Takin – An inelastic neutron scattering software package

T. Weber

21st June 2016

Contents

Introduction

Takin is a software package for inelastic neutron scattering which features a full GUI, but can also be used as a library. It can be employed for lattice and scattering plane visualisation, triple-axis and time-of-flight spectrometer resolution calculation, planning of measurements using a live convolution preview, and convolution fitting.

For questions please contact: `tobias.weber@tum.de`

1 Basic Usage

The main window is divided into two parts: The left-hand side is used for reciprocal-space visualisations: The tab "Reciprocal Lattice" shows the Bragg peaks in the current scattering plane and the scattering triangle, the tab "Projection" shows various projective views, e.g. for calculating Laue patterns. The right-hand side of the window depicts several real-space visualisations: Here, the real lattice and the crystal's unit cell are shown (tab "Real Lattice") as well as the current instrumental position for triple-axis spectrometers (tab "TAS Instrument") and for time-of-flight spectrometers (tab "TOF Instrument").

The lattice constants and angles of the crystal's unit cell are entered in the lower-left part of the Takin main window using either the "Real Lattice" or the "Recip. Lattice" tab. The tab "Recip. Plane" defines the scattering plane used by the spectrometer.

If a spacegroup is defined (lower-central part of the main window), the forbidden reflexes are excluded in the reciprocal-lattice view. Furthermore, the structure factor and the crystal's unit cell are displayed if the atom positions are known: they can be entered using the "Atoms..." button.

2 Resolution Calculation

Set-up

For calculating the instrumental resolution at a given q=(hkl) and E position the instrumental parameters have to be set-up first. This can be done using the option "Resolution" -> "Parameters..." in the menu bar.

In the "Resolution Parameters" dialog box, an algorithm has to be set first. This is done in the tab "Calculation". Depending on the selected algorithm various settings tabs will be enabled. These contain – among other things – the geometrical description of the instrument.

Once all parameters are entered, a valid resolution matrix should be displayed in the "Calculation" tab.

Instrument configurations can either be saved using the "Save..." button in the "Resolution Parameters" dialog or via the menu bar in the Takin main window. The latter option saves the resolution settings together with the crystal definition.

Resolution Visualisation

Clicking on "Resolution" -> "Ellipses..." in the menu bar shows a live view of the resolution function at the current instrument position. The blue (green) ellipse is a sliced (projected) view of the four-dimensional ellipsoid using the given projection planes.

Visualisation of the resolution ellipses can be done in several coordinate systems that are selected using the combo box at the bottom of the dialog: The first option shows the momentum transfers in inverse Angstroms in the standard coordinate system with x along Q and y perpendicular to Q. The second option chooses the crystal's rlu coordinate system with the h axis along x, k along y and l along z. The third option also uses the crystal's rlu system, but with the x and y axis along the Bragg reflexes chosen on the "Recip. Plane" tab in the Takin main window.

3 Resolution Convolution

In order to plan an experiment or to get an overview before using the more involved convolution fitter, a live Monte-Carlo convolution can be performed by selecting "Resolution" -> "Convolution..." in the menu bar.

In the "Convolution" dialog four things have to be defined: (1) a crystal, (2) an instrument, (3) a theoretical model for the dynamical structure factor S(q,w), and (4) a scan direction:

(1) and (2): The crystal (together with a scattering plane) and the instrument parameter files are entered in the "Crystal File" and "Resolution File" edit fields, respectively. These files are the same ones that are defined in the Takin main window and in the "Resolution Parameters" dialog.

(3): The S(q,w) model is a user-provided theoretical formula or algorithm for the physical system to be measured. It can be either provided as a Python script or as a plugin. Several simple S(q,w) models are included internally.

(4): The actual scan path is defined in the "Scan Steps" group at the bottom of the dialog. Here, the Q=(hkl) and E coordinates of the start end end position are entered along with the number of subdivisions between these positions ("Steps").

4 Convolution Fitting

The convolution fitter finds the best least-squares fit of one or several free parameters in an S(q,w) model using a Monte-Carlo convolution approach. The S(q,w) model is fitted to one or more given measurement data files.

The fitter is called on the command-line by: "convofit my_fit.job" The input job file "my_fit.job" in this example has the following structure:

```
; convofit sample job file

; input files
input
{
    ; file with the TAS scan
    scan_file       "my_measurement.dat"

    ; temperature column in the scan file
    temp_col        "Ts"

    ; field column in the scan file
    field_col        "Bs"

    ; counter column in the scan file
    counts_col      "ctr1"

    ; monitor counter column in the scan file
    monitor_col     "mon2"

    ; the instrument definition created by
    ; the Takin resolution parameter dialog
    instrument_file "my_instrument.taz"


    ; which S(q,w) model to use?
    sqw_model       "py"

    ; S(q,w) input file. Here, a Python script
    sqw_file        "my_sqw.py"

    ; how is the temperature/field variable named
    ; the S(q,w) model? Here, "g_T" and "g_H" are
    ; global variables in the "my_sqw.py" script.
    sqw_temp_var    "g_T"
    sqw_field_var   "g_H"
    ; fix some variables in the S(q,w) model
    sqw_set_params  "g_my_param = 12.3"
}


; output files
output
{
    ; a simplified copy of the original scan file for easier reading
    ; in an external plot program
    scan_file        "my_scan.dat"

    ; a file describing the fit results
    model_file       "my_model.dat"

    ; logs
    log_file         "my_log.dat"

        ; show a plot at the end of the fit
    plot              1

    ; show plots during fitting (very useful for debugging)
    plot_intermediate 1
}

; includes a fit settings file (see below)
#include "my_settings.job"
```

