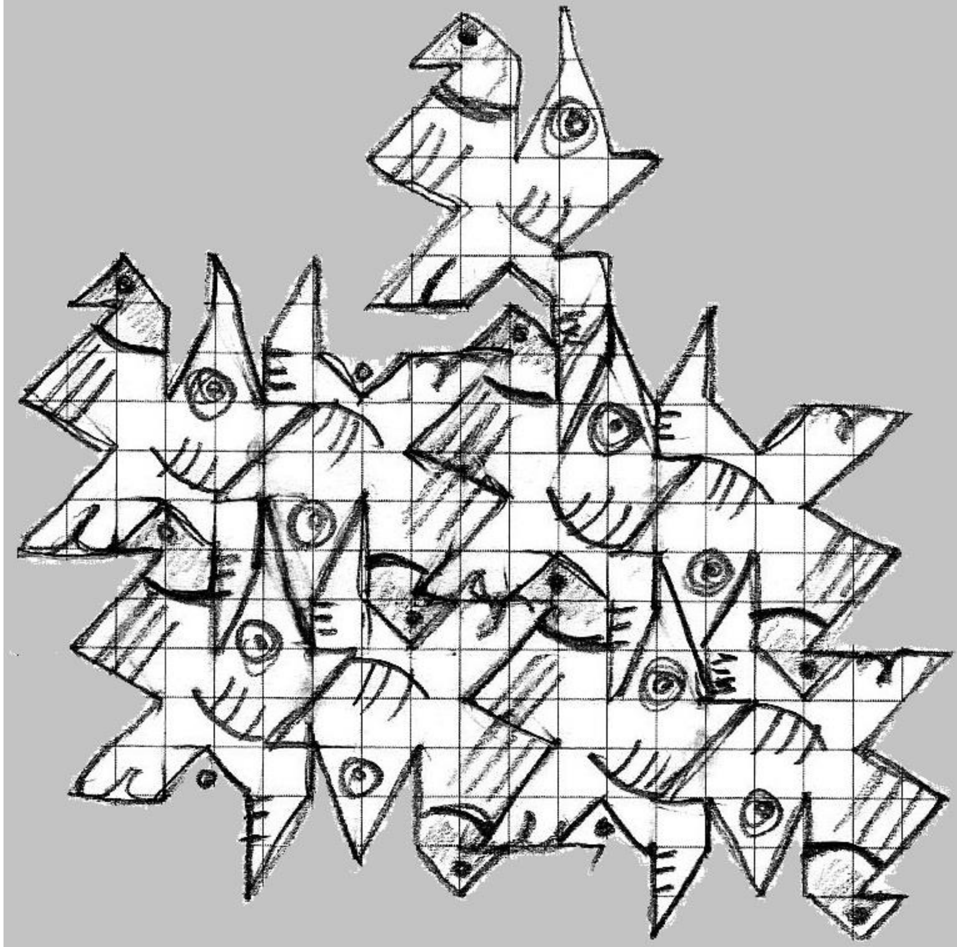
MiCMoS



Milano Chemistry Molecular Simulation

User Manual v.2.3

MiCMoS
Milano Chemistry Molecular Simulation
Description and User Manual
Release 2.3, July 30, 2024

Leonardo Lo Presti
leonardo.lopresti@unimi.it

Dipartimento di Chimica
Università degli Studi di Milano

MiCMoS is a set of Fortran computer programs for the simulation of the condensed states of non-polymeric organic materials, using empirical potential energy schemes, *ab initio*-derived semiempirical energy calculation schemes; in a static approach, analyzing only one point in phase space, or in an evolutionary approach for phase-space sampling using Monte Carlo (MC) and classical Molecular Dynamics (MD) techniques. The platform includes modules for the appropriation of crystallographic information with standardization of hydrogen-atom positions, for molecular model building, for the calculation of lattice energies and the analysis of crystal packing, as well as for the equilibrium or dynamic simulation of bulk or cluster-like liquids, crystals and solutions with trajectory analysis. MiCMoS evolves out of previously presented platforms, CLP-Pixel and CLPDyn, and like its predecessors is user-oriented with opportunity or necessity for human intervention in the proceedings.

The original CLP-Pixel and CLPDyn cores, later evolved in MiCMoS, come from the forty-year research work of prof. Angelo Gavezzotti (retired; agavezzotti@gmail.com), to whom we are indebted for constant support and encouragement.

The platform includes subdirectories *batch*, with running macros, *Tutorials*, with documentation and worked examples, *exe*, with compilation macros and executables, and *SourceA* and *SourceB*, with source codes. All these directories and their documents and codes are accessible by the user for possible modification.

All programs run from MS-Dos (Windows) or from any textual user interface environment (Unix-Linux machines). All input-output is numerical in text files and there are no pull-down menus or selection windows. There is no graphics module in the package, but interfacing to one of the many available graphics platforms is easy as structural results are displayed in a *.dat* or *.xyz* file format.

MiCMoS is oriented toward the MD simulation of small- to medium-size organic molecules in condensed phases from a crystallographic viewpoint. At variance with other MD software devoted to the simulation of biomolecules, it focuses on intermolecular interactions, solvation and molecule-molecule recognition.

The MiCMoS project is subject to continuing development. Comments, criticism and suggestions from users are most welcome, especially concerning inconsistencies or unclear procedures.

List of MiCMoS Contributors

Angelo Gavezzotti
Leonardo Lo Presti
Luca Sironi
Giovanni Macetti
Silvia Rizzato

What’s new

Release 2.3 2024, July, 30 2024

- Due to the incompatibility of newer Windows versions (10 and 11) with some MiCMoS executables, a new procedure is suggested to run the program on Windows machines, which essentially relies on the installation of a Unix-compatible environment (Cygwin).
- New **post-processing routines** were added. More in detail:
    - The routine *vanhove.for* performs the van Hove analysis of the trajectory, focusing on molecular center of mass.
    - The routine *trajedit.for* allows to edit any MC or MD trajectory, for example by selecting specific frames, extracting specific solute/solvent molecules, changing the coordinate system or the origin of the simulation box.
    - The routine *renergy.for* applies LJC or CLP intermolecular potentials to a previous trajectory. This can be useful, for example, if one wants to test another Force Field on a trajectory previously run, or to perform the energy analysis of any subgroup of molecules, after extraction with *trajedit*.
    - The routine *denflu.for* computes time-averaged local density fluctuations.
    - The routines *clusters.for* and *conta.for* analyze a trajectory to find persistent intermolecular clusters, which may be defined based on either purely geometric or energetic criteria.
- Implemented more efficient compilation instructions to **speed up the parallel MD engine** up to 20-30 % in Linux environments.
- Implemented more efficient algorithms for the calculation of intermolecular Lennard-Jones potentials and the handling of molecules within the cutoff limit **to speed up the MD engine** by a further 15-20 % (both Windows and Linux environment).
- Fixed some bugs and errors (see below).

The changes affect only the MD part of the package, and specifically the libraries *mdlibs.for* and *mcmdpo.for*. A minor adjustment was also made in the routine *solution.for*, to reduce the odds of unwanted steric clashes at the beginning of the simulation. The module *correl.for* was updated.

In this manual: New Sections 8.10-8.14 were added to describe full input and output options of the new routines. Section 5.7 (Solution module) was updated. The Windows part of Installation Notes was rewritten, and the corresponding file system of the program updated.

Release 2.2 2023, July, 30 2023

- A routine was added for MD simulations in **confined environments** (nanolayer, nanotube and nanocavity).
- Fixed some bugs and errors (see below).

The changes affect only the MD part of the package (*mdlibs.for*, *mdmain.for* and *mdviri.for*). Bugs were corrected in *mdlibs.for*. Minor adjustments were also done in *redene.for* and *mclibs.for,* to handle a *.ene* file with format compatible with the MD part.

In this manual: A new Section 5.8 was added to describe the preparatory routine (*confbox.for*) for the definition of the simulation box for confined spaces. A new Section 7.2.5 was added to describe the confinement algorithm. Section 7.6.2 (description of the MD run control file) was updated to include information on the new parameters in the input stream.

IMPORTANT (only for the MD part). Due to these changes, a further option (inano) must be added now at the end of the second parameter line in the MD control file (*.mdi*). If missing, the molecular dynamics program *mdmain.for* will stop with an I/O error. See Section 7.2.5 for details. An example of input with full parameters is shown at the end of Section 7.6.2.

Release 2.1 2022, September 30, 2022

- A **Linux-parallel version** of the Molecular Dynamics (MD) module is now available. The program can be still used in serial mode.
- **Empirical anharmonic correction** to **bending** motion was introduced in MD.
- New features in the part A of the package (static lattice analysis):
    - A new crystallographic analysis program, ***Statimpa***, was introduced. It scans a series of static crystal structures in *.oeh* format, searching for short atom-atom contacts, including hydrogen bonds.
    - ***Retcor*** now prints also the structure of the isolated asymmetric unit in Cartesian coordinates using the standard *.xyz* format. This is useful for visualization purposes with external programs.
- A new interface for structural files, ***Solution***, is included. When used in conjunction with *boxliq.for* it allows to easily produce solution (solute+solvent) boxes with the desired solute concentration.
- Increased atom limits in *naverag.for* (now 35,000).
- Fixed some bugs and errors (see below).

Parallelization affects only the MD part of the package (*mdlibs.for*, *mdmain.for* and *mdviri.for*). A patch was added to *retcor.for* to write the new *.xyz* file and the *run.retcor* batch file were updated accordingly. The bugs were corrected in *datgro.for*, *mdlibs.for*, *mdmain.for* and *naverag.for*.

In this manual: The Installation notes section was updated to describe the new features, especially as concerns the new parallel-MD executables. **A more user-friendly installation procedure for Windows users is also proposed**. Section 1.3 was updated with the new features of ***Retcor***. New Sections 1.5 (***Statimpa***) and 5.7 (***Solution***) were added. New Section 7.4.1.2 was added (anharmonic corrections to bending). Sections 7.4.1.1 (anharmonic corrections to stretching) and 7.6.2 (MD run control file) were edited to explain the new features. A warning on the ***Geomet*** usage was added in Section 8.1. Section 8.2 (***Analys***) was also updated for a slight correction in the input stream.

Release 2.0 2021, July 31, 2021

- New **crystallographic utilities** were introduced:
    - *naverag.for*, to produce a spacetime average structure from a whole trajectory;
    - *debye.for*, to compute the total Debye scattering from a MD trajectory file;
    - *nanocut.for*, to produce a nanoparticle of the desired shape, that is, bound by specific crystallographic planes.
- **A biased Molecular Dynamics (MD)** algorithm was added to **simulate aggregation phenomena**.
- **Empirical anharmonic correction** to **stretching** motion was introduced in MD.

- A new interface for structural files, ***Nanosolv***, is included. When used in conjunction with *nanocut.for* and *boxliq.for* it allows to easily produce solvated nanoclusters of the desired dimensions and shape.
- Fixed some bugs and errors (see below).

IMPORTANT (only for the MD part). Due to these changes, two more options must be added at the end lines #3 and #6 in the MD control file (*.mdi*). If missing, the molecular dynamics program *mdmain.for* will stop with an I/O error. More in detail, an integer parameter ibias is now the last entry of line #3 and controls whether a biased MD run is required or not. Another integer, ianh, must be added at the end of line #6 to specify whether the stretching potential will be fully harmonic or not. See Sections 7.2.4, 7.4.1.1 and 7.6.2 for details. Some input examples with the complete set of parameters are shown at the end of Section 7.6.2.

The changes affect only the MD part of the package (*mdlibs.for*, *mdmain.for* and *mdviri.for*), while the bugs were corrected in *mclibs.for*, *mcmain.for*, *mdmain.for* and *geomet.for*.

In this manual: New Section 5.6 was added (new structural interface ***Nanosolv***). Sections 7.2.4, 7.4.1.1 and 7.6.2 (MD run control file) were either added or edited to include and explain the new features. New Sections 8.7–8.9 were added (crystallographic utilities).

Release 1.2 2021, January 31, 2021

- A **new integration algorithm** (velocity–Verlet) is available.
- A **new thermostat** (Bussi–Donadio–Parrinello) is available.
- Fixed some minor bugs and errors (see below).

The changes affect only the MD part of the package (*mdlibs.for*, *mdmain.for* and *mdviri.for*), while the bugs were corrected in *mclibs.for* and *boxcry.for*.

In this manual: Sections 7.2 (thermostats and integration algorithms) and 7.6.2 (The MD run control file) were edited to add information on the new options.

IMPORTANT (only for the MD part). Due to these changes, two more options (Emolim and iengt) must be added now at the end of the first parameter line in the MD control file (*.mdi*). If missing, the molecular dynamics program *mdmain.for* will stop with an I/O error. See Section 7.6.2 for details. An example of input with full parameters is shown at the end of Section 7.6.2.

Release 1.1 2020, July 30, 2020

A new routine has been added to the general Parrinello–Rahman barostat to perform MD simulations with an external, user–defined stress field (S. Rizzato, A. Gavezzotti & L. Lo Presti, Crystal Growth Des. 2020, 20,7421–7428). The new instructions are meaningful only for periodic structures. The change affects the MD part of the package (*mdlibs.for*, *mdmain.for* and *mdviri.for*).

In this manual: See new Section 7.3.4 for details. Section 7.6.2 (The MD run control file) was edited accordingly to add information on the new options.

Fixed known bugs and errors

MiCMoS is an ongoing open-source project that could benefit from the feedback by the community. Please report any bug and error you may find to leonardo.lopresti@unimi.it

Release 2.3 2024, July, 2024

(i) In release v2.2, a set of instructions was erroneously erased in the *mcmdpo.for* library, which prevented the correct evaluation of solvent-solvent intermolecular potentials. The error affected both the Monte Carlo and Molecular Dynamics engines. Solute-only simulations were not affected. The error is specifically present only in the v2.2 release and did not affect the previous versions of the program (up to v2.1).

(ii) In the program *distrib.for*, the labels of center of mass Radial Distribution Function between solute and solvent and that between solvent and solvent were erroneously switched in the output. This error affected only the labels of the distributions, which however were correctly computed.

(iii) Due to a missing instruction, in *mdmain.for* and *pmdmain.for* the number of degrees of freedom of the solute was overestimated by +3, only in the absence of solvent. This minor discrepancy does not affect the MD trajectories significantly, as the number of degrees of freedom is normally much larger.

(iv) Corrected some example files in Tutorials T3, T6, T7, T8 whose format was not up to date with respect to the current MiCMoS requirements.

Release 2.2 2023, July, 2023

(i) Due to a misplaced instruction in the subroutine *roteva* (*mdlibs.for*), the molecular centre of mass of tethered molecules in nanodroplets was incorrectly computed, resulting in unphysical deformations of the molecules themselves. The bug affected only Molecular Dynamics simulations of nanodroplets.

(ii) Due to a repeated instruction in the *retopo* routine of *mdlibs.for*, the atom-atom Coulomb contributions between solvent molecules in molecular dynamics of solutions were overestimated. The error did not affect the solute molecules.

(iii) In this manual, description of `irvel` options 1 and 2 (Section 7.6.2) were exchanged. The error was now corrected.

(iv) In equation (7.24) of this manual (virial of the forces), a ½ factor was missing. Note that the virial was computed correctly in the main code.

Release 2.1 2022, September 30, 2022

(i) Corrected a typo in the error printing routine for `ierr=8` in *naverag.for*.

(ii) Corrected a typo in *mdmain.for* (last printouts, total energy changes): `deuv=ecouv-ecouvzz` was changed into `deuv=ecouv-ecouvz`. The error affected calculations of deuv only when two molecular species were present.

(iii) Changed all `sqrt()` instructions in *mdlibs.for* into `dsqrt()` to uniform double precision throughout.

(iv) An instruction was corrected in *mdlibs.for* (sobroutine *Retopo*) when the program renormalizes charges for kJ/mol conversion. When no solvent is present, the program printed neither molecular information, nor it echoes the topology file. The correction influences only the output in the printfile *.pri*, not calculations.

(v) In *mclibs.for* (subroutine *writen*) the output cell density became infinity if no box was present (`boxvo=0`). A patch was added to fix the problem; if no box is present, now the cell density is reported as zero.

(vi) Contrary to what is said in the manual, giving 0 0 as starting and ending frame number in the *debye.inp* command file for ***Debye*** (Section 8.8) did not allow for automatic calculation on the whole trajectory (or at least on the first 1,000 frames). A patch has been added to the source code to fix the problem.

(vii) In the utility program *analys.for* (Section 8.2), if no calculation of centre of mass-based radial distribution function $g(R)$ is required, the program has no C_{pack} for the calculation of an approximate box volume, and atom-atom $g(R)$ is not calculated. Fixed by reading C_{pack} before questions on types of $g(R)$. Section 8.2 in this manual was updated accordingly.

Release 2.0 2021, July 31, 2021

IMPORTANT: The 2.0 version fixes a previously undetected error, that prevented MiCMoS from correctly computing nonbonded **intramolecular** potential energy contributions. This problem emerged when the parent program, CLPdyn, was evolved into the present package and some blocks of instructions had to be updated. The error affects only systems that require nonbonded pairs to be defined (see NLISTU/NLISTV entry in Section 7.6.4), that is, flexible molecules prone to intramolecular steric clashes.

(i) An error was discovered in *mclibs.for* (subroutine *eintra*), which prevented the Monte Carlo engine from correctly computing the nonbonded intramolecular energies. More in detail, a residual “go to” statement from a previous algorithm in the parent CLPdyn program suppressed the torsional part of the intramolecular energy. At the same time, intramolecular Coulomb contributions were underestimated due to an incorrect scaling factor. This error affected only the Monte Carlo part and is now fixed.

(ii) An error was discovered in the main Molecular Dynamics engine (*mdmain.for*). The damping parameter FACTIN (Sections 7.4.1, 7.6.2 and 7.6.4) for computing the nonbonded intramolecular energy contributions was not properly defined in the memory allocations. Therefore, the nonbonded contributions were always zero or undefined. <u>The error affected only the Molecular Dynamics part</u> and is now fixed.

(iii) A bug was fixed in *geomet.for*. The program failed to handle more than 20 torsions due to an erroneous format statement. Now *geomet* can handle up to 60 different torsions.

(iv) A bug was fixed in the main Monte Carlo engine (*mcmain.for*): due to a couple of misplaced instructions, the actual number of solute and solvent molecules, computed for checking purposes, could have been undefined in certain circumstances.

(v) The memory limits of *datgro.for* were upgraded to a maximum of 80,000 atoms to handle also very large simulation boxes.

(vi) The *excbox.for* service program was updated to be fully compatible with new modules.

<u>Release 1.2 2021, January 31, 2021</u>

(i) An error was discovered in the Molecular Dynamics input instructions, Section 7.6.2, line #3, instructions idstr and Emolim) concerning the calculation of histograms of intermolecular interaction energies (see Section 7.5.3). The Emolim parameter (upper energy limit for histogram calculation) was not read as expected right after idstr on the same input line. Rather, the program expected to find it in a new line if idstr=1. The error was fixed in the code (*mdmain.for*). The instructions in Section 7.6.2 are now correct.

(ii) There was a little formatting mismatch in the MC and MD *writra* routines, which made slightly different the trajectory printout of the Monte Carlo module with respect to that of the Molecular Dynamics module. This implied that, for example, the utility *datgro.for* (conversion of MiCMoS trajectory *.dat* files to *.gro* format) could not properly read Monte Carlo trajectories. The bug was fixed in *mclibs.for*. Now the two formats are equivalent and *datgro.for* is fully compatible with both MD and MC trajectories.

(iii) A bug was fixed in *boxcry.for*: when two molecular units (“solute” and “solvent”) were present in the crystal, the program failed to correctly account for slave atoms when the crystalline simulation box was built. The error affected only the Monte Carlo part.

(iv) A bug was fixed in the library *mclibs.for* for Monte Carlo simulations (subroutine *writen*): the program failed to correctly compute the cell density.

License

Index

How to cite

A general survey of the MiCMoS features: A. Gavezzotti, L. Lo Presti and S. Rizzato, *CrystEngComm*, 2022,24, 922-930 (https://doi.org/10.1039/D1CE01360B)

Specific applications and features (please select those that best fit your needs):

- **PIXEL module (Pixel method):** A. Gavezzotti, *Mol. Phys*. 2008, 106, 1473–1485, https://doi.org/10.1080/00268970802060674.
- **Monte Carlo calculations:** A. Gavezzotti, *New J. Chem.* 2011, 35, 1360-1368, https://doi.org/10.1039/C0NJ00982B; A. Gavezzotti, *New J. Chem*, 2013, 37, 2110-2119, https://doi.org/10.1039/C3NJ00181D.
- **Molecular dynamics of liquids:** A. Gavezzotti and L. Lo Presti, *New J. Chem*. 2019, 43, 2077–2084, https://doi.org/10.1039/C8NJ05825C.
- **Molecular dynamics of solids:** A. Gavezzotti and L. Lo Presti, J. Appl. Cryst. 2019, 52, 1253–1263, https://doi.org/10.1107/S1600576719012238; A. Gavezzotti, L. Lo Presti and S. Rizzato, *CrystEngComm* 2020, 22, 7350-7360, https://doi.org/10.1039/D0CE00334D.
- **Simulations under external stress field:** S. Rizzato, A. Gavezzotti and L. Lo Presti, *Crystal Growth Des*. 2020, 20,7421–7428, https://doi.org/10.1021/acs.cgd.0c01098
- **Kinetic bias algorithm:** L. Lo Presti, S. Rizzato and A. Gavezzotti, *Crystal Growth Des*. 2022, 22, 1857-1866, https://doi.org/10.1021/acs.cgd.1c01410
- **Simulation in confined spaces**: L. Sironi, G. Macetti and L. Lo Presti, *Physical Chemistry Chemical Physics* 2023, 25, 28006-28019, https://doi.org/10.1039/D3CP02886K
- **Aggregation and clustering**: L. Sironi, G. Macetti and L. Lo Presti, *J. Mol. Liquids* 2024, *Under review*.

MiCMoS applications

Recent works that profitably employed some MiCMoS (or MiCMoS-related) features

L. Sironi et al., *Nanoscale Inhomogeneities in Undercooled Benzoic Acid: A Molecular Dynamics Study*. J. Mol. Liquids **2024**, *Under review.*

L. Sironi *et al.*, *Molecular Dynamics Investigation of Benzoic Acid in Confined Spaces*. Physical Chemistry Chemical Physics **2023**, 25, 28006-28019, https://doi.org/10.1039/D3CP02886K

G. Macetti *et al.*, *Classical molecular dynamics simulation of molecular crystals and materials: old lessons and new perspectives*, in "*Comprehensive Computational Chemistry*" Elsevier, Book series in Theoretical and Computational Chemistry n°21, **2023**; https://doi.org/10.1016/B978-0-12-821978-2.00107-0

L. Sironi *et al.*, *Why Is α-D-Glucose Monomorphic? Insights from Accurate Experimental Charge Density at 90 K*. Cryst. Growth Des. **2022,** 22, 11, 6627–6638; https://doi.org/10.1021/acs.cgd.2c00846

A. Gavezzotti, *Crystallography without Crystals: A Structural Study of Fakein*. Helv. Chim. Acta 2022, 105, e202200059; https://doi.org/10.1002/hlca.202200059

A. Gavezzotti, *Dynamic simulation of orientational disorder in organic crystals: methyl groups, trifluoromethyl groups and whole molecules*. Acta Cryst. **2022**, B78, 333-343; https://doi.org/10.1107/S2052520621012191

R. Destro *et al.*, *Anharmonic Thermal Motion Modelling in the Experimental XRD Charge Density Determination of 1-Methyluracil at T= 23 K*. Molecules **2021**, 26(11), 3075; https://doi.org/10.3390/molecules26113075

A. Gavezzotti, *Collective Variables for the Simulation of Crystallization of Organic Compounds: Some Case Studies*. Israel J. Chem. **2021**, 61, 498-506. https://doi.org/10.1002/ijch.202100039

A. Gavezzotti, in "*The Crystalline States of Organic Compounds*", Elsevier, Book series in Theoretical and Computational Chemistry n°20, **2021**. https://doi.org/10.1016/B978-0-12-823747-2.00004-4

Tutorials

Some tutorials to get acquainted with the main program features and the I/O procedures are freely available online. Step-by-exercises are proposed and thoroughly explained, with pertinent reference to this Manual. If you are a new user, it is **strongly recommended** that you follow at least the basic tutorials before using extensively MiCMoS. Tutorial 1 will allow you to quickly learn how to handle the file interface; Tutorials 2–5 are not strictly necessary if you are interested in Monte Carlo or Molecular Dynamics. Tutorials 6–8 describe Monte Carlo in detail, while Tutorials 9 and 10 will give you the basics of the Molecular Dynamics procedures. Have also a look to Tutorial 11 to see how to analyze trajectories. Tutorial 12 proposes advanced exercises focused on molecular nanoparticles.

The material can be found at this link: https://sites.unimi.it/xtal_chem_group/index.php/tutorials. A list is given below.

Tutorial	Topic	Purpose
1	Generation of structural data files	It is described how a *.cif* crystallographic information file is interpreted by the ***Retcif***, ***Retcor*** and ***Retcha*** sequence to produce a *.oeh* structural file, ready for subsequent calculations (static lattice, MC or MD).
2	LJC parametrization and charge density file	It is described how to prepare a *.oeh* structure file compatible with the LJC parametrization, that is, containing Electrostatic Potential (ESP) charges rather than the default Extended Hückel ones. At the same time, the production of a charge density file to be used in subsequent ***Pixelc*** calculations is also shown.
3	Using the ***Retcor*** module as a molecule builder	It is described how to build chemically sound molecular models for systems whose atomic coordinates are unknown for subsequent use in Monte Carlo and Molecular Dynamics calculations.
4	Calculation of atom-atom lattice energies	It is described how to use module ***Crysaa*** to compute static atom-atom interaction energies and lattice energies.
5	Lattice energy by ***Pixelc***	It is described how to use module ***Pixelc*** to compute static charge density-based interaction energies and lattice energies
6	Monte Carlo on liquids (rigid mol)	It is described how to prepare and equilibrate a liquid box of benzene with the Monte Carlo technique.
7	Monte Carlo on liquids (flexible mol)	It is described how to prepare and equilibrate a liquid box of n-pentane with the Monte Carlo technique. The aim is also to get the user acquainted with the use of slave atoms for the Monte Carlo routine of MiCMoS.
8	Monte Carlo on crystals	It is described how to prepare and equilibrate a periodic crystal of acetanilide with the Monte Carlo technique. The aim is also to get the user acquainted with the use of slave atoms for the Monte Carlo routine of MiCMoS.
9	Molecular Dynamics of liquids	It is described how to prepare and equilibrate a very simple liquid (benzene) with the Molecular Dynamics technique.
10	Molecular Dynamics on crystals	It is described how to prepare and equilibrate the $P2_12_12$ RT phase of pyridone with the Molecular Dynamics technique.
11	Analyzing MC and MD trajectories	It is described how to employ service programs ***Geomet***, ***Analys***, ***Distrib***, ***Correl***, ***Redene***, ***Datgro***, ***Naverag*** and ***Debye*** programs to analyze MC and MD trajectories.
12	Molecular Dynamics of molecular nanoparticles	It is described how to prepare isolated (non-periodic) simulation boxes that include either free or solvated nanoparticles of small organic molecules.

Installation notes

To install MiCMoS, you need a valid Fortran compiler, such as `gfortran`. All modules are written in featureless, unspecific Fortran that can be compiled in Unix and Windows 32/64 bit environments. No machine libraries are required (‘`-static`’ option of the compiler). MiCMoS is provided as a compressed `.zip` archive. Upon un-zipping the following directories must appear:

- `MiCMoS`
  - o `SourceA` (interface programs and static energy calculations)
  - o `SourceB` (Monte Carlo and Molecular Dynamics machines)
    - ▪ `parallel` (parallel version of libraries and molecular dynamics module)
  - o `Inputs` (templates of input files with steering parameters for specific MiCMoS programs)
  - o `Unix`
    - ▪ `batch` (macros to run MiCMoS programs in Unix/Linux syntax)
    - ▪ `compile` (macros to compile MiCMoS programs in Unix/Linux syntax)

Under Windows machines, please follow the instructions detailed in the next paragraph. In a Unix/Linux environment, copy the archive in your `~/programs` directory. If need arises to store the `MiCMoS` folder in any other location, remember to update the path in both installation and batch macros (see below).

Four installation macros are available into the “`compile`” subdirectory:

- `compileA`: compiles preliminary modules for interpreting crystallographic information files (*.cif*; ***Retcif, Retcor, Retcha***), the modules for lattice energies (***Crysaa, Pixmt2, Pixelc***) and for atom-atom contact analysis (***Statimpa***).
  This macro also produces a `~/programs/MiCMoS/doc/static` (Unix) empty directory, which might be employed to store charge density files (extension *.den*) for ***Pixelc*** calculations.
- `compileB`: compiles Monte Carlo, Molecular Dynamics and their service modules;
- `compileMC`: compiles only the Monte Carlo module and its libraries;
- `compileMD`: compiles only the Molecular Dynamics module and its libraries.

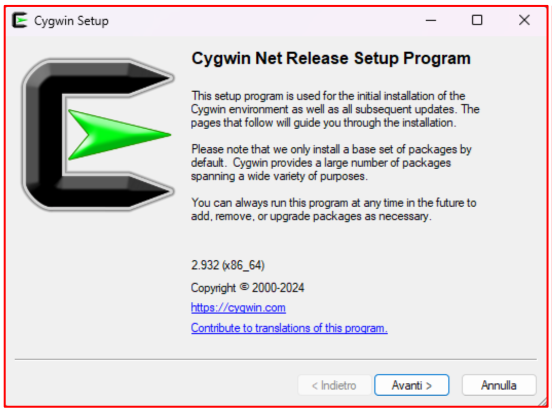
Running `compileA` and `compileB` in sequence ensures complete installation.

Preliminary operations for Windows users only

Sadly, the modern versions of Windows (Win10 and Win11) are incompatible with several key features of MiCMoS. Apparently, the programs are compiled, but the system stubbornly refuses to start some executables, especially those that are dependent on static libraries. This is the case of the Monte Carlo and Molecular Dynamics engines for example.

The procedure detailed in the previous versions of the manual (up to v2.2) worked seamlessly under Windows up to version 7. From MiCMoS v2.3 onwards, the following procedure is recommended.

Download the Cygwin setup (setup-x86_64.exe) from https://www.cygwin.com/. In these notes, the 64 byte utility is shown, but you could download the 32 byte version if you know it is more appropriate for you. Execute the self-installing tool and select "Install from internet":



Cygwin Setup

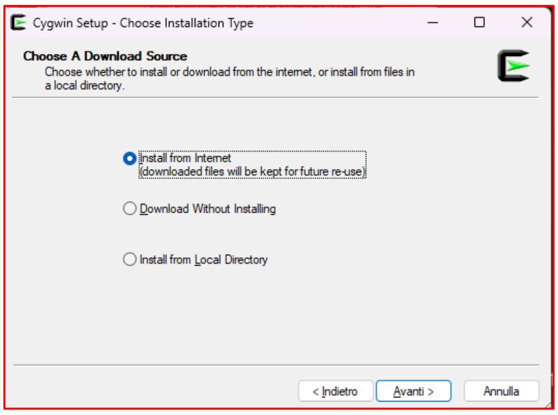
Cygwin Net Release Setup Program

This setup program is used for the initial installation of the Cygwin environment as well as all subsequent updates. The pages that follow will guide you through the installation.

Please note that we only install a base set of packages by default. Cygwin provides a large number of packages spanning a wide variety of purposes.

You can always run this program at any time in the future to add, remove, or upgrade packages as necessary.

2.932 (x86_64)
Copyright © 2000-2024
https://cygwin.com
Contribute to translations of this program.

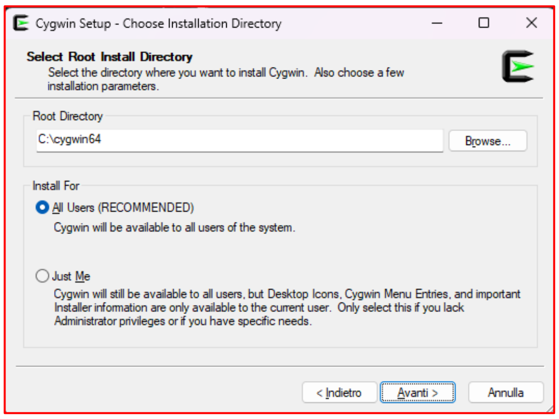


Cygwin Setup - Choose Installation Type

Choose A Download Source
Choose whether to install or download from the internet, or install from files in a local directory.

(●) Install from Internet (downloaded files will be kept for future re-use)
( ) Download Without Installing
( ) Install from Local Directory

Select the main directory for the Cygwin environment. The default C:\cygwin64 works fine. Then, choose the directory you wish to use to keep installation files (C:\Cygwin for example). These files may be cancelled after the installation; however, if you have enough disk space, you should keep them to facilitate any future update or change of configuration of the Cygwin environment.

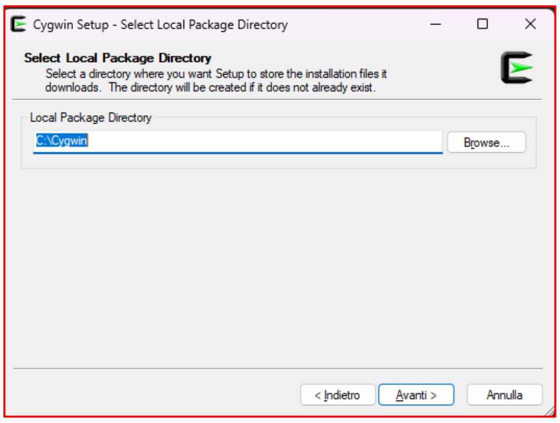


Cygwin Setup - Choose Installation Directory

Select Root Install Directory
Select the directory where you want to install Cygwin. Also choose a few installation parameters.

Root Directory: C:\cygwin64

Install For
(●) All Users (RECOMMENDED)
Cygwin will be available to all users of the system.
( ) Just Me
Cygwin will still be available to all users, but Desktop Icons, Cygwin Menu Entries, and important Installer information are only available to the current user. Only select this if you lack Administrator privileges or if you have specific needs.

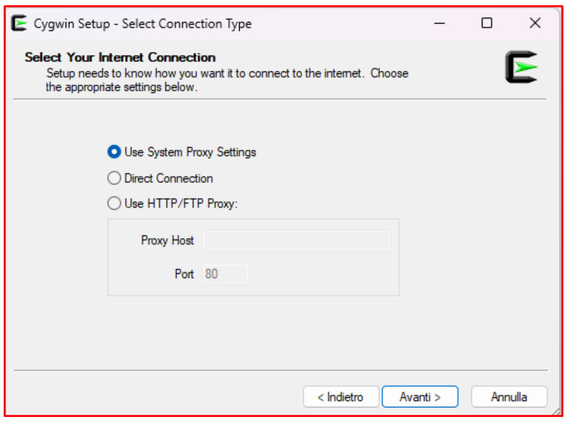


Cygwin Setup - Select Local Package Directory

Select Local Package Directory
Select a directory where you want Setup to store the installation files it downloads. The directory will be created if it does not already exist.

Local Package Directory: C:\Cygwin

Next, you will be prompted to select your favourite internet connection (keep the default settings) and choose a valid mirror to go ahead with the download of Cygwin system files. A good choice is https://sourceware.mirror.garr.it.



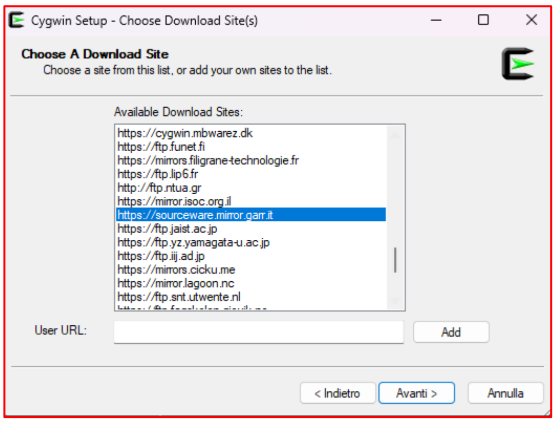
Cygwin Setup - Select Connection Type

Select Your Internet Connection
Setup needs to know how you want it to connect to the internet. Choose the appropriate settings below.

[✓ Use System Proxy Settings]
Direct Connection
Use HTTP/FTP Proxy:

Proxy Host
Port 80

< Indietro | Avanti > | Annulla



Cygwin Setup - Choose Download Site(s)

Choose A Download Site
Choose a site from this list, or add your own sites to the list.

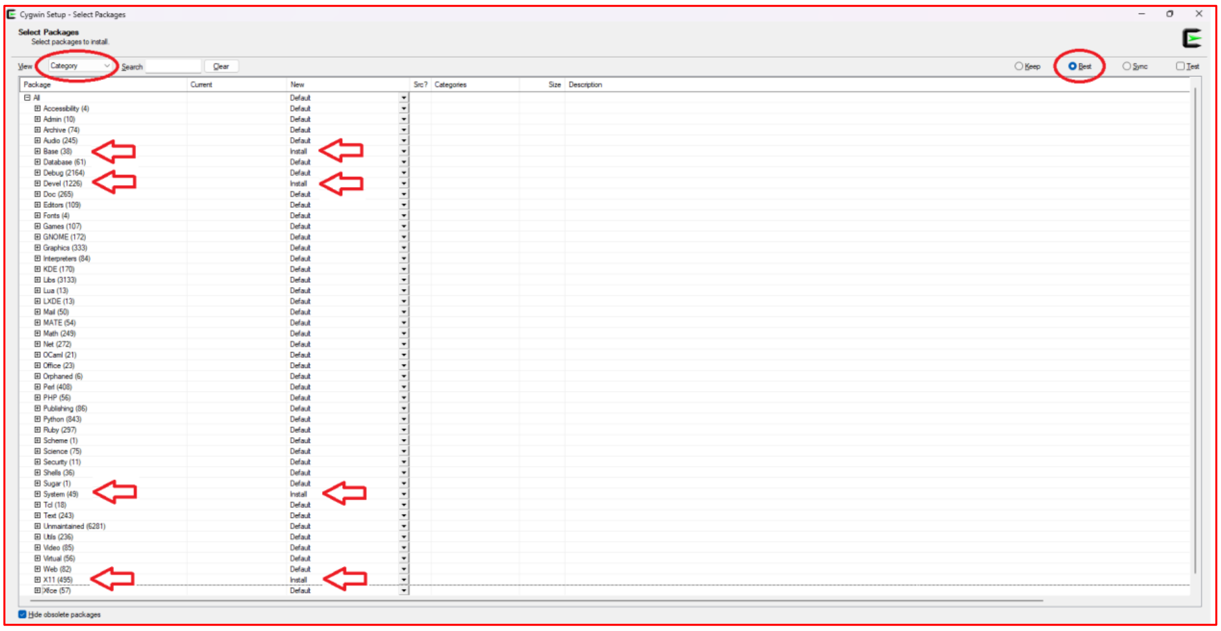
Available Download Sites:

https://cygwin.mbwarez.dk
https://ftp.funet.fi
https://mirrors.filigrane-technologie.fr
https://ftp.lip6.fr
http://ftp.ntua.gr
https://mirror.isoc.org.il
[✓ https://sourceware.mirror.garr.it]
https://ftp.jaist.ac.jp
https://ftp.yz.yamagata-u.ac.jp
https://ftp.iij.ad.jp
https://mirrors.cicku.me
https://mirror.lagoon.nc
https://ftp.snt.utwente.nl

User URL:
Add

< Indietro | Avanti > | Annulla

In the next prompt, you must select what packages you wish to install. Make sure that you visualize the "Category" classification and use your mouse to mark the following packages with the "Install" flag: Base, Devel, System, X11. This choice ensures full operability of MiCMoS and keeps the whole installation as cheap as possible. Obviously, you are free to customize the Cygwin installation according to your specific needs.



