

RMC_POT user guide for version 1.1

Orsolya Gereben

07.03.2011

I. The Reverse Monte Carlo algorithm

The Reverse Monte Carlo (RMC) computer simulation technique is capable of building 3-dimensional structural models in agreement with the experimental (mainly diffraction) data. Only a very brief description of the algorithm is given below, more details can be found elsewhere^{1,2,3}. Several computer version of RMC exist, one of the newest implementation (RMC++)⁴ was written in C++ (RMC++_new) and was parallelized and improved with new capabilities (RMC++_multi)⁵. These later software were used as the starting point of the present development, RMC_POT. Here the usage of the program will be described.

The starting point for RMC_POT was the unification of the codes RMC_new and RMC_multi codes. Now compiler options regulate the compilation of the code. Parallelization was achieved by using the POSIX standard (which can also be used under Windows) and can make execution faster on multiprocessor computers, but standard, consecutive compilation is also possible. New features were also added depending on the compilation, introducing the calculation of non-bonded potentials and handling flexible molecules kept together by forces, and local invariance calculation.

The RMC_POT program was written in C++, using only the standard statements as far as it was possible. There were however few cases, where operation system-specific statements have to be used, these are located in *altern.h* and *altern.cpp* files. The program was tested on Windows and Linux platform, if other platform is used, some alteration might be necessary!

A. The brief description of the RMC algorithm

The idea of the RMC algorithm is the following. We have a cubic simulation box containing fixed number of atoms having number density $\rho=N/V$. From this initial configuration the histogram can be calculated. The count in the k th histogram bin is the number of atoms between $k*dr$ and $(k+1)*dr$, where dr is the size of the histogram bin. Lets consider a multi-component system. Here partial functions have to be defined between the atom types. The number of the partials is $n_{types} \cdot (n_{types}+1)/2$ (where n_{types} is the number of atom types), so the mixed partials are just defined once. Order of the partials is row-continuous following the order of the elements in an upper diagonal matrix: (for example for 3 types: 1-1, 1-2, 1-3, 2-2, 2-3, 3-3). The order of the RMC atom types is defined by the order they are given in the configuration file. The partial pair correlation function (*ppcf*) which can be called radial distribution function (*prdf*) as well, can be calculated from the partial histograms by normalization:

$$g_{ij}(r) = \frac{n_j(r)}{4\pi r^2 \Delta r \rho_j} \quad \text{E 1}$$

where n_j is the number of atoms of type j at a distance between r and $r+\Delta r$ from a central atom of type i , averaged over all atoms of the central type i . ρ_j is the number density of the type j atoms.

The partial structure factor can be calculated by Fourier transformation from the *prdf*:

$$S_{ij}^{RMC} = S_{ij}(Q) - 1 = \frac{4\pi\rho}{Q} \int_0^{\infty} r g(r)_{ij} \sin Qr dr \quad \text{E 2}$$

The total structure factor can be given as

$$S_T^{RMC}(Q) = \sum_i \sum_j c_i c_j \overline{b_i b_j} S_{ij}(Q) \quad \text{E 3}$$

where Q is the scattering vector, c_i and c_j are the molar fractions and b_i and b_j s are the (neutron) scattering length of the components. For X-ray the formula is very similar, only instead of b the Q -dependent X-ray scattering coefficient $f(Q)$.

The calculated and the measured data are compared by calculating the χ^2 of the data set i :

$$\chi_i^2 = \frac{\sum_k (S_i^C(Q_k) - S_i^E(Q_k))^2}{\sigma_i^2} \quad \text{E 4}$$

where k is going through the points of the data set.

Then a chosen number of atoms are moved in the configuration. It has to be noted, then the more atoms are moved the less likely that the move will be accepted due to the possible violation of the hard sphere cutoff and the largest change in the calculated data, but if it is accepted then the configuration changes more rapidly mapping quicker the available configuration space. The new χ^2 is calculated, and if it is lower, than the old one, then the move is accepted. If not, then the move is only accepted with $\exp[-(\chi_{new}^2 - \chi_{old}^2)]$ probability. If there are too-close atoms in the configuration (atoms closer, than their hard sphere cutoff values), and the move is moved them above the hard sphere cutoff, or at least their distance increased, then the move accepted regardless the change in the χ^2 .

Repeating this procedure, we arrive at a configuration hopefully producing a calculated data very close the experimental one, and therefore we have a 3-dimensional configuration, which is the good representation of the real system.

There can be used neutron, X-ray and EXAFS experimental data for the fitting, and total or partial $g(r)$ functions as well, as many as we like. The EXAFS data calculation is a bit different from the others, will be discussed below.

B. The units used

The RMC program is traditionally using Ångstrom to measure distance, and Å⁻¹ for the inverse space. Ångstrom is displayed in file headers and printouts. Regardless of this, data can be given in other units, but the direct and inverse space data has to be consistent. If potential is used, then the non-bonded LJ sigma parameter has to be given in Angstrom in the *.dat* file, but due to the fact, that the GROMACS topology file format is adopted, in the *.top* and *.itp* files the GROMACS units, namely nm for distance has to be used in the topology and include topology files. Energy related parameters are given in kJ/mol.

C. The ‘tooclose’ atoms, moveout option

In RMC the atoms are represented as hard spheres having a hard sphere cutoff distance specified in the *.dat* file for each atom type pair (partials). It can happen, that the initial configuration does not satisfy the cutoff, and some of the atoms are closer to each other, than their cutoff distance. These are the ‘tooclose’ atoms. To facilitate the elimination of these atom pairs, the moveout option in the *.dat* file can be switched on. In this case the moved atoms are chosen in TOO_CLOSE_FRACTION*100 % of the moves from the ‘tooclose’ atoms, and not from the entire configuration. The TOO_CLOSE_FRACTION constant is located in the *units.h*, the default value is 0.5. If a ‘tooclose’ atoms is moved, than the move is accepted regardless the χ^2 , if the distance of the ‘tooclose’ atom pair(s) moved above the cutoff, or at least their distance increased. If the pair is not a ‘tooclose’ pair, or the moveout option is not used, then change of the χ^2 decides as it described in I.A., whether the move is accepted or not. When all the ‘tooclose’ atoms have been eliminated the simulation continues normally.

D. The molecular move

In the normal RMC algorithm one or more atoms are moved during a RMC step. However for molecular system it can be more advantageous to move a whole molecule together. This can be done by choosing the molecular move option in the *.dat* file. But in this case, the program has to be compiled supplying an adequate molecular constructor and a makemove function located in the *makemovecus.cpp* file. Obviously the molecular move for different molecules can be very different, so no standard molecular move can be written.

As a default the molecular move for the CCl₄ molecule is present in the program, but two other *makemovecus.cpp* files for C₂Cl₄ and H₂O are supplied in the Source code directory. If one of it is should be used, then the existing *makemovecus.cpp* has to be substituted with the new one. Even if normal atomic move is used some *makemovecus.cpp* file has to be present at compilation time, as the program is looking for it!

If molecular move for other type of molecules should be used, then it has to be written by the user following the guidelines visible form the existing molecular move files. The basic concept is, that at each move a molecule is randomly chosen for moving with a random distance to a random direction, and then depending on the molecular symmetry, some rotations and individual atomic moves are performed as well. The custom makemove function has to perform the molecular move, and some chosen parameters (like the maximum rotational angle, maximum atomic moves...) can be set in the *.cus file, which parameters can be changed without recompiling the program, see the example in CCL4_mol in the validation suite.

It is advisable to use the molecular move in case of simple molecules, as it results in quicker change in the system during the simulation, although the acceptance ration will most probably be lower.

E. Gridding of the simulation box

Gridding means that the simulation box is divided into a given number of sub-cells in each direction. If we know about each particle, in which grid cell it is located, and we want to calculate certain properties for only those particles that are not farther from the chosen particle than a given distance, then it is enough to calculate only for those particles, which are located in the same, and a given number of neighbouring grid cells. As checking, whether the move can be acceptable based on the satisfaction of the cutoff distances falls in this category, calculation can be quicker, if the grid-based cutoff check is performed before entering the lengthy histogram change calculation. The number of particles in the gridcell is regulated by this parameter. Based on this, the average number density of the sample and an additional safety parameter, SAFE_ADD located in the *units.h*, the program calculates the number of grid cells in each direction t. As due to inhomogeneity in the sample these numbers can vary, the program always checks, whether the maximum is not exceeded, before attempting to write into the arrays. If the maximum was exceeded, the program gives a message, resize the necessary arrays automatically, and continues execution. Gridding can only increase speed, if the number of particles in a grid cell is chosen adequately! It is preferable to set the desired maximum number of particles in a grid cell in the *.dat file to a relatively low value (5-10) depending of course on the system size to ensure at least 5 or preferably more grid cell in each direction. The actual number of grid cells is calculated by the program and printed on screen during the initialisation period, or can be found in the *.grid file. It has to be kept in mind, that even if the largest cutoff distance is smaller than the length of one grid cell, not just the cell containing the central particle, but all its closest neighbour cells are checked, as the central particle can be close to the edge. This way normally at least 27 cells are checked. Speed increase due to gridding can only be expected, if the number of grid cells to be checked is smaller, than the total number of grid cells!

F. The EXAFS data fitting

EXAFS stands for the extended X-ray Absorption Fine Structure spectrum.

XAS (X-ray absorption spectroscopy) is an element specific method to investigate the bond angles, bond lengths and coordination numbers. During the experiment the material under investigation is targeted with monochromatic X-ray beam (produced by synchrotron radiation). Some of the X-ray photons are absorbed by the material, and the rate of the absorption is measured versus the X-ray photon energy.

The X-ray absorption coefficient of a homogenous material can be given by

$$\mu(E) = \frac{1}{d} \ln \left(\frac{I_0(E)}{I(E)} \right) \quad \text{E 5}$$

where E is the photon energy and d is the thickness of the sample. Generally, the absorption of X-ray is decreasing with the increasing energy of the X-ray photons, but distinct spikes corresponding to a drastic increase of the absorption can be detected at some energy. These are the absorption edges, and they correspond to the binding energies of the inner-shell electrons (K, L, M..). As each chemical element has specific, well-defined binding energies, it is possible to select an energy range for the X-ray beam sweeping

specifically only an absorption edge (and the following) region of a selected element. This way information of the neighbourhood of the atoms of this chosen chemical element can be obtained.

XAS spectrum can be divided into different parts based on the energy range of the X-ray beam compared to the absorption edge, but there is no consensus in the literature neither in the number nor in the limiting energy values of the ranges. A possible division is given here.

- 1.) Directly before the absorption edge can be found the pre-edge region, where no ionisation occurs, only transition to higher, non-completely filled or empty orbits.
- 2.) In the edge region, where photon energy, $E \leq E^0 + 10$ eV (E^0 is the ionisation energy) the XANES (X-ray Near Edge Structure) is observable.
- 3.) NEXAFS (Near-Edge X-ray Absorption Fine Structure) region is between $E^0 + 10 < E \leq E^0 + 50$ eV.
- 4.) EXAFS region is where $E > E^0 + 50$ eV.

It has to be noted, that sometimes there is no division between the XANES and NEXAFS region, and the two acronyms are used as synonyms, although NEXAFS is usually used in connection with organic molecules and surfaces. The absorption edge itself sometimes is not considered to be in the XANES region, and the beginning of this region is set at $E > E^0 + 5$ eV. Sometimes the division between the XANES-NEXAFS and EXAFS region is set around 150 eV. From now on the acronym XANES will be used for the description of the XANES-NEXAFS range.

In the EXAFS region the kinetic energy of the photoelectron is higher and consequently the wave length and the scattering amplitude is smaller, so mainly single scattering of the photoelectron by the neighbouring atoms takes place.

The scattering amplitude and phase shift caused by the backscatterer depends on of the neighbour's type, and the phase and the amplitude of the backscattered wave depends on the inter-atomic distance between the absorber and the backscatterer as well.

After the ejection of the photoelectron the absorber atom will be in an excited state due to the core hole. Relaxation can occur when an electron occupying a higher energy level jump down into the core hole. The energy difference between the levels is either emitted as a fluorescence photon (only rarely occur), or absorbed by an other higher-level electron, which is emitted from the atom (Auger-electron).

The wave vector of the photoelectron (k) can be calculated from the X-ray photon energy, E and the ionization energy, E^0 by:

$$k = \sqrt{\frac{2m}{\hbar^2} (E - E^0)} \quad \text{E 6}$$

The oscillating absorption coefficient is normalized by the smooth atomic absorption background, μ^0 defining the EXAFS signal, $\chi(k)$:

$$\chi(k) = \frac{\mu(k) - \mu^0(k)}{\mu^0(k)} \quad \text{E 7}$$

F.1. RMC specific remarks

It has to be noted, that in RMC χ^2 is applied for denoting the difference between the experimental and calculated data, so it should not be confused with $\chi(k)$. For fitting EXAFS data in RMC, the values of k and $\chi(k)$ has to be given in the EXAFS data file as the input experimental EXAFS data (see III.E.4).

The EXAFS experimental data used in the fitting is denoted by $E(k)$ and it is calculated by

$$E_i^E(k_j) = \chi_i^E(k_j) \frac{k^n}{k_{\max}^n} \quad \text{E 8}$$

after reading the k and $\chi(k)$ data from the file. The weighting factor n has to be given in the *.dat file, and it is a small integer usually between 1 and 3. The reason for using this weighting is to compensate for the amplitude decay, fitting $E(k) = (k/k_{\max})^n \chi(k)$ with $n=1, 2, 3$ results in a physically more realistic configuration than in case of $n=0$, where k_{\max} is the largest k value of the data set.

The coefficients, $c(r,k)$ for the Fourier-transformation of the r -space information to k -space are r,k dependent, and has to be given in the coefficient file. The coefficients can be calculated for example by program [FEFF](#), see some information how to do it in the document [exafs_feff_rmc.pdf](#) writte by Pál Jóvári available on from the EXAFS page of the RMC_POT web site. Care has to be taken, that the same r -points has to be used during the coefficient calculation as used in the RMC simulation. Keep in mind, that in RMC always the middle of the bin is used for the given bin! It is possible to only a range of the r -dependent coefficients given in the coefficient file, this can be specified in the **.dat* file.

Different ranges can be specified for the different partials (all the possible atom type pairs of the edge particle type) for an absorption edge. The minimum and maximum indices of the r points (columns) for the partials are given in the **.dat* file, in the order $rmin_1, rmax_1, rmin_2, rmax_2, \dots, rmin_N, rmax_N$, where the indexing refers to the partials contributing to the given edge. For example for a 3-component system of As, Se, I for the As edge there are three partials (As-As, As-Se, As-I), for which the r index data has to follow in RMC order for the partials.

The calculation of $E^C(k)$ data is performed according to the following formula:

$$E_i^C(k) = \sum_{ip}^{np} \sum_{ir}^{nr} H(ip, ir) \frac{c(ir, k)}{N_e} \frac{k^n}{k_{max}^n} \quad \text{E 9}$$

where $H(ip, ir)$ is the value of the ip -th partial histogram ir -th histogram bin, N_e is the number of the particles with the absorption edge. The first sum is going through all the partials containing the particle type producing the absorption edge.

The squared difference, χ_i^2 is calculated as

$$\chi_i^2 = \frac{\sum_{j=1}^{N_{points}} (a_i E_i^E(k_j) + b_i - E_i^C(k_j))^2}{\sigma_i^2} \quad \text{E 10}$$

only using constant and multiplication factor for renormalizations.

G. The renormalization of the data

The $g(r)$, $S(Q)$, $F(Q)$ and $E(k)$ data sets can be renormalized to correct some errors due to the measurement. If renormalization is used, then not simply the squared difference between the calculated and experimental data set divided by sigma square of the data set is calculated during the χ^2 calculation, but the experimental data is renormalized the following way:

$$\chi_i^2 = \frac{\sum_{j=1}^{N_{pi}} (a_i A_{i,j}^E + b_i + c_i x_{i,j} + d_i x_{i,j}^2 - A_{i,j}^C)^2}{\sigma_i^2} \quad \text{E 11}$$

where N_{pi} is the number of data points for data set i , $A_{i,j}^E$ is the experimental, $A_{i,j}^C$ is the calculated data, for neutron and X-ray fitting $x_{ij}=Q_{i,j}$ is the j th data point of the i th set, a_i , b_i , c_i and d_i are the renormalization coefficients, controlled by the vary amplitude, constant, linear and quadratic switch in the **.dat* file, and displayed during the simulation and written to the **.fit* file. The same formula is used for $E(k)$ fitting, except that $x_{ij}=k_{ij}$ and only the a_i and b_i coefficients used and featured in the **.dat* file, and for $g(r)$ fitting $x_{ij}=k_{ij}$ where only a_i exists. The program determines the coefficients, where the difference between the calculated and the experimental sets is the minimum. (for the sake of clarity see the Quadratic background correction within RMC by László Teimleitner on the RMC web site).

H. R_w calculation

The R_w value used usually in crystallography to characterize the goodness of the fit is also calculated for the $g(r)$, $S(Q)$, $F(Q)$ and $E(k)$ data series, according to the formula:

$$R_{w,i} = \sqrt{\frac{\sigma_i^2 \chi_i^2}{\sum_{j=1}^{Np_i} (a_i A_{i,j}^E + b_i + c_i x_{i,j} + d_i x_{i,j}^2)^2}}$$

E 12

where i means the i th data series, $x=r$, $b=c=d=0$ for the $g(r)$ data, $x=Q$ for the X-ray and neutron data, $x=k$, $c=d=0$ for EXAFS data, A_{ij}^E is the experimental. R_w is printed on screen together χ^2 and given in the *.hst file for the end of the run.

I. The cosine distribution of bond angles constraint

The angles between the bonds of three atom types can be given as a constraint in the *.dat file (see III.C. and III.F for the syntactic). Any number of constraint can be given, and RMC_POT makes a distinction between the types of the neighbours, so for a two component system, if the middle atom denotes the central type, then for a type A central A-A-A, B-A-A and B-A-B type different cosine distributions can be calculated (the A-A-B would be the same as B-A-A.) Take care, that in the *.dat file after the mode indicator first the central then the two neighbour types has to be given!

The constraint can work in 4 different mode depending on the method given:

- 0: The theoretical distribution is calculated as a step function;
- 1: The theoretical distribution is calculated as a Gaussian distribution.
- 2: No angles are required in a certain range
- 3: Experimental distribution should be read from a file.

In case of method 0-2 the spacing of the cosine distribution of bond angles histogram is determined by the $\text{dcos}(\theta)$ parameter given in the *.dat file.

