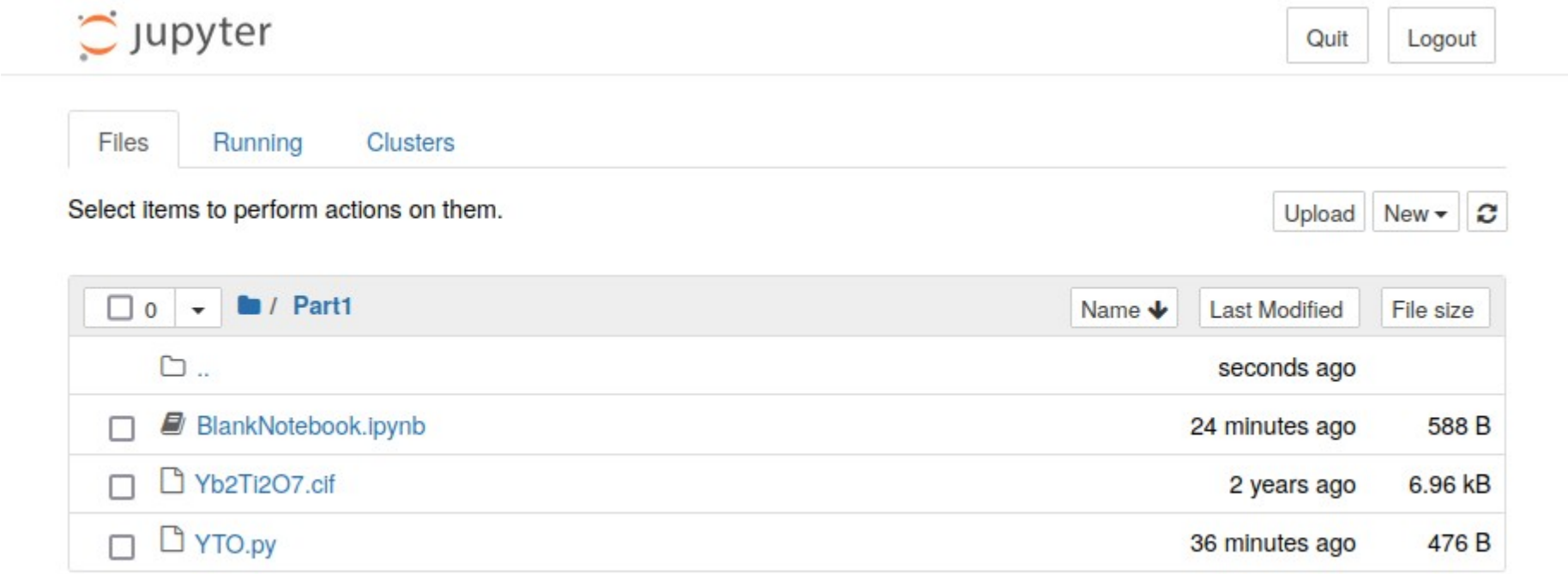
Part 1: Import a .cif and calculate a point charge model

Open Jupyter notebook: run the following in a prompt/terminal window:



Then, navigate to the `PCF_tutorial/Part1` and open `BlankNotebook.ipynb`

Open Jupyter notebook: run the following in a prompt/terminal window:



Then, navigate to the `PCF_tutorial/Part1` and open `BlankNotebook.ipynb`

First things first: import the libraries

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

```

Line 4, Column 1 Tab Size: 4 Python

First things first: import the libraries

numerics and
plotting libraries

PyCrystalField

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef
```

Line 4, Column 1 Tab Size: 4 Python

Next, use PycrystalField to import a .cif file

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')
```

Line 7, Column 43 Tab Size: 4 Python

Next, use PycrystalField to import a .cif file

ImportCIF output:

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')
```

Line 7, Column 43 Tab Size: 4 Python

“Ligands” object
Used for point charge calculations

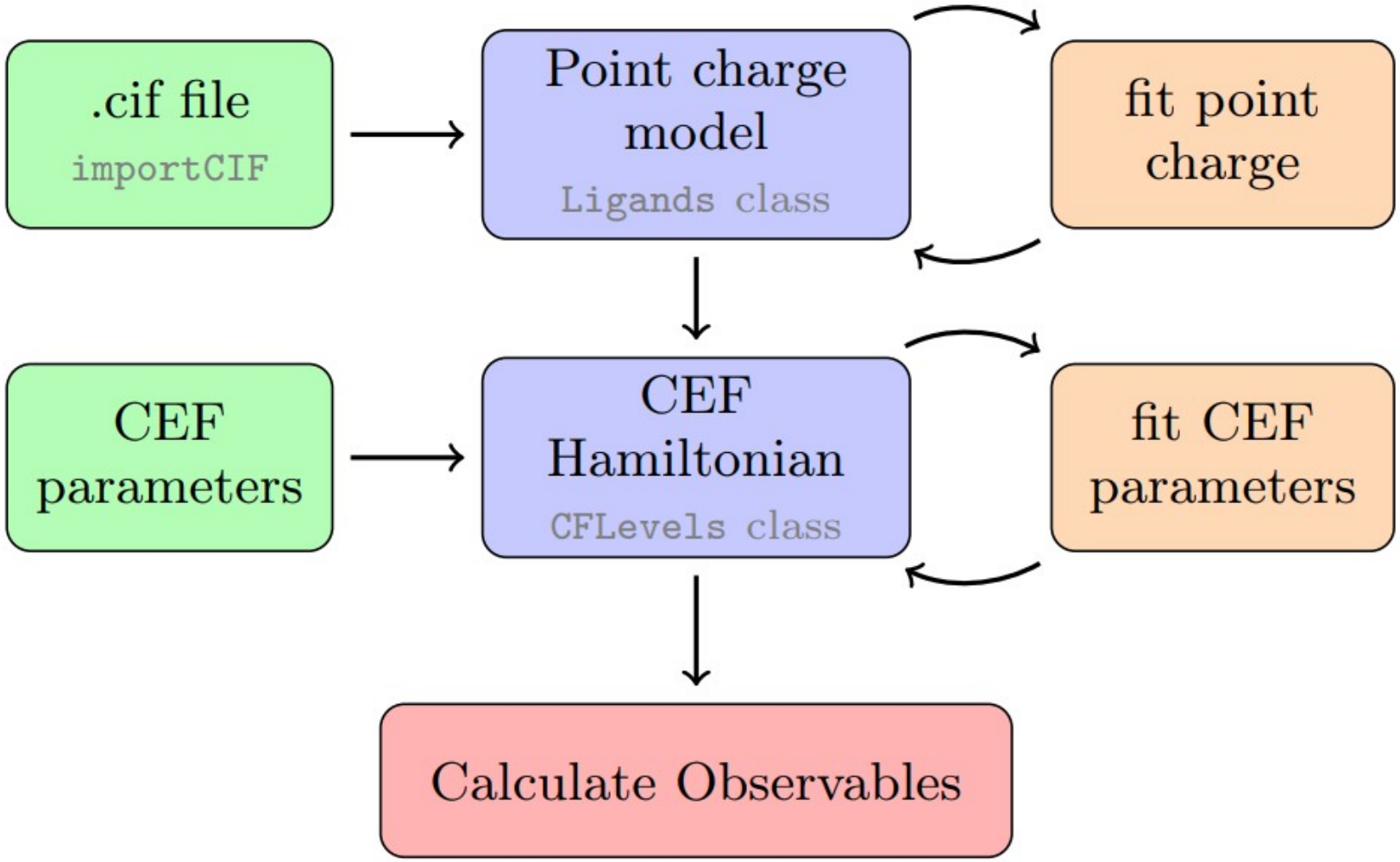
“CFLevels” object
Used for diagonalizing CEF Hamiltonians and calculating observables

ImportCIF input:

- .cif file
- Magnetic ion (automatically picks the first rare earth ion if none specified)
- Number of neighbors (optional)

...

