Mufit Manual

Table of Contents

Introduction

Mufit was developed to fit nuclear and magnetic structures measured by neutron diffraction and polarimetry.

How to install Mufit

- You should have a recent version of Octave >= 3.1.55 (www.octave.org)
- At the Octave prompt type 'pkg install mufit_version_number.tar.gz'. If this does not work you should install the program manually: at the Octave prompt type 'mkoctfile __polmat.cc' and 'siman.cc' to compile these files and set the path to the files in your ~./octaverc
- You should also have a recent version of Gnuplot (>= 4.2)
- You might install zenity from Octaveforge, but it is not mandatory
- You should install a vrml-viewer to view the magnetic structure. The path to your vrml-viewer has to be set in the function 'vrml_browse.m'

Using Mufit

- Mufit needs 3 input files. Examples can be found in the subdirectory "examples".
- When the input files are ready, and the c++ files compiled just type 'Mufit' at the Octave prompt and choose one of the available options.

Distribution

Warranty

Bugs

Please report bugs to bertrand.roessli_at_psi.ch

Nuclear structures

The extinction is calculated according to the Zachariasen formula. The structure factor

$$F(hkl) = \sum_{\vec{a},d} b_d \exp(i \cdot \vec{a}_d \cdot \vec{hkl}) n_d$$

is corrected for extinction through

$$F_{corrected} = F(1 + 0.001x)^{-0.25}$$

with

$$x = (scalefactor)|F|^2\lambda^3/\sin(2\theta).$$

The isotropic thermal factor is given by

$$T(\vec{h}) = \exp(-2\pi^2 < (\vec{h} \cdot \vec{u})^2 >)$$

and

$$\vec{h} = \vec{Q}/(2\pi).$$

For isoptropic dispacements

$$u_x = u_y = u_z = u$$

such as

$$T(\vec{h}) = \exp(-2\pi^2 h^2 u^2).$$

As

$$h^2 = (2\sin(\theta/\lambda))^2,$$

the isotropic Debye-Waller factor is

$$T(|\vec{h}|) = \exp(-8\pi^2 u^2 (\sin(\theta)/\lambda)^2).$$

The program fits the Debye-Waller factor as $\exp(-B(Q/(2\pi))^2)$ which is related to the isotropic diplacement u by $B = 2\pi^2 u^2$.

Anisotropic thermal temperature factors are fitted with the expression:
$T(\vec{h}) = \exp(-2\pi^2[U_{11}h^2 + U_{22}k^2 + U_{33}l^2 + 2U_{12}hk + 2U_{13}hl + 2U_{23}kl])$, the anisotropic displacement parameters U have the units $(length)^2$.

Input file

Example of a 'nuc.inp' input file used to fit chemical structures. As a rule the input file contains a comment line (that starts with !) followed by a parameter value and a parameter label. Note that the parameter value is used to perform calculations only, while the starting value parameter when used for fitting is given in the line beginning with "!N1 N2 ...".

1. Neutron wavelength
   !wavelength
   1.526
2. Extinction parameter calculated from Zaccharian formula
   !extinction
   0.00
   abs(N2) 0.000
3. Scale factor
   !scale
   58.24
   N1 0.000
4. lattice parameters in A and angles in deg.
   !lattice parameters and angles
   7.220708 8.461032 5.660867 90.000000 90.000000 90.000000
5. number of atoms
   ! # atoms
   7
6. atomic label, scattering length, position parameters (x,y,z), isotropic B-factor and occupation number
   ER 0.779 0.13658 0.17096 0.00000 0.367 0.50000
7. parameter label (e.g. N3, fixed if error = 0, see below) or value if not fitted
   N3 N4 0.00000 N16 0.50000
8. number of symmetry operations
   !# symmetry operations in space group and symmetry elements
   8
9. example of symmetry operations. Note that Mufit will not recognize lower case labels and that translation comes first. E.g. '-x+1/2' will lead to an error.
   X Y Z
   1/2-X 1/2+Y Z
   1/2+X 1/2-Y Z
10. number of translations in the space group. Optionally the translations can be given here, instead of the line above.
    !# number of translations and translations in space group
    1
11. translation operation
    0 0 0
12.

    starting parameter value. Must have 10 parameters/line and 3 such lines –> 30 parameters in total. If necessary, the maximum number of parameters can be increased by

editing 'nucfafit2.m'
!N1 N2 N3 N4 ...
906. 10000 0.13758 0.17149 ...