A constraint can be "positive", which means, that bond angles are needed in the given region, and "negative" meaning that angles are not wanted in the given region. Positive constraint can be set up by specifying either 0 for the calculation method indicating a step function, or 1 meaning Gaussian for the shape of the constraint. Step function means having uniform distribution with integral 1 between angle -wcontrol \rightarrow angle + wcontrol, (both given in degrees), 0 otherwise. Gaussian indicates a normal distribution with integral 1 having the maximum at angle (given in degree), which is converted to radians and the cosine of it is calculated. This $\cos(\theta)$ value will give the peak position, and the width is controlled by $\text{sigma}=\text{wcontrol}$, where wcontrol is used as it is given in the *.dat file, so it is a $\text{dcos}(\theta)$ value. So it is a good approximation, that the theoretical distribution will span the $\cos(\theta)-3*\text{wcontrol} \rightarrow \cos(\theta)+3*\text{wcontrol}$ range in $\cos(\theta)$ units, which corresponds to the confidence interval $\sigma=3$ (99.7 % of the data is inside the range). The size of the confidence interval is regulated by the CONF_INT constant in *units.h*, the default is 3.

In these cases the theoretical distribution is calculated for all the bins spanning the range $-1 < \cos(\theta) < 1$, so only one positive constraint (desired angle) can be meaningfully given for a neighbour1-central-neighbour2 triplet, as more than one constraint would ruin each others effect.

Negative constraints, that no angles desired in a given region can be set up using calculation method=2. In this case the constraint is for the given region and not for all the bins, as it could interfere otherwise with a "positive" constraint for the same particle types. A normal distribution curve similarly to method=1 is calculated from that part of $-\text{CONF_INT}*\text{wcontrol} \rightarrow +\text{CONF_INT}*\text{wcontrol}$, which is in the $-1 \rightarrow +1$ range, centred on the not desired angle. The calculated curve is used during the χ^2 calculation to provide an additional, $\cos(\theta)$ -dependent weight by multiplying the calculated distribution with it, and calculating the product's squared difference from 0. Make sure, that the interval of a negative constraint does not overlap with a positive constraint for the same particle triplet!

In case of method 3 the experimental distribution is read from a file. The $\cos(\theta)$ values have to span the whole $-1 \rightarrow 1$ interval, has to denote the middle of the bins, and have to be equidistant! See the file structure in chapter III.F. This constraint can be used together with other type of cosine distribution of bond angles constraints, and in this case, the spacing for the experimental data does not have to be the same as for the other constraints!

J. Coordination number constraint

It was discussed in the literature, that RMC tends to produce the most disordered structure consistent with experimental data. So it can be a good idea to use some extra knowledge about the structure, as for example the preferred coordination number. This means, that we can specify that around an atom of the central type between r_{min} and r_{max} what should be the preferred coordination number of the atoms of the neighbour type for the desired fraction of the central atoms. It is possible to have more, than one neighbour type for the same constraint. For example in case of 3 atom types it is possible to define a constraint that the central atom of type1 should have 3 atoms belonging to type2 and type3 between $r_{min}[type2] \rightarrow r_{max}[type2]$ and $r_{min}[type3] \rightarrow r_{max}[type3]$ respectively, which means, that there are several possibilities satisfying this constraint (type2-type2-type2, type2-type2-type3, type2-type3-type3 and type3-type3-type3). Coordination number constraint contributes to the χ^2 according to the formula below:

$$\chi^2 = \sum_m^{n_{cc}} \frac{\left(\frac{N_m^S}{N_m^C} - N_m^f \right)^2}{\sigma_m^2} \quad \text{E 13}$$

where σ_m , is the respective standard deviations, n_{cc} is the number of coordination number constraint, N_m^C is the number of central atoms and N_m^S is the number of atoms satisfying the m th coordination constraint, N_m^f is the desired fraction of the m th coordination constraint.

There can be more, than one constraints specified especially in a multi-components system. If multiple constraints are necessary between the same type of atoms, which only differ in the desired coordination number and fraction, then only one constraint has to be specified but with more than one sub-constraint. Each sub-constraint will have their own sigma (see the III.C how to specify the constraint). Using sub-constraints instead of using separate coordination constraints save memory and time, so it is highly advisable to use them.

K. Average coordination constraint

Sometimes it is better to constrain not the coordination number of each individual central atom, but the average coordination number of the neighbour type atoms around the central type atoms between r_{min} and r_{max} . The χ^2 contribution is calculated as

$$\chi^2 = \sum_n^{n_{ac}} \frac{\left(\frac{N_n^{sn}}{N_n^C} - N_n^d \right)^2}{\sigma_n^2} \quad \text{E 14}$$

where σ_n , is the respective standard deviations, n_{ac} is the number of average coordination number constraint, N_n^C is the number of central atoms and N_n^{sn} is the total coordination number for the n th average coordination constraint, N_n^d is the desired average coordination number. See the III.C how to specify the constraint.

L. The Fixed Neighbour Constraint (FNC)

The FNC is used to keep molecules together in a relatively rigid way. For this a *.fnc file (see the structure later in III.G) has to be supplied, where for each atom of the configuration the number of neighbours, their indices, and their FNC constraint type has to be given. For each FNC constraint type the minimum and maximum distances between which the distance of the atom pairs belonging to this constraint should stay is specified in the header of the file. In the original implementation the simulation can only be started, if the constrained distances are already in the range, which is not entirely practical. This corresponds to the FNC switch fnc=1 in the older an in this new version. See about the new possibilities later.

II. Description of the new features

A. The improved Fixed Neighbour Constraint (FNC)

As it was mentioned before, the FNC is used to keep molecules together in a relatively rigid way keeping the FNC constrained pairs in the given range. To remove the constraint, that the initial configuration should satisfy the FNC constraint, two new possibilities were included. To be able to start the simulation, even if not all the constrained pairs are in range by resetting the maximum and minimum constrained FNC distances can be achieved by using `fnc=2`. If we do not want to widen the FNC range, then `fnc=3` could be used, in this case at the beginning warning is generated about the out-of-range pairs, and only those moves are accepted, where the new distance is closer to or in the desired range, than the old one, hopefully eliminating all the out-of-range pairs.

There is an other possibility for keeping molecules together, if the program was compiled with `_POTENTIAL` compiler option, in which case using `fnc=4` means flexible, MD-like molecules, kept together by forces (see II.C).

B. The local invariance

The local invariance was introduced based on the work of M. J. Cliffe et al⁶. In their work an additional cost term increasing with the deviation of the local $g(r)$ of each atom from the experimental $g(r)$ was used. For practical reasons a coarse-grained approach was used, instead of storing all the pair distances, a local histogram is calculated and stored for each atom. Storing all the distances even for a system consisting of few thousand atoms would have too large memory requirement, and for a larger system sometimes used in RMC would have been completely impossible to do. There are two possibilities for the calculation of the local χ^2 -contribution. The first one is the bin-based approach, where the deviation of the local histogram from the normal, average histogram is calculated according to E 15:

$$\chi_{loc}^2 = \sum_{a=1}^{N_t} \sum_{i=1}^{N(a)} \sum_{b=1}^{N_t} \sum_{j \neq i}^{N(b)} \sum_{k=0}^{N_{bin}} \frac{\left(H_{ab}^i(k) - \overline{H_{ab}(k)} \right)^2}{\sigma^2 dV(k)} \quad \text{E 15}$$

where N_t is the number of atom types, $N(a)$ and $N(b)$ are the number of atoms of type a and b , $N_{bin} = \text{int}((loc_ratio_2 - loc_ratio_1) * d / 2 / dr_{loc}) + 1$ is the number of local histogram bins involved in the calculation where d is the size of the simulation box in Å, dr_{loc} is the width of the local histogram bin in Å, loc_ratio_1 and loc_ratio_2 are the minimum and maximum distances for the local histogram calculation in reduced units. $H_{ab}^i(k)$ is the k th bin of the i th atom's local histogram for the ab type partial, $\overline{H_{ab}(k)}$ is the average histogram for the ab partial's k th bin, $dV(k)$ is the volume of the k th histogram bin and σ^2 is the weight parameter.

The other approach is distance based. This is closer to the original implementation of Cliffe. The χ^2 is calculated as the squared difference of the distance (which in this coarse grained approach is the middle value of the histogram bin which it can be found) of the j th neighbour atom of type b of a central atom i from type a compared to the average distance of the j th neighbour of type b for all the central atoms of type a . The j th neighbour of a central atom means the one with the j th largest distance among all the b -type neighbours. In case of the distance-based approach it is possible to divide the interval to several consecutive parts with their own weight parameters to be able to make the contributions of the different intervals separate. The χ^2 contribution of the n th interval is calculated as

$$\chi_{loc_n}^2 = \sum_{a=1}^{N_t} \sum_{i=1}^{N(a)} \sum_{b=1}^{N_t} \sum_{j=J_n(b)}^{J_{n+1}(b)} \frac{\left(I_{ab}^i(j) - \overline{I_{ab}(j)} \right)^2 dr_{loc}^2}{\sigma_n^2 dV(\overline{I_{ab}(j)})} \quad \text{E 16}$$

where j is running through the neighbours of type b of atom i of type a arranged according to increasing distance from i starting with index $J_n(b)=\text{int}(\text{loc_ratio}_n*N(b))$ and ending with $J_{n+1}(b)=\text{int}(\text{loc_ratio}_{n+1}*N(b))-E$ where loc_ratio_n and loc_ratio_{n+1} are the ratio of neighbour atoms for the interval boundaries, and E is 0 for the last interval and 1 otherwise. $I_{ab}^i(j)$ is the index of the local histogram bin in which the j -th type b neighbour of central atom i of type a is found, and $\overline{I_{ab}}(j)$ is the average bin index for the j -th neighbours of type b for all the central atoms of type a , dr_{loc} is the width of the local histogram bin in Å, $dV(\overline{I_{ab}}(j))$ is the volume of the average bin and σ^2 is the weight parameter. In this case the local histogram is calculated and stored for the maximal range of 1.732 reduced distance, as it is not known, how much the local atomic environments differ, but only a portion of this regulated by loc_ratio_1 and $\text{loc_ratio}_{N_{loc}+1}$ is used for the χ^2 calculation.

The bin size of the local histogram calculation can be different from the normal histogram used for the calculation of the data sets. As statistic is not an issue here, it can be chosen finer, than the normal histogram. The smaller the bin size is, the most this coarse grain approach will mimic the original idea. The limiting factor for the fines is the larger memory requirement. The bin size of the local histogram is given in the **.dat* file.

This necessitates the storage of the local histogram ($Nt \cdot Nbin$ elements for each atom), which can have large memory requirements for larger systems. The local histogram is saved to the **.lhgm* file, saving and loading can take noticeable time, sometimes more, than recalculation!

The local invariance calculation is only performed, if the code is compiled with the `_LOCAL_INV` compiler option. In this case from a new line after the number of threads to use first the number of local invariance intervals are read, then the weight parameters for the local invariance intervals. After that the calculation mode can be given, and then $N_{loc}+1$ real values for the atom ratios to use (where N_{loc} is the number of intervals), loc_ratio_n . In case of the bin-based calculation (where only one interval can exist) the two real values mean the starting and limiting reduced distances, for which the local invariance is calculated, the histograms are stored only between them. In case of the distance-based approach, the n . and $n+1$. ratio sets the boundaries of the n th interval they represent the fraction of neighbour atoms (the same fraction value resulting in different atom number for each type according to the different number of atoms/type) to begin and end the local invariance calculation with. Check the actual boundaries in the **.hst* file, and in the screen output. So in both case it is possible to calculate the local invariance only for a portion of the range the normal histogram is calculated!

Only compile the code with this option, if you really want to calculate the local invariance, as it is much slower, than the normal RMC run!

C. The potential-related features

The handling of the potential-related interactions in RMC is performed very similarly to a molecular dynamics (MD) simulation.

Both non-bonded and bonded potential can be calculated, if the `_POTENTIAL` compiler option is used. For the non-bonded potential calculation the potential switch in the **.dat* file has to be set to 1 instead of 0. The non-bonded potential can be calculated for both atomic and molecular systems. In each case a GROMACS-type topology (**.top*) and if necessary, additional include topology (**.itp*) files have to be supplied, with some additional parameters, discussed later.

The potential energy of the system will make a contribution to the χ^2 , so guiding the system to reach an energetically more favourable state. More precisely, if non-bonded interactions are present contributes to a second, potential-related χ_P^2 . The contribution is defined, as the V_{NB}/σ_{NB}^2 and so forth can be negative, this is the reason, that it is not added to the normal χ^2 , as it could wipe out the deviation of the data set. It is optional, whether the whole systems' potential will make only one contribution or the potential of the RMC partials of a multi-component system can each make their own contribution with their individual sigma value to χ_P^2 , so weighted separately from one another. This is decided by the weight mode parameter following the potential switch. In this case a separate sigma value for all the partials has to be given in the **.dat* file.

After it is checked, that the hard sphere cutoff distances are not violated, the histogram and together the non-bonded potential is updated.

The non-bonded interaction consists of a dispersion and repulsion term making up the van der Waals (vdW) term, and a Coulomb term.

Non-bonded interactions are calculated up to a certain cutoff radius, which not necessarily coincide with the r_{max} value of the histogram calculation. The cutoffs can be set in the **.dat* file separately for the vdW and the Coulomb interaction.

Bonded interactions can be calculated for molecules. This feature was essentially introduced to provide an alternative, more flexible way to keep molecules together instead of the rather rigid FNC constraint. Hopefully it will provide a physically more realistic distribution of the bond lengths, angles and dihedral angles. These interactions can be calculated only if the program is compiled with the `_POTENTIAL` compiler option, and the Fixed Neighbour Constraint (FNC) switch is set to 4, so the normal FNC constraint and the new flexible molecule handling cannot be used together. Furthermore, as the same class is handling the potential and the conventional FNC in the program, normal FNC (1, 2,3) cannot be used, together with non-bonded potential calculation. The molecular topology has to be given in the **.top* (or **.itp*) file.

The bonded interactions consist of bond stretching, angle bending and three differently defined dihedral potentials. The bonded interactions give a contribution to χ^2 , the larger their energy is the larger the χ^2 so guiding the system to reach configurations closer to the favourable equilibrium value. If non-bonded interaction is present, then the bonded interaction contribution is added to the χ_p^2 forming a purely energy-based contribution, if not, then to the normal χ^2 , as their contribution cannot be negative. Their contribution is calculated similarly to the non-bonded interaction ($\chi_B^2 = V_B/\sigma_B^2$). It is possible to have only one contribution coming from the given interaction type (for example bonds), if for each of the different bond types the same sigma value is given in the **.top* (or **.itp*) file. If different sigma values are given for the different bond types, then each will give its own contribution to χ^2 and displayed separately.

Although in GROMACS the interaction parameters like the LJ sigma and epsilon or the force constant and equilibrium values for the bonded interactions come usually from data base files and does not have to be specified in the topology for RMC they have to be specified directly in the topology file. This way other force field parameters can be used easily, but it have to be kept in mind, that the parameters have to be consistent.

If non-bonded interactions are present, then first the χ_p^2 is calculated from the potential related contributions, and based on this it is decided, whether the move is acceptable or not. If χ_p^2 decreased, then the move is accepted, if not then it is accepted with $\exp[-(\chi_{Pnew}^2 - \chi_{Pold}^2)]$ probability. If there are “tooclose” atoms, and the move decreased its number or at least increased the distance between “tooclose” atoms, then the move is accepted regardless the χ_p^2 change, similarly as in case of the normal χ^2 based acceptance procedure. If the move passed this test, then the data set and constraint calculations and the normal χ^2 calculation follows, and a second acceptance test based on the normal χ^2 .

C.1. The interaction functions

C.1.1. Non-bonded interactions

The non-bonded interactions potential consists of a van der Waals term and a Coulomb term. Presently only Lennard-Jones 6-N potential is included for the van der Waals term, N can be specified in the **.dat* file, if not given, the default 12 is used.

The potentials are calculated according to the formulas given below:

$$V_{LJ}(r_{ij}) = 4\varepsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r_{ij}} \right)^N - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right) = \frac{C_{ij}^{(N)}}{r_{ij}^N} - \frac{C_{ij}^{(6)}}{r_{ij}^6} \quad \text{E 17}$$

$$V_{Coulomb}(r_{ij}) = f \frac{q_1 q_2}{r_{ij}}$$

where ε_{ij} (kJ/mol) and σ_{ij} (Å) or alternatively $C_{ij}^{(N)} = 4\varepsilon_{ij} \sigma_{ij}^N$ are the LJ parameters, which should be supplied for all different GROMACS-type of atoms in the **.dat* file.

C.1.2. Bonded-interactions

For molecules the bond stretching potential between atom i and j can be calculated as a harmonic potential:

$$V_b(r_{ij}) = \frac{1}{2} k_{ij}^b (r_{ij} - b_{ij})^2 \quad \text{E 18}$$

where the k_{ij}^b is the force constant in kJ/mol, b is the equilibrium distance in nm should be given in the topology..

The harmonic angle bending potential for atoms i, j and k (j is in the middle) is defined as:

$$V_a(\theta_{ijk}) = \frac{1}{2} k_{ijk}^a (\theta_{ijk} - \theta_{ijk}^0)^2 \quad \text{E 19}$$

where the k_{ijk}^a is force constant in kJ/mol, θ^0 is the equilibrium angle in degree.

Dihedral angle can be proper or improper. Proper dihedral angle is defined according to the IUPAC/IUB convention as the angle between the planes ijk and jkl (see Figure 1) with zero corresponding to the cis conformation.

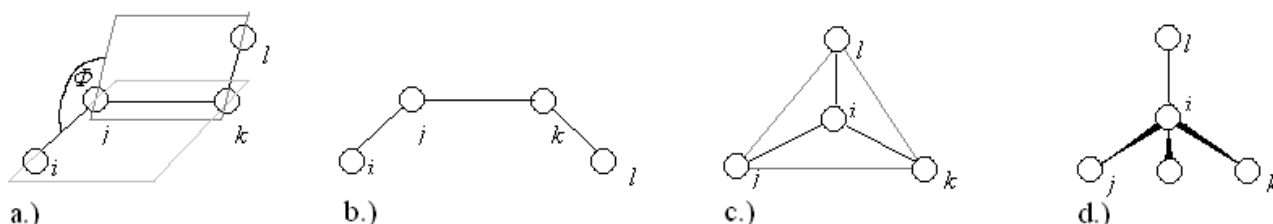


Figure 1: a.) Proper dihedral angle, b.) improper dihedral for rings, c.) planar group and d.) chiral centre.

Improper dihedrals are used to keep planar groups planar, or to prevent chiral groups to transform to their mirror image.

Three different dihedral angle definition is possible based on GROMACS, the periodic, harmonic, and Ryckaert-Bellemans dihedral definition. It depends on the force field, how the proper and improper dihedrals are calculated.

