# `ESpinS` package: User Guide

Nafise Rezaei, Mojtaba Alaei

August 8, 2021

# Chapter 1

# ESpinS

`ESpinS`, Esfahan Spin Simulation package, is a classical Monte-Carlo simulation package for calculation of thermodynamic properties of magnetic material. It is written in Fortran90. `ESpinS` is a free software, under the GNU General Public License.

### Citation

**Credits** `ESpinS` was written by Nafise Rezaei (main developer) and Mojtaba Alaei. Some subroutines were taken from `wannier90` package and were modified according to especial purposes.

### Notes

- `ESpinS` is running in both single and MPI parallel mode.
- A Monte-Carlo simulation is done in three steps in `ESpinS`, two for initialization and one for Monte-Carlo simulation. Each step can be performed independently. Indeed, initializations are steps to ease the generating of the input file for the Monte-Carlo simulation.
- The input file is not sensitive to the uppercase and lowercase letters.
- Everything after ! or # is considered as a comment and will not be read.
- Logical keywords can be specified as: `T` or `true` or `.true.`.
- Ordering of keywords is not important.
- Each input file keyword in each step can be used in other steps without encountering any error. In fact, the program first read the keyword and if it is not relevant to the present step, the program just ignores it.
- The spin model Hamiltonian is as follows:
$$\mathcal{H} = \underbrace{-\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j}_{\text{exchange term}} + \underbrace{\frac{1}{2}\sum_{i,j} B_{ij}(\mathbf{S}_i \cdot \mathbf{S}_j)^2}_{\text{Bi-quadratic term}} + \underbrace{\frac{1}{2}\sum_{i,j} \mathbf{D}_{ij} \cdot (\mathbf{S}_i \times \mathbf{S}_j)}_{\text{Dzyaloshinskii-Moriya term}} + \underbrace{\sum_i \Delta_i \left(\hat{z}_i \cdot \mathbf{S}_i\right)^2}_{\text{single-ion term}} + \underbrace{\sum_i \mathbf{B} \cdot \mathbf{S}_i}_{\text{magnetic field}}$$
The exchange term is mandatory while the rest of them are optional.
- Keywords in the input files can be specified as follows:
```
tems_num   :   1
tems_num   =   1
tems_num       1
```
- To specify some set of parameters such as atoms positions, block concept is used. The begin and end of a block are defined by `begin name_of_block` and `end name_of_block`, respectively.

# Chapter 2

# Installation

As mentioned above, the code is written in Fortran90 language. The code can be compiled using either Intel Fortran Compiler (IFORT) or GNU Fortran compiler (GFortran) [1] . The code can be compiled by using the make tool. To set the compiler and compiler options (flags), there is a file (`make.sys`) in the code main directiory. User can change the default compiler and compiler options inside `make.sys`.

Example of `make.sys` file for IFORT compiler:

```
#=============
# For Linux
#=============
F90 = ifort
COMMS=mpi
MPIF90=mpifort
FCOPTS=-O2
LDOPTS=-O2
#=========================================================
# Intel mkl libraries. Set LIBPATH if not in default path
#=========================================================

LIBDIR = /opt/intel/mkl/lib/intel64
LIBS   =  -L$(LIBDIR) -lmkl_core -lmkl_intel_lp64 -lmkl_sequential -lpthread
```

After setting, by typing `make` command in the terminal, the code will be compiled. To make life easy, there is a directory (`config`) which contains make.sys templates for IFORT and GFortran compilers.

[1] Essentially the code can be compiled by any Fortran90 compilers but the developers checked the code only using IFORT and GFortran.

# Chapter 3

# Initialization

## 3.1 First initialization step: inp1 step

| | |
|---|---|
| input file | seedname.inp1.mcin |
| Output files | seedname.inp2.mcin, seedname.neigh, seedname.mcout, seedname.xsf |
| run command | mc.x -inp1 seedname |

An input file with the file extension of .inp1.mcin is needed for this step. This file contains information about the unit cell, magnetic atoms positions, and shell number for Hamiltonian terms. In the utility directory, there is a inpfile.tem file which is a bash file and print a sample file of seedname.inp1.mcin. Just write this command at the terminal:
./inpfile.tem
The output is:

```
Begin  Unit_Cell_Cart
 !Bohr
 A_1x        A_1y       A_1z
 A_2x        A_2y       A_2z
 A_3x        A_3y       A_3z
End Unit_Cell_Cart

Begin  Atoms_Frac
 atom_symbol  atom_pos_x  atom_pos_y  atom_pos_z  magnetic_moment
End  Atoms_Frac

Shells_jij      = 1
!! Ham_bij         = T
!! Shells_bij      = 1
!! Ham_dij         = T
!! Shells_dij      = 1

!! Length_unit     = Bohr
!! Parameter_unit  = Ryd
!! Coordinate      = Cart
```

The unit cell vectors, atoms positions, corresponding magnetic moment for each atom and shell number for Hamiltonian should be written in this file.

### 3.1.1 Keywords of seedname.inp1.mcin step:

In the following, the keywords of seednam.inp1.mcin are described:

- Unit Cell information

  A unit cell is identified by a block which its begin and end are specified by begin unit_cell_cart and end unit_cell_cart, respectively. In this block, three vectors of the unit cell are specified in Cartesian coordinate:

  begin unit_cell_cart
  [units]
  $A_{1x}$ $A_{1y}$ $A_{1z}$
  $A_{2x}$ $A_{2y}$ $A_{2z}$
  $A_{3x}$ $A_{3y}$ $A_{3z}$
  end unit_cell_cart

`units` specifies the length unit for the components of cell vectors. The options for `units` are:

- `ang` (default)
- `bohr`

Writing `units` is optional. If it is not present, the unit is the default value *i.e.* Ang.

- Atomic Positions

  Position of atoms can be defined in two kinds of coordinate systems:

| 1) Fractional coordinates | 2) Cartesian coordinates |
|---|---|
| `begin atoms_frac` | `begin atoms_cart` |
| | `[units]` |
| $I_1 \quad f_{1,\mathtt{A}_1} \quad f_{1,\mathtt{A}_2} \quad f_{1,\mathtt{A}_3} \quad \mu_1$ | $I_1 \quad R_{1,x} \quad R_{1,y} \quad R_{1,z} \quad \mu_1$ |
| $I_2 \quad f_{2,\mathtt{A}_1} \quad f_{2,\mathtt{A}_2} \quad f_{2,\mathtt{A}_3} \quad \mu_2$ | $I_2 \quad R_{2,x} \quad R_{2,y} \quad R_{2,z} \quad \mu_2$ |
| $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ | $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ |
| $I_i \quad f_{i,\mathtt{A}_1} \quad f_{i,\mathtt{A}_2} \quad f_{i,\mathtt{A}_3} \quad \mu_i$ | $I_i \quad R_{i,x} \quad R_{i,y} \quad R_{i,z} \quad \mu_i$ |
| $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ | $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ |
| `end atoms_frac` | `end atoms_cart` |

  - Fractional coordinates:
    The positions of atoms are specified in `atoms_frac` block. $I_i$ indicates atomic symbol of the $i$th atom. $f_{i,\mathbf{A}_1}$, $f_{i,\mathbf{A}}$ and $f_{i,\mathbf{A}_3}$ are position components of the $i$th atom in fractional coordinates and $\mu_i$ is magnetic moment of the $i$th atom.
    In fractional coordinate, atomic positions are in relative coordinates of the unit cell vectors [1] *i.e.* $\mathbf{R}_i = f_{i,\mathbf{A}_1}\mathbf{A}_1 + f_{i,\mathbf{A}_2}\mathbf{A}_2 + f_{i,\mathbf{A}_3}\mathbf{A}_3$.