13.
parameter step, 0 if fixed.
!errors dN1 to dN10
1 1 1. 1. ...

14. choice of fit
! fit algorithm: levenberg-marquard (0), simplex (1), simulated annealing (2)
1

15. fit options: exit when change in cost function < stol or number of iterations (niter) exceeded. see 'leasqr.m' for details
!levenberg-marquard options: stol niter verbose (0/1)
0.001 1000 1

16. fit options: as above, plus simplex-type (regular (0) or right-angled (1) simplex). see 'nmsmax.m' for details
!simplex options: stol niter chisq_min simplex verbose (0/1)
1e-12 5000 5 0 1

17. fit options as described in 'samin.cc'
!simulated anneling options
* nt - integer: # of iterations between temperature reductions
* ns - integer: # of iterations between bounds adjustments
* rt - (0 < rt <1): temperature reduction factor
* maxevals - integer: limit on function evaluations
* neps - integer: number of values final result is compared to
* functol - (> 0): the required tolerance level for function value comparisons
* paramtol - (> 0): the required tolerance level for parameters
* verbosity - scalar: 0, 1, or 2.
5 5 0.8 100 5 0.1 0.1 2

18. 0: intensitites are fitted; 1: structure factors^2 are fitted
!flag intensities/structure factors^2
1
Note: If neutron intensitites are fitted the structure factor $|F|^2$ is multiplied by the Lorenz factor $L = 1/\sin(2\theta)$.

19. label (not used, but necessary), H,K,L, I, standard deviation
! h k l I(obs) sqrt(I)
N 4.000000 0.000000 0.000000 6751.580000 95.110000

Output file

When the fit is completed the file 'crysfit.nuc' is created that contains the result of the calculations

Magnetic structures

Input files

Two input files are necessary to fit magnetic intensities:

- 'crys.inp' that is described below
- 'mag.inp' that is described in the polarimetry section

An input file should be present in the ./examples directory.

Description of a 'crys.inp' input file used to fit magnetic structures.

1. wavelength, extinction and scale factor. These parameters cannot be fitted.
   ! lambda, extinction, scale factor
   1.18 0 51
2. flag for fit of integrated intensities (0) or structure factors^2 (1)
   ! Int (0) / fsqr (1)
   1
3. hkl-list, observed intensity, standard deviation
   ! h k l I(obs) sqrt(I)
   M +0.500000 +0.000000 +0.250000 9.620000 0.237000

Output files

At the end of the fit the following files will be created.

1. mag.new: parameter file
2. mag.str: magnetic model
3. mag.plot: plot file for DrawXtl

The output will be improved in future versions

Polarimetry

Mufit can calculate and fit polarimetry matrices. The options (4) of the menu calculates the polarisation matrices with the Blume's equations. The option (5) calculates the polarisation matrices by evaluating the polarized neutron cross-section, which allows to take into account for the presence of magnetic domains and to correct for the finite polarization of the beam. The option (8) allows to fit polarimetry data using one of the 3 algorithms described in the manual. The xyz-components of the incident and scattered polarisations are defined according to Blume's convention with x along the scattering vector Q, z up and $y = cross(z, x)$.

Magnetic Model

The magnetisation density is written as

$$\vec{M}(\vec{r}+\vec{l}) = \vec{R}M_p(\vec{r})\cos(\vec{\tau}\cdot\vec{l}+\phi_r) + \vec{I}M_q(\vec{r})\sin(\vec{\tau}\cdot\vec{l}+\phi_r)$$

with R and I two orthogonal unit vectors. Polar coordinates can be used and are defined in a cartesian coordinate system (with x along a) with $M_x = M\cdot\sin(\theta)\cdot\cos(\phi)$; $M_y = M\cdot\sin(\theta)\cdot\sin(\phi)$; $M_z = M\cdot\cos(\theta)$ and ϕ the angle in the (x, y)-plane and θ is the angle from the z-axis. The phase is given in units of 2π. Alternatively the magnetic density can be described directly using (x,y,z) components.