Cygwin Setup - Select Packages

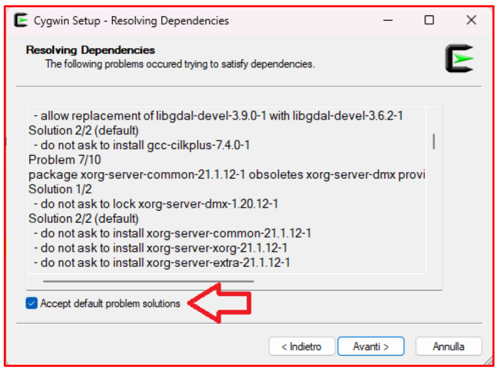
Select Packages
Select packages to install.

View: Category   Search   Clear   Keep [✓ Best] Sync   Test

Package	Current	New	Src?	Categories	Size	Description
⊟ All		Default				
⊞ Accessibility (4)		Default				
⊞ Admin (10)		Default				
⊞ Archive (74)		Default				
⊞ Audio (245)		Default				
⊞ Base (38)		Install				
⊞ Database (61)		Default				
⊞ Debug (2164)		Default				
⊞ Devel (1226)		Install				
⊞ Doc (265)		Default				
⊞ Editors (109)		Default				
⊞ Fonts (4)		Default				
⊞ Games (107)		Default				
⊞ GNOME (172)		Default				
⊞ Graphics (333)		Default				
⊞ Interpreters (84)		Default				
⊞ KDE (170)		Default				
⊞ Libs (3133)		Default				
⊞ Lua (13)		Default				
⊞ LXDE (13)		Default				
⊞ Mail (50)		Default				
⊞ MATE (54)		Default				
⊞ Math (249)		Default				
⊞ Net (272)		Default				
⊞ OCaml (21)		Default				
⊞ Office (23)		Default				
⊞ Orphaned (6)		Default				
⊞ Perl (408)		Default				
⊞ PHP (56)		Default				
⊞ Publishing (86)		Default				
⊞ Python (843)		Default				
⊞ Ruby (297)		Default				
⊞ Scheme (1)		Default				
⊞ Science (75)		Default				
⊞ Security (11)		Default				
⊞ Shells (36)		Default				
⊞ Sugar (1)		Default				
⊞ System (49)		Install				
⊞ Tcl (18)		Default				
⊞ Text (243)		Default				
⊞ Unmaintained (6281)		Default				
⊞ Utils (236)		Default				
⊞ Video (85)		Default				
⊞ Virtual (56)		Default				
⊞ Web (82)		Default				
⊞ X11 (495)		Install				
⊞ Xfce (57)		Default				

[✓ Hide obsolete packages]

As the Cygwin project is under continuous development and you are asking to download full file systems under each "category", it is possible that the installer will find unwanted cross-dependencies: in the next prompt, flag the "Accept default problem solutions" button (see next page) and go ahead. The installation will start. The whole process may take a while, depending on the velocity of your internet connection and of your processor.

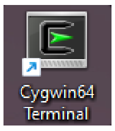


The following problems occured trying to satisfy dependencies.

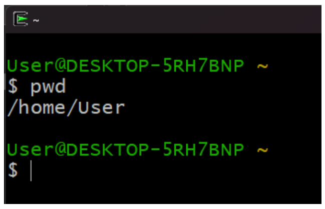
```
 - allow replacement of libgdal-devel-3.9.0-1 with libgdal-devel-3.6.2-1
Solution 2/2 (default)
 - do not ask to install gcc-cilkplus-7.4.0-1
Problem 7/10
package xorg-server-common-21.1.12-1 obsoletes xorg-server-dmx provi
Solution 1/2
 - do not ask to lock xorg-server-dmx-1.20.12-1
Solution 2/2 (default)
 - do not ask to install xorg-server-common-21.1.12-1
 - do not ask to install xorg-server-xorg-21.1.12-1
 - do not ask to install xorg-server-extra-21.1.12-1
```

[✓] Accept default problem solutions

Finally, you will end up with this desktop icon:



Double click on the Cygwin icon. A terminal window will open. The terminal will open into the default user home, which is dependent on the root Cygwin directory defined above. For example:



```
User@DESKTOP-5RH7BNP ~
$ pwd
/home/User

User@DESKTOP-5RH7BNP ~
$ |
```

This home directory has full path C:\cygwin64\home\User in Windows, where "User" is your specific Win user.

Now, create a "programs" directory in /home/User.

```
mkdir programs
cd programs
```

Copy the MiCMoS compressed archive in "programs". In a normal Windows environment, you may simply drag the MiCMoSv2.3.*zip* archive from your download directory. Extract the files with any archive extractor you have at hand (WinRaR, 7zip, WinZip…). You are finally ready to install MiCMoS. Follow the same installation instructions For Linux/Unix Users below. To use the program, you must keep working in the Cygwin terminal.

To check what Cygwin packages are installed, enter the following command in the terminal:

```
cygcheck -c -d | sed -e "1,2d" -e 's/ .*\$//' > packagelist.txt
```

This will create "packagelist.txt" ASCII text file with the full list of installed packages, together with their version. If, for some reason, you want to change something, please execute again the Cygwin setup program. Good luck!

For Unix/Linux users

Go to the `MiCMoS` folder for compilers, usually `~/programs/MiCMoS/Unix/compile`. Attribute execution permissions to your macros; for example:

```
chmod 755 run.compile*
```

Since Release 2.1, the Molecular Dynamics (MD) module is compiled in both serial and parallel mode under Unix/Linux. This implies that two versions of the MD engine, `mdmain.for`, now exist. The parallel one is called `pmdmain.for`.

Compilation

The compilation command is `run.compileX`, X being A, B, MC or MD. Both the serial and parallel versions of the MD part are compiled.
Installation macros expect that source files are stored in `~/programs/MiCMoS/SourceA` and `~/programs/MiCMoS/SourceB`. The only exception is the program ***Pixmt2***, whose source code should be placed in `~/programs/MiCMoS/Unix`. If file locations are different, you should update the path into the macros.

As above, the following commands

```
./run.compileA
./run.compileB
```

ensure full installation, with MD compiled both in parallel and serial mode.

CAUTION (WINDOWS): Sometimes, the installation macro prompts some error messages “`Fatal Error: Can't open module file 'omp_lib.mod' for reading at (1): No such file or directory`”. The error is due to a missing omp_lib.mod module in the Cygwin environment – this likely depends on the choices made by the installer to resolve internal conflicts in Cygwin. As under Windows the parallel execution of MiCMoS is *not* supported, you can safely ignore this error and proceed with the serial version only. If the error is shown, the only effect is that the pmdmain executable for parallel Molecular Dynamics will be missing in your MiCMoS/exe directory. You can always perform MD simulations in serial configuration (see “Serial execution” below).

Executables will be put into a `~/programs/MiCMoS/exe` folder, which will be created by the macro itself if not already present. Executables might be loaded into the `~/bin` directory if desired; however, note that macros in the `Unix/batch` directory expect to find executables into `~/programs/MiCMoS/exe`.

Serial execution

The batch macros are fully compatible with the previous versions of MiCMoS. To use the various program modules, make executables all the batch macros contained in `~/programs/MiCMoS/Unix/batch` with command

```
chmod 755 run.*
```

Then, you may copy whatever of them in your working directory. As for MD, the batch command is always `run.mdmain`, which is identical to the macro of the previous releases of MiCMoS.

When you execute a batch file, note that all the necessary files, such as data and input instructions, **must** be present as well in your working directory. Refer to the detailed descriptions of the programs and modules in this Manual to use them properly.

Parallel execution

Only the Molecular Dyanamics (MD) routines were parallelized. All the other programs, including Monte Carlo, are still serial and the usual batch commands can be executed seamlessly.

Parallelization was carried out through OpenMP libraries and is fully operational in the `pmdmain.for` code. Differences with respect to the serial version (`mdmain.for`) concern calls to openMP-embedded functions, like get_omp_max_threads(), which can be recognized by the compiler only when the openMP flags are active. This is obtained through the compiler option `-fopenmp`, in conjunction with `-frecursive` that allows indirect recursion by forcing all local arrays to be allocated on the stack.

CAUTION: Note that the parallel MD executable is named *pmdmain*.

The macro `run.pmdmain` is available to execute *pmdmain.for* locally (not recommended in a distributed architecture environment: see below). The runfile differs from the usual `run.mdmain` one (see Section 7.6.1) only in the first instruction:

```
export OMP_NUM_THREADS=<N>
```

Where `<N>` is the actual number of threads you want to employ, for example: `export OMP_NUM_THREADS=3` or `export OMP_NUM_THREADS=4` will distribute the workload to 3 or 4 threads, respectively.

For usage in a distributed environment, please conform to the queuing system of your parallel architecture. Two job scripts (`job.p1-mdmain` and `job.p2-mdmain`) are available in the `~/programs/MiCMoS/Unix/batch` directory, which call *pmdmain* in either a SLURM or PBS context. A couple of examples asking for a run up to 1 hour long on 4 threads are shown in the next page for your convenience. You should edit the job script that best fits your needs according to what you want to do (for example, change strings “yoursubstance”, “yourbox” and “youroutput” in the script with the actual filenames you need. As usual, the two job scripts must be made executable before use through

```
chmod 755 job.p*
```

Testing was carried out on a local cluster equipped with nodes mounting 2x Intel® Xeon® CPU E5-2650v2 @ 2.60GHz, 8 cores and 64 GB RAM. Runs on the pyridone crystal at room temperature using the same starting box as in MiCMoS Tutorial 10 (200 molecules, 12 atoms each, 2,400 atoms, see https://sites.unimi.it/xtal_chem_group/index.php/tutorials/28-tutorial-10) showed that a 5,000 MD steps-long run with 10 threads allows a time gain of ~ 4 times.

SLURM workload manager (`job.p1-mdmain`):

```
#!/bin/bash --login
#======================================================================
#SBATCH -J job_name
#SBATCH --nodes=1 --ntasks=4
#SBATCH --time=01:00:00
#SBATCH --partition yourqueue
#======================================================================
#
cp yoursubstance.mdi mdyn.mdi
cp yoursubstance.top mdyn.top
cp yourbox.dat mdyn.bxi
cp barrier.par mdyn.par
~/programs/MiCMoS/exe/pmdmain
rm youroutputmdc.dat
mv mdyn.mdc youroutputmdc.dat
rm youroutputmdo.dat
mv mdyn.mdo youroutputmdo.dat
rm youroutputmd.pri
mv mdyn.mdp youroutputmd.pri
rm youroutputmd.ene
mv mdyn.ene youroutputmd.ene
rm youroutputbias.tab
mv bias.tab youroutputbias.tab
rm mdyn.mdi
rm mdyn.top
rm mdyn.bxi
rm mdyn.par
```

PBS scheduler (`job.p2-mdmain`):

```
#!/bin/bash --login
#======================================================================
#PBS -N job_name
#PBS -l nodes=1:ppn=4
#PBS -l walltime=01:00:00
#PBS -q yourqueue
#======================================================================
#
cp yoursubstance.mdi mdyn.mdi
cp yoursubstance.top mdyn.top
cp yourbox.dat mdyn.bxi
cp barrier.par mdyn.par
~/programs/MiCMoS/exe/pmdmain
rm youroutputmdc.dat
mv mdyn.mdc youroutputmdc.dat
rm youroutputmdo.dat
mv mdyn.mdo youroutputmdo.dat
rm youroutputmd.pri
mv mdyn.mdp youroutputmd.pri
rm youroutputmd.ene
mv mdyn.ene youroutputmd.ene
rm youroutputbias.tab
mv bias.tab youroutputbias.tab
rm mdyn.mdi
rm mdyn.top
rm mdyn.bxi
rm mdyn.par
```

Part A

Crystal lattice energy calculations

1. Preparation of structural data files

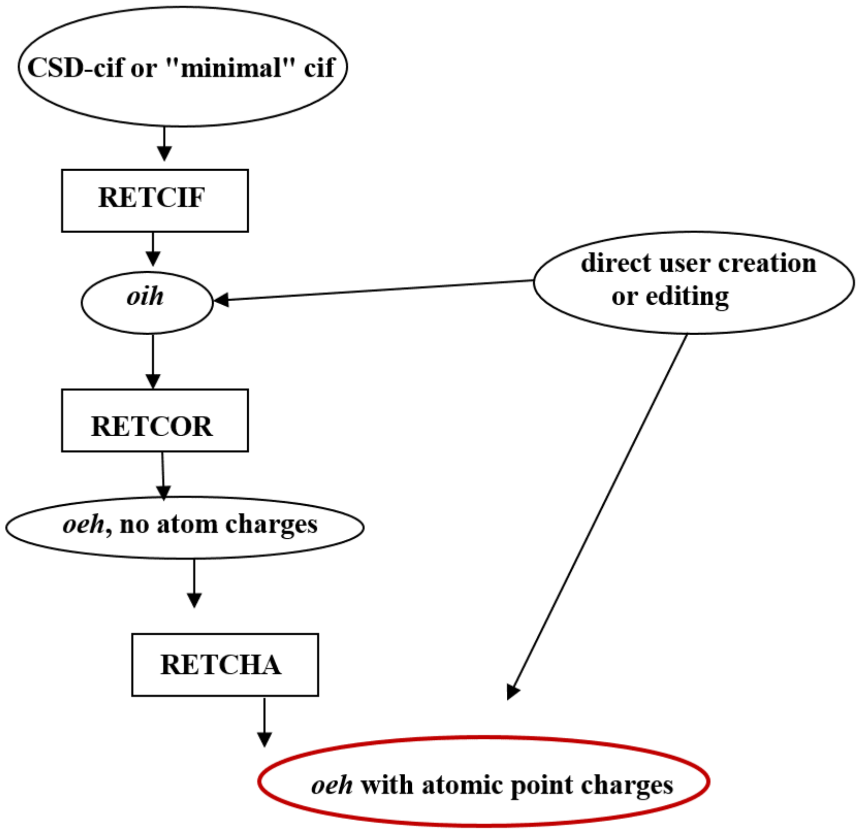


Figure 1. Block diagram of the organization of preliminary modules. Items in square boxes are programs, items in round boxes are files with *italic* extension names. CSD is the Cambridge Structural Database. The final file *.oeh* is the key structure-data file of the whole package.

1.1 The *Retcif* module: Data retrieval from CSD or from reduced cif files

Module ***Retcif*** retrieves crystal structure data from files in *cif (crystallographic information file)* format. *Retcif* is designed only for the particular *cif* file organization of standard Cambridge Structural Database (CSD) entries; the flexibility of a *cif* file is in this case a disgrace. It is easy however to convert any user *cif* file, or any user-generated list of atomic coordinates in any format, into a "minimal *.cif*' format" compatible with ***Retcif***. Technical details and full description on how the algorithm works are available in the Appendix, Section A1.

CAUTION: On running Conquest, the main CSD structure search engine, please ensure that the "*Additional CIF data items*" checkbox is flagged in the on-screen menu section "*Select options*" reachable through "*File/Export entries as...*". This way, ***Retcif*** will be able to interpret correctly the parameters. If a *.cif* file contains multiple structures, they will be all printed sequentially in the output *.oih* file.

Normally, X-ray atom positions are discarded and H atoms are replaced according to standard geometrical rules; the output *.oih* file has coordinates for non-H atoms and symbolic codes for the generation of standardized H-atom positions ("implicit hydrogen"). The module can operate even if no hydrogen-atom coordinates are present in the *.cif* file. The assignment of atom types and of coordinates for H-atoms is based on standard bond lengths at the molecular environment (Appendix, Section A1.1). The number of assigned hydrogens is checked against the structural formula stored in the *cif* file (Appendix, Section A1.3, Figures A1.1-A1.2). By answering the dialog mode (see below), the user can force retrieval anyway for further check and adjustment, or to preserve hydrogen atom positions as they are in the *.cif* file (e.g. for neutron-diffraction structures).

The retrieval and H-atom assignment procedures have been thoroughly tested for ordinary organic compounds, but chemical bonding is so multiform that the procedure may fail in some particular instances, or for low-accuracy structures where even the position of non-H atoms is questionable. The direct preparation or alteration of *.oih* files by user-generated scripts or simple file editing is also possible and relatively easy after some practice.

Running command :

run.retcif NAME (answer dialog mode)
where NAME.*cif* is the input file(s), NAME.*oih* is the output file(s)

run.retcif module (Unix/Linux)

```
rm retcif.inp
cp $1.cif retcif.inp
~/programs/MiCMoS/exe/retcif
rm $1ret.pri
mv retcif.pri $1ret.pri
rm $1.oih
mv retcif.out $1.oih
rm retcif.inp
rm retcif.tmp
```

The output file NAME*ret.pri* has a printed message on how the retrieval procedure has been carried out, with error messages for borderline cases.

The renormalization of H-atom positions and R-H distances for X-ray structures is not an option, but a must since it is an essential part of all potential energy schemes in the platform.

CAUTION: Some CSD *.cif* files contain coordinates for symmetry-dependent atoms to be deleted from the output *.oih* files that must contain all and only the atoms in the exact stoichiometry of the crystal.

Answer the dialog mode, which asks for:

(1) IHOT	Normalization of hydrogen atoms.
	=0 H are renormalized (normal option).
	=1 CSD H's are left unchanged (useful for neutron data).
(2) IPRINT	Controls the amount of output.
	=0 Normal output.

	=1	Extended output, normally for checking purposes.

(3) INONSE

Retcif controls whether structure could be wrong for some reasons. If INONSE=0, the program stops when an error is detected. INONSE=1 forces the program to print the output *.oih* file. *In this case, check carefully your structure*! A list of possible problems follows. For each case, the program prints a specific warning message.

- The H count does not coincide with that expected based on the molecular formula. Check the formula unit and verify for undetected/missing H atoms in the *.cif* file; look for wrong CH or NH group assignments.

- Unrecognized atom types. A dummy specie code (Table 1.1) of 99 is assigned to unknown atoms. MiCMoS can handle only atoms listed in Table 1.1.

- Unnatural number of bonds or attached hydrogens for C, N, O, S atoms. Are chemical groups in your structure correct? Are some covalent bonds unnaturally short or long? Is there any mistake in some specific sites (e.g. two hydrogens attached to the same non–terminal vinyl or aromatic carbon)? Are there any misplaced, duplicated, disordered atoms?

- Too many fragments in the asymmetric unit. The maximum allowed is 20.

- Too many symmetry operations. The maximum allowed is 250.

- Failure in reconstructing terminal H atoms, including methyl groups. This could be due to problems in defining the correct torsions and could highlight wrong symmetry operations or misplaced/disordered atoms in the original structure.

- No H atoms available for N, O or S. Explicit coordinates of H attached to N, O and S should be always explicitly given in the *.cif* file; if they are missing, you should add them manually. The CSD program Mercury can assist you in doing this.

- Impossible terminal O–O groups. Something is wrong with your structure: check carefully.

1.2 The *Retcor* interface module

Module ***Retcor*** reads a NAME.*oih* file and prepares a NAME.*oeh* file with complete ("explicit hydrogen") coordinates for all atoms. The reason for separating the procedure in two steps is that in this gives the user a chance to modify the procedure for molecular-model building. Normally ***Retcif***–***Retcor***–***Retcha*** are run in blind sequence. Technical details are given in the Appendix, Section A2 and Table A2.1.

Running command :

```
run.retcor NAME
```

run.retcor module (Unix/Linux)

```
cp $1.oih retcor.oih
~/programs/MiCMoS/exe/retcor
rm $1.oeh
mv retcor.oeh $1.oeh
rm $1cor.pri
mv retcor.pri $1cor.pri
rm $1.dat
mv retcor.dat $1.dat
rm $1.xyz
mv retcor.xyz $1.xyz
rm $1ort.oeh
mv retcor.ort $1ort.oeh
rm retcor.oih
```

File NAME*cor.pri* has a printed description of the procedure. ***Retcor*** also checks for subgroups in the space group: for example if the molecule is centrosymmetric and the molecular center of symmetry coincides with a crystallographic center of symmetry, the centrosymmetric space group operations will be deleted in the output *oeh* file. **Care must be taken if atoms are on crystallographic special positions.** Separate action may be needed if the crystal structure contains many units/molecules in different crystallographic position (e.g. one molecule on a center of symmetry, one on a twofold axis). Error messages and warnings are issued both on line and on the *.pri* file.

Apart the main *.oeh* and *.pri* files, normal output of *Retcor* are also:

- A NAME.*dat* file, suitable for being interpreted by the SchaKal graphics program;
- A NAME.*xyz* file, to be graphically displayed with Mercury, MolDraw, Vesta…
- A NAMEort.*oeh* file, with unitary cell and cartesian coordinates (isolated asymmetric unit), useful to generate liquid phases and solutions (see Part B).

It is advisable to check the result using the generated *.dat* and *.xyz* files in a graphics program (Section 5.1.3). Format *.dat* can be read by SchaKal (E. Keller, https://ekkristufr.neocities.org/schakal.html), while the more portable *.xyz* one can be interpreted bessentially by all modern graphical user interface systems, including Mercury (C. F. Macrae *et al.*, J. Appl. Cryst., 2020, 53, 226-235, DOI: 10.1107/S1600576719014092), Vesta (K. Momma & F. Izumi, J. Appl. Crystallogr., 2011, 44, 1272-1276) and GaussView (GaussView, Version 6.1, Roy Dennington, Todd A. Keith, and John M. Millam, Semichem Inc., Shawnee Mission, KS, 2016). Some crystallographic experience may be needed with crystals with more than one chemical or crystallographic unit, and in unusual space groups. Anyway decades-long experience has shown that ***Retcif-Retcor*** can handle 90% of organic crystal structures without human intervention.

The ***Retcor*** module can be used as a molecule model builder. Given the Cartesian coordinates for at least 3 atoms, the module can build all others by a number of different procedures (trigonal, methylene-like, pyramidal, or distance-angle torsion). This is thoroughly illustrated in Tutorials and Appendix, Section A2.

1.3 The *Retcha* module. Calculation of atomic point charge parameters for atom-atom potentials

Module ***Retcha*** reads a NAME.*oeh* file without charges and calculates atomic charge parameters from a Mulliken population analysis on a modified Extended Hückel molecular orbital calculation for closed shell, neutral molecules (*e.g.* for zwitterions but not for ions). Original data are saved in NAME*noq.oeh* and atomic point charges are re-written on NAME.*oeh*. The EHT Hamiltonian only includes the Valence Orbital Ionization Potential so the result is just a consequence of relative electronegativities: it is a cheap and convenient way of assigning a set of neutral point charges. These charges are needed for all applications using the CLP intermolecular potential energy scheme; they are rescaled by appropriate factors derived from optimization of the force field. Using other point charges along with the rest of the CLP formulation is not advisable. The module recognizes only C, H, N, O, F, Cl, Br, I, S, P atoms.

Running command:

run.retcha NAME
Where *NAME.oeh* is the input *.oeh* file and the output *.oeh* file with charges

run.retcha module (Unix/Linux)

```
rm $1noq.oeh
cp $1.oeh $1noq.oeh
~/programs/MiCMoS/exe/retcha < $1.oeh
rm $1cha.pri
mv retcha.pri $1cha.pri
rm $1.oeh
mv retcha.out $1.oeh
```

CAUTION: for crystals, if there is more than one fragment in the asymmetric unit (ASU) the *Retcha* module renumbers atoms (each unit numbered sequentially). An error message is issued if the calculation was unsuccessful (unrecognized atoms, open shells, etc.; see file NAME*cha.pri*).

1.4 The *.oih* and *.oeh* structural files

The following is a detailed description, but the prospective user should be aware that in routine applications the ***Retcif-Retcor-Retcha*** sequence is wholly automatic and proceeds in fractions of a second.

1.4.1 Filename extension *.oih*

The file .oih contains atomic coordinates and crystal symmetry data (if any), with “implicit” atomic positions. See Section 1.4.3 for a detailed description of the *.oih* format. The positions of some atomic nuclei (not less than three) are given as *x*,*y*,*z* coordinates, while the positions of other nuclei are given as a series of indicators which allow the calculation of explicit *x*,*y*,*z* coordinates. For molecular structures derived from X-ray diffraction determinations, this concerns primarily hydrogen atoms. Given the importance of H-atom positions in crystal energy calculations, this renormalization is integral part of the parameterization, and therefore mandatory. All force fields have been calibrated using such renormalized positions.

1.4.2 Filename extension *.oeh*

Atomic coordinates and crystal symmetry data (if any), explicit *x*,*y*,*z* coordinates for all nuclei including hydrogens.

CAUTION: For lattice energy calculations, a *.oeh* file must contain coordinates for a full molecule, so for crystals in which the asymmetric unit (ASU) is a fraction of the molecule the entire molecule must be reconstructed (this is the case in CSD *.cif* files) and the appropriate space subgroup must be used (this is provided by *Retcor*).

On the other hand, the reference molecular unit can consist of more than one chemical unit, as for example in crystals with more than one molecule in the asymmetric unit (ASU), or a dimer made of a host and guest compound, or a salt with two or more ions (with ionic species, user-supplied point charges must be used). When the ASU consists of one independent molecule and half a molecule on a center of symmetry, one must repeat the entire molecule and complete the half molecule (the final file must contain three molecules).

Each atom type is identified/ in a *.oeh* file by a code number (Table 1.1, next page). The force field formulations may distinguish several types of atoms of the same chemical species in different bonding environments.

Table 1.1.
Atomic species code numbers. In Molecular Dynamics, for species 1-9, if the atom code is <0 in the topology file the atom is assigned the weight of deuterium, 2.0141. Corresponding atomic properties are listed in Block Data `Alldat.for` (double precision) or `Alldas.for` (single precision). van der Waals radii in Å. See Table A1.3 (Section A1.1) in the Appendix for a list of covalent radii.

	code n.		code n.
hydrogen van der Waals radius 1.10		**oxygen** van der Waals radius 1.58	
acetylene CH	1	-O-	23
=(C)H_2, aromatic (C)H	2	H_2O (water)	24
aliphatic CH, CH_2, CH_3	3	C=O, COO^-	27
R-OH, R-SH alcohol, thiol	5	(C=O)-OH	28
COO-H acid	6	R-OH	29
CO(N)-H amide	7	N=O	30
R_2NH, RNH_2, $(R_3N^+)H$	8	S=O	31
H_2O (water)	9	P=O	32
carbon van der Waals radius 1.77		**sulfur** van der Waals radius 1.81	
carbonyl C=(O)	10	-S-	34
≡C-	11	(C)=S	35
sp^2 or allene C	12	(O)=S	36
sp^3 C	13	R-S(H)	37
aromatic core C	14		
nitrogen van der Waals radius 1.64		**heteroatoms** van der Waals radius	
$(R_nH_{4-n})N^+$	16	P 1.9	38
$(R_nH_{3-n})N$	17	F 1.46	41
Aromatic N, R=N(H)	18	Cl 1.76	42
-C≡N,-N=N	19	Br 1.87	43
nitro N	20	I 2.03	44
amide N (CONH,$CONH_2$)	21		

1.4.3 Detailed *oih* and *oeh* formats

Section 1, common to *oih* and *oeh*:

line 1) formatted field: 1x,10a4,f8.3,f8.1,3x,f5.0,1x,f6.1

- a title line (40 characters);
- crystal exptl. density (if known/needed);
- temperature (in K);
- year of the X–ray determination (if known/needed);
- R-factor (if known/needed). The last four items are supplied by the Retcif module.

From now on, all data in free format.

CAUTION: Recall that in free format a blank is not equivalent to a zero.

line 2) IOPT Dummy entry. Set it to 0. It is here maintained for consistency with previous versions.

line 3) Cell parameters *a, b, c,* α, β, γ (if the file refers to a crystal) or metric parameters (if the file refers to an isolated molecule); in this last case, they may be 1.0 1.0 1.0; 90. 90. 90. and coordinates may be given in angstrom units.

line 4) sublimation enthalpy of the crystal (if known)

line 5) NATOM, number of atoms with explicit *x, y, z* coordinates

lines 6) NATOM lines, each with: NUME X, Y, Z MLC ISPEN QRG, where

NUME:	atom sequence number: in *oih* files, atoms need not be input in sequential order; in *.oeh* files NUME is a dummy and atoms are numbered sequentially on input;
X, Y, Z:	fractional oblique (crystal) or angstrom orthogonal (single molecule) coordinates;
MLC:	number of fragment the atom belongs to (e.g. =1 or =2 for a crystal with two molecules in the ASU or for solute and solvent)
ISPEN:	atomic species code number (see Table 1.1);
QRG:	atomic point-charge parameter.

line 7) NHYD number of replacement code lines (a *.oeh* file must have NHYD=0 by definition)

Section 2: only for oih files

Atom placement codes for use in *Retcor*

In operation from a standard *.cif* file, all this is automatically provided for hydrogen atoms by the ***Retcif*** module.

lines 8) NHYD code lines, each with:
n_1, n_2, n_3, n_4, n_5, n_6, MLC, ISPEN, QRG, R, TORS, ALPH, where

n_1, n_2, n_3, n_4, n_5, n_6:	six identification codes (Figure 1.2 and Table A2.1 in the Appendix).
MLC:	number of molecular unit, or fragment, to which atom n_1 or n_1–n_2 belong;
ISPEN:	atomic species indicator (Table 1.1);
QRG:	atomic point charge parameter (zero if calculation by *Retcha* is required);
R:	bond distance;
TORS:	torsion angle;
ALPH:	bond angle as necessary (refer to Figure 1.2).

Section 3, common to *oih* and *oeh*:

Space group block

line 9) NPE	number of pairs of symmetry lines for a crystal space group
lines 10) NPE pairs of lines:	
	equivalent position matrix (identity must be the first). equivalent position vector (zero must be the first). For fractional molecular units in the asymmetric unit, NPE is the number of equivalent positions in the subgroup; e.g. NPE = 2 for $P2_1/c$, Z=2.

For an isolated molecule NPE=1 and the matrix/vector pair are identity and zero. In normal operation from a standard cif file, all this is automatically provided by the ***Retcif*** module.

Last line(s)

line 11) IL1	Molecular reference system indicator (only for the ***Pixmt2*** or ***Pretop*** modules, see Sections 3.2.5 and 5.4). If IL1=0, coordinates will be transferred as such in subsequent ***Pixelc*** or MC/MD modules (useful to deal with liquids). If IL1=3, the reference system is changed into the internal inertial one (normal use for ***Pixelc***).
line 12) NEXTRA	Number of non-library 6-12 parameters when the Lennard–Jones–Coulomb potential is used (IPOTS=1, Section 3.1.2), if needed. Leave 0 if no extra parameters are needed. IPOTS=0 and NEXTRA $\neq$ 0 are incompatible.

If NEXTRA ≠ 0, add NEXTRA lines with *i*, *j*, A6(*i*,*j*),A12(*i*,*j*) each. *i*,*j* are atom id numbers (same order in the *.oih*/*.oeh* file), A6 and A12 are the coefficients employed in equation (2.6), Section 2.1.2.

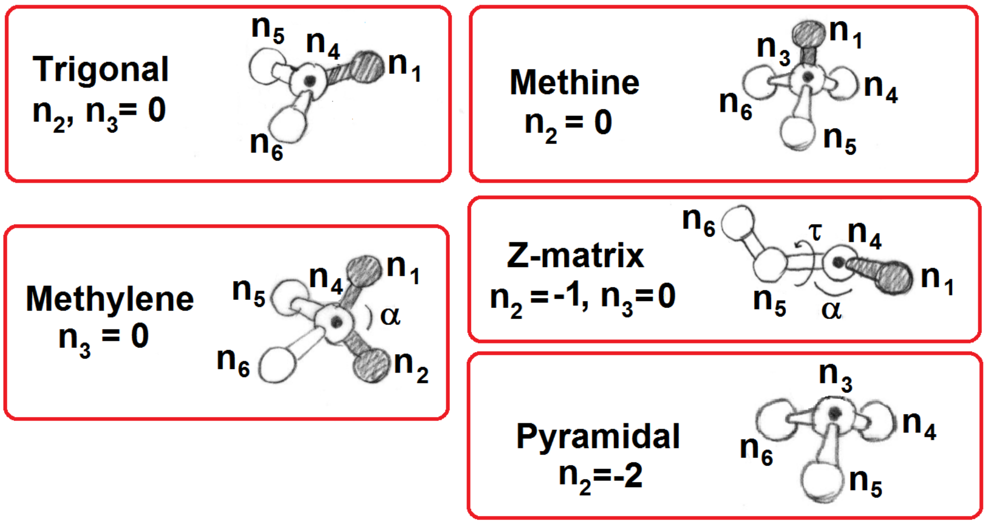


Figure 1.2. Geometry of the model building routines in ***Retcor*** driven by the codes supplied by the user or automatically by ***Retcif***. The angles α and τ corresponds to parameters ALPH and TORS to be given in lines 8ff (see above). If n_2, n_3, n_5 and n_6 are all zero, atom n_1 is re-positioned by changing the n_1–n_4 distance to the value given in the following parameter R. Shaded atoms are those that are generated from the automatic building procedure. See Appendix, Section A2 and Table A2.1 for technical description of how the various cases are handled by the program.

1.5 The *Statimpa* module

Retcif, *Retcor* and *Retcha* modules can read a *.cif* file containing multiple structures, obtaining eventually a *.oeh* file with sequentially ordered crystal data (unit cell, atom coordinates and symmetry operations). If one of the starting modules fails to correctly read or handle a structure (see Section 1.1), this is skipped but the others are printed regularly. Note that, if you are not interested in specific structures and you want to collect large databank information, missing a few *.cif* files is not a serious problem. Rather, the *Retcif–Retcor–Retcha* procedure might help you to recognize problematic or wrong cases.

Dealing with multiple-structure *.oeh* files is very useful to retrieve massive data on close intermolecular contacts, particularly hydrogen bonds (see L. Lo Presti, *CrystEngComm*, 2018, **20**, 5976-5989, https://doi.org/10.1039/C8CE00674A). This can be done using the *statimpa.for* routine, which applies periodic boundary conditions to each static crystal structure in the *.oeh* input file and detects relevant intermolecular atom-atom contacts, based on purely geometrical criteria. A "contact" is defined for every atom pair *i*,*j* belonging to different molecules, whenever it is satisfied either

(i) $$R_{ij} < P \cdot (R_i + R_j)$$

or

(ii) $$R_{ij} - (R_i + R_j) < P$$

Here R_{ij} is the distance between atoms i and j, R_i and R_j are the corresponding standard atomic radii (SAR, see below), and P is a user defined tolerance parameter. The user is also free to decide which criterion ((i) or (ii)) will be employed to define "contacts".

Three libraries of SAR can be selected by the user, namely those by Rowland & Taylor (*J. Phys. Chem*., 1996, **100**, 7384–7391),[1] Alvarez (*Dalton Trans*., 2013, **42**, 8617–8636) or Bondi (*J. Phys. Chem*., 1964, **68**, 441–451). User-defined atomic radii can be also used seamlessly. However, only atoms recognized by MiCMoS can be analyzed (see Table 1.1).

statimpa.for requires that two service files, *radii.par* and *impa.inp*, be present in the working directory. These are available in a subdir of the Source A directory of the source code (~/programs/MiCMoS/SourceA/util) and are automatically copied in the working directory by the *run.statimpa* command. The user can edit them both, before the program is launched.

The file *radii.par* looks like

```
      Rowta    Alvar    Bondi    Usdef
H     1.10     1.20     1.20     0.00
C     1.77     1.77     1.70     0.00
N     1.64     1.66     1.55     0.00
O     1.58     1.50     1.52     0.00
P     1.84     1.90     1.80     0.00
F     1.46     1.46     1.47     0.00
S     1.81     1.89     1.80     0.00
CL    1.76     1.82     1.75     0.00
Br    1.87     1.86     1.85     0.00
I     2.03     2.04     1.98     0.00
```

[1] The SAR of phosphorus was taken from M. Mantina, A. C. Chamberlin, R. Valero, C. J. Cramer & D. G. Truhlar, *J. Phys. Chem*. A 2009, **113**, 5806–5812.

Each column contains the SAR (in Å) for the various chemical species according to Rowland & Taylor (`Rowta`), Alvarez (`Alvar`), Bondi (`Bondi`). If you want to employ radii from other sources, incuding your own ones, you must add them in the "`Usdef`" column. It is wise to edit the master *radii.par* file stored in the `SourceA/util` directory, as this is the file called by the macro when you execute the program.

The *impa.inp* file contains specific steering instructions to run the program:

```
Rowta
0.95 1 2 2 2 0
```

First row: a5 format, keyword to select the SAR you intend to use. Admitted choices are `Rowta`, `Alvar`, `Bondi` or `Usdef`, with the same meaning as above. The program will read the file *radii.par* and will store the corresponding set of atomic radii.

Second row: free format, numerical parameters, one floating and 5 integers (*atol*, *iverb*, *ipack1*, *ipack2*, *ipack3*, *igave*).

atol: the *P* tolerance parameter in equations (i) and (ii) above.

iverb: verbosity flag to control the amount of output. Zero corresponds to the minimum printout. If 1, two more output files are written (see below), with explicit interaction tables for atom-atom contacts and full list of atom types (see Table 1.1) for statistical purposes. If greater than 1, the full printout is displayed, including atom coordinates, metric tensors and orthogonalization matrices. *iverb*=1 is the normal choice.

ipack1, *ipack2*, *ipack3*: three packing factors defining integer translations along the unit cell edges (*a*, *b* and *c*). Atom-atom contacts will be searched around the asymmetric unit in the supercell defined by *ipack1 x a*, *ipack2 x b*, *ipack3 x c*. In most cases, 2 2 2 works fine.

igave: Controls whether the equation (i) or (ii) will be employed to define a "contact". If =0, the condition is $d_{ij} < P \cdot (R_i + R_j)$; if =1, the condition is $d_{ij} - (R_i + R_j) < P$. Note that the *atol* parameter *P* has different meanings depending on the value of *igave*. A warning is printed in the ASCII output specifying the scheme that is selected.