  - Cartesian coordinates:
    The positions of atoms are specified in `atoms_cart` block. `units` is optional and determines the unit length of vector components of atomic positions in Cartesian coordinates.
    The options for `units` are:
    - `ang` (default)
    - `bohr`

    Same as the `unit_cell_cart` block, `units` can be omitted, in this case the default value is used *i.e.* Ang. $I_i$ indicates atomic symbol of the $i$th atom. $R_{i,x}$, $R_{i,y}$ and $R_{i,z}$ are position components of the $i$th atom in Cartesian coordinates and $\mu_i$ is magnetic moment of the $i$th atom.

- `neighbors_tol: [real]`
  If the distance difference between neighbor atoms of an atom is less than `neighbors_tol` Angstrom, they belong to the same neighbor shell.
  - The default is `0.001 Ang`.
- `shells_jij: [integer]`
  Shell number for exchange term of Hamiltonian.
  - The default is `1`.
- `spin_glass : [logical]`
  If set to `.true.`, the keywords to include spin glass in Heisenberg exchange parameters of Hamiltonian are written in `seedname.inp2.mcin` file.
- `ham_bij: [logical]`
  If set to `.true.`, Hamiltonian includes the bi-quadratic term.

[1] The unit cell vectors are defined in block of `unit_cell_cart`

- The default is `.false.`.

- `shells_bij: [integer]`
  Shell number for bi-quadratic term of Hamiltonian.
  - If `ham_bij = True`, the default is `1`.

- `ham_dij: [logical]`
  If set to `.true.`, Hamiltonian includes the Dzyaloshinskii-Moriya interaction.
  - The default is `.false.`.

- `shells_dij: [integer]`
  Shell number for the Dzyaloshinskii-Moriya interaction in Hamiltonian.
  - If `ham_dij = True`, the default is `1`.

- `length_unit: [character]`
  The length unit at the output files.
  Available options are:
  - `ang` (default)
  - `bohr`

- `parameter_unit: [character]`
  The energy unit of parameters in Hamiltonian at the output files.
  Available options are:
  - `ev` (default)
  - `ryd`

- `coordinate: [character]`
  The coordinate of positions at the output files.
  Available options are:
  - `frac` or `fractional` (default)
  - `cart` or `cartesian`

### 3.1.2 Output files of `inp1` step:

- `seedname.inp2.mcin`:

  This is the main output file of this step. This file is required for the next step. It contains the informations about the unit cell and atomic positions same as `seedname.inp1.mcin` file. In addition, it contains the blocks for the exchange parameters and other terms of the Hamiltonian that should be completed in the next step. If uncomment other quantities in this file, the relevant keywords will be written in `seedname.mcin` file for Monte-Carlo simulation step.

```
Begin Unit_Cell_Cart
    5.29177211   0.00000000   0.00000000
    0.00000000   5.29177211   0.00000000
    0.00000000   0.00000000   5.29177211
End Unit_Cell_Cart

Begin Atoms_Frac
 Mn       0.0000000   0.0000000   0.0000000    1.00
End Atoms_Frac

Parameter_unit  = Ryd
!! Order_parameter  = .True.
!! Sfactor          = .True.
!! Staggered_m      = .True.
!! Binning_error    = .True.
!! Spin_correlation = .True.
!! Energy_write     = .True.

## Hamiltonian
!! Boundary         =  Open
!! Ham_singleion    = .True.
!! Ham_field        = .True.
!! Spin_glass       = .True.  !Add the sigma parameters as sig=.. in Parameters_Jij Block

Begin Parameters_Jij
 Ryd
 t1=  1:t2=  1:sh= 1:Jij= ??????!:sig=?????!:d= 5.29177211
End Parameters_Jij
```

- `seedname.neigh`:

  It contains informations about the neighbors of the atoms in the unit cell.

- `seedname.xsf`:

  This is a xsf file for visualization of the unit cell which can be read by the `Xcrysden` or `Vesta` program. The run command for `XCrySDen` is:
  `xcrysden --xsf seedname.xsf`
  The run command for `Vesta` is:
  `VESTA seedname.xsf`

## 3.2 Second initialization step: `inp2` step

| | |
|---|---|
| input file | `seedname.inp2.mcin` |
| Output files | `seedname.mcin, seedname.neigh, seedname.mcout, seedname.xsf` |
| run command | `mc.x -inp2 seedname` |

### 3.2.1 Keywords of `seedname.inp2.mcin` step

In this step, the previous output file, `seedname.inp2.mcin`, should be modified as the input file. The parameters of Hamiltonian should be added to `seedname.inp2.mcin` file. For writing the keywords related to the calculation of the quantities in the main MC simulation, the related keywords to these quantities should be uncommented.

- `unit_cell_cart:`
  Same as described in keywords of seedname.inp1.mcin file.

- `atoms_frac:`
  Same as described in keywords of seedname.inp1.mcin file.

- `atoms_cart:`
  Same as described in keywords of seedname.inp1.mcin file.

- `order_parameter: [logical]`
  If set to `.true.`, then the keywords to calculate the order parameter are written in `seedname.mcin` file.

- `sfactor : [logical]`
  If set to `.true.`, then the keywords to calculate the neutron structure factor are written in `seedname.mcin` file.

- `staggered_m : [logical]`
  If set to `.true.`, then the keywords to calculate the staggered magnetization are written in `seedname.mcin` file.

- `binning_error : [logical]`
  If set to `.true.`, then the keywords to calculate the binning error are written in `seedname.mcin` file.

- `spin_correlation : [logical]`
  If set to `.true.`, then the keyword to calculate the spin correlation average is written in `seedname.mcin` file.

- `energy_write : [logical]`
  If set to `.true.`, then the keyword to write energies of each step of Monte-Carlo simulation is written in `seedname.mcin` file.

- `ham_bij: [logical]`
  Same as described in keywords of `seedname.inp1.mcin` file.