Description of the input file 'mag.inp' that is required to fit the single crystal data and the polarimetry matrices:

1.
   !lattice
   7.2208 8.4610 5.6609 90 90 90
2.
   !# atoms
   12
3.
   !AT x y z B occ
   Mn41 0.00000 0.50000 0.74300 0.370 1
4.
   ! input switch 0: components along axis RMx,RMy,RMz,Imx,Imy,Imz,phase; 1: polar coordinates Rm,Rphi,Rtheta,Im,Iphi,Itheta,phase
   1
5.
   ! F(Q) Rm Rphi Rtheta Im Iphi Itheta phase (units of 2*pi)
   12
6.
   Mn41 MF_Mn4 3.270 -1.8 90.000 0.44 180.000 +0.000 0.1537
   C6 C1 90.000 C8 180.000 +0.000 0.125+C12

7.
   !# symmetry operations in space group and symmetry elements
   1
8.
   X Y Z RMx RMy RMz IMx IMy IMz 0.0
9.
   !# of translations and translations
   0
10.
   ! propagation vectors
   1
   k01 0.5 0 0.25
11.
   !#spin domains, populations and parameters
   2
12.
   !population and rotation matrices for domains
   0.10 1 0 0 0 1 0 0 0 1 T
   0.10 -1 0 0 0 -1 0 0 0 -1 T
13.
   !c1 c2 c3 c4 c5 c6 c7 c8 c9 c10
   +5.70 +14.41 +157.16 +37.41 +1.22 +2.71 +4.13 +0.61 +1.68 +1.76
14.
   !steps dC1 to DC30
   +0.50 +0.50 +0.50 +0.50 +0.50 +0.50 +0.50 +0.50 +0.50 +0.50
15.
   !D1 D2 D3 D4 D5 D6 D7 D8 D9 D10
   +0.50 +0.50 +0.00 +0.00 +0.00 +0.00 +0.00 +0.00 +0.00 +0.00
16.
   !steps dD1 to DD30
   +0.5 +0.5 +0.00 +0.00 +0.00 +0.00 +0.00 +0.00 +0.00 +0.00
17.
   ...
18.
   ! flag for calculation k0 + -k0 (1); flag for pure magnetic reflection (0)
   0 0

Polarimetry matrices

Description of the part of the input file 'mag.inp' (continuation) that is special to neutron polarimetry.

1. Efficiency of the polarizers. The beam is fully polarized when P_i = 1 and depolarized for P_i = 0 (i=1,2)
   ! polarizer/analyzer efficiencies 0 <= P_1, P_2 <= 1
   1 1

2. input of the observed polarisation matrices
3. example
   ! hkl, vector in scattering plane, P_i, P_f, error
   +0.5 +3 +0.25 0 -1 0 1 0 0 -0.9 +0.0 -0.1 +0.02 +0.02 +0.02
   description of the input line:
   - hkl : +0.5 +3.0 +0.25
   - Mufit needs a second vector (A_y) in the scattering plane to define the sample orientation: 0 -1 0
   - the incident polarization (Px along Q, Pz perpendicular to the scattering plane, and Py in the scattering plane such as the coordinate system is right-handed): 1 0 0
     Care as to be taken that the zone axis is defined properly as it is calculated from $cross([HKL], A_y)$.
   - scattered polarization along x, y and z: -0.99 +0.06 -0.10
   - standard deviation: +0.01 +0.02 +0.02

Fit routines

The fit functions have been adapted from the 'optim' package in Octave forge. Three fit algorithms are implemented: Levenberg-Marquard, Simplex and Simulated Annealing. They can be used to fit nuclear, magnetic and polarimetry data. Note that at moment it is not possible to fit simultaneously nuclear and magnetic intensities. If the polarimetry matrices contain nuclear-magnetic intereference terms, it is not possible to fit the nuclear parameters.

