RMC++ for RMCA users

Guillaume Evrard[1], SzFKI, Budapest, 16th of May 2003.

RMC++ is an implementation of the Reverse Monte Carlo Algorithm written in C++. It basically follows the same scheme as the existing RMCA (Fortran) code. Externally, the differences between the two codes are kept to a minimum, so that RMCA users should be able to use RMC++ without much additional effort. New features of the 'standard RMC' version (i.e. with one atom moved at a time) are optional.
Hereafter, *practical* similarities and differences between the two codes are explained.

1 Standard Input

The standard input of RMC++ takes the same form as the RMCA input, i.e.:

- a configuration file (`.cfg`);
- a run file (`.dat`);
- the data files: PPCF's or diffraction (neutron or x-ray);

1.1 Input files format

The formats of the `.dat`, `.cfg`, and data files are the same as for RMCA, i.e. they correspond to the description given in the RMCA manual[2]. In principle, RMC++ input files are strictly indentical to the RMCA files. However, practice has shown that the RMC++ routines used to read the files are less tolerant than those of RMCA, in particular, concerning **separators** and **comments**.
It is important to stick to the format described in the following, for the `.dat` file in particular (see example data file below):

- keep the comments after data values, even for multiple data sets.
- the range of data points to be used for each data set must be separated by a 'space' or a 'tab' character, NOT by a comma (,).
- keep the blank lines in all files, do not add blank lines
- you can add whatever you wish at the end of the `.dat` file, this is not read.

This is not very much user friendly, but once you have a set of working files, it is easy to use it as a template for making other files.

1.2 Starting RMC++

The easiest way to use RMC++ is to put all the necessary files (executable, `.dat`, `.cfg`, data files) in the same folder, and start the executable.
When starting RMC++, the console asks for the generic name of the files used. The relevant extensions (`.cfg`, `.dat`, . . .) are added automatically by the code when necessary.
Histograms are computed, unless a RMC++ histogram file (`.hgm`) file is present, and coordination constraints (CC) are not used.
Please note:

- There are no density checks, or date checks as in RMCA, to recalculate the histograms.
- Files for the histograms (`.hgm`) and the output (`.out` file) are created if they do not exist already, and updated otherwise.
- the configuration (`.cfg`) file will be modified by the run (as in RMCA), do not forget to make a copy before the run if you want to keep your starting configuration!

[1]E-mail: evrard@power.szfki.kfki.hu
[2]available on the Studsvik webpage: www.studsvik.uu.se

2 Output

2.1 During the run

At the start of the run, the programme reads data, possibly computes the new histograms, *etc.*.. The sequence of the initialisation phases is prompted, and the run parameters are displayed before the programme enters the 'main loop'.
During the main loop, the χ^2 values are displayed every n steps (n defined in the `.dat` file), with each component (the value used in the Metropolis test, the χ^2 per number of points for each data set, and the coordination constraints add-ons).
Configurations, histograms and a `.out` file are saved at regular time intervals (defined in the `.dat` file) as in RMCA and at the end of the run.
nota bene: the RMCA option 'collect several configurations' is not implemented at present.

2.2 The .out file

At regular intervals defined in the `.dat` file, and when the programme leaves the main loop, the histograms are saved (`.hgm` file), as well as the configuration (`.cfg`), and a `.out` file is produced as in RMCA. Note however, that the `.out` file is not strictly identical as its RMCA counterpart, although it can still be used by e.g. RMCPLOT.
The major difference is the **ordering of the partials**, which is
1-1, 2-1, 2-2, 3-1, 3-2, 3-3, 4-1,..., 4-4,..., p-1,..., p-p.
instead of
1-1, 1-2, 1-3,...,1-p, 2-2, 2-3,..., 2-p,..., p-p
as in RMCA.
If there are **more than one data set**, then the `.out` file will include the partials (either PPCFs or structure factors) for each data set (the reason is that RMC++ allows input data values to be taken at different r or q values). As in RMCA, the calculated and the (renormalised) experimental data are also saved, so that finally the `.out` file includes:

- the PPCF partials at each histogram bin r value;
- for each neutron data set, the Structure Factors partials at the experimental q values;
- for each x-ray data set, the Structure Factors partials at the experimental q values;
- the RMC++ and (renormalised) input $g(r)$'s, for each $g(r)$ set;
- the RMC++ and (renormalised) experimental $S(q)$ for each neutron data set;
- the RMC++ and (renormalised) experimental $S(q)$ for each x-ray data set.

3 RMC++ options

RMC++ includes the possible use of some algorithmic options detailed below, and the writing of an optional `.hst` file recording the run history. All the information needed to activate these options are stored in a `.add` file. If this file is not found, then the default values are used (standard RMCA), and no history file is written.

3.1 The fixed-neighbours constraints

The most important option is the possible use of fixed neighbours constraints (FNC), that bind groups of atoms together to mimick molecules. The binding consists in the definition of a minimum and a maximum value for the distance between a neighbour and a central atom. This feature is not new[3], and is of great interest for molecular systems.