- `ham_dij: [logical]`
  Same as described in keywords of `seedname.inp1.mcin` file.

- `ham_singleion : [logical]`
  If set to `.true.`, then the keywords to include the single-ion term in Hamiltonian are written in `seedname.mcin` file.

- `ham_field : [logical]`
  If set to `.true.`, then the keywords to include the magnetic field term in Hamiltonian are written in `seedname.mcin` file.

- `spin_glass : [logical]`
  If set to `.true.`, then the keywords to include spin glass in exchange parameters of Hamiltonian are written in `seedname.mcin` file.

- `neighbors_tol: [real]`
  Same as described in keywords of `seedname.inp1.mcin` file.

- `parameters_jij:`
  This block specifies the exchange parameters of Hamiltonian.

```
begin Parameters_jij
   [units]
   type1:type2:shell:Jij_param!:sigma!:distance
end Parameters_jij
```

  ◇ `units`
  Optional: The first line determines the units of exchange parameter and length. This line is optional. Available options for `[units]` are:

  - `ev,ang` (default)
  - `ev,bohr`
  - `ryd,ang`
  - `ryd,bohr`

  The order of length and parameter units is not important. If `units` not present, the default value is taken. If one of the units is not specified, the default value is set to that, e.g. if units set to `Bohr`, in this case the unit of length and parameters are Bohr and eV, respectively.

  ◇ `type1:type2`
  In second line `type1` is the type of first atom and `type2` is the type of neighbors of first atom.
  Example: `t1= 1;t2= 1`

  ◇ `shell`
  It is the shell number between `type1` and `type2` atoms.
  Example: `sh= 1`

  ◇ `Jij_param`
  It is the value of the exchange parameter for these two atoms (`type1` and `type2` atoms).
  Example: `Jij= 0.00022047`

  ◇ `sigma`
  Optional: If `spin_glass = True`, the `sigma` should be specified, if don't specify the zero value is written for `sigma` in `seedname.mcin` file.
  Example: `sig= 0.00002`

  ◇ `distance`
  It specifies the distance between `type1` and `type2` atoms. It is just for helping and it will not be read.
  Example: `d= 5.29177211`

Full example: `t1= 1:t2= 1:sh= 1:Jij= 0.00022047!:sig=?????!:d= 5.29177211`
Which means type 1 atoms are first neighbors and have exchange interaction equal to 0.00022047 in units specified by `units` in the first line. Because of `!` sign, the two other parameters (*i.e.* `sig` and `d`) have not been read by the program. If `spin_glass = True`, `!` sign that is before `sig` keywords should be deleted and the user should write the value of sigma.

- `parameters_bij:`
  Same as the `parameters_jij` block. In this block `bij` is the bi-quadratic constant of Hamiltonian. The only difference is that keyword `sig` does not present here.

```
begin Parameters_bij
   [units]
   type1:type2:shell:Bij_param!:distance
end Parameters_bij
```

  Example: `t1= 1:t2= 1:sh= 2:Bij=-0.00004400!:d= 5.57609693`

- `parameters_dij:`
  Same as the `parameters_bij` block. In this block `dij` is the Dzyaloshinskii-Moriya interaction.

```
begin Parameters_dij
   [units]
   type1:type2:shell:Dij_param!:distance
end Parameters_bij
```

  Example: `t1= 1:t2= 1:sh= 1:Dij=-0.00003000!:d= 5.57609693`

### 3.2.2 Output files of `inp2` step:

- `seedname.mcin`

  The input file for Monte-Carlo step. It contains informations about the unit cell, position of atoms, blocks for specifying the Hamiltonian and relative keywords to Monte-Carlo simulations which can be modified.

```
Begin Unit_Cell_Cart
    5.29177211   0.00000000   0.00000000
    0.00000000   5.29177211   0.00000000
    0.00000000   0.00000000   5.29177211
End Unit_Cell_Cart

Begin Atoms_Frac
 Mn       0.0000000   0.0000000   0.0000000    1.00
End Atoms_Frac

tem_start          =   5
tem_end            =   5
tems_num           =   1
!! tems_mode          = man
!! tems               = 5.00 10.00 15.00 20.00

!! Pt                = .True.
!! Pt_steps_swap      = 10

steps_warmup      =       100000
steps_mc          =       200000
steps_measure     =            2

initial_sconfig   =        ferro
mcarlo_mode       =       random

supercell_size    =      4     4     4

Parameter_unit    = Ryd

 ## Hamiltonian
Begin Jij_parameters
 Ryd
 f1=    0.000000,    0.000000,    0.000000:f2=    1.000000,    0.000000,    0.000000:jij=  0.00022047!:sh=  1!:t1=  1:t2=  1
 f1=    0.000000,    0.000000,    0.000000:f2=    0.000000,    1.000000,    0.000000:jij=  0.00022047!:sh=  1!:t1=  1:t2=  1
 f1=    0.000000,    0.000000,    0.000000:f2=    0.000000,    0.000000,    1.000000:jij=  0.00022047!:sh=  1!:t1=  1:t2=  1
 f1=    0.000000,    0.000000,    0.000000:f2=    0.000000,    0.000000,   -1.000000:jij=  0.00022047!:sh=  1!:t1=  1:t2=  1
 f1=    0.000000,    0.000000,    0.000000:f2=    0.000000,   -1.000000,    0.000000:jij=  0.00022047!:sh=  1!:t1=  1:t2=  1
 f1=    0.000000,    0.000000,    0.000000:f2=   -1.000000,    0.000000,    0.000000:jij=  0.00022047!:sh=  1!:t1=  1:t2=  1
End Jij_parameters
```

- `seedname.neigh`

  same as described above.

- `seedname.xsf`

  same as described above.

# Chapter 4

# Monte-Carlo simulation: `mc.x` step

| input file | `seedname.mcin` |
|---|---|
| Output files | `seedname.mcout, seedname_mc.dat, seedname_sconfig.dat, seedname_pm.dat, seedname_spincorr.dat, seedname_op.dat, seedname_staggered.dat` |
| run command | Serial: `mc.x seedname`<br>Parallel: `mpirun -np num_proc seedname` |

### 4.0.1 Keywords of `seedname.mcin` file

A file with `.mcin` suffix is the input file of Monte-Carlo step, which has been generated by previous step. In addition to information about the unit cell and atoms, keywords of Monte-Carlo simulation like temperature are listed in this file, that needs to be modified.