Mufit calculates χ^2 and R_{Bragg} defined as

$$\chi^2 = \frac{\sum_{i=1}^{N}((y_i - f_i) * \sigma_1)^2}{n_p}$$

$$R_{Bragg} = 100 * \frac{\sum_{i=1}^{N}(|y_i - f_i|)}{\sum_{i=1}^{N}|y_i|}$$

For the polarisation matrices only χ^2 is calculated

Algorithm

1. The choice between the different fit algorithms is defined by a switch with values 0/1 or 2 in the input files 'mag.inp' and 'nuc.inp'
2. at the lines:
   ! fit algorithm: levenberg-marquard (0), simplex (1), simulated annealing (2)
   1

Levenberg-Marquard

1. fit switch (0):
   Levenberg-Marquard with exit options: change in cost function, maximum number of iteration; verbose =1 if fit progress shown, = 0 otherwise. The algorithm uses numerical derivatives calculated in the function dfdp.m (forward method). See 'leasrq.m' for details.
2. input lines:
   !levenberg-marquard options: stol niter verbose (0/1)
   0.1 100 1

Nelder-Mead

1. fit switch (1)
   direct search method based on the Nelder-Mead algorithm with exit options stol, niter and chisq_min. The initial simplex size depends on the number of parameters. It is then advised that niter depends on the number of parameters (˜ 1000 cycles for 20 parameters). The additional parameter 'simplex' is a switch for use of either the regular simplex (0) or right-angled simplex (1). If verbose is set to 1 then the fit progress is shown. See 'nmsmax.m' for details.

2. input lines:
   !simplex options: stol niter chisq_min simplex verbose (0/1)
   0.001 3000 5 0 1

Simulated Annealing

1. fit switch (2):
   algorithm based on simulated annealing. The starting temperature is obtained automatically. See 'samin.cc' for details.
2. input lines:
   !simulated anneling options
   * nt - integer: # of iterations between temperature reductions
   * ns - integer: # of iterations between bounds adjustments
   * rt - (0 < rt <1): temperature reduction factor
   * maxevals - integer: limit on function evaluations
   * neps - integer: number of values final result is compared to
   * functol - (> 0): the required tolerance level for function value comparisons
   * paramtol - (> 0): the required tolerance level for parameters
   * verbosity - scalar: 0, 1, or 2.
   20 5 0.95 100000 5 0.1 0.1 2

Annexe

List of functions with a short description:

chi.m

`chi (`*p*`,`*varargin*`)` [Function File]
`[`*chisq*`,`*f*`,`*p*`,`*R_Bragg*`]= chi (`*p*,*dp*,*varargin*`)` [Function File]

Calculate ChiSquare and R_Bragg; *p* should be a vector containing the parameters; *dp*: vector containg the parameter steps; *varargin* is a cell array containing additional variables necessary to calculate the function defined in *varargin(1)*; the function returns ChiSquare, the function value, the parameters and R_Bragg

crys_inp.m

`crys_inp (`*as*`,`*bs*`,`*cs*`,`*aa*`,`*bb*`,`*cc*`)` [Function File]
`[`*genpar*`,`*singlecrys*`]= crys_inp (`*as*,*bs*,*cs*,*aa*,*bb*,*cc*`)` [Function File]

The function imports wavelength, extinction/scale parameters and hkl's/intensities/standard deviations from file crys.inp and returns information to mufit

crys_int_fit2.m

`crys_int_fit2 (`*p*`,`*varargin*`)` [Function File]
`[`*I_crys*`,`*param*`]= crys_int_fit2 (`*p*,*varargin*`)` [Function File]

Calculate single crystal intensitites or structure factors using model parameters stored in array p and return intensity list and parameters

dfdp.m

`dfdp (`*f*`,`*p*`,`*dp*`,`*func*`,`*varargin*`)` [Function File]
`[`*prt*`,`*param*`] = ` *dfdp* `(`*f*,*p*,*dp*,*func*,*varargin*`)` [Function File]

Calculate numerical Jacobian for use in leasqr (Leveberg-Marquard)
Use forward difference
The step was taken from __bfgsmin.cc
Return the Jacobian and the parameter list

extinc.m

`extinc (`*fsqr*`,`*lambda*`,`*c*`,`*hkl_AA*`)` [Function File]
`[`*zacor*`]= extinc (`*fsqr*,*lambda*,*c*,*hkl_AA*`)` [Function File]

Calculate extinction based on Zachariasen formula like Shelx

input:
fsqr: structure factor^2
lambda: wavelength

c: scale factor
hkl_AA: hkl in A

output:
zacor: extinction correction

formfac.m

`formfac (Q,at_n)` [Function File]
`[j_0= formfac (Q,at_n)` [Function File]