The running command is:

run.statimpa NAME
Where *NAME* is the input NAME.*oeh* file

The program will copy the necessary files *radii.par* and *impa.inp* from the `SourceA/util` repository and will automatically try to open them with your associated ASCII text editor. Make sure that *.par* and *.inp* extensions are associated to an appropriate text editor, according with your working environment (e.g. Wordpad in Windows). In Unix/Linux, the editor vi is used instead.

run.statimpa module (Unix/Linux)

```
#
rm $1HB.pri
rm $1impa.pri
rm $1tabe.pri
rm $1Bs.pri
rm $1stat.pri
#
cp $1.oeh statimpa.oeh
cp ~/programs/MiCMoS/SourceA/util/radii.par radii.par
vi radii.par
cp ~/programs/MiCMoS/SourceA/util/impa.inp impa.inp
vi impa.inp
~/programs/MiCMoS/exe/statimpa > $1impa.pri
rm statimpa.oeh
mv HB.out $1HB.pri
mv tabe.out $1tabe.pri
mv Bs.out $1Bs.pri
mv stat.out $1stat.pri
```

The following output files are produced:

1. `NAMEimpa.pri`. This is the main output file. After a brief summary of the chemical structure retrieved from the NAME.*oeh* databank (CSD refcode, unit cell, input parameters, chemical formula, symmetry operations), atom-atom contacts are printed. For example, a typical output looks like:

```
At. Mol  At. Mol      d      Op   Translations      Bs%
N     1  H     1    2.5680    1 -1.0  1.0  0.0     1.346
N     1  H     1    1.7914    1 -1.0  1.0  0.0    31.180
O     1  N     1    2.7466    1  1.0 -1.0  0.0    10.212
```

   Where “At” is the atomic specie, “Mol” the molecule ID number, “d” the distance in Å, “Op” the symmetry operation ID number, “Translations” are the corresponding translations and “Bs%” is the bond shrinking parameter defined as

$$B_s = 100 \cdot \frac{R_{ij}^0 - R_{ij}}{R_{ij}^0}$$

   B_s summarizes the total percent reduction of a specific contact distance, R_{ij}, with respect to the corresponding sum of SAR’s, R_{ij}^0 (see *CrystEngComm*, 2018, **20**, 5976-5989 and references therein). Then, hydrogen bonds are analyzed in detail:

```
Number of H bonded contacts found    4

D - - - H (mol) . . . A (mol)      Donor group       Acceptor group      d(D-H)    d(H...A)  d(D...A)   alpha(D-H-A)
O   4 H  34 (1)     N   7 (1)   -COOH acid         N aromatic or =N(H)   1.0000     2.5680    3.3029    130.19
O   4 H  34 (1)     N   8 (1)   -COOH acid         N aromatic or =N(H)   1.0000     1.7914    2.7466    158.55
C  15 H  27 (1)     N   7 (1)  H aromatic or =CH2  N aromatic or =N(H)   1.0800     2.5498    3.4596    141.35
C  17 H  28 (1)     O   3 (1)  Aliphatic CH,CH2,CH3   C=O carbonyl       1.0800     2.3052    3.3517    162.66
```

Finally, the close contact count is summarized:

```
Total number of close contacts within atol   0.95000 :         6
Total number of H...A contacts:    4
H ...N :                           3
H ...O :                           1

Total number of other contacts:    2
H ...C :                           1
N ...O :                           1

Number of atoms that form primary interactions satisfying the Delta limit:   12
H (HB):                            4
H (NOT HB):                        1
C :                                1
N :                                4
O :                                2

Number of naked (=without acceptors) hydrogens in this structure             8
```

After the whole databank is scanned, a summary of total number of donor-acceptor contacts, as well as the corresponding average contact distances Rav (with standard deviations), is printed. “Nc” stands for number of contacts.

```
=======================================================
Total number of Donor...Acceptor contacts included in the databank
=======================================================
Donor                    Acceptor                           Nc       Rav           sigma(Rav)
H aromatic or =CH2      N aromatic or =N(H)                 5        2.54149        0.02518
H aromatic or =CH2      C=O carbonyl                       32        2.39803        0.01745
```

Averages over the acceptors are also given:

```
========================================================
Averages over the whole set of acceptors
========================================================
Acceptor                            Nc       Rav            sigma(Rav)
C=O carbonyl                        82        2.11540        0.03828
(C=O)-OH Acidic hydroxy              8        2.41939        0.01771
```

2. `NAMEHB.pri`. All the hydrogen bonds retrieved from the whole databank are summarized in this file in the usual DH···A atom sequence. For each interaction, geometrical parameters, CSD refcode, symmetry operation for the acceptor A, chemical nature of the donor and acceptor groups are given.
3. `NAMEBs.pri`. For each intermolecular close contact *i-j* (not only H bonds), the following information are given: identity of atoms involved, their distance, order number of the symmetry operation that produces the atom *j*, the corresponding translations, the bond shrinking parameter B_S and the CSD refcode.
4. `NAMEtabe.pri`. This file is printed only if *iverb*>0 in the *impa.inp* input file. A contact matrix for each structure is printed, which summarizes how many specific atom-atom contacts are set up across the databank.
5. `NAMEstat.pri`. This file is printed only if *iverb*>0 in the *impa.inp* input file. For statistical purposes, for each structure the number of atom types is printed. See Table 1.1 for the meaning of the various atom ID’s.

2. Potential energy schemes

2.1 Atom-atom potential forms: CLP and LJC

CLP and LJC are theoretical approaches to the evaluation of intermolecular potential energies, in the assumption that interaction centers are restricted to atomic nuclear positions and that all energy terms depend only on distances between them. Energies can be subdivided into a Coulomb-polarization term, a dispersion term (London) and a repulsion term (depending on electron density overlap, Pauli exclusion), hence the CLP acronym. The Lennard-Jones-Coulomb LJC approach consists of a unified polarization-dispersion term plus a repulsion term along with the usual Coulomb electrostatic term. These potential forms are totally empirical. The evaluation of the lattice energy of a large crystal takes less than one second, requiring only cell dimensions and atomic nuclear coordinates. Atom-atom potentials are a useful option for preliminary screening of large databases or for quick calculations of the order of magnitude of lattice energies of a static crystal; they are the only option for Monte Carlo and Molecular Dynamics simulation.

2.1.1 CLP potentials

The form of the CLP atom-atom *i-j* energy is

$$E(i,j) = \left\{\frac{1}{4\pi\varepsilon^0}\frac{F_Q q(i)\cdot F_Q q(j)}{R(i,j)}\right\} - \frac{F_P P(i,j)}{R(i,j)^4} - \frac{F_D D(i,j)}{R(i,j)^6} + \frac{F_R T(i,j)}{R(i,j)^{12}} =$$

$$= E(Coul) - \frac{A4}{R(i,j)^4} - \frac{A6}{R(i,j)^6} + \frac{A12}{R(i,j)^{12}} \tag{2.1}$$

$$\alpha(eff) = \sqrt{\left\{\frac{\alpha_i(Z_{v,i}-q_i)}{Z_{v,i}}\cdot\frac{\alpha_j(Z_{v,j}-q_j)}{Z_{v,j}}\right\}} \tag{2.2}$$

$$P(i,j) = \alpha(eff)\cdot|q_i\cdot q_j| \tag{2.3}$$

$$D(i,j) = \alpha(eff)\cdot n_i n_j\sqrt{I_i I_j} \tag{2.4}$$

$$T(i,j) = (1 + H_{Bd}H_{Ba})(Z_{v,i}-q_i)(Z_{v,j}-q_j)\sqrt{(B_i B_j)} \tag{2.5}$$

With $H_{Bd}\,H_{Ba} = 0$ if $H_{Bd}\,H_{Ba} > 0$

with *q* atomic point charges, *R* atom-atom distances, α atomic polarizabilities, *Z*v number of valence electrons, *n* quantum number of valence orbitals, *I* atomic ionization potential, *B* empirical diffuseness parameters, and *H* empirical hydrogen-bonding propensity (see Table 2.1). F_Q ,F_P ,F_D ,F_R are general scaling parameters (see Section 3.1.2). The *B* parameters are assigned using carbon = 1 and decreasing for more electronegative atoms. The H_B parameters are numbers between 0 and 1, negative for acceptors and positive for donors, so repulsion is greatly reduced over hydrogen-bonding contacts. All data are stored in the program and potentials are automatically supplied on the basis of the atomic species codes. The potential 2.1 is then calculated for each pair of atoms in the molecule(s) and stored for future use. Since 2.2 to 2.5 depend on current atomic charges, the CLP potentials are not assigned for given atomic types, but are adjusted on the basis of local environment for each molecule. Energies are meaningless if the *.oeh* file does not have charges, and warning messages are issued. Charges obtained by methods other than the *Retcha* module may give unpredictable results. Standard CLP operation includes a charge rescaling factor F_Q (that can be set = 1 when user-optimized charges are applied).

Table 2.1
Properties of atomic species considered in the CLP intermolecular energy scheme. Data are included in Block Data Alldat.for (double precision) or Alldas.for (single precision).

	indicator	atomic polarizability, Å^3	ionization potential, a.u. I° eq. 4.8	space diffusion parameter	H-bond propensity factor
hydrogen radius 1.10		0.39	0.500		
acetylene CH	1			0.60	0.20
$=CH_2$, arom.CH	2			0.62	0.10
aliphatic CH, CH_2, CH_3	3			0.64	0.05
R-OH, R-SH alcohol, thiol	5			0.75	0.99
COO-H acid	6			0.80	0.99
CON)-H amide	7			0.80	0.90
R_2NH, RNH_2, $(R_3N^+)H$	8			0.80	0.99
H_2O (water)	9			0.80	0.99
unnormalized hydrogen atom from Cambridge files	99				
carbon 1.77			0.414	1.00	0.00
carbonyl C=(O)	10	1.05			
≡C-	11	1.35			
sp^2 or allene C	12	1.35			
sp^3 C	13	1.05			
aromatic core C	14	1.90			
nitrogen 1.64		0.95	0.534		
$(R_nH_{4-n})N^+$	16			0.63	0.00
$(R_nH_{3-n})N$	17			0.63	-0.97
arom.N, R=N(H)	18			0.58	-0.99
-C≡N,-N=N	19			0.70	-0.70
nitro N	20			0.63	0.00
amide N (CONH,$CONH_2$)	21			0.63	-0.85
oxygen 1.58		0.75	0.500		
-O-	23			0.45	-0.90
H_2O (water)	24			0.70	-0.99
C=O, COO^-	27			0.50	-0.99
(C=O)-OH	28			0.50	-0.90
R-OH	29			0.45	-0.99
N=O	30			0.50	-0.95
S=O	31			0.75	-0.90
P=O	32			0.75	-0.90
sulfur 1.81		3.00	0.381		
-S-	34			2.00	-0.5
(C)=S	35			2.00	-0.5
(O)=S	36			2.50	0.0
R-S(H)	37			2.00	-0.5
heteroatoms					
P 1.9	38	*1.54*	0.386	3.0	0
AS 1.8	39	*3.5*	0.400	5.0	0
Se 1.8	40	*3.5*	0.400	6.0	0
F 1.46	41	0.55	0.640	0.20	0.00
Cl 1.76	42	2.50	0.477	2.40	-0.20
Br 1.87	43	3.27	0.434	1.50	0.00
I 2.03	44	5.00	0.384	5.00	0.00

Table 2.1b
Further atomic species.

	indicator	atomic polarizability, Å^3	ionization potential, a.u.	space diffusion parameter	H-bond propensity factor
transition metals*					
Ti	51	4.18	0.25	0.80	-0.5
V	52	3.31	0.25		
Cr	53	2.86	0.25		
Mn	54	2.93	0.27		
Fe	55	2.81	0.29		
Co	56	2.62	0.29		
Ni	57	2.61	0.28		
Cu	58	2.81	0.285		
Zn	59	3.63	0.345		
positive ions**					
Li+	61	0.10	1.00	0.2	0.0
Na+	62	0.20	0.85	0.3	
K+	63	0.30	0.70	1.5	
Rb+	64	0.40	0.50	3.0	
Cs+	65	0.30	0.45	5.0	
Ca+	66	0.70	0.70	1.5	
negative ions					
F-	67	0.40	0.75	0.5	
Cl-	68	2.50	0.65	3.0	
Br-	69	3.27	0.50	4.0	
I-	70	5.00	0.40	5.0	

*Optimized: A.G.P. Maloney, P. A. Wood and S. Parsons, *CrystEngComm* **2015**, 17, 9300–9310
** Tentative values: J. D. Dunitz, A. Gavezzotti, S. Rizzato, *Cryst. Growth Des.* **2014**, *14*, 357–366.

2.1.2 LJC potentials

The form of the LJC potential is:

$$E(i,j) = \frac{1}{4\pi\varepsilon_0}\frac{q(i)\cdot q(j)}{R(i,j)} - \frac{A6(i,j)}{R(i,j)^6} + \frac{A12(i,j)}{R(i,j)^{12}} = \\ = E(Coul) - \frac{A6(i,j)}{R(i,j)^6} + \frac{A12(i,j)}{R(i,j)^{12}} \tag{2.6}$$

A library of A6 and A12 parameters for the most common atomic species are supplied (Table 2.2). They have been optimized using high-level point charges q_i from an MP2/6-31G** wavefunction (see J. D. Dunitz, A. Gavezzotti, S. Rizzato, *Cryst. Growth Des.* **2014**, *14*, 357–366, https://doi.org/10.1021/cg401646t). Use of cheaper charges may lead to unpredictable results. User defined parameters and charges can be accepted, allowing the use of literature potential energy schemes. The LJC potentials were also recently implemented in the Molecular Dynamics module (see A. Gavezzotti, L. Lo Presti, S. Rizzato, CrystEngComm **2020**, 22, 7350–7360 https://doi.org/10.1039/D0CE00334D).

Table 2.2
A6 and A12 library parameters for the LJC potential scheme. Cross interactions should be derived by the geometrical mean rule. Data are included in Block Data `alldat.for` (double precision) or `alldas.for` (single precision) but can be updated by the user. The coefficients are consistent with distances in Å and energies in kJ/mol.

Atomic specie	A6	A12
hydrogen non h-bonding	73.8	14500.0
hydrogen H-bonding	0	0
carbon any	2280.0	4.5600d+06
nitrogen any	2200.0	2.32d+06
oxygen any	1650.0	1.22d+06
water oxygen	2470.0	2.27d+06
sulfur any	10000.0	1.3d+07
Fluorine	1080.0	7.6d+05
Chlorine	6400.0	7.65d+06
Bromine	11900.0	1.58d+07
iodine (tentative)	=A6(Br)·1.2	=A12(Br)·1.2

2.2 The PIXEL form

Intermolecular energies for crystals are calculated as numerical integrals over a large number (20,000 for a typical medium-size organic molecule) of electron-density units ("pixels", hence the name, although it has been correctly pointed out that they should be called "voxels"). The method requires one *ab initio* molecular orbital calculation (for which programs are not supplied) to prepare the molecular electron density in the form of discrete points on a grid. The calculation of the lattice energy for the crystal of a medium size organic molecule (25 atoms) then takes some15 minutes. Use of this scheme in Monte Carlo or Molecular Dynamics simulation, where energies must be evaluated millions of times, is obviously impossible.

3. Lattice energy calculation modules

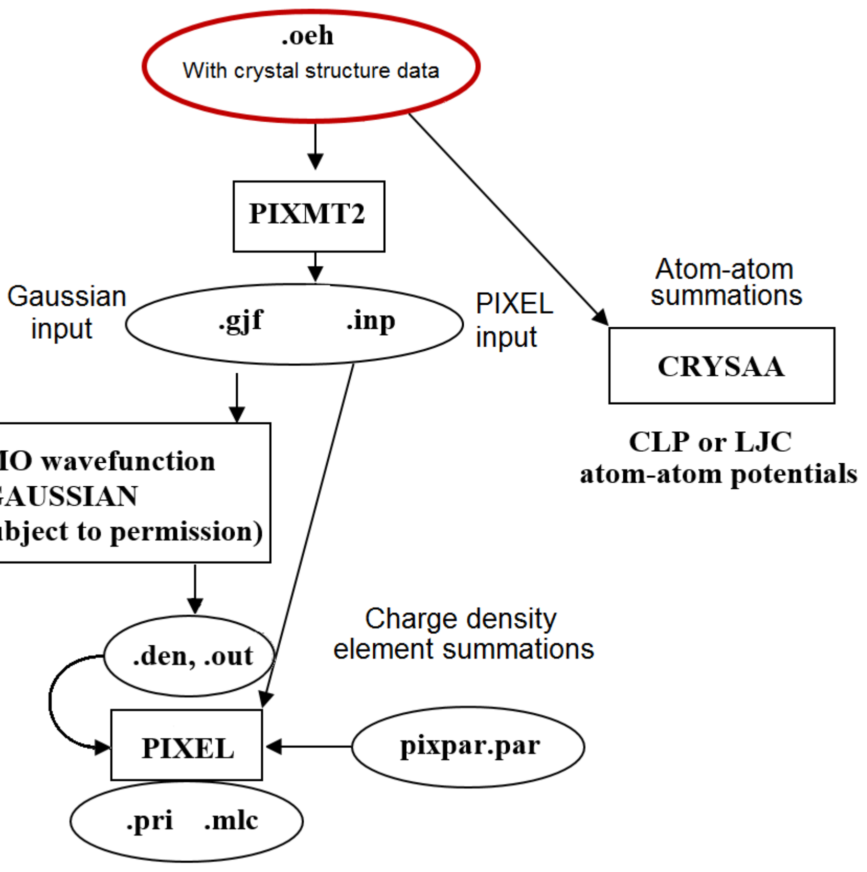

.oeh
With crystal structure data
PIXMT2
Gaussian input
.gjf
.inp
PIXEL input
Atom-atom summations
CRYSAA
CLP or LJC atom-atom potentials
MO wavefunction GAUSSIAN (subject to permission)
Charge density element summations
.den, .out
PIXEL
pixpar.par
.pri .mlc

Figure 3.1 Block diagram of the modules for crystal lattice-energy calculations. All modules use a rigid molecular unit and there is no evaluation of intramolecular energies.

3.1 The *Crysaa* module

Crystal lattice energies by CLP or LJC atom-atom potentials

Running command :

run.crysaa NAME

where NAME.*oeh* is the input file with one or many sets of crystal structure data; output is in NAME*cry.pri*.

run.crysaa module (Unix/Linux)

```
~/programs/MiCMoS/exe/crysaa <$1.oeh
rm $1cry.pri
mv cryout.pri $1cry.pri
```

3.1.1 General description of crysaa

Module *Crysaa* reads a NAME.*oeh* file with crystal coordinates of one reference molecular group (RMG). A molecular group can consist of a single molecule or of several molecules. A model of the crystal is constructed by forming a cluster of molecules using the symmetry operations of the space group. The program reads a parameter, Vec_{max}, and calculates $N_a = Vec_{max}/a + 2$; all integer t_i cell translation from $-N_a$ to $+N_a$ are considered, thus forming a parallelepiped of repeated cells of dimensions $2N_a+1$, where a is any of the three cell edges. A translation vector $T_{abc} = t_a \cdot a + t_b \cdot b + t_c \cdot c$ identifies a translated unit cell. The program loops over translation vectors T_{abc} and over equivalent positions within the cell, generating a number of surrounding molecular groups (SMG). Whole SMGs are always included in the lattice energy summations, if the distance between SMG and RMG centers of mass is below V_{max}; using cutoffs on atom-atom distances cuts off parts of molecules and leads to charge imbalance and lack of convergence. For atom-atom calculations, a typical value of V_{max} is 40-100 Å, but it is very easy to perform calculations with V_{max} = 500 or even 1000 Å. Inclusion of 50,000,000 molecules in the cluster is quite affordable, although mostly useless because convergence is reached at much shorter ranges.

The program calculates lattice energies and separate pairwise energies between molecules in the cluster, *E*(mol-mol). These can be useful in deciding which are the most cohesive interactions in the crystal. *Crysaa* performs a structure check for unreasonable intermolecular distances, destabilizing energies, wrong crystal densities ($< 0.7\ g \cdot cm^{-3}$), wrong charge balance, or other patent errors. Error messages are printed in the output file **NAMEcry.*pri***. Note that some 10-20% of entries in the Cambridge Structural Database contain errors in atomic coordinates, space group, etc., or unnoticed disorder, or other kinds of inconsistencies that prevent the calculation of lattice energies.

3.1.2 Running parameters, file crypar.par

Some parameters are in a separate file, *crypar.par*, supplied by the user once for all the crystal structure set when dealing with many crystal structures at a time. This file must reside in the same directory as the *oeh* file; if the file is not present, defaults will be assumed. The file contains:

line 1) IPRI IPOTS

• IPRI	0 or 1 controls the level of output
• IPOTS	=0 CLP potentials, =1 LJC potentials (default =0)

line 2) FQ, FP, FD, FR actual values needed only if IPOTS=0; otherwise set zero

These are the four coefficient in equation (2.1) for scaling point charges, polarization, dispersion and repulsion terms. Universal values are suggested, but fine tuning over classes of compounds is possible. Defaults: 0.41, 235, 650, 77000.

line 3)

• VECMAX	translation search parameter, default 40 Å
• EWRONG	a warning message is issued if any E(mol-mol) > EWRONG, def. +2 kJ/mol
• ELIMIT	E(mol-mol) is printed if abs(E) > ELIMIT, def. 3 kJ/mol
• CONLIM	atom-atom distance is printed if less than *conlim* times sum of atomic radii (Table 1.1); default = 0.9
• RPLIMI	E(mol-mol) is printed only if distance between centers of mass is < RPLIMI

3.1.3 Lattice energies

When there is only one unit in the RMG, the total intermolecular non-bonded potential energy of the molecule in the crystal cluster is calculated as:

$$E(pot, tot) = \sum_i \sum_j E(i, j) \tag{3.1}$$

$$E(latt) = -\Delta H(subl) = {}^1\!/_2\, E(pot, tot) \tag{3.2}$$

where *i* labels any atom in the RMG and *j* labels any atom in any SMG. *E(pot,tot)* is the potential energy of one mole of molecules in the crystal, while ½ *E(pot,tot)* is the gain in energy when one mole of molecules at infinity are brought into contact in the crystal (the computational equivalent of the sublimation energy). The Coulombic part of the sums does not converge properly only for structures with a large cell dipole in polar space groups (see below).

If there are *n* molecular units in the asymmetric unit (ASU), *E(pot,tot)* = *E*+*E'*+*E''*+..., where *E* is the equivalent of equation (3.1) for the packing of the whole cluster of *n* molecular units all together, each interacting with all the other symmetry–dependent clusters. *E'*, *E''*... are instead the energies between units1-2, 1-3...1-n, 2-3, 2-4,...2-*n*, ... *n*-*n*. For example, for three molecules in the ASU:

$$E(pot, tot) = E + E(1,2) + E(1,3) + E(2,3) \tag{3.3}$$

$$E(lattice) = {}^1\!/_2\, E + E(1,2) + E(1,3) + E(2,3) \tag{3.4}$$

Eq. (3.4) is the lattice energy of a mole of three units; if they are all equal, the true sublimation energy per mole is Δ*H(subl)* = –*E(lattice)*/3. If the units are different, like for example in an A-B molecular complex:

$$E(pot, tot) = E + E(A, B) \tag{3.5}$$

$$E(lattice) = -\Delta H(subl) = {}^1\!/_2\, E + E(A, B) \tag{3.6}$$

(3.6) is the computational equivalent of the heat of sublimation for one mole of complexes bound in the crystal to one mole of A and one mole of B separated in the gaseous state.

Crysaa always writes energies "*per entire molecule in asymm. unit*"; if all molecules are equal, this is the heat of sublimation per mole; if units are different, multiply by the number of units in the ASU, *n*.

3.1.4 Intermolecular analysis

Crysaa provides also the following intermolecular information:
1) For each atom of the RMG:
- all short intermolecular atom-atom distances, including hydrogen bonds;
2) For each nearest neighbour molecular pair in the crystal:
- distance from center of mass of each unit in the SMG to center of mass of each unit in the RMG and molecule-molecule interaction energy.

3.1.5 Coulomb sums in polar space groups

In polar space groups the calculation of Coulombic energies is critical because the lattice sums either are non-convergent, or, more fundamentally, because the method of summing all contacts to a central molecule becomes inadequate. Surface/termination effects become important and the Coulombic energy may in principle depend also on the shape of the crystal (*e.g,* influencing comparisons between polymorphs, crystal structure predictions, etc.) although this complication is usually neglected. Increasing the cutoff distance in the summations will not help. The problem is significant only when the molecule has a large dipole oriented along the polar direction. An easy but not exhaustive way of spotting a polar direction is to examine the translation vectors in the unit cell symmetry operations: any direction *t* for which there is no inversion of the *t*-coordinate is a polar direction (e.g. *y* in $P2_1$, *z* in $Pna2_1$, all three directions in $P1$). A centrosymmetric crystal structure is obviously non polar.

The calculated Coulombic energy is underestimated, and a correction must be applied; this can be done by Ewald-Bertaut reciprocal space methods (Williams, D.E., *Acta Cryst*. 1971, A27, 452–455), of considerable mathematical complexity. An alternative real-space method has been proposed (Kroon, J. and van Eijck, B.P., *J. Phys. Chem.* 1997, B101, 1096–1100), based on the assumption that the cutoff sphere is large enough that the surface cells can be seen as dipoles, and that the molecules at the surface of the cutoff sphere can be treated as a uniform distribution of dipoles. Integration of the dipole-dipole energies leads to the dipole correction energy per molecule (kJ/mol):

$$E(cell, dip) = -1389.355\,\frac{2\pi}{\mu^2}(3NV_{cell}) \tag{3.7}$$

where V_{cell} is the cell volume, *N* is the number of molecules in the cell, and μ is the module of the *cell* dipole moment vector, in electron·Å units. This correction energy has been checked to correspond very nearly to the Ewald sum result in test cases and should be summed to the Coulombic lattice energy calculated at the current cutoff.

Crysaa calculates the molecular dipole vector as the vector joining the centroid of positive charges ($\boldsymbol{d}(+)$) to the centroid of negative charges ($\boldsymbol{d}(-)$):

$$\left.\begin{aligned}\boldsymbol{d}(+) &= \frac{\sum_i \boldsymbol{x_i} q_i(+)}{\sum_i q_i(+)}\\ \boldsymbol{d}(-) &= \frac{\sum_i \boldsymbol{x_i} q_i(-)}{\sum_i q_i(-)}\end{aligned}\right\} \tag{3.8}$$

$\Sigma_i\, q_i(+)$, that should be equal to $\Sigma_i\, q_i(-)$, is the total dipole charge. The *molecular* dipole moment is then:

$$\boldsymbol{D} = [\boldsymbol{d}(+) - \boldsymbol{d}(-)] \cdot \textstyle\sum_i q_i(+) \tag{3.9}$$

and the *total cell dipole* is the vector sum over all *N* molecules in the cell:

$$\boldsymbol{\mu} = \textstyle\sum_{k=1}^{N} \boldsymbol{D_k} \tag{3.10}$$

Extensive experience on uncharged molecular species indicate that this correction seldom exceeds a few kJ/mol, but the problem becomes acute with ionic or zwitterionic species, a typical example being the crystals of natural L-aminoacids.

For a test, the Coulombic energies of glycine have been computed (in kJ/mol) by the two methods with the following results:

Reference	β–glycine	α–glycine	γ–glycine	Method
Van Ejick-Kroon[1]	–234.8	–239.5	–232.3	40 Å cutoff + equation (3.7)
Other Literature[2]	–235.0	–239.5	–231.5	Ewald summation

[1] Kroon, J. and van Eijck, B.P., *J. Phys. Chem.* 1997, B101, 1096–1100.
[2] See Voogd, J.; Derissen, J. L.; Van Duijneveldt, F. B., *J. Am. Chem. Soc*. 1981, 103, 7701–7706; Jönsson, P.–G., Kvick, Å, *Acta Crystallogr*. 1972, B28, 1827–1833; Iitaka, Y. Acta Crystallogr. 1960, B13 35–45, Kvick, Å; Canning, W. M.; Koetzle, T. F.; Williams, G. J. B. *Acta Crystallogr*. 1980, B36, 115–120.

Dipole moments are usually given in Debye units, 1 Debye = 10^{-18} esu cm, with 1 esu = 1 StatCoulomb. The conversion factor is 1 C = 2.9979 10^{9} StatCoulomb, the factor being 10 times the speed of light. Using the charge of the electron in Coulomb, the final conversion factor is 1 electron·Å angstrom = 4.80318 Debye. The cell dipole energy is summed into the Coulombic energy calculated by ordinary lattice sums.

3.2 The *Pixelc* module: Calculation of intermolecular energies by the PIXEL method

The PIXEL model allows the calculation of intermolecular energies by a distributed charge description. The model requires a preliminary evaluation of the molecular charge density by some quantum chemical method, presented in the form of a numerical grid; presently, all modules are designed to read the CUBE output of the GAUSSIAN package but adaptation to other density outputs is relatively easy.

3.2.1 Perspective

The PIXEL calculation of intermolecular interaction energies rests upon a representation of a molecular object with a large collection of electron density points ("pixels") instead of just a limited set of nuclear positions as is done in the atom-atom approach. Interactions are then to be computed as discrete sums of pixel-pixel contributions. The Coulombic integral results in interaction energies almost undistinguishable from those obtained by analytical integration. For the calculation of polarization and dispersion energies as pixel-pixel sums, the key approach is the estimation of distributed ionization potentials and polarizabilities in an empirical way. Repulsion energies use numerical overlap integrals partitioned over atomic species and a coefficient depending on the difference between electronegativities, the concept being that atom pairs where the difference is large must show some amount of intermolecular chemical "bonding".

3.2.1.1 Full list of bibliographic references

For more information and practical applications, see:

- Gavezzotti, A. *J. Phys. Chem*. 2002, B106, 4145–4154;
- Gavezzotti, A, *J. Phys. Chem*. 2003, B107, 2344–2353;
- Gavezzotti, A. *J. Chem. Theor. Comput*. 2005, 1, 834–840;
- Gavezzotti, A. *Z. Krist.* 2005, 220, 499–510;
- Gavezzotti, A. In: Newsletter nov. 2006, International Union of Crystallography, Commission for Crystallographic Computing (Chair: A.L.Spek), pp. 45-58;http://www.iucr.org/resources/commissions/crystallographic-computing/newsletters/7
- Maschio, L. *et al.* J. *Phys. Chem*. A2011, 115, 11179-11186;
- Dunitz, J. D. & Gavezzotti, A. *Cryst. Growth Des*. 2012, 12, 5873-5877;
- Dunitz J.D. *et al. Cryst. Growth Des*. 2014, 14, 357-366;
- Colombo V. *et al. CrystEngComm* 2017, 19, 2413-2423;
- Carlucci, L. & Gavezzotti, A. *Phys. Chem. Chem. Phys.* 2017, 19, 18383-18388;
- Gavezzotti, A. *et al. Cryst. Growth Des.* 2018, 18, 7219-7227;
- Chickos, J. S. & Gavezzotti, A. *Cryst. Growth Des*. 2019, 19, 6566−6576;
- Gavezzotti, A. *Mol. Phys*. 2008, 106, 1473–1485;
- Gavezzotti, A. Molecular aggregation, Structure analysis and molecular simulation of crystals and liquids, Oxford University Press, Oxford 2007, Chapter 12.;
- Dunitz, J. D. and Schweizer, W. B. *Chem. Eur. J.* 2006, 12, 6804–6815;
- Gavezzotti, A. and Eckhardt, C. J. *J. Phys. Chem*. 2007, B111, 3430–3437;
- Schweizer, W.B. and Dunitz, J.D. *J. Chem. Theor. Comp.* 2006, 2, 288–291;
- Gavezzotti, A. *Acta Crystallogr*. 2010, B66, 396-406.
- Gavezzotti, A. & Dunitz, J.D. *J. Phys. Chem*. B, 2012, 116, 6740–6750.

3.2.2 PIXEL Theory

Consider a molecule (**1**) with nuclei of charge Z_j at points (j) = [x_j y_j z_j]. Let ρ_k be the electron density in an elementary volume V_k centered at point (k) = [x_k y_k z_k]. ρ_k is usually derived from MP2/6-31G** wavefunctions, but less demanding basis sets are acceptable for large molecules. Each e-pixel has charge $q_k = \rho_k V_k$. In an usual MO calculation for a medium size organic molecule, with typical steps of 0.08 Å, one has some 10^6 pixels, too many for practical use; the distribution is then contracted into n x n x n super-pixels, n being called the contraction level. Each pixel is assigned to a particular atom in the molecule, as follows. Let p be the number of atoms for which the nucleus-pixel distance is smaller than the atomic radius. If p=1, the pixel is assigned to that atom (Figure 3.1, case A, nucleus a). If $p > 1$, the pixel is assigned to the atom from which the distance is the smallest fraction of the atomic radius (case B, nucleus b). If $p = 0$, the pixel is assigned to the atom whose atomic surface is nearest (case C, nucleus c).

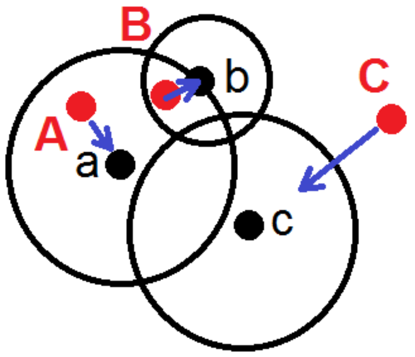


Figure 3.1. Allotment of electron density points to atomic spheres.

3.2.2.1 Calculation of the Coulombic Energy

Consider now a second molecule, **2** with nuclei of charge Z_m at points (m) = [x_m y_m z_m], and whose e-pixels of charge $q_i = \rho_i V_i$ are at positions (i) = [x_i y_i z_i]. Let R_{ln} be the distance between any two centers of pixels or nuclear positions l and n; the electrostatic potential Φ_i generated by molecule **1** at point (i) of the charge density of molecule **2** and that generated by molecule **1** at nucleus m of molecule **2**, Φ_m , with the corresponding Coulombic potential energies E_i and E_m, are respectively:

$$\Phi_i = \frac{1}{4\pi\varepsilon_0}\left[\sum_k \frac{q_k}{R_{ik}} + \sum_j \frac{Z_j}{R_{ij}}\right];\ E_i = q_i\Phi_i \tag{3.11a}$$

$$\Phi_m = \frac{1}{4\pi\varepsilon_0}\left[\sum_k \frac{q_k}{R_{km}} + \sum_j \frac{Z_j}{R_{jm}}\right];\ E_m = Z_m\Phi_m \tag{3.11b}$$

$$\mathrm{E}_{Coul,1-2} = \sum_i E_i + \sum_m E_m \tag{3.11c}$$

When e-pixels of two approaching molecules overlap, besides the un-physical aspect of the matter, numerical singularities in the R^{-1} dependence may result for very short pixel-pixel distances; all pixel-pixel distances shorter than half the stepsize of the pixel mesh are reset at half the stepsize (the 'collision avoidance' procedure). For the Coulombic energy of a crystal of polar molecules in a polar space group the van Eijck correction of eq. 3.7 is calculated using the nuclei as positive charges, and the electron charge pixels as negative charges. Interestingly, molecular dipoles calculated by eqs. (3.8) (3.9) are identical to the dipoles calculated by the GAUSSIAN program. E(cell dip) is added into the total PIXEL energy.

3.2.2.2. Calculation of the polarization energy

Let ε_i be the total electric field exerted by surrounding molecules at pixel i, α_i the polarizability at pixel i, and μ_i the dipole induced at pixel i by that field. The linear polarization energy is:

$$E_{Pol,i} = -\frac{1}{2}\mu_i\varepsilon_i = -\frac{1}{2}\alpha_i\varepsilon_i^2 \tag{3.12}$$

α_i is empirically approximated in the PIXEL scheme as $\alpha_i = (q_i/Z_{\text{atom}})\ \alpha_{\text{atom}}$, where Z_{atom} and α_{atom} are the atomic charge and polarizability of the atom to whose basin the pixel belongs (Figure 3.1 and Tables 2.1 and 2.2). The sum of α_i 's is equal to the total volume polarizability of the molecule.
As before, when e-pixels of two molecules overlap, pixel-pixel distances are subjected to the 'collision avoidance' scheme (see above); then, the polarization energy at pixel i is damped as:

$$\left.\begin{array}{c} E_{Pol,i} = -\frac{1}{2}\alpha_i[\varepsilon_i d_i]^2;\ \varepsilon \le \varepsilon_{max} \\ E_{Pol,i} = 0;\ \varepsilon > \varepsilon_{max} \\ d_i = e^{-\left[\frac{\varepsilon_i}{(\varepsilon_{max}-\varepsilon_i)}\right]} \end{array}\right\} \tag{3.13}$$

Where ε_{max} , the limiting field, is an adjustable empirical parameter in the formulation. The total polarization energy at a molecule is the sum of polarization energies at each of its electron density pixels, $E_{\text{Pol,TOT}} = \Sigma\ E_{\text{Pol,i}}$.

3.2.2.3. Calculation of dispersion energies

Dispersion energies are calculated as a sum of pixel-pixel terms in a London-type expression:

$$\left.\begin{array}{c} E_{Disp,1-2} = -\frac{3}{4}\sum_{i,1}\sum_{j,2}\frac{E_{\text{OS}}\cdot f(R)\cdot\alpha_i\cdot\alpha_j}{(4\pi\varepsilon^0)^2(R_{ij})^6} \\ f(R) = exp\left[-\left(\frac{D}{R_{ij}}-1\right)^2\right] \text{ for } R_{ij} < D \end{array}\right\} \tag{3.14}$$

where D is an adjustable empirical parameter with measure units of length. E_{OS} is the 'oscillator strength'. It is empirically approximated by considering each pixel as a separate oscillator, with a formal ionization potential I_i, which in turn is a function of the ionization potential, $I°$ of the atom to whose basin the pixel belongs, and of the distance between the pixel and the atomic nucleus, R_i:

$$\left.\begin{array}{c} E_{\text{OS}} = \sqrt{(I_i \cdot I_j)} \\ I_i = I^{\text{o}}e^{-\beta\cdot R_i} \end{array}\right\} \tag{3.15}$$

The empirical parameter β is a function of the atom type (see Table 2.1a).

3.2.2.4. Calculation of the repulsion energy

For the repulsion energy, the total charge density overlap integral between molecules **1** and **2** is subdivided into contributions from pairs of atomic species m and n, S_{mn}. The expressions are:

$$\left.\begin{array}{c} S_{1-2} = \sum_{i,1}\sum_{j,2}[\rho_i(1)\cdot\rho_j(2)]\cdot V = \sum_{i,1}\sum_{j,2}S_{mn} \\ E_{\text{Rep},mn} = (K_1 - K_2\Delta\chi_{mn})\cdot S_{mn} \end{array}\right\} \tag{3.16}$$

where $\Delta\chi_{mn}$ is the corresponding difference in Pauling electronegativity. K_1 and and K_2 are positive disposable parameters. For atoms with Z >30 (in this case Br and I) the presence of the *d*-electrons in the valence shell produces larger overlap and hence a slight (8%) decrease in K_1 is introduced. The total repulsion energy is the sum over all *m*-*n* pairs, $E_{\text{Rep,tot}} = \sum_{m,n} E_{\text{Rep},mn}$.

The total charge density overlap integral between molecules A and B is calculated over the original uncontracted charge densities. Therefore, the repulsion energy does not depend on the contraction level.

CAUTION: The integration is done numerically by counting all pairs of overlapping charge density elements, for each of which the integral is ρ(i)·ρ(j)·dV, dV being the charge density elementary volume. The procedure is sensitive to the stepsize and to the accuracy of symmetry transformations. Pairs of symmetry-related molecules in crystals may have slightly different repulsion energies (0.5-1.0 kJ/mol).

3.2.2.5. Calculation of the total interaction energy

The total intermolecular Pixel interaction energy is:

$$E_{\text{Tot}} = E_{\text{Coul}} + E_{\text{Pol}} + E_{\text{Disp}} + E_{\text{Rep}} \tag{3.17}$$

In the above formula, E_{Coul} should be corrected by the cell dipole energy of equation (3.7).

The empirical parameters in the PIXEL formulation, that is, ε_{max}, D, K_1 and K_2, were optimized considering (i) the agreement between calculated lattice energies and experimental heats of sublimation for organic crystals, (ii) interaction energies between molecular dimers in comparison with ab initio calculations, and (iii) qualitative agreement between PIXEL partitioned energies and Intermolecular Perturbation Theory (IMPT) partitioned energies.

CAUTION: The numbers are ε_{max} = 150 10^{10} V m^{-1} in eq. (3.13), D = 3.0 Å in eq. (3.14), K_1 = 4800 and K_2 = 1200 in eq. (3.16) for energies in kJ mol^{-1} with electron densities in electrons Å^{-3}. While these are suggested as universal parameters, very minor adjustments can be made to fit any desired thermochemical or structural property of the particular system under investigation, without substantial loss of physical realism.

Coulombic, dispersion and repulsion energies are pairwise additive and hence can be subdivided into contributions from two molecules in the asymmetric unit (ASU). Polarization energies are not pairwise additive as they depend on the overall crystal electric field (for more detail see A. Gavezzotti, CrystEngComm 2008, 10, 389). Polarization energies over molecular pairs do not add up exactly to total lattice polarization energies.

3.2.3 General layout

The Pixel module calculates coulombic, polarization, dispersion and repulsion energies between separate, rigid molecules.