- `unit_cell_cart:`
  Same as described in the previous chapter.
- `atoms_frac:`
  Same as described in the previous chapter.
- `atoms_cart:`
  Same as described in the previous chapter.
- `length_unit: [character]`
  Same as described in the previous chapter.
- `parameter_unit: [character]`
  Same as described in the previous chapter.
- `coordinate: [character]`
  Same as described in the previous chapter.
- `tems_num: [integer]`
  Number of temperatures for calculation of thermodynamic properties.
  - The default value is `1`.
- `tem_start: [real]`
  Starting temperature in Kelvin unit.
  - The default value is `5.0`K.
- `tem_end: [real]`
  Ending temperature in Kelvin unit.
  - The default value is `5.0`K.
- `tems_mode: [character]`
  Determining how the temperatures are set.
  Available options are:

- ➢ `lin or linear` (default): The range between `tem_start` and `tem_end` is uniformly divided by the number of temperatures *i.e.* `tems_num` keyword ($T_i = T_s + (T_e - T_s) \times (i-1)/(N_T - 1)$).
- ➢ `inv or inverse`: The range between inverse `tem_start` and inverse `tem_end` is uniformly divided by the number of temperatures ($1/T_i = 1/T_s + (1/T_e - 1/T_s) \times (i-1)/(N_T - 1)$).
- ➢ `log or logarithmic`: The range between `tem_start` and `tem_end` is logarithmically divided by the number of temperatures ($T_i = T_s \times exp(\frac{ln(T_e/T_s)(i-1)}{(N_T-1)})$
- ➢ `man or manual` : Temperatures are determined manually via `tems` keyword.

- `tems(tems_num): [real]`
  If `tems_mode = man`, then the temperatures read from `tems` keyword.
  - ➢ No default.
- `steps_warmup: [integer]`
  Number of steps for initial warm-up of the system.
  - ➢ The default value is `100000`.
- `steps_mc: [integer]`
  Number of steps for sampling from system.
  - ➢ The default value is `200000`.
- `steps_measure: [integer]`
  Number of steps between successive sampling [1] .
  - ➢ The default value is `2`.
- `supercell_size(3): [integer]`
  Dimensions of the simulation supercell.
  - ➢ The default is `4 4 4`.
- `initial_sconfig: [character]`
  Initial spin configuration of system.
  Available options are:
  - ➢ `ferro` or `ferromagnetism` (default).
  - ➢ `rand` or `random`.
  - ➢ `file`: Spin configuration reads from `seedname_sconfig.dat` file.
- `mcarlo_mode: [character]`
  Determining how the new direction of spin is chosen during each Monte-Carlo step.
  Available options are:
  - ➢ `rand` or `random` (default): Completely at random
  - ➢ `const` or `constraint`: The new direction of spin is chosen randomly inside a cone. The cone axis is the previous direction of the spin and the apex angle of cone ($2\Delta\Theta$) can be specified by `tilt_angles_max` keyword.
- `tilt_angles_max: [real]`
  If `mcarlo_mode = const`, $2\Delta\Theta$ =`2*tilt_angles_max*`$\pi$ determines the apex angle of cone which is explained above.
  - ➢ If `mcarlo_mode = const`, the default is `0.125`.
- `seeds(number of nodes): [integer]`
  A list of seeds for generation of the random numbers.
  - ➢ If not specify, seeds getting from reading of the computer's clock.
    Example: to enter seeds of 1, 19, 5, 6, 7, 8, 10, 12:
    `seeds = 1,19,5-8,10,12`

[1] The total number of MC steps is `steps_mc` × `steps_measure`.

- `pt: [logical]`
  If `pt = .true.`, the parallel tempering algorithm is done.

  ➢ The default is `.false.`.

- `pt_steps_swap: [integer]`
  If `pt = .true.`, after `pt_steps_swap` Monte-Carlo steps, configurations swap is done.

  ➢ The default is `1`.

- `pt_print_swap: [integer]`
  If `pt = .true.`, after `pt_print_swap` Monte-Carlo steps, ordering of configurations is printed.

  ➢ The default is `10000`.

- `staggered_m: [logical]`
  If set to `.true.`, the staggered magnetization will be calculated and printed to a file `seedname_staggered.dat`. The staggered magnetization is the summation of magnetic moments with $\pm 1$ coefficients inside the unit cell *i.e.* $\frac{\sqrt{(\sum_{i=1}^{N} c_i \mathbf{S}_i)\cdot(\sum_{i=1}^{N} c_i \mathbf{S}_i)}}{N}$ where $c_i$ is either $+1$ or $-1$ and can be specified by `staggered_m_coeff`.

  ➢ The default is `.false.`.

- `staggered_m_coeff(number of atoms): [integer]`
  The coefficients for calculating staggered magnetization.

  ➢ The default is `1` for all atoms.

- `order_parameter: [logical]`
  If `order_parameter= .true.`, user defined order parameter, its susceptibility and its fourth-order Binder cumulant are calculated. The quantities will be written to `seedname_op.dat` file. The order parameter is defined by $\frac{\sum_{i=1}^{N} \mathbf{S}_i.\mathbf{d}_i}{N}$ where $\mathbf{d}_i$ vectors are specified inside `order_parameter_axes` block.

- `order_parameter_axes:`
  If `order_parameter = .true.`, the $\mathbf{d}_i$ vectors should be specified for each atom. They can be introduced in two coordinates like atomic positions:

| 1) Fractional coordinates | 2) Cartesian coordinates |
|---|---|
| `begin order_parameter_axes_frac` | `begin order_parameter_axes_cart` |
| | `[units]` |
| $I_1 \quad f_{1,\mathtt{A}_1} \quad f_{1,\mathtt{A}_2} \quad f_{1,\mathtt{A}_3}$ | $I_1 \quad d_{1,x} \quad d_{1,y} \quad d_{1,z}$ |
| $I_2 \quad f_{2,\mathtt{A}_1} \quad f_{2,\mathtt{A}_2} \quad f_{2,\mathtt{A}_3}$ | $I_2 \quad d_{2,x} \quad d_{2,y} \quad d_{2,z}$ |
| $\vdots \quad \vdots \quad \vdots \quad \vdots$ | $\vdots \quad \vdots \quad \vdots \quad \vdots$ |
| $I_i \quad f_{i,\mathtt{A}_1} \quad f_{i,\mathtt{A}_2} \quad f_{i,\mathtt{A}_3}$ | $I_i \quad d_{i,x} \quad d_{i,y} \quad d_{i,z}$ |
| $\vdots \quad \vdots \quad \vdots \quad \vdots$ | $\vdots \quad \vdots \quad \vdots \quad \vdots$ |
| `end order_parameter_axes_frac` | `end order_parameter_axes_cart` |

  Its not necessary for the vectors to be normal (the program will normalize them).

- `binning_error: [logical]`
  If `binning_error = .true.`, error of the energy, magnetization and order parameter, (if `order_parameter = .true.`) will be computed via binning analysis method ($\Delta_A^{(l)} \approx \sqrt{\frac{1}{M_l(M_l-1)}\sum_{i=1}^{M_l}(A_i^{(l)} - \bar{A}^{(l)})}$), where $M_l = \frac{M}{2^l}$ and $M$ is number of Monte-Carlo steps. The values of binning error will be written to a file `seedname_binerror.dat`.