Calculate magnetic form factor using the parameters tabulated in the International Tables by P.J. Brown
The function does contains the tabulated parameters of the magnetic form factor for only few ions at moment. Formfac.m should therefore be edited and adapted. The list of tabulated form factors will be will be completed with time.

input:
Q: scattering vector
at_n: atom label from file mag.inp

output:
j_0: magnetic form factor at *Q*

hklprep.m

`hklprep` [Function File]
`(as,bs,cs,aa,bb,cc,lambda,singlecrys,domains_cell,moment_cell,p)`
`[QQ,fstar,forfmactor,popul,llor] =` [Function File]
hklprep (*as,bs,cs,aa,bb,cc,lambda,singlecrys,domains_cell,moment_cell,propvec_cell_rlu*)

Utility function that creates various lists like HKL in rlu or A to take into account the presence of the magnetic domains
Called by mufit.m

lat2cart.m

`lat2cart (as,bs,cs,aa,aa,bb,cc)` [Function File]
[Function File]
`[Mcryst2cart,Mrec2cart] =` lat2cart `(as,bs,cs,aa,aa,bb,cc)`

Calculate transformation matrix to caretesian coordinates

input:
1.- lattice constants and angles

output:
1.- *Mcryst2cart*: transformation matrix from crystal to cartesian coordinates
2.- *Mrec2cart*: transformation matrix of reciprocal lattice to cartesian coordinates

magfac2.m

`moment_fit (`*`spins_cell,fstar,Q,flag_k0,ffq`*`)` [Function File]
`[`*`mag_inter_vec,fsq,mag_vec_cart`*`] =` *moment_fit* [Function File]
(*spins_cell*,*fstar*,*Q*,*flag_k0*,*ffq*)

Calculate magnetic interaction vector and structure factor
Function is vectorized
input: *1.- *spins_cell*: magnetic model
2.- *fstar*: list flag +/- magnetic satellite
2.- *Q*: list of [h,k,l]
3.- *flag_k0*: switch k_0 = -k_0 ?

output:
1.- *mag_inter_vec*: magnetic interaction vector
2.- *fsq*: structure factor^2
3.- *mag_vec_cart*]: component of structure factor

magfac.m

`magfac` [Function File]
`(`*`spins_cell,Q,hhkl_rlu,propvec_cell,flag_k0,dtype`*`)`
`[`*`mag_inter_vec,fsq,mag_vec_cart`*`] =` *magfac* [Function File]
(*spins_cell*,*Q*,*hhkl_rlu*,*propvec_cell*,*flag_k0*,*dtype*)

Calculate magnetic interaction vector and structure factor

input:
1.- *spins_cell*: magnetic model
2.- *Q*: norm of scattering vector
3.- *hhkl_rlu*: [h,k,l]
4.- *propvec_cell*: propagation vector
5.- *flag_k0*: switch k_0 = -k_0 ?
6.- *dtype*: not used

output:
1.- *mag_inter_vec*: magnetic interaction vector
2.- *fsq*: structure factor^2
3.- *mag_vec_cart*]: component of structure factor

moment_fit.m

`moment_fit` [Function File]
  (lat_cell,ion_cell,spgsym_cell,spgtrans_cell,propvec_cell_AA,moment_cell,pa

[S_cart_sys_cell]`= moment_fit` [Function File]
  (lat_cell,ion_cell,spgsym_cell,spgtrans_cell,propvec_cell_AA,moment_cell,param)

Calculate the magnetic moment configuration from the paramaeter of the magnetic model in mag.inp
Used to fit magnetic intensities and polarimetry matrices

input:
1.- the lattice parameters (*lat_cell*)
2.- the atomic positions (*ion_cell*)
3.- the magnetic symmetry operations (*spgsym_cell*)
4.- the list of translation (*spgtrans_cell*)
5.- the propagation vector in rlu and AA^1 (*propvec_cell_AA*)
6.- the magnetic model (*moment_cell*)
7.- the parameter list

output:
the magnetic moment directions/size calculated from the *param* list

moment.m

`moment` [Function File]
  (lat_cell,ion_cell,moment_cell,propvec_cell_rlu,propvec_cell_AA,spgsym_cell

[M_cell,S_cart_sys_cell] `=` *moment* [Function File]
  (lat_cell,ion_cell,moment_cell,propvec_cell_rlu,propvec_cell_AA,spgsym_cell,spg