3.1.1 Numbering of atoms

In RMC++, each atom can be identified by its *index*, *i.e.* its position in the `.cfg` file: the index of the first atom is 1, the index of the second atom is 2,..., the index of the last atom is N.
If atomic types are to be distinguished, the first n_1 atoms are of type 1, the following n_2 atoms are of type 2, ...etc....

[3]The FNC's have been introduced in 1996 as a 'patch' to the rmca code by László Pusztai, and extensively used, but never documented (!). In the corresponding programme (**rmc_fi**) the network of atoms is defined in a **.fnc** file (RMC++ uses the same format), and there is an additional line introduced in the **.dat** file.

3.1.2 Definition of FNCs

One FNC is defined by the set of two numbers: the minimal and the maximal allowed distance for this constraint. The numbering of the FNC's is not linked to the atom types, but it is defined by the original sequence of extremal distances defined in the `.fnc` file.

3.1.3 The .fnc file

The `.fnc` file contains the defining distances of each constraint and the network of linked atoms in the following way:

- every 'central' atom is indicated by its index in the configuration
- the next number indicates the number of 'neighbours' this central atom has
- then the configuration indexes of each neighbour are indicated
- finally for each neighbour, the type of the FNC is indicated

Such a sequence reads for instance

```
2919    4
871         4967         7015         9063
1 2 2 2
```

meaning that atom number 2919 has 4 neighbours, of indexes 871, 4967, 7015 and 9063 linked to it by the constraints number 1, 2, 2 and 2 respectively.

- Note that all atoms must be indicated sequentially as central atoms in the `.fnc` file, possibly with 0 neighbours if they are not linked.
- The total number of atoms (redundant) must be indicated at the beginning of the `.fnc` file

See the example file in Sec.6.3.

3.2 The initial shift of the histograms

RMC++ allows a more precise definition of the histograms than RMCA. In RMCA, the histogram bins are defined by: $[(i-0.5)\Delta_x, (i+0.5)\Delta_x[$ with $i = 1$ to `nbins` (number of histogram bins). The r value associated with each bin is $i\Delta_x$, and distances greater than the configuration box half-length are not used.
RMC++ uses histograms bins defined by $[(x_0+i)\Delta_x, (x_0+i+1)\Delta_x]$ ($i = 0$ to `nbins-1`), with the r value associated with each bin (in the `.out` file) equal to $(x_0+i+0.5)\Delta_x$.
The default value for x_0 is 0, but RMC++ allows the user to adjust it (e.g. $x_0 = 0.5$ to mimick RMCA).

3.3 The maximum distance used

RMC++ allows the use of all the possible interatomic distances in the cell, by using the appropriate normalisation. More precisely, RMC++ allows to set the greatest used distance up to the theoretical maximum of $\sqrt{3}\,L$, where L is the box half-length. Actually, a value greater than 1.5 L is not recommended since the number of distances is then too low to provide a good statistical accuracy. Furthermore, there might be some normalisation artifacts at $\sqrt{2}\,L$.
This maximum distance is set by its ratio to L, so that the default value is 1.0, and a value of 1.41 is a 'safe' optimal value.
Note: If a value greater than $\sqrt{2}$ is used, RMC++ needs the file '`sfactorcube`' to be put in the same file as the executable, in order to run. This file contains the discretised values of the normalising factor to be used beyond $\sqrt{2}\,L$.
Also note that the number of histograms bins is set by the range of used distances and the bin width, and adjusted so that the last bin is fully included in the distance range.

3.4 The run history

For the sake of comfort and analysis of the RMC runs, an optional `.hst` file can be produced by the run. It records

- the number of generated moves
- the number of tried moves
- the number of accepted moves
- the CPU time spent in the Metropolis loop
- the 'instant' ratio tried/generated moves
- the 'instant' ratio accepted/tried moves
- the total χ^2 value (NOT divided by the number of points)
- the χ^2 component for each data set
- the χ^2 component for each coordination constraint

These numbers are taken at regular intervals defined in the following way: the usual run information on screen is displayed every `iprint` generated moves (`iprint` is defined in the `.dat` file -see the RMCA manual-), the history information above will be collected every `isave` *screen display updates*, i.e. every `isave * iprint` generated moves. The number `isave` is defined in the `.add` file.
These numbers are stored in an array (history buffer) of chosen size and are being flushed to the disk when the buffer is full, and also at the end of the programme. The size of the buffer is defined in the `.add` file.
'Instant' ratios are calculated over the last `iprint` generated moves.
Note that as in RMCA, there is a display only if the number of accepted moves is greater or equal to `iprint` AND the last generated move is accepted. If no move is accepted (*e.g.* because FNC's or cut-offs are not satisfied), there is no regular display after the initialisation phase.