Usually periodic dihedral is used to handle the proper dihedral interaction, and calculated as:

$$V_{pd}(\Phi_{ijkl}) = k^\Phi (1 + \cos(n\Phi_{ijkl} - \Phi^s)) \quad \text{E 20}$$

where k^Φ is the force constant in kJ/mol, n is the multiplicity and Φ^s is the equilibrium angle in degree.

Improper dihedrals are defined as harmonic dihedrals, and calculated as:

$$V_{hd}(\xi_{ijkl}) = \frac{1}{2} k_{ijkl}^\xi (\xi_{ijkl} - \xi^0)^2 \quad \text{E 21}$$

where k^ξ is the force constant in kJ/mol/rad², and ξ^0 is the equilibrium angle value.

For alkanes the Ryckaert-Bellemans dihedral definition is often used:

$$V_{RBd}(\psi_{ijkl}) = \sum_{n=0}^5 C_n (\cos(\psi))^n \quad \text{E 22}$$

where $\psi = \phi - 180^\circ$ 0 corresponding to the trans conformation, and C_n in kJ/mol are the RB constants.

The OPLS force field is using Fourier dihedrals with $\phi=0$ corresponding to cis:

$$V_{Fd}(\phi_{ijkl}) = \frac{1}{2} [F_1(1 + \cos(\phi)) + F_2(1 - \cos(2\phi)) + F_3(1 + \cos(3\phi)) + F_4(1 - \cos(4\phi))] \quad \text{E 23}$$

which can be converted into RB dihedrals, as

$$C_0 = 1/2(F_1 + F_3) + F_2$$

$$C_1 = -1/2(F_1 - 3F_3)$$

$$C_2 = 4F_4 - F_2$$

$$C_3 = -2F_3$$

$$C_4 = -4F_4$$

$$C_5 = 0$$

so the literature values has to be treated accordingly. The program internally uses RB dihedrals.

In case of the OPLS force field improper dihedrals are not used for chiral atoms, and for planar groups the periodic E 20 function is used as improper dihedral angle function.

C.2. The description of the GROMACS-type topology, and the connection to RMC

The same format text type RMC configuration file **.cfg* and/or the binary format **.bcf* files are used, as before. Here the concept is that for a multi-component system (which can be atomic or can contain molecules) usually one RMC-type depicts a chemical element, so there are as many RMC-types, as the number of different chemical elements. The coordinates of the atoms belonging to the same RMC-type are making up a block, and these blocks follow each other consecutively in the order the types are given in the file header. The partial *ppcf*-s are calculated based on this type selection. When a bit more complicated system is used, (for example propanol), it might be necessary to use more than one RMC-type for the same chemical element depending on the location of the atoms in the molecule to be able to hold the molecule together properly with a Fixed Neighbour Constraint (FNC).

In molecular dynamics simulation programs (as GROMACS) a lot of different type (referred to as GROMACS-types) are used for the same chemical element depending on the environment of the atom, as the non-bonded parameters are strongly dependent on the atomic environment.

Even in that case we might want to use only one RMC-type for a chemical element, as the neutron or X-ray diffraction properties are the same. Therefore, the rendering of the GROMACS-atom indices to the RMC atom indices has to be given in the topology file. The parameters for the non-bonded interactions for each GROMACS-types has to be given in the **.dat* file, and the weight parameter for the potential-based χ^2 contribution as well.

As we use the GROMACS molecular dynamics suite, the same format topology files for describing the molecules are used in RMC_POT with some additional parameters, but the additions are always given at the end of a line, so this extended topology file can be used for both GROMACS and RMC, which simplifies our work. It might be useful to consult the GROMACS manual 4.0, which is available from the GROMACS web site <http://www.gromacs.org/Documentation/Manual>, as in this documents only the basics of the topology is given. It might be advisable to start a simple MD GROMACS simulation with your newly created topology, as GROMACS performs more extensive checks regarding the integrity of the topology file than RMC does!

C.2.1. The concept of charge groups

Non-bonded interactions are calculated up to a cutoff value. Therefore for molecules containing atoms with partial charges it would create a problem, if some of the atoms would fall below cutoff and other above it. This is why nearby atoms are rendered to charge groups. Preferably the net charge of the charge group should be 0, but this not always possible. For example the three atoms of a water molecule belong to the same charge group with 0 net charge. For larger molecules it is not possible to render all the atoms of the molecule to the same charge group, here a chemically meaningful part of a molecule form a charge group, as for example a methyl (-CH₃) or methylene (-CH₂) group. The atoms of a charge group are not separated from each other by the cutoff during non-bonded interaction calculation, the geometric centre of the charge group decides for all the atoms of the group, whether it is inside or outside cutoff. The atoms of a charge group has to follow each other in the molecule's **.top* or **.itp* file having the same charge group number.

C.2.2. The exclusions

The interactions of the atoms connected to each other by 1-2 chemical bonds are handled quantum-mechanically by the bond stretching (1 bond) or angle bending (2 bonds). Therefore they have to be excluded from the normal non-bonded interaction. This is achieved by setting the **number of exculsions** parameter in the topology file, and handled by RMC exactly the same way, as by GROMACS.

C.2.3. The 1-4 interactions

For the atoms separated by 3 bonds (1-4 neighbours) usually the normal non-bonded interactions are too strong. The handling of these pairs depends on the parameterisation of the force field, but usually an other set of parameters are used for the 1-4 interactions (as in GROMOS force field) or the normal parameters are scaled as in OPLS. Whether to use 1-4 interactions together with dihedral angle interaction also depend on the applied force field and the dihedral interaction type, for example for GROMOS force field if the periodic dihedral potential is used, then a special 1-4 interaction has to be included (the LJ1-4 parameters

has to be given for RMC in the topology file after the [pairs] directive), if RB dihedrals are used for alkanes, then no 1-4 interaction has to be calculated at all. In either case, the 1-4 interaction has to be excluded from normal non-bonded interaction calculation by setting the number of exclusions to 3. If OPLS-aa force field is used, then all the 1-4 interactions (both the vdW and the Coulomb) is scaled down by a factor of 2, so no additional parameters are required and together with this RB dihedrals has to be used. The 1-4 atom pair indices has to be given after the [pairs] directive, if 1-4 interaction has to be calculated either by using special parameters or by scaling!

If scaling is used, then it can be given as the vdW and Coulomb fudge in the *.dat file for RMC, for OPLS it has to be 0.5 for both Lennard-Jones and Coulomb interaction.

C.2.4. The combination rule, and the non-bonded parameters for the mixed partials

There are different ways the non-bonded parameters can be specified similarly to GROMACS. For the LJ interaction always two parameters are required for an atom type, (denoted generally by V and W), which can be either the $\sigma - \epsilon$ pair, or the $C^{(6)} - C^{(N)}$. The parameters for the mixed partials can either be directly specified if their calculation does not follow any rule, or calculated according to some formulas, where either arithmetic or geometric averages can be used. The combination rule given in the *.dat file for RMC decide about the meaning of the given parameters, and how the vdW parameters for the mixed partials has to be calculated, if they are not supplied. These are the following options:

Combination rule	Supply V, W	Formula
0	σ_{ij} [Å], ϵ_{ij} [kJ/mol]	given for all the partials
1	$C_i^{(6)} = 4\epsilon_i \sigma_i^6$ [kJ·Å ⁶ /mol] $C_i^{(N)} = 4\epsilon_i \sigma_i^N$ [kJ·Å ^N /mol]	$C_{ij}^{(6)} = (C_i^{(6)} C_j^{(6)})^{1/2}$ $C_{ij}^{(N)} = (C_i^{(N)} C_j^{(N)})^{1/2}$
2	σ_i [Å], ϵ_i [kJ/mol]	$\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j)$ $\epsilon_{ij} = (\epsilon_i \epsilon_j)^{1/2}$
3	σ_i [Å], ϵ_i [kJ/mol]	$\sigma_{ij} = (\sigma_i \sigma_j)^{1/2}$ $\epsilon_{ij} = (\epsilon_i \epsilon_j)^{1/2}$

Combination rule 1-3 are the same in GROMACS (only the unit for distance is nm), 0 only exist in RMC, in this case the σ and ϵ values for all the GROMACS type partials has to be given.

C.2.5. The topology (*.top) and include topology (*.itp) file

The topology file contains the information about the system, how many different molecule types can be found, and how many molecules exist for each type. This has to be in the *.top file. The definition of the different molecule types can go into their own *.itp file, or if there is just one molecule type into the *.top file itself. The *.itp files can be included into the *.top file with #include itpfilename.itp. All the C-Preprocessor directives can be used in a topology file for GROMACS, but only the following ones are recognised by RMC: #include, #ifdef, #else, #endif. The last three can help to create topology files, where the active content can be regulated by passing or not passing compiler options through the *.dat file. For example if a part of a topology file is

```
[ atoms ]
1  opls_209  1  DMeS  CA      1  -0.013      ;      1      2
#ifdef _DEUTERIUM
2  opls_140  1  DMeS  DA1     1  0.06  2.014102  ;      31     34
3  opls_140  1  DMeS  DA2     1  0.06  2.014102  ;      32     35
4  opls_140  1  DMeS  DA3     1  0.06  2.014102  ;      33     36
#else
2  opls_140  1  DMeS  HA1     1  0.06      ;      31     34
3  opls_140  1  DMeS  HA2     1  0.06      ;      32     35
4  opls_140  1  DMeS  HA3     1  0.06      ;      33     36
```

#endif

If `_DEUTERIUM` is passed to the topology file, then the blue part using deuterium atoms are used, if not then the green part using normal hydrogen atoms. Arguments can be passed to the topology file after the `FNC` switch preceded with `-D` like `-D_DEUTERIUM`.

Topology file has to be given even for atomic or ionic system, in this case only the `[moleculetype]` and `[atoms]` directive has to be used for the molecule description. The available directives are discussed below.

C.2.6. The GROMACS directives

GROMACS topology files are directive driven, the directives has to be specified in square brackets. The order of some directives is important, while others can be arbitrary. The lines following a directive are supposed to be of the type indicated by the directive and treated accordingly, while other directive is found. Empty lines can be anywhere, and they are ignored. RMC uses only some of the existing GROMACS topology directives, the others are ignored. It has to be noted, that the semicolon is treated by GROMACS as text qualifier, and everything after it is ignored, so additional information which is required by RMC are sometimes given after that, as RMC, if expecting an additional entry for that line ignores the semicolon, but at the beginning of a line treats it as text qualifier, and do not try reading after it. Comment lines beginning with text qualifier can be anywhere in the file.

The directives recognised by RMC are the following:

Recognised GROMACS directives	Format
<code>[moleculetype]</code>	molecule name, number of exclusion, (<u>index offset</u>)
<code>[atoms]</code>	serial number, type , residue number, residue name, atom name, charge group, charge (e), mass, <u>first index, second index</u>
<code>[pairs]</code>	atom index i, atom index j, type, {$V^{(cr)}$}, {$W^{(cr)}$}
<code>[bonds]</code>	atom index i, atom index j, type, b (nm), k^b (kJ/mol), (<u>sigma</u>)
<code>[angles]</code>	atom index i, atom index j, atom index k, type, θ^0 (degree), k^a (kJ/mol), (<u>sigma</u>)
<code>[dihedrals]</code>	atom index i, atom index j, atom index k, atom index l, 1, Φ^s (degree), k^ϕ (kJ/mol), multiplicity, (<u>sigma</u>)
<code>[dihedrals]</code>	atom index i, atom index j, atom index k, atom index l, 2, ξ^d (degree), k^ξ (kJ/mol/rad²), (<u>sigma</u>)
<code>[dihedrals]</code>	atom index i, atom index j, atom index k, atom index l, 3, $C_0, C_1, C_2, C_3, C_4, C_5$, (kJ/mol), (<u>sigma</u>)
<code>[system]</code>	system name
<code>[molecules]</code>	molecule name, number of molecules

Bold underlined parameters are only used by RMC, has to be preceded by a semicolon, if the topology is used for GROMACS. If it is in (), like the sigma parameter, then has to be given only for the occurrence of the given interaction type. For the index offset it is necessary only for the second and higher indexed molecule types, if there is more, than one molecule type.

Bold parameters are used both by GROAMCS and RMC.

Parameters with normal letters are only used by GROMACS, but value matching the data type of the parameter has t be given for RMC as well, only the value does not matter.

Parameters in {} are needed only depending on the force field type, for OPLS-aa it is not required.

Some of the parameters are explained below:

`[moleculetype]`

This can be in the `*.itp` file.

- The **molecule name**'s size is set by the NAME_SIZE define statement in the *units.h*, and the default value is 50, in the GROMACS manual the maximum size for GROMACS was not found, but 50 characters is accepted as well.
- Atoms separated by the number of bonds(= **number of exclusions**) are excluded from the normal LJ interactions.
- index offset is the total number of atoms before this molecule type according to the GROMACS configurations. This way the same **.itp* with the same first and second index can be used either it is alone in the system or together with other molecule types. First and second index has to be changed only if the number of molecule is changed.

[atoms]

This can be in the **.itp* file.

- **type** is the GROMACS atom type of the atom, for RMC it can be anything, but has to be the same for the same type of atoms;
- **atom name** is the unique identifier of the atom max 5 character;
- index of the **charge group**;
- first index is the RMC index of the atom in the first occurrence of the molecule;
- second index is the RMC index of the atom in the second occurrence of the molecule;

[pairs]

This can be in the **.itp* file.

Here the atom pairs for which 1-4 interaction has to be calculated can be given:

- **atom index i** and **atom index j** are the GROMACS serial indices of the atoms in the topology file;
- **type** is the GROMACS interaction type, has to be set to 1 for both RMC and GROMACS for the normal 1-4 interaction discussed above;
- $V^{(cr)}$, $W^{(cr)}$ are **vdW parameters** according to the combination rule, see C.2.4.

There are usually predefined bonds, angles, dihedrals between given type of atoms, check the force field's *ff*bon.itp* file for the parameters.

[bonds]

This can be in the **.itp* file.

- **atom index i**, **atom index j** are the GROMACS serial indices of the atoms in the topology file;
- **type** is the GROMACS interaction type, should be 1 for harmonic bond potential, RMC try to proceed, even it is not;
- **b (nm)** is the equilibrium distance, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;
- **k^b (kJ/mol)** is the force constant, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;
- **sigma** parameter to weight the contribution of the bond type for χ^2 calculation, has to be given only for the first occurrence of a bond type; Bonds with the same equilibrium bond and force constant belong to a given bond type.

[angles]

This can be in the **.itp* file.

- **atom index i**, **atom index j** and **atom index k** are the GROMACS serial indices of the atoms in the topology file (*j* being the middle atom);
- **type** is the GROMACS interaction type, should be 1 for harmonic angle potential, RMC try to proceed, even it is not;
- **θ^0 (degree)** is the equilibrium angle, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;
- **k^a (kJ/mol)** is the force constant, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;

- **sigma** parameter to weight the contribution of the angle type for χ^2 calculation, has to be given only for the first occurrence of an angle type. Angles with the same equilibrium angle and force constant belong to a given angle type.

[dihedrals]

This can be in the *.itp file.

- **atom index i , atom index j , atom index k and atom index l** are the GROMACS serial indices of the atoms in the topology file (atoms following each other in the chain in alphabetical order);
- **type** is the GROMACS interaction type, 1 for periodic, 2 for harmonic and 3 for Ryckaert-Bellemans dihedral potential, RMC quit, if it is neither of them.
- θ^0 or ξ^0 (**degree**) are the equilibrium angle depending on the dihedral type, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;
- k^ϕ (**kJ/mol**) or k^ξ (**kJ/mol/rad²**) are the force constant depending on the dihedral type, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;
- $C_0, C_1, C_2, C_3, C_4, C_5$, (**kJ/mol**) are the parameters for the Ryckaert-Bellemans potential, not necessary for GROMACS as it comes from force field files, but has to be given before or after semicolon for RMC;
- **sigma** parameter to weight the contribution of the dihedral type for χ^2 calculation, has to be given only for the first occurrence of a dihedral type; Dihedral angles with the same type, and other parameters belong to a given dihedral angle type.

[system]

Has to be in the *.top file, the moleculetype definitions should precede this.

- system name is not relevant for RMC

[molecules]

Has to be in the *.top file following the [system] directive.

- **molecule name** has to be the same declared with the [moleculetype] directive.
- **number of molecules**

C.2.7. Rendering the RMC configuration to the topology

The structure of the text type RMC configuration file was not changed. As a reminder the structure of the *.cfg file follows the simple rule, that the total number of atoms, the number of atom types, the half length of the simulation box and the number of atoms for each type are specified in the header of the *.cfg file, then all the coordinates of the first type of atoms (each a separate line), then the coordinates of the second and so on atom types can be found consecutively. Because of this, if there are molecules consisting of different type of atoms, then the coordinates of the atoms belonging to the same molecule can be found in non-consecutive lines of the file. Even if there are more than one atom from the same RMC type in the molecule, they not necessarily located consecutively, but sometimes in blocks, depending on the structure of the *.cfg and the accompanying *.fnc file, if no potential is used. The only important thing is, that has to be logic in the arrangements of the atoms, so if we know the atom indices of the atoms belonging to the first and second molecule, we can calculate the atom indices of the atoms in any molecule. The coordinates are reduced, going between -1 and $+1$ and the box half-length and other distance related properties are given in Ångstrom in RMC.

For example there are two different representations, two sets of *.cfg with their matching *.fnc file for a normal simulation, where no topology and potential is used of SnI4, see the atom indices in Table 1.

Table 1: The indices of the atoms in the two differently structured *.cfg and *.fnc file fro SnI4.

Atoms	First representation	Second representation
		Atom indices

	first mol.	second mol.	first mol.	second mol.
Sn	1	2	1	2
I	1001	1005	1001	1002
I	1002	1006	2001	2002
I	1003	1007	3001	3002
I	1004	1008	4001	4002

On the other hand in GROMACS the distances and coordinates are given in nm, and the simulation box goes from 0 → box length in the *.gro file, but this is not relevant for RMC.

In GROMACS atoms have a name and a type. It has to be emphasized, that the GROMACS atom types should not be confused with the atom types in RMC. In GROMACS atoms of the same chemical type can have different GROMACS types, as the GROMACS type have to reflect the chemical role, bonds, hybridisation state of the atom causing it to have different Lennard-Jones parameters or charge. Each force field has its own defined different GROMACS atom types. You have to use one of the atom types of the force field you chose, if you want to use the tology for GROMACS as well, so check the *ff*.atp* file of the force field for the available types in the top directory of the GROMACS installation. If the topology is only used for RMC, then of course one does not have to stick to the atom types of a force field, any type name can be used, but always use a consistent set of parameters! The atom names can be freely chosen (max 5 character).