CAUTION: No intramolecular energies are calculated. The reference coordinate frame for position of atomic nuclei and of electron density pixels is the one used in the GAUSSIAN calculation. The procedure applies to a crystal with one (A) or two (A and B) molecular species, one or two molecules per asymmetric unit (ASU), and there is no way of extending it to more fragments.

Module *Pixmt2* reads an *.oeh* file and calculates the matrix/vector operations that transform from the molecular reference frame to coordinates in the crystal structure (see 3.2.9) and prepares an input file to GAUSSIAN (extension *.gjf*) with appropriate limits for the electron density cube, and the input file to PIXEL (extension *.inp*). As in the atom-atom modules (see Section 3.1 above), molecules in the cluster that represent the crystal structure are obtained by space group symmetry operations, subject to a cutoff distance between centers of mass of reference and surrounding molecules.

3.2.4 Program inputs and outputs

The input to PIXEL consists of:

1) an electron density file for the A molecule, and one for the B molecule if any, extension *.den;* these should come from a GAUSSIAN calculation which in turn requires an input file, extension *gjf*. The *.den* file is in the usual *.cube* format and can be also produced by running the *formchk* and *cubegen* ancillary programs of the GAUSSIAN package (see Section 3.2.5 below).

2) a file with atomic parameters, extension *.inp*.

3) a file (*pixpar.par*) with the parameters of the theory.

The outputs are a printout file with the results of the calculation, extension *.pri*, and a file with molecule-molecule energies, extension *.mlc*.

3.2.5 How to run a PIXEL calculation

1) Prepare a molecular model for the A molecule (*x,y,z* coordinates) and set up a *oeh* file. This could be the output of *Retcif-Retcor* ;

2) Run module *Pixmt2*: the running command is

run.pixmt2 NAME   (for file NAME.*oeh*)

run.pixmt2 module (Unix/Linux)

```
~/programs/MiCMoS/exe/pixmt2 <$1.oeh
rm $1.gjf
mv pixmt2.gjf $1.gjf
rm $1.inp
mv pixmt2.inp $1.inp
```

Pixmt2 transforms from coordinates (*.oeh*) to local reference systems in the *.gjf* file. The final entry in the *.oeh* file (IL1, see also at the end of Section 1.4.3) can be zero, in which case the coordinates are left as they are, or = 3, in which case the coordinates are transformed to the inertial reference frame (normal use). PIXEL will then apply the appropriate transformations to prepare the cluster of molecules for the actual calculation on crystals. *Pixmt2* will also generate a *NAME.inp* file for PIXEL input, including for the crystal case, also the matrix/vector $\mathbf{M}_2$/$\mathbf{t}_2$ pair that relate the reference system of the unit cell with the one of the molecular inertial axes (see Section 3.2.8 and Appendix, Sections A3 and A4).

3) Run GAUSSIAN for the electron density calculation. *Pixmt2* prepares a *.gjf* input file that can be directly read by the program. If a different MO package is used, the input data and the output electron density file must be converted accordingly (see Appendix, Section A5 for a description of the *.den* density file format).

Typically, the main input string is written as follows:

```
#MP2/6-31G** guess=core nosym density=MP2 pop=esp cube=cards
cube=frozencore
```

The keyword *cube=cards* requires that specific instructions to generate the cube density file (Appendix, Section A5) be read just after the atom coordinates block, as follows:

```
<blank line>
Path_to_locate_density_file
N0 X0 Y0 Z0
N1 X1 Y1 Z1
N2 X2 Y2 Z2
N3 X3 Y3 Z3
<blank line>
```

where N0 is a format flag that should be set equal to 0, X0, Y0 and Y0 are the coordinates of the initial grid point in bohr, and Nn, Xn, Yn, Zn the number of points and step sizes along the three Cartesian axes. See https://gaussian.com/cubegen/ for more information. "*Path_to_locate_density_file*" must be replaced by the full path of the directory where the density file should be placed, such as, for example, c:\users\yourname\yourjob.den for Windows users or ~/yourdirectory/yourjob.den for Linux/Unix users.

CAUTION: The last blank line below this set of instructions is mandatory. Otherwise, the program will end with an error message after the SCF procedure without printing the density!

Normal use requires a valence only density (*cube=frozencore*), which in the recent releases of the program is the default calculation mode for post–Hartree–Fock Hamiltonians. It implies that inner–shell electrons do not contribute to the total correlation. More information (and options) can be found in the GAUSSIAN manual (https://gaussian.com/frozencore/). Please notice that the *NoSym* option must be chosen, so the automatic symmetry recognition and consequent, further transformations operated by GAUSSIAN are suppressed ($\mathbf{M}_1$ = identity matrix, $\mathbf{t}_1$ = $\mathbf{0}$, see Section 3.2.7 and Section A4 in the Appendix).

CAUTION: **The above-described procedure is recommended**. However, the *.gjf* input file makes use of some obsolete/deprecated keywords (have a look at https://gaussian.com/obsolete/), in particular *pop=esp* and *cube*. Luckily, the current (2016) release of GAUSSIAN is still able to correctly handle these keywords. Future program developments might make the whole procedure obsolete, though.

We here provide an alternative procedure, that employs up–to–date program commands and strings. First, *pop=esp* should be replaced by the equivalent *pop=MK* one. This implies that, if the *density=MP2* keyword is also specified, atomic charges are evaluated by fitting the MP2–derived electrostatic potential at the points of a grid defined according to Mertz & Kollman (B. H. Besler, K. M. Mertz, P. A. Kollman, J. Comput. Chem. 1990, 11, 431–439). See https://gaussian.com/population/ for more information. Second, you should explicitly require that the checkpoint file be written. Third, the "*cube=*" instructions must be omitted, as well as the corresponding cube card strings after the atom coordinates input block. In summary, the typical GAUSSIAN input command line should now look something like:

```
%Chk=yourjob.chk
#MP2/6-31G** guess=core nosym density=MP2 pop=MK
```

After the calculation is complete, you should call the ancillary program *formchk* to format the checkpoint file according to:

```
formchk yourjob.chk yourjob.fchk
```

Finally, the density file can be produced by the *cubegen* routine:

```
cubegen n density=MP2 yourjob.fchk yourjob.den -3 h
```

Where *n* is the number of processors you choose to employ, *density=MP2* calls for the Møller–Plesset density, *.fchk* and *.den* are the input and output files, –3 is a flag corresponding to a medium–spaced grid (6 points/bohr) and *h* is a flag to include the header in the density cube file. More information can be found at https://gaussian.com/cubegen/.

The two procedures for generating the density grid file give the same cohesive energies within ~2 kJ/mol (0.5 kcal/mol) per ASU. For example, the energies of the $P2_12_12_1$ polymorph of naphthalene are as follows:

Procedure	E_{Coul}	E_{Pol}	E_{Disp}	E_{Rep}	E_{Tot}
recommended	-29.6	-11.9	-94.9	53.6	-82.8
formchk+cubegen	-31.1	-12.4	-95.0	54.2	-84.4

Incidentally, this dependence on grid shape is another of the many computational uncertainty factors that may well be of the same order of magnitude as differences between crystal polymorphs.

4) Repeat steps 1-3 for the B molecule if needed.

5) Prepare the parameter file, *pixpar.par* (see Section 3.2.6).

6) Run PIXEL making sure that all necessary data are in the same directory.

Pixel running command (make sure file *pixpar.par* is present, see Section 3.2.6):

run.pixelc name1 name2 name3 name4

Where *name1–name4* have the following meaning:

-	name1.inp	line input file

-	name2pix.pri	printout with energies
-	name2.mlc	*.mlc* file with molecule-molecule energies
-	name3.den	A-molecule electron density file
-	name4.den	B-molecule electron density file (name4=blank if there is only one molecular species in the calculation)

run.pixelc module (Unix/Linux)

```
rm fort.*
cp pixpar.par pixel.pmt
cp $1.inp pixel.inp
cp $3.den solu.den
cp $4.den solv.den
~/programs/MiCMoS/exe/pixelc
rm $2pix.pri
mv pixel.oxp $2pix.pri
rm $2.mlc
mv pixel.mlc $2.mlc
rm fort.*
rm solu.den
rm solv.den
rm pixel.pmt
rm pixel.inp
```

3.2.6 Description of the pixpar.par file

CAUTION: if the default version of the theory is adopted, all these inputs are zero and the default quantities are automatically used.

All free format

1) first line:

• collis	collision parameter (write 0; defaults to one half the step in electron density)
• ddamp	damping distance for dispersion (defaults to 3.00)
• fiemax	max field in calculation of polarization (defaults to 150) 10^{10} implicit
• corep1	factor K1 in calculation of overlap repulsion energy, E = K1+K2(Del) where Del is a difference in electronegativity; default is K1=4800
• corep2	factor K2 as above; default is K2=1200

2) second line, electron density trimmers for A molecule:

• nrdu	contraction level (defaults to 4)
• ivalu	= 0 valence electron density (normal operation), =1 total electron density (e.g. in positive ions)
• romiu	minimum value of charge in a density pixel (defaults to 0.000001)
• romau	max value for a charge density pixel (defaults to 9999)

3) the same for B molecule (give zeros if nmsolv=0)
nrdv, ivalv, romiv, romav

4) threen	threshold for printing absolute molecule-molecule energies in the output
idibar	= 0 molecule-molecule distances between centers of mass = 1 distances between centers of coordinates

If this input line is missing the parameters are set to 3.0 and 0.

3.2.7 Description of the *.inp* file

All free format.
Line 1) A title line

Line 2) molecular specifiers:

• nmsolu	number of A molecules for the crystal case, NMSOLU=1 (the n. of molecules is determined by the program)
• nmsolv	number of B molecules for the crystal case, NMSOLV=1 for 2 molecules in ASU
• nasolu	number of atoms in A molecule
• nasolv	number of atoms in B molecule

Line 3a) chargu net charge for the A molecule

Line 4a) for each atom in the A molecule:

- sequence number
- atom type (Table 1.1)

- a dummy "atomic charge" parameter (for compatibility with old versions); = 0.0
- atomic polarizability; polarizabilities are taken from database if left = 0 (suggested)

Line 3b) if nmsolv > 0:	chargv net charge, as in 3) above, for molecule B
Line 4b) if nmsolv > 0:	as in 4a), for each atom in the B molecule

The Pixmt2 module usually supplies the following input lines in file *.inp* automatically:

Line 5) cutmi, cutma
Symmetry operations that produce repeated molecules whose distance from the central one is between cutmi and cutma are automatically included in crystal model. When there are two molecules in the ASU, a symmetry operation is included if at least one of the four distances (A-A', A-B', B-A' or B-B') is in the range. A cutoff of 15 Å for non-charged compounds, and of 30 Å for zwitterions, is usually enough to ensure convergence.
Setting cutmi=cutma=0 suppresses the neighbor search, and the calculation includes only molecules for the supplied symmetry operations. In this case, energies are potential energies for the central molecule in the field of the surrounding ones, and not lattice energies (i.e. they are not multiplied by 0.5 as normal in lattice sums).

Line 6) cell parameters

Line 7) M_1, t_1	matrix/vector pair (matrix by rows); see Appendix, Section A4. Normally $\mathbf{M}_1$=unit matrix and $\mathbf{t}_1$= [0 0 0] if the *Nosym* option is used in GAUSSIAN.
Line 8) M_2, t_2	matrix/vector pair (matrix by rows), from ***Pixmt2*** module (see also the Reference Materials in the Appendix, Section A4).

9) NPZ, number of symmetry operations in space group (or of included molecules if cutma=0)

10) npz matrix/vector pairs, one pair for each symmetry operation

3.2.8 The PIXEL output files

While the program is running, some output appears on screen. The *pri* file has a title, some echo of input parameters, and detail on the electron density screenout and condensation procedures. Then the PIXEL energies: Coulombic, polarization, dispersion, repulsion and total. For crystals, factors of 1/2 are appropriately applied so that these numbers are the computational equivalent of the enthalpy of sublimation. At the end, the output file has a list of molecule-molecule energies:

A···A	*i*-th molecule A to *j*-th symmetry-transformed molecule A; distance between centers of mass, coul, pol, disp, rep, total PIXEL energy;
A···B	*i*-th molecule A to *j*-th symmetry-transformed molecule B; etc. for the B···A and B···B lists.

These molecule-molecule energies along with the symmetry operation connecting the reference and symmetry-related molecule, also appear in a separate file extension *.mlc*.

CAUTION: More than total lattice energies, this is the key feature that makes PIXEL a useful and reliable tool: perusal of the relative entity of these energies, along with the structure of the corresponding dimer, give solid indications on the relative importance of various kinds of interactions in crystal packing. It is usually if not always the case that these indications are in contrast with conclusions derived from geometric analysis of short contacts, or from fancy surmise of exotic bond types. It is usually the case that packing factors thought important on a geometrical basis, like short atom-atom intermolecular distances giving rise to a fanciful bonding literature, are vigorously contradicted by consideration of the relative energies.

Scripts for the interpretation of this output have been prepared by some PIXEL users groups and may be available in the net.

Part B

Monte Carlo and Molecular Dynamics simulation

4. General flowchart of Monte Carlo and Dynamics modules

The MiCMoS platform includes modules to perform Monte Carlo (MC) or Molecular Dynamics (MD) simulations of the condensed states of organic compounds. Figure 4.1 gives a flow diagram of the organization of the MC and MD simulations. Auxiliary programs help with the preparation of force field files and of structure-data files for liquids or for solids. Other modules provide trajectory analysis and other kinds of structural analysis. Please refer to the following Sections for a full description.

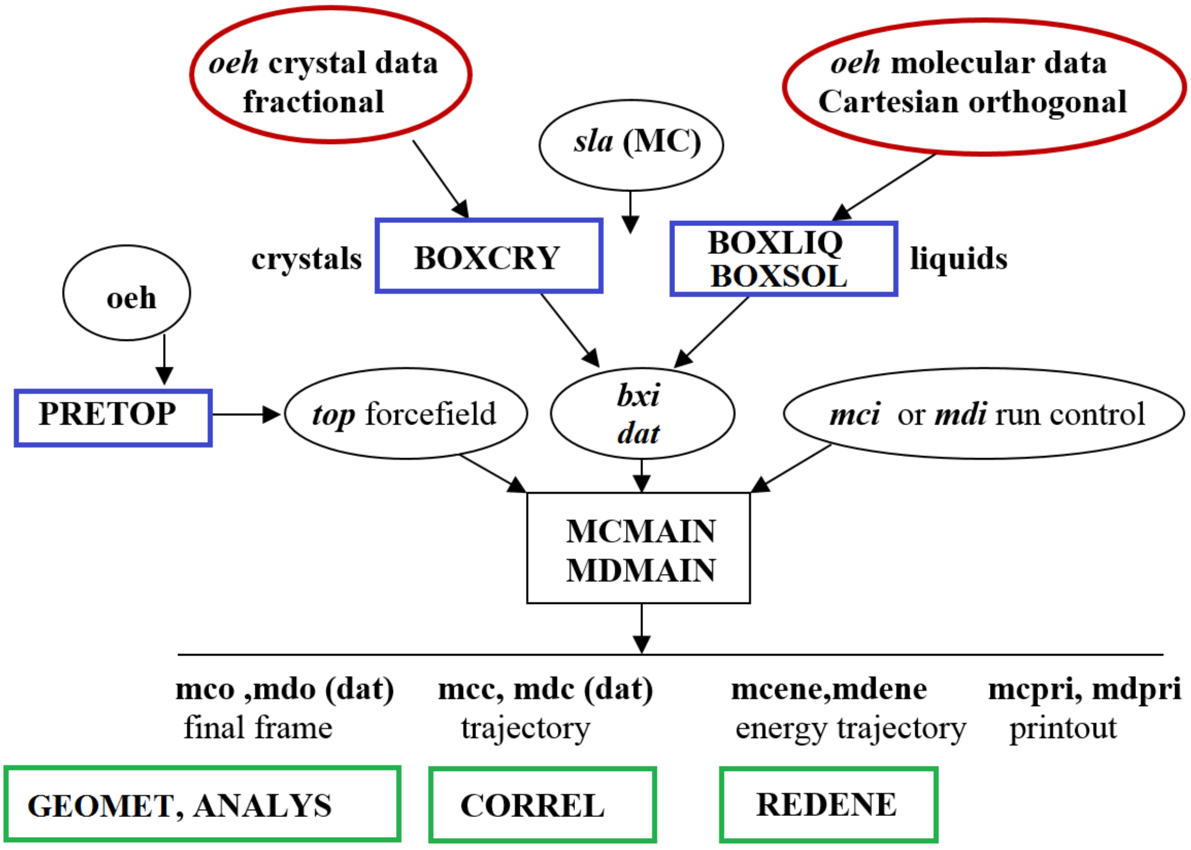


Figure 4.1. Block diagram of the Monte Carlo and Molecular Dynamics manifold. Circles are files, squares are program modules. Blue (green) boxes denote auxiliary (analysis) programs.

The ***Pretop*** module (Figure 4.1, Section 5.4) reads an *.oeh* file and prepares the best possible approximation to the pertinent force field file, except for the separation between core and slave atoms in MC (see Sectios 6.2 and 6.6.4) that must be handled by the user. The ***Boxcry*** and ***Boxliq*** routines prepare the starting simulation boxes of pure substances, whereas ***Boxsolv*** produces two-component simulation boxes of various kind. They are described in Sections 5.1–5.3.

4.1 Available intermolecular potentials

Monte Carlo e Molecular Dynamics calculations rely on the same AA potentials described in Section 2, that is, AA–CLP (All Atoms–Coulomb, London and Pauli, Section 2.1.1) or AA–LJC (All Atoms–Lennard-Jones and Coulomb, Section 2.1.2).

5. Interface between structural files and MC or MD files

5.1 The *Boxcry* module

This module prepares a computational box with molecules in a crystal structure, for space groups up to orthorhombic and for up to 2 molecules in the asymmetric unit. The formats are *.bxi* for MC and *.dat* for MD. Boxes of *.bxi* type for space groups with non-diagonal symmetries (e.g fourfold axes) cannot be dealt with in this routine. For MD, crystal boxes of any symmetry in *.dat* form can be prepared by user-defined scripts.

CAUTION: Singularities can arise for molecules in special crystallographic positions (see Appendix, Section A6). Moreover, sometimes the box thus obtained is not very compact, with protrusions and voids due to uncomfortable layout of the crystal fractional atomic coordinates (*e.g.*, coordinates given very far from the cell origin). A preliminary MC or MD run will take care of this by replacing molecules out of box boundaries.

The ***Boxcry*** running command is:

run.boxcry NAME

where NAME is the name of a NAME.*oeh* file with fractional crystal coordinates, and also the name of the NAME.*sla* file needed only if there are slave atoms (only for MC, see Sections 6.2 and 6.6.4). There must be two sets of slave atom lines if there are solute and solvent (*i.e.* two molecular species in the crystal).

run.boxcry module (Unix/Linux)

```
cp $1.oeh boxcry.oeh
cp $1.sla boxcry.sla
~/programs/MiCMoS/exe/boxcry
rm $1cry.bxi
rm $1cry.dat
rm $1box.pri
mv boxcry.dat $1cry.dat
mv boxcry.bxi $1cry.bxi
mv boxcry.pri $1box.pri
rm boxcry.oeh
rm boxcry.sla
```

Answer the dialog mode, which asks for:

NREPA, NREPB, NREPC	Number of cells along *a*, *b* and *c* to build the simulation box

The output is in part printed on screen, but the program generates also the following files:

- NAMEcry.*bxi*	This is the MC input simulation box, in a contracted format
- NAMEcry.*dat*:	*.dat* file with atomic coordinates for MD input or for graphics of the entire crystal box (format: see Section 5.1.3)
- NAMEbox.*pri*:	detailed printout with orthogonal coordinates of all molecules in cell

5.1.1 The *.bxi*/*.bxo* format (MC only)

Boxcry builds the initial simulation box by replicating the crystallographic unit cell a certain number of times along *a*, *b* and *c* vectors. The user specifies the number of replicas for each direction (see above). The program uses the information contained in the *.oeh* file produced by the ***Retcif***, ***Retcor*** and ***Retcha*** modules (see Section 1). The *.bxi* file, which will enter the subsequent MC procedure, contains all the information concerning the position and mutual orientation of the molecules in the simulation box in a rigid-body six-parameter format (Table 5.1). A MC run also produces an output *.bxo* file in the same format, which can be used to restart the simulation if needed.

Table 5.1
The MC-box file (input *.bxi* and output *.bxo*, all free format). It carries information on the molecular position and orientation and on the number of slave atoms (for MC only, Section 6.6.4). They are prepared by ***Boxliq*** and ***Boxcry*** but should not be conusfed with *.dat* files (Section 5.1.3) that carry atomic coordinates.

1) NMSOLU	number of solute molecules, number of the last simulation step

NMSOLU blocks, each with:

a) *x*, *y*, *z* of the centre of mass of each solute molecule (Å units), three Euler rotation angles (degrees, see Appendix, Section A5), ISYMM indicator (see below), three step type number *n* for variation of *x*, *y*, *z* (usually 1) and three *step* type number for variation of the three angles (usually 2). These numbers correspond to a series of stepsizes specified in the run-control, *mci* input file, Section 6.6.2: the actual step in an MC move is (*rand*-0.5)·*step*, where *rand* is a random number, and *step* is the stepsize specified in the list. These numbers are usually 1 and 2, respectively, meaning that the first two stepsizes in the list define the rigid-body overall molecular motion. Setting these number(s) to zero suppresses the variation of the corresponding degree of freedom (*e.g.*, to constrain some center-of-mass displacements and/or rotations). Similar step type numbers are also defined for internal degrees of freedom, bond stretching, bending and torsions, in MC moves (Section 6.6.4).

b) Only if some slave atoms are present in a MC run, the contents of the *.sla* file for the solute (see Sections 6.2 and 6.6.4).

2) NMSOLV	number of solvent molecules
NMSOLV blocks	Same as 1a), 1b) for the solvent

3) BOXX, BOXY, BOXZ, ALF, BET, GAM, NX, NY, NZ
BOXX, BOXY, BOXZ: Computational box dimensions edges (in Å);
ALF, BET, GAM: Computational box angles (deg);
NX, NY, NZ: Number of repetitions of the unit cell along the three axes for a crystal calculation (1,1,1 if box is for a liquid). Needed only for periodic-box runs, otherwise may be set to zero.

5.1.2 How *Boxcry* works: the ISYMM indicator (MC only)

For the simulation of crystal structures, an ISYMM indicator (Table 5.2) is needed to specify which symmetry operation must be performed on the fundamental molecule to obtain the proper orientation of each molecule in the crystal box. This information is automatically provided by the ***Boxcry*** module.

CAUTION: **Known bug**: the automatic algorithm runs into singularities for some special positions in some space groups. You may solve these issues by lowering the symmetry of your space group before entering the MiCMoS system. To this end, the free Bilbao Crystallographic web Server can assist you (https://www.cryst.ehu.es).

Table 5.2
ISYMM codes to be specified in the *.bxi*/*.bxo* files (Section 5.1.1).

Operation	ISYMM	Operation	ISYMM	Operation	ISYMM	Operation
Pure translation	0	*x, y, z*				
Mirror plane or glide plane	1	*–x, y, z*	2	*x, –y, z*	3	*x, y, –z*
Twofold axis or twofold screw	4	*–x, –y, z*	5	*–x, y, –z*	6	*x, –y, –z*
Inversion center	7	*–x, –y, –z*				

The *.bxi* crystal box has the number of molecules corresponding to the number of cell replications given in the input. Figure 5.1 shows some pictorial schemes for the operation of ***Boxcry***. The ISYMM indicator is prepared by comparing the signs of symmetry-related atoms in different molecules, when all molecules have been set in their inertial reference frame. This does not work when molecules are at some special positions with zero coordinates or when there are coordinate interchanges, *e.g.* $x = -y$. All this is unnecessary in MD that works on explicit coordinate files.

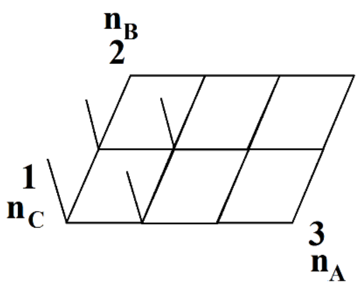


Figure 5.1. Building of a crystal simulation box

Together with Euler angles and centre of mass position, ISYMM allows to properly generate the coordinates of all the symmetry–related molecules in the box. This procedure should work well for all the space groups up to the Orthorhombic system.

5.1.3 Format of *.dat* files

These files contain explicit coordinates of all atoms in the simulation box in Å units in the Cartesian reference frame of the simulation box (the corresponding *.gro* format of the GROMACS platform uses *nm* units). The *.dat* format covers several purposes, including MC trajectories (*mcc.dat* and *mco.dat*, Section 6.6), MD input/output coordinates (*md.dat* and *.mdo*, Section 7.1) and trajectories (*.mdc*, Section 7.1). In MD also the atomic velocities and forces are printed in the last trajectory frame (*.mdo* file) as $m{\cdot}s^{-1}$ and *N*.
The *.dat* format is set for graphics by SchaKal (E. Keller) but conversion to any other graphics file format is straightforward.
Note: a maximum of 2000 molecules with at most 100 atoms per molecule is allowed.

1) “TITL” (a4), Title line (10a4), NMOVE (i10).

“TITL” is the line identifier in SchaKal and MiCMoS analysis modules (do not change!);

'Title line' is any string you want (typically, something like "trajectory at MD/MC step number…");

NMOVE; number of MC or MD move to which coordinates are referring to.

2) "#" (1x), NMSOLU, NMSOLV, NASOLU, NASOLV, IVEL (5i5)

"#" is the line identifier (do not change!). This means that the following information are skipped when read by SchaKal, but they are still used by utilities programs in the MiCMoS environment.

NMSOLU: number of solute molecules;

NMSOLV: number of solvent molecules;

NASOLU: number of atoms in each solute molecule;

NASOLV: number of atoms in each solvent molecule;

IVEL: a 0/1 =0: atomic velocities and forces are not present;
=1: atomic velocities and forces are read in MD after the "END" keyword (see below).

3) "#" (1x); *a, b, c* (3f10.4); α, β, γ (3f10.3); NREPA, NREPB, NREPC (3i5)

This line must bear non–zero quantities whenever a periodic simulation box is employed (see for example the MD case, Section 7.6.3).

"#" is the line identifier (do not change!).

a, b, c: edge lengths of the whole simulation box, in Å.

α, β, γ: angles of the whole simulation box, in degrees.

NREPA, NREPB, NREPC: number of repetitions of the crystallographic unit cell to build the whole simulation box. This means that the actual crystallographic cell edges a_c, b_c, c_c can be found according to $a_c = a$ / NREPA $b_c = b$ /NREPB and $c_c = c$ / NREPC.

3) "CELL" (a4); X, Y, Z, ALP, BET, GAM (6f6.1)

After the line identifier, this is a dummy line, which must be 1. 1. 1. 90. 90. 90. in the correct format, as coordinates are always Cartesian orthogonal (see above).

4) "ATOM" (a4, 2x); IAT (a2); X, Y, Z (3f10.3)

"ATOM" is the line identifier (do not change!).

IAT: symbol of atom species.

X, Y, Z: atomic coordinates, in Å.

Line (4) is replicated NASOLU·NMSOLU + NASOLV·NMSOLV times, one for each atom in the simulation box.

5) "END" (a3)

"END": line identifier (do not change!). It signals the end of the coordinates section.

6) Only for MD, if IVEL = 1, further NASOLU·NMSOLU + NASOLV·NMSOLV lines are attached to the coordinates section, after the "END" keyword. Information are given as follows:

NMOL, NATOM, V_x, V_y, V_z, F_x, F_y, F_z (i5, i3, 3d14.6, 1x, 3d12.4)

NMOL: molecule id number;

NATOM: atom id number for the NMOLth molecule;

V_x, V_y, V_z: Cartesian components of the atomic velocity ($m \cdot s^{-1}$);

F_x, F_y, F_z: Cartesian components of the atomic force (N).

5.2 The *Boxliq* module

This module prepares a computational box containing molecules in an approximate liquid structure. Molecules are at sites of a body-centered pseudo-cell.

Running command:

run.boxliq NAME

NAME refers to a *NAME.oeh* file with orthogonal coordinates in a local (e.g. inertial) reference (for a molecule out of a crystal structure, this can be obtained *e.g.* from the ***Retcor*** module, file NAME*ort.oeh*: see Section 1.1), and the name of the *.sla* file with slave atoms if any (only for MC simulations, Sections 6.2 and 6.6.4).

run.boxliq module (Unix/Linux)

```
cp $1.oeh boxliq.oeh
cp $1.sla boxliq.sla
~/programs/MiCMoS/exe/boxliq
rm $1liq.bxi
mv boxliq.bxi $1liq.bxi
rm boxliq.oeh
rm boxliq.sla
```

The output is file NAME*liq.bxi*: input liquid box. This file has the same format detailed in Table 5.1 above, but there is no corresponding *.dat* file, as there was in *Boxcry*. Some information is printed on screen.

Answer the dialog mode, which asks for:

- NREP		The number of repetitions of a body centered pseudo-cell; the final box contains $2(\text{NREP}+1)^3$ molecules
- FACT		Expansion factor, controls the spacing between molecules and the final box density. Usually FACT = 1.0 to 1.5 is ok. Adjust until the final printed density is acceptable, somewhat below the experimental value to facilitate the subsequent optimization;
- RANDT		Small random displacement from cell sites. Usually 0.5 Å is ok.
- TAUSPREAD		Controls the spread of starting random torsion angles for slave atoms, from the values given in the *.sla* file (see Sections 6.2 and 6.6.4). Usually 10° is ok.
- ISYMM		Controls whether the liquid is a racemate or not.
	= 0	All molecules have the same chirality,
	= 1	Generates half molecules of each chirality for a racemic liquid by setting half molecules as mirror images.

FACT, TAUSPREAD and RANDT control the expansion and randomization of the original box. Usually many hard contacts arise, but running a preliminary MC run (even with a fully rigid molecule without slave atoms *etc.*, if the aim is MD) will get rid of these contacts and prepare a reasonable starting geometry. Figure 5.2 shows some pictorial schemes for the operation of *Boxliq*. A preliminary MC run is anyway always needed to run MD in order to have a starting *.dat* file (*mco.dat*).

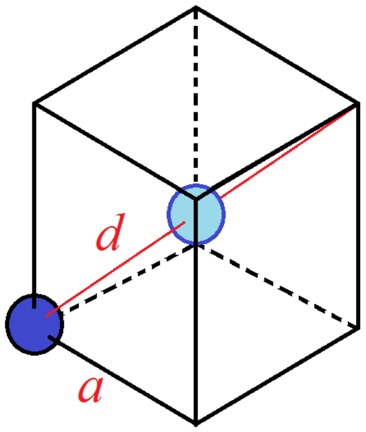


Figure 5.2. Generation of next–neighbors pair in a simulation box of the liquid phase. Two molecules are shown in a "body centered" cubic lattice, with $a = 3^{1/2}\cdot d/2$; each molecule has a random orientation and is displaced at random from perfect lattice nodes. The starting box is repeated n = NREP times in x, y, z, generating a total of $2(n+1)^3$ molecules. If a racemic liquid is desired, every second molecule is set with ISYMM = 7 in the *.bxi* file (Table 5.1).

5.3 The *Boxsol* module

This module merges two MC computational boxes (extension *.bxi* or *.bxo*) into a single box. The program reads a "solute" box, a "solvent" box, and deletes as many solvent molecules as necessary to make approximate space for the solutes. Each box may come from *Boxliq*, *Boxcry*, or manual preparation. The procedure is very approximate and relies on an estimate of the diameters of the two molecules. Quite often very hard contacts result, and extensive MC energy minimization cycles may be needed to reach a satisfactory configuration.

Running command:

run.boxsol *name1 name2 name3*
where *name1* is the solute box, *name2* is the solvent box, and *name3* is the resulting solution file.

run.boxsol module (Unix/Linux)

```
cp $1 mcboxu.sol
cp $2 mcboxv.sol
~/programs/MiCMoS/exe/boxsol
rm $3
mv mcboxs.out $3
rm mcboxu.sol
rm mcboxv.sol
```

You should answer the dialog mode, which asks for:

- n. of slave atom lines (Sections 6.2 and 6.6.4) in the *name1* and *name2* boxes;
- X, Y, Z shifts for origin in *name1* box (usually all zero);
- X, Y, Z shifts for *name2* box (usually zero);
- approximate molecular diameters for solute and solvent, and a 'tolerance factor'; solvents are deleted when the distance to a solute is less than the sum of molecular radii times the tolerance. In practice, go by trial and error until the desired solute/solvent ratio is reached.

5.4 The *Pretop* module

Reads a *.oeh* file and generates a template topology file with all possible stretch and bend potential sites. Depending on the value of the penultimate *.oeh* entry, The force constants are estimated by a balanced choice between ab initio results on test systems and comparison with force constants in the Gromacs environment. Tentative torsion functions are generated for each atom quartet. The program asks for 3 scaling factors that modulate the estimated force constants of stretching (k_s), bending (k_b) and torsions (k_{tors}), to be given as input from keyboard. The automatic force constants are slightly underestimated for flexible molecules; fudge factors of about 1.5 should favor molecular stiffness reducing excess distortions. A detailed description of the procedure is given in the Appendix, Section A7.
The coordinate reference frame is also changed according to the following procedure. First, crystallographic coordinates are translated into the crystallophysical Cartesian reference frame. Then, atomic coordinates are back-translated so that the origin coincides with the centre of mass and eventually refereed to the inertial reference frame by multiplication of the matrix of inertial eigenvectors.

Running command:

run.pretop NAME

where NAME is the name of the NAME.*oeh* file (Section 1.4.3) and possibly of the NAME.*sla* file (only for MC simulations, see Sections 6.2 and 6.6.4). The program prints an output NAMEpre.*pri* file with various comments and a NAMEtry.*top* file, with the computed topology.

run.pretop module (Unix/Linux)

```
cp $1.oeh pretop.oeh
cp $1.sla pretop.sla
~/programs/MiCMoS/exe/pretop
rm $1try.top
mv pretop.top $1try.top
rm $1pre.pri
mv pretop.pri $1pre.pri
rm pretop.oeh
rm pretop.sla
```

Answer the dialog mode, which will ask for:

ks,kb,kt	Three fudge factors that multiply stretch, bend and torsion force constants given in the output *.top* file. With this tool, the user can tune the force constants; give 1.0 to keep standards, or some factor > 1 or < 1 to increase or decrease the respective force constants.

CAUTION: ***Pretop*** is a tentative procedure and the resulting force constants and interaction sites should be carefully checked. No force field is really universal; in classical simulations of organic crystals, torsional potentials are crucial. A library of such profiles from quantum chemistry has been published in Gavezzotti & Lo Presti, J. Appl. Cryst. 2019, 52, 1253–1263 and is available in Section A7.3 of the Appendix (Table A7.5). A databank of *.top* files of simple organic molecules for MC and MD runs of MiCMoS, is also available on https://sites.unimi.it/xtal_chem_group.

5.5 The *Excbox* module

Excbox trims a mc.*bxi* and mc.*dat* files by deleting all molecules whose centre of mass is farther than a preset limit from the cluster center. This module allows editing the computational box, shaping a cluster of molecules with a roughly spherical radius. This can be useful, for example, to simulate a liquid droplet. Note that you need both a *.bxi* and a *.dat* file to run ***Excbox***, but only the latter must be used, for example, in MD simulations (see Section 7).

Running command:

```
run.excbox name1 name2
```

Here name1 is the name of the *.bxi* (see Section 5.1.1) and *.dat* (see Section 5.1.3) files, while name2 is the corresponding output flag. The program will create name2.*bxi* and name2.*dat* files, which contain only the information on the molecules that survived this trimming procedure.

run.excbox module (Unix/Linux)

```
cp $1.bxi excbin.bxi
cp $1.dat excbin.dat
~/programs/MiCMoS/exe/excbox
rm $2.bxi
mv excbout.bxi $2.bxi
rm $2.dat
mv excbout.dat $2.dat
rm excbin.*
```

Answer the dialog mode, which asks for:

dimu,dimv	Cutoff distance for solute and solvent molecules. Molecules more distant by dimu or dimv from the system centre of mass (in Å) will not be included in the final cluster.
nslavu,nslavv	Number of slave atom cards (see Sections 6.6.3 and 6.6.4) for solute and solvent, if any. Input 0 if no slave atoms are present.

The program checks for consistency of the number of solute and solvent molecules present in *.bxi* and *.dat* files. If the counts do not correspond to each other, ***Excbox*** stops with an error message.

CAUTION: The program does not update the number of molecules in the output .dat and .bxi files. The user must edit such files and correct the header.

5.6 The *Nanosolv* module

Nanosolv merges two *.dat* frame files, one containing a nanoparticle produced by the routine ***Nanocut*** (Section 8.9), and the other a liquid, possibly equilibrated, produced by either the ***Boxliq***+MC or MD procedures. The program makes the centre of mass of the nanoparticle coincident with that of the liquid and erases all the solvent molecules that lie at contact distance with any of the atoms in the nanoparticle. The resulting *.dat* file should be a reasonable starting point for any subsequent MD simulation.

Running command:

run.nanosolv name1 name2

Where “name1” is the name of the *.dat* frame of the nanoparticle, and “name2” that of the liquid.

run.nanosolv module (Unix/Linux)

```
rm $1solv.dat
cp $1.dat nanoparticle.dat
cp $2.dat liquid.dat
~/programs/MiCMoS/exe/nanosolv
mv nanosolv.dat $1solv.dat
rm nanoparticle.dat
rm liquid.dat
```

The user is prompted to give from keybord the following quantity:

atol	Tolerance parameter (in Å) to define the allowed contact distance between solute (nanoparticle) atoms and solvent ones. A solvent molecule is erased if and only if any of its atoms *i* lies closer than rvdW(i) + rvdW(j) + atol to any solute (nanoparticle) atom *j*. rvdW(i) and rvdW(j) are the corresponding van der Waals radii of the two atoms, taken from J.D. Dunitz, A. Gavezzotti, Attractions and Repulsions in Molecular Crystals, Acc. Chem. Res., 1999, 32, 677.

The program produces a third *.dat* frame, called name1solv*.dat*, with the solvated nanoparticle. Some useful information (number of erased molecules, atom count…) is printed on screen.

If the number of molecules in the liquid phase is reduced by more than 50 % during merging, the user is prompted to choose to either stop the calculation or to continue anyway. Note that too few molecules in the liquid box might be insufficient to produce a fully solvated nanoparticle, especially if the latter is relatively large.

CAUTION: Note that ***Nanosolv*** cannot deal with slave atoms.

5.7 The *Solution* module

The program *solution.for* merges two *.dat* frame files, both containing (possibly) equilibrated liquids. The idea is to prepare a solution with the desired concentration, ready to enter the Molecular Dynamics module without the need of pre-equilibration with Monte Carlo. The solute box is put at the centre of the solvent box. A certain number of solute molecules is conserved, depending on the desired concentration. Then, all the solvent molecules that lie too close to any of the surviving solute molecules are erased. It is applied a proximity criterion: any solvent molecule with any atom closer than $\Sigma[R_{\mathrm{vdW}}]+atol$ to any solute atom will be erased. R_{vdW} is the van der Waals radius (see taken from J. D. Dunitz, A. Gavezzotti, Acc. Chem. Res., 1999, 32, 677) and *atol* is a user-defined tolerance parameter, expressed in Å, that is given from keyboard when the program is executed.