Calculate the magnetic moment configuration from the magnetic model in mag.inp

input:
1.- the lattice parameters (*lat_cell*)
2.- the atomic positions (*ion_cell*)
3.- the magnetic model (*moment_cell*)
4.- the propagation vector in rlu and AA^1 (*propvec_cell_rlu*,*propvec_cell_AA*)
5.- the magnetic symmetry operations (*spgsym_cell*)
6.- the list of translation (*spgtrans_cell*)

output:
1.- the spins coordinates in the unit cell used to calculate the magnetic structure factor (*M_cell*)
2.- the magnetic moment directions/size in different cells according to the list of translations (*S_cart_sys_cell*)

mt_draw_mag.m

`mt_draw_mag (`*varargin*`)` [Function File]
`[]= mt_draw_mag (`*varargin*`)` [Function File]

Draw the magnetic structure

input:
1.- *varargin(1)* transformation matrix to cartesian axis
2.- *varargin(2)* lattice constants and angles
3.- *varargin(3)* not used
4.- *varargin(4)* atomic positions and magnetic moments
5.- *varargin(5)* switch polar/cartesian coordinates

mufit.m

Mufit calculates or fits neutron nuclear/magnetic intensities or neutron polarisation matrices. The program has been tested with Octave-3.2.0 and Gnuplot 4.2

- The 'Mufit' function does not take any argument but uses 3 input files: 1- 'nuc.inp'; 2- 'crys.inp'; 3- 'mag.inp':
- The nuclear model is given in the file 'nuc.inp'; The magnetic model is given in the file 'mag.inp'; To fit polarimetry data, the file 'nuc.inp' is required in case of nuclear-magnetic interaction terms.
- Lists of HKL/Intensities/Standard deviations are given in the files 'nuc.inp' and 'crys.inp'. The polarisation matrices are given in the file 'mag.inp'.
- To calculate the neutron polarisation matrices it is necessary to give a second vector in the scattering plane to define the crystal orientation. (see mag.inp for an example) Note that the incident polarisation is defined according to Blume, i.e. P_x along the scattering vector, P_z perpendicular to the scattering plane, and P_y=P_z \cross P_x.
- The coefficients of the magnetic formfactor are given in the function 'formfac.m'. You will probably need to add the coefficients for your case, as currently only a few examples are implemented.
- The data can be fitted with the Levenberg-Marquard method or the Nelder-Mead algorithm. Please note that the size of the initial Simplex depends on the number of parameters. Therefore it is recommended that if the Simplex method is used the number of cycles increases with increasing number of parameters (~1000 cycles for 20 parameters)
- Simulated Annealing has also been implemented. The algorithm is described in the function samin.cc.
- The fit functions have been adapted from the 'optim-1.0.6' package at www.octaveforge.org
- The fit options should be given in the files 'nuc.inp' and 'mag.inp'. See leasrq.m, nmsmax.m and samin.cc for details. Documentation can be obtained by typing `"help function_name"` at the Octave prompt.
- To install mufit you will need to compile __polmat.cc and samin.cc. At the Octave prompt, type mkoctfile ***.cc

- If you use the standalone application, just run the mufit executable. The scripts, cpp functions and input files should be in the same directory than the executable. Please note that Octave needs to be installed even if you prefer to use the executable.
- Please report bugs to bertrand.roessli_at_psi.ch

ncs.m

`ncs (`*`H,K,L,A1,P0_x,P0_y,P0_z,mag_inter_vec,flag_nmi,det_r`*`)` [Function File]

`[`*`P_f,cs_n`*`]= ncs (`*`H,K,L,A1,P0_x,P0_y,P0_z,mag_inter_vec,flag_nmi,det_r`*`)` [Function File]

Calculate neutron cross section and polarisation matrices with the Blume's equations

input:
1.- [H,K,L]
2.- *A1* second vector in scattering plane (A^-1, cartesian coordinates)
3.- incident polarisation [P0_x;P0_y;P0_z]
4.- *mag_inter_vec*: the magnetic interaction vector in crystal coordinate system
5.- *flag_nmi*:flag for mixed magnetic/nuclear reflections
6.- *det_r*: determinant of the domain operation

output:
1.- *P_F*: the scattered polarisation
2.- *ncs*: the neutron cross-section

nucfacfit2.m

`nucfacfit2 (`*`p,varargin`*`)` [Function File]