D. Scalable sigma parameters, leading series

If more, than one data series and constraints are used, it can be a lengthy and tiresome job to set the sigma (or other words weight) parameters properly, to ensure the proper convergence. It has to be kept in mind, that the largest χ^2 data set will guide the system, and other data series or constraints with small sigma will not have virtually any effect on the outcome of the simulation. Therefore it is important to set the sigma values adequately to make sure, that the χ^2 of all the series and constraints should make its contribution. For this point this could only be done by trial an error, restarting the simulation, as the deviation od the data sets and constraints and therefore their χ^2 cannot be known beforehand.

To make this easier, there is a possibility to choose a leading series, and to set the χ^2 values of the other series and constraints to the percentage of the χ^2 of the leading series. If we want to use this scaling feature, then for the sigma values for the data series and constraints to be scaled a negative fraction, which absolute value specifies the desired ratio of the given series per the leading series initial χ^2 should be given. The negative value only indicates, that not normal sigma, but scaling should be used.

The index of the leading series is the first series of the simulation by default (the order of the data sets is the same as they appear in the *.dat file, ($g(r)$, $S(Q)$, $F(Q)$, $E(k)$ data, cosine distribution of bond angles, coordination number constraints including the sub-constraints, average coordination constraints, local invariance (and bonds, angles and periodic, harmonic and RB dihedrals, if no non-bonded potential is calculated)). The indexing starts with 1.

If non-bonded parameters are present, then a second χ^2 is calculated including the contributions of the non-bonded and bonded potential-related contributions as it is discussed in chapter II.C. In this case another leading series index for this potential related contributions can be given in the *.dat file (see line 54 in Table 6). The indexing for the leading potential series starts with 1 for the (first partial of the) vdW potential, and they follow in the order vdW partials, Coulomb partials, bonds, angles and dihedrals. 1-4 non-bonded interaction partials, and interactions with zero sigma parameter (see later) cannot be given as leading series index!

The χ^2 of the leading potential series can be scaled to the χ^2 of the normal, not potential related leading series, if the leading potential series' sigma value is given as a negative fraction. Those other potential related contributions, where the sigma is a negative fraction will be scaled to the leading potential series' χ^2 .

It can be desirable, that the ratio of all the bonded potential interactions of a molecule should not change compared to each other. This can either be done by using the same fixed sigma for all of them in the *.top or *.itp file of the molecule, but in this case no scaling can be applied, or by the more convenient way of

applying zero sigma values the following way: For a molecule type one bond type has to be selected, and it has to have a non-zero (a fixed positive or scalable negative) sigma in the *.top or *.itp file. If this sigma is negative, than the χ^2 of this bond type will be scaled to the leading (potential) series' χ^2 giving a sigma for this bond type. For all the other bond types and angles and dihedrals of this molecule zero sigma has to be given in the *.top or *.itp file, indicating, that the sigma of the first bond type of this molecule with non-zero sigma has to be used. Even in case of fixed sigma it is better to use this 'one non-zero bond sigma all the other sigma is zero for the non-bonded interaction of this molecule' formalism, as it is a lot easier to change all the bonded interactions' sigma in this case, if it is necessary. It has to be emphasized, that in case of a system with more, than one molecule type, each molecule type can be handled separately, each having one non-zero sigma. It is possible, although it might not make much sense, to use more non-zero sigma values for a molecule. Always the first non-zero bond sigma will be used for setting the zero values of the molecules, the other non-zero values will be used normally (meaning, that their sigma will not be set to the selected bond type's sigma, but their own sigma value will be use for them).

Similarly we might want to ensure that the ratio of the non-bonded interactions potential does not change compared to each other. This can be done similarly to the method described above: The NB weight mode has to be set 0, (ensuring, that all the vdW or Coulomb partials will have the same sigma and will be displayed together, one vdW and one Coulomb interaction). Than the desired (fixed or scalable) sigma has to be given for the vdW interaction, and 0 for the weight of the Coulomb interaction, indicating, that the vdW weight should be used for this too.

If the ratio of all the potential interactions shouldn't change compared to each other, (the total potential energy of the system should be used), than the simplest way to do this (only if non-bonded potential is used), is to specify 2 for the NB weight mode. In this case the screen and the history display will be collapsed (only one item for each interaction type, similarly to NB weight mode 0). Only one sigma parameter has to be given for the vdW interaction, this can be positive, or negative. No other sigma will be read (though the line for the Coulomb weight has to be present). This sigma (after it was calculated, if it was scalable, negative) will be used for all the potential-related interactions.

For example lets consider the following simulations with the data sets, constraint and potential components given in Table 2. The leading series index will be 2, meaning the X-ray data series, and its sigma has to have an explicit value. The initial χ^2 of the neutron data series will be set to 10 % of the initial χ^2 of the X-ray series, the second cosine distribution of bond angles constraint's initial χ^2 to 5 % and the local invariance's to 1 %. The leading potential series (with index 3) will be the 3rd vdW partial in this case, and in itself will be scaled to the leading series having 20 % initial χ^2 compared to this. The other potential-based series will be scaled to this 3rd vdw partials χ^2 , except the angle, and the first RB dihedral, which have fixed sigma. The sigma values used during the run for series to be scaled are calculated to satisfy the ratio between the initial χ^2 of the series to initial χ^2 of the leading series and it is constant during the run.

Table 2: Data sets, constraint and potential components for a simulation example demonstrating the usage of χ^2 scaling. The lines with yellow shading are the normal data series and constraints contributing to the normal χ^2 , the blue shaded potentials contribute to the χ_p^2 . The leading series and leading potential series is printed in bold.

serial index	series/constraint	sigma
1	neutron	-0.1
2	X-ray	3e-5
3	cosine distr. of bond angles	2e-6
4	cosine distr. of bond angles	-0.05
5	local invariance	-0.01
1	vdW partial 1	-0.8
2	vdW partial 2	-1.0
3	vdW partial 3	-0.2
4	Coulomb partial 1	-0.5
5	Coulomb partial 2	-0.4
6	Coulomb partial 3	-0.4
7	bond type 1	-0.2
8	bond type 2	-0.1

9	angle	1e-3
10	RB dihedral 1	2e-7
11	RB dihedral 2	-0.2
12	RB dihedral 3	-0.3

E. Exact continuation of the run

It might be desirable to be able to continue a simulation exactly. If no potentials are present, then this could be done earlier, since the binary **.bcf* file was present, only the previous history (**.hst*) file was overwritten, and the number of generated, tried and accepted moves reset at each new start.

In case of the potential, however as it is represented by double type variable, exact continuation of the run is not possible without the binary **.pot* file, as the potential at the end of a run can never be the same as the one recalculated at the beginning of the next run based on the **.bcf* file due to the rounding errors and the different order the contributions to the potential components are calculated.

An additional problem is, if any of the sigma values were scaled in the original run, because without saving and reloading the originally calculated sigma values new sigma values would be calculated when the run is restarted, based on the χ^2 values at the end of the first run, and the proportion of the different χ^2 components would change.

Therefore a new command line option was created (**cont**), which would start the exact continuation of the run. For this the **.bcf*, the binary **.state* file and in case of potential the binary **.pot* file is necessary. The syntactic of the command line will be discussed later (IV.D).

III.The RMC_POT program

A. The structure of the program

The program is written in C++ and consists of several header and source files listed below.

Table 3: Header files of the RMC_POT program

<i>altern.h</i>	Header file including the necessary built-in headers, containing the options regulating the building of the code, declaration of the functions differing according to the options chosen.
<i>classes1.h</i>	Declaration of basic classes: ExptsData, SimpleCfg, CoordNumbConst, AvCoordConst, CosDistrConst, RunParams, FNC_POT, HistoSet and the classes' destructors.
<i>classes2.h</i>	Declaration of the classes involved in the calculation: DataMat, PPCFSet, CalcPart, CalcData, ChiSquared, History and the classes' destructors
<i>global.h</i>	Global file name, file extension and potential related declarations.
<i>Move.h</i>	Declaration of the Move class responsible for making the movement.
<i>Neighbourlist.h</i>	Declaration of the NeighbourList class related to the gridding of the simulation box, and in case of cosine distribution of bond angles constraint handling the neighbour list. It contains the class' destructor as well.
<i>Threads.</i>	Declaration of the Threads class handling the multi-threading and the class' destructor.
<i>Topology.h</i>	Declaration of the Topology class handling the molecular topology, only included if the _POTENTIAL compiler option is switched on.

<i>units.h</i>	Declaration of the used constants.
<i>utilities.h</i>	Declaration of the auxiliary functions and definition of the function templates.

Table 4: Source files of RMC_POT

<i>altern.cpp</i>	Definition of the function differing according to the chosen options.
<i>AvCoordConst.cpp</i>	Definition of the AvCoordConst class describing the average coordination number constraint.
<i>CalcData.cpp</i>	Definition of the CalcData class holding and calculating the calculated data comparable with the experimental.
<i>CalcPart.cpp</i>	Definition of the CalcPart class holding and calculating the partial $g(r)$, $S(Q)$, $F(Q)$ and $E(k)$.
<i>ChiSquared.cpp</i>	Definition of the ChiSquared class holding and calculating difference between the calculated and the experimental data.
<i>CoordNumbConst.cpp</i>	Definition of the CoordNumbConst class describing the coordination number constraint.
<i>CosDistrConst.cpp</i>	Definition of the CosDistrConst class describing the cosine distribution of bond angles constraint.
<i>DataMat.cpp</i>	Definition of the DataMat class containing the conversion table for the normalization of the $g(r)$, and the Fourier transformation matrices.
<i>ExptsData.cpp</i>	Definition of the ExptsData class containing the experimental data.
<i>FNC_POT.cpp</i>	Definition of the FNC_POT class handling the normal FNC or the potential depending on whether the compiler option <code>_POTENTIAL</code> is switched off or on.
<i>History.cpp</i>	definition of the History class gathering and outputting information about the run
<i>HistoSet.cpp</i>	Definition of the HistoSet class containing and calculating the histogram.
<i>makemove.cpp</i>	Containing the makemove function creating the movement of the atoms.
<i>makemovecus.cpp</i>	Containing the custom makemove function creating the movement of the molecules.
<i>Move.cpp</i>	Definition of the Move class, regulating the movements of the atoms, functions connected with the 'tooclose' atoms, other functions of the Move class.
<i>NeighbourList.cpp</i>	Definition of the NeighbourList class containing the gridding of the simulation box, and in case of cosine distribution of bond angles constraint handling the neighbour list.
<i>PPCFSet.cpp</i>	Definition of the PPCFSet class, handling the ppcf calculation and storage.
<i>RMC_POT.cpp</i>	The main program for RMC_POT regulating the run, creating the instances of the classes, containing the simulation loop.
<i>RunParams.cpp</i>	Definition of the RunParams class describing the parameters of the simulation, calculating some necessary parameters.
<i>SimpleCfg.cpp</i>	Definition of the SimpleCfg class containing the coordinates of the atoms and in case of the <code>_POTENTIAL</code>

	option is switched on, the charge group centres.
<i>Threads.cpp</i>	Definition of the Threads class containing the variables and function necessary for multi-threading, but a basic Thread class without multi-threading-specific variables is used even in case of consecutive compilation.
<i>Topology.cpp</i>	Definition of the Topology class reading and handling the molecular topology, only included if the <code>_POTENTIAL</code> compiler option is switched on.
<i>utilities.cpp</i>	Definition of the auxiliary functions.

B. Files used by RMC_POT

These are the files used by the simulations performed by RMC_POT program for input and/or output.

- The input files written in bold are mandatory, but it is enough, if either the text or the binary coordinates files is given, text type has precedence over binary, if both is given and they are not compatible.
- If neither INPUT or OUTPUT specified, then the file is used for both of them.
- The software can be started without giving any *experimental* data, in this case a hard sphere simulation is performed, constraints can be present.
- The files given in brackets are produced only at the beginning of a whole simulation process.
- * means the general file name of the run, and shared by all the file containing the *.

Table 5: The files used by RMC_POT

*.dat	INPUT: the data file containing the parameters of the run.
*.cfg	The text type configuration file containing the coordinates.
*.bcf	The binary type configuration file containing the coordinates.
<i>sfactorcube</i>	In case of $x_{max} > \sqrt{2}$, this is needed for the surface factor calculation, can be found in the test_run/dummy directory of the validation suite.
<i>anyname</i>	INPUT: The files containing the experimental data, name can be free choice.
*.cus	INPUT: The parameters for molecular move.
*.fnc	INPUT: The Fixed Neighbour Constraint file.
.top (.itp)	INPUT: Only if compiled with <code>_POTENTIAL</code> and any kind of potential is calculated, the topology and include topology files, to describe the molecular topology.
*BOND.fnc, *ANGLE.fnc, *DIHEDRAL.fnc	OUTPUT: only in case of it was compiled with <code>_POTENTIAL</code> switched on, and <code>fnc=4</code> : The RMC index lists of the bonds, angles and dihedrals.
*.hgm (*start.hgm)	Partial histograms.
*lhgm (*start.lhgm)	Only if compiled with <code>_LOCAL_INV</code> switched on: partial local histograms.
*grid (*start.grid)	Gridding of the simulation box.
*.cnc (*start.cnc)	The coordination number constraint.
*.can (*start.can)	The average coordination number constraint.
*.tca (*start.tca)	The indices of the 'tooclose' atoms in case of the moveout option.
*.state	The binary state file needed for exact continuation of the run (the program started with the <code>cont</code> command line option).
*.pot	Only in case of it was compiled with <code>_POTENTIAL</code> switched on and non-bonded and/or bonded potential used: the binary potential file.
*.cos (*start.cos)	OUTPUT: The result of the cosine distribution of bond angles constraints.
*.calcdat	OUTPUT: The initial calculated $g(r)$, $S(Q)$, $F(Q)$, $E(k)$ data.
*.fit (*fit_start)	OUTPUT: The total calculated and renormalized experimental $g(r)$, $S(Q)$,

	$F(Q, E(k))$ data, (<i>*.fit_start</i> only if compiled with <code>_TEST_MODE</code> compiler option).
<i>*.expt</i>	OUTPUT: The experimental data is saved to it for checking.
<i>*.ppcf</i>	OUTPUT: The partial radial distribution function $g(r)$ with the same spacing as the histogram.
<i>*.pgr</i>	OUTPUT: The calculated $g(r)$ partials on the same r points, as the 'experimental' $g(r)$.
<i>*.psq</i>	OUTPUT: The calculated $S(Q)$ partials for the neutron data sets.
<i>*.pfq</i>	OUTPUT: The calculated $F(Q)$ partials for the X-ray data sets.
<i>*.pek</i>	OUTPUT: The calculated $E(k)$ partials for the EXAFS data sets.
<i>*.hst</i>	OUTPUT: The history file containing information about the run.
<i>*.chi</i>	OUTPUT: The initial χ^2 for each data set.
<i>*.datmat</i>	OUTPUT: The conversion tables of the DataMat object (only if compiled with <code>_TEST_MODE</code> compiler option).
<i>*.out</i>	OUTPUT: The total and partial calculated and total renormalized experimental $g(r)$, $S(Q)$, $F(Q)$, $E(k)$ data in RMCA format, only if compiled with <code>_OLDFORMAT_OUT</code> compiler option.
<i>*.nei</i>	OUTPUT, only in case the code was compiled with <code>_NEI</code> switched on: containing the neighbour list, and in case compiled with <code>_NEIE</code> the squared distances as well
<i>*.excl</i>	OUTPUT, only in case of it was compiled with <code>_POTENTIAL</code> switched on and non-bonded potential is calculated: the indices of the excluded atoms for each atom.

C. The structure of the *.dat file

The main parameters of the simulation can be found in this file. The file has a fixed format in the sense that the order of the lines is fixed. Majority of the lines has to be always present in the file, but if certain options are used (either regulated by compiler option or by a switch in the *.dat file), additional lines at certain place of the file has to be included. These lines will be highlighted with colours in the example.

All lines will begin with data, and after the data remarks can follow. The remarks does not have to begin with a exclamation mark, it just helps to mark its beginning. The data can be separated by any number of spaces and/or tabs. If multiple data has to be given in one line, they have to be in the same line, and not split into several lines, although this was possible with earlier versions. No empty lines can be present, except the end of the file. The file does not refer to any actual simulation, it features all the possible line types.

Table 6: The structure of the *.dat file. The data given underlined and italic means that they are optional, and do not have to be specified, the values given here are the default values used in the case nothing is given. The first column is just a serial index to reference the lines, it is not part of the file! Only the lines without shading mandatory in each case, the lines with coloured shading are only necessary, if the data set, constraint or option they are referring to are used.