Relationship between `Ligands` and `CFLevels` objects:



Scheie, J. Appl. Cryst. (2021)

Running the code as is:

- imports the .cif file
- identifies the nearest neighbor ligand
- Builds a ligand shell
- Calculates the highest symmetry direction to put the z axis (in this case, along a 3-fold axis)
  - *Note: if symmetry is very low, PyCrystalField calculates near-symmetries using continuous shape measures to assign axes.*

Stevens operators from a point charge model

```
File  Edit  View  Search  Terminal  Help
(base)                :~/Documents/Conferences/2022_CrystalFields/PCF_tutorial/Part1$ python YTO.py
 ******************************************************
 *                PyCrystalField 2.3.3                *
 *  Please cite  J. Appl. Cryst. (2021). 54, 356-362  *
 *    <https://doi.org/10.1107/S160057672001554X>     *
 ******************************************************

Importing atoms
   104 atoms added
.cif import complete.
No mag_ion ion listed, assuming Yb1 is the central ion.
Central ion: Yb3+ at [0.5, 0.5, 0.5]
    AAAH! There is a super-close atom. Removing it...
 Nearest ligand: O2-
   Identified 8 O2- ligands.
   Found 3 fold axis about [1. 1. 1.]
   Found mirror plane: [-0.70710678  0.          0.70710678]

  Axes for point charge model (in ABC space):
        X axis = [-0.5  1.  -0.5]
        Y axis = [-1.  0.  1.]
        Z axis = [1. 1. 1.]

   Creating a point charge model...
B_2 0  =  0.53552196
B_2 1  =  -0.0
B_2 2  =  -0.0
B_4 0  =  -0.04430884
B_4 1  =  0.0
B_4 2  =  -0.0
B_4 3  =  0.33472584
B_4 4  =  -0.0
B_6 0  =  0.00032957
B_6 1  =  -0.0
B_6 2  =  -0.0
B_6 3  =  0.00302494
B_6 4  =  -0.0
B_6 5  =  -0.0
B_6 6  =  0.00313667
(base)                :~/Documents/Conferences/2022_CrystalFields/PCF_tutorial/Part1$
```

Now, let’s calculate the eigenvectors and eigenvalues:

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printEigenvectors()
```

Line 11, Column 23 Tab Size: 4 Python

Now, let’s calculate the eigenvectors and eigenvalues:

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printEigenvectors()
```

Line 11, Column 23 Tab Size: 4 Python

```
File  Edit  View  Search  Terminal  Help
B_6 3  =  0.00302494
B_6 4  =  -0.0
B_6 5  =  -0.0
B_6 6  =  0.00313667

 Eigenvalues     Eigenvectors
                ----------------------------------------------------------------
0.00000         |  [-0.374  0.     0.    -0.921  0.     0.     0.105  0.   ]  |
0.00000         |  [ 0.     0.105  0.     0.     0.921  0.     0.    -0.374]  |
38.61543        |  [ 0.     0.001  0.     0.    -0.376  0.     0.    -0.926]  |
38.61543        |  [-0.926  0.     0.     0.376  0.     0.     0.001  0.   ]  |
47.10950        |  [ 0.     0.     0.218  0.     0.    -0.976  0.     0.   ]  |
47.10950        |  [0.    0.    0.976 0.    0.    0.218 0.    0.   ]  |
75.14028        |  [0.04  0.    0.    0.097 0.    0.    0.994 0.   ]  |
75.14028        |  [ 0.     0.994  0.     0.    -0.097  0.     0.     0.04 ]  |
(base)          :~/Documents/Conferences/2022_CrystalFields/PCF_tutorial/Part1$
```

Output:
- level energies
- Eigenvectors

Now, let’s calculate the eigenvectors and eigenvalues:

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printEigenvectors()
```

Line 11, Column 23 Tab Size: 4 Python

```
File  Edit  View  Search  Terminal  Help
B_6 3  =  0.00302494
B_6 4  =  -0.0
B_6 5  =  -0.0
B_6 6  =  0.00313

 Eigenvalues      Eigenvectors
                 ----------------------------------------------------------------
0.00000          |  [-0.374  0.     0.    -0.921  0.     0.     0.105  0.   ]  |
0.00000          |  [ 0.     0.105  0.     0.     0.921  0.     0.    -0.374]  |
38.61543         |  [ 0.     0.001  0.     0.    -0.376  0.     0.    -0.926]  |
38.61543         |  [-0.926  0.     0.     0.376  0.     0.     0.001  0.   ]  |
47.10950         |  [ 0.     0.     0.218  0.     0.    -0.976  0.     0.   ]  |
47.10950         |  [0.    0.    0.976 0.    0.    0.218 0.    0.   ]  |
75.14028         |  [0.04  0.    0.    0.097 0.    0.    0.994 0.   ]  |
75.14028         |  [ 0.     0.994  0.     0.    -0.097  0.     0.     0.04 ]  |
(base)           :~/Documents/Conferences/2022_CrystalFields/PCF_tutorial/Part1$
```

Yb^{3+} is a J=7/2 ion, so the matrix shows
m_J=-7/2, m_J=-5/2, m_J=-3/2, m_J=-1/2, m_J=1/2, m_J=3/2, m_J=5/2, m_J=7/2

Output:
- level energies
- Eigenvectors

Now, let’s calculate the eigenvectors and eigenvalues:

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printEigenvectors()
```

Line 11, Column 23 Tab Size: 4 Python

```
File  Edit  View  Search  Terminal  Help
B_6 3  =  0.00302494
B_6 4  =  -0.0
B_6 5  =  -0.0
B_6 6  =  0.00313

 Eigenvalues      Eigenvectors
                 ----------------------------------------------------------------
0.00000          |  [-0.374  0.     0.    -0.921  0.     0.     0.105  0.   ]  |
0.00000          |  [ 0.     0.105  0.     0.     0.921  0.     0.    -0.374]  |
38.61543         |  [ 0.     0.001  0.     0.    -0.376  0.     0.    -0.926]  |
38.61543         |  [-0.926  0.     0.     0.376  0.     0.     0.001  0.   ]  |
47.10950         |  [ 0.     0.     0.218  0.     0.    -0.976  0.     0.   ]  |
47.10950         |  [0.    0.    0.976 0.    0.    0.218 0.    0.   ]  |
75.14028         |  [0.04  0.    0.    0.097 0.    0.    0.994 0.   ]  |
75.14028         |  [ 0.     0.994  0.     0.    -0.097  0.     0.     0.04 ]  |
(base)           :~/Documents/Conferences/2022_CrystalFields/PCF_tutorial/Part1$
```

Yb^{3+} is a J=7/2 ion, so the matrix shows
m_J=-7/2, m_J=-5/2, m_J=-3/2, m_J=-1/2, m_J=1/2, m_J=3/2, m_J=5/2, m_J=7/2

Output:
- level energies
- Eigenvectors

<u>What this tells us:</u>
1. The ground state doublet is ~40 meV separated from the first excited state
2. The ground state doublet is dominated by m_J=±1/2

Side-note: the command printLaTexEigenvectors outputs a LaTeX readable table:

```
Yb.printLaTexEigenvectors()
```

```
File  Edit  View  Search  Terminal  Help
\begin{table*}
\caption{Eigenvectors and Eigenvalues...}
\begin{ruledtabular}
\begin{tabular}{c|cccccccc}
E (meV) &$| -\frac{7}{2}\rangle$ & $| -\frac{5}{2}\rangle$ & $| -\frac{3}{2}\rangle$ & $| -\f
rac{1}{2}\rangle$ & $| \frac{1}{2}\rangle$ & $| \frac{3}{2}\rangle$ & $| \frac{5}{2}\rangle$
& $| \frac{7}{2}\rangle$ \tabularnewline
 \hline
0.000 & -0.3742 & 0.0 & 0.0 & -0.9214 & 0.0 & 0.0 & 0.1053 & 0.0 \tabularnewline
0.000 & 0.0 & 0.1053 & 0.0 & 0.0 & 0.9214 & 0.0 & 0.0 & -0.3742 \tabularnewline
38.615 & 0.0 & 0.0008 & 0.0 & 0.0 & -0.3764 & 0.0 & 0.0 & -0.9265 \tabularnewline
38.615 & -0.9265 & 0.0 & 0.0 & 0.3764 & 0.0 & 0.0 & 0.0008 & 0.0 \tabularnewline
47.109 & 0.0 & 0.0 & 0.2183 & 0.0 & 0.0 & -0.9759 & 0.0 & 0.0 \tabularnewline
47.109 & 0.0 & 0.0 & 0.9759 & 0.0 & 0.0 & 0.2183 & 0.0 & 0.0 \tabularnewline
75.140 & 0.0404 & 0.0 & 0.0 & 0.0972 & 0.0 & 0.0 & 0.9944 & 0.0 \tabularnewline
75.140 & 0.0 & 0.9944 & 0.0 & 0.0 & -0.0972 & 0.0 & 0.0 & 0.0404 \tabularnewline
\end{tabular}\end{ruledtabular}
\label{flo:Eigenvectors}
\end{table*}
(base) 1o1@lap113771:~/Documents/Conferences/2022_CrystalFields/PCF_tutorial/Part1$
```

Render with LaTeX...