The fitting steps are described in the include file "my_settings.job" which is given here:

```
; convofit sample fit settings file

; Monte-Carlo settings
montecarlo
{
        ; number of Monte-Carlo neutrons
        neutrons    10000
}

; Resolution algo settings
resolution
{
    ; which algorithm to use?
    algorithm    "eck"    ; "cn", "pop", "eck", or "viol"

    ; include the "resolution volume" prefactor?
    use_r0       1
```

```
	; use optimum vertical/horizontal monochromator/analyser focusing?
	focus_mono_v 1
	focus_mono_h 0
	focus_ana_v  0
	focus_ana_h  1
}


; Fitter settings
fitter
{
	; do a convolution fit or just a plain convolution
	; using the initial values?
	do_fit         1

	; which minimiser to use?
	minimiser    "simplex"    ; "simplex" or "migrad"

	; which Minuit strategy?
	strategy       1          ; 0 (low), 1 (medium) or 2 (high)

	; number of maximum function calls
	max_funccalls 100

	; Minuit's targeted "estimated distance to minimum"
	tolerance      10.

	sigma          1.
}


; which S(q,w) model parameters should be fitted?
; don't remove "scale" and "offs"!
fit_parameters
{
	; the fit parameters: "scale" and "offs" are internal variables,
	; "g_linewidth" is a parameter in the S(q,w) model. In this example,
	; it is a global name in the "my_sqw.py" script.
	params  "scale " \
	        "offs " \
	        "g_linewidth "

	; initial values of the three parameters
	values  "1e9 " \
	        "1e-6 " \
	        "0.02 "

	    ; errors of the three parameters
```

```
    errors   "0.5e9 " \
             "0.5e-6 " \
             "0.01 "

    ; which parameters should be fitted?
    ; here, the third parameter, i.e. g_linewidth, is the only
    ; fit parameter
    fixed    "1 1 0 "
}
```

5 S(Q,w) Models

5.1 Python S(q,w) Models

A S(q,w) Python module has to define the two functions "TakinInit" and "TakinSqw".

"TakinInit" is called after one or several parameters have changed (for example after each minimisation step in the convolution fitter). It can be used to check if e.g. pre-calculated tables, variables, etc. need to be recalculated.

The "TakinSqw" function receives four floating-point parameters h,k,l, and E and returns a single floating-point value S, the dynamical structure factor. The function is called for every Monte-Carlo point.

All global variables that are defined in an S(q,w) Python module are made available as fit parameters for the convolution fitter.

The interface is defined as follows (a full example can be found in the subdirectory "examples/sqw_py"):

```
def TakinInit():
        # reinitialise variables here
        pass

def TakinSqw(h, k, l, E):
        S = 0.
        # calculate S here
        return S
```

5.2 Native S(q,w) Models

Takin can load S(q,w) plugins via native shared libraries (SO or DLL files). A minimal example to build upon is given in the subdirectory "examples/sqw_module".

5.3 Internal S(q,w) Models

Internal S(q,w) models include the following:

Elastic Model

The elastic S(q,w) model consists of a list of elastic peaks which are used to simulate Bragg tails. Each line defines a peak, with the columns of each line being: h, k, l, q width, E width, and S.

Here's a sample input file defining three Bragg peaks:

```
1 0 0     0.002 0.01     0.5
1 1 0     0.002 0.01     1
2 2 0     0.002 0.01     5
```

Tabulated Model

This model loads a table of S(Q,w) points in the four-dimensional (Q,E) space and constructs a kd-tree out of it. As with the "elastic model" each line defines one point. The columns are h, k, l, E, and S.

Here's a (very unrealistic) sample input file defining three points. A realistic input file would include tens of thousands of points to cover a fine and large enough grid in (Q,E) space.

```
1 0 0     0     1
1 0 0   0.5     0.5
1 0 0  −0.5     0.5
```

Simple Phonon Model

With the simple phonon model sinusoidal phonon branches can be defined around a given Bragg peak.

In the following input file we define a longitudinal [110] and two transverse ([001] and [1-10]) branches around the (440) peak:

```
G               = 4,  4, 0
TA1             = 0,  0, 1
TA2             = 1, −1, 0
LA_amp          = 25
LA_freq         = 0.5∗pi
LA_E_HWHM       = 0.1
LA_q_HWHM       = 0.1
LA_S0           = 1
TA1_amp         = 15
TA1_freq        = 0.5∗pi/sqrt(2)
TA1_E_HWHM      = 0.2
TA1_q_HWHM      = 0.2
TA1_S0          = 1
TA2_amp         = 10
TA2_freq        = 0.5∗pi/sqrt(2)
```

```
TA2_E_HWHM         = 0.2
TA2_q_HWHM         = 0.2
TA2_S0             = 1
```

Simple Magnon Model

The simple magnon model creates quadratic (ferromagnetic) or linear (antiferromagnetic) branches spherically around a given Bragg peak.

In the following input file we define magnons with a stiffness of D=20 around the (110) peak:

```
disp               = 0
G                  = 1, 1, 0
D                  = 20
offs               = 0.
E_HWHM             = 0.05
q_HWHM             = 0.05
S0                 = 1
num_points         = 50
```