1	test_run	
2	0.076605545	! number density in (1/A3)
3	1.3 2.2 1.5	! cutoffs for each RMC partials in A
4	0.1 0.2	! maximum moves for each types in A
5	0.1 <u>0.1</u>	! r spacing (A), <u>r spacing local inv. (A)</u>
6	.true.	! whether to use moveout option (0 or .false.: false, 1 or .true. : true)
7	10 <u>1</u>	! no of configurations to collect, frequency
8	1000	! step for printing
9	120 30	! time limit, step for saving in minute
10	1 1 1 1 <u>1</u>	! no. of $g(r)$, neutron, X-ray and EXAFS data series, leading

		series index
11	mydata.gr	! g(r) file name
12	1 305	! range of r points (start with 1)
13	0.00	! constant to subtract
14	0.2 0.5 0.3	! partial coefficients in RMC order
15	1e-5	! standard deviation
16	.false.	! whether to vary amplitudes (0 or .false.: false, 1 or .true. : true)
17	neutron.sq	! file name for the neutron data
18	10 101	! range of Q points (start with 1)
19	0.000	! constant to subtract
20	0.2 0.1 0.7	! partial coefficients in RMC order
21	3e-12	! standard deviation
22	0	! whether to vary amplitudes (0 or .false.: false, 1 or .true. : true)
23	1	! whether to vary constant (0 or .false.: false, 1 or .true. : true)
24	0	! whether to vary linear (0 or .false.: false, 1 or .true. : true)
25	0	! whether to vary quadratic (0 or .false.: false, 1 or .true. : true)
26	xray.fq	! name of the X-ray file
27	1 139	! range of Q points (start with 1)
28	0.000	! constant to subtract
29	3e-12	! standard deviation
30	.true.	! whether to vary amplitudes (0 or .false.: false, 1 or .true. : true)
31	.true.	! whether to vary constant (0 or .false.: false, 1 or .true. : true)
32	.true.	! whether to vary linear (0 or .false.: false, 1 or .true. : true)
33	.false.	! whether to vary quadratic (0 or .false.: false, 1 or .true. : true)
34	21 31 <u>21 31 21 31</u>	! range of histogram bin points to use for EXAFS (r space), one first used-last used pair has to be given at least. If it differs for the partials, then ntypes partial pairs has to be given
35	3	! chi(k) weighting
36	exafs.ek	! name of the k-E(k) file
37	1 444	! no. of k data points (starting with 1)
38	60 260	! range to be used in fitting
39	1	! type of absorbing particle
40	exafs_coeff.dat	! coefficient file for EXAFS
41	-0.0005	! weight parameter
42	1	! whether to vary amplitudes (0 or .false.: false, 1 or .true. : true)
43	0	! whether to vary constant (0 or .false.: false, 1 or .true. : true)
44	4 <u>0.05</u>	! number of cosine distr. of bond angles constraint, dcos_theta (spacing in cos(theta) space)
45	1 2 1 1 2.1 2.1 2.7 2.7 110.5 0.17 0.001	! calc method (0:step, 1:Gauss, 2:no angle, 3:file),central, neigh1, neigh2 type, dmin1, dmin2, dmax1, dmax2, 0-2:(angle, wcontrol) 3:filename, weight
46	2 3 1 2 2.3 3.0 3.0 3.8 41 0.2 0.0001	! calc method (0:step, 1:Gauss, 2:no angle, 3:file),central, neigh1, neigh2 type, dmin1, dmin2, dmax1, dmax2, 0-2:(angle, wcontrol) 3:filename, weight
47	0 2 1 1 2.1 2.1 2.7 2.7 110.5 20 0.001	! calc method (0:step, 1:Gauss, 2:no angle, 3:file),central, neigh1, neigh2 type, dmin1, dmin2, dmax1, dmax2, 0-2:(angle, wcontrol) 3:filename, weight
48	3 1 2 2 2.1 2.1 2.7 2.7 SeAsSe.cos 0.0002	! calc method (0:step, 1:Gauss, 2:no angle, 3:file),central, neigh1, neigh2 type, dmin1, dmin2, dmax1, dmax2, 0-2:(angle, wcontrol) 3:filename, weight
49	2 1 2 2 1	! no of coordination constraints, number of neighbour type/constraint, number of sub-constraints/constraint

50	2 1 2.1 2.7 2 3 0.4 0.6 0.00015 0.00025	! central type, neighbour type(s), rmin[first_neightype] ... rmin[last_neightype], rmax[first_neightype] ... rmax[last_neightype], desired coordination number[first_subconst]... desired coordination number[last_subconst], fraction[first_subconst]...fraction[last_subconst], sigma[first_subconst]...sigma[last_subconst]
51	3 1 2 2.4 3.0 2.95 3.5 2 1.0 0.00025	! central type, neighbour type(s), rmin[first_neightype] ... rmin[last_neightype], rmax[first_neightype] ... rmax[last_neightype], desired coordination number[first_subconst]... desired coordination number[last_subconst], fraction[first_subconst]...fraction[last_subconst], sigma[first_subconst]...sigma[last_subconst]
52	1	! number of average coordination constraints
53	1 1 34.0 40.0 12 0.004	! type of the central atom, type of neighbour, min dist, max dist, desired average coord numb, sigma
54	1 1 3 0.5 0.5 7 <u>12</u>	! whether to use a potential, weight mode, combination rule, vdW fudge, Coulomb fudge, lead series index2, LJ_power
55	0.9 1.1	! vdW and Coulomb cutoff in A
56	4	! number of different GROMACS types
57	3.5 2.5 3.6 3.5	! LJ sigma in A
58	0.276144 0.12551 0.148532 0.276144	! LJ epsilon in kJ/mol
59	1e-3 2e-3 3e-3 4e-3 5e-3 6e-3 7e-3 8e-3 9e-3 1e-4 1.1e-4 1.2e-4 1.3e-4 1.4e-4 1.5e-4 1.6e-4 1.7e-4 1.8e-4 1.9e-4 2.0e-4 2.1e-4	! vdW weights (max number of RMC partials)
60	1e-5 2e-5 3e-5 4e-5 5e-5 6e-5 7e-5 8e-5 9e-5 1e-6 1.1e-6 1.2e-6 1.3e-6 1.4e-6 1.5e-6 1.6e-6 1.7e-6 1.8e-6 1.9e-6 2.0e-6 2.1e-6	! Coulomb weights (max number of RMC partials)
61	4 -D_FF_OPLS -D_FLEX - DORI_ANGLE -DORI_BOND	! FNC switch, define options for the topology in case of FNC=4
62	0	! initial bin shift
63	1.0	! xmax used in the run (Å)
64	1	! number of atoms moved in a single move
65	2	!size of the history buffer (0->no history record, 200 is good)
66	100	! number of saves between each history buffering
67	0	! indicator of custom move (0:no, 1:yes)
68	1	! whether to load the histogram (and coord. numbers, cos distr. local hist, if they are used), if the files are available (0 or .false.: false, 1 or .true. : true)
69	14	! maximum number of atoms in a gridcell
70	0.3 1	! fraction of swaps; ntypes*(ntypes-1)/2 entry: 0 (not allowed) or 1 (allowed) for the possible mixed partial (in order 1-2, 1-3,... 2- 3...)
71	2	! total number of threads to use
72	2 -0.3 -0.1 1 0.0 0.2 0.4	! number of local invariance intervals (N_{loc}), sigmas (N_{loc} entry), mode, loc_ratio ($N_{loc} + 1$ entry)

Detailed description of some of the parameters, the parameters will be referenced by the serial number in the first column.

- 1.) Title of the run, maximum 80 character.
- 2.) Number density for the system in \AA^{-3} . If this does not correspond to the half box length and number of atoms in the *.cfg file, then the simulation box will be rescaled according to this value in the *.dat file, and this will be used during the simulation, and the rescaled half box length will be written in the *.cfg file at saving.
- 3.) Cutoff values for the hard sphere pair types in Angstrom, ntypes*(ntypes+1)/2.
- 4.) Maximum moves for each RMC type in \AA .
- 5.) Width for the histogram bins in \AA , if compiled for local invariance calculation, then the width for the local histogram bins can be given in \AA , if not the same is used as for the normal histogram.
- 6.) If there are particles closer to each other than the cut-off distance (see I.C), using the moveout option (1) means to move more frequently the 'tooclose' particles to increase their distance above the cut-off. (0) means not to use moveout option. There can be more, than one 'tooclose' particles among the n_{moved} moved particles, and it can be used even in the case of the molecular move!
- 7.) The number of configuration to be collected after the time limit has been reached. A configuration will be saved at each frequency*savetime. They will be collected in the *_coll.cfg file.
- 8.) Printing to the standard output in every number of step specified here.
- 9.) Time limit for the simulation in minute, step for saving in minute. Timelimit is ignored, if compiled in _TEST_MODE, as in this case it runs to the number of generated steps specified by LAST_GEN in *altern.h*.
- 10.) Number of $g(r)$, *neutron*, *X-ray* and *EXAFS* data, index of the leading series (see II.D above).
- 11-16.) Only has to be included, if $g(r)$ data series is present, and as many blocks of it after each other as the number of $g(r)$ series is present.
- 11.) The name of the $g(r)$ data file, with maximum filename size regulated by FILE_NAME_SIZE located in *units.h*.
- 12.) The first and last r data point to use (indexing start with 1).
- 13.) Constant to be subtracted from the experimental data after it was read, shifting the data along the y-axis.
- 14.) Partial coefficients following each other in RMC order, as many as the number of RMC partials, their sum normalised to 1.
- 15.) Sigma value. The deviation of the experimental and calculated data set is divided by its square, it is used for scaling the contribution of the different data series and constraint to each other. The smaller the value, the larger the contribution of the given data set will have to the total χ^2 .
- 16.) Whether to use renormalization of the data set by varying the amplitude (see I.G. for details).
- 17-25.) Only has to be included, if $S(Q)$ data series is present, and as many blocks of it after each other as the number of $S(Q)$ series present.
- 17.) The name of the $S(Q)$ data file, with maximum filename size regulated by FILE_NAME_SIZE located in *units.h*.
- 18.) The first and last Q data point to use (indexing start with 1).
- 19.) Constant to be subtracted from the experimental data after it was read, shifting the data along the y-axis.
- 20.) Partial coefficients following each other in RMC order, as many as the number of RMC partials, their sum normalised to 1.
- 21.) Sigma value. The deviation of the experimental and calculated data set is divided by its square, it is used for scaling the contribution of the different data series and constraint to each other. The smaller the value, the larger the contribution of the given data set will have to the total χ^2 .
- 22.) Whether to use renormalization of the data set by varying the amplitude (see I.G. for details).
- 23.) Whether to use renormalization of the data set by changing the constant (see I.G. for details).
- 24.) Whether to use renormalization of the data set by varying the linear coefficient (see I.G. for details).
- 25.) Whether to use renormalization of the data set by changing the quadratic coefficient (see I.G. for details).
- 26-33.) Only has to be included, if $F(Q)$ data series is present, and as many blocks of it after each other as the number of $F(Q)$ series present.

- 26.) The name of the $F(Q)$ data file containing not only the Q and $F(Q)$ data points, but the Q -dependent coefficients as well always normalised that their sum is 1 for each Q point, with maximum filename size regulated by `FILE_NAME_SIZE` located in `units.h`.
- 27.) The first and last Q data point to use (indexing start with 1).
- 28.) Constant to be subtracted from the experimental data after it was read, shifting the data along the y -axis.
- 29.) Sigma value. The deviation of the experimental and calculated data set is divided by its square, it is used for scaling the contribution of the different data series and constraint to each other. The smaller the value, the larger the contribution of the given data set will have to the total χ^2 .
- 30.) Whether to use renormalization of the data set by varying the amplitude (see I.G. for details).
- 31.) Whether to use renormalization of the data set by changing the constant (see I.G. for details).
- 32.) Whether to use renormalization of the data set by varying the linear coefficient (see I.G. for details).
- 33.) Whether to use renormalization of the data set by changing the quadratic coefficient (see I.G. for details).
- 34-43.) Only has to be included, if $E(k)$ data series is present, and as many blocks of it after each other as the number of $E(k)$ series present.
- 34.) The range of histogram points to be used for the calculation starting with 1 (r space). At least one first used-last used pair has to be given, if different starting or end point should be used for the different partials, then it has to be given one pair after the other in the same line for all the partials, which means `nytypes` pair, as only the partials containing the edge particles exist.
- 35.) The weight parameter n in $E(k)=k^n\chi(k)$ (see I.F.1.)
- 36.) The name of the $E(k)$ data file, with maximum filename size regulated by `FILE_NAME_SIZE` located in `units.h`.
- 37.) The number of k data points in the file.
- 38.) The first and last k data point to use (indexing start with 1).
- 39.) The RMC type of the absorbing particle (starting with 1).
- 40.) The name of the coefficient file containing the (r,k) dependent coefficients for the Fourier transformation. Data belonging to the same r are in columns and belonging to the same k form rows. It has to be at least as many columns as the last used r value given in line 34 and as many rows as the last used k point given in line 38.
- 41.) Sigma value. The deviation of the experimental and calculated data set is divided by its square, it is used for scaling the contribution of the different data series and constraint to each other. The smaller the value, the larger the contribution of the given data set will have to the total χ^2 .
- 42.) Whether to use renormalization of the data set by varying the amplitude (see I.G. for details).
- 43.) Whether to use renormalization of the data set by changing the constant (see I.G. for details).
- 44.) Number of cosine distribution of bond angle constraint, and if there is any the spacing in $\cos(\theta)$ space, `dcos(θ)`.
- 45-48.) Only has to be included, if cosine distribution of bond angle constraint is present, and as many line as the number of constraint.
- 45.) Calculation method (0: step, 1: Gauss, 2: no angle, 3: experimental distribution from file), central type, neighbour1 type, neighbour2 type starting with 1, minimum and maximum distances d_{min1} , d_{min2} , d_{max1} , d_{max2} , for method 0-2: desired angle in degree, wcontrol parameter, for method 3: file name, for each method: sigma (see I.I.).
- 49.) Number of the coordination number constraints, number of neighbour type for each constraint, number of sub-constraint for each constraint (see I.J).
- 50-51.) Only has to be included, if coordination number constraint is present, a line for each constraint.
- 50.) First comes the central type, neighbour type(s), $r_{min}[\text{first_neightype}] \dots r_{min}[\text{last_neightype}]$, $r_{max}[\text{first_neightype}] \dots r_{max}[\text{last_neightype}]$, desired coordination number[`first_subconst`]... desired coordination number[`last_subconst`], fraction[`first_subconst`]...fraction[`last_subconst`], sigma[`first_subconst`]...sigma[`last_subconst`]
- 52.) Number of the average coordination constraints (see I.K).
- 53.) Only has to be included, if there are average coordination constraint, a line for each constraint. Type of the central atom, type of neighbour, min distance, max distance, desired average coordination number and sigma

- 54.) Potential switch: 0: no potential is used, 1 Lennard-Jones potential is used. Weight mode can be 0, in this case all the vdW partials will be summed and weighted with one sigma, the same goes for Coulomb partials, so only one sigma has to be given for vdW and one for the Coulomb interaction. If it is 1, then the vdW partials and Coulomb partials will be weighted separately, and each requires its own sigma parameter, so npartials sigma has to be given for both the vdW and for the Coulomb interaction. In case of 2 all the bonded and non-bonded potential related interactions will be weighted with the same sigma, so only one sigma for the vdW interaction in line 59 has to be given, and line 60 (Coulomb weight) has to be present, but not read. No weight parameters are read from the *.top or *.itp file either. The combination rule parameter is discussed earlier (II.C.2.4), can be 0-3. vdW fudge, Coulomb fudge (II.C.2.3), lead potential series index (II.D), and the optional LJ_power for the repulsion term, if it is not given, then 12 is assumed.
- 55-60.) Only used in case of non-bonded potential calculation.
- 55.) vdW and Coulomb cutoff in **reduced units**, like x_{max} , the potential will be calculated only up to this reduced distance.
- 56.) Number of different GROMACS type of atoms, (can differ from the number of RMC types (II.C.2)).
- 57.) Lennard-Jones sigma parameter will be read according to the combination rule (II.C.2.4) for all the GROMACS partials in case of combination rule=0, or for all the GROMACS types in case of combination rule>0. LJ sigma has to be given in Ångstrom.
- 58.) Lennard-Jones epsilon parameter will be read according to the combination rule (II.C.2.4) for all the GROMACS partials in case of combination rule=0, or for all the GROMACS types in case of combination rule>0. LJ epsilon has to be given in kJ/mol.
- 59.) vdW weight parameter, only one entry, if weight mode=0 in line 54, number of RMC partials entry, if weight mode=1. Can be positive or negative (this means scalable, see II.D for details).
- 60.) Coulomb weight parameter, only one entry, if weight mode=0 in line 54, number of RMC partials entry, if weight mode=1. Can be positive, negative or only if weight mode=1, zero (see II.D for details).
- 61.) FNC switch, if 0, no FNC is applied. fnc=1 means normal FNC; fnc=2: normal FNC, but if there are out-of range FNC pairs, then the constraint range will be set according to the largest distance of the given FNC constraint, fnc=3: in case of out-of range pairs, only moves, where the distance of them is closer to the desired range are accepted (I.L, II.A). For option 1-3 *.fnc file has to be supplied (III.G). For fnc=4 the program has to be compiled with the _POTENTIAL define option, and means the usage of flexible, MD-like molecules kept together by forces (II.C). No *.fnc file is needed, but *.top describing the molecular topology has to be given (II.C.2.5). In case of fnc=4 after the FNC switch the define options for the processing of the topology file can be given in the format, that the actual define option given in the topology has to be preceded without space by -D (see II.C.2.5). There can be any number of define options.
- 62.) Number of bins to leave out from the calculation at the small distance end (before the first used histogram bin). Can be a fraction! 0.5 for example means, that the lower edge of the first histogram bin starts at bin_size*0.5.
- 63.) x_{max} : largest distance between the particles to include in the histogram calculation in reduced unit (maximum is $\sqrt{3}$). If $x_{max} > \sqrt{2}$, then the sfactorcube file (can be found in the validation suite test_run/dummy) containing the tabulated values for the surface factor calculation has to be present.
- 64.) Number of atoms moved in a RMC move step. If molecular move is used, then the number of moved atoms has to be the same, as the number of atoms in the molecule.
- 65.) Size of the history buffer, each containing the χ^2 for a given phase of the simulation. When the buffer is full, the data is written to disc. Large value means less frequent disc writing, so greater program speed. (0->no history record, 50-100 is recommended).
- 66.) Regulates the interval, the χ^2 is saved to the history buffer. Integer number, denotes the savings to disc between each history buffering, 0 means history is recorded at each .
- 67.) If 0, normal atomic RMC, if 1, molecular move is applied. For this later the program has to be compiled with an adequate makemovecus.cpp file, and the *.cus file has to be given with the molecular move related parameters (see I.D).
- 68.) If the histogram file *.hgm, {and the *.cnc and *.acn files if there is/are (average) coordination number constraint(s)} and the *.tca file if the moveout option is on} are available, in case of (1)

loading of them will be attempted, and if they are compatible with the given constraints and parameters, then initial histogram calculation will not be done. In case of option 0 initial histogram calculation will be performed, and if they were existing files, they will be overwritten. The values of the initial calculation will be saved in the **start.hgm*, **start.cnc* and **start.acn* files as well to be preserved. It has to be emphasized, that in case of loading the histogram and coordination constraint files, the validity of the actual values cannot be checked, so care must be taken to use files corresponding to the **.cfg* and/or **.bcf* files! If local invariance is calculated, then the loading of the **.hgm* file is attempted as well in case of option 1. If potential is calculated, then the loading of the binary **.pot* file is attempted as well.

- 69.) The desired number of atoms in a grid cell (see I.E). It is preferable to set the desired maximum number of particles in a grid cell in the **.dat* file to a relatively low value (5-10) depending of course on the system size to ensure at least 5 or preferably more grid cell in each direction.
- 70.) Only has meaning for multi-component systems, where particles from different types can be swapped with each other to help the mixing of the simulation box. First swap fraction has to be given, a real number (between 0-1), which regulates the fraction of swaps related to the moves. In case of (1) only swaps, (0) no swap at all. After this in case of swaps as many integer as the number of mixed partials (their number is $n_{types}*(n_{types}-1)/2$) specify, which mixed partials can participate in the swap, (0) not involved, (1) involved. The order of the partials is the same, as usual, without the 'clean' partials (1-2, 1-3,...2-3,...). Cannot be used together with $fnc > 0$, and for molecular move!
- 71.) Number of threads for parallel execution. Has to be present even in case the program was compiled for standard consecutive execution, only the value is ignored.
- 72.) Only read, if the program is compiled with the `_LOCAL_INV` compiler option (I.L.B): number of intervals (N_{loc}) (only one can be given for bin-based calculation); sigma for each interval, calculation mode (0: bin-based, default, 1: distance-based); $n+1$ real values, loc_ratio_n which in case of the bin-based calculation mean the starting and limiting reduced distances, for which the local invariance is calculated, the histograms are stored only between them. In case of the distance-based approach loc_ratio_n and $n+1$, they represent the fraction of neighbour atoms (the same fraction value resulting in different atom number for each type according to the different number of atoms/type) to begin and end the local invariance calculation with. If mode is not given after sigma, than mode=0 and the local histogram is calculated for the same interval, as the normal histogram.