  ➢ The default is `.false.`.

- `binning_level(:): [integer]`
  Determines a list of the $l$ values for binning analysis method.

  ➢ The default is `1-5`.
  Example: to compute the binning error for binning levels of 3, 4, 5, 8 and 10:
  `binning_level = 3-5, 8, 10`

- `spin_correlation: [logical]`
  If `spin_correlation = .true.`, spin correlation will be calculated and written to a file `seedname_spincorr.dat`. These quantities will be calculated: $\langle\sum_{i,j}^{N}\mathbf{S}_i\cdot\mathbf{S}_j/N\rangle$,$\langle\sum_{i,j}^{N}|\mathbf{S}_i\cdot\mathbf{S}_j|/N\rangle$, and $\langle|\sum_{i,j}^{N}\mathbf{S}_i\cdot\mathbf{S}_j/N|\rangle$

- `energy_write: [logical]`
  If `energy_write = .true.`, energy of each Monte-Carlo step will be written to a file `seedname_energy-T#.dat`.

  ➢ The default is `.false.`.

- `energy_num_print: [integer]`
  After `energy_num_print` Monte-Carlo steps, the energies of these steps will be printed.

  ➢ The default is `1000`.

- `sfactor: [logical]`
  If `sfactor = .true.`, neutron structure factor will be calculated at a plane.

  ➢ The default is `.false.`.

- `sfactor_polar(3): [real]`
  The polarization vector in reduced coordinates.

  ➢ No default.

- `sfactor_corner(3): [real]`
  Corner of the plane in the reduced coordinates of q-space.

  ➢ The default is `0.0 0.0 0.0`.

- `sfactor_q1(3): [real]`
  The first vector defining the plane in the reduced coordinates of q-space.

  ➢ The default is `1.0 0.0 0.0`.

- `sfactor_q2(3): [real]`
  The second vector defining the plane in the reduced coordinates of q-space.

  ➢ The default is `0.0 0.0 1.0`.

- `sfactor_2dqmesh(2): [integer]`
  Dimensions of the q-point grid covering the plane.

  ➢ The default is `50 50`.

- `sfactor_steps_measure: [real]`
  After `sfactor_steps_measure` Monte-Carlo steps, the neutron structure factor is computed.

  ➢ The default is `100`.

- `spin_glass: [logical]`
  If `spin_glass = .true.`, the exchange parameters of Hamiltonian are chosen via Gaussian random numbers.

  ➢ The default is `.false.`.

- `spin_glass_seed: [integer]`
  Seed for the Gaussian random numbers generator, if `spin_glass = .true.`.

  ➢ If not specify, seed gets from the computer's clock.

- `boundary: [character]`
  Determines the interaction of spins on the boarders of supercell. Available options are:

  ➢ `peri` or `periodic` (default)

  ➢ `open`

- `jij_parameters:`
  This block specifies the position of atoms, their neighbors and Heisenberg exchange interaction between them.

```
Begin Jij_parameters
 [units]
 site1:site2:jij_param:[shell]:[sigma]
End Jij_parameters
```

- ◇ `units`
  Optional: The units of Heisenberg exchange parameters and sites specify in first line. `eV` and `Ryd` are options for exchange parameters and `Ang` and `Bohr` for atomic sites if given in Cartesian coordinates. Available options for `[units]` are:
  - ➢ `ev,ang` (default)
  - ➢ `ev,bohr`
  - ➢ `ryd,ang`
  - ➢ `ryd,bohr`
- ◇ `site1:site2:jij_param`
  Atom at the `site1` position and its neighbor at the `site2` position are specified in the Cartesian or fractional coordinates, have `jij_param` exchange interaction.
  Example: `c1=0.0,0.0,0.0:c2=2.5,2.5,2.5:jij=0.002`
  Which means two atoms are located in (0.0,0.0,0.0) and (2.5,2.5,2.5) respectively in Cartesian coordinates and have exchange interaction equal to 0.002 in energy unit specified by the `units` in the first line.
  Example: `f1=0.0,0.0,0.0:f2=0.5,0.5,0.5:jij=0.004`
  Which means two atoms are located in (0.0,0.0,0.0) and (0.5,0.5,0.5) respectively in fractional coordinates have exchange interaction equal to 0.004 in energy unit specified by the `units` in the first line.
- ◇ `shell`
  Optional: If `spin_correlation = true`, the shell number should specify as "`sh=?`" in `jij_parameters` block.
  Example: `sh = 1`
  Which means the atoms at the `site1` and `site2` positions are the first neighbors of each other.
- ◇ `sigma`
  Optional: If `spin_glass = true`, the value of broadening of Gaussian function for random number generator determine as `sig=` in units specified by the optional `units` in the first line
  Example: `sig = 0.2`
  Which sets the sigma parameter to 0.2 in the units specified by `units`.

- `ham_bij: [logical]`
  Determining whether the Hamiltonian includes the bi-quadratic term.
- `bij_parameters:`
  This block specifies the position of atoms, their neighbors and bi-quadratic interaction between them.

```
Begin Bij_parameters
 [units]
 site1:site2:bij_param
End Bij_parameters
```

`units`
Same as described above for `jij_parameters`.
`site1:site2:bij_param`
Same as described above for `jij_parameters`.
Example: `c1=0.0,0.0,0.0:c2=2.5,2.5,2.5:bij=0.0002`

- `ham_dij: [logical]`
  Determining whether the Hamiltonian includes the Dzyaloshinskii-Moriya term.
- `dij_parameters:`
  This block specifies the position of atoms, their neighbors and Dzyaloshinskii-Moriya interaction between them.

```
Begin Dij_parameters
 [units]
 site1:site2:dij_param
End Dij_parameters
```

`units`
Same as described above for `jij_parameters`.
`site1:site2:dij_param`
Same as described above for `jij_parameters`.
Example: `c1=0.0,0.0,0.0:c2=2.5,2.5,2.5:dij=0.0002`

- `dij_vectors:`
Direction of dij vectors specifies by this block. Ordering of vectors should be same as `dij_parameters` block. Vectors can be defined in two coordinates:

| 1) Fractional coordinates | 2) Cartesian coordinates |
|---|---|
| `begin dij_vectors_frac` | `begin dij_vectors_cart` |
| $f_{1,\mathtt{A}_1}$ $f_{1,\mathtt{A}_2}$ $f_{1,\mathtt{A}_3}$ | `[units]` |
| $f_{2,\mathtt{A}_1}$ $f_{2,\mathtt{A}_2}$ $f_{2,\mathtt{A}_3}$ | $d_{1,x}$ $d_{1,y}$ $d_{1,z}$ |
| $\vdots$ $\vdots$ $\vdots$ | $d_{2,x}$ $d_{2,y}$ $d_{2,z}$ |
| $f_{i,\mathtt{A}_1}$ $f_{i,\mathtt{A}_2}$ $f_{i,\mathtt{A}_3}$ | $\vdots$ $\vdots$ $\vdots$ |
| $\vdots$ $\vdots$ $\vdots$ | $d_{i,x}$ $d_{i,y}$ $d_{i,z}$ |
| `end dij_vectors_frac` | $\vdots$ $\vdots$ $\vdots$ |
| | `end dij_vectors_cart` |

Options are `Ang` and `bohr` for `units`. Since only the direction is important and the program will normalize the vectors, being of units does not matter.