3.5 The optional .add file

All the information needed to activate the above options are stored in the optional `.add` file. If you want some options to be used, this file must be put in the same folder as the `.cfg`, `.dat`, etc... If the file is not found, the program will run with the default values for the bin shift and the maximum distance range, FNC's will not be used and no history file will be written.
The `.add` file must contain the following information

- the FNC boolean switch (0 or 1)
- the initial bin shift (in Δr units).
- the maximum distance to be used (in L units)
- the number of atoms moved in a single move (this should be 1 for standard RMC)
- the size of the history buffer (0 yields no `.hst` file, otherwise 200 is a good value)
- a boolean switch for custom move (this should be set to 0 for standard RMC)

The same recommendations as for the `.dat` file apply to the `.add` file, i.e.: keep the initial comment line (you can change it, but do not suppress it), similarly keep the comments after each value.

4 What RMC++ can/cannot do

At the moment, there are some RMCA features that are not implemented in RMC++:

RMC++ cannot

- use a non-cubic cell;
- collect configurations;

- use a potential.

On the other hand, RMC++ has been implemented so that

- it is now possible to use $\{[r, g(r)]\}$ or $\{[q, S(q)]\}$ data sets with different number of data points and even different r or q values,
- the FNC's have priority over the cut-offs, *i.e.* it is possible to distinguish *intra*molecular distances from*inter*molecular distances.
- the `moveout` option has also been noticeably improved (see below).

5 Differences between RMCA and RMC++

The `moveout` option allows one to move preferably atoms too close to each other (i.e. not satisfying the cutoffs). In RMC++, if the `moveout` option is switched on in the `.dat` file, 1 move in 10 concerns these atoms, which are deleted from the 'too close' list as soon as they satisfy the cutoffs. When all atoms satisfy the cutoffs, the .moveout. option is switched off automatically.

For the same configuration, the χ^2 values computed by RMCA and RMC++ can be very different. This is because the χ^2 is very sensitive to variations in the (calculated) data. Differences in RMCA and RMC++ calculated data originate in the different ways the histograms are computed and normalised. This in turns yields differences in the structure factors at very low or very high q-values, to which the χ^2 is very sensitive. Nevertheless, significant features of configurations obtained by RMCA and RMC++ should remain unchanged.
The sensivity of the resulting configuration to the algorithmic parameters such as the way the histograms are defined and normalised, and the way the sine-Fourier transform is approximated remains to be investigated. Similarly, possibilities of assessing the uncertainties on conclusions driven from RMC simulations are still to be studied.

6 Examples of input files

As noted above, the RMC++ input format is identical to the RMCA input format. Note however that commas (,) must NOT be used as separators. They must be replaced by 'space' or 'tab' or 'new line' characters. Comments after values must be kept, even in the case of multiple data sets. Comments lines and blank lines at the beginning of the files must be maintained, and no additional comment line is allowed (except at the end of the files).

The `.cfg` file is described in the RMCA manual.

6.1 The .dat file

The `.dat` file is identical to the one described in the RMCA manual, excepted that the comments are **necessary** if the file is to be read correctly.

```
CCl4
0.0319                  ! number density
3.3 1.69
2.7                    ! cut offs
0.1 0.1                 ! max moves
0.1                     ! r spacing
.false.                   ! whether to use moveout option
0                        ! no of configurations to collect
500                     ! step for printing
2 2                     ! time limit, step for saving
0 1 0 0                  ! no. of g(r), neutron, xray and exafs expts
c4.sq
1 100                     !range of points
0.000                    ! constant to subtract
  0.0218    0.2520
  0.7262 ! partial coefs
```

```
0.001                    ! standard deviation
.true.                   ! whether to vary amplitudes
.true.                   ! whether to vary constant
0                        ! no of coordination constraints
0                        ! no of average coordination constraints
.false.                  ! whether to use a potential
```

Note to **rmc_fi** users: users of the `rmc_fi` (*i.e.* the version that includes the FNC's) will notice that the added line in the rmc_fi `.dat` file for the FNC switch must be suppressed (*i.e.*, back to standard RMCA `.dat` format) and be put in the `.add` file.

6.2 The .add file

The `.add` file for RMC++ options must have the following format

```
!this file contains optional parameters for a RMC++ run!
1 !FNC switch
0 !initial bin shift;
1.41 !xmax used in the run
1 !number of atoms moved in a single move
200 !size of the history buffer (0->no history record, 200 is good)
5 !number of display updates between each history buffering
0 !indicator of custom move
```

6.3 The .fnc file

This is an example of `.fnc` file. Keep the blank lines at the beginning!

```
FNC for H2O+LiCl

 No. of possible rmin-rmax pairs:
  2
    9.000000E-01        1.400000
        1.100000        1.700000

       11000

           1           0
           2           0
           3           0
...
        1999           0
        2000           0
        2001           2
        5001        8001
           1           1
        2002           2
        5002        8002
           1           1
...
10999           2
        4999        7999
           1           2
       11000           2
        5000        8000
           1           2
```