D. The structure of the **.cfg* file

The format of the text-type coordinate file is did not change.

First a header can be found with general information. The coordinates of the atoms are arranged according to types, first are all the coordinates of the first type, then the second and so on. It has to be noted, that regardless the three separate box vectors, only cubic simulation box can be handled, and only the first box vector is read.

```
(Version 3 format configuration file) !file created by SimpleCfg::save !
BMtMe_(with FNC)_only_for_checking

1000          783          225 moves generated, tried, accepted
  0          configurations saved

1300 molecules of all types
  3 types of molecules
  1 is the largest number of atoms in a molecule
  0 Euler angles are provided

F (box is cubic)
Defining vectors are:
  30.450200   0.000000   0.000000
   0.000000  30.450200   0.000000
   0.000000   0.000000  30.450200
```

```

300 molecules of type 1
  1 atomic sites
    0.000000  0.000000  0.000000

200 molecules of type 2
  1 atomic sites
    0.000000  0.000000  0.000000

800 molecules of type 3
  1 atomic sites
    0.000000  0.000000  0.000000

-0.401797913580607  -0.456779264844952  0.595887013035009
-0.880737716836848  0.494151149455218  -0.656959975967055
-0.554530511055525  -0.353245955004950  0.885121211703789
0.406319420672594  0.389833805383036  -0.855405069648957
0.919944814288837  0.762568608721879  -0.185056930847424
0.573462973900463  0.034026607847307  0.202465206711665
-0.746369290392583  0.178957135325712  -0.948424915243704
-0.862191047832178  -0.514506207176586  0.355296772334447
0.408792067729221  -0.001200463338156  0.229725315721684
0.432388851173119  0.878801946366788  0.200428054316574
-0.116400037693675  -0.860609330853663  -0.947427893912050
      .
      .
      .
-0.550081821216744  -0.851298541754620  -0.924841685076493
-0.709210176450055  -0.991277379975797  0.994665938293646
-0.819921420524851  -0.922772485395741  -0.989073934225886
-0.790493404801405  -0.970241507519512  0.990513018228407

```

E. The structure of the experimental data files

The multiple data entry in a line can be separated by as many spaces and or tabulators, as you like. The data does not have to be equidistant, the r , Q , k values are read from the files.

E.1. $g(r)$ data

First the number of data points has to be given. The title of the data series can be given following this in the first line, or in the second line, anyway it is not read. The data will start at the 3rd line, the r and the $g(r)$ data has to be given, for one data point /line.

```

305      ! md_opls_S-S

0.05     0
0.15     0
0.25     0
0.35     0
0.45     0
0.55     0
      .
      .
      .
29.85    0.999753
29.95    1.00012
30.05    1.00042
30.15    1.00077
30.25    1.00019
30.35    1.00031
30.45    1.00016

```

E.2. Neutron scattering, $S(Q)$ data

First the number of data points has to be given. The title of the data series can be given following this in the first line, or in the second line, anyway it is not read. The data will start at the 3rd line, the Q and $S(Q)$ data has to be given, for one data point /line.

```
290      !hs_c2_cc12_2

0.001000000    1.726670E-08
0.001050000    1.717757E-08
0.001102500    1.707971E-08
0.001157625    1.697231E-08
0.001215506    1.685451E-08
0.001276282    1.672537E-08
0.001340096    1.658387E-08
0.001407100    1.642894E-08

      .
      .
      .

0.445000000    2.221700E-12
0.455000000    7.536251E-15
0.465000000    2.333310E-12
0.475000000    7.845965E-12
0.485000000    1.539494E-11
0.495000000    2.366464E-11
```

E.3. X-ray scattering, $F(Q)$ data

First the number of data points has to be given. The title of the data series can be given following this in the first line, or in the second line, anyway it is not read. The data will start at the 3rd line, Q and $F(Q)$ data followed by Q -dependent X-ray coefficients (the sum of the coefficients belonging to the same Q point normalised to 1) has to be given, for one data point /line.

```
444
!As42.5Se42.5I15. xrd data
0.6      -1.13286    0.14809    0.30537    0.16809    0.15743    0.17332    0.0477
0.65     -1.10374    0.14807    0.30536    0.1681     0.15744    0.17333    0.04771
0.7      -1.08888    0.14805    0.30534    0.1681     0.15744    0.17335    0.04772
0.75     -1.07465    0.14803    0.30533    0.1681     0.15745    0.17337    0.04772
0.8      -1.05628    0.14801    0.30531    0.16811    0.15745    0.17339    0.04774
0.85     -1.0301     0.14798    0.30529    0.16811    0.15746    0.17341    0.04775

      .
      .
      .

22.45    -0.00339    0.13772    0.28333    0.18344    0.14572    0.18869    0.06109
22.5     -0.00411    0.13778    0.28347    0.18334    0.14581    0.18861    0.06099
22.55    -0.00475    0.13783    0.28361    0.18324    0.1459     0.18853    0.0609
22.6     -0.0053     0.13788    0.28375    0.18314    0.14598    0.18844    0.06081
22.65    -0.00579    0.13793    0.28388    0.18304    0.14607    0.18836    0.06072
22.7     -0.00617    0.13798    0.28402    0.18294    0.14615    0.18828    0.06063
22.75    -0.00645    0.13803    0.28415    0.18284    0.14624    0.18819    0.06055
```

E.4. EXAFS data, $E(k)$

Two files are needed for each $E(k)$ series, the $E(k)$ data file, and the coefficient file.

The $E(k)$ data file is very similar to the $g(r)$ or $S(Q)$ data file, first the number of data points has to be given. The title of the data series can be given following this in the first line, or in the second line, anyway it is not read. The data will start at the 3rd line, the k and $E(k)$ data has to be given, for one data point /line.

```
266
As42.5Se42.5I15, As edge
0.6      0
0.65     0
```

```

0.7      0
0.75    0
0.8     0
      .
      .
      .
13.6    2.7357E-4
13.65   1.99599E-4
13.7    1.24289E-4
13.75   5.2489E-5
13.8    -1.16595E-5
13.85   -6.4922E-5
13.9    -1.05104E-4

```

In a coefficient file the coefficient for the Fourier transformation from the histogram to the $E(k)$ data is given, (see I.F) for details. The file has to contain as many blocks (given here as an example) following each other without empty lines separating them as the number of RMC types. The first header line contain the number of RMC types (actually the number of partials the edge particle is involved in), then separated by a comma the number of Q data points for a block. The block should be repeated begins here. The next line should contain the number of the r columns and then the name of the partial. If number of r columns is given, then the program can check, whether there is enough data in the line which should be used according to line 34 in the **.dat* file. If it is not given, then the program will still continue, but no checking can be performed. Then has to follow as many line as the number of Q points, containing the coefficients separated by comma.

```

3,444
34 As-As
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.07098, 0.17069, 0.16248,
0.16508, 0.1661, 0.17524, 0.17698, 0.19435, 0.22409, 0.15616, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.07542, 0.14545, 0.15335,
0.15229, 0.14975, 0.15474, 0.15183, 0.16232, 0.18129, 0.2682, 0, 0, 0, 0
      .
      .
      .
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.08905, 0.13954, 0.14374,
0.13931, 0.13355, 0.13474, 0.12779, 0.13216, 0.14169, 0.20472, 0, 0, 0, 0
34 As-Se
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.06068, 0.15069, 0.16248,
0.16508, 0.1661, 0.17524, 0.17698, 0.19435, 0.22409, 0.33616, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.07542, 0.14545, 0.15335,
0.15229, 0.14975, 0.15474, 0.15183, 0.16232, 0.18129, 0.2682, 0, 0, 0, 0
      .
      .
      .
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.08905, 0.13954, 0.14374,
0.13931, 0.13355, 0.13474, 0.12779, 0.13216, 0.14169, 0.20472, 0, 0, 0, 0
34 As-I
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.06068, 0.15069, 0.16248,
0.16508, 0.1661, 0.17524, 0.17698, 0.19435, 0.22409, 0.89616, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.07542, 0.14545, 0.15335,
0.15229, 0.14975, 0.15474, 0.15183, 0.18232, 0.18129, 0.9282, 0, 0, 0, 0
      .
      .
      .
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.08905, 0.13954, 0.14374,
0.13931, 0.13355, 0.13474, 0.12779, 0.13216, 0.14169, 0.20472, 0, 0, 0, 0

```

F. Experimental cosine distribution of bond angles data

In case of calculation method 3 for the cosine distribution of bond angles instead of the angle and wcontrol parameter the name of the experimental distribution file has to be given in the **.dat* file. The structure of the cosine distribution file is very similar to the $g(r)$ or $S(Q)$ data file, the first line contains the number of data points, the second can contain some text or be empty, and from the third line should be given a $\cos(\theta)$ and a distribution value. The $\cos(\theta)$ values have to be equidistant, and denote the middle

of the interval, and the total $-1 \rightarrow +1$ interval has to be divisible with the $\text{dcos}(\theta)$ value (in the example 0.05), and the distribution has to be given for the whole interval. The distribution does not have to be normalized, this is done if necessary by RMC_POT.

An example is:

```
40
-0.975  0.00003511
-0.925  0.0001345
-0.875  0.0004726
-0.825  0.00152295
      .
      .
      .
0.875   0.00000001
0.925   0
0.975   0
```

G. The structure of the *.fnc file

The *.fnc file is needed, if fnc option 1-3 is used. It has to be compatible with the *.cfg file. The number of different FNC constraints has to be given in line 4. In line 5 the minimum distances in line 6 the maximum distances for the FNC constraint types has to be given. In line 8 the number of atoms has to follow. From line 10 begins the 3 lines blocks for each atom. The first of it contains the index of the atom, then the number of FNC neighbours. The next line contains the indices of the neighbours, and the following line the indices of the constraint types. Even if there is no constraint for an atom, a line with the atom index and 0 for the number of neighbours has to be given, but the second and the third line will be missing.

Fixed neighbours constraints (.FNC) file for XYZ3

```
No. of possible rmin-rmax pairs:
4
2.06999993  1.01999998  2.54999995  1.66999996
2.19000006  1.13999999  2.78999996  1.90999997

10000

1 4
2001 4001 4002 4003
1 2 2 2
2 4
2002 4004 4005 4006
1 2 2 2
3 4
2003 4007 4008 4009
1 2 2 2
4 0
      .
      .
      .
9999 4
2000 4000 9998 10000
2 3 4 4
10000 4
2000 4000 9998 9999
2 3 4 4
```

H. The structure of the topology file

As a simple example first the topology file of the spce water containing 2000 molecules will be given:

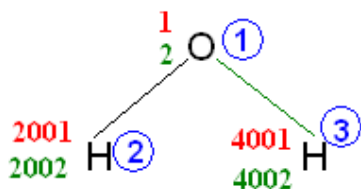


Figure 2: Numbering of the atoms for water: blue circled numbers are the GROMACS atom indices; red numbers are the RMC indices in the first molecule, green in the second molecule.

```
#include "ffoplsaa.itp"

[ moleculetype ]
; molname      nrexcl
SOL           2

[ atoms ]
; nr  type      resnr residue  atom  cgnr  charge      mass RMC_index

      1  opls_116  1      SOL    OW    1     -0.8476           ; 1      .2
      2  opls_117  1      SOL    HW1   1     0.4238           ; 2001  2002
      3  opls_117  1      SOL    HW2   1     0.4238           ; 4001  4002

[ bonds ]
; i j      funct length      force.c. RMC_sigma
1  2      1     0.1         345000    ; 4.5e-3
1  3      1     0.1         345000

[ angles ]
; i j      k      funct angle      force.c.  RMC_sigma
2  1      3      1     109.47    383       6e-4

[ system ]
; Name
spce_water

[ molecules ]
      SOL           2000
```

The second example will show the topology and two include topology files with the molecule type definitions of a bis(methylthio)methane and dimethylsulfide system consisting 100 BMtMe and 10 DMeS molecules.

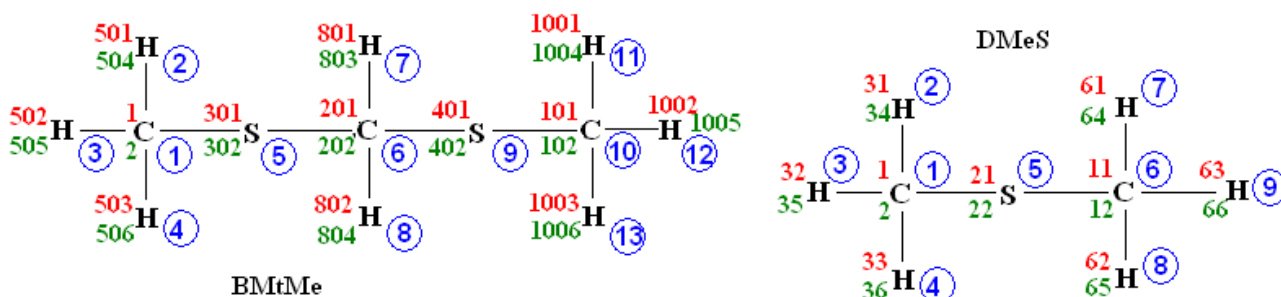


Figure 3: The GROMACS atom numbers (blue circled) and the RMC indices for the first (red) and second (green) molecules of the system.

bis(methylthio)methane.itp

```
[ moleculetype ]
; molname      nrexcl      index_offset for RMC
```

BMtMe 3

```
[ atoms ]
; nr      type      resnr  residue  atom  cgnr  charge  mass  first_i
sec_i
#ifdef _FF_OPLS
  1  opls_209  1      BMtMe   CA    1  -0.013          ; 1    2
#endif
#ifdef _DEUTERIUM
  2  opls_140  1      BMtMe   DA1   1  0.06   2.014102  ; 501  504
  3  opls_140  1      BMtMe   DA2   1  0.06   2.014102  ; 502  505
  4  opls_140  1      BMtMe   DA3  1  1  0.06   2.014102  ; 503  506
#else
  2  opls_140  1      BMtMe   HA1   1  0.06          ; 501  504
  3  opls_140  1      BMtMe   HA2   1  0.06          ; 502  505
  4  opls_140  1      BMtMe   HA3   1  0.06          ; 503  506
#endif
  5  opls_202  1      BMtMe   S1    2 -0.335          ; 301  302
  6  opls_210  1      BMtMe   CB    3  0.216          ; 201  202
#ifdef _DEUTERIUM
  7  opls_140  1      BMtMe   DB1   3  0.06   2.014102  ; 801  803
  8  opls_140  1      BMtMe   DB2   3  0.06   2.014102  ; 802  804
#else
  7  opls_140  1      BMtMe   HB1   3  0.06          ; 801  803
  8  opls_140  1      BMtMe   HB2   3  0.06          ; 802  804
#endif
  9  opls_202  1      BMtMe   S2    4 -0.335          ; 401  402
 10  opls_209  1      BMtMe   CC    5 -0.013          ; 101  102
#ifdef _DEUTERIUM
 11  opls_140  1      BMtMe   DC1   5  0.06   2.014102  ; 1001 1004
 12  opls_140  1      BMtMe   DC2   5  0.06   2.014102  ; 1002 1005
 13  opls_140  1      BMtMe   DC3   5  0.06   2.014102  ; 1003 1006
#else
 11  opls_140  1      BMtMe   HC1   5  0.06          ; 1001 1004
 12  opls_140  1      BMtMe   HC2   5  0.06          ; 1002 1005
 13  opls_140  1      BMtMe   HC3   5  0.06          ; 1003 1006
#endif
#endif
```

```
[ pairs ]
; i j      funct
 2  6      1
 3  6      1
 4  6      1
 1  7      1
 1  8      1
 1  9      1
 5 10      1
 7 10      1
 8 10      1
 6 11      1
 6 12      1
 6 13      1
```

```
[ bonds ]
#ifdef _ORI_BOND
;original OPLS bonds
; i j      funct r      k      RMC_sigma
 1  2      1      ;0.109      284512.0      1e-3
 1  3      1      ;0.109      284512.0
 1  4      1      ;0.109      284512.0
 1  5      1      ;0.181      185769.6      2e-3
 5  6      1      ;0.181      185769.6
 6  7      1      ;0.109      284512.0
 6  8      1      ;0.109      284512.0
 6  9      1      ;0.181      185769.6
 9 10      1      ;0.181      185769.6
```

```

10 11 1 ;0.109 284512.0
10 12 1 ;0.109 284512.0
10 13 1 ;0.109 284512.0
#else
;from Page (2000) J. Phys. Chem. A Vol. 104 p 6672
1 2 1 0.1108 284512.0 ; 3e-3
1 3 1 0.1108 284512.0
1 4 1 0.1108 284512.0
1 5 1 0.1805 185769.6 ; 4e-3
5 6 1 0.1806 185769.6
6 7 1 0.1108 284512.0
6 8 1 0.1108 284512.0
6 9 1 0.1806 185769.6
9 10 1 0.1805 185769.6
10 11 1 0.1108 284512.0
10 12 1 0.1108 284512.0
10 13 1 0.1108 284512.0
#endif

```

```

[ angles ]
#ifndef _ORI_ANGLE
;original OPLS angles
; i j k funct theta k RMC_sigma
2 1 3 1 ;107.8 276.144 1.5e-4
2 1 4 1 ;107.8 276.144
3 1 4 1 ;107.8 276.144
7 6 8 1 ;107.8 276.144
11 10 12 1 ;107.8 276.144
11 10 13 1 ;107.8 276.144
12 10 13 1 ;107.8 276.144
2 1 5 1 ;109.5 292.880 1.6e-4
3 1 5 1 ;109.5 292.880
4 1 5 1 ;109.5 292.880
7 6 5 1 ;109.5 292.880
8 6 5 1 ;109.5 292.880
7 6 9 1 ;109.5 292.880
8 6 9 1 ;109.5 292.880
11 10 9 1 ;109.5 292.880
12 10 9 1 ;109.5 292.880
13 10 9 1 ;109.5 292.880
1 5 6 1 ;98.9 518.816 2.1e-4
6 9 10 1 ;98.9 518.816
#else
;angles coming from Page (2000) J. Phys. Chem. A Vol. 104 p 6672
2 1 3 1 108.9 276.144 ; 1.7e-3
2 1 4 1 108.9 276.144
3 1 4 1 108.9 276.144
7 6 8 1 108.9 276.144
11 10 12 1 108.9 276.144
11 10 13 1 108.9 276.144
12 10 13 1 108.9 276.144
2 1 5 1 107.5 292.88 ; 1.8e-3
3 1 5 1 107.5 292.88
4 1 5 1 107.5 292.88
7 6 5 1 107.5 292.88
8 6 5 1 107.5 292.88
7 6 9 1 107.5 292.88
8 6 9 1 107.5 292.88
11 10 9 1 107.5 292.88
12 10 9 1 107.5 292.88
13 10 9 1 107.5 292.88
1 5 6 1 102.8 518.816 ; 1.9e-3
6 9 10 1 102.8 518.816
#endif
5 6 9 1 115.9 418.400 ; 2e-3