`[`*`strufac2,param`*`] =` *nucfacfit2* `(`*`p,varargin`*`)` [Function File]

Fit the nuclear structure factors or intensities

input:
1.- *p*: list of parameters *p*
2.- *varargin{1}*: hkl-list (rlu)
3.- *varargin{2}*: hkl-list (AA)
4.- *varargin{3}*: wavelength, extinction
5.- *varargin{4}*: lattice parameters
6.- *varargin{5}*: intensities
7.- *varargin{6}*: atomic parameters
8.- *varargin{7}*: space group symmetry elemetns
9.- *varargin{8}*: translations of the space group (can be put in *varargin{7}*)

output:

strufac2: nuclear structure factor^2
parameters: parameters

nucfac.m

`nucfac (hkl_crys_rlu,hkl_crys_AA)` [Function File]
`[strufac2] = nucfac(hkl_crys_rlu,hkl_crys_AA)` [Function File]

Calculate the nuclear structure factors to obtain the nuclear-magnetic interference term in the polarisation matrices; The nuclear model is defined in the file 'nuc.inp'

input:
1.- *varargin{1}*: hkl-list (rlu)
2.- *varargin{2}*: hkl-list (AA)

output:
strufac2: nuclear structure factor^2

onedomain.m

`onedomain` [Function File]
`(P_1,P_2,hhkl,Q_rlu,AA1,flag_k0,flag_nmi,spins_cell,dtype,pi_direction,propv`
`[Ip_ij,Im_ij,polar] = onedomain` [Function File]
`(P_1,P_2,hhkl,Q_rlu,AA1,flag_k0,flag_nmi,spins_cell,dtype,pi_direction,propvec_ce`

Calculate scattered polarization of a single magnetic domain

input:
1.- *P_1*, *P_2*: polarizer efficiencies
2.- *hhkl*: hkl
3.- *Q_rlu*: scattering vector
4.- *AA1*: second vector in scattering plane
5.- *flag_k0*: flag k_0 = -k_0
6.- *flag_nmi*: flag nuclear-magnetic interaction vector
7.- *spins_cell*: magnetic model
8.- *dtype*: not used
9.- *pi_direction*: switch for incident polarization direction (+z/-z)
10.- *propvec_cell*: propagation vector
11.- *det_r*: determinant of domain creation matrix

output:
1.- *Ip_ij*: polarized neutron intensity (P_i +z)
2.- *Im_ij*: polarized neutron intensity (P_i -z)
3.- *polar*: scattered polarization matrix

polar_fit2.m

`polar_fit2 (p,varargin)` [Function File]
`[polar_final_row,param] = polar_fit2 (p,varargin)` [Function File]

Used to fit the polarization matrices
Calculate polarized cross-section and calls __polmat.cc that actually performs the calculation of the polarization matrices

input:
1.- *p* should be a vector containing the parameters
2.- *varargin* is a cell array containing the other variables

output:
1.- *polar_final_row*: polarisation matrices
2.- *param*: parameter list

pol_sd.m

`pol_sd (hhkl,AA1,Q_rlu,flag_k0,flag_nmi,spins_cell,propvec_cell_rlu,dtype,det_r)` [Function File]
`[poln_cs] = pol_sd (hhkl,AA1,Q_rlu,flag_k0,flag_nmi,spins_cell,propvec_cell_rlu,dtype)` [Function File]

Calculate neutron intensities in one magnetic domain as 2x2 matrix

input:
1.-: *hhkl*: list of hhkl
2.-: *AA1*: second vector to define scattering plane
3.-: *Q_rlu*: Q vector
4.-: *flag_k0*: k_0=-k_0?
5.-: *flag_nmi* : flag for pure magnetic/mixed nuclear-magnetic reflections
6.-: *spins_cell*: spin positions and directions
7.-: *propvec_cell_rlu*: propagation vector
8.-: *dtype*: not used
9.-: *det_r*: determinant of domain operation

output:
1.-: *poln_cs*: neutron cross-section

readmu.m

`readmu (varargin)` [Function File]
`[vargout] = readmu (varargin)` [Function File]