- `ham_singleion: [logical]`
If set to `.true.`, Hamiltonian has the single-ion term ($\mathcal{H} = \sum_{i=1}^{N} \Delta_i \, \hat{z}_i \cdot \mathbf{S}_i$).
  - The default is `.false.`.

- `singleion_axes(:,:):`
This block specifies the direction of anisotropy and anisotropy parameter. The axes can be defined in two coordinates:

| 1) Fractional coordinates | 2) Cartesian coordinates |
|---|---|
| `begin singleion_axes_frac` | `begin singleion_axes_cart` |
| `[units]` | `[units]` |
| $I_1$ $f_{1,\mathtt{A}_1}$ $f_{1,\mathtt{A}_2}$ $f_{1,\mathtt{A}_3}$ $\Delta_1$ | $I_1$ $\mathcal{Z}_{1,x}$ $\mathcal{Z}_{1,y}$ $\mathcal{Z}_{1,z}$ $\Delta_1$ |
| $I_2$ $f_{2,\mathtt{A}_1}$ $f_{2,\mathtt{A}_2}$ $f_{2,\mathtt{A}_3}$ $\Delta_2$ | $I_2$ $\mathcal{Z}_{2,x}$ $\mathcal{Z}_{2,y}$ $\mathcal{Z}_{2,z}$ $\Delta_2$ |
| $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ | $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ |
| $I_i$ $f_{i,\mathtt{A}_1}$ $f_{i,\mathtt{A}_2}$ $f_{i,\mathtt{A}_3}$ $\Delta_i$ | $I_i$ $\mathcal{Z}_{i,x}$ $\mathcal{Z}_{i,y}$ $\mathcal{Z}_{i,z}$ $\Delta_i$ |
| $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ | $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ |
| `end singleion_axes_frac` | `end singleion_axes_cart` |

Program will normalize the direction of vectors.
Available options for `units` in `singleion_axes_frac` block are:

- `ev` (default)
- `ryd`

Available options for `units` in `singleion_axes_cart` block are:

- `ev,ang` (default)
- `ev,bohr`
- `ryd,ang`
- `ryd,bohr`

The order of length and parameter units is not important. If `units` not present, the default value is taken. If one of the units is not specified, the default value is set to that.

- `ham_field: [logical]`
  If set to `.true.`, the magnetic field term add to Hamiltonian ($\mathcal{H} = \sum_{i=1}^{N} B_i \hat{\mathcal{Z}}_i \cdot \mathbf{S}_i$).

  ➢ The default is `.false.`.

- `field_axes(:,:):`
  This block specifies the direction of applied magnetic field and its value . The axes can be defined in two coordinates:

| 1) Fractional coordinates | 2) Cartesian coordinates |
|---|---|
| `begin field_axes_frac` | `begin field_axes_cart` |
| | `[units]` |
| $I_1 \quad f_{1,\mathtt{A}_1} \quad f_{1,\mathtt{A}_2} \quad f_{1,\mathtt{A}_3} \quad B_1$ | $I_1 \quad \mathcal{Z}_{1,x} \quad \mathcal{Z}_{1,y} \quad \mathcal{Z}_{1,z} \quad B_1$ |
| $I_2 \quad f_{2,\mathtt{A}_1} \quad f_{2,\mathtt{A}_2} \quad f_{2,\mathtt{A}_3} \quad B_2$ | $I_2 \quad \mathcal{Z}_{2,x} \quad \mathcal{Z}_{2,y} \quad \mathcal{Z}_{2,z} \quad B_2$ |
| $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ | $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ |
| $I_i \quad f_{i,\mathtt{A}_1} \quad f_{i,\mathtt{A}_2} \quad f_{i,\mathtt{A}_3} \quad B_i$ | $I_i \quad \mathcal{Z}_{i,x} \quad \mathcal{Z}_{i,y} \quad \mathcal{Z}_{i,z} \quad B_i$ |
| $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ | $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$ |
| `end field_axes_frac` | `end field_axes_cart` |

  Program will normalize the direction vector.
  The unit of B is Tesla.

### 4.0.2 Output files of `mc.x` step:

- `seedname.mcout`

  The information about the input file keywords, number of rejected and accepted steps are written in this file.

- `seedname_mc.dat`

  This file contains the quantities such as the magnetization, average energy, magnetic specific heat ($C_M = \frac{\langle E^2\rangle - \langle E\rangle^2}{Nk_BT^2}$), susceptibility ($\chi = \frac{\langle M^2\rangle - \langle M\rangle^2}{Nk_BT}$), fourth-order cumulant of energy ($U_E = 1 - \frac{1}{3}\frac{\langle E^4\rangle}{\langle E^2\rangle^2}$) and fourth-order Binder cumulant of magnetization ($U_M = 1 - \frac{1}{3}\frac{\langle M^4\rangle}{\langle M^2\rangle^2}$) for each temperature.

  Example of `seedname_mc.dat` file:

```
#written on 25Jul2019 at 17:15:19
#------------------------------------------------ MONTECARLO ----------------------------------------------------#
# Temp      Magnetization     Energy_ave          C_M              Sus              U_E              U_M       #
#----------------------------------------------------------------------------------------------------------------#
  4.0000   0.97301411E+00  -0.10039260E+03  0.10223041E+01  0.40787762E-03  0.66666450E+00  0.66666437E+00
  5.5238   0.96244063E+00  -0.98832823E+02  0.10273965E+01  0.56983079E-03  0.66666239E+00  0.66666214E+00
  7.0476   0.95169101E+00  -0.97259004E+02  0.10410840E+01  0.74522096E-03  0.66665938E+00  0.66665894E+00
  8.5714   0.94074662E+00  -0.95672098E+02  0.10474720E+01  0.92460633E-03  0.66665546E+00  0.66665474E+00
 10.0952   0.92959523E+00  -0.94069462E+02  0.10571282E+01  0.11155972E-02  0.66665044E+00  0.66664931E+00
 11.6190   0.91824981E+00  -0.92451287E+02  0.10697373E+01  0.13109812E-02  0.66664414E+00  0.66664260E+00
```

- `seedname_sconfig.dat`

  The first line gives the date and time that the file was created. The second line clarifies the size of supercell. The third line states the number of total atoms (`num_atom*supercell_size(1)*supercell_size(2)*supercell_size(3)`) and number of temperatures (`num_tems`). Then there are `num_tems` blocks of data, each starts by a line states the temperature followed by (number of total atoms) lines containing the spin direction for each site of atom. The spin of atoms are written in `x, y,` and `z` order. This file will be read, if `initial_sconfig = file`. In this case, temperatures and number of total atoms must be the same as `seedname.mcin` input file.