```

```

[ dihedrals ]
;ai aj   ak   al   funct   C0   C1   C2   C3   C4   C5 RMC_sigma
2  1   5   6   3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000 3e-3
3  1   5   6   3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
4  1   5   6   3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
7  6   5   1   3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
8  6   5   1   3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
7  6   9   10  3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
8  6   9   10  3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
6  9   10  11  3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
6  9   10  12  3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
6  9   10  13  3 ; 1.35352 4.06057 0.00000 -5.41410 0.000 0.000
; no opls param for Ct-S-CT-S,the CT-CT-S-CT will be used instead
#ifdef _SMALL_DIH_BAR
; divided by 4 to facilitate free rotation not to let it froze
1  5   6   9   3 0.23535 0.5784375 0.6024975 -1.416285 0.000 0.00 ; 5e-4
5  6   9   10  3 0.23535 0.5784375 0.6024975 -1.416285 0.000 0.00 ; 2.0e-4
#else
1  5   6   9   3 0.94140 2.31375 2.40999 -5.66514 0.000 0.00 ; 3.2e-4
5  6   9   10  3 0.94140 2.31375 2.40999 -5.66514 0.000 0.00 ; 3.3e-4
#endif

```

Include topology file for DMeS, *dimethylsulfide.itp*. The index offset 1300 shows, that there are 1300 other atoms (100*13 atoms of the BMtMe), so therefore the first index in the RMC configuration will begin with 1301. If the number of molecules and therefore atoms for the component(s) preceding the DMeS in the system description, then only the index offset has to be changed and not all the first and second RMC indices.

```

[ moleculetype ]
; molname      nrexcl      index_offset
DMeS           3           ;      1300

[ atoms ]
; nr   type   resnr   residue   atom   cgnr   charge   mass.first_ind   second_ind
#ifdef _FF_OPLS
1   opls_209  1   DMeS     CA    1   -0.013           ;      1      2
#endif

#ifdef _DEUTERIUM
2   opls_140  1   DMeS     DA1   1   0.06   2.014102 ;      31     34
3   opls_140  1   DMeS     DA2   1   0.06   2.014102 ;      32     35
4   opls_140  1   DMeS     DA3   1   0.06   2.014102 ;      33     36
#else
2   opls_140  1   DMeS     HA1   1   0.06           ;      31     34
3   opls_140  1   DMeS     HA2   1   0.06           ;      32     35
4   opls_140  1   DMeS     HA3   1   0.06           ;      33     36
#endif

5   opls_202  1   DMeS     S     2   -0.334           ;      21     22
6   opls_209  1   DMeS     CB    3   -0.013           ;      11     12

#ifdef _DEUTERIUM
7   opls_140  1   DMeS     DB1   3   0.06   2.014102 ;      61     64
8   opls_140  1   DMeS     DB2   3   0.06   2.014102 ;      62     65
9   opls_140  1   DMeS     DB3   3   0.06   2.014102 ;      63     66
#else
7   opls_140  1   DMeS     HB1   3   0.06           ;      61     64
8   opls_140  1   DMeS     HB2   3   0.06           ;      62     65
9   opls_140  1   DMeS     HB3   3   0.06           ;      63     66
#endif
#endif

[ pairs ]
; i j   funct
2  6   1

```

```

3 6 1
4 6 1
1 7 1
1 8 1
1 9 1

```

```
[ bonds ]
```

```
#ifdef _ORI_BOND
```

```
;original OPLS bonds
```

```

; i j      funct length      force.c      RMC_sigma
1 2      1      ;0.109      284512.0      1.4e-3
1 3      1      ;0.109      284512.0
1 4      1      ;0.109      284512.0
6 7      1      ;0.109      284512.0
6 8      1      ;0.109      284512.0
6 9      1      ;0.109      284512.0
1 5      1      ;0.181      185769.6      1.2e-3
5 6      1      ;0.181      185769.6

```

```
#else
```

```
;reset according to T. Iilija, S. Tsuchiya, M. Kimura: Bull. of Chem. Soc of Japan
Vol. 50, No.10 (1977) p2564
```

```

; i j      funct length      force.c      RMC_sigma
1 2      1      0.1116      284512.0      ;1.5e-4
1 3      1      0.1116      284512.0
1 4      1      0.1116      284512.0
6 7      1      0.1116      284512.0
6 8      1      0.1116      284512.0
6 9      1      0.1116      284512.0
1 5      1      0.1807      185769.6      ;2.3e-4
5 6      1      0.1807      185769.6

```

```
#endif
```

```
[ angles ]
```

```
#ifdef _ORI_ANGLE
```

```
;original OPLS angles
```

```

; i j      k      funct angle      force c.      RMC_sigma
2 1      3      1      ;107.8      276.144      5.1e-4
2 1      4      1      ;107.8      276.144
3 1      4      1      ;107.8      276.144
7 6      8      1      ;107.8      276.144
7 6      9      1      ;107.8      276.144
8 6      9      1      ;107.8      276.144
2 1      5      1      ;109.5      292.880      2.3e-4
3 1      5      1      ;109.5      292.880
4 1      5      1      ;109.5      292.880
7 6      5      1      ;109.5      292.880
8 6      5      1      ;109.5      292.880
9 6      5      1      ;109.5      292.880
1 5      6      1      ;98.9      518.816      1.8e-4

```

```
#else
```

```
;reset according to T. Iilija, S. Tsuchiya, M. Kimura: Bull. of Chem. Soc of Japan
Vol. 50, No.10 (1977) p2564
```

```

; i j      k      funct angle      force.c      RMC_sigma
2 1      3      1      109.3      276.144      ; 1.1e-3
2 1      4      1      109.3      276.144
3 1      4      1      109.3      276.144
7 6      8      1      109.3      276.144
7 6      9      1      109.3      276.144
8 6      9      1      109.3      276.144
2 1      5      1      109.3      292.880      ; 7.1e-3
3 1      5      1      109.3      292.880
4 1      5      1      109.3      292.880
7 6      5      1      109.3      292.880
8 6      5      1      109.3      292.880
9 6      5      1      109.3      292.880

```

```

1 5 6 1 99.05 518.816 ; 9.1e-3
#endif

[ dihedrals ]
;ai aj ak al funct C0 C1 C2 C3 C4 C5 RMC_sigma
2 1 5 6 3 ; 1.35352 4.06057 0.0000 -5.41410 0.0000 0.000 2.6e-3
3 1 5 6 3 ; 1.35352 4.06057 0.0000 -5.41410 0.0000 0.000
4 1 5 6 3 ; 1.35352 4.06057 0.0000 -5.41410 0.0000 0.000
1 5 6 7 3 ; 1.35352 4.06057 0.0000 -5.41410 0.0000 0.000
1 5 6 8 3 ; 1.35352 4.06057 0.0000 -5.41410 0.0000 0.000
1 5 6 9 3 ; 1.35352 4.06057 0.0000 -5.41410 0.0000 0.000

```

The topology file: *BMtMe_DMeS.top*:

```

#include "ffoplsaa.itp"

#include "bis(methylthio)methane.itp"
#include "dimethylsulfide.itp"

[ system ]
; Name
BMtMe-DMeS

[ molecules ]
BMtMe 100
DMeS 10

```

The include topology file contains conditional parts, so it depends on whether the arguments like `_DEUTERIUM` or `_ORI_BOND` are passed to the processing of the topology file, which part of the topology is active. In case of the DMeS bond declaration, in the `_ORI_BOND` section the `r` and `k` values are after the semicolon, which means, that GROMACS will ignore them and use the values given in the force field files (which actually are the same as given here) only RMC uses them, but in the other part based on the work of Ilija both GROMACS and RMC are using the values given here (no semicolon).

At the beginning of the topology file the `#include "ffoplsaa.itp"` specifies the force field, it is necessary for GROMACS, and it is checked by the RMC program, whether it is OPLS, as in that case certain thing has to be done differently (like the 1-4 interactions are calculated by scaling) than in case of GROMOS force fields frequently used by GROMACS.

IV. Usage of the RMC program

A. Compilation of the program

As the program was developed with the possibility to be run on different platforms, due to the differences of the operating systems and the available compilers some code changes are necessary before compilation. The program was tested both on PC having Windows operation system using Microsoft Visual C++ compiler, and on GNU/LINUX platform.

A.1. Pre-processor directive for code building

Pre-processor directives regulate the conditional building of the code. The description of the options regulating the different building of the code can be found in the header file *altern.h* having the simple form of `#define PARAMETER`. The options can be turned on in LINUX environment by passing the appropriate command line argument to make. In WINDOWS environment, if we want to choose the given option, then the `#define PARAMETER` belonging to it has to be in the code, if not then it has to be commented out from the code preceding it with `//`. Here the available option will be given in their order appearing in the file.

First option: choosing the platform

`#define _MICROSOFT_VC`//If switched on, compiling using MS Visual C++ with WINDOWS is assumed. If option `_GNU_LINUX` is not passed to the compiler explicitly, then

`_MICROSOFT_WINDOWS` is assumed. DO NOT SWITCH THIS OFF, as in case of the linux Makefile the `_GNU_LINUX` option is automatically switched on.

Second option: whether the code will be built for normal running (this is what usually needed), or for running in test mode. The later was introduced during the testing of the programs to ensure, that the random number generator start with the same value each time, and the program will run to a given number of generated steps specified by `LAST_GEN` to make the results produced by the different version comparable. It is also useful for performance testing.

```
#define _TEST_MODE //this has to be disabled, if the program is used in normal running mode
#define LAST_GEN 1000//the run will end at ngenerated=LAST_GEN in test mode (this does NOT
have to be disabled!!!)
```

Third option: whether build a multi-threading application, the thread libraries has to be present, but the program can still run with only one thread! ONLY switch it off, if you do not have the tread libraries, in this case of course only one thread is assumed, (usual consecutive code) is generated.

```
#define _MULTI
```

Fourth option: the *ppcf*-s will be summed at each saving, and the average is calculated and saved too

```
//#define _SUM_PPCF
```

Fifth option: regulating the LINUX platform based ATLAS library usage. The libraries have to be installed separately (see the [ATLAS](#) web site), only use this option if they are installed! The installed ATLAS libraries are using the BLAS routines, optimised for the given platform, and can increase the speed of the vector-vector, matrix-vector and matrix-matrix operations. In RMCSANS ATLAS is applied for the *ppcf* calculation and the partial $S(Q)$ Fourier-transformation

```
//#define _ATLAS//use the ATLAS libraries for matrix operations
```

Sixth option: only if old style header files are used (this depends on the compiler), which is most probably not the case.

```
//#define _OLD_HEADER //if the old style header (name.h) are used, THE NEW STYLE headers are
used by default, this option has to be commented out normally
```

Seventh option: for Code Warrior on Macintosh, but the RMC++ code was never tested on this platform, so there is no guarantee, that it can be compiled without any change!

```
//#define _CODE_WARRIOR_MAC //DEFAULT is the PC or UNIX version, this option has to be
commented out normally
```

Eights option: additional to the normal output, a *.out file will be created to be compatible with RMCA containing the PPCF-s and partial $S(Q)$ data, then the RMC and (renormalised) experimental total $I(Q)$.

```
#define _OLDFORMAT_OUT
```

Ninth option: `_Q_SWITCH_SQ_FQ` if it is on, then for all the $S(Q)$ and $F(Q)$ data sets the normally calculated $S(Q)$ and $F(Q)$ will be multiplied by Q

```
//#define _Q_SWITCH_SQ_FQ//if it is on, then for all the S(Q) and F(Q) data sets the normally
calculated S(Q) and F(Q) will be multiplied by Q
```

Tenth option: `_NEI`, `_NEIE` only has effect, if cosine distribution constraint is applied. If `_NEI` is on, then the neighbour list is saved to the `*.nei` file. If `_NEIE` is also defined, then the squared neighbour distances and vector is also saved

```
//#define _NEI //if it is on, saving the neighbour list into the *.nei file
```

```
//#define _NEIE //if it is on, saving the neighbour list and also the squared neighbour distances and vector components into the *.nei file
```

Eleventh option: if it is on, the distance so far the atoms moved from their starting positions in each direction is separately accumulated for every atom, and the average is displayed on screen at every run status screen display,

```
//#define _AV_MOVE//if it is on, the move in each direction separately accumulated for every atom and displayed.
```

Twelfth option: if it is on, the chi2 components will be written in to the `*.chi` file for every rejected move.

```
//#define _WRITE_CHI2_DETAIL//if it is on, the chi2 components will be written in to the *.chi file for every rejected move
```

Thirteenth option: if it is on, the code is handling molecules with MD-like bond, angle and dihedral potentials, and the potential energy is calculated, and gives a contribution to χ^2 .

```
#define _POTENTIAL//if it is on, the code is handling molecules with MD-like bond, angle and dihedral potentials, and the potential energy is calculated, and gives a contribution to  $\chi^2$ .
```

Fourteenth option: if it is on, local invariance is calculated.

```
//#define _LOCAL_INV//use local invariance added to  $\chi^2$ 
```

Fifteenth option: if it is on, the local invariance histogram is not saved to save time

```
//#define _LOCAL_INV_NS//do not save local invariance histogram
```

Sixteenth option: if it is on, then hopefully 8 bytes (64-bit) long integers will be used for typedef longint variables, if not then probably 4 bytes. See chapter IV.B for details.

```
//#define _USE_INT64//use integer 64 (8byte) for longint, otherwise int will be used(4 byte)
```

A.2. Constant values

There are some preset constant values in the `units.h` file, which can be altered if need arises. These are:

```
#define PI 3.14159265359
#define SQRPI 1.7724539 //square root of Pi
#define INVPI 0.3183099 //inverse of Pi
#define SQRT3 1.7320508075688772 //sqrt(3) (maximum value in sftable)
#define SFACTOR_SIZE 501 //number of elements inb the sfactorcube file
```

The tolerable difference coming from the different number representation in the binary and decimal number system (if the numbers are represented by 15 digits after the decimal point).

```
#define TOLERANCE 1.0e-15
```

This is used to ensure the accuracy of the bin->dr conversion

```
#define GRID_TOL 1.0e-14
```

This is a safety increase for array dimensions in NeighbouList object

```
#define SAFE_ADD 20;
```

Used during the load of CoordNumbConst and AvCoordConst

```
#define LOAD_TOL 1.0e-8
```

Defining the confidence interval used for the calculation of "negative" cosine distribution of bond angles constraints.

```
#define CONF_INT 3
```

Size of a line for the line buffer for some file processing

```
#define LINE_SIZE 200
```

//the fraction the moved atoms are chosen from among the 'tooclose' atoms rather than from all the atoms

```
#define TOO_CLOSE_FRACTION 0.5
```

The CACHE related things are architecture dependent, the given values and caching concept is for the Intel64 architecture.

```
#define NUMBER_OF_CACHE_LINES_TO_FETCH 1//Number of cache lines to cache in the same time (prefetch)
```

Size of the L1 cache, needed in some cases to optimise cache usage. False sharing between threads has to be prevented. This can happen, when although different threads are writing different memory addresses, but the addresses are so close to each other, that they would be cached together into the same cache line (are inside the same CACHE_ALIGNMENT block). Because of this, if one part of a cache line is modified, the whole cache line is written back to memory, so different threads may want to write the same part of the memory holding back each other causing if this happens too often to slow the performance down.

```
#define CACHE_LINE_SIZE 64 //Byte
```

The thread segments of some arrays have to be separated at least with CACHE_PADDING-size(data_type) amount of bytes have to be kept between the threads segment data.

```
#define CACHE_PADDING NUMBER_OF_CACHE_LINES_TO_FETCH*CACHE_LINE_SIZE
```

The size of the file names

```
#define FILE_NAME_SIZE 50
```

The size of names

```
#define NAME_SIZE 50
```

These are needed for the topology in case of MD-like molecules

If `_POTENTIAL` is used:

number of recognized GROMACS directives

```
#define N_GR_DIR 8
```

number of recognized pre-processor directives

```
#define N_COMP_DIR 4
```

number of force field types

```
#define N_FORCEF 10
```

default value for the number of molecule types

```
#define N_MOLTYPE 5
```

Maximum number of segments in the pre-processor arrays (number of topology and include top. files)

```
#define N_TOPFILE 5
```

Maximum number of active (embedded) ifdef statements in a file in the same time

```
#define N_ACT_IFDEF 5
```

number of different potential types (harmonic bond and angle and Ryckaert-Bellemans dihedral)

```
#define N_POT_TYPE 3
```

number of different dihedral functions

```
#define N_DIH_FUNCT
```

for the Coulomb interaction ($1/4/\pi/\epsilon_0$) (kJ*A/mol/e2)

```
#define f_Coulomb 1389.35485
```

If `_LOCAL_INV` is used:

```
//step for resetting the thread boundaries for the neighbour atoms, if distance based calculation is
used the
#define LOC_LOAD_BALANCE_STEP 1000
```

A.3. Compilation on Linux platform, the usage of the Makefile for RMCSANS

The supplied Linux Makefile can have the following command line options, which will regulate the building of the code. The status of the switches in *altern.h*, whether they are commented out or not is of no consequence, as the Makefile will always pass the Linux platform specific `_GNU_LINUX` switch to the compiler, and the options in the *altern.h* will be bypassed. Instead, an option can be switch on by passing command line arguments to the make. The name of the executable will contain indicators of the used option switches to avoid confusion. The file names will always begin with 'rmcp' and end with '.exe'.

Table 7: The command line options for make, and the indicators in the executable name are the following:

Command line argument	file name indicator	option switch
TEST=X	_t	for <code>_TEST_MODE</code> , X is the number of generated steps to make. The value given here overwrites the one given in the <i>altern.h</i> file.
MULTI=0	_multi	for <code>_MULTI</code> threading
SUMP=0	_sp	for <code>_SUM_PPCF</code>
AT=0	_atlas	for <code>_ATLAS</code>
OH=0	_oh	for <code>_OLD_HEADER</code>
MAC=0	_mac	for <code>_CODE_WARRIOR_MAC</code>
OLDOUT=0	_oo	for <code>_OLDFORMAT_OUT</code>
MULQ=0	_mq	for <code>_Q_SWITCH_SQ_FQ</code>
NEI=0	_nei	for <code>_NEI</code>
NEI=e	_neie	for <code>_NEI</code> and <code>_NEIE</code>
AVM=0	_avm	<code>_AV_MOVE</code>
POT=0	_pot	<code>_POTENTIAL</code>
LOC=0	_loc	<code>_LOCAL_INV</code>
LOC=ns	_locns	<code>_LOCAL_INV_NS</code>
I64=0	_i64	<code>_USE_INT64</code>
CHI=0	_chi	<code>_WRITE_CHI2_DETAIL</code> is ON (the χ^2 components will be written in to the *.chi file for every rejected move)
ARCH=X	X	where X is a number, this adds the X to the end of the file name before extension to differentiate between different architecture, if necessary
INT=0	_i	compile with Intel icc

For example compiling the RMC code for multi-threaded execution with potential usage, summing the *ppcf* and compiling for consecutive execution on a 64-bit architecture use

```
make MULTI=0 POT=0 SUMP=0 ARCH=64
```

This will result in executable named `rmcp_multi_sp64.exe`

Always delete the *.o object files before starting compilation with a new set of options, as make cannot detect in the object files which options were used during their compilation, and it will use the old compilation's object files instead of recompiling them if the source was not modified!