Current program limits are 10,000 atoms and 8,000 molecules in the two boxes. The running command is:

run.solution name1 name2

where name1 is the box containing the <u>solute</u> molecules, and name2 that containing the <u>solvent</u> molecules. The output is a third *.dat* box file named name1solv*.dat*, which contains the desired number of solute units embedded in solvent molecules.

run.solution module (Unix/Linux)

```
#
rm $1solv.dat
cp $1.dat solute.dat
cp $2.dat solvent.dat
~/programs/MiCMoS/exe/solution
mv solution.dat $1solv.dat
rm solute.dat
rm solvent.dat
```

The user is prompted to give from keybord the following quantities:

conc, *atol*,*enlarge*	*conc* is the desired concentration, in mol·L^{-1}. A number of solute molecules compliant with *conc* will be selected randomly from the solute box name1*.dat*. *atol* is the contact tolerance parameter, in Å. The two starting boxes will be superimposed, and the minimum allowed contact distance for solute-solvent atom pairs *i*,*j* will be $R_{\mathrm{vdW}}(i)+R_{\mathrm{vdW}}(j)+atol$. Any solvent molecule with one or more contacts closer than this limit with surviving solutes will be erased. Larger *atol*'s mean that more solvent molecules will be eliminated. *enlarge* is the cell enlargement factor. It has been introduced since MiCMoS v2.3 to avoid clashes on the borders of the merged simulation box. These could arise if one or more solute molecule falls close to the box boundaries. This parameter must be 1.0 or greater; cell edges of the

final box are multiplied by *enlarge*. Thus, for example, 1.2 means that the cell boxes are enlarged by 20 % in the initial frame.

CAUTION: Note that ***Solution*** cannot deal with slave atoms.

CAUTION: Obviously, the resulting merged box is *not* equilbrated, as it may contain voids due to the elimination of solvent units. Moreover, some residual solute-solvent steric clashes could survive, especially if *atol* is low or zero. If this is the case, re-run ***Solution*** by increasing *atol*. Probably, you should play a bit with concentration and *atol* to produce a satisfactory starting box.

Re-equilibration of the solution through Molecular Dynamics might cause the volume of the box to change. This in turn implies that the actual concentration could be slightly different from the one set up in your keyboard input. It is always wise to check the actual concentration c (in mol·L^{-1}) at equilibrium according with:

$$c = \frac{n_{solu}}{N_A} \cdot \frac{10^{27}}{V_{box}}$$

Where n_{solu} is the number of solutes, V_{box} is the volume of the simulation box in Å^3, N_A is the Avogadro constant (6.02214076·10^{23} mol^{-1}) and 10^{27} is the conversion factor from Å^{-3} to L^{-1}.

5.8 The *Confbox* module

The program *confbox.for* allows to shape a previously equilibrated liquid box with extension *.dat* into a simulation box suitable for molecular dynamics run in confined space. Please refer to Section 7.2.5 for a complete explanation of the confinement algorithm.

The routine *confbox.for* fulfills two tasks:

(i) prepares the parameter file barrier*.par*, which specifies the geometrical details of the confined space and the force field parameters of the barrier;

(ii) prepares a new simulation box named nameconfined*.dat*, where “name” is usually the name of the substance you are dealing with, which is ready for the confined simulation. Essentially, *confbox.for* deletes all the molecules which bear an atom in close contact (less than the sum of the van der Waals radii) with the barriers that set the limits of the confined space.

The running command is:

```
run.confbox NAME
```

where NAME correspond to the NAME*.dat* box of any previously equilibrated liquid. The program produces a file barrier*.par* (the parameter file of the barrier) and a new file NAMEconfined*.dat* (the new simulation box). At the same time, the original NAME*.dat* box file is renamed into NAMEunconfined*.dat* to avoid that it erroneously enters the dynamics when the *mdmain* job is executed (see Section 7.6.1).

run.confbox module (Unix/Linux)

```
rm input.dat
rm intest.dat
rm barrier.par
cp $1.dat input.dat
~/programs/MiCMoS/exe/confbox
cat confined.dat >> intest.dat
mv intest.dat $1confined.dat
mv $1.dat $1unconfined.dat
echo ' '
echo '#################################################'
echo '# ORIGINAL .DAT FILE SAVED INTO UNCONFINED.DAT #'
echo '#################################################'
echo ' '
rm input.dat
rm confined.dat
```

The following parameters are requested from keyboard:

inano	type of confinement. Type either 0 for none (full periodic system) or 1 for the nanolayer, 2 for the nanotube and 3 for the nanocavity.
iplane	active confining planes (type either XY, XZ or YZ), not required if inano=3. This instructions sets how the barriers are oriented with respect to the main crystallophysical reference system.
thickness	starting distances between opposite pairs of barriers (in Å). Type 0 to set the thickness of the confined space equal to the corresponding whole box edge.
rvdw, *ispbar*, *offset*	Each barrier is built as a plane made by an uniform grid of dummy atoms (pixels); see Section 7.2.5 for a complete description of the algorithm. *rvdw* is the van der Waals radius of each pixel. *ispbar* is the corresponding atomic species code number, according with the entries in Table 1.1 (Section 1.4.2). Thus, *ispbar* sets the A6, A12 Lennard-Jones parameters of the barrier pixels and determines the functional employed to compute all the molcule-barrier interactions. *offset* is the offset distance (in Å) between the barrier and the starting simulation box boundary. A value greater than zero is useful to avoid the deletion of a large number of molecules (see Section 7.2.5 for more details).
iattr	determines whether to use or not the attractive part of the potential for the description of the barriers. 0 means that only the repulsive part is employed, that is, the A6 coefficient is set to 0; *iattr* = 1 implies that the full Lennard-Jones potential is used, that is, both A6 and A12 coefficients are nonzero.
dampk(XY), dampk(XZ), dampk(YZ)	3 scaling factors applied to the force constants along Z, Y, and X direction, used to tune the stiffness of the barrier. See Section 7.2.5 for full information.

zacsize, *nmolzacu*, *nmolzacv*

zacsize is the target equilibrium distance between the barriers of the nanolayer or the nanotube. During the simulation, the barostat will take care of modifying the distance between opposite pairs of barriers to reach the desired thickness, if physically possible. For cubic nanocavities, *zacsize* is automatically set to 0 as the equilibrium barrier-barrier distances corresponds to the edge lengths of the cavity, which are set by the program according to the target packing efficiency (see below).

nmolzacu is the number of molecules to consider for the determination of the equilibrium volume of the simulation box. By default (*nmolzacu* = 0), the program uses the number of molecules in the original box to set the volume that corresponds to a packing efficiency, C_{pack}, of 0.66, which is the theoretical limit for close packing of random spheres (see Zaccone, Phys. Rev. Lett., **2022**, 128, 028002).

For the nanocavity, no periodic directions exist, and the edge length of the cubic space is computed as:

$$l = \sqrt[3]{nmolzacu \cdot V_{mol}}$$

For the nanolayer, the lengths of the periodic edges of the simulation box are computed taking into account the desired *zacsize*:

$$l = \sqrt[2]{\frac{nmolzacu \cdot V_{mol}}{zacsize}}$$

When dealing with the nanotube, the formula becomes:

$$l = \frac{nmolzacu \cdot V_{mol}}{zacsize^2}$$

Thus, the *nmolzacu* parameter may be increased to achieve lower packing efficiencies, as it increases the dimensions of the simulation box. This is exploited along the periodic directions in the nanolayer and in the nanotube. If one tries to use *nmolzacu* values lower than those in the original fully periodic simulation box, the program stops and issues a warning. Note that no molecules are erased in any case.

nmolzacv has the same meaning as *nmolzacu* but refers to solvent molecules.

CAUTION: The confinement procedure was not tested for solutions, that is, in the presence of simulation boxes containing both solute and solvent. Please report any issue you may experience to leonardo.lopresti@unimi.it.

5.8.1 Format of barrier.*par* file

This file is created by confbox.*for* (see Section 5.8) and contains all the geometric and force field information necessary to build the barrier(s) of confined space MD simulations. The parameters are mostly the same described in Section 5.8 and are automatically written by confbox.*for* according to the user's choices. The reading format is free; in the following table, all the parameters that begin with "*i*" or "*n*" are integers; the others are floating. It is also possible to edit this file manually if desired (e.g., to simulate non-neutral barriers).

1) #comment line
2) *iplane*(XY), *iplane*(XZ), *iplane*(YZ), *iattr*

iplane()	3 values that set the confining planes to use (plane XY, XZ, YZ) =0 inactive =1 active
iattr	Determines whether to use the attractive part of the potential for the description of the barriers =0 repulsive-only potential =1 full van der Waals potential

3) #comment line
4) *ispbar*, *rvdw*, *qqbar*, *offset*

ispbar	atom type to describe the C_6 and C_{12} Lennard Jones parameters of the pixels that make up the barrier
rvdw	radius of the pixels
qqbar	charge of the pixels. It is defined here for future program improvements and is set to 0 by confbox.
offset	offset distance (in Å) between the barrier and the starting simulation box to avoid the deletion of a large number of protruding molecules

5) #comment line
6) *dampk*(XY), *dampk*(XZ), *dampk*(YZ)

dampk()	3 scaling factors applied to the force constants along Z, Y, and X direction, used to tune the stiffness of the barrier.

7) #comment line
8) *zacsize*, *nmolzacu*, *nmolzacv*

zacsize	equilibrium distance between opposite barriers; it is set to 0 for the nanocavity. See Section 5.8 for a full explanation.
nmolzacu	number of molecules to consider for the determination of the equilibrium volume of the simulation box. See Section 5.8 for more explanations.
nmolzacv	the same of *nmolzacu*, for the solvent molecules.

6. Monte Carlo (MC) simulation

6.1 Overview

MC requires a central computational box with a sample of the system under consideration, typically 300-2000 molecules. Periodic boundary conditions can be applied in one, two or three dimensions. Temperature and pressure control are included when applicable. The main module uses a force field (topology) file (extension *.top*, Section 6.6.3), a run control file with all the pertinent commands (extension *.mci*, Section 6.6.2), and a starting computational box file (extension *.bxi*, Section 5.1.1). The output consists of energies (*.ene*, Section 8.5.1) and trajectories (in format *.dat*, Section 5.1.3). A snapshot of the simulation box corresponding to the final trajectory frame is also printed (format *.dat*), as well as an extended printout summarizing the general program outcomes at user–specified time intervals (again in format *.dat*).
The final frame (*.mco*), the energy file (*.ene*) or the whole trajectory (*.mcc*) are analyzed using the ***Analys, Correl, Geomet*** and ***Redene*** modules (Section 8). Some information at running time is also printed on screen.

6.2 Construction of molecular frameworks

In a MC run molecules are assigned first six rigid-body degrees of freedom (d.o.f.). To allow for molecular flexibility, *i.e.*, to explicitly deal with internal degrees of freedom, the present MC code partitions each molecular object into a number of "**core**" atoms (≥ 3) and a number of "**slave**" atoms (≥ 0). The coordinates of the core atoms are given numerically, and define an invariant rigid part of the molecule, that can be a group whose instantaneous distortions are irrelevant, such as a phenyl ring. While the relative positions of core atoms never change, slave atoms rely on an implicit definition, that is, only their distance and relative orientation with respect to neighboring atoms are specified. For example, the three hydrogen atoms in a rotatable methyl group may be defined by setting a single C–H parameter for the C–H distances, a single parameter for the H–C–C bond angles and a torsion that describes the overall orientation of the $-CH_3$ group. In this way the internal molecular d.o.f.'s (bond distances, bond angles and torsions) can be changed by random moves as well; for each MC step, the explicit coordinates of slave atoms are recomputed while the internal d.o.f.'s vary. In practice however, for the simulation of condensed states of organic small molecules only torsional variables are relevant in most cases, and distance/angle ones can be safely left unchanged. A list of available slave group definitions is given in Table 6.2 at the end of Section 6. See also Section 6.6.4 for a detailed description of slave atom parameters.

One or two molecular species are allowed in the computational box, formally called **solute** and **solvent**. For each molecule, solute or solvent, Cartesian coordinates are calculated for slave atoms using current values for all parameters, mostly torsional, thus defining the current conformation, in a local reference frame whose origin is determined by the specification of core atom positions. The molecular object in its current conformation is then "inserted" into the computational box by applying a rigid-body translation vector and a rigid-body rotation by three Euler angles, plus one indicator that specifies the chirality (if needed). The total degrees of freedom are then the conformational ones plus 3+3 rigid-body ones. The approach provides the full range of choices between one completely rigid molecular species and two completely flexible (minus 3 atoms) molecular objects.

One advantage of this way of proceeding over the traditional, all-coordinate approach is that no computing time is wasted in probing irrelevant degrees of freedom (stretching, bending) or in computationally heavy algorithms (*e.g.* SHAKE) to preserve rigid conformations.

6.3 Computational boxes

A computational box is an ensemble on *N*(*u*) solute molecules and *N*(*v*) solvent molecules, enclosed in a parallelepiped box with dimensions *boxx*, *boxy*, *boxz* and angles α, β and γ. The ***Boxliq*** module (Section 5.2) reads a file with orthogonal coordinates of a molecular model (*.oeh* format, see Section 1) and prepares a cubic box containing a number of molecules, a rough start for the simulation of an isotropic liquid. The ***Boxcry*** module (Section 5.1) reads a *.oeh* file with crystallographic information and prepares an oblique box with multiples of the crystal unit cell in three dimensions. Any pair of boxes coming from *Boxliq* or *Boxcry* modules can be merged using module ***Boxsol*** (Section 5.3). This module produces a solvation box by deleting solvent molecules in close contact with solute molecules.
Note: a maximum of 2000 molecules with at most 100 atoms per molecule is allowed.

6.4 Force fields

The total MC configurational energy is a sum of intramolecular stretch, bend, torsion and non-bonded terms, and intermolecular terms:

$$E(tot) = \left[\sum_i E_i(strecth, bend) + \sum_i \varphi_i(\tau) + \sum_i u_i(R, intra)\right] + \sum_i u_i(R, inter) = \\ = E(intra) + E(inter) \qquad (6.1)$$

where each summation runs over the appropriate number of degrees of freedom. The first term accounts for stretching and bending potential (Section 6.4.1); φ represent the torsion potential (Section 6.4.1), *u*(*R*, *intra*) the intramolecular potential due to non–bonded interactions at distance *R* (Section 6.4.2), and *u*(*R*, *inter*) the intermolecular potential (Section 6.4.3).

6.4.1 Stretching, bending and torsion

Bond stretching and bond angle bending potentials are provided in the quadratic forms:

$$E(stretch) = \frac{1}{2}k_s(R - R^0)^2 \qquad (6.2)$$
$$E(bending) = \frac{1}{2}k_b(\cos\vartheta - \cos\vartheta^0)^2 \qquad (6.3)$$

As already mentioned, these are seldom applied in the Monte Carlo simulation of condensed phases of organic molecules. The torsional intramolecular part $\varphi(\tau)$ is a trigonometric function in τ ($0 < \tau < 180°$):

$$\varphi(\tau) = K\{1 + f \cdot \cos[n\tau]\} \qquad (6.4)$$

where *n* is a frequency term that can be set equal to 1, 2 or 3 and *f* a phase factor, which can be either +1 or –1. *n* and *f* are specified by the user in the *.top* file.

CAUTION: Torsional potentials are indispensable in molecules with rotatable bonds. For example, biphenyl is usually modeled as an object formed by two rigid moieties joined by a disposable, torsional degree of freedom.

The procedure to determine torsion angles according to standard conventions is summarized in the Appendix, Section A8.

6.4.2 Intramolecular non-bonded interactions

Intramolecular nonbonded interaction energies are described by the same potentials that are used for intermolecular interactions, that is, CLP or LJC, damped by a factor FACTIN set in the run control file *.mci*. Values of 0.5-0.7 usually apply successfully. The intramolecular contacts to be considered are specified in an input pair list, chosen among sensitive 1–*n* distances with $n \geq 4$ (obviously) not within rigid core atom groups.

CAUTION: This procedure is somewhat improper because the summation of Coulombic energies runs on a sporadic number of terms and not on a neutral ensemble. These "energies" have anyway little physical significance and should only act as a mean of preventing hard contacts in extreme molecular conformations (*e.g. cis*–*n*–butane) or as a way of taking into account stabilizing intramolecular hydrogen bonding in a tentative way. To summarize, this should be taken as a better-than-nothing procedure. However, it is computationally very cheap and significantly improves the quality of the simulation results.

6.4.3. Intermolecular force fields

These are the CLP or LJC schemes described in detail in Sections 2 and 4. Following a well established convention, intermolecular energies are subdivided in Coulombic and non-Coulombic terms, the latter sometimes going under the name (a misnomer) of "van der Waals" terms:

$$E(tot, inter) = \sum_{i,\alpha} \sum_{j,\beta} [E_{ij}] = E(disp) + E(Coul) \qquad (6.5)$$

where α and β denote different molecules, E_{ij} is the total potential energy between atoms i and j, and $E(disp)$ and $E(Coul)$ are the total non–Coulombic and Coulombic terms for molecule–molecule interactions.

CAUTION: Summations (6.5) are truncated by applying a centre–of–mass distance cutoff that is specified in the input *.mci* file (Section 6.6.2). This cutoff applies to centres of coordinates rather than to single atom-atom distances. In this way, the sums always span entire molecules (neutral charge units) thus substantially reducing truncation effects even if no convergence correction is applied. For polar crystals, the van Eijck–Kroon energy correction for polar boxes (Section 3.1.5) to Coulombic energies can be applied.

6.5 Simulation details

The degrees of freedom (*dof*) are three center-of-mass coordinates and three orientation angles for each molecule in the box, plus the slave-atom parameters (Section 6.6.4). A tag in the input *.mci* file specifies whether each parameter is to be altered or kept fixed during the run. The maximum stepsizes for each kind of MC move, molecular translation, rotation, or change in internal *dof*'s, are also specified.

When box periodicity is imposed, each molecule in the original box has translated counterparts for a ±1 addition of three periodicity vectors. As soon as the center of a molecule moves outside the box boundary, the molecule re-enters the box at the opposite end.

A MC move or action can be any of the following:

a) Variation of molecular dof's. A random number $0 < r_1 < 1$ is generated and the number n of the parameter P to be varied is determined as $n = \text{int}(r_1 \cdot N_{\text{tot}}) + 1$ where N_{tot} is the total number of variable parameters and "int" denotes the integer part of the number. Positive or negative steps are taken by using $P' = P° + (r_2 - 0.5)*step$ with r_2 another random number. Step magnitudes for each dof are specified in the *.mci* input file (Section 6.6.2, instruction 8).

b) Suppression of center-of mass drift. The overall motion of the center of mass of the whole box can be stopped at selected intervals during the simulation, by resetting all molecular position vectors to the origin in the current center of mass.

c) Box dimensions change. The box dimensions can be changed at selected intervals according to the *nboxc* input parameter (see Table 6.1 and Section 6.6.2), with or without pressure control. Isotropic variation with cubic boxes (liquids), or anisotropic variation with oblique boxes (crystals, Figure 6.1) are possible. The computational box is identified by three box edges, *a*, *b*, *c*, and three box angles, α, β, γ. Whenever one of these box dimension is changed by a MC step, the positions of all molecules undergoes a rigid-body change by the following procedure:
(i) calculate fractional coordinates for the molecular centres of coordinate (com) in the old box metrics;
(ii) calculate new orthogonal coordinates of the centres in the new box metrics;
(iii) calculate the components of the displacement of each molecular centre, *dx*, *dy*, *dz*, and apply the same displacement vector to all atoms in the molecule. The process is repeated until all the atoms in teh computational box are translated. This procedure avoids the molecular structure distortion that would accompany a change of metrics on atomic coordinates.

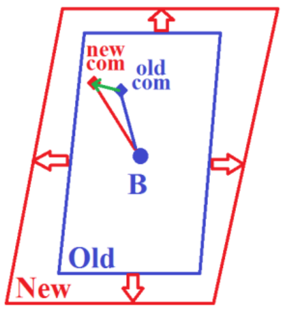


Figure 6.1. A scheme showing how the displacements after a change in box dimensions are calculated. B is the centre of mass of the old cell (blue) with "old" metrics. The "new" box metrics is in red. The green vector is the displacement vector of the centres of mass from coordinates in the old and new metrics, that is applied to all atomic coordinates.

d) Pressure control. There is no temperature rescaling in MC, as temperature is a preset invariable parameter. When box periodicity is present and box dimension are allowed to vary pressure control is possible by the standard **isothermal-isobaric ensemble** (IIE) method. The advantage is that the procedure samples directly from the proper probability density of the NPT ensemble, and is therefore rigorous from the statistical thermodynamics point of view. Its application is straightforward for isotropic and anisotropic cases, and is also relatively fast, as it does not involve a calculation of forces. However, pressure is set at a constant value *P*, thus pressure fluctuations are not allowed.
When cell dimensions are varied the new total box energy *U'* and the new volume *V'* are calculated, along with the quantity δH:

$$\delta H = U' - U^0 + P(V' - V^0) - NkT \ln\left(\frac{V'}{V^0}\right) \qquad (6.6)$$

For more information, see Allen, M.P. and Tildesley, D.J. (1989) *Computer simulation of liquids*, Oxford University Press, 1989, pp. 41 and 124). The usual Metropolis algorithm (see Section 6.5.1) is implemented on δH instead of ΔU. The second term is interpretable as expansion work, while the third term has no immediate interpretation. *N* is the number of "particles" in the system, that comes originally from simulations of simple monoatomic species and is somewhat ill-defined in a simulation involving large flexible molecules. In the default option it is taken as the number of molecules but could as well be the number of flexible sub-molecules or even the number of atoms, depending on the structure and setup of the molecular object (options are provided at running time). By appropriate combinations of input control indices, the program allows runs without box periodicity (isolated clusters, no *P* control); with box periodicity; no box change (NVT simulation); box change without *P* control (sometimes useful is intermediate steps); box change with isotropic or anisotropic IIE *P* control. A full list of options is provided in Table 6.1.

Table 6.1
Pressure and box control options. Sequence of parameters to be used in the input *.mci* file depending on your purpose. Refer to Section 6.6.2 for detailed explanation of the meaning of individual parameters. An integer *n* means that changes in box dimensions (*nboxc*) or pressure checks of the isothermal–isobaric ensemble (*npres*) are carried out every *n* simulation steps.

#	Options	nboxc	npres	ianis
(1)	no box change, no *P* control, NVT run	0	0	0
(2)	isotropic box change every n steps, no *P* control	*n*	0	3
(3)	anisotropic box change every n steps, no *P* control	*n*	0	4
(4)	as (3), with van Eijck-Kroon box dipole energy correction	*n*	0	5
(5)	isotropic IIE Pressure control, liquids	0	*n*	3
(6)	anisotropic IIE Pressure control, crystals	0	*n*	4
(7)	as (6), with van Eijck-Kroon box dipole energy correction	0	*n*	5

6.5.1 The Metropolis criterion

Each MC move (either a parameter change or a box periodicity variation without pressure control) is accepted according to the usual Metropolis criterion: calling ΔE the energy change, each move is accepted if $\Delta E < 0$ or, when $\Delta E > 0$, it is accepted only if $\exp(-\Delta E/RT) > r$, where *r* is a random number between 0 and 1. *T* is the formal, constant temperature parameter of the MC run. Setting *T* very small (say 10 K) is then equivalent to forced energy decrease and sets MC into an excellent energy-optimization tool.

6.6 Running a Monte Carlo job

Templates and standard values for all the data input files are available in the manual. Tutorials are also avialable to provide working examples.

6.6.1 Batch runfile

The files needed to run a Monte Carlo calculation for a compound NAME are:

- NAME.*top*, coordinate and force field input;
- NAME.*mci*, the run control file;
- name2.*bxi*, a file with the box description in a special MC-box format prepared by *Boxcry* or *Boxliq*.

A typical MC run provides the following output:

- name3mc.*pri*, output, printfile
- name3mcc.*dat*, output, trajectory file in *dat* format (Section 5.1.3)
- name3mco.*dat*, output, final frame position file in *dat* format (Section 5.1.3)
- name3.*bxo*, output, final box file for restart, MC-box format (Section 5.1.1)

- name3.*ene*, output, energy trajectory file (Section 8.5.1)
-

Monte Carlo running command

run.mcmain NAME name2 name3

where NAME must correspond to the NAME.*mci* control file and NAME.*top* topology file; name2 is the full name of the input box file name2.*bxi*, and name3 is the prefix of all output files.

run.mcmain module (Unix/Linux)

`cp $1.mci    mc.mci`	MC control file
`cp $1.top    mc.top`	forcefield file
`cp $2        mc.bxi`	input box file, MC-box format
`~/programs/MiCMoS/exe/mcmain`	run execution module
`mv mc.mcp    $3mc.pri`	output, printfile
`mv mc.mcc    $3mcc.dat`	output, trajectory file in *dat* format
`mv mc.mco    $3mco.dat`	output, final frame position file in *dat* format
`mv mc.bxo    $3.bxo`	output, final box file for restart, MC-box format
`mv mc.ene    $3.ene`	output energy trajectory file

Detailed explanations on the meaning of the control parameters and file format follow below.

6.6.2 The MC run-control file (*.mci*)

CAUTION: #lines are comment lines at fixed places: do not change their position or introduce new # lines! These #lines usually show the identity of the various parameters but may contain user's own comments.

Extension . *mci*; all free format.

1) A title line
2) #line --------------------------------
3) iprint, ivarib, iwrh, ipots, FACTIN

iprint	Controls the amount of information printed in the *.pri* output file. =0 minimum printout, =1, =2 detailed printout
ivarib	How to treat *N* in eq. (6.6) =0, N = n.of molecules (normal use, recommended); =1 N = n.of molecules times torsional degrees of freedom
iwrh	Printout for H coordinates =0 write (CH) H-atom coordinates on output trajectory file =1 write H-solute not solvent, =2 H-solvent not solute, =3 no H written
ipots	Controls the energy functional of the Force Field. =0 use AA-CLP =1 use AA-LJC
FACTIN	Damping factor for intramolecular nonbonded interactions (see Section 6.4.2).

4) #line --------------------------------
5) cutoff, boxx, boxy, boxz, alf, bet, gam, variation indices (3 for lengths, 3 for angles), irbox

cutoff	max distance between centres of coordinates in <u>intermolecular</u> energy calculation
boxx, boxy, boxz	three box edges in Å (only if irbox =0 – see below; otherwise, set =0.0)
alf, bet, gam	three box angles in deg (only if irbox =0 – see below; otherwise, set =0.0)
IBX,IBY,IBZ	control whether a change in box length parameters is allowed or not. The absolute values of IBX, IBY and IBZ determine the maximum step as |IBX|·0.01 Å along *a*, |IBY|·0.01 Å along *b* and |IBZ|·0.01 Å along *c*. =0 do not vary the corresponding box length ≠0 vary the corresponding box length; if < 0, couple those cell lengths for which the indices are negative: *e.g.* –5 –5 5 means *a*=*b*
IAL, IBE, IGA	same as IBX,IBY,IBZ above, now for cell angles. Coupling is no allowed for angles. The maximum step is computed depending on to the absolute values of these indices, according to: |IAL|·0.1° for α, |IBE|·0.1° for β and |IGA|·0.1° for γ. =0 do not vary the corresponding parameter; ≠0 do vary.
irbox	Determines where the lattice parameters are to be read in. = 0 use box dimensions given in the *.mci* file, as detailed above. ≠ 0 read box dimensions from the input box file (*.bxi*).

6) # line --------------------------------

7) temp, n.moves, ncom reset, nbox reset, nwri, nwre, npri/steps

temp	set temperature (in K).
nmoves	total number of MC moves.
ncom	stop the drift of the overall centre of mass every ncom steps (see Section 6.5).
nboxc	change box every nboxc steps; for box variation control, see Table 6.1 above. More information in Section 6.5.
nwrite	write trajectory coordinate file every nwrite moves (output file NAME*mcc.dat*)
nwre	write the energy data in the output file *.ene* (Section 8.5.1) every nwre moves;
npri	print accepted energies every npri moves (on screen and printfile, *.pri*)

CAUTION: All the outputs specified on line (7) can be suppressed by setting the corresponding index to zero (except npri). If nmoves =0 the program stops after computing starting energies.

8) 10 steps — 10 floating numbers that control the maximum-stepsizes.
Each of them is associated with a d.o.f. type as indicated by its sequence number in *.bxi* or *.sla* files. Therefore, the actual meaning of the various entries may change according with the user's needs. In any case, the actual stepsize that is applied is always (*rand*–0.5)·*step*, where *rand* is a random number, and *step* is any of the specified stepsizes. Molecular and internal *dof*'s are associated to entries in the maximum stepsize array through the step type numbers *n* indicated either in the *.sla* file (Section 6.6.4, NCARDU/NCARDV instructions – internal *dof*'s) or in the *.bxi* /*.bxo* files (Section 5.1.1, Table 5.1, NMSOLU/NMSOLV blocks, molecular translations and rotations). Advisable values are 0.3 Å for centre of mass motion, 20° for Euler angle variation (rotational *dof*'s of the whole molecule), 5° for torsion angle variations. Larger stepsizes may help in the preliminary energy minimizations of approximate box construction.
An explanatory example on how all this works is provided in Section 6.6.4.1.

9) # line ------------------------------------

10) Pressure, npres, ianis

P		Set overall pressure, in bar.
npres		Determines the number of steps after which the box dimensions are changed. The Metropolis algorithm is then applied to see whether the change is accepted, depending on δH (equation 6.7). For box variation control, see Table 6.1 above. More information in Section 6.5.
ianis		Controls how the cell shape is changed. For box variation control, see Table 6.1 above. More information in Section 6.5.
	=0	No changes (box fixed).
	=3	Only isotropic changes are allowed.
	=4	Anisotropic changes are applied.
	=5	As *ianis*=4, and the Eijck-Kroon box dipole energy correction is also applied. This is meaningful only if your space group is polar. See also Section 3.1.5.

6.6.3 The forcefield input file (*.top*)

The ***Pretop*** module (Figure 4.1, Section 5.4) reads an *.oeh* file and prepares the best possible approximation to the pertinent force field file, except for the separation between core and slave atoms that must be handled by the user (Section 6.6.4). Otherwise, use of templates available in the Tutorials (deposited on https://sites.unimi.it/xtal_chem_group) will make things easy. In the *.top* file, as follows, all data except the title line is free format.

Extension *.top*; all free format.

1) A title line	format 1x,10a4
2) NCOREU	number of core atoms, solute
NCOREU lines	core atom id number, *x*, *y*, *z*, flag for atom species (see Table 1.1), raw charge
3) NSLAVU	number of slave atom lines, solute (see Section 6.6.4).
NSLAVU lines	n_1, n_2, n_3, n_4, n_5, n_6 integer codes (see Table 6.2 for meaning), flag for atom species (see Table 2.1), raw charge. See also Section 6.6.4 for correspondence of n_1, n_2, n_3 numbers in the *.sla* file.
4) NCOREV	number of core atoms, solute
NCOREV lines	core atom id number, *x*, *y*, *z*, flag for atom species (see Table 2.1), raw charge
5) NSLAVV	number of slave atom lines, solvent (see Section 6.6.4).
NSLAVV lines	as with NSLAVU lines.
6) VOLUU, VOLUV	approximate molecular volumes for solute and solvent. They are estimated on the basis of the van der Waals atomic radii and can be useful only for an estimate of cluster volumes; they are both supplied by ***Pretop*** (Section 5.4).
7) NSTRU	number of bond stretching functions
NSTRU lines	4 entries, as follows: two atom id numbers of the atoms involved in the bond, k_S and $R°$ for $E(stretching)=1/2\cdot k_S\cdot(R - R°)^2$, equation (6.2). See Section A7.1 in the Appendix for suggestions on meaningful k_s parameters if needed: in most cases stretching degrees of freedom are irrelevant in MC simulations and are not sampled.
8) NSTRV	as NSTRU (bond stretching), for the solvent
NSTRV lines	as NSTRU lines (bond stretching parameters), for the solvent
9) NBENDU	number of bending function, solute
NBENDU lines	5 entries, as follows: three atom id numbers of the atoms involved in the bending interaction, k_b and $\theta°$ for equation (6.3), $E(bending)=1/2\cdot k_b\cdot(\cos\theta - \cos\theta°)^2$. See Section A7.2 in the Appendix for suggestions on meaningful k_b parameters if needed: in most cases bending degrees of freedom are irrelevant in MC simulations and are not sampled.
10) NBENDV	as NBENDU (bond bending), for the solvent
NBENDV lines	as NBENDU lines (bond bending parameters), for the solvent

11)	NTORSU	number of torsion functions, solute
	NTORSU lines	7 entries, as follows: four atom id numbers, identifying the atoms involved in the torsion; *K*, *f* and *m* parameters in E(*tors*) = *K*{1 + cos *f* [*m*τ]}, equation (6.4). The program ***Pretop*** assigns just standard values for *K* (50), *f* (–1) and *m* (+1). These **must** be reset with actual values, which can be found for example in Table A7.5 (Appendix, Section A7.3). ***Pretop*** also automatically assigns improper dihedrals to keep planar groups with sp^2 hybridization as *K* = 100, *f* = –1 and *m* = +1. You may want to check them, but in most cases no external intervention is required.
12)	NTORSV	as NTORSU, for the solvent
	NTORSV lines	as NTORSU lines (torsion parameters), for the solvent
13)	NLISTU	number of intramolecular contacts, solute

NLISTU pairs of atom id numbers, solute, for a total of NLISTUx2 entries (see Section 6.4.2). These flag the intramolecular contacts, for which a FACTIN dampening factor is applied to scale down the potential (see Section 6.4.2). FACTIN must be given in the *.mci* instruction file (Section 6.6.2).

14)	NLISTV	number of intramolecular contacts, solvent

NLISTV pairs of atom id numbers, solvent, for a total of NLISTVx2 entries. See NLISTU above for explanation.

15)	FQ, FP, FD, FR	force field scaling parameters in eq. (2.1) (standards: 0.41, 235, 650, 77000); zero if the LJC force field is used.

Add the following instructions only if Lennard-Jones potentials are used (IPO=1 in the *.mci* or *.mdi* file, Sections 6.6.2 and 7.6.2):

16)	NEXTRA	number of extra L-J parameters. Non-zero only if non-library 6-12 parameters are used.
17)	NEXTRA lines	I, J, A6, A12 in equation (2.6). A6 and A12 are the 6-12 coefficients for the atom-atom contact between atom species *i* and *j*.

6.6.4 The slave atom parameter file (*.sla*)

The NCARDU and NCARDV lines in the preceding section 6.6.3 (instructions (3) and (5)) specify the slave atom construction codes. To use the ***Boxcry*** or ***Boxliq*** modules to prepare the starting computational boxes, the actual values for the parameters of the slave atoms must be supplied in a separate file, extension *.sla*. If there are no slave atoms, in a fully rigid molecule, no *.sla* file need be prepared and these modules will ignore them.

The advantages of using a semi-rigid approach will counterbalance the apparent complexity of preparing the *.sla* files. With a little practice and use of worked examples the procedure will become routine.

Extension: *.sla;* all free format.
The *.sla* files contain:

1) NCARDU	number of slave atom groups, solute

NCARDU lines	bond distance; bond angle; torsion angle; step type number (distance); step type number (angle); step type number (torsion); 3 integer id atom numbers. Bond distance, bond angle and torsion angle must be added according to the conventions specified in Table 6.2 below. Set 0.0 for parameters that are not required; Section 6.6.4.1 describes a worked example. The last 3 integers are the id numbers to be assigned to the slave atoms when their coordinates are computed at each MC step (Section 6.6.4.1 gives a practical example). They will label atoms highlighted in grey in Table 6.2.
2) NCARDV	number of slave atom groups, solvent
NCARDV lines	as above for NCARDU

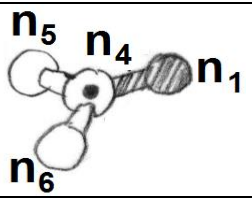
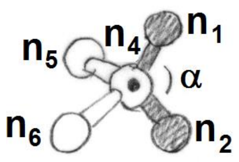
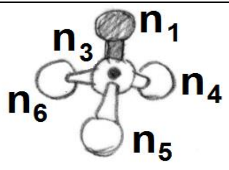
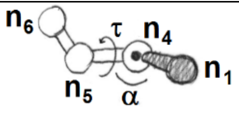
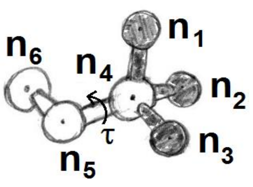
Note that ***Boxcry*** and ***Boxliq*** also automatically load the same instructions in the *.sla* file into the *.bxi*/*.bxo* files that specify the content of the simulation box (see Section 5.1.1).

CAUTION: Care must be taken to number atoms so that the *N* core atoms are numbered from 1 to *N*, *i.e.*, they must come first. Successive slave atoms can be built only if core atoms, necessary for the construction, have been already built.

Summarizing: The NSLAVU and NSLAVV lines in the *.top* file specify the atom numbers and the procedure to be followed to build the slaves among those in Table 6.2; the lines in the *.sla* file specify the actual values of the distance, torsion angle and bond angle to be used as starting values in *.bxi* files. The *.bxo* files will contain the MC-modified values of distances, angles and torsions. Of course any *.bxo* file can be used as input for a continuation MC run. The reason for using two separate files for the specification of molecular geometry is that slave atom parameters are variable for each molecule and cannot be set from the topology file in continuation runs.

All this looks more complex than it actually is and becomes much simpler after the first use; see the following worked example (Section 6.6.4.1).

Table 6.2
Atom positioning options in the Monte Carlo *.top* and *.sla* files. Dark grey atoms in the "group" column are the slaves, generated by the automatic construction procedure for a given NCARDU/NCARDV line in the *.top* file (Section 6.6.3).

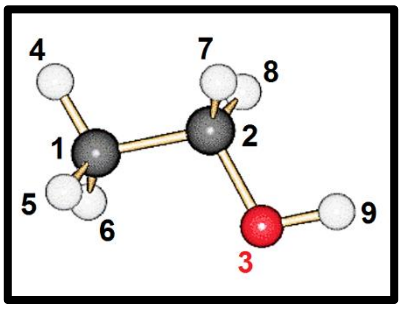
Group	Type	Integer codes in *.top* file		Parameters in the *.sla* file
		≠ 0	= 0	
	Trigonal	n_1, n_4, n_5, n_6	n_2, n_3	n_1–n_4 distance. Last 3 integers[*]: n_1 0 0
	Methylene	n_1, n_2, n_4, n_5, n_6	n_3	$n_{1,2}$–n_4 distance and n_1–n_4–n_2 angle α. Distances n_1–n_4 and n_2–n_4 are set equal. Last 3 integers[*]: n_1 n_2 0
	Methine	n_1, n_3, n_4, n_5, n_6	n_2	n_1–n_3 distance. Last 3 integers[*]: n_1 0 0
	Z–matrix	n_1, n_4, n_5, n_6 $n_2 = -1$	n_3	n_1–n_4 distance, n_1–n_4–n_5 angle and n_1–n_4–n_5–n_6 torsion angle. Last 3 integers[*]: n_1 0 0
	RX_3 group	n_1, n_2, n_3, n_4, n_5, n_6	None	One $n_{1,2,3}$–n_4 distance, one $n_{1,2,3}$–n_4–n_5 angle and one $n_{1,2,3}$–n_4–n_5–n_6 torsion τ. All distances and angles are set equal; torsions are automatically computed from τ as τ+120 and τ+140. Last 3 integers[*]: n_1 n_2 n_3

[*] Sequence of atom id numbers to be specified at the end of each NCARDU/NCARDV string in the description of the *.sla* file. See Section 6.6.4.1 for a practical example.