Table I. Eigenvectors and Eigenvalues...

E (meV)	$\lvert -\frac{7}{2}\rangle$	$\lvert -\frac{5}{2}\rangle$	$\lvert -\frac{3}{2}\rangle$	$\lvert -\frac{1}{2}\rangle$	$\lvert \frac{1}{2}\rangle$	$\lvert \frac{3}{2}\rangle$	$\lvert \frac{5}{2}\rangle$	$\lvert \frac{7}{2}\rangle$
0.000	-0.3742	0.0	0.0	-0.9214	0.0	0.0	0.1053	0.0
0.000	0.0	0.1053	0.0	0.0	0.9214	0.0	0.0	-0.3742
38.615	0.0	0.0008	0.0	0.0	-0.3764	0.0	0.0	-0.9265
38.615	-0.9265	0.0	0.0	0.3764	0.0	0.0	0.0008	0.0
47.109	0.0	0.0	0.2183	0.0	0.0	-0.9759	0.0	0.0
47.109	0.0	0.0	0.9759	0.0	0.0	0.2183	0.0	0.0
75.140	0.0404	0.0	0.0	0.0972	0.0	0.0	0.9944	0.0
75.140	0.0	0.9944	0.0	0.0	-0.0972	0.0	0.0	0.0404

Last step, let's calculate the neutron spectrum with the CFLevels object:

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printLaTexEigenvectors()

########### plot neutron spectrum

hw = np.linspace(0,100,200)
intens = Yb.normalizedNeutronSpectrum(hw, Temp=20,
    ResFunc= lambda x: 4, gamma = 1)
```

Line 16, Column 52 Tab Size: 4 Python

What is this function?

```
intens = Yb.normalizedNeutronSpectrum(hw, Temp=20, ResFunc= lambda x: 4, gamma = 1)
```

Array of energy values

Temperature (in Kelvin)

Gaussian peak width resolution function, in meV
(can in general be energy dependent, in this case it is not.)

Lorentzian peak width in meV

What is this function?

```
intens = Yb.normalizedNeutronSpectrum(hw, Temp=20, ResFunc= lambda x: 4, gamma = 1)
```

Array of energy values

Temperature (in Kelvin)

Gaussian peak width resolution function, in meV
(can in general be energy dependent, in this case it is not.)

Lorentzian peak width in meV

Note: "`normalizedNeutronSpectrum`" should be used when the k_f/k_i factor is not present in the data (typical for time of flight measurements).

If this correction is present, use "`neutronSpectrum`" (must provide the incident energy).

If you're not sure, ask your instrument scientist!

...and plot the calculated data with MatPlotLib

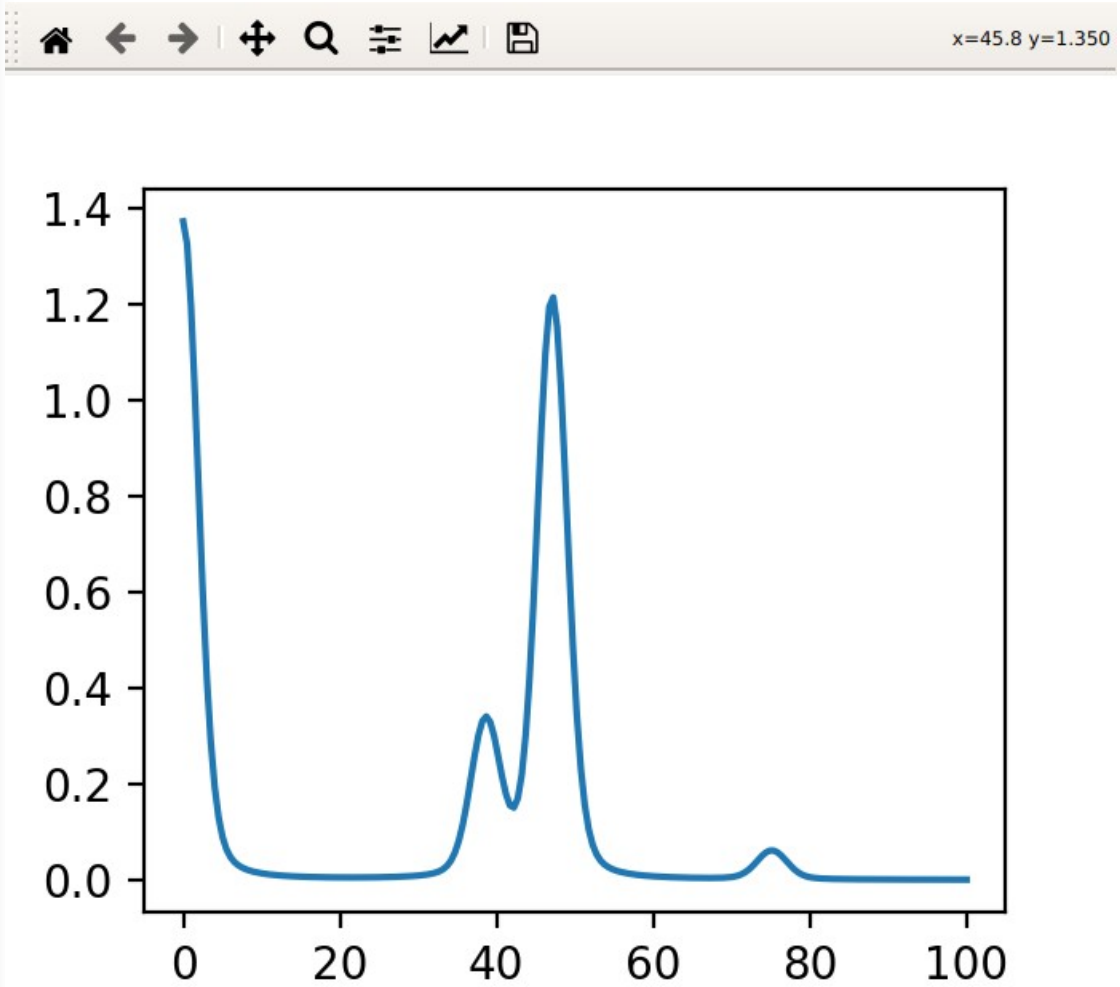
```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printEigenvectors()

########### plot neutron spectrum

hw = np.linspace(0,100,200)

intens = Yb.normalizedNeutronSpectrum(hw, Temp=20,
    ResFunc= lambda x: 4, gamma = 1)

plt.figure()
plt.plot(hw, intens)
plt.show()
```

Line 22, Column 11 Tab Size: 4 Python

...and plot the calculated data with MatPlotLib

```python
import numpy as np
import matplotlib.pyplot as plt
import PyCrystalField as cef

########### Import CIF file

YTOLig, Yb = cef.importCIF('Yb2Ti2O7.cif')

########### print eigenvectors

Yb.printEigenvectors()

########### plot neutron spectrum

hw = np.linspace(0,100,200)

intens = Yb.normalizedNeutronSpectrum(hw, Temp=20,
    ResFunc= lambda x: 4, gamma = 1)

plt.figure()
plt.plot(hw, intens)
plt.show()
```

Line 22, Column 11 Tab Size: 4 Python



Final bit: calculate the g-tensor

```python

########### plot neutron spectrum

hw = np.linspace(0,100,200)

intens = Yb.normalizedNeutronSpectrum(hw, Temp=20,
    ResFunc= lambda x: 4, gamma = 1)

plt.figure()
plt.plot(hw, intens)
plt.show()

########### print g tensor

g = Yb.gtensor()
print('G tensor:\n',g)
```

Output:

```
G tensor:
 [[-4.11887922  0.          0.        ]
 [ 0.         -4.11887922  0.        ]
 [ 0.          0.          2.02703635]]
```

Part 2: Fit neutron scattering data

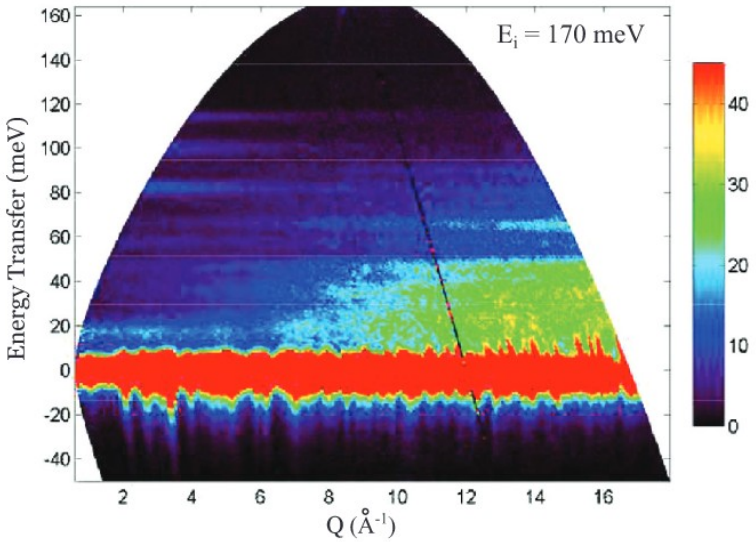
Let’s use some legacy data:

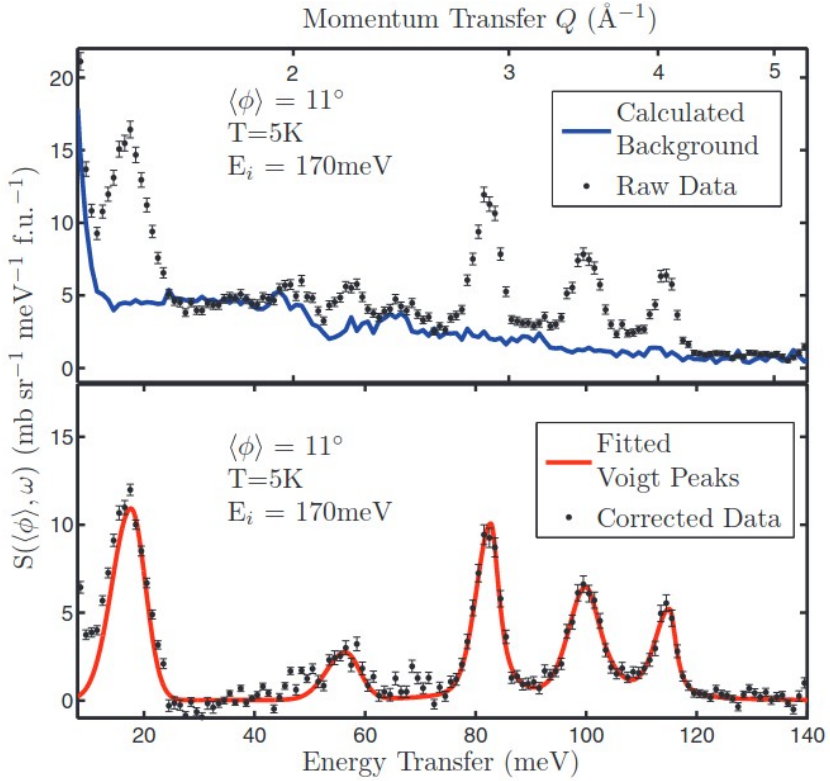
PHYSICAL REVIEW B **88**, 104421 (2013)

Crystal-field states of Pr^{3+} in the candidate quantum spin ice $Pr_2Sn_2O_7$

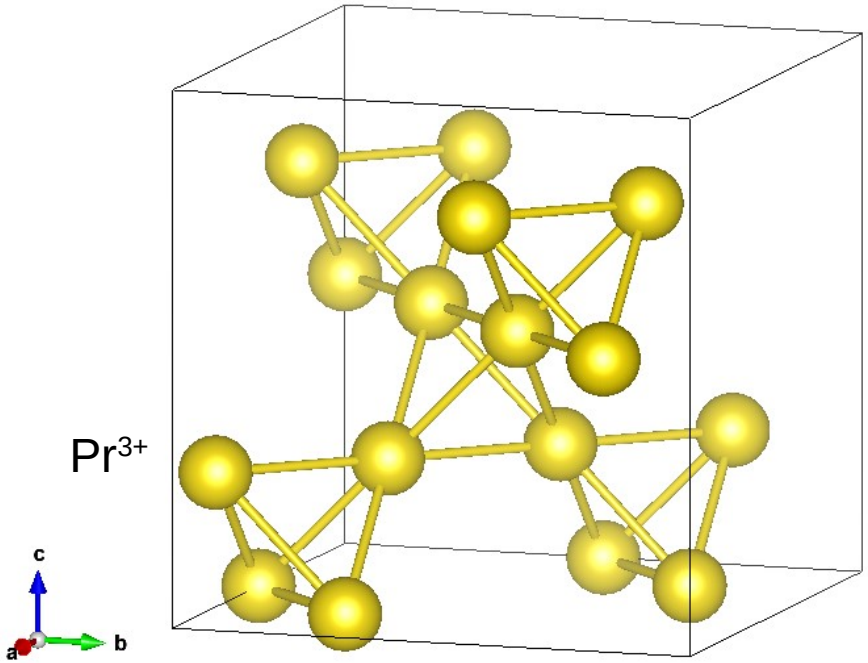
A. J. Princep,* D. Prabhakaran, and A. T. Boothroyd†
Department of Physics, University of Oxford, Clarendon Laboratory, Parks Road, Oxford, OX1 3PU, United Kingdom

D. T. Adroja
ISIS Facility, Rutherford Appleton Laboratory, STFC, Chilton, Didcot, Oxon, OX11 0QX, United Kingdom
(Received 4 June 2013; revised manuscript received 2 September 2013; published 23 September 2013)

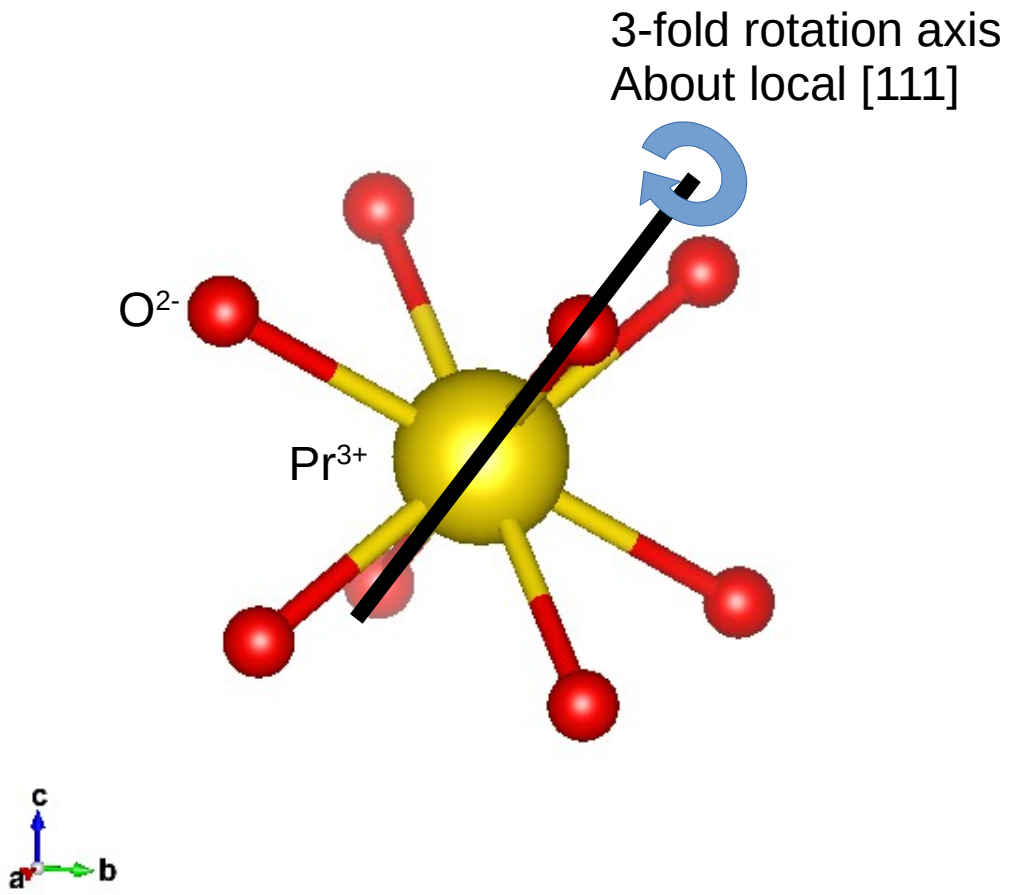




$Pr_2Sn_2O_7$ crystal structure:



Pyrochlore lattice

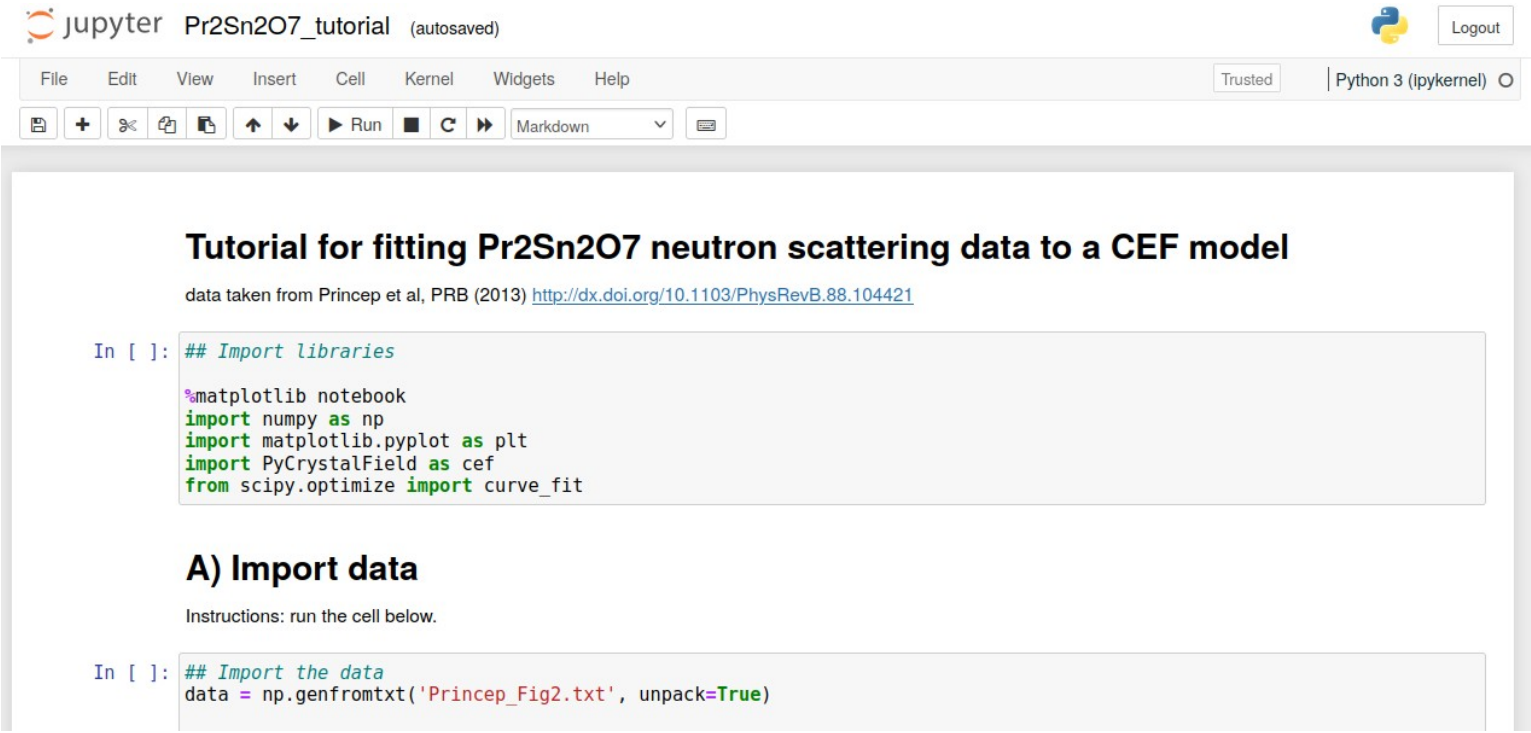


The procedure we'll follow:

1. Get eigenvalues from the data
2. Create a point charge model
3. Fit the effective charges (not necessary here, but sometimes this helps)
4. Fit the nonzero Stevens operators

Get started:

- Start Jupyter notebook
- Open Part2/PrSn2O7_tutorial.ipynb