A.4. Compilation under Windows with Microsoft Visual C++

There will be two set of workspace files (*.dsw, *.dsp, *.ncb, *.opt, *.plg), the *: RMC_POT_s is for the standard consecutive version, and the *: RMC_POT is for the parallel version. The parallel version needs the POSIX thread libraries, this is the reason for having two workspaces. Both workspaces use the same header and source files. It makes only sense to use the thread libraries, if you have multiple processors, or your processor is capable of hyper-threading.

The `_MULTI` switch has to be turned off for both workspaces, as the parallel workspace's setting is automatically passes the `_MULTI` option to the compiler!

Choose among the other preferred option switches in the *altern.h* file by turning them off (commenting out) or turning on. Then build the application. The consecutive executable will be named `RMC_POT_s.exe`, and the parallel `RMC_POT.exe`, regardless the chosen option switches.

B. Using 4 or 8 bytes integers for typedef longint

Some variables were defined with the custom type **longint**. This makes it possible, that at compilation time can be decided, whether 4 or 8 bytes integers should be used. The advantages of the 4 bytes integers is that the memory requirements is smaller, but in case of larger systems there can be overflow in the normal or total histogram for example, and the usage of the 8 bytes integers would be inevitable. To complicate the matter, the variable name of the 8 bytes long integers is not part of standard C++, and can depend on the chosen platform/compiler. Therefore the `_USE_INT64` compiler option switches between the two possibilities.

The name of the 64-bit integer is not part of the standard C++, and it differs for Windows and the used Linux platform and compiler. The **typedef** part is in *altern.h*. As a default, **int** type is used for 4 bytes integer. For the 8-bytes integers as default in case of the Windows version `__int64` is used, and in case of the Linux version **long int** is used.

It has to be noted, that the **long int** type in case of some platform/compiler combination can mean only 4 bytes integer, on Linux then try to alter the appropriate part of *altern.h* to **long long int** instead. The actual size used by the program is displayed at the beginning of each run. If your platform/compiler is different, it can happen, that the sizes given here are different for your compilation. It is important to know, that the actual size of the **longint** variables are written at the beginning of the binary *.bcf file, as in that case it is important to know it to be able to read the file correctly, and the program checks it, and only reads the file, if the code with which the file was generated used the same **longint** size, as the present compilation! For downward compatibility, if the size of the **longint** variable is missing from the beginning of the *.bcf file, it tries to interpret it as an old format *.bcf file beginning with the number of atoms and reading it accordingly!

C. About multi-threading

Computationally heavy parts are parallelized, if the program is compiled when the `_MULTI` compiler option is on. The multi-threaded version, uses the Portable Operating System Interface (POSIX) applicable on shared memory multi-processor computers. The workload is equally (as possible) divided among the threads. Parallelization is used during the histogram and its change calculation, the *ppcf*, the initial and modified partial and *total g(r)*, *S(Q)*, *F(Q)* and *E(k)* calculation and the copying of the changed histogram, *ppcf* and partial parts after acceptance/rejection was decided. The local invariance update and the non-bonded potential calculation is also parallelized. The calculation of the neighbour list connected cosine distribution of bond angles is handled exclusively by the main thread.

For the comparison of the standard and the multi-threaded performance efficiency is used:

$$E(p) = \frac{S(1)}{S(p)p} 100\%, \text{ where } p \text{ is the number of processors, and } S(1) \text{ is the elapsed time for the single-}$$

threaded standard and $S(p)$ for the p -threaded application.

POSIX is natively UNIX-LINUX based, but there is a freely available interface, [POSIX for Win32](#), which makes it easy to run the program under WINDOWS.

If you have POSIX, then use the multi-threaded version, this can be used the standard consecutive way as well, so it is recommended to use as many threads as the number of available processors, as it can speed up the calculation. The speedup caused depends very much on the system simulated. If you do not have

multiple processor and POSIX, then you have to be content to compile the program without the `_MULTI` option. The functionality is the same in both cases.

Some additional information about the multi-threading used in RMC_POT and speed test results can be found in the document [RMC_POT_speed_test.pdf](#) or can be downloaded from the [Documentation](#) page of the website.

D. Starting the program

The program can be started by the executable file name followed by the following variation of command line arguments:

- `exename`
- `exename filename`
- `exename filename cont`
- `exename filename y`
- `exename filename cont y`

If the program is started without any command line option, then it will ask for the *filename*, give it without any extension. For example if you have *my_rmc.cfg* and *my_rmc.dat*, then give *my_rmc* for the filename.

'y' indicates to close the window at the end of simulation, which is useful, if the output is redirected to a file, and the program is running in the background.

cont option is for continuation of the run (see II.E).

V. Auxiliary programs

These programs help to create starting configuration and/or *.fnc file. Theoretically RMC started from any initial configuration should reach the same results, only the amount of simulation time is different. Care has to be taken, that at first not too small sigma should be applied (enough 'bad', χ^2 increasing configuration should be accepted), otherwise the simulation can get stuck in a local minimum.

For atomic systems the initial configuration can be some sort of a crystal configuration (for example created by *Crystal*), in this case it is practical to shake the crystal configuration, which can be achieved running a hard sphere RMC simulation without any experimental data, but already using the right hard sphere cutoffs and number density. This will result in a random configuration. Coordination number or average coordination number constraint can be applied, if necessary even in this stage.

To create multi-component random system from a one-component system, then simply the configuration has to be split up into appropriately sized blocks, and as usually the atoms in a block would be close to each other sometimes even causing 'phase separation' in the box due to the creation method a hard sphere RMC with swaps for all the available partial should be run, while the system is adequately mixed.

If the initial configuration should have a certain coordination number around a central atom, then FNC_config can be used for creating the *.cfg and *.fnc file as well. FNC_config can also be used for creating only an *.fnc file based on a *.dat file for a *.cfg file.

For molecules the easiest way to create a prototype of the molecule with the roughly right coordinates in GROMACS *.gro file format (see the GROMACS manual and the web site for details). Then use the GROMACS auxiliary program *genconf* to multiply the molecule in the box. Make sure, that the correct box size is set at the and corresponding to the number density.

The Auxiliary programs except the original GROMACS programs can be found on the Auxiliary programs page of the RMC web site.

A. Creating fcc starting configuration with Crystal

Crystal is a simple fortran program written by Furio Ercolessi, SISSA in 1997. It can create an fcc lattice based on the size of the unit cell (does not really matter, as it will be rescaled), the number of cells in each direction. Keep in mind, that RMC can only handle cubic box! It can perturb the perfect lattice, if random displacement is set to >0. The coordinates generated by this program are in a box from 0→box size, make sure to transform them to reduced coordinates from -1→+1 box (for example it can easily be done in

Excel), and apply the correct RMC version 3 format header, which can be copied from the example given in the validation package. In the RMC header the half box size has to be given in Angstrom! If this (and the number of atoms) does not result in the same number density given in the *.dat file, then the value in the *.dat file will be used, and the simulation box will be rescaled, and this fact is written at the screen output at the beginning of the simulation.

Crystal can only generate a one component system with certain number of atoms ($4*N_{\text{cell}}^3$) as it is an fcc crystal cell, but this can be easily transformed to a multi-component system with any number of atoms, only use the appropriate header, and generate a configuration large enough to hold the required total number of atoms (the extra atomic coordinates at the end of the file will not be used).

B. Using FNC_config

FNC_config can be used two ways. Using option 1 we can create a configuration, when a given number of atoms can be found around the central atoms between the given distance range. For this an RMC format configuration file containing the coordinates of the central atoms has to be present with name *filename.incfg*.

If option 2 is used, then only an *.fnc file is created for a configuration. The configuration file itself is not needed, only based on the structure of this parameter *.dat file gives the rendering of the atoms to the molecules. The concept is very similar to how the RMC indices of the first and second instances of a molecule have to be given in the topology. The structure of the *.dat file should be the following for 10 molecules of dimethylsulfide, the numbering of the atoms id the same as in Figure 3.

Constraint type 1 is for the C-S bond, 2 for C-H bonds, 3 for C-C non-bonding distance, 4 for H-H non-bonding distance and 5 for non-bonding S-H distance.

First 5 lines form a header part, than two lines for each constrained pair, in the first line the index of the constraint type and for the first occurrence the minimum and maximum distance in Å, in the second line the RMC indices of the fnc atom pair in the first and in the second molecule.

```

90          !number of atoms in the configuration
1           !number of molecule types
10          !number of molecules/type following each other in the same line
9           !number of atoms/molecule type following each other in the same line
21          !number of fnc constrained pairs/molecule following each other in the same
line
1 1.638 1.96      !constraint index
1 21 2 22        !indices of the constrained pair in the first and second molecules
1             !constraint index
11 21 12 22      !indices of the constrained pair in the first and second molecules
2 0.944 1.24     !constraint index
1 31 2 34        !indices of the constrained pair in the first and second molecules
2             !constraint index
1 32 2 35        !indices of the constrained pair in the first and second molecules
2             !constraint index
1 33 2 36        !indices of the constrained pair in the first and second molecules
2             !constraint index
11 61 12 64      !indices of the constrained pair in the first and second molecules
2             !constraint index
11 62 12 65      !indices of the constrained pair in the first and second molecules
2             !constraint index
11 63 12 66      ! indices of the constrained pair in the first and second molecules
3 2.36 3.11      !constraint index
1 11 2 12        ! indices of the constrained pair in the first and second molecules
4 1.44 2.0       !constraint index
1 32 34 35       ! indices of the constrained pair in the first and second molecules
4             !constraint index
31 33 34 36      !indices of the constrained pair in the first and second molecules
4             !constraint index
32 33 35 36      !indices of the constrained pair in the first and second molecules
4             !constraint index
61 62 64 65      !indices of the constrained pair in the first and second molecules
4             !constraint index
61 63 64 66      !indices of the constrained pair in the first and second molecules

```

```

4          !constraint index
62 63 65 66 !indices of the constrained pair in the first and second molecules
5 2.04 2.74 !constraint index
21 31 22 34 !indices of the constrained pair in the first and second molecules
5          !constraint index
21 32 22 35 !indices of the constrained pair in the first and second molecules
5          !constraint index
21 33 22 36 !indices of the constrained pair in the first and second molecules
5          !constraint index
21 61 22 64 !indices of the constrained pair in the first and second molecules
5          !constraint index
21 62 22 65 !indices of the constrained pair in the first and second molecules
5          !constraint index
21 63 22 66 !indices of the constrained pair in the first and second molecules

```

C. Converting configuration files from GROMACS *.gro: convert_gromacs

Converting configuration between RMC and GROMACS in both ways can be done by the `convert_gromacs` software. It uses a parameter file and the initial configuration file and produces the other type coordination file. It can convert consecutive GROMACS configurations from the same file to separate RMC files, or consecutive RMC (*coll.cfg) to GROMACS as well. In case of RMC_GROMACS conversion a starting topology file can be created as well, if the `fnc` option is 1. The constrained distances related to bonds can be converted to bonds, the 1-3 non-bonded distances to angles, and other distances more than two bonds away can be converted to type 2 constraints (see the GROMACS manual about bonds, angles and constraints).

Only the bond and angle types has to be given for a molecule, not all the bonds and angles. So if there are more `fnc` pairs in a molecule with the same `fnc` constraint, then only one has to be given, even in the case of the angles, specify one central atom only for an angle constraint. The program will determine all the others.

```

1 3          ! conversion type (0) RMC->MD, (1) MD->RMC number of conf. to
  convert
DMeS_flex_md.cfg 3 ! RMC configuration file, number of RMC types only for MD->RMC
  conversion
DMeS_flex_md_3conf.gro ! GROMACS *.gro file
DMeS.top          ! GROMACS *.top file
DMeS all-atom     ! Title for the MD
ffoplsaa.itp 1 1 ! force field parameter file; bond type, angle type: 'D' for
  default

0 0.0103135      ! whether to rescale box (0: no, 1: yes), new number density,
  if necessary

1              ! number of different residue types
1              ! character string (max 3 char) for each residue defining X in
  the bonded and nonbonded constraint name definition cbX_index and ncbX_index12:30
  11/15/2008

1331 9 DMeS      3 ! as many line as residue types containing first the number of
  this residue, number of atoms and the name (max 5 char) for the residue, number of
  bonds for exclusions (default=3)

1 2 CA  opls_209 12.011 1 -0.013 1 ! The index of this atom
  in the conf. in the first then in the second molecule, atom name, atom type, mass,
  charge group, charge,RMC-type(only MD->RMC)
31 34 HA1 opls_140 1.008 1 0.06 3
32 35 HA2 opls_140 1.008 1 0.06 3
33 36 HA3 opls_140 1.008 1 0.06 3
21 22 S   opls_202 32.060 2 -0.334 2
11 12 CB  opls_209 12.011 3 -0.013 1
61 64 HB1 opls_140 1.008 3 0.06 3
62 65 HB2 opls_140 1.008 3 0.06 3
63 66 HB3 opls_140 1.008 3 0.06 3

0 c2cl4.fnc     ! whether to use FNC, name of the FNC file (0:no, q1:use FNC lower
  limit, 2: use FNC upper limit, 3:use FNC average limit)
2              ! number of bond types to create for each residue

```

```

1 2          ! constraint indices in the FNC file for each desired bond const. for each
  residue
185769.6 284512.0 ! force constants for each desired bond constraints separated by
  space
3           ! number of angle types to create for each residue
3 4 5 ! constraint indices in the FNC file for each desired angle const. for each
  residue
518.816 276.144 292.88! force constants for each desired angle constraints separated
  by space
5 1 1! index of the central atom in the molecule (according to topology) for each
  desired angle const. for each residue
99.05 109.3 109.3      ! desired angle in degree for each desired angle const. for
  each residue09:45 11/15/2008
0           ! number of non-bonded constraint types to create
  for each residue
10 11 10 11          ! constraint indices in the FNC file for
  each desired non-bonded const. for each residue

```

For more detailed description see the document [RMC-GROMACS_converter.pdf](#).

D. Converting RMC configurations to CrystalMaker text format *.cmtx

Both the text type and the binary configurations file of RMC_POT can be converted to the text type *.cmtx file of the visualization software CrystalMaker. The converter is self-explanatory, and needs a parameter file to contain the conversion details.

It can use the *.cfg, *.bcf or *_coll.cfg file for RMC coordinate file input, the last one being the text type configuration file produced with RMC with collecting the configurations options containing multiple configurations consecutively.

It is possible to convert only a part of the configurations, if not all the atoms required for the visualization, this is governed by the mode option, there are 4 possible ways to select the atoms.

- option 1: The indices of the specified number of atoms for each type will be spread among the atoms of this type with as equal step as possible. As usually atoms with similar indices are close to each other, especially if no swaps were used during the simulation, this way the atoms can be found everywhere in the simulation box.
- option 2: The first given number of atoms will be used from a type. (Most probably atoms close to each other are chosen this way!)
- option 3: Atoms of given indices specified at the end the parameter file will be selected. (Index can range from 1 -> ntotal, and has to follow the order of atom types.)
- option 4: Only particles between two parallel planes will be saved into the file. The planes have to be parallel to one of the sides of the simulation box! There have to be an integer after option 4 to specify which axis is perpendicular to the cutting plane (x:0, y:1, z:2), followed by two real numbers between -1 and +1) to specify the reduced coordinates, between which the atoms will be outputted. Example for this line of the parameter file choosing the atoms, with reduced y-coordinates are between -0.7 -> 0.5: 4 1 -0.7 0.5.

If all the atoms of a configurations has to be selected, then the quickest way to do it to use option 2 and give the number of all the atoms of each type to use.

An example parameter file is given here:

```

ccl4.cfg      ! name of the input configuration file
5            ! number of configurations to convert (only meaningful for *coll.cfg)
2            ! number of used types
3 1 -0.7 0.5 ! mode option (for option 4 index of the perpendicular axis(x:0, y:1,z:2),min, max reduced coord)
C 2          ! Atom's chemical symbol, number of atoms to use for each type
Cl 8        ! Atom's chemical symbol, number of atoms to use for each type
1           ! Atom indices to select, only in case of option 3
2           ! Atom indices to select, only in case of option 3
2049       ! Atom indices to select, only in case of option 3
4097       ! Atom indices to select, only in case of option 3

```

6145	! Atom indices to select, only in case of option 3
8193	! Atom indices to select, only in case of option 3
2050	! Atom indices to select, only in case of option 3
4098	! Atom indices to select, only in case of option 3
6146	! Atom indices to select, only in case of option 3
8194	! Atom indices to select, only in case of option 3

The green part has to be included only for option 4, the blue for option 3, the number of atoms denoted by pink only for options 1-3.

If no atoms of an atom type *ha* to be used in case of option 1-3, than give zero for the number of atoms.

The name of the output file will be generated by the input file by replacing the original file extension by **.cmtx*. The program will decide based on the original file extension about the type of the input file, so it is important, that the original RMC file extensions should be used, (**.cfg* 1 text type configuration, **.bcf* 1 binary configuration, **_coll.cfg* multiple text configurations. In case of multiple configurations atoms of the first *nconfig* configuration will be selected for output, and put into separate files with name **_coll000X.cmtx*.

The program can be started by **executablename** then it asks for the parameter file name, or by **executablename parfilename**.

¹ McGreevy, R.L., Pusztai, L.: *Molec. Simul.* **1**, (1988) 359.

² Pusztai, L.: *J. Non-Cryst. Sol.* **227-230**, (1998) 88.

³ McGreevy, R.L.: *J. Phys.: Cond. Matter* **13**, (2001), R877.

⁴ Evrard, G., Pusztai, L.: *J. Phys.: Cond. Matter.* **17**, (2005) S1.

⁵ Gereben, O., Jóvári, P., Temleitner, L., Pusztai, L.: *J. Optoelectron. Adv. Mater.* **9**, (2007) 3021

⁶ M J Cliffe, M T Dove, D A Drabold, A L Goodwin (2010) *Phys Rev Lett.* **104**, 125501.