6.6.4.1 Building slave atoms for ethanol

An example of a valid topology *.top* file (Section 6.6.3) for ethanol (CH_3–CH_2–OH, inset) is given below.

```
    ethanol
      3   ncoreu
    1  -1.53000  0.0000   0.0000   13 -0.9000
    2   0.00000  0.0000   0.0000   13  0.1100
    3   0.44190 -1.3600   0.0000   29 -1.4500
      3   nslav-u
  4  5  6  1  2  3   3  0.3000
  7  8  0  2  1  3   3  0.2700
  9 -1  0  3  2  1   5  0.8000
  0   ncorev
  0 nslav-v
   52.0    0.0 volu-u,volu-v
 0  nstretch -u
 0   -v
 0  nbend-u
 0   -v
 2  ntors-u
 4  1  2  3  3.  1.0  3
 9  3  2  1  2. -1.0  2
  0   ntorsv
  0   nlistu
  0   nlistv
0.41 235.0 650.0 70000.0
   0  nextra
```



The three atoms of the C1–C2–O3 chain are the core atoms. Methyl hydrogens (id numbers 4–6, atom type 3 according to Table 1.1 and raw charge 0.3) are built attached to atom C1 as a RX_3 group (Table 6.2). Methylene atoms 7 and 8 (atom type 3, raw charge 0.27) are built as a “Methylene group” type, *i.e.* they are attached to atom C2 on the bisector of the C1–C2–O3 angle. The alcohol hydrogen (id 9, atom type 5 and raw charge 0.8) is built attached to the O atom by “Z-matrix” (Table 6.2). There are no stretch or bend potentials, but there are two torsional potentials for rotation around the C1–C2 and C2–O3 bonds. Instructions after `ntors-u` define the corresponding force field parameters (see instruction 11 in Section 6.6.3).

Actual MC stepsizes are governed by a *.mci* run–control file (Section 6.6.2). A typical input is:

```
  Ethanol, liquid
#  iprint ivarib iwrh ipots  factin
     0     0     0     0      0.7
#  cutoff  boxx     boxy     boxz    alf  bet  gam  var.indices    irbox
   16.0   0.0000   0.0000   0.0000   0.0  0.0  0.0  1  1  1  0 0 0   1
# temp n.moves  ncom nboxc nwri   nwre npri/steps
 293.    50000    0     0  1000   1000  1000
  0.30 20.0  3.0 5.0 10.0  0.   0.   0.   0.   0.
#   P,  npres   ianis
   1.0  1000      3
```

The floating number sequence “0.30 20.0 3.0 5.0 10.0 0. 0. 0. 0. 0.” corresponds to the maximum stepsize array specified in Section 6.6.2, line 8. The actual step in a MC move is (*rand*-0.5)·*step*, where *rand* is a random number and *step* the n^{th} element of this array. Numbers "*n*" unequivocally associate a certain variable to a definite *step* parameter and are both specified in the *.sla* file (internal molecular *dof*'s) and in the *.bxi* file (Table 5.1, molecular translational and rotational *dof*'s). Note that random changes of the simulation box edges and angles are controlled by IBX, IBY, IBZ and IAL, IBE, IGA parameters described in the instruction (5) of the *.mci* file, Section 6.6.2. Centre of mass translations and rotations of the whole molecule are activated in the *.bxi* file.

The corresponding *.sla file* of ethanol looks like:

```
 3
  1.0800 109.47 180.00  0  0  3   4  5  6
  1.0800 110.00   0.00  0  0  0   7  8  0
  1.0000 109.00 180.00  0  0  4   9  0  0
```

There are 3 chemically different groups to be defined, namely a methyl (hydrogens 4–6), a methylene (hydrogens 7 and 8) and a hydroxy (hydrogen 9). Thus, the *.sla* file also contains 3 lines plus the heading. The first row specifies the methyl: the C1–H distance, bond angle and torsion are 1.08 Å, 109.47° and 180° respectively. The next three numbers (0 0 3) specify that no stretch or bend degrees of freedom are allowed, while the methyl group torsion is allowed by stepsize nº3 in the maximum stepsize array of the *.mci* file (3.0 deg in this example). The last three numbers define the sequence id numbers of atoms that are built by the procedure: for methyl, hydrogens 4, 5 and 6. They correspond to the atoms coloured in grey in the first column of Table 6.2.

The same applies also to other groups, as detailed below.

CAUTION: The order of the lines in the *.sla* file must be the same as of instructions in the topology *.top* file.

In methylene, the C2–H distance is also 1.08 Å, the H–C2–H angle is 110° and there is no need for a torsion angle (Table 6.2). As for the MC step, methylene has no degrees of freedom (0 0 0) and the two hydrogens have id numbers 7 and 8. The O3–H distance is 1.00 Å, the C2–O3–H angle is 109° and the torsion angle is 180° as the O–H bond staggers the methylene group. The alcohol hydrogen has a torsional motion, governed by maximum stepsize nº4 (5.0 deg) in the list specified in the *.mci* file.

7. Molecular dynamics (MD) simulation

7.1 Introduction

The ***Mdmain*** module of the MiCMoS package is designed for the simulation of the dynamics of aggregates of molecular substances, allowing for one or two chemical species, formally a solute and a solvent. It can be used without periodic box conditions for isolated clusters, providing original algorithms for the quenching of translational and rotational global motion, or with periodic box conditions, with (NPT) or without (NVT) isotropic or anisotropic pressure control.

The ***Mdmain*** module (see also Figure 4.1) reads input starting coordinates from a *.dat* file (Section 5.1.3), for liquids typically resulting from a preliminary MC simulation to provide an initial energy optimization (Section 6), or directly from the ***Boxcry*** module for crystals (Section 5.1). Atomic and force field information is in the *.top* file (same format as in MC modules, Section 6.6.3). Input instructions are read from a *.mdi* run-control file. The main module integrates a leap-frog algorithm and produces *.mdo* (final frame, with atomic coordinates and velocities in $m{\cdot}s^{-1}$), *.mdc* (structural trajectory), *.ene* (energy trajectory, Section 8.5.1) files of the same format as their MC counterparts, and *.pri* (line print). In MD there is no use of core and slave atoms and all atomic coordinates are dealt with explicitly. All input and output structural files carry atomic coordinates in Å units and are in *.dat* format.

The final frame (*.mdo*), or a final frame averaged over some last steps, the energy file (*.ene*) or the whole trajectory (*.mdc*) are analyzed using the ***Analys, Correl, Geomet*** and ***Redene*** modules (Section 8). Some information at running time is also printed on screen.

CAUTION: When a large energy jump is detected due to potential or dynamic malfunctions (**a crash**), the program stops printing the crash energies and the crashed frame.

Table 7.1
Units of physical properties in eq.s (7.5)–(7.14). Vector quantities are in boldface. Input-output atomic coordinates and box dimensions are in Å, atomic masses in amu, atomic charges in electrons. The MD programs then convert to and use SI units. Output configurational and molecular energies are in $kJ{\cdot}mol^{-1}$.

Symbol	Physical property	SI unit
P	Pressure	Pascal (Pa)
V	Volume of the computational box	Cubic meters (m^3)
M_i	Molecular mass	Kilograms (kg)
$\mathbf{V}_i$	Velocity of the ith center of mass	Meters per second ($m{\cdot}s^{-1}$)
$\mathbf{v}_k$	Velocity of the kth atom	Meters per second ($m{\cdot}s^{-1}$)
E_{kin}	Total kinetic energy of the centers of mass	Joule (J)
W	Center-of-mass virial	Joule (J)
$\mathbf{F}_{ij}$	Force between centers of mass of i-j molecules	Newton (N)
$\mathbf{R}_{ij}$	Distance between centers of mass of i-j molecules	Meters (m)
$\mathbf{f}_{kl}$	Force between atoms k and l in different molecules	Newton (N)
μ_0	Isothermal compressibility	Reciprocal Pascal (Pa^{-1})
q_k	Atomic charge	Coulomb (C)
$\mathbf{x}$, a	position coordinates or box dimensions	Meters (m)

Atomic coordinates for input are in Å, atomic masses in amu, atomic charges in electrons. The programs then use time in seconds (*s*), mass in kg, lengths in meters (*m*) and velocities in $m{\cdot}s^{-1}$. All energies are in joule or in *kJ*/*mol* (Table 7.1). These units are embedded in the code at several places and cannot be changed by the user.

7.2 MD layout

The main modules read starting atomic coordinates for a computational box in *dat* format (Section 5.1.3). *Note: a maximum of 2000 molecules with at most 100 atoms per molecule is allowed*.

7.2.1 Zero-step atomic velocities

If not present in the input file, starting velocities V can be assigned (in module) by an approximate Maxwellian distribution according to:

$$V = \left[\frac{k_b T}{M}\right]^{1/2} \left[\sum_{i=1}^{12}(r_i) - 6\right] \tag{7.1}$$

where k_b is the Boltzmann constant, T is the set temperature, M is the atomic mass, and r_i is a random number between 0 and 1. For a better randomization of velocities, to reduce translational or rotational biases, the components of any velocity vector **V** are determined as follows:

$$\left.\begin{array}{c} \mathbf{V} = a\mathbf{V_x} + b\mathbf{V_y} + c\mathbf{V_z} \\ b^2 + c^2 = 1 - a^2 \\ b^2 = a'(1 - a^2) \\ c^2 = (1 - a')(1 - a^2) \end{array}\right\} \tag{7.2}$$

Where a' is a random number between 0 and 1. Moreover, each component $\mathbf{V_x}$, $\mathbf{V_y}$ and $\mathbf{V_z}$ is assigned a plus or minus sign according to a random number being grater or smaller than 0.5.

7.2.2 Integration

MiCMoS is equipped with two second order symplectic integrators: the leapfrog and the velocity–Verlet algorithms. They are reasonably simple and are time reversible. Moreover, they both conserve the Hamiltonian of the system. These strengths make them very appealing in MD simulations and well suited to all the applications for which MiCMoS is designed.

7.2.2.1 Leapfrog algorithm

After calculation of potentials and forces the trajectory is integrated by:

$$\mathbf{V}\left(t + \tfrac{1}{2}\Delta t\right) = \mathbf{V}\left(t - \tfrac{1}{2}\Delta t\right) + \frac{\Delta t}{M}\mathbf{F}(t) \tag{7.3a}$$

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \Delta t \cdot \mathbf{V}\left(t + \tfrac{1}{2}\Delta t\right) \tag{7.3b}$$

where the symbols have obvious meaning of time (t), velocity (**V**), position (**r**), mass (M) and force (**F**). **r**, **V** and **F** are three–component vectors and apply to each atom in the simulation box.

7.2.2.2 Velocity–Verlet algorithm

The velocity–Verlet (VV) algorithm can be considered as a variant of leapfrog, which determines **r** and **V** at the same time. The VV equations read as:

$$\mathbf{V}\left(t + \tfrac{1}{2}\Delta t\right) = \mathbf{V}(t) + \frac{1}{2}\frac{\Delta t}{M}\mathbf{F}(t) \tag{7.4a}$$

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{V}\left(t+\tfrac{1}{2}\Delta t\right)\Delta t \tag{7.4b}$$

This means that velocities at the previous half step are no longer required. Now the forces are updated according to the new positions $\mathbf{r}(t+\Delta t)$, and used to update velocities at $t + \Delta t$:

$$\mathbf{F}(t+\Delta t) = \mathbf{F}[\mathbf{r}(t+\Delta t)] \tag{7.4c}$$

$$\mathbf{V}(t+\Delta t) = \mathbf{V}(t) + \frac{1}{2}\frac{\Delta t}{M}\mathbf{F}(t+\Delta t) \tag{7.4d}$$

This algorithm is equivalent to leapfrog, in the sense that it generates identical trajectories if *corresponding* starting points are employed (*i.e.*, if velocities at $t_0 = t - \frac{1}{2}\Delta t$, and not at $t_0 = 0$, are used as VV starting velocities). Otherwise, the two trajectories will be generally different, as leapfrog interprets starting velocities as $\mathbf{V}\left(t_0 - \frac{1}{2}\Delta t\right)$ rather than as $\mathbf{V}(t_0)$. The VV algorithm is slightly more expensive, as it requires to compute forces at t to update positions, and at $t + \Delta t$ to update velocities. MiCMoS saves as much time as possible by storing the vector $\mathbf{F}(t+\Delta t)$ to use it in the ($t + \Delta t$) step, but this can be done only if the coordinates are not further changed after the main integration procedure. In MiCMoS, the routines for stopping the drift and the rotation of the whole cluster (Section 7.5.1), suppressing the evaporation (Section 7.5.2) and applying the barostat (Section 7.3) could all modify the coordinate vector computed by the integrator. Consequently, whenever one of these corrections is applied, the forces must be computed again to account for the updated atomic positions. This implies that the computational cost of the VV algorithm is slightly higher than that of leapfrog and increases with the call frequency of the above-mentioned correction routines.

7.2.3 Temperature control

The temperature of the simulation can be kept at T_{set} in three ways.

7.2.3.1 Stiff coupling

A **stiff coupling** can be used, which means that all velocities are rescaled by a factor $(T_{set}/T)^{1/2}$.

7.2.3.2 Berendsen thermostat

A **weak coupling** procedure can be applied by using the Berendsen thermostat, which implies that a velocity rescaling factor is computed as:

$$\lambda(t) = \sqrt{\left[1 + \frac{dt}{\tau}\cdot\left(\frac{T_{set}}{T} - 1\right)\right]} \tag{7.5}$$

where dt is the MD timestep and τ is the temperature relaxation time (in practice the dt/τ ratio is approximated by an empirical coefficient, 0.5-0.6).

This very simple rescaling procedure is computationally inexpensive but is known to produce artifacts in the trajectories. The reason is that the Berendsen thermostat suppresses the fluctuations of the kinetic energy, preventing a proper microcanonical NVT ensemble from being generated. Unsatisfied detailed balance leads to the unphysical redistribution of energy from high frequency into low frequency modes, resulting in the well–known “flying ice cube” artefact (see for example E. Braun, S. M. Moosavi, B. Smit, J. Chem. Theory Comput. 2018, 14, 5262−5272, https://doi.org/10.1021/acs.jctc.8b00446 for more information). This is actually a serious problem in systems where the accurate description of low frequency modes is mandatory to catch the correct physics, such as isolated clusters and flexible networks. These include, among others, diffusion of small molecules through molecular sieves and metal–organic frameworks. However, when totally rigid or semi–rigid systems are considered, such as perfect crystals at temperatures far from the melting point, the error that is introduced can be safely neglected in most cases. As for the dynamics of isolated clusters (Section 7.5), where sampling of low–

frequency modes is much more important, we tackled the incorrect energy partition by providing MiCMoS with routines that artificially suppress the translation and rotation of the cluster, thus alleviating the overweighting of low–energy translational and rotational modes (see Section 7.5.1 and Gavezzotti & Lo Presti, New J. Chem., 2019, 43, 2077–2084, https://doi.org/10.1039/C8NJ05825C).
If a more accurate temperature control is required, *i.e.*, whenever fluctuations of kinetic energy are important to determine the observables that are looked for, the CSVR thermostat should be preferred (see Section 7.2.3.3).

7.2.3.3 Bussi–Donadio–Parrinello thermostat

The **Canonical Sampling through Velocity Rescaling** (CSVR) **thermostat** is implemented in MiCMoS following G. Bussi, D. Donadio & M. Parrinello, J. Chem. Phys. 126, 014101, 2007, https://doi.org/10.1063/1.2408420. The idea is to improve simpler rescaling algorithms based on kinetic energies or temperatures with a stochastic term, which allows a correct sampling of the kinetic energies of the canonical ensemble. A time–dependent velocity–rescaling factor α is applied, which reads

$$\alpha = \sqrt{\frac{K_t}{K}} \qquad (7.6)$$

Where K_t is the target value for the kinetic energy, as extracted from the canonical equilibrium distribution:

$$\bar{P}(K_t)dK_t \propto K_t^{\frac{N_{dof}}{2}-1} e^{-\beta K_t} dK_t \qquad (7.7)$$

N_{dof} being the number of degrees of freedom and $\beta = 1/k_bT$ the usual Boltzmann's factor. In practice, whenever the thermostat is called, the kinetic energy K is evaluated. Then, an auxiliary continuous stochastic dynamic is used to compute the target K_t, which is expected if the system samples the correct canonical distribution. According to Bussi, Donadio and Parrinello, the stochastic correction to the kinetic energy is

$$dK = (K_t - K)\frac{dt}{\tau} + 2\sqrt{\frac{K_t K}{N_{dof}}}\frac{dW}{\sqrt{\tau}} \qquad (7.8)$$

In (7.8), *dt* is the simulation timestep and τ the time constant of the thermostat. *dW* is a Wiener noise, that is, a gaussian–distributed random perturbation. Whenever $\tau \rightarrow 0$ (in practice, when $\tau < 0.1$), the algorithm ignores the last term of the summation. This implies that the Wiener contribution is instantaneously thermalized, and the algorithm reduces to a stochastic velocity rescaling.

Following Bussi, Donadio and Parrinello, an effective energy $\widetilde{H}$ is defined as the total energy (kinetic + potential) minus the cumulative sum of all the drifts *dK* to the kinetic energy due to the CSVR thermostat (Figure 7.1). The quantity $\widetilde{H}$ should be conserved along the trajectory; the program prints the effective energy whenever the CSVR thermostat is called. It should remain constant when the system is equilibrated. A systematic drift of $\widetilde{H}$ throughout the trajectory is likely a warning of numerical discretization errors, meaning that shorter timesteps and/or more frequent calls to the thermostat are required. Fundamentally, the *dt*/τ ratio is crucial in this respect.

CAUTION: Experience shows that very short time steps d*t*, like 0.00025 ps or lower, might be necessary in most cases to avoid discretization errors when the CSVR thermostat is employed.

Testing was carried out for crystalline paracetamol (phase I, $P2_1/n$) at p = 1 bar and T = 100 K, from a previously Monte Carlo thermalization at room temperature, with the LJC Force Field. Trajectories were all 100 ps long; results averaged over the last 50 ps of the simulation are shown in the table below. For comparison, the experimental cell at 100 K (CSD label HXACAN) is a= 7.0915(3) Å, b= 9.2149(4)Å, c= 11.6015(5) (1) Å, β = 97.865(1) deg, density: 1.337 g/cm^3.

Thermostat → Integrator ↓	Weak coupling / Å, deg, g/cm^3, kJ/mol dt = 0.002 ps	CSVR / Å, deg, g/cm^3, kJ/mol dt = 0.0001 ps
Leapfrog	a = 6.884(2), b = 9.669(3), c = 11.090(4), α = 89.93(3), β = 95.91(4), γ=89.99(2), Density: 1.367(1), E_{coh} = –109.8(2)	a = 6.870(3), b = 9.689(1), c = 11.113(11), α = 89.65(1), β = 95.66(4), γ=89.64(2), Density: 1.364(1), E_{coh} = –109.7(3)
Velocity–Verlet	a = 6.884(3), b = 9.685(5), c = 11.100(4), α = 89.93(4), β = 95.91(4), γ=89.99(2), Density: 1.364(1), E_{coh} = –106.6(2)	a = 6.869(3), b = 9.688(1), c = 11.113(10), α = 89.65(1), β = 95.65(4), γ=89.64(2), Density: 1.364(1), E_{coh} = –109.7(3)

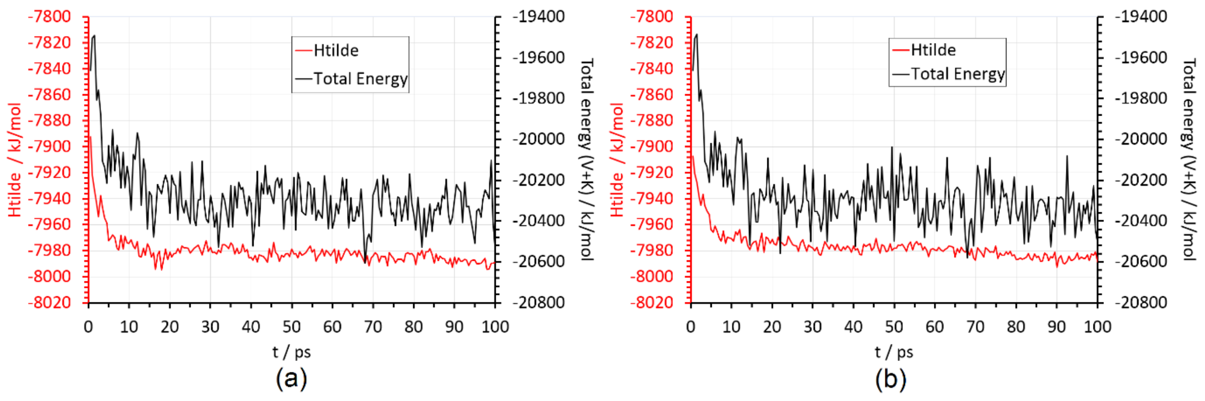


Figure 7.1. Effective energy ($\widetilde{H}$, red) and total energy (kinetic + potential, black) for 100 ps long MD simulation of monoclinic paracetamol at T = 100 K and p = 1 bar using a Parrinello–Rahman barostat (see Section 7.3.3) without external stress field and a thermostat time constant τ = 0.6 ps. The thermostat and barostat algorithms were applied every 100 and 50 MD steps, respectively. (a) Leapfrog; (b) velocity–Verlet.

CAUTION: For isolated small clusters, an accurate description of temperature is impossible; in a cautionary attitude, simulation temperature can be regarded as just a measure of dynamic freedom without much connection with the corresponding thermodynamic quantity.

7.2.4 Bias MD

From *v2.0* onwards, MiCMoS is equipped with the *modivel* routine, which artificially redistributes the molecular kinetic energies to drive the simulation softly toward the "spontaneous" formation of stable clusters and nucleation events. The routine *modivel* is called as the last step of the *mdmain* engine, just before printing the trajectory: this means that it does not affect directly the forces. Rather, it rescales the velocities just before the next integration step.

The idea is to advantage stable molecular pairs and small clusters within a disordered system, like a bulk liquid phase. Such pairs and clusters are always present, mostly as transient entities. They might act as supramolecular synthons during nucleation, especially if their formation is kinetically favoured and they survive the thermal agitation for sufficiently long times. Our biased MD algorithm should be considered as a tool to speed up the process of driving the molecules toward the most effective recognition modes.

Upon self-recognition, molecules form pairs and clusters that are more stable than the individual components. The potential energy in excess is redistributed to the surrounding molecules as heat. We thus expect that molecules outside the cluster will gain kinetic energy, while those inside will get stuck in a local minimum of potential energy that reduces their accessible phase space. To simulate this process, the *modivel* algorithm selects attractive pairs in a user-defined energy range, that is, those whose intermolecular energy E_{ij} lies in between a user-defined interval E_{bias}(lower threshold)–E_{bias}(upper threshold). It is thus possible to advantage specific recognition modes, even at intermediate energies, to try driving the system towards ordered patterns. The advantage of this procedure is that no constraints are imposed on intermolecular geometric parameters (*e.g.* centre of mass distances, reciprocal molecular orientations) based on structural properties that need to be known *a priori*. The user can also choose to bias only very attractive pairs, *i.e.* all those with $E_{ij} < E_{bias}$(upper limit).

modivel needs information on the actual distribution of molecule–molecule interaction energies. These are computed only if the idistr flag in the third line of the MD control file (*.mdi*) is active (idistr = 1) and an upper threshold Emolim for the distribution is defined (see Section 7.6.2 for the full description of the available commands and options). For the same reason, E_{bias} parameters must be strictly lower (more negative) than Emolim, the maximum energy threshold for computing the distribution. MiCMoS checks the internal consistency of input parameters; if errors are found, the program stops with a warning message.

In practice, a bias scaling factor $g \leq 1$ is computed:

$$g = 1 - \frac{|E_{ij} - E_{bias}(upper)|}{|E_{ij}|} \text{ if } E_{ij} < E_{bias}(upper) \qquad (7.9a)$$

$$g = 1 \text{ if } E_{ij} \geq E_{bias}(upper) \text{ or } E_{ij} < E_{bias}(lower) \qquad (7.9b)$$

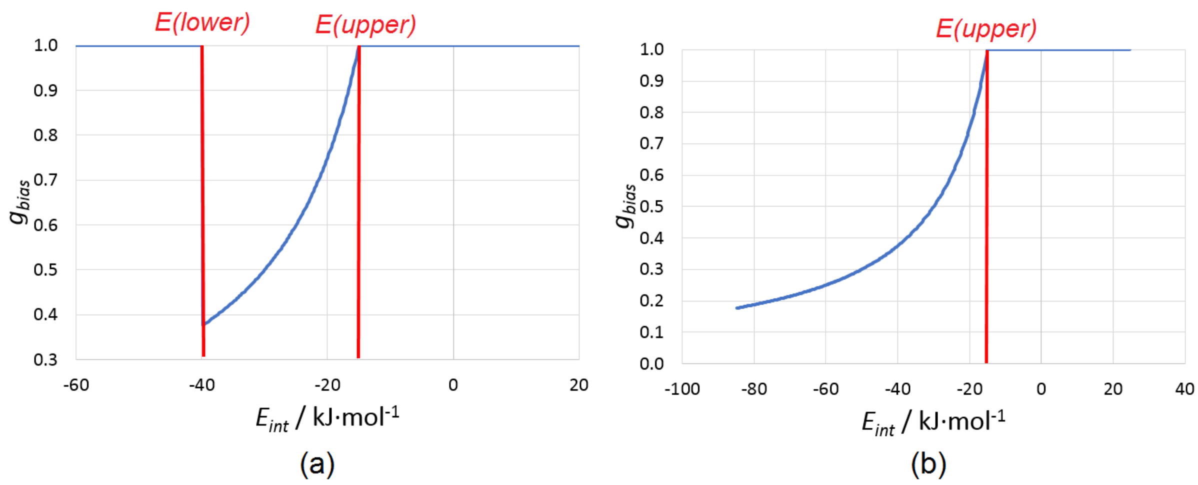


Figure 7.2. (a) Bias scaling function *vs.* molecule-molecule interaction energies E_{ij} for $E_{bias}(lower) < E_{ij} < E_{bias}(upper)$, with $E_{bias}(lower) = -40\ kJ/mol$. (b) Same as (a), without a lower limit. In both panels, $E_{bias}(upper) = -15\ kJ/mol$.

The functional g(E_{ij}) has the form shown in Figure 7.2: the larger the $|E_{ij} - E_{bias}(upper)|$ difference, the smaller is g; when $E_{ij} \ll E_{bias}$(upper), $g \to 0$, while $g \to 1$ when E_{ij} approaches E_{bias}(upper). Both the molecules i and j involved in the attractive pair will have their kinetic energies, T_i and T_j, scaled according to:

$$T_i^{new} = g \cdot T_i \tag{7.10a}$$
$$T_j^{new} = g \cdot T_j \tag{7.10b}$$

To the sake of simplicity, we will refer just to the ith single molecule hereinafter. In the algorithm, the same operations are obviously repeated also for the jth molecule of each pair.
The ith molecule undergoes a kinetic energy change of

$$\Delta T_i = T_i^{new} - T_i = g \cdot T_i - T_i = T_i \cdot (g - 1) \tag{7.11}$$

ΔT_i is negative, as $g < 1$ and T_i must be positive. When $g \to 0$, it turns out that the whole molecular kinetic energy is suppressed ($\Delta T_i = -T_i$). On the contrary, no changes are made ($\Delta T_i = 0$) in unbiased steps ($g = 1$). The reduction of the molecular kinetic energy is equally distributed across the N_i atoms in the molecule i. Thus, each atom loses an amount of kinetic energy $\Delta T_{i,a}$:

$$\Delta T_{i,a} = \frac{\Delta T_i}{N_i} = T_{i,a}^{new} - T_{i,a} < 0 \tag{7.12}$$

where $T_{i,a}$ is the contribution to the molecular kinetic energy due to the atom $a \in i$ with velocity module $v_{i,a}$:

$$T_{i,a} = \frac{1}{2} m_{i,a} v_{i,a}^2 \tag{7.13a}$$
$$T_i = \sum_a^{N_i} \frac{1}{2} m_{i,a} v_{i,a}^2 = \sum_a^{N_i} T_{i,a} \tag{7.13b}$$

A correspondent equality holds true for the biased kinetic energy of the molecule i:

$$T_i^{new} = \sum_a^{N_i} \frac{1}{2} m_{i,a} v_{i,a,new}^2 = \sum_a^{N_i} T_{i,a}^{new} = g \cdot T_i \tag{7.14}$$

Substituting (7.13*b*) into (7.14) one gets the equality (7.15):

$$\sum_a^{N_i} \frac{1}{2} m_{i,a} v_{i,a,new}^2 = g \cdot \sum_a^{N_i} \frac{1}{2} m_{i,a} v_{i,a}^2 = \sum_a^{N_i} \frac{1}{2} m_{i,a} g\left(v_{i,a}^2\right) \tag{7.15}$$

which is certainly true if

$$v_{i,a,new}^2 = g\left(v_{i,a}^2\right) \tag{7.16}$$

In other words, the bias scaling function g rescales the square modules of atomic velocities, whose individual components can be obtained by taking the corresponding square roots:

$$\begin{cases} \pm v_{i,a,x}^{new} = \pm\sqrt{g} \cdot v_{i,a,x} \\ \pm v_{i,a,y}^{new} = \pm\sqrt{g} \cdot v_{i,a,y} \\ \pm v_{i,a,z}^{new} = \pm\sqrt{g} \cdot v_{i,a,z} \end{cases} \tag{7.17}$$

To conserve the total kinetic energy of the ensemble, the reduction of kinetic energy ΔT_i must be compensated by a corresponding gain from all the molecules whose kinetic energies are not going to be reduced, that is, those N_k molecules that have $E_{kl} > E_{bias}$. The change ΔT_i is equally distributed across these N_k molecules as $\Delta T_k = -\Delta T_i / N_k > 0$. By expressing this energy change as a kinetic energy rescaling of each k^{th} molecule, one gets:

$$T_k^{new} = \alpha \cdot T_k = T_k + \Delta T_k \tag{7.18}$$

The factor α for the k^{th} non-frozen molecule can be computed from (7.18):

$$\alpha = 1 + \frac{\Delta T_k}{T_k} \tag{7.19}$$

As it should be, $\alpha > 1$. From $T_k^{new} = \alpha \cdot T_k$ (7.18) we can proceed analogously to (7.14)–(7.16) to find the scaling factor for the atomic velocities:

$$\begin{cases} \pm v_{k,a,x}^{new} = \pm\sqrt{\alpha} \cdot v_{k,a,x} \\ \pm v_{k,a,y}^{new} = \pm\sqrt{\alpha} \cdot v_{k,a,y} \\ \pm v_{k,a,z}^{new} = \pm\sqrt{\alpha} \cdot v_{k,a,z} \end{cases} \tag{7.20}$$

The rescaling procedure is applied again and again to the same molecule, one time for each intermolecular interaction that fulfills the biasing criterion. This means that molecules involved at the same time in more than one stable pair, that is, forming a cluster, are frozen more rapidly, increasing the probability of obtaining stable aggregates. Eqs. (7.17) and (7.20) provide a very simple recipe to rescale all the velocities in the simulation box in such a way that the total kinetic energy is conserved upon the application of (7.10) and (7.19). In practice, discretization errors and numerical approximations could make the total kinetic energy not fully conserved. The algorithm checks the kinetic energy before and after the application of the bias and stops with a warning message if an error larger than 1 % is detected. This is just a precaution, as the thermostat should take care of restoring periodically the correct kinetic energy of the ensemble.

In summary, the user can select the thresholds of the bias (parameters E_{bias}(upper) and E_{bias}(lower), see Section 7.6.2) and the frequency of its application (parameter Nbias, see Section 7.6.2). The algorithm selects the best scaling factor g depending on the strength of interaction between the involved molecules (Figure 7.2). It is possible to keep the bias active for the whole trajectory, or to repeatedly switch it on/off for specific user–defined time periods (parameters tinon and tinof, see Section 7.6.2).

7.2.5 Molecular dynamics in confined spaces

The model for confined space relies on neutral, rigid and stretchable barriers that are added on the boundaries of the simulation box.

To perform a confined molecular dynamics simulation, MiCMoS needs (i) a pre-equilibrated simulation box and (ii) a topology file as input. These can be obtained with standard procedures, which include for example routines like *boxliq.for* (Section 5.2) and *pretop.for* (Section 5.4), followed by a fast 1-2 Msteps long run of Monte Carlo (Section 6) to dispose of hard contacts and adjust the starting density.

The unconfined liquid is given in input to a new module, confbox.*for* (Section 5.8). This program prepares the parameter file barrier.*par* (Section 5.8.1), which specifies the geometrical details of the confined space and the force field parameters of the barrier. At the same time, confbox.*for* converts the standard simulation box into a new one, ready for the confined simulation, by deleting all the molecules which bear an atom in close contact (less than the sum of the van der Waals radii) with the barrier pixels.

The barriers consist of regular square/rectangular grids of massless pixels, neutral by deafult. The user is free to set the pixel diameter and the Force Field parameters that determine molecule-pixel interactions during the dynamics. In practice, at the user's convenience, the same atom id code is attached to all pixels according with Table 1.1 (Section1.4), which selects the corresponding A6, A12 Lennard–Jones parameters of the LJC parametrization (Section 2.1.2). It is wise to set pixel dimensions similar to the van der Waals diameter of the selected atom type, but smaller or larger pixels can be employed as well if desired, for example to set up simulations in a coarse-grained fashion.

Upon addition of the barriers, at least one direction out of X, Y or Z becomes non periodical. The user sets the desired equilibrium distance between pairs of opposite barriers, wihich may or not be different from the starting one. The program confbox.*for* takes care of tuning the length of the simulation edges to have the desired target packing efficiency. In other words, the volume of the confined box is variable and depends on the number of molecules and the desired packing efficiency (C_{pack}). By default, the program sets the box volume to have the theoretical maximum efficiency for the random packing of spheres, 0.66 (see Zaccone, Phys. Rev. Lett., **2022**, 128, 028002). If desired, the user can start with less dense liquids by acting on the parameters *nmolzacu* and *nmolzacv* specified in the barrier.*par* file (Sections 5.8 and 5.8.1). By increasing these parameters, a larger box will be produced, and C_{pack} will be consistently lowered. When dealing with nanotubes and nanolayers, the user sets the barrier-to-barrier distances they desire, and confbox.*for* tunes the length of the simulation box along the periodic directions. For nanocavities, no periodic directions exist, and the target equilibrium edge length are automatically computed according to the target C_{pack}. This procedure ensures that no steric clashes should be produced, as the program prevents the user from setting box dimensions that are too small with respect to the number of the molecules in the liquid.

From a geometrical viewpoint, confbox.*for* places the barriers onto the surfaces that bound the original simulation box. This implies that the barriers are always parallel in pairs, and only squared sections are available for nanotubes and nanocavities. An offset parameter can be controlled at the confbox.*for* stage, to tune the distance between the barriers and the box boundaries. The offset parameter allows to perform fine adjustments of the barrier positions with respect to the simulation box surface. Confbox.*for* deletes those molecules that have at least one atom below the van der Waals distance with at least one pixel of the barrier (see above). This step is mandatory to prevent steric clashes at the beginning of the simulation. Thus, the offset parameter is useful to avoid the deletion of many molecules.

The number of pixels that define the barriers is calculated by rounding down the ratio between the equilibrium length of the box edges, as computed by the program according with the desired C_{pack}, and the user-defined pixel diameter.

The confinement partitions the available space into a "bulk" region, where molecules are distant from the barrier and no significant pixel-molecule interactions are set up, and a "barrier" region, where on the contrary pixel-molecule interactions are not negligible (Figure 7.3). A reasonable empirical cutoff of 10^{-3} kJ·mol^{-1} is usually employed to discriminate between such regions, but the user is free to make their own choice to establish the corresponding boundaries.

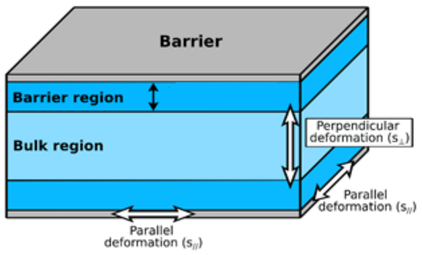

Barrier
Barrier region
Bulk region
Perpendicular deformation ($s_\perp$)
Parallel deformation ($s_{//}$)
Parallel deformation ($s_{//}$)

Figure 7.3. Structure of the confined space for a nanolayer. The barriers (in grey) can be stretched or compressed along direction parallel to their surface. At the same time, they are allowed to vibrate along the perpendicular direction. The amplitudes of these motions are controlled by a force constant (see text). Molecule-barrier interactions are nonzero ($E_B > 0.001$ kJ·mol^{-1}) in the barrier region (dark blue), while they are negligible in the bulk region (light blue).

The barriers are stretchable: every time the barostat varies the dimensions of the simulation box, the pixel positions are modified accordingly, but their number remains constant. If the pressure of the system is high and positive (negative) along a specific direction parallel to the barrier, the box dimension in that direction would increase (decrease). To avoid unphysical stretching, the following solution is employed. A fictitious pressure is added in each laboratory direction X, Y, and Z. This counterpressure is proportional to the difference between the actual box size and the equilibrium box size, divided by the area of the face orthogonal to that direction. By default, the force constant *k* corresponds to the stretching of an aromatic Csp^2-Csp^2 bond (3400 kJ·mol^{-1}·Å^{-2}). Damping scale factors can be also applied by the user at the confbox.*for* stage, which are also specified in the barrier.*par* parameter file. Thus, the user may control the stiffness of the system, along both the parallel and perpendicular directions with respect to the barrier(s). To this end, three damping factors are available (*dampk*(XY) along Z, *dampk*(XZ) along Y and *dampk*(YZ) along X, see also Section 5.8), to change the relative strength of the force constant along the X, Y and Z directions (Figure 7.3). Contradictory statements (for example, different parallel and perpendicular force constants in a nanocavity) are recognized by the program, which in that case stops before entering the simulation.

To start the dynamics, a new parameter, *inano*, must be specified in the input *.mdi* file (see Section 7.6.2 below). *inano* = 0 means unconfined simulations and implies that the confbox procedure described above is not needed, while *inano* = 1, 2, 3 may be set for nanolayer, nanotube, and nanocavity simulations respectively; in other words, the confinement is applied on the starting simulation box by reducing the corresponding number of periodic directions.

Then, the dynamics proceeds as usual; the only difference with respect to standard (unconfined) simulations is that atom-pixel interactions are evaluated at every steps. Every atom interacts with all the pixels lying within the usual cutoff distance. In particular, the solute-solute cutoff (see Section 7.6.2) is applied also for atom-pixel distances that involve the solute, while the same solvent-solvent cutoff is applied for atom-pixel distances that involve the solvent.

As the barrier has not a true chemical structure, it is only affected by stretching deformations, as prompted by the barostat. Accordingly, the atom-pixel energy terms are not explicitly considered to compute the total virial of the forces of the system, which still refers to the usual solute-solute, solute-solvent and solvent-solvent terms. As a consequence, the forces exerted onto the barriers by the liquid are totally ignored. This prevents the barrier from changing its shape, for example from being bent. On the contrary, the molecules fully feel the force exerted by the barriers, as their dynamics (in terms of translation, rotation and velocity) is influenced by the interactions with the pixels. In other words, the presence of the barriers has an indirect effect on the total virial and henceforth on the (an)isotropic pressure.

At the very beginning of the simulation, in particular if a large offset is set between the faces of the simulation box and the barriers, a high negative pressure could be developed in the direction perpendicular to the barriers, which might rapidly compress the system and produce clashes. To avoid such problems, when the confinement is active (*inano* > 0), a further damping is automatically applied to the barostat algorithm (Section 7.3). In particular, the barostat is prevented from applying (an)isotropic scaling factors lower than 0.95 to the box edge lengths. Equivalently, changes not greater than 5 % are allowed for the box edge lengths in a single MD step.