  Example of `seedname_sconfig.dat` file:

```
Created on 26Jul2019 at 14:57:10
   12    12    12
 3456    64
   4.000000
```

```
 0.021731    0.170958    0.985039
-0.111871    0.155007    0.981559
-0.163746    0.029309    0.986067
-0.232591   -0.086163    0.968750
 0.427986    0.014272    0.903673
-0.257394    0.058855    0.964512
 .
 .
 .
 5.206349
 0.167293   -0.149844   -0.974454
 0.278237    0.188479   -0.941839
-0.186044    0.137737   -0.972839
 0.066792   -0.105264   -0.992199
 0.324208   -0.060875   -0.944025
 0.006523   -0.219877   -0.975506
 .
 .
 .
```

There is a utility program for visualization of system spin configuration in utility directory. The `vispin.x` command will arrange the required data for visualization with the `Xcrysden` program. The `vispin.x` needs two files, `seedname.xsf` and `seedname_sconfig.dat` files. To run just type:
`vispin.x seedname`
The program will ask the size of supercell for visualization:
`Please eneter the supercell size for visualization (for example:  2 2 2):`
For each temperature, a file is created. The output is as `seedname_vispini#.dat` where # shows the temperature number.

- `seedname_pm.dat`

  Probability distribution function of the magnetization, order parameter(if `order_parameter = .True.`), and staggered magnetization (if `staggered_m = .True.`) per unit cell, are written in this file.

  Example of `seedname_pm.dat` file:

```
#written on 27Jul2019 at 14:26:46
#--------------------------- MONTECARLO -------------------------------#
#       M or OP                 P(M)                    P(OP)         #
#----------------------------------------------------------------------#
#T=    45.0000
 -0.40000000E+01           0.00000000E+00           0.00000000E+00
 -0.39900000E+01           0.00000000E+00           0.21000000E-05
 -0.39800000E+01           0.00000000E+00           0.10700000E-04
 -0.39700000E+01           0.00000000E+00           0.38000000E-04
 -0.39600000E+01           0.00000000E+00           0.74200000E-04
 -0.39500000E+01           0.00000000E+00           0.12770000E-03
 -0.39400000E+01           0.00000000E+00           0.21910000E-03
 -0.39300000E+01           0.00000000E+00           0.30930000E-03
 -0.39200000E+01           0.00000000E+00           0.42990000E-03
 -0.39100000E+01           0.00000000E+00           0.57120000E-03
 -0.39000000E+01           0.00000000E+00           0.74520000E-03
 -0.38900000E+01           0.00000000E+00           0.91300000E-03
 -0.38800000E+01           0.00000000E+00           0.11194000E-02
 -0.38700000E+01           0.00000000E+00           0.13601000E-02
 -0.38600000E+01           0.00000000E+00           0.15820000E-02
 -0.38500000E+01           0.00000000E+00           0.18151000E-02
 -0.38400000E+01           0.00000000E+00           0.20443000E-02
```

- `seedname_op.dat`

  It will be created if `order_parameter = .True.`. The file contains the value of order parameter, susceptibility derived from order parameter and fourth-order Binder cumulant of order parameter for each temperature.

  Example of `seedname_op.dat` file:

```
#written on 27Jul2019 at 14:26:46
#------------------------------ MONTECARLO ----------------------------------#
#    Temp              OP                 Sus_OP                 U_OP        #
#----------------------------------------------------------------------------#
     3.0000      0.98380800E+00       0.28590184E-03       0.66666637E+00
     3.6667      0.98008392E+00       0.36023649E-03       0.66666621E+00
     4.3333      0.97634897E+00       0.42725623E-03       0.66666602E+00
     5.0000      0.97250993E+00       0.51417504E-03       0.66666576E+00
     5.6667      0.96863205E+00       0.57603718E-03       0.66666551E+00
     6.3333      0.96471222E+00       0.63757355E-03       0.66666522E+00
     7.0000      0.96071416E+00       0.73423020E-03       0.66666481E+00
```

- `seedname_spincorr.dat`

  It will be created if `spin_correlation = .True.`.

  Example of `seedname_spincorr.dat` file:

```
#written on  10ct2019 at 17:48:28
#------------------------------------------------- MONTECARLO ---------------------------------------------------#
#  Temp    Shell   Atom1   Type1   Atom2   Type2   <\sum_{ij} Si.Sj/N>   <\sum_{ij}|Si.Sj|/N>   <|\sum_{ij}Si.Sj|/N> #
#----------------------------------------------------------------------------------------------------------------#
  4.0000     1      Mn       1      Mn       1        -0.004510787           0.947671182             0.004510787
  4.0000     2      Mn       1      Mn       1        -0.965193441           0.965193441             0.965193441
  4.0000     3      Mn       1      Mn       1         0.002716083           0.947556974             0.002716083
  4.0000     4      Mn       1      Mn       1         0.957626944           0.957626945             0.957626944
```

- `seedname_staggered.dat`

  It will be created if `staggered_m = .True.`.

  Example of `seedname_staggered.dat` file:

```
#written on 14Aug2019 at 12:42:25
#------------------------------ MONTECARLO --------------------------------#
#    Temp         Staggered_m          Sus_Staggered_m       U_Staggered_m  #
#--------------------------------------------------------------------------#
     4.0000      0.10000000E+01       0.00000000E+00        0.66666667E+00
     5.2063      0.10000000E+01       0.00000000E+00        0.66666667E+00
```

- `seedname_binerror.dat`

  It will be created if `binning_error = .True.`. If `staggered_m = .True.`, binning error of staggered magnetization and if `order_parameter = .True.`, binning error of order parameter in addition to binning error of energy and magnetization are written in this file.

