CrysFiPy Documentation

Release 0.5

Petr Cermak

Jan 18, 2018

Contents

class crysfipy.reion.**cfpars**(**args*, ***kwargs*)

Class representing set of crystal field parameters.

It simplifies the creation of the CF parameter sets considering symmetry of the environment. Other modules expect that CF parameters are in *meV* (SI units). But if you just want to diagonalize Hamiltonian, it is possible to use *K* (and results will be in *K*).

Initialization can be done with named arguments or without. If arguments are not named, symmetry is considered from the first string argument. Stevens parameters are considered differently for different symmetries in following order:

cubic: B40, B60

hexagonal: B20, B40, B44, B66

tetragonal: B20, B40, B44, B60, B64

orthorombic: B20, B22, B40, B42, B44, B60, B62, B64, B66

BXY

float, optional – Attribute corresponding to B_X^Y Stevens Parameters. See Hutchings. If at least one CF parameter is specified as a named argument, non-named numerical parameters are ignored.

sym

str, optional – Symmetry of the crystal field

c - cubix

h - hexagonal

t - tetragonal

o - orthorombic (default)

Examples

Create set of CF parameters by named parameters:

```
>>> print(cfpars(sym = "c", B40 = 10))
Set of CF parameters for cubic symmetry:
B40 = 10.0000
B60 = 0.0000
B44 = 50.0000
B64 = 0.0000
```

Use of non-named parameters:

```
>>> print(cfpars("c", 10, 1))
Set of CF parameters for cubic symmetry:
B40 = 10.0000
B60 = 1.0000
B44 = 50.0000
B64 = -21.0000
```

crysfipy.reion.**neutronint**(*ion*, *T*, *direction='t'*)

Returns matrix of energy and transition intensity at given temperature

Parameters

- **ion** (*crysfipy.reion.re*) – Rare-earth ion object
- **T** (*float*) – temperature in *K*
- **direction** (*str*) – Direction of the Q in which to calculate

t - powder (default)
x - using J_x
y - using J_y
z - using J_z

class crysfipy.reion.**re**(*name*, *field*, *cfp*, *calculate=True*)

Object representing rare-earth ion in CF potential

name
str – Name of the ion.

field
1D array of floats – external magnetic field applied in *T*.

cfp
crysfipy.reion.cfpars – Crystal field parameters

calculate
bool, optional – If true (default) then it automatically diagonalizes Hamiltonian and calculates energy levels.

Examples

```
>>> ce = re("Ce", [0,0,0], ["c", 10])
>>> print(ce)
Energy levels:
E(0) =  0.0000   2fold-degenerated
E(1) =  3600.0000        4fold-degenerated
```

getlevels()
Calculate degeneracy of the levels and sort the matrix

crysfipy.reion.**susceptibility**(*ion*, *T*)

Returns susceptibility calculated for given ion at given temperature

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

CHAPTER 2

Notebooks

For an easy introduction to CrysFiPy, we recommend to check following iPython notebooks:

```
In [10]: from crysfipy.reion import re, neutronint, susceptibility as susc
         import crysfipy.const as C
         import numpy as np

         np.set_printoptions(linewidth=130)
         np.set_printoptions(precision=2)
```

2.1 1) const.py

```
In [13]: template = """Information about {0.name} ion:
         ************************
         {0.J2p1:.0f} energy levels, J = {0.J}, gJ = {0.gJ}
         alpha = {0.Alpha}, beta = {0.Beta}, gamma = {0.Gamma}
         """
         print(template.format(C.ion("Ce")))
```

```
Information about ce ion:
************************
6 energy levels, J = 2.5, gJ = 0.857142857143
alpha = -0.0571428571429, beta = 0.00634920634921, gamma = 0.0
```

2.2 2) CF hamiltonian diagonalization

```
In [16]: # create reion object:
         ce = re("Ce", [1,0,0],
                 ["t", -0.173477508,
                 0.001084591,
                 -0.012701252,
                 -3.34835E-06,
```

```
                0.0000097,
                ])
        ce.calculate()
        ce.getlevels()
        print(ce)
```

```
Energy levels:
E(0) =  0.0000   1fold-degenerated
E(1) =  0.4180   1fold-degenerated
E(2) =  2.0374   1fold-degenerated
E(3) =  2.0787   1fold-degenerated
E(4) =  3.5225   1fold-degenerated
E(5) =  4.7905   1fold-degenerated
```

2.3 3) Calculation of susceptibility

```
In [17]: temps = [5,10,50,100,300]
         for T in temps:
             print("T = {0} K \tchi_CF = {1} uB/T".format(T, susc(ce, T)))
```

```
T = 5 K         chi_CF = 0.0288552475141 uB/T
T = 10 K        chi_CF = 0.00716223624106 uB/T
T = 50 K        chi_CF = 0.000224584463989 uB/T
T = 100 K       chi_CF = 3.65918038894e-05 uB/T
T = 300 K       chi_CF = -4.60660668533e-06 uB/T
```

2.4 4) Neutron intensities

```
In [20]: # tbd
```

CHAPTER 3

Installing CrysFiPy

There are two ways how to install CrysFiPy package.

3.1 Python Package Index - pip

Easier is to use pip.

If you do not already have `pip`, to install it first download get-pip.py and run it with the following command:

```
python get-pip.py
```

With `pip` installed, you can install the latest version of crysfipy with the command:

```
pip install crysfipy
```

New releases will be pushed to the package index automatically. If you wish to install the development version, you will need to follow the instructions for installation from source.

3.2 Installation from Source

To install from source, either download the master branch source from Bitbucket or clone the repository:

```
git clone https://bitbucket.org/cermak/crysfipy.git
```

From inside of the crysfipy directory install the package using:

```
python setup.py install
```

CHAPTER 4

Changelog

- : Innitial non-alpha release
- : Added first version of documentation

CHAPTER 5

Contact

If you have any questions regarding use of CrysFiPy, please contact authors of the code:

- Petr Čermák
- Jan Zubáč

To file new bugs or search existing ones, you may visit CrysFiPy's Bitbucket Issues page. In order to submit an issue, you need to register on Bitbucket.

CHAPTER 6

CrysFiPy

Crystal field suite for python. It allows to calculate energy levels, bulk properties and neutron scattering intensities on basis of crystal field calculations.

Warning:

Currently this is beta release. Many functions are missing and documentation is not finished. Feel free to contact us for help.

6.1 Roadmap

- Finish integration with BFK from McPhase, currently not public
- Finish hybridization with phonons, currently not public
- Integration to PIP repository
- Specific heat calculation (Schottky contribution)
- Integration of simulated annealing
- GUI (based on QT)

6.2 Installation

- TBD

6.3 Documentation

Documentation is available at http://crysfipy.rtfd.io/, or can be built using sphinx by navigating to the doc/ folder and executing `make html`; results will be in the `doc/_build/` folder.

To ask questions you may create a Bitbucket issue.

6.4 Contributions

Contributions may be made by submitting a pull-request for review using the fork-and-pull method on Bitbucket. Feature requests and bug reports can be made using the Bitbucket issues interface.

6.5 Copyright & Licensing

6.6 Disclaimer

Python Module Index

C

Index