Finally, the user can employ either full Lennard-Jones barriers or repulsive-only ones. This options is controlled by the parameter *iattr*, which is given as input during the confbox.*for* procedure and is consequently set in the barrier.*par* file. If *iattr* is 0, the attractive A6 coefficient (Section 2.1.2) is switched off (A6 = 0) and the only possible atom-barrier interactions are the short-range repulsive ones, as governed by the A12 coefficient. This also implies that the "barrier" part of the simulation box (Figure 7.3) becomes thinner, while the "bulk" one enlarges. If the A6, A12 coefficients of Carbon are used, for example, the thickness of the "barrier" layer is ~11.6 Å for a full Lennard-Jones potential and shrinks to ~6.4 Å for a repulsive-only one.

7.3 Periodic boundary conditions and pressure control

Periodic boundary conditions can be imposed in three dimensions. To this end, the program automatically loads 27 replicas of the main box along the three directions. At preset intervals and/or after box changes, the center of mass motion is stopped (subroutine *comres*) and out of box molecules re-enter the box from the opposite side.

The pressure of the simulation can be controlled with two algorithms. When simulating a liquid with a cubic simulation box, pressure rescaling should be **isotropic** (Section 7.3.1), the box being constrained to stay cubic. In principle, isotropic rescaling could apply also to orthogonal crystal boxes as in orthorhombic or higher-symmetry systems, although this is not advisable. In general, for the simulation of a crystal, pressure rescaling should be **anisotropic** (Section 7.3.3), allowing full relaxation of different box dimensions and angles. This can be achieved by a "**minimal barostat**" option (Section 7.3.2), or by the **Parrinello–Rahman** algorithm (Section 7.3.3).

7.3.1 Isotropic pressure control

The isotropic pressure control on orthogonal computational boxes is achieved by use of the virial theorem. The algebra is from GROMOS96 manual: see Van Gunsteren, W. F., Billeter, S. R., Eising, A. A., Hunenberger, P. H., Kruger, P., Mark, A. E., Scott, W. R. P. & Tironi, I. G., *Biomolecular simulation: The GROMOS96 manual and user guide*, BIOMOS B.V. Zurich, Groningen, vdf Hochschulverlag A.G. and der ETH Zurich (1996).

Consider an ensemble of molecules each with total mass M_i , made of atoms of mass m_k and velocity $\mathbf{v}_k$. Let $\mathbf{V}_i$, be the centre of mass velocity:

$$\mathbf{V}_i = \sum_{k=1}^{atoms \in i} \frac{m_k \mathbf{v}_k}{M_i} \tag{7.21}$$

Thus, the molecular kinetic energy is $\frac{1}{2} M_i V_i^2$, and the total kinetic energy of the ensemble is:

$$E_{kin} = \sum_{i=1}^{molecules} \frac{1}{2} M_i V_i^2 \tag{7.22}$$

The *k*th atom of the *i*th molecule interacts with all the *l* atoms in each of the surrounding molecules, *j*. The corresponding forces can be computed with the established force field (Section 7.4). Atom–atom force contribution $\mathbf{f}_{kl}$ will sum up to give the total force acting on the centre of mass of the *i*th reference molecule due to the presence of the *j*th one:

$$\mathbf{F}_{ij} = \sum_{k,l}^{atoms \in i,j} \mathbf{f}_{kl} \tag{7.23}$$

If we now define as $\mathbf{R}_{ij}$ the *i*–*j* centre of mass vector distance, the centre of mass virial *W* can be computed as:

$$W = -\frac{1}{2} \sum_{i<j}^{molecules} \mathbf{F}_{ij} \cdot \mathbf{R}_{ij} \tag{7.24}$$

Eventually, the pressure experienced by the whole molecular ensemble can be estimated from the classical virial theorem, *V* being the volume of the computational box:

$$P = \frac{2}{3V}(E_{kin} - W) \tag{7.25}$$

Table 7.1 above shows the measure units that are employed throughout. At preset time intervals, virial (*W*) and kinetic energy (E_{kin}) are calculated and current pressure *P* is obtained by (7.25); then, compliance towards the set pressure, P_{set} (usually 1 atm = 101300 Pa) is achieved using the relationships (7.14)–(7.18).

First, an adimensional shift factor μ is computed from the corresponding compressibility coefficient, μ_0, which is given in the run control file *.mdi* (see Section 7.6.2 below; this being an empirical factor whose values is usually 0.3 or 0.4):

$$\mu = \sqrt[3]{[1 - \mu_0(P_{set} - P)]} \tag{7.26}$$

Then, the new box dimensions are computed and centre–of–mass vector ($\mathbf{r}_{\text{com}}$) components accordingly updated:

$$a' = \mu a \tag{7.27}$$

$$\mathbf{r}'_{\text{com}}(i) = \frac{a'}{a}\mathbf{r}_{\text{com}}(i) \tag{7.28}$$

For each molecule, the centre–of–mass displacement is applied to all the atomic coordinates $\mathbf{x}_k$ of the *i*th molecule:

$$\Delta\mathbf{x} = \mathbf{r}'_{\text{com}}(i) - \mathbf{r}_{\text{com}}(i) \tag{7.29}$$

$$\mathbf{x}'_k = \mathbf{x}_k + \Delta\mathbf{x} \tag{7.30}$$

7.3.2 Anisotropic pressure control for oblique boxes

For monoclinic or triclinic crystals, in the **minimal barostat option** the anisotropic pressure control proceeds with the 3-dimensional equivalent of the algorithm detailed in equations (7.9)–(7.14): the separate x, y, z components of virial, kinetic energy and forces are computed to obtain three separate box update parameters, μ_x, μ_y and μ_z, which rescale each of the three components of box vectors. Moreover, the whole vector algebra in an oblique system must go through an orthogonalization and deorthogonalization procedure as detailed below.

CAUTION: With this procedure, no explicit equations of motions are solved for the simulation box. Thus, if the starting cell has angles equal to 90°, its shape will remain fixed, as no off–diagonal elements are present in the transformation matrix, which could mix edge vectors and affect angles.

Let a, b, c, α, β, γ be the unit cell parameters, repeated by N_a, N_b, N_c times along the three dimension to form a computational box with cell parameters A, B, C and α , β, γ, where $A = a{\cdot}N_a$, $B = b{\cdot}N_b$ and $C = c{\cdot}N_c$. Let cosα, cosβ, cosγ, sinα, sinβ, sinγ be the cosines and sines of the three box angles. From these values, an orthogonalization matrix **O** can be computed according to:

$$\mathbf{O} = \begin{pmatrix} A & B\cos\gamma & C\cos\beta \\ 0 & B\sin\gamma & C\left[\dfrac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}\right] \\ 0 & 0 & C\left[\dfrac{f_V}{\sin\gamma}\right] \end{pmatrix} \tag{7.31}$$

Where the scalar f_V is

$$f_V = (1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma)^{1/2} \tag{7.32}$$

Any column vector of fractional coordinates x_f, y_f, z_f can thus be orthogonalized to give the corresponding x_o, y_o, z_o coordinates in a Cartesian reference frame following the transformation $\mathbf{x_o} = \mathbf{O}\,\mathbf{x_f}$:

$$\left.\begin{aligned} x_0 &= Ax_f + B\cos\gamma \cdot y_f + C\cos\beta \cdot z_f \\ y_0 &= B\sin\gamma \cdot y_f + C\left[\frac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}\right]\cdot z_f \\ z_0 &= C\left[\frac{f_V}{\sin\gamma}\right]\cdot z_f \end{aligned}\right\} \tag{7.33}$$

Where f_V is defined in (7.32) and the volume of the simulation box is

$$V_{box} = (ABC)f_V \tag{7.34}$$

For crystals, starting orthogonalized atomic coordinates in files *boxcry.dat* are generated by program ***Boxcry*** (Section 5.1) using (7.31)–(7.34) from crystallographic data. In liquid boxes, $A = B = C$ with three angles 90° and all coordinates are directly generated orthogonal by the ***Boxliq*** module (Section 5.2).

The edge vectors of the computational $\mathbf{B_x}$, $\mathbf{B_y}$, $\mathbf{B_z}$ are also computed in the new orthogonal reference frame by applying the same transformation **O** to the fractional unit translations corresponding to the crystallographic computational box: (1 0 0), (0 1 0), (0 0 1). This leads to the following expressions for the components of **B**$_x$, **B**$_y$, **B**$_z$:

$$\mathbf{B_x} = \begin{pmatrix} A \\ 0 \\ 0 \end{pmatrix}; \mathbf{B_y} = \begin{pmatrix} B\cos\gamma \\ B\sin\gamma \\ 0 \end{pmatrix}; \mathbf{B_z} = \begin{pmatrix} C\cos\beta \\ B\left[\frac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}\right] \\ C\left[\frac{f_V}{\sin\gamma}\right] \end{pmatrix} \tag{7.35}$$

To implement box periodicity, 27 vectors **n** of components n_1, n_2, n_3 = -1, 0 or +1 in all combinations are generated and then the actual translation vectors **T** are:

$$\left.\begin{aligned} \mathbf{T}(1) &= n_1\mathbf{B_x}(1) + n_2\mathbf{B}_y(1) + n_3\mathbf{B_z}(1) \\ \mathbf{T}(2) &= n_1\mathbf{B_x}(2) + n_2\mathbf{B}_y(2) + n_3\mathbf{B_z}(2) \\ \mathbf{T}(3) &= n_1\mathbf{B_x}(3) + n_2\mathbf{B}_y(3) + n_3\mathbf{B_z}(3) \end{aligned}\right\} \tag{7.36}$$

All coordinates in the computational box are periodically repeated by summing the possible vectors **T**, in all their 27 combinations. The intermolecular energy is computed between all molecules in the computational box, and all translated molecules within a cutoff distance between centers of mass (usually, 15-18 Å). In this way, summations are always carried out between full molecules (neutral units), thus reducing if not disposing of truncation and edge effects. Vectors **T** are established at the beginning of the calculation in subroutine *rebox*, and are updated as soon as vectors **B**$_x$, **B**$_y$ and **B**$_z$ are updated during pressure control.

In the "minimal barostat" option, the new pressure components *P*(*i*) (former (7.25)) and the corresponding shift factors μ(*i*) (former (7.26)) are:

$$\left.\begin{aligned} P_x &= \frac{2}{3V}\left(E_{kin,x} - W_{xx}\right);\ \mu_x = \sqrt[3]{[1 - \mu_0(P_{set} - P_x)]} \\ P_y &= \frac{2}{3V}\left(E_{kin,y} - W_{yy}\right);\ \mu_y = \sqrt[3]{\left[1 - \mu_0\left(P_{set} - P_y\right)\right]} \\ P_z &= \frac{2}{3V}\left(E_{kin,z} - W_{zz}\right);\ \mu_z = \sqrt[3]{[1 - \mu_0(P_{set} - P_z)]} \end{aligned}\right\} \tag{7.37}$$

Where μ_0 is the compressibility coupling parameter detailed in Section 7.3.1. Translation vectors are updated accordingly:

$$\begin{bmatrix} \mathbf{B'_x}(1) = \mu_x\mathbf{B_x}(1) \\ \mathbf{B'_x}(2) = \mu_y\mathbf{B_x}(2) \\ \mathbf{B'_x}(3) = \mu_z\mathbf{B_x}(3) \end{bmatrix}; \begin{bmatrix} \mathbf{B'_y}(1) = \mu_x\mathbf{B_y}(1) \\ \mathbf{B'_y}(2) = \mu_y\mathbf{B_y}(2) \\ \mathbf{B'_y}(3) = \mu_z\mathbf{B_y}(3) \end{bmatrix}; \begin{bmatrix} \mathbf{B'_z}(1) = \mu_x\mathbf{B_z}(1) \\ \mathbf{B'_z}(2) = \mu_y\mathbf{B_z}(2) \\ \mathbf{B'_z}(3) = \mu_z\mathbf{B_z}(3) \end{bmatrix} \tag{7.38}$$

after which also the translation vectors **T** are updated (eq. (7.36)). Once the new vector components are available, the box parameters in the new crystallographic reference frame are recalculated as:

$$\left.\begin{aligned} A_{new} &= \sqrt{\mathbf{B}'_{\mathbf{x}}(1)^2 + \mathbf{B}'_{\mathbf{x}}(2)^2 + \mathbf{B}'_{\mathbf{x}}(3)^2} \\ B_{new} &= \sqrt{\mathbf{B}'_{\mathbf{y}}(1)^2 + \mathbf{B}'_{\mathbf{y}}(2)^2 + \mathbf{B}'_{\mathbf{y}}(3)^2} \\ C_{new} &= \sqrt{\mathbf{B}'_{\mathbf{z}}(1)^2 + \mathbf{B}'_{\mathbf{z}}(2)^2 + \mathbf{B}'_{\mathbf{z}}(3)^2} \end{aligned}\right\} \tag{7.39}$$

And accordingly, the new angles are given by:

$$\left.\begin{aligned} \alpha_{new} &= \cos^{-1}\left[\frac{\mathbf{B}'_{\mathbf{y}} \cdot \mathbf{B}'_{z}}{B_{new}C_{new}}\right] \\ \beta_{new} &= \cos^{-1}\left[\frac{\mathbf{B}'_{\mathbf{x}} \cdot \mathbf{B}'_{z}}{A_{new}C_{new}}\right] \\ \gamma_{new} &= \cos^{-1}\left[\frac{\mathbf{B}'_{\mathbf{y}} \cdot \mathbf{B}'_{x}}{B_{new}A_{new}}\right] \end{aligned}\right\} \tag{7.40}$$

All orthogonal center of mass coordinates x_o, y_o, z_o are updated as follows (subroutine *boxexp*). First, old coordinates are de-orthogonalized with the old metrics A_{old}, B_{old}, C_{old} and old α_{old}, β_{old}, γ_{old} with the inverse matrix $\mathbf{O}^{-1}$ (inverse of (7.31), see Appendix Section A3): new c.o.m. coordinates are then generated by re-orthogonalizing with the new metrics:

$$\left.\begin{aligned} \mathbf{r}(\text{old}, \text{fract}) &= \mathbf{O}_{\text{old}}^{-1} \cdot \mathbf{r}(\text{old}, \text{orthog}) \\ \mathbf{r}(\text{new}, \text{orthog}) &= \mathbf{O}_{\text{new}} \cdot \mathbf{r}(\text{old}, \text{fract}) \end{aligned}\right\} \tag{7.41}$$

New c.o.m. coordinates are then generated by re-orthogonalizing $[r_x, r_y, r_z]$ with the new metrics (matrix $\mathbf{O}$, eq. (7.31)), such that $\mathbf{r}'_{\text{com}} = \mathbf{O} \cdot \mathbf{r}_{\text{com}}$. Finally, the displacements are calculated as $\Delta\mathbf{x} = \mathbf{r}'_{\text{com}} - \mathbf{r}_{\text{com}}$ (as in eq. (7.29)) and the coordinates of all atoms are displaced by $\Delta\mathbf{x}$ (as in eq. (7.30)). This procedure preserves rigid molecular conformations, as it shifts molecules as rigid bodies. Velocities are left unchanged after box update (an acceptable approximation).

7.3.3 Parrinello-Rahman barostat

The original procedure is described in Parrinello, M. & Rahman, J. Appl. Phys. 1981, 52, 7182–7190; Phys. Rev. Letters 1980, 45,1196-1199. The code for the P-R barostat is embodied in in the *mdviri* library of the MD setup. All quantities are expressed in S.I. units, with energy in kJ/mol, velocity in m/s, distance in meters and forces in Newton.

1) An array *vl*(4000,3) is defined, where the velocities of the centers of mass (c.o.m.) of solute (from 1 to NMSOLU, see Table 5.1) and solvent (from NMSOLU+1 to NMSOLV, see Table 5.1) are progressively stored. At the same time, the halved negative virial tensor components due to solute-solute, solvent-solvent and solute-solvent interactions are computed according to:

$$\mathbf{W} = -{}^{1}\!/_{2}\, \mathbf{f}_{ij} \otimes \mathbf{R}_{ij} \tag{7.42}$$

Eventually, the tensor contributions coming from each molecular pair are summed up, and the total virial tensor is obtained. $\mathbf{f}_{ij}$ is the force between molecules *i* and *j* and $\mathbf{R}_{ij}$ is the corresponding c.o.m. distance (see also Table 7.1). The symbol ⊗ denotes the tensor product between the two vector (1st–rank tensor) quantities.

CAUTION: For the moment, a maximum number of 4000 molecules can be dealt with, including solute and solvent. If your box contains more molecules, the array dimensions must be incremented directly in the source code, and the program recompiled.

2) The total (3x3) kinetic energy tensor is computed according to:

$$\mathbf{E} = \sum_i \frac{1}{2} m_i \left(\mathbf{v}_i \otimes \mathbf{v}_i\right) \tag{7.43}$$

$\mathbf{v}_i$ being the *i*-th vector stored in the *vl*(4000,3) array. The summation goes up to the total number of molecules, NMSOLU+NMSOLV.

3) Eventually, the individual components of kinetic energy and virial are summed each other, and the pressure tensor is computed as

$$\mathbf{P} = \frac{2}{V}(\mathbf{E} + \mathbf{W}) \tag{7.44}$$

where *V* is the volume of the simulation box and takes care of converting energy units into pressure units. Overall, this procedure implements the definition of the pressure tensor proposed by Parrinello & Rahman:

$$\mathbf{P} = \frac{2}{V}\left\{\sum_i \left[\frac{1}{2} m_i \left(\mathbf{v}_i \otimes \mathbf{v}_i\right)\right] - \frac{1}{2}\sum_{i<j}\left[\mathbf{f}_{ij} \otimes \mathbf{R}_{ij}\right]\right\} \tag{7.45}$$

4) The scalar hydrostatic pressure, *p*, is defined, as usual, as one third of the trace of the **P** tensor:

$$p = \frac{1}{3}\sum_{i=1}^{3} P_{ii} \tag{7.46}$$

5) Subroutines *metric* and *boxconv* take care of computing **H**, that is, the matrix of the box edges arranged as column vectors in a crystallophysical cartesian reference frame. Orthogonalization is carried out with the same procedure detailed in Section 7.3.2.

6) The box edges experience their own equation of motion, which can be written as:

$$\mathbf{F} = w\frac{d^2\mathbf{H}}{dt^2} = [\mathbf{P} - p\mathbf{I}]\boldsymbol{\sigma} \tag{7.47}$$

Here, **F** is the force acting on the simulation box, **H** the matrix of the box edges in cartesian coordinates, **P** the pressure tensor, *p* the scalar hydrostatic pressure, **I** the identity matrix and $\boldsymbol{\sigma}$ the volume-scaled reciprocal cell matrix. The coupling parameter *w* determines the strength of the pressure coupling and must be given in input within the *.mdi* parameter file (see Section 7.6.2). It has dimensions of a mass (kg) and determines the inertial response of the lattice to the pressure unbalance.

CAUTION: Strictly speaking, *w* should be chosen so that the relaxation time be of the same order of magnitude as *L*/*c*, *L* being the length of the simulation box and *c* the velocity of sound in the bulk. In fact, *w* determines the relaxation time for recovery from an imbalance between external pressure and internal stress. In practice, equilibrium properties are independent on the fictitious mass and *w* can be arbitrarily chosen to have a convenient relaxation time. Something like *w*=1.0-3.0 is normally appropriate.

7) The standard leap–frog algorithm is exploited. First, the velocity matrix associated to cell edge displacements ($\mathbf{V_H}$) is computed according to:

$$\mathbf{V_H} = \frac{1}{2}\left(\frac{n \cdot \Delta t}{w}\right)\mathbf{F} \tag{7.48}$$

where *n* is the number of MD steps between two subsequent pressure controls, Δ*t* the simulation time step (as defined in the input *.mdi* file, Section 7.6.2) and *w* the coupling parameter above discussed. As usual in the leap–frog procedure, velocities are evaluated at *n*·Δ*t*/2 seconds. Then, the **H** matrix of cell edges is finally updated from the velocity tensor at Δ*t*/2:

$$\mathbf{H}(t + \Delta t) = \mathbf{H}(t) + (n \cdot \Delta t)\mathbf{V_H} \tag{7.49}$$

Note that the velocity verlet algorithnm cannot be applied to the dynamics of the cell edges.

8) The c.o.m. positions of all the molecules in the simulation, including the 26 translationally–dependent images of the simulation box, are updated according to the change in the cell edge vectors, analogously to what detailed in Section 7.3.2.

7.3.4 External pressure

NpT MD simulations can be also carried out under arbitrary external stress. The original algebra was developed by Parrinello & Rahman (J. Appl. Phys. 1981, 52, 7182–7190) and relies on Lagrangian dynamics applied to the 3x3 cell edge tensor, **H**, that is, the array of three column vectors expressing cell edges Cartesian coordinates in the crystallophysical reference frame $\{\hat{\boldsymbol{e}}_{1,2,3}\}$ ($\hat{\boldsymbol{e}}_i \cdot \hat{\boldsymbol{e}}_j = \delta_{ij}$; $|\hat{\boldsymbol{e}}_{i,j,k}| = 1$). We associate the ordinary X, Y and Z labels to the directions expressed by the $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$, and $\hat{\boldsymbol{e}}_3$ versors. The external stress field can be applied along any combination of the X, Y, Z directions. We adopt a standard transform, pivoting on the crystallographic **a** cell axis, to define the laboratory Cartesian reference system, according to the orthogonalization procedure reported in Section A3 of this Manual. In this frame, X, Y and Z are exactly oriented as the cell edges **a**, **b** and **c** if the crystallographic system has angles $\alpha = \beta = \gamma = 90°$. If this is not the case, Y and Z may be not perfectly parallel to the crystallographic vectors **b** and **c**. For the moment, different choices of the laboratory axes are not allowed; we plan to include a routine to allow the user to rotate the laboratory reference frame as he/she wish in a forthcoming release of the package

According to classical elasticity theory of solids (Landau & Lifshitz, *Theory of Elasticity*, Pergamon, Oxford, 1959), when an external stress field is applied, an excess elastic energy is transferred to the lattice. To compute this term, a reference structure must be defined. Following Parrinello & Rahman, the cell corresponding to the first simulation frame (t = 0, $\mathbf{H_0}$) is chosen as the least biased and most practical choice to this purpose. If the simulation under stress starts from a previously equilibrated structure, and ends with a fully equilibrated structure as well, the elastic energy (V_{ela}) has the usual thermodynamic connotation and corresponds to a meaningful correction – within the limits of the Force

Field – to the steric energy. Eventually, the generalized enthalpy of a (H_{stress}, N) ensemble with cohesive energy E and compliant with the Lagrangian under stress is:

$$H_{stress} = E + V_{ela} \tag{7.50}$$

The V_{ela} component can be computed from the volume of the reference (starting) cell, V_0, as:

$$V_{ela} = p(V - V_0) + V_0 \cdot \mathrm{Tr}[(\mathbf{S} - p\mathbf{I}) \cdot \boldsymbol{\varepsilon}] \tag{7.51}$$

Where **S** is a generalized 2nd–order symmetric tensor that is given in input by the user in units of GPa, "Tr" means "trace" and **ε** is the symmetrical strain tensor. The latter is evaluated by comparing actual and reference (t = 0) cell parameters in Cartesian coordinates, as given by orthogonalized matrices **H** and $\mathbf{H_0}$:

$$\varepsilon_{ii} = \frac{[\mathrm{H}_{ii}(t) - \mathrm{H}_{ii}(0)]}{\mathrm{H}_{ii}(0)} \tag{752a}$$

$$\varepsilon_{ij} = \frac{1}{2}\left[\frac{\mathrm{H}_{ij}(t)}{\mathrm{H}_{jj}(0)} + \frac{\mathrm{H}_{ji}(t)}{\mathrm{H}_{ii}(0)}\right] \tag{7.52b}$$

The symmetry properties of the **S** matrix mirror the symmetry of the external stress field in the reference crystallophysical frame. For example, an external hydrostatic isotropic compression corresponds to S_{11}=S_{22}=S_{33} and S_{ij} = 0 ∀ i,j with magnitude equal to the trace of **S** (Tr[**S**]), while any difference among the diagonal elements reflects into an anisotropic tensile stress along some laboratory direction(s) X, Y or Z. Nonzero off-diagonal elements ($S_{ij} \neq 0$) correspond to applied shear stresses. According to the Cauchy notation, the index i identifies the shear plane, taken as the one normal to the corresponding i-th axis, while j specifies the laboratory axis along which the stress is applied.

A symmetric tensor **Σ** is defined, which converts the applied stress, **S**, into a surface energy density (force/m or energy/m^2) by accounting for the shape of the unit cell.

$$\boldsymbol{\Sigma} = \left[\mathbf{H_0^{-1}}(\mathbf{S} - p\mathbf{I})\widetilde{\mathbf{H}}\mathbf{_0^{-1}}\right] \cdot V_0 \tag{7.53}$$

$\mathbf{H_0}$ here represent the starting (undistorted) simulation box tensor and the "–1" and "~" symbols have the usual meaning of "inverse" and "transpose" matrix operations. Once the actual metric tensor (**G**) is known, **Σ** modifies the Lagrangian according to:

$$\mathcal{L}_{stress} = \mathcal{L} - \frac{1}{2}\mathrm{Tr}[\boldsymbol{\Sigma} \cdot \mathbf{G}] \tag{7.54}$$

Starting from equation 7.35, showed that this Lagrangian corresponds to the following equation of motion, which applies to the cell edge vectors and, indirectly, to all atom coordinates:

$$\mathbf{F} = w\frac{d^2\mathbf{H}}{dt^2} = [\mathbf{P} - p\mathbf{I}]\boldsymbol{\sigma} - \mathbf{H} \cdot \boldsymbol{\Sigma} \tag{7.55}$$

In practice, Equation (7.55) is made compliant with standard leap–frog integrator used in MiCMoS through Equations (7.48) and (7.49). Accordingly, all the molecules in the simulation box are them rigidly translated to account for the cell edges displacement, $\Delta\mathbf{H} = \mathbf{H}(t + \Delta t) - \mathbf{H}(t)$.

In this framework, perfect hydrostatic conditions are simulated by setting $S_{11} = S_{22} = S_{33}$ and $S_{12} = S_{13} = S_{23} = 0.0$. Any unbalance in the stress components can be can be defined accordingly; for example, $S_{11} = S_{22} = 1.0133 \cdot 10^{-4}$ GPa (=1 atm) and $S_{33} = 1.0$ GPa means an excess stress field, 1 GPa large, along the Z laboratory axis (see above).

CAUTION. Following Parrinello & Rahman (J. Appl. Phys. 1981, 52, 7182–7190), the target pressure used by normal anisotropic scaling without external stress field (entry Pset in Section 7.6.2, line #9) should be set to 0 when $\mathbf{S} \neq \mathbf{0}$. At the same time, larger values of the coupling parameter *w* are advised to avoid too large displacements, especially in the first steps of the simulation, when the system may be very far from equilibrium. Experience showed that *w* = 8–10 kg could be a reasonable choice.

7.4 Force fields

7.4.1 Intramolecular force field

The intramolecular force field includes bond stretching, bond bending, torsional and non-bonded terms:

$$E(stretch) = \frac{1}{2} k_{str} (R - R^0)^2 \tag{7.56}$$

$$E(bend) = \frac{1}{2} k_{bend} (\cos\theta - \cos\theta^0)^2 \tag{7.57}$$

$$E(tors) = k_{tors} [1 + f \cos m\varphi] \tag{7.58}$$

where R is a bond distance, θ is a bond angle, φ is a torsion angle, and the k's are parametric force constants. In Equation (7.58) f is a phase factor equal to ±1, and m is an integer equal to 1, 2 or 3 (see Appendix, Figure A7.1 in Section A7.3). If required by the user, specific atom–atom nonbonded terms can be also added, typically to avoid clashes when rotatable groups become very close to each other, or to other groups attached to the main molecular backbone. The calculation of these intramolecular nonbonding energies proceeds as in the MC setup, *i.e.* by the same potentials used in intermolecular interaction, damped by a factor FACTIN (see Section 6.4.2) that must be given in the *.mdi* run control file (Section 7.6.2). See Section 7.6.4 (instructions NLISTU and NLISTV) to see how to define nonbonded contacts I practice.

In MC stretching and bending potentials are seldom applied; in MD they are instead vital for the conservation of molecular shape, because MD acts on separate atomic coordinates rather than on global fragments. However, the precise values of the force constants, once a reasonable order of magnitude is supplied, are not crucial. Usually, one stretching potential is applied to each chemical bond in the molecule, one bending potential to each bond triad, and one torsion potential along each chemical bond. Equation (7.58) is also used with a very large force constant for improper dihedrals to keep sp^2 centers planar (Appendix, Section A7.3). The formulation and the algebra for the calculation of forces over potentials in Equations (7.56-58) can be found in the GROMOS96 manual cited at the beginning of Section 7.3.1. Sections A7 and A8 in the Appendix describe (i) how tentative torsional parameters are assigned by ***Pretop*** (Section 5.4); (ii) how torsion angles are actually computed, and (iii) provide values afir k_{tors}, f and m that are best suited to model specific functional groups (Table A7.5 in the Appendix, Section A7.3).

7.4.1.1 Anharmonic correction to the stretching potential

Since the program version 2.0, it is possible to explicitly introduce an anharmonic correction to the stretching motion (see Section 7.6.2 for input instructions).

CAUTION. The anharmonic correction is applied to all the covalent bonds in the solute molecule, including the X–H ones; in other words, solvent is not affected, and different bond types cannot be differentiated based on their harmonic or anharmonic behavior. The reason is to avoid possible unbalance effects in the force components acting on atoms. This option was introduced to work with pure substances, especially one-component crystalline solids. Thus, pay attention if you switch on the anharmonicity of the solute in a two-component system (e.g. a solution).

If required by the user (see the description of parameter ianh in Section 7.6.2), MiCMoS modifies the harmonic function (7.56) with a cubic term that makes the potential no more symmetric around the equilibrium bond distance R^0 (Figure 7.4). Specifically, the potential is made steeper at short distances and softer at longer ones, mimicking the Morse potential for small deviations from R^0.

$$E(stretch) = \frac{1}{2}k_{str}(R - R^0)^2 - \alpha k_{str}(R - R^0)^3 \tag{7.59}$$

α is a multiplier scaling factor set in the program source code (see below). At higher deformations, the cubic correction implies that the potential has an unphysical maximum at

$$R^{MAX} = \frac{1}{2}\left[\left(2R^0 + \frac{1}{3\alpha}\right) + \sqrt{\left(2R^0 + \frac{1}{3\alpha}\right)^2 - 4\left((R^0)^2 + \frac{R^0}{3\alpha}\right)}\right] \tag{7.60a}$$

$$E^{MAX}(stretch) = \frac{1}{2}k_{str}(R^{MAX} - R^0)^2 - \alpha k_{str}(R^{MAX} - R^0)^3 \tag{7.60b}$$

To avoid singularities, MiCMoS sets $E(stretch) = E^{MAX}(stretch)$ whenever $R > R^{MAX}$, that is, it makes the potential constant for very large deformations. This corresponds to the bond dissociation plateau, where the contribution to the stretching force is exactly 0 for both the atoms involved. Overall, the cubic correction is intermediate between the pure harmonic potential and the more accurate Morse form (Figure 7.4).

CAUTION. This very simple model cannot catch the correct physics when a bond is close to its dissociation limit; thus, any prediction that would imply very large deformations of covalent bonds should be interpreted with care.

The anharmonic constant is estimated empirically by multiplying k_{str} by an adimensional scaling constant α, which is now set at 0.48 in the source code. This empirical value was chosen as it guarantees that the bond dissociation energies in organic compounds are reasonably close to the experimental ones (see Table 4.11 in Dean, J. A. *Lange's Handbook of Chemistry*, 15th Ed. McGraw Hill (1999), ISBN 0-07-016384-7).

The force resulting from this correction has the following form:

$$F_x(stretch) = [-k_{str}(|\mathbf{r_1} - \mathbf{r_2}| - R^0) + 3\alpha k_{str}(|\mathbf{r_1} - \mathbf{r_2}| - R^0)^2]\frac{x_1 - x_2}{|\mathbf{r_1} - \mathbf{r_2}|} \qquad (7.61a)$$

$$F_y(stretch) = [-k_{str}(|\mathbf{r_1} - \mathbf{r_2}| - R^0) + 3\alpha k_{str}(|\mathbf{r_1} - \mathbf{r_2}| - R^0)^2]\frac{y_1 - y_2}{|\mathbf{r_1} - \mathbf{r_2}|} \qquad (7.61b)$$

$$F_z(stretch) = [-k_{str}(|\mathbf{r_1} - \mathbf{r_2}| - R^0) + 3\alpha k_{str}(|\mathbf{r_1} - \mathbf{r_2}| - R^0)^2]\frac{z_1 - z_2}{|\mathbf{r_1} - \mathbf{r_2}|} \qquad (7.61c)$$

Where $\mathbf{r}_1$ and $\mathbf{r}_2$ are the vector positions of bonded atoms 1 and 2 in the crystallophysical reference frame and *x*, *y* and *z* the corrisponding coordinates.

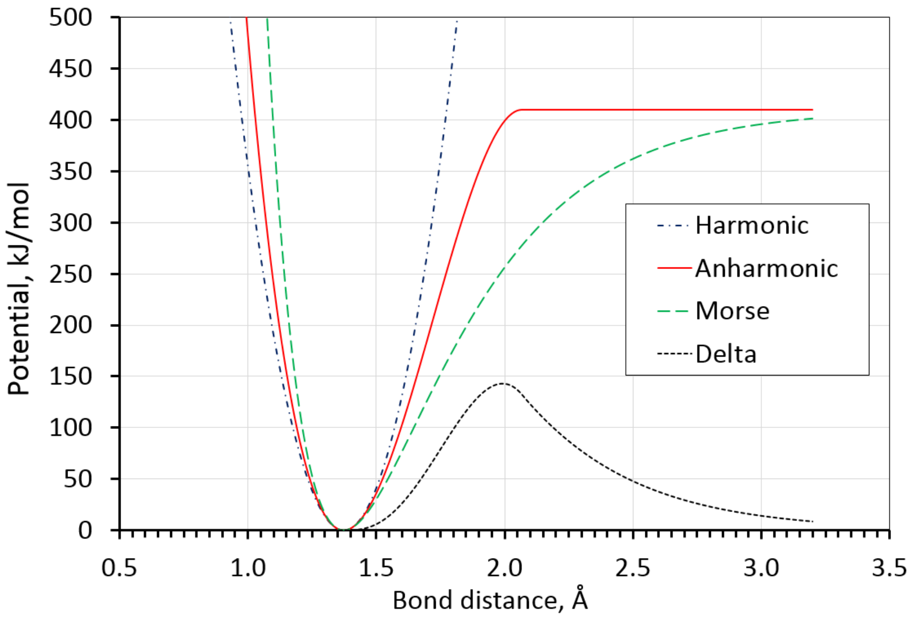


Figure 7.4. Comparison of the harmonic stretching potential (equation (7.56), blue line), the anharmonic–corrected cubic form (equation (7.59), full red line) and the Morse potential (dashed green line) for a covalent bond with R^0 = 1.373 Å, k_{str} = 5100 kJ·mol^{-1}·Å^{-2} and dissociation energy of 410 kJ/mol. The dotted black line shows the point–by–point difference for a positive stretching between the anharmonic–corrected form and the Morse one. The error is very small for small displacements from R^0 but increases rapidly at large displacements and is maximum at R^{MAX}.

CAUTION. It is advisable to work with smaller time steps when the anharmonicity is introduced in the model.

7.4.1.2 Anharmonic correction to the bending potential

Expression (7.57) shows the cosine potential that is employed to model bending deformations in MiCMoS. Note that it is not purely harmonic already, even though it closely resembles the parabolic potential for small deformations, especially when the angle closes (Figure 7.5).

Since the release v2.1, MiCMoS can handle a further anharmonic correction to bond angle bending. The philosophy is the same illustrated for the anhramonic correction to the stretching potential (Section 7.4.1.1). In the following, the bending interaction is defined by an atom triple *i*-*j*-*k*, *j* being covalently bonded to both *i* and *j*, and θ being the angle *ijk* (Figure 7.5).

CAUTION. The anharmonic correction to bending is applied to all the covalent angles in the solute molecule, that is, those defined by atom triples *i-j-k*; in other words, solvent is not affected, and different angle types cannot be differentiated based on their harmonic or anharmonic behavior. The reason is to avoid possible unbalance effects in the force components acting on atoms.

Starting from (7.57), we expand the bending potential in power series up to the 4th order:

$$E(bend) = \frac{1}{2}k_{bend}(\cos\theta - \cos\theta^0)^2 + \frac{1}{3!}k_{bend}(\cos\theta - \cos\theta^0)^3 + \frac{1}{4!}k_{bend}(\cos\theta - \cos\theta^0)^4 \quad (7.62)$$

The advantage of this very simple model is that the correction to force components due to the 3rd and 4th terms are additive. The factorial coefficients serve just as damping factors for high-order contributions; no explicit definitions of the anharmonic bending constants are thus required. A bit of algebra shows that the 3rd-order correction to force component *x* on atom *i* in the crystallophysical reference frame read

$$F_{i,x}(bend,III) = -\frac{1}{2}\, k_{bend}(cos\theta - cos\theta^o)^2 \left[\frac{x_k - x_j}{r_{ij}\cdot r_{jk}} - \frac{x_i - x_j}{r_{ij}^2}cos\theta\right] \quad (7.63)$$

Where r_{ij} and r_{jk} are the corresponding distances between *i-j* and *j-k*. Analogue expressions can be written for $F_{i,y}$ and $F_{i,z}$ by substituting the *x* coordinate with either *y* or *z*.

The 3rd-order correction to the force acting on atom *k* is:

$$F_{k,x}(bend,III) = -\frac{1}{2}\, k_{bend}(cos\theta - cos\theta^o)^2 \left[\frac{x_i - x_j}{r_{ij}\cdot r_{jk}} - \frac{x_k - x_j}{r_{jk}^2}cos\theta\right] \quad (7.64)$$

And that acting on *j* is given by the sum of these contributions, so that the total sum of bending forces on the atom triple *i-j-k* is zero. This is a necessary requirement, as bending concerns the relative motion of atoms in the triple and must cause neither a dispacement nor a rotation of the whole *i-j-k* system.

$$F_{j,x}(bend,III) = -F_{i,x}(bend,III) - F_{k,x}(bend,III) \quad (7.65)$$

The 4th-order term also produces the following expressions for forces:

$$F_{i,x}(bend,IV) = -\frac{1}{6}\, k_{bend}(cos\theta - cos\theta^o)^3 \left[\frac{x_k - x_j}{r_{ij}\cdot r_{jk}} - \frac{x_i - x_j}{r_{ij}^2}cos\theta\right] \quad (7.66a)$$

$$F_{k,x}(bend,IV) = -\frac{1}{6}\, k_{bend}(cos\theta - cos\theta^o)^3 \left[\frac{x_i - x_j}{r_{ij}\cdot r_{jk}} - \frac{x_k - x_j}{r_{jk}^2}cos\theta\right] \quad (7.66b)$$

$$F_{j,x}(bend,IV) = -F_{i,x}(bend,IV) - F_{k,x}(bend,IV) \quad (7.66c)$$

Finally, 3rd and 4th-order corrections are summed to the 2nd-order contribution, $-k_{bend}(cos\theta - cos\theta^o)\left[\frac{x_k - x_j}{r_{ij}\cdot r_{jk}} - \frac{x_i - x_j}{r_{ij}^2}cos\theta\right]$, to give the total force on each atom in the triple.

$$F_{ijk,x}(bend) = F_{ijk,x}(bend,II) + F_{ijk,x}(bend,III) + F_{ijk,x}(bend,IV) \quad (7.67)$$

The anharmonic corrections make the potential softer toward the angle opening and stiffer against the opposite deformation (Figure 7.5).