Read parameters from file 'mag.inp' for magnetic and polarimetry calculations/fit

output: *vargout*
1.-: *as*,*bs*,{...}, *cc*: cell parameters
2.-: *parameters*: parameters
3.-: *dparameters*: parameter steps
4.-: *flag_k0*: flag k_0=-k_0
5.-: *flag_nmi*: flag nuclear-magnetic interaction vector
6.-: *k0*: propagation vector (rlu)
7.-: *hkl_AA*: hkl list in AA^-1
8.-: *hkl_rlu*: hkl-lit in rlu
9.-: *A1*: second vector in scattering plane
10.-: *Pi*: incident polarization
11.-: *Pf0*: scattered polarization
12.-: *dPfo*: standard deviation of the scattered polarization
13.-: *nr*: number of spins
14.-: *P_1*, *P_2*: polarizer efficiencies
15.-: *moment_type*: switch polar/ cartesian coordinate
16.-: *mfit_type*: switch to select fit algorithm (0/1/2)
17.-: *simplexnuc_options*: simplex options
18.-: *lmfitnuc_options*: levenberg-marquard options
19.-: *nsim_ann_par*: simulated annealing options
20.-: *lat_cell*: lattice parameters
21.-: *ion_cell*: atomic positions
22.-: *moment_cell*: magnetic model
23.-: *propvec_cell_rlu*: propagation vector in rlu
24.-: *propvec_cell_AA*: propagation vector in AA^-1
25.-: *spgsym_cell*: symmetry operations
26.-: *spgtrans_cell*: translations (used to propagate spins)
27.-: *domains_cell*: symmetry operations for magnetic domains

readnuc.m

`readnuc ()` [Function File]
`[...] =` *readnuc* () [Function File]

Read parameters from file 'nuc.inp' fir nuclear fit (see ./examples/nuc.inp)

output:
1.-: *genpar* :wavelength, extinction, scale factor
2.-: *as*,*bs*,{...}, *cc*: cell parameters
3.-: *ion_cell*: atomic paramters
4.-: *spgsgsym_cell*: space group symmetry elements
5.-: *spgtrans_cell*: space group translations
6.-: *singlecrys*: switch structure factor/intensities and list of hkl with intensities and standard deviations
7.-: *nfit_type*: switch to select fit algorithm (0/1/2)
8.-: *simplexnuc_options*: simplex options

9.-: *lmfitnuc_options*: levenberg-marquard options
10.-: *nsim_ann_par*: simulated annealing options

sigma_tot.m

`sigma_tot (`*`f1`*`,`*`f2`*`,`*`f3`*`,`*`f4`*`,`*`poln_cs`*`,`*`P_1`*`,`*`P_2`*`)` [Function File]
`[`*`Intensity`*`] = ` *sigma_tot* `(`*`f1`*`,`*`f2`*`,`*`f3`*`,`*`f4`*`,`*`poln_cs`*`,`*`P_1`*`,`*`P_2`*`)` [Function File]

Calculate polarised neutron intensity taking into account the efficiency of the polarizers

input: *f1->f4*: incident and scattered neutron spin state *poln_cs*: neutron interaction potential *P_1* and *P_1*: polariser efficience 0<=P<=1

output: *Intensity*: polarised neutron intensity

Plot

The option 9 of the Mufit menu will draw the magnetic structure. The plotting routines are based on the vrml-1.0.10 package from Octave forge. It is necessary to edit the file vrml_browse.m and to define the path to the vrml-viewer installed in your system by setting the variable vrml_b_name, e.g. vrml_b_name = "/home/broessli/view3dscene/view3dscene" ;

The color and size of the atoms can be set in the file mag.inp. At moment it is not possible to modify the length and colors of the arrows. If you want to do so you should modify the default settings in the function 'mt_draw_mag.m' and redefine the variables len, alen, dc, dr and col_arrow.

A file 'magstru.wrl' is created that allows to view the magnetic structure with a vrml-browser.

The following functions are used from the vrml-1.0.10 package

vrml_arrow
vrml_browse
vrml_ellipsoid
vrml_faces
vrml_group
vrml_kill
vrml_lines
vrml_material
vrml_surf
vrml_text
vrml_thick_surf
vrml_transfo
save_vrml

To get help about these functions type 'help ***' at the mufit prompt

Table of Contents