Switch to the Jupyter notebook...

Our result:

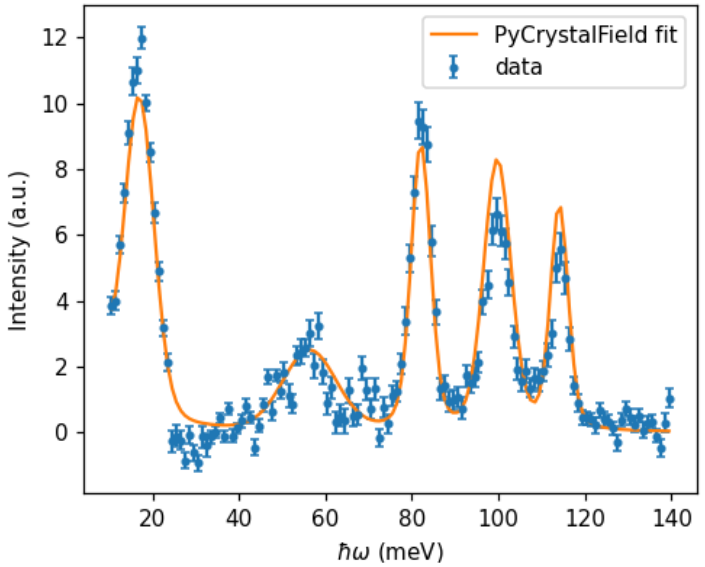
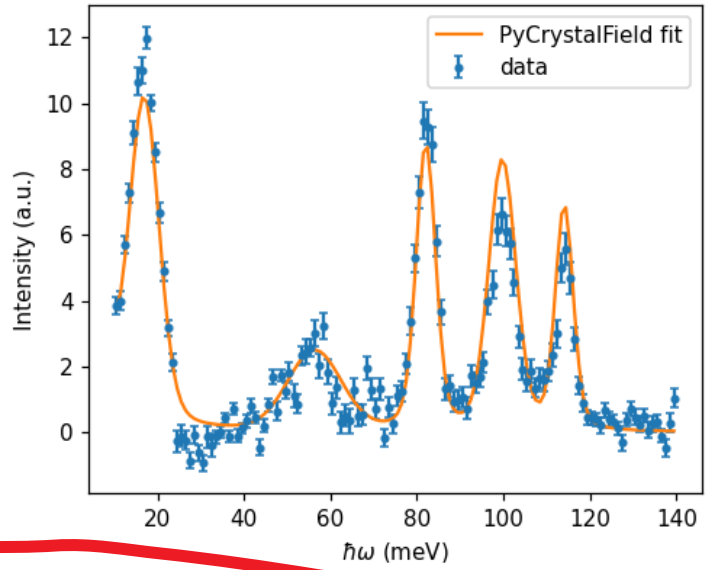


Table I. Eigenvectors and Eigenvalues...

E (meV)	$\|-4\rangle$	$\|-3\rangle$	$\|-2\rangle$	$\|-1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|2\rangle$	$\|3\rangle$	$\|4\rangle$
0.000	-0.944	0.0	0.0	-0.2746	0.0	0.0	0.183	0.0	0.0
0.000	0.0	0.0	0.183	0.0	0.0	0.2746	0.0	0.0	-0.944
16.916	0.0	0.2944	0.0	0.0	0.9092	0.0	0.0	-0.2944	0.0
56.315	-0.2907	0.0	0.0	0.9544	0.0	0.0	-0.0675	0.0	0.0
56.315	0.0	0.0	0.0675	0.0	0.0	0.9544	0.0	0.0	0.2907
82.063	0.0	-0.6429	0.0	0.0	0.4163	0.0	0.0	0.6429	0.0
99.752	0.0	0.0	0.9808	0.0	0.0	-0.117	0.0	0.0	0.1561
99.752	0.1561	0.0	0.0	0.117	0.0	0.0	0.9808	0.0	0.0
114.093	0.0	-0.7071	0.0	0.0	0.0	0.0	0.0	-0.7071	0.0

Our result:

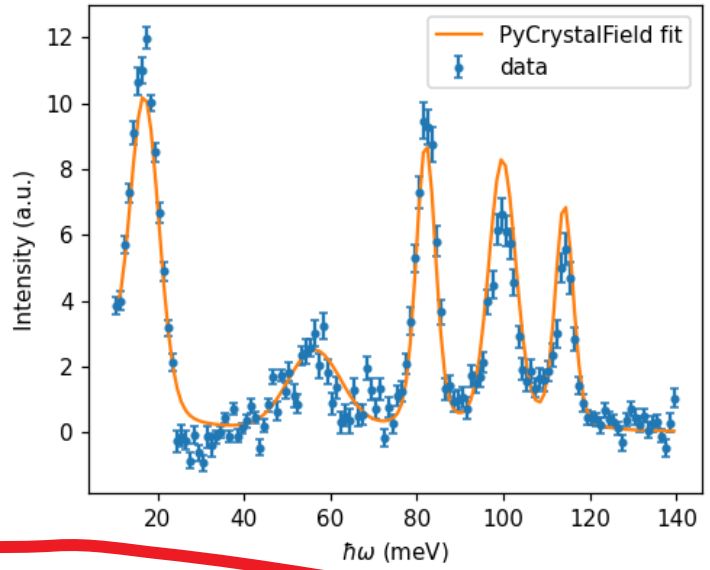


Acts very classically

Table I. Eigenvectors and Eigenvalues...

E (meV)	$\|-4\rangle$	$\|-3\rangle$	$\|-2\rangle$	$\|-1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|2\rangle$	$\|3\rangle$	$\|4\rangle$
0.000	-0.944	0.0	0.0	-0.2746	0.0	0.0	0.183	0.0	0.0
0.000	0.0	0.0	0.183	0.0	0.0	0.2746	0.0	0.0	-0.944
16.916	0.0	0.2944	0.0	0.0	0.9092	0.0	0.0	-0.2944	0.0
56.315	-0.2907	0.0	0.0	0.9544	0.0	0.0	-0.0675	0.0	0.0