  Example of `seedname_binerror.dat` file:

```
#----------------------------- MONTECARLO -------------------------------#
#  Binning_level     Error(E)          Error(M)      Error(Staggered_m)  #
#------------------------------------------------------------------------#
#T=     4.0000
         3         0.87918552E-03    0.11215477E-04    0.11585869E-04
         4         0.12131111E-02    0.14096887E-04    0.14483712E-04
         5         0.16413111E-02    0.16722497E-04    0.17084558E-04
         8         0.32116285E-02    0.19928343E-04    0.22639476E-04
        10         0.41449552E-02    0.15621591E-04    0.37641995E-04
#T=     5.2063
         3         0.10832774E-02    0.12533707E-04    0.12822853E-04
         4         0.14891987E-02    0.15463829E-04    0.15806900E-04
         5         0.20082889E-02    0.17339297E-04    0.17794397E-04
         8         0.37238151E-02    0.17278975E-04    0.16483297E-04
        10         0.50696623E-02    0.20335130E-04    0.21200919E-04
```

- `seedname_energy-T#.dat`

  It will be created if `energy_write = .True.`. For each temperature, a file is created. # in the filename is representative of the temperature number. The files contain the energy of each Monte-Carlo step. The first line gives the date and time in which the file is created. The second line states the temperature. The third line gives the number of Monte-Carlo steps (`steps_mc`). In the subsequent lines(`steps_mc` lines), Monte-Carlo steps are written in the left column, and the corresponding energies in units of Kelvin are written in the right column.

  Example of `seedname_energy-T#.dat` file:

```
#written on 14Aug2019 at 11:51:17
  # T =    4.0000
           10000
           1     -0.10729348E+03
           2     -0.10727368E+03
           3     -0.10727368E+03
           4     -0.10726382E+03
           5     -0.10719288E+03
           6     -0.10728927E+03
           7     -0.10730791E+03
           8     -0.10734059E+03
           9     -0.10744699E+03
          10     -0.10748358E+03
```

There is a utility program for computing the histogram of energy in utility directory. The `mc-hist.x` computes the energy histogram, for example just type:
`mc-hist.x seedname_energy-T001.dat`
The program will ask a value for Gaussian broadening:
`Enter gaussian broadening =`
If the Gaussian broadening is set to zero no smearing is included in the histogram, in otherwise a smearing with the Methfessel function is included. The output file for above example is `seedname_energy-T001.dat.hist`.

- `seedname_sf-T#.dat`

  It will be created if `sfactor = .True.`. This file is created for each temperature. # in the filename is the temperature number. The first and second column are the grid coordinates in unit of $\mathtt{Ang}^{-1}$. The third column gives the neutron structure factor.

  Example of `seedname_sf-T#.dat` file:

```
0.00000000E+00    0.00000000E+00    0.00000000E+00
0.00000000E+00    0.13816693E+00    0.16880325E-01
0.00000000E+00    0.27633385E+00    0.69726350E-01
0.00000000E+00    0.41450078E+00    0.22007001E+00
0.00000000E+00    0.55266770E+00    0.47667495E+00
0.00000000E+00    0.69083463E+00    0.55251300E+03
0.00000000E+00    0.82900155E+00    0.47667495E+00
0.00000000E+00    0.96716848E+00    0.22007001E+00
0.00000000E+00    0.11053354E+01    0.69726350E-01
0.00000000E+00    0.12435023E+01    0.16880325E-01
0.00000000E+00    0.13816693E+01    0.11538101E-01
0.00000000E+00    0.15198362E+01    0.16880325E-01
0.00000000E+00    0.16580031E+01    0.69726350E-01
0.00000000E+00    0.17961700E+01    0.22007001E+00
0.00000000E+00    0.19343370E+01    0.47667495E+00
0.00000000E+00    0.20725039E+01    0.55251300E+03
0.00000000E+00    0.22106708E+01    0.47667495E+00
```

- `seedname_sf.py`

  It will be created if `sfactor = .True.`, a python script to plot the neutron structure factor.

- `seedname_sf.gnu`

  It will be created if `sfactor = .True.`, a gnuplot script to plot the neutron structure factor.

## 4.1 Optimized temperatures

To have a more efficient parallel tempering simulation, it is better to optimize the temperature set. The `optimize_T.py` program in the utility/optimize-temperatures directory, optimizes temperatures in a way that the acceptance ratio in parallel tempering swapping becomes equal for all temperatures. The program needs `python3` and scientific python libraries: `NumPy`, `Scipy`, and `Matplotlib`. To use this program, run a short MC simulation (for example, with 100000-200000 steps) to obtain the energies of the system for a range of temperatures. Then, extract energies and temperatures data from `seedname_mc.dat` file. To extract data, you can use the following `awk` command:

- `awk '{print $1,$3}'  seedname_mc.dat > E.dat`

The default name of the data file is `E.dat`, and contains two columns. The first column is temperatures, and the second column is energies.

To run the program, type:

- `python optimize_T.py`

  or

- `python optimize_T.py --data E.dat --alpha 0.9 --epsilon 1e-8`

To get help type:

- `python optimize_T.py -h`

If there is a convergence problem, use a smaller mixing $\alpha$ parameter by `--alpha` option in the command line.

In the case that this program doesn't work, use a program developed by Dr. Ruben Andrist. The program can be downloaded from Github (`https://github.com/andrist/tempering`). The program is written in the `R` language. So you need to install the `R` language. To prepare the input file for the program, type the following commands:

- `awk '{print $1,$3}'  seedname_mc.dat > E.dat`
- `sed -i '/#/d'  E.dat`
- `sed -i '1 i\T e'  E.dat`

Example of `E.dat` file for Andrist's code:

```
T e
10.2857 -0.16536985E+03
11.3333 -0.16428562E+03
12.3810 -0.16319797E+03
13.4286 -0.16210542E+03
14.4762 -0.16100851E+03
15.5238 -0.15990662E+03
16.5714 -0.15880025E+03
17.6190 -0.15768768E+03
18.6667 -0.15657148E+03
19.7143 -0.15544733E+03
20.7619 -0.15431814E+03
21.8095 -0.15318409E+03
22.8571 -0.15204195E+03
23.9048 -0.15089321E+03
24.9524 -0.14973823E+03
26.0000 -0.14857329E+03
```

Then run the following command:

- `tempering.R E.dat`

# Chapter 5

# Heisenberg Hamiltonian: `-ham` command

| | |
|---|---|
| input file | `seedname.inp1.mcin` |
| Output files | `seedname.ham, seedname.neigh, seedname.mcout, seedname.xsf` |
| run command | `mc.x -ham seedname` |

One characteristic of `ESpinS` is its ability for calculation of the coefficients of Heisenberg Hamiltonian for a specific spin configuration. One procedure to finding the effective spin Hamiltonian from *ab initio* based methods is mapping the total energies of some spin configurations to an appropriate spin Hamiltonian. This procedure gives the coupling constants of spin Hamiltonian for real material. `mc.x -ham` run can help to achieve the coefficients of Heisenberg Hamiltonian for each spin configuration.

## 5.1 Input file Keywords

Same as the `seedname.inp1.mcin` file.

## 5.2 Output files:

- `seedname.ham`

  The main output file of this step. The coefficient of Heisenberg Hamiltonian are written in this file.

- `seedname.neigh`

  Same as described in `-inp1` step.

- `seedname.xsf`

  Same as described in `-inp1` step.

- `seedname.mcout`

  Same as described in `-inp1` step.