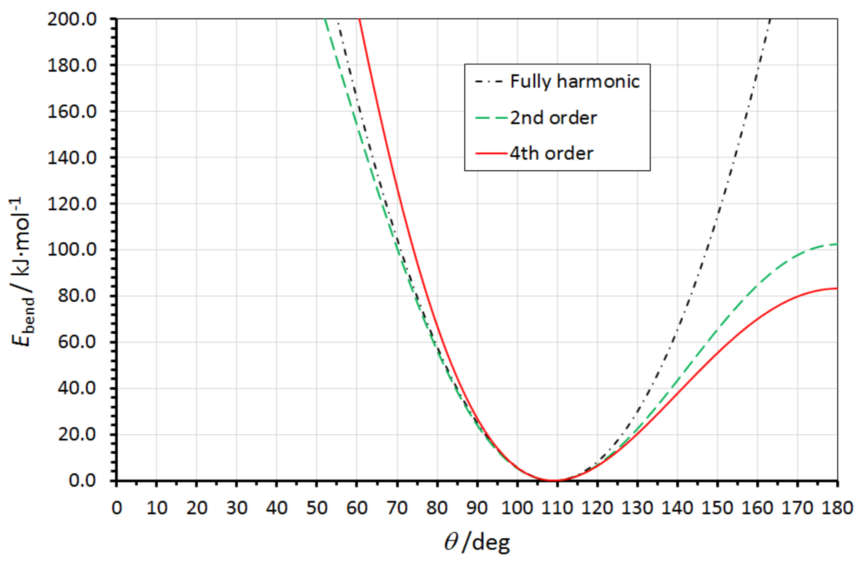


Figure 7.5. Comparison of the of the cosine bending potential (7.57) (dashed green line) with a truly harmonic potential of the form ½$k_{\mathrm{bend}}(\theta - \theta^{\circ})^2$ (black dashed/dotted line, not available in MiCMoS) and that resulting from the application of the anharmonic correction up to the 4th-order (Equation 7.62). The curves were plotted for an ideal system with θ° = 109° and k_{bend} =450 kJ·mol^{-1}.

CAUTION. It is advisable to work with smaller time steps when high order anharmonic terms are used. Moreover, pay attention to atom triple definitions in the topology file. Make sure that they are defined consistently with the desired greater or smaller ease of bending deformation, especially when 4th-order anharmonicity correction is included in the model.

7.4.2. Intermolecular force field

Also in the MD module, a choice can be made between the atom–atom Coulomb–London–Pauli (AA–CLP) and the Lennard–Jones–Coulomb (LJC) force fields (see also Sections 2.1.1 and 2.1.2). The intermolecular force field between two atoms *i* and *j* at a distance R_{ij} and bearing point charges q_i and q_j is:

(1) in the AA-CLP form:

$$E(\text{inter}, i, j, \text{CLP}) = \frac{q_i q_j}{R_{ij}} - \frac{A}{R_{ij}^4} - \frac{B}{R_{ij}^6} + \frac{C}{R_{ij}^{12}} \tag{7.68}$$

The corresponding force experienced by the atom i due to the presence of j (and *vice–versa*) as a function of distance is:

$$F(R_{ij}) = -\frac{dE}{dR} = -\frac{q_i q_j}{R_{ij}^2} + \frac{4A}{R_{ij}^5} + \frac{6B}{R_{ij}^7} - \frac{12C}{R_{ij}^{13}} \tag{7.69}$$

Coefficients *A*, *B*, *C* are computed as described Section 2.1.1.

(2) in the LJC form, as described in 2.1.2:

$$E(\text{inter}, i, j, \text{LJC}) = \frac{q_i q_j}{R_{ij}} - \frac{B}{R_{ij}^6} + \frac{C}{R_{ij}^{12}} \qquad (7.70a)$$

$$F(R_{ij}) = -\frac{dE}{dR} = -\frac{q_i q_j}{R_{ij}^2} + \frac{6B}{R_{ij}^7} - \frac{12C}{R_{ij}^{13}} \qquad (7.70b)$$

7.5 Simulation of isolated clusters

7.5.1 Translational (T) and rotational (R) momentum

In simulations of isolated clusters, net translational and rotational momenta may develop, affecting the evaluation of correlation functions with spurious faster decrease of the rotational correlation coefficient, $C(\mathbf{u}, t)$ (see Section 8.4), and faster increase of the translational diffusion coefficient, $D(t)$ (see Section 8.4). Several possible analytical constraints could be proposed to take care of this problem, most of which implying very complex algebra and time consumption. We have devised a simpler procedure, as follows. First, the array of centers of mass in the starting configuration (step zero for crystal clusters) is taken as a reference.

To dispose of T-drift, a) one atom is assigned zero velocity and zero forces, thus keeping its position constant throughout the simulation, and b) every N_r moves (typically 500-1000) the whole cluster is reset to its current center of mass.

For R-momentum, a rotation matrix is prepared as the product of three rotation matrices, one around the *x*, *y*, and *z* axis, and an array of possible back-rotations of the whole cluster is explored by varying the three rotation angles from -15 to +15° in steps of 3°. The back-rotation actually performed every N_r moves is the one (out of 11 x 11 x 11) for which the sum of the distances between current centers of mass and the reference configuration is minimum. This is an empirical equivalent of the analytical back-mapping of actual coordinates onto reference ones, with the advantage that only *N* points have to be considered for an *N*-molecule system.

As these corrections tackle the problem with a very simple strategy, they avoid complex algebra. Moreover, they affect the whole set of atomic coordinates in a rigid manner without discontinuities in the trajectories. Their efficiency has been checked by tests on correlation functions and distributions of centers of mass with and without the corrections.

7.5.2 Evaporation control

In simulations of small clusters without periodic boundary conditions, some loosely bound surface molecules may drift away in the simulation analog of evaporation. To deal with this problem, every time the translational motion is stopped by resetting the center of coordinates (see Section 7.5.1), all molecules are also checked for evaporation. Two strategies can be selected, namely a "*tethering algorithm*" and a "*deletion algorithm*".

(I) Tethering algorithm: if the center-of-mass vector of a molecule from the overall center of mass becomes longer than an assigned threshold R_{ev}, to be specified in input (Section 7.6.2, instruction line (13), it is reduced by a factor $0 < F_{ev} < 1$, where F_{ev} is another input parameter, and the whole molecule is pulled back by a vector $1–F_{ev}$ toward the centre of mass of the cluster. Using a R_{ev} threshold approximately equal to 1.5–2.0 times the cluster radius and $F_{ev} \sim 0.9$ keeps a few molecules out of the main cluster but still in its orbit, thus effectively preventing evaporation.

(II) Deletion algorithm: if the center-of-mass vector of a solvent molecule becomes longer than an assigned threshold R_{ev}, the molecule is deleted from the cluster i.e. coordinates and velocities are deleted and a new box file is produced. This strategy applies to larger clusters where deletion of a few molecules is scarcely relevant
Often the internal cohesion of the sample is such that no evaporation occurs, and no evaporation control need be applied. For example, at room T only small nonpolar solvents like chloroform may evaporate.

7.5.3 The distribution-analysis option.

In the *.mdi* input file an option is present (labels idistr and Emolim, see Section 7.6.2) that allows to store and write in the output *.pri* file molecule-molecule pair energies and distances below a certain (negative) threshold, and provides a histogram of the corresponding distributions. The distribution of center-of-mass velocities is also computed and printed in form of a histogram. This might be useful to locate specific molecular pairs that are binded by strongly attractive potentials, with application in molecular recognition studies.

CAUTION. The distribution-analysis option must be made active if a biased MD run is required.

An analysis module, *histog.for*, is available since MiCMoS v2.0 to estimate average histograms from a *.pri* file (see Chapter 8).

7.6 Running a Molecular Dynamics job

Templates and standard values for all input files are available in the manual. Tutorials are also avialable to provide working examples.

7.6.1 Batch runfile

The files needed to run a Molecular Dynamics calculation for a compound NAME are:

- NAME.*mdi*, the run control file (Section 7.6.2);
- simulation_box.*dat*, a file with the box description in *.dat* format (Sections 5.1.3 and 7.6.3);
- NAME.*top*, force field input (Section 7.6.4).

A typical MD run provides the following output:

- name3md.*pri*, output, printfile
- name3mdc.*dat*, output, trajectory file in *.dat* format (Section 5.1.3)
- name3mdo.*dat*, output, final frame position file in *.dat* format (Section 5.1.3)
- name3md.*ene*, output, energy trajectory file (Section 8.5.1)

Molecular Dynamics running command

run.mdmain NAME name2 name3

NAME must be the name of both the MD control and topology files (*.mdi* and *.top*); name2 is the full name of the simulation box (*.dat*) and name3 is the flag for naming output files.

run.mdmain module (Unix/Linux)

```
cp $1.mdi mdyn.mdi                  !MD control file
cp $1.top mdyn.top                  !forcefield file
cp $2 mdyn.bxi                      !input box file, .dat format
cp barrier.par mdyn.par             !barrier parameter file
~/programs/MiCMoS/exe/mdmain        !run execution module
rm $3mdc.dat
mv mdyn.mdc $3mdc.dat               !output, trajectory file in .dat format
rm $3mdo.dat
mv mdyn.mdo $3mdo.dat               !output, final frame position file in .dat format
rm $3md.pri
mv mdyn.mdp $3md.pri                !output, printfile
rm $3md.ene
mv mdyn.ene $3md.ene                !output energy trajectory file
rm mdyn.mdi                         !Deleting service files
rm mdyn.top
rm mdyn.bxi
rm mdyn.par
```

Detailed explanations on the meaning of the control parameters and file format follow.

7.6.2 The MD run control file (*.mdi*)

CAUTION: #lines are comment lines at fixed places: do not change their position or introduce new #lines! These #lines usually show the identity of the various parameters but may contain user's own comments.

Extension: *.mdi*; all free format.

1) A title line
2) #line --------------------------------
3) n.steps, IRVEL, ipri, ibox, idistr, timestep, Emolim, iengt, ibias

n.steps — Number of MD steps

IRVEL — Describes how velocities should be treated. For the format of the *.dat* file, see Section 5.1.3. Velocities and forces are always written in the last frame of the trajectory (file *.mdo*) after the final atomic coordinates.

- =0 Assign but do not write velocities in the output *.dat* file (this is the normal option if you are entering the MD program from a preliminary MC run).
- =1 Assign starting velocities, and write them on the output *.dat* file.
- =2 Read velocities from the input .dat file, but do not write in the output one.
- =3 Read velocities from the input .dat file, and write them in output.

ipri — Controls the amount of information printed in the *.pri* output file.

- =0 Minimum printout.
- =1 Detailed printout (force field echo, stress tensor for Parrinello–Rahman, etc.).

ibox — Determines whether the simulation uses or not periodic boundary conditions.

- =0 No periodicity (isolated cluster). If this option is chosen, the instructions on line (15) below will control the rotation–evaporation issues as detailed in Sections 7.5.1 and 7.5.2. Note that ibox = 0 is incompatible with confined simulations (inano > 0, see below). The program recognizes the incompatible instructions and stops with a warning message.
- =1 Periodic box (in three dimensions).

idistr — Controls whether the energy/velocity distribution analysis must be carried out (see Section 7.5.3).

- =0 No distribution analysis is done.
- =1 The distribution analysis of molecule–molecule pair energies ($< E_{\text{molim}}$) and their centre–of–mass velocities is carried out and written in the *.pri* file. This option is mandatory if one wants to perform a biased MD run (see parameter ibias below).

timestep — Simulation time step, expressed in ps. A value is 0.002 (2 fs) is recommended for best compromise between speed and accuracy of results.

Emolim — If idistr=1, this is the energy limit (<0) to store molecule–molecule energies in the distribution (see Section 7.5.3). A value roughly equal to the 20% of lattice energy is normally acceptable. In a biased MD run, Emolim must be defined and less negative than the upper limit for bias energy Ebiau (see line #4 below). If idistr=0, this is a dummy entry (leave 0.0).

iengt Switch parameter to decide the integration algorithm (Section 7.2).
- =0 The leapfrog integrator is used.
- =1 The velocity–Verlet integrator is used.

ibias Controls whether the distribution of kinetic energy of the MD run is biased (Section 7.2.4).
- =0 Unbiased run.
- =1 The kinetic energy distribution is biased. It is mandatory that idistr=1 (see above) if ibias =1. An extra bias.*tab* output file is also produced if ibias=1 (see below). Change its name if you don't want that it will be overwritten by further MiCMoS biased MD runs.

4) Biasing parameters (Ebial, Ebiau, Nbias, tinon, tinof).

Add this line only if ibias = 1 in the last instruction of line #3. Otherwise, skip and continue with line #5.

Ebial and Ebiau are the biasing threshold energies in kJ/mol (lower and upper limit). As the bias algorithm is intended to be applied to negative intermolecular E_{ij}'s to select and favour stabe pairs, MiCMoS expects that Ebial is more negative that Ebiau. If this is not the case, the program stops with an error message. Ebial=0.0 is a special flag to get rid of a lower bias threshold and to bias all pairs with E_{ij} < Ebiau. In any case, MiCMoS must compute the distribution of intermolecular energies to recognize what molecules fulfil the bias conditions. For this reason, it is mandatory that idistr=1 and Ebiau < Emolim when ibias = 1, or the program stops with a warning message.

Nbias determines the bias frequency, in units of MD step numbers.

The last two entries (tinon and tinof) specify the time intervals (in ps) for which the biasing algorithm is active (tinon) or inactive (tinof). For example, tinon = 5.0 and tinof = 2.0 mean that the bias will be kept active for 5 ps, then it will be switched off for 2 ps, then it will be active again for the next 5 ps, and so on. The whole on/off procedure is repeated until the last MD move is done. Note, however, that the program can handle a maximum of 5,000 on/off intervals; you should avoid splitting the simulation time into too many short periods. If you want to keep the bias active for the whole trajectory, you can either set tinon=n.steps·timestep and tinof=0.0, or directly tinon=0.0. This is a dummy entry to flag that the user wants to keep the bias active for the whole trajectory.

For example, "–20.0 –10.0 10 2.0 3.0" means that the bias algorithm will be applied every 10 MD moves for pairs whose E_{ij} fulfil $-20 \leq E_{ij} \leq -10$ kJ·mol^{-1}. The bias will be repeatedly switched on for 2.0 ps and then switched off for 3.0 ps, until the end of the dynamics. Another example: the string "0.0 –15.0 10 0.0 3.0" implies that the bias is active every 10 MD steps whenever $E_{ij} < -15$ kJ·mol^{-1} and it will never be switched off, as tinon=0.0. In this case, tinof is ignored and its value is irrelevant.

During each MD move in which the bias is active, the program prints an extra bias.*tab* ASCII text file, which summarizes the time (in ps), the number of interactions that fulfil the Ebial $\leq E_{ij} \leq$ –Ebiau condition, and the complete list of the corresponding molecular pairs.

5) #line --------------------------------

6) cutoffu, cutoffv, cutoffuv, FACTIN, ipots, ianh, inano
Energy cutoff parameters are specified in this instruction line. Refer to Section 7.3.2 for more details. All cutoffs are expressed in Å.

CAUTION: values of 16.0–18.0 are acceptable in most cases for neutral molecules, but in the presence of ions the convergence of Coulombic summations should be checked. Please note however that the standard force fields have not been calibrated for ions.

cutoffu	Distance cutoff in intermolecular sums (solute–solute).
cutoffv	Distance cutoff in intermolecular sums (solvent–solvent).
cutoffuv	Distance cutoff in intermolecular sums (solute–solvent).

CAUTION: Cutoff values must not exceed 0.55 times any of the three box lengths.

FACTIN — Damping factor for intramolecular nonbonded interactions (see Sections 6.4.2, 7.4.1 and 7.6.4).

ipots — Controls the energy functional of the Force Field.
=0 use AA–CLP
=1 use AA–LJC

ianh — Controls the function used to simulate the intramolecular stretching and bending potentials of the solute (see Sections 7.4.1.1 and 7.4.1.2). Solvent is never affected.
=0 the stretching potential is fully harmonic, as in Equations (7.56) and (7.57).
=1 the MiCMoS third order anharmonic correction is included to the stretching potential of the solute, as in Equation (7.59).
=2 the MiCMoS 4th-order anharmonic correction is applied to the bending potential of the solute, according to Equation (7.62). The stretching is treated as fully harmonic.
=3 same as options 1 and 2. Both the anharmonic corrections to stretching and bending potentials are activated at the same time.

CAUTION: It is usually a good idea to reduce the timestep of the simulation to 0.001 ps or lower when dealing with anharmonic motion.

inano — Controls whether the simulation is fully periodic in all directions, or some confinement is applied in some directions. See Section 7.2.5 for a full description of the confinement algorithm. Note that inano > 0 requires a valid barrier.*par* file (Section 5.8.1) in your working directory. This can be produced automatically by the confbox.*for* utility (Section 5.8).
= 0 the simulation is fully periodic in all directions (normal MiCMoS usage). Please set inano = 0 if you deal with nonperiodic simulations like nanoclusters, nanodroplets and nanocrystals.
= 1 one direction is non-periodic (nanolayer, confined in one direction and periodic along the other two).
= 2 two directions are non-periodic (squared nanotube, confined in two mutually orthogonal directions and periodic along the third one).
= 3 three directions are non-periodic (cubic nanocavity).

CAUTION: inano > 0 is incompatible with ibox = 0 (see above). If this parameter cobination is entered in the *.mdi* input file, the program stops with a warning message.

7) #line --------------------------------

8) N(T), Tset, Tstart, relax, itrel
This instruction controls the thermostat (Section 7.2).

N(T)	The temperature of the ensemble is updated every N(T) steps. Its value depends on the nature of the ensemble and on the run type (preliminary, production). For neutral organic molecules, N(T)~1000.
Tset	Target temperature (K).
Tstart	Starting temperature (K) (usually = Tset)
Trelax	Temperature relaxation time (τ) in equations (7.5) and (7.8). A value of 0.5–0.6 is recommended. This entry is relevant only for the weak coupling algorithm (itrel=0) or the CSVR algorithm (itrel=2).
itrel	Switch parameter to decide the T control algorithm (Section 7.2.3). =0 Weak coupling algorithm is selected. =1 Stiff coupling algorithm is selected (not recommended) =2 CSVR (Bussi–Donadio–Parrinello) thermostat is selected

9) #line --------------------------------

10) N(P), Pset, comprs, ianis, ipr, ww, iexstr
This instruction controls the barostat (Section 7.3). P control can be applied only in the presence of a periodic box, *i.e.*, ibox must be =1 in the instruction (3).

N(P)	Step frequency. The pressure of the ensemble is updated every N(P) steps.
Pset	Target pressure, in atm. This value is converted into Pa in the program. If you want to apply an external stress field, set it to 0.
comprs	Compressibility (μ_0) in equation (7.14). This parameter is relevant only if the minimal barostat algorithm is used (ipr=0). A value of 0.3–0.4 is usually appropriate but trial and error on each kind of ensemble is recommended. A larger value makes convergence faster but too large a value may also lead to sudden crashes.
ipr	Controls the barostat algorithm. =0 Minimal barostat option (Section 7.3.2). =1 Parrinello–Rahman barostat (Section 7.3.3).
ianis	Flag for isotropic or anisotropic pressure control. =0 Isotropic box (Section 7.3.1). =1 Anisotropic box (Sections 7.3.2 and 7.3.3).
ww	Coupling parameter w for the Parrinello–Rahman procedure (equation 7.43). Values of 1.0 or 2.0 kg are normally appropriate for simulations without external stress field, while larger couplings (w = 10 kg) are advised in the presence of an external stress. This parameter is active only if ipr=1, otherwise it can be safely set to 0.0.

CAUTION: Experience shows that fine tuning of ww, N(P) and N(T) parameters is crucial to avoid divergence in the Parrinello–Rahman algorithm. The reason is that Lagrangian dynamics on cell edges translates into a slight excess of molecular kinetic energy, which must be frequently dissipated through the thermostat to avoid overheating. Strategies to cope with possible instabilities that may occur in the first thousands steps of the simulation include (i) making controls over T and P more frequent (try lowering both at 200 or below); (ii) making P control slightly more frequent than over T (N(P)< N(T)); (iii) increasing the ww parameter up to 2.5–3.0. You should try different setups and choose the one that best fits your needs.

iexstr — Controls the external stress options.

- =0 No external stress. Skip to instruction line #12 below.
- =1 External stress field (Section 7.3.4). Meaningful only for anisotropic pressure control with the Parrinello–Rahman barostat (Table 7.3) and periodic structures (no isolated clusters). If iexstr = 1, the following instruction line #10 must be added.

11) Components of the external stress field

Add this line only if iexstr = 1 in the last instruction of line #10. Otherwise, skip and continue with line #12. If iexstr = 1, these are the independent components of the 2nd–rank symmetric stress tensor **S** (see Section 7.3.4). They must be given exactly in the following order: S_{11}, S_{22}, S_{33}, S_{12}, S_{13}, S_{23} with units of GPa.

Table 7.3
Summary of pressure control parameters in MD simulations.

Option	**ipr**	**ianis**	**comprs**	**ww**
Isotropic pressure control	0	0	needed	0.0
Anisotropic pressure control, minimal barostat	0	1	needed	0.0
Anisotropic pressure control, Parrinello–Rahman barostat	1	1	0.0	needed

12) #line -------------------------------

13) N(com), nwbox, nwre, npri

N(com)	Reset frequency of the centre–of–mass of the ensemble. The translational drift is suppressed every N(com) steps, according with the procedure detailed in Section 7.5.1.
nwbox	Write frequency of the trajectory. The atomic coordinates of the whole ensemble are stored in the trajectory output *.dat* file (Section 5.1.3) every nwbox steps.
nwre	Same as nwbox, for printing the energies in the *.ene* output file (Section 8.5.1).
npri	On–screen write frequency.

14) #line -------------------------------

15) N(rot-ev), romin, maxs, stepr, Rev, fact, icut

This instruction specifies the rotation–evaporation control for simulation of isolated clusters (Sections 7.5.1 and 7.5.2). These parameters are needed only if ibox=0 in instruction (3) (no periodicity).

N(rot-ev)	Reset frequency of the rotational momentum, which is suppressed every N(rot-ev) steps, according with the procedure detailed in Section 7.5.1.
romin	Minimum value of the back–rotation range to be exlpored, in degrees.
maxs	Number of rotation steps to scan the rotation space.
stepr	Step size, in degrees.

CAUTION: Recommended values for the three last parameters are: romin –15, maxs 11, stepr 3.

Rev	Evaporation distance, in Å. A molecule is considered as “evaporated” if its distance from the centre of mass of the whole cluster is > Rev (see Section 7.5.2). Rev should be at least ~ 1.5 times the average radius of the cluster.
factor	Tethering scaling parameter, F_{ev} (Section 7.5.2; only if icut=2). This number should be chosen so that the molecule is kept in the orbit of the cluster but does not crash back into the cluster. A value of 0.9 is prudential.
icut	Decide which strategy is employed to deal with evaporation. =0 No evaporation check or control is carreid out. =1 The deletion strategy is applied (Section 7.5.2). =2 The thetering strategy is applied (Section 7.5.2).

CAUTION: The program checks for incompatible options and in case stops with a warning message. For example, N(P) in instruction (9) and N(rot-ev) in instruction (13) cannot be both nonzero; P control can be applied only when ibox=1 in instruction (3); combinations such as ipr=1 (Parrinello–Rahman barostat) and ianis=0 (isotropic pressure scaling) are contradictory; etc.

Some examples of complete .mdi input files
See also Tutorials 9–11. The first input calls an unbiased (ibias=0) and unconfined (inano=0) LJC MD run (ipots=1) with a fully harmonic stretching potential (ianh=0) and full periodic boundary conditions (ibox=1), with the leapfrog integrator (0/1LF/VV =0) and the CSVR thermostat (0/1/2weak/stiff/CSVR=2). The pressure is set to 1 atm (Pset=1) with the anisotropic Parrinello–Rahman algorithm (ianis=1, ipr=1) but without external stress (iextstr=0). The last input line (#Nrot-ev…) is dummy because ibox=1.

```
Example LJC PR no stress LF CSVR
# n.steps irvel ipri ibox idstr  timestep  Emolim  iengt ibias +Ebias Nbias tinon tinof
  200000    0     0   1    0     0.002      0.0      0      0
# cutoffu cutoffv cutoffuv factin  ipots  ianh   inano
   12.0    0.0     0.0       0.7      1     0      0
#  N(T)   Tset Tstart Trelax  0/1/2weak/stiff/CSVR
    100    298   298    0.6      2
#  N(P)   Pset  comprs  0/1ianis ipr  ww   iextstr + stra11 22, 33, 12 13 23, GPa
     50    1.0   0.0      1      1    3.0    0
#  N(com)  nwbox   nwre  npri
     50    1000    1000  1000
#  Nrot-ev romin   maxs  stepr   Rev   fact   icut
    200    -15     11    3.0    30.0    0.9    2
```

The following input calls a biased (ibias=1) CLP MD run (ipots=0) with an anharmonic stretching potential (ianh=1). The bias is applied for 10 ps every 15 MD steps (*i.e.* 0.015 ps), and then switched off for the next 5 ps. Only molecules with interaction energy more negative than -20 kJ/mol will be biased. The leapfrog integrator (0/1LF/VV =0) is used, in conjuction to the Berendsen rescaling of temperature. The pressure is controlled by the anisotropic Parrinello–Rahman algorithm (ianis=1, ipr=1) under an external (iextstr=1) isotropic stress of 1 GPa. The last input line (#Nrot-ev…) is dummy because ibox=1. Note that ibias=1 implies that idistr must be 1 as well, with Ebias < Emolim.

```
Example CLP PR stress 1GPa LF Berendsen biased
# n.steps irvel ipri ibox idstr  timestep  Emolim  iengt ibias +Ebias Nbias tinon tinof
  200000    0     0   1    1     0.001     -5.0      0      1
      -20  15  10.0  5.0
# cutoffu cutoffv cutoffuv factin  ipots  ianh   inano
   12.0    0.0     0.0       0.7      0     1      0
#  N(T)   Tset Tstart Trelax  0/1/2weak/stiff/CSVR
    100    298   298    0.6      0
#  N(P)   Pset  comprs  0/1ianis ipr  ww   iextstr + stra11 22, 33, 12 13 23, GPa
     50    0.0   0.0       1      1    8.0    1
     1.0   1.0   1.0   0.0   0.0   0.0
#  N(com)  nwbox   nwre  npri
     50    1000    1000  1000
#  Nrot-ev romin   maxs  stepr   Rev    fact   icut
    200    -15     11    3.0    30.0    0.9    2
```

The next input calls for the unbiased (ibias=0) simulation of a liquid phase into a nanotube (ibox=1, inano=2) in LJC fashion (ipots=1) at 350 K with anisotropic minimal barostat (ianis=1). Note that to run this simulation, it is necessary to have a valid barrier.*par* file in your working directory, or the program will stop issuing an I/O error. See Sections 5.8 and 5.8.1 for more information.

```
Example of simulation LJC 350 K unbiased nanotube
# n.steps irvel ipri ibox idstr timestep Emolim iengt ibias + Ebias Nbias
  500000     0    0    1    1     0.001   -5.0    0     0
# cutoffu cutoffv cutoffuv factin  ipots ianh inano
   16.0    0.0     0.0       0.7      1    0    2
#  N(T)   Tset Tstart Trelax  0/1 weak/stiff
    100    350   350    0.6      0
#  N(P) Pset comprs 0/1ianis ipr ww  iextstr+stra11 22,33,12,13,23,GPa
     50    1.0   0.4      1      0    0.0    0
#  N(com)  nwbox   nwre  npri
    100     500    500    500
```

The last input is an example of MD run without periodic boundary conditions (ibox = 0). The box contains both solute and solvent molecules, as it can be appreciated by the fact that cutoffv and cutoffuv parameters are now defined. Moreover, the only possible specification for inano is 0 as well. No pressure scaling is applied (N(P)=0), as ibox in the first instruction line is 0 (the system is isolated). Accordingly, the last instruction line (# Nrot-ev…) is mandatory and specifies that suppression of rotational momentum occurs every 200 steps, while the thetering algorithm (icut=2) is selected to cope with solvent evaporation. The thethering radius (Rev) is here 30 Å with respect to the centre of mass of the simulation box. See Section 7.5 for more information.

```
Example solute+solvent clp 300 K aperiodic
# n.steps  irvel  ipri  ibox idstr  timestep Emolim iengt ibias + Ebias Nbias tinon tinof
    2000     0      0      0    1     0.001   -5.0    0     0
# cutoffu cutoffv cutoffuv factin  ipots ianh  inano
   20.0   20.0    20.0       0.7      0    0     0
#  N(T)   Tset Tstart Trelax  0/1 weak/stiff
     10    300   300    0.6      0
#  N(P)   Pset  comprs  0/1ianis ipr  ww   iextstr + stra11 22, 33, 12 13 23, GPa
      0    0.0   0.0       0      0    0.0    0
#  N(com)  nwbox   nwre  npri
     50      100   100    100
#  Nrot-ev romin   maxs  stepr   Rev    fact   icut
     200    -15     11    3.0    30.0    0.9    2
```

7.6.3 The input box file (*.dat*)

Extension *.dat*

This can be any *.dat* file in the format described in Section 5.1.3. If resulting from a previous MD run, the file has also atomic velocities and forces, as determined by the IRVEL control indicator in instruction (3) of the *.mdi* file (Section 7.6.2). This file should carry box dimension information if a periodic box is required.

7.6.4 The forcefield file (*.top*)

This has the same format as described for MC in Section 6.6.3, except that NCARDU and NCARDV must be zero, and all atom coordinates and charges must be given explicitly (no slave atoms can be defined in MD). The ***Pretop*** module (Figure 4.1, Section 5.4 and Section A7 in the Appendix) reads an *.oeh* file and prepares the best possible approximation to the pertinent force field file. Use of templates available in the Tutorials (deposited on https://sites.unimi.it/xtal_chem_group) will make things easy. In the *.top* file, as follows, all data except the title line is free format. See Section A7.1 in the Appendix for suggestions on meaningful force constants and other relevant force field parameters.

Extension *.top*.

1) A title line	format 1x,10a4
2) NCOREU	number of core atoms, solute
NCOREU lines	core atom id number, *x*, *y*, *z*, flag for atom species (see Table 2.1), raw charge. In the MD algorithm these coordinates are immaterial; the list is used only as a reminder of the pristine molecular model (***Geomet*** module, Section 8.1) and for atom type indicators and atomic point charges.
3) NCARDU	Number of slave atom cards, solute. This entry must be equal to 0 in a MD simulation.
4) NCOREV	number of core atoms, solute
NCOREV lines	core atom id number, *x*, *y*, *z*, flag for atom species (see Table 2.1), raw charge
5) NCARDV	Number of slave atom cards, solvent. This entry must be equal to 0 in a MD simulation.
6) VOLUU, VOLUV	approximate molecular volumes for solute and solvent. They are supplied by ***Pretop*** (Section 5.4).
7) NSTRU	number of bond stretching functions
NSTRU lines	4 entries, as follows: two atom id numbers of the atoms involved in the bond, k_S and $R°$ for $E(stretching)=1/2\cdot k_S\cdot(R - R°)^2$, equation (7.56). The same parameters are also used to automatically estimate anharmonic corrections if required (see Section 7.4.4.1).
8) NSTRV	as NSTRU (bond stretching), for the solvent
NSTRV lines	NSTRV lines (bond stretching parameters), for the solvent
9) NBENDU	number of bending function, solute
NBENDU lines	5 entries, as follows: three atom id numbers of the atoms involved in the bending interaction, k_b and $\theta°$ for equation (7.57, $E(bending)=1/2\cdot k_b\cdot(\cos\theta - \cos\theta°)^2$. The same parameters are also used

	to automatically estimate further anharmonic corrections if required (see Section 7.4.4.2).
10) NBENDV	as NBENDU (bond bending), for the solvent
NBENDV lines	NBENDV lines (bond bending parameters), for the solvent
11) NTORSU	number of torsion functions, solute
NTORSU lines	14 entries, as follows: four atom id numbers, identifying the atoms involved in the torsion; *K*, *f* and *m* parameters in $E(tors) = K\{1 + \cos f\,[m\tau]\}$, equation (7.58). The program ***Pretop*** assigns just standard values for *K* (50.0), *f* (–1) and *m* (+1). These **must** be reset with actual values, which can be found in most cases in Table A7.5 (Appendix, Section A7.3). ***Pretop*** also automatically assigns improper dihedrals to keep planar groups with sp^2 hybridization as $K = 100.0$, $f = -1$ and $m = +1$. Have a look at them to verify that all is ok, but usually no external intervention on improper dihedrals is required (if this is not the case, it is wise to carefully check your structure!).
12) NTORSV	as NTORSU, for the solvent
NTORSV lines	NTORSV lines (torsion parameters), for the solvent
13) NLISTU	number of intramolecular contacts, solute

NLISTU pairs of atom id numbers, solute, for a total of NLISTUx2 entries. These flag the intramolecular contacts, for which a FACTIN dampening factor is applied to scale down the potential (see Sections 6.4.2 and 7.4.1). FACTIN must be given in the *.mdi* instruction file (Section 7.6.2).

14) NLISTV	number of intramolecular contacts, solvent

NLISTV pairs of atom id numbers, solvent, for a total of NLISTVx2 entries. See NLISTU above for explanation.

15) FQ, FP, FD, FR	force field scaling parameters in eq. (2.1) (standards: 0.41, 235, 650, 77000); set them to zero if the LJC force field is used.

Add the following instructions only if Lennard-Jones potentials are used (IPO=1 in the *.mci* or *.mdi* file, Sections 6.6.2 and 7.6.2):

16) NEXTRA	number of extra L-J parameters. Non-zero only if non-library 6-12 parameters are used.
NEXTRA lines	I, J, A6, A12: equation (2.6), A6 and A12 are the 6-12 coefficients for the atom-atom contact between atom species *i* and *j*.

CAUTION: In MD stretching and bending potentials are indispensable to prevent molecular distortions. Improper torsions are also mandatory to preserve planarity of trigonal groups. Most of the MD *.top* files is directly provided by module ***Pretop*** (Section 5.4).

8. Analysis of MC and MD results

Trajectory analysis modules read atomic data from MC or MD runs in *.dat* files (Section 5.1.3). Several checks and analyses are carried out on final frames of a MC or MD simulation.

The following Table summarizes the I/O requirements of the modules illustrated in Section 8. The user must replace the strings "name1", "name2" and "name3" with the actual name(s) of the file(s) he/she is using.

Table 8.1
Modules to analyze MiCMoS trajectories (*.dat* format, see Section 5.1.3), with specified the files requested in input and the main ones produced as output. If a keyboard input is required from the user, a flag "Y" is indicated in the "Dialog mode" column.

Program	*Section*	*Input 1*	*Input 2*	*Input 3*	*Dialog mode*	*Main output*
Geomet	8.1	name1.*top*	name2.*oeh*	name3.*dat*	Y	name3geo.*pri*
Analys	8.2	name1.*top*	name2.*dat*	–	Y	name2anl.*pri*
Distrib	8.3	–	name2anl.*pri*	–	N	name2distr.*pri*
Correl	8.4	name1.*dat*	–	–	Y	name1cor.*pri*
Redene	8.5,8.5.1	name1ene.*pri*	–	–	Y	name1ene.*pri*
Datgro	8.6	name1.*dat*	–	–	Y	name1.*gro*
Naverag	8.7	name1.*dat*	–	–	Y	name1ave.*res* name1ave.*dat*
Debye	8.8, 8.8.1	name1.*inp*	name2.*oeh*	name3.*dat*	N	name3deb.*pri* name3prof.*pri*
Nanocut	8.9, 8.9.1	name1.*inp*	name2.*oeh*	–	N	name2cut.*dat*
Trajedit	8.10, 8.10.1	name1.*inp*	name2.*dat*	–	N	name2edit.*dat*
Vanhove	8.11, 8.11.1	name1.*inp*	name2.*dat*	–	N	name2van.*pri*
Renergy	8.12, 8.12.1	name.1.*inp*	name2.*top*	name3.*dat*	N	name3rene.*pri* name3rene.*ene*
Denflu	8.13	name.1.*inp*	name2.*dat*	–	N	name2denflu.*pri*
Clusters	8.14	name1.*top*	name2.*dat*	name3	Y	name3_xxx.*pri* name3_histene.*pri* name3_xxx_breaking.*pri*
Conta	8.15	–	–	name3_xxx.*pri*	N	name3_timespan.*pri* name3_dimension.*pri* name3_ordered.*pri*

8.1 The *Geomet* module

This module serves for studying the molecular conformations after a MC or MD run. It is also wise to employ it for checking purposes, *i.e.*, to verify that the simulation ended with an ensemble of reasonable structures. It reads a *.dat* file containing either a unique frame or a whole trajectory, a topology *.top* file and a molecular *.oeh* file. It performs the following tasks (all optional):

1) Check solute and solvent connectivity and bond distances (useful to find errors in the construction of the molecular geometry); check for very short intramolecular distances (impossible conformations from wrong force fields or erratic runs);
2) Produce a histogram for the distribution of torsion angles in the force field. The distribution ranges from –180° to +180°, with the usual convention for the angle signs (looking from atom 1 down the 2-3 bond, angle positive if atom 4 turn to the right). See also Section A8 in the Appendix – Reference Materials and Technical Details.
3) Calculate the root mean square deviation (rmsd) of bond stretch and bond bend in the force field according to:

$$\mathrm{rmsd}(R) = \sqrt{\sum \frac{(R - R^0)^2}{N_R}} \tag{8.1}$$

$$\mathrm{rmsd}(\theta) = \sqrt{\sum \frac{(\theta - \theta^0)^2}{N_\theta}} \tag{8.2}$$

In eq. (8.1) and (8.2), reference values R^0 and θ^0 refer to the pristine molecular model, that is, that loaded into and topology *.top* file (Section 6.6.3 or 7.6.4). The summations run on all bond distances and bond angles in all molecules in the sample. These indices and the spread of torsion angles are monitored to make sure that they stay within reasonable energy-equipartition limits without spurious molecular distortions.

CAUTION: This module had been written for solutes only with a project of extending it to solvents. The project was never carried out, however. The current status is such that the program stops without notice if a solvent is present. No wrong data are produced but the program does not work for solvents.

Running command:

```
run.geomet name1 name2 name3
```

Where the labels have the following meaning:

name1	name1.*top* file, with all bond stretch bends and torsions to be analyzed
name2	name2.*oeh* file (Section 1.4);
name3	name3.*dat* file (Section 5.1.3). If the .dat file contains several frames, the geometrical study is performed for each frame.

Printed output is on name3geo.*pri*.

The *run.geomet* instructions (Linux/Unix) are arranged as follows:

```
cp $1.top geomet.top
cp $2.oeh geomet.oeh
cp $3.dat geomet.dat
~/programs/MiCMoS/exe/geomet
rm $3geo.pri
mv geomet.pri $3geo.pri
rm geomet.top
rm geomet.oeh
rm geomet.dat
```

Answer the dialog mode, which asks for:

- TMIN, TMAX, NTBIN	Minimum torsion angle in the distribution, maximum torsion angle in the distribution (normally –180.0 and 180.0), number of bins in the distribution. NTBIN must be lower or equal 300. The step will be (TMAX–TMIN)/NTBIN.
- RLIM, TELIM	ΔR and $\Delta\theta$: limit values of distance and angle deviations, in Å and deg, from the reference geometry from atomic coordinates in the *.top* file. Bond lengths and angles that exceed the limits are printed. Recommended: 0.05 Å and 10°.
- NSTPF	when checking a trajectory *.mdc* file, the analysis is performed every NSTPF steps. This number is =1 if checking a *.mdo* or *.mco* single frame.

8.2 The *Analys* module

The ***Analys*** module computes the radial distribution function distributions for both molecular centre of mass and atom–atom intermolecular contacts. It reads a *.dat* file containing either a single frame or a whole trajectory, plus a topology *.top* file, and writes a *.pri* output file. It performs the following tasks (all optional):

1) Produce a list of short intermolecular distances below a fraction of the sum of intermolecular