56.315	0.0	0.0	0.0675	0.0	0.0	0.9544	0.0	0.0	0.2907
82.063	0.0	-0.6429	0.0	0.0	0.4163	0.0	0.0	0.6429	0.0
99.752	0.0	0.0	0.9808	0.0	0.0	-0.117	0.0	0.0	0.1561
99.752	0.1561	0.0	0.0	0.117	0.0	0.0	0.9808	0.0	0.0
114.093	0.0	-0.7071	0.0	0.0	0.0	0.0	0.0	-0.7071	0.0

Our result:



Agrees with Princep, PRB (2013)

$$\Gamma_3^+(0\,\mathrm{meV}) = 0.88|^3H_4, \pm 4\rangle$$

Acts very classically

Table I. Eigenvectors and Eigenvalues...

E (meV)	$\|-4\rangle$	$\|-3\rangle$	$\|-2\rangle$	$\|-1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|2\rangle$	$\|3\rangle$	$\|4\rangle$
0.000	-0.944	0.0	0.0	-0.2746	0.0	0.0	0.183	0.0	0.0
0.000	0.0	0.0	0.183	0.0	0.0	0.2746	0.0	0.0	-0.944
16.916	0.0	0.2944	0.0	0.0	0.9092	0.0	0.0	-0.2944	0.0
56.315	-0.2907	0.0	0.0	0.9544	0.0	0.0	-0.0675	0.0	0.0
56.315	0.0	0.0	0.0675	0.0	0.0	0.9544	0.0	0.0	0.2907
82.063	0.0	-0.6429	0.0	0.0	0.4163	0.0	0.0	0.6429	0.0
99.752	0.0	0.0	0.9808	0.0	0.0	-0.117	0.0	0.0	0.1561
99.752	0.1561	0.0	0.0	0.117	0.0	0.0	0.9808	0.0	0.0
114.093	0.0	-0.7071	0.0	0.0	0.0	0.0	0.0	-0.7071	0.0

A method for defining uncertainty:

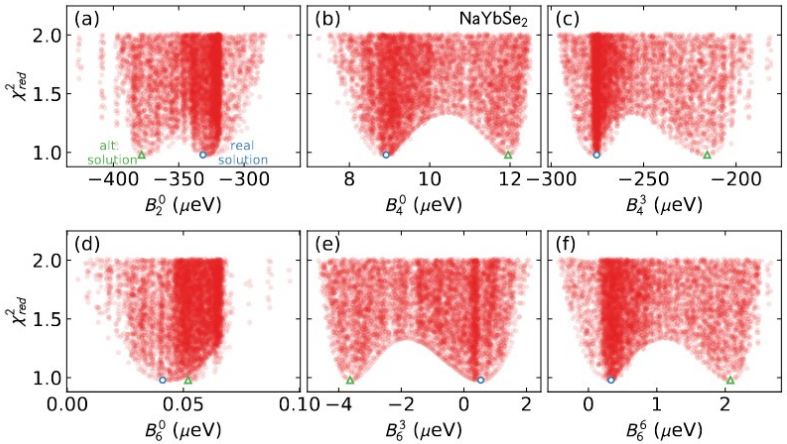
SciPost Phys. Core 5, 018 (2022)

Quantifying uncertainties in crystal electric field Hamiltonian fits to neutron data

Allen Scheie

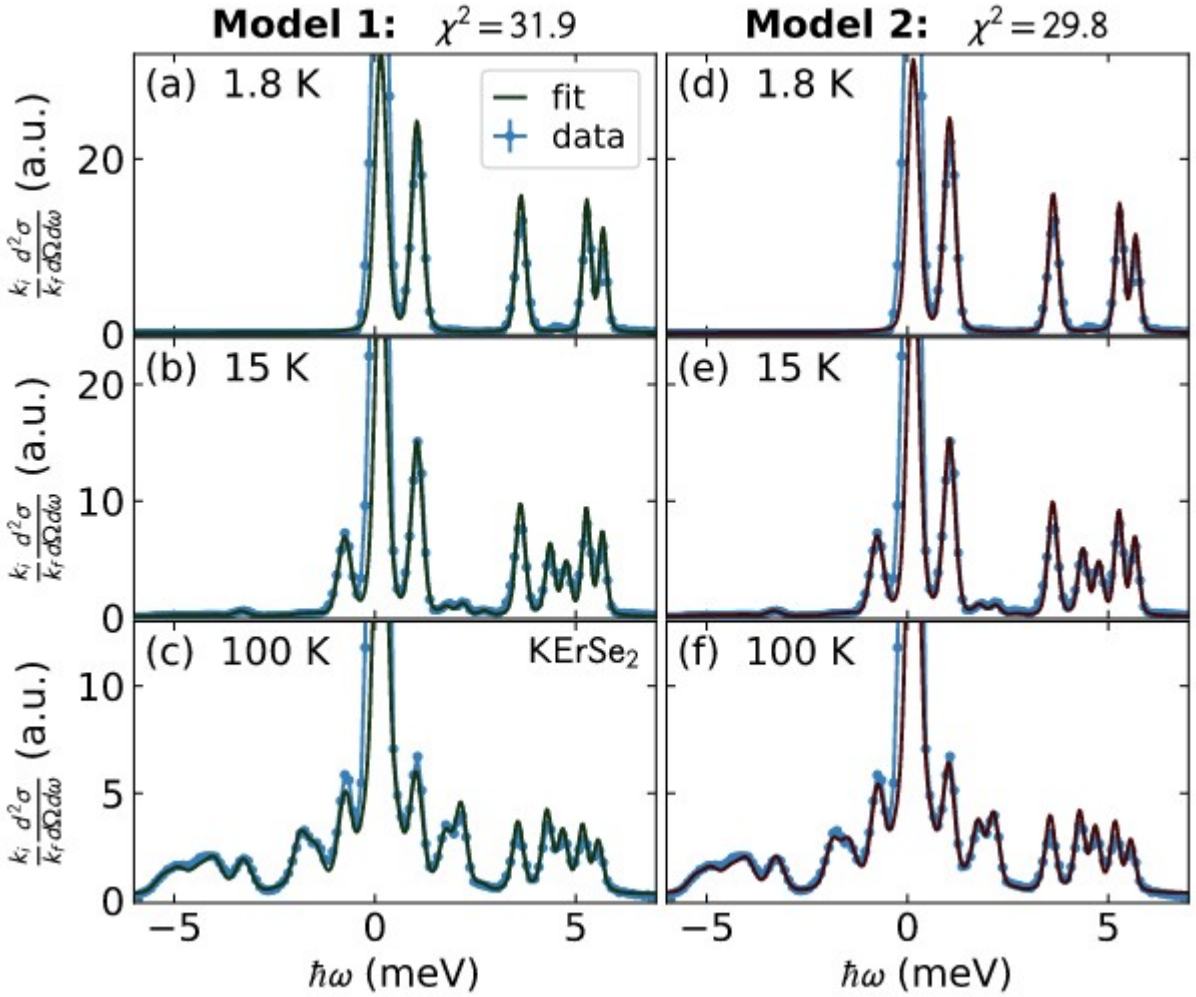
Neutron Scattering Division, Oak Ridge National Laboratory,
Oak Ridge, Tennessee 37831, USA

Key result: a neutron scattering data fit can be badly underdetermined! Explore nearby local minima and compare to bulk data.



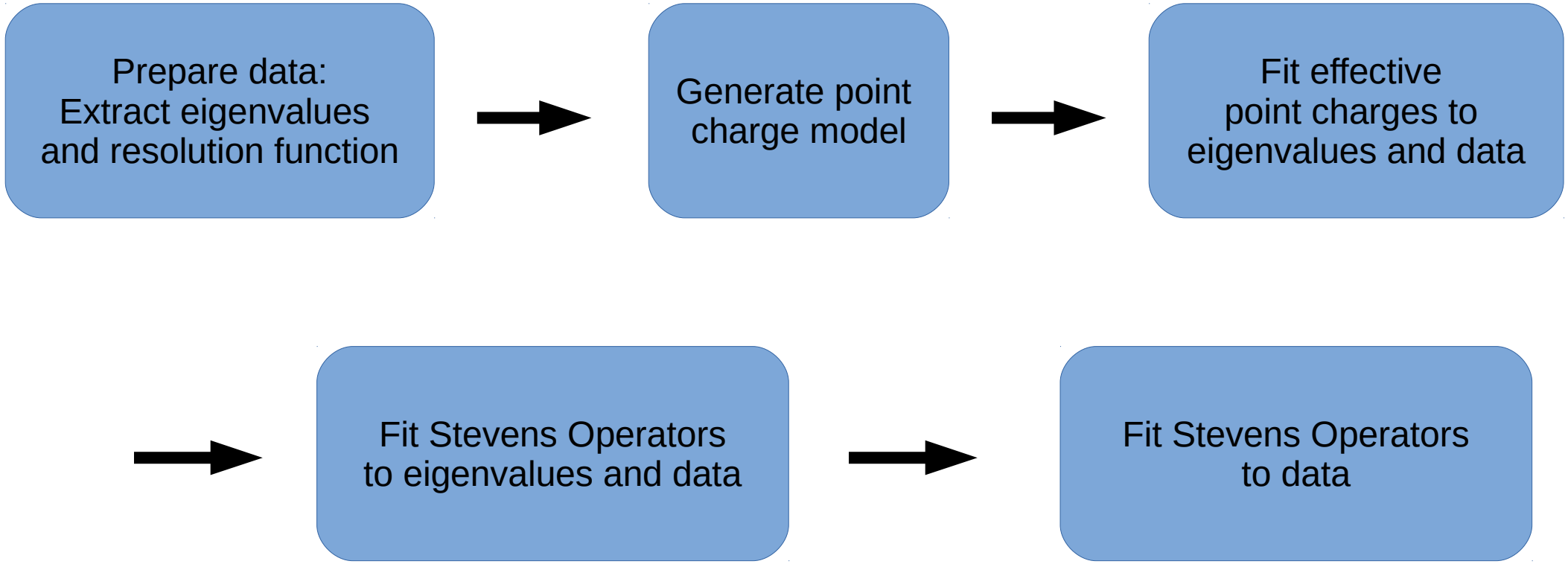
Example of this ambiguity: $KErSe_2$

Magnetization showed that **Model 1** was correct!



Scheie et al, PRB **101**, 144432 (2020)

Summary of the general workflow:

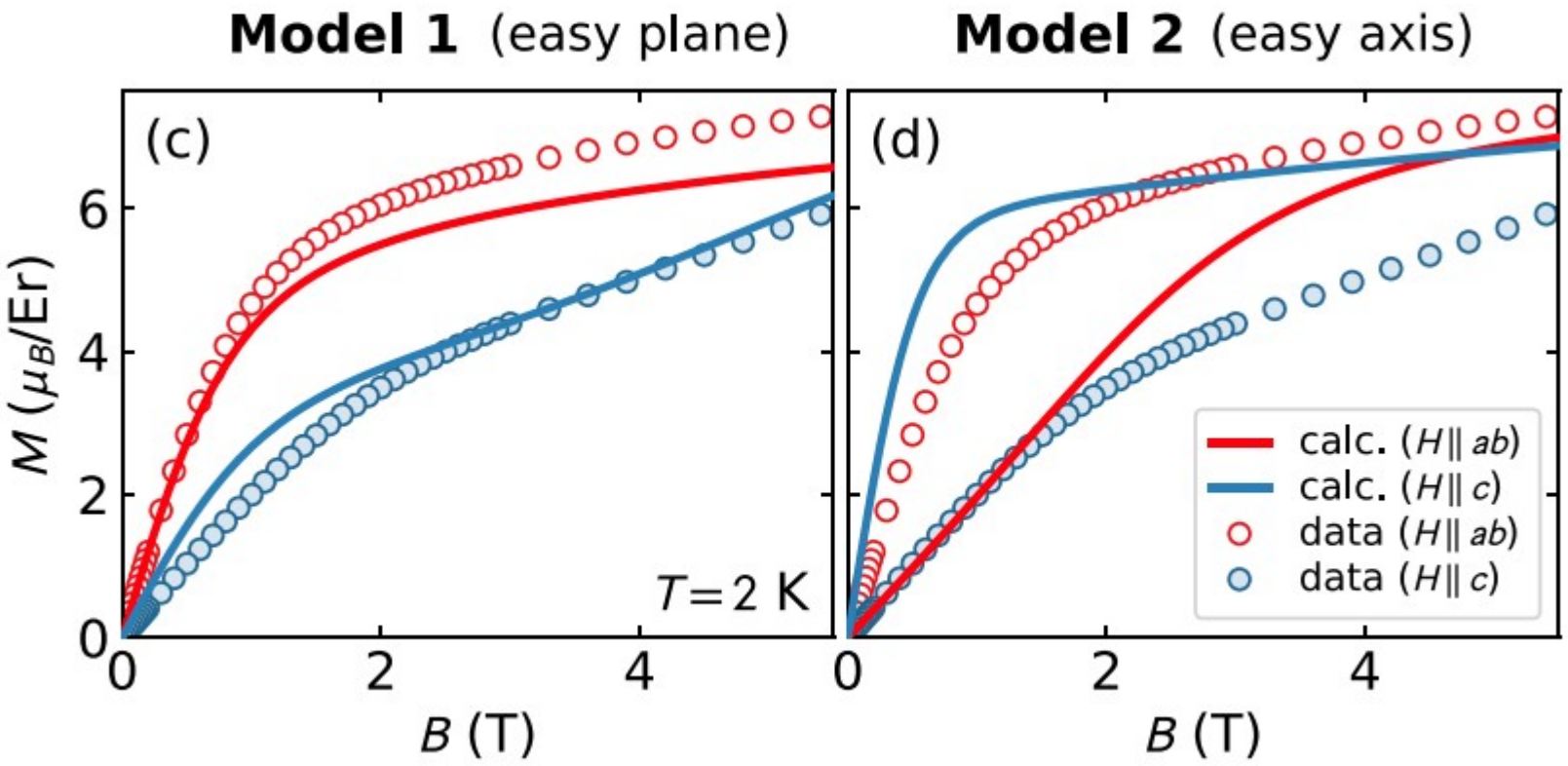


Tips for getting fits to converge:

- Fit iteratively: run a new fit using the starting values from the previous fit, but with a different weight given to the eigenvalue χ^2.
  - Might need to “kick” it out of a local minimum by changing a parameter or two.
- Define the resolution function carefully so the peak widths match experiment.
- Try different optimization routines (e.g., “Powell” vs. “Nelder-Mead”)

Part 3: calculating magnetization and susceptibility

It is helpful to compare fitted Hamiltonians to bulk heat capacity and magnetization.



Scheie et al, PRB **101**, 144432 (2020)

An example of how to do this with PyCrystalField: Part3/KES.py

We use a point charge model of delafossite $KErSe_2$

```
########### Import CIF file

KESLig, Er = cef.importCIF('KErSe2.cif')

########### print eigenvectors

Er.printEigenvectors()
```

An example of how to do this with PyCrystalField: Part3/KES.py

Calculating magnetization:

```
########### calculate magnetization at 2 K

temp = 2 #temperature in K

FieldStrengths = np.linspace(-10,10,101)
magnetization = np.zeros((len(FieldStrengths), 3, 3))
ion='Er3+'
for i, fs in enumerate(FieldStrengths):
    magnetization[i,0] = Er.magnetization(ion, temp, [fs,0,0]) # field along X
    magnetization[i,1] = Er.magnetization(ion, temp, [0,fs,0]) # field along Y
    magnetization[i,2] = Er.magnetization(ion, temp, [0,0,fs]) # field along Z
```

Notes:

- The `Er.magnetization` command returns a three-component vector, in case off-diagonal magnetization response is of interest.
- This is in the local axis frame, NOT the unit cell frame. If you have multiple ions at different angles (like pyrochlores), you need to calculate many magnetization directions and average.

An example of how to do this with PyCrystalField: Part3/KES.py

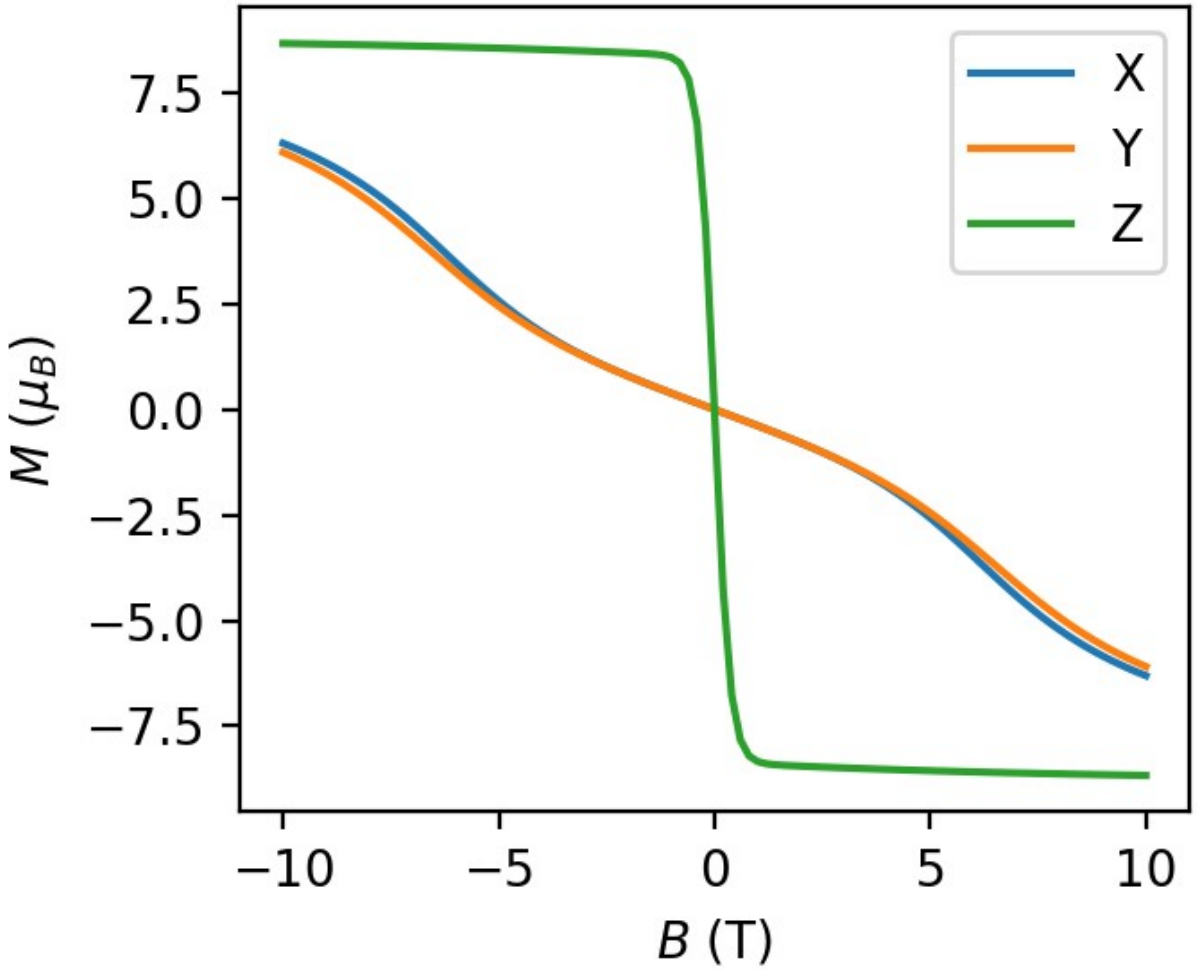
Calculating susceptibility:

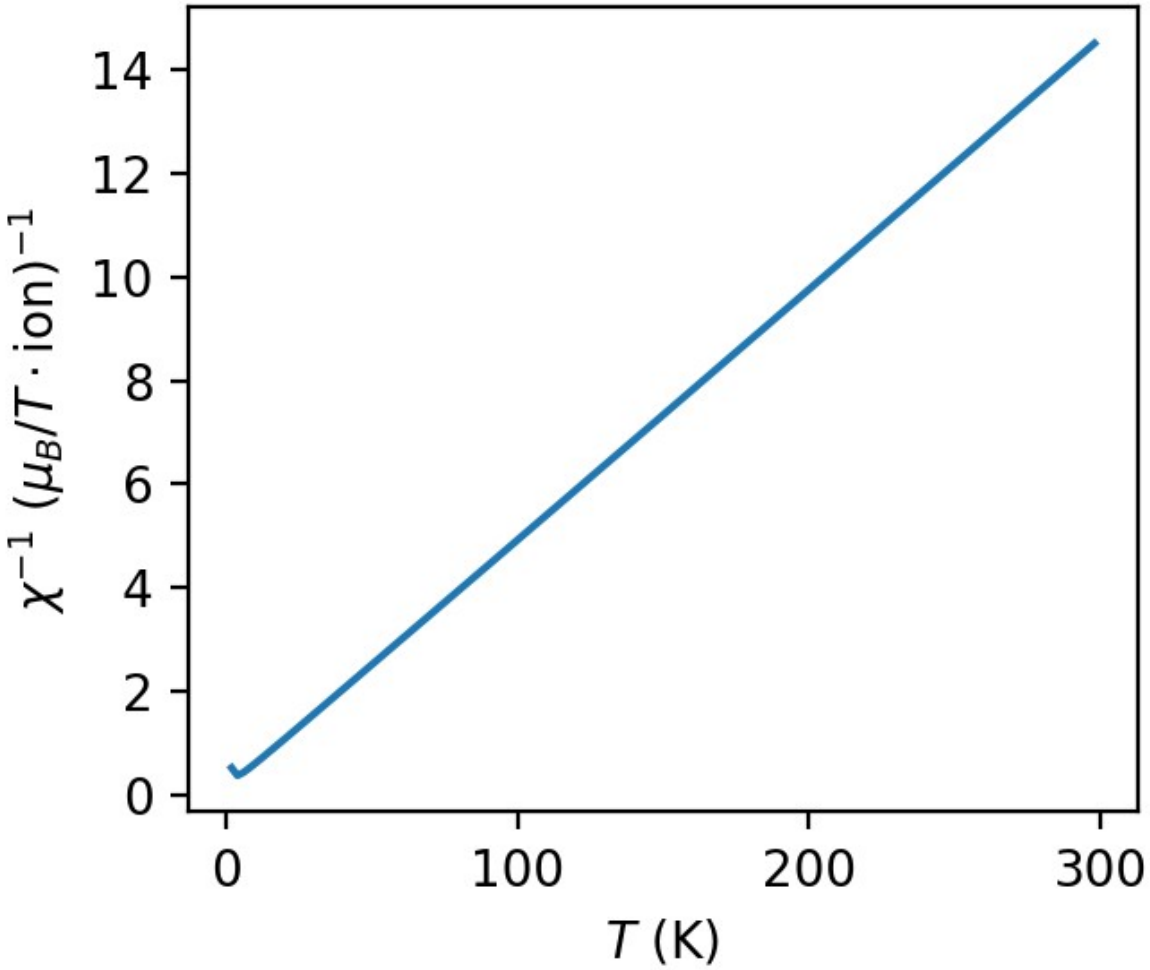
```
########### Calculate temperature dependent susceptibility
### in a 0.5 T field


Temperatures = np.arange(2,300,2)

# # For single crystal susceptibility, Field is a vector
# CalcSuscep = Er.susceptibility(ion='Er3+', Temps=Temperatures,
#                                Field=[0.5, 0 ,0], deltaField=0.001)

# For powder average susceptibility, Field is a scalar
CalcSuscep = Er.susceptibility(ion='Er3+', Temps=Temperatures,
                               Field=0.5, deltaField=0.001)
```

Notes:

- `deltaField` is the difference in field used to take a numerical derivative.
- For single crystal susceptibility, the `Field` variable should be a three-component vector.

Output:





Magnetization/susceptibility can be used to check fits.

- Be careful: magnetic exchange will in general affect the magnetization, which can be hard to model.
- Because of this, I have had very limited success fitting a model using magnetization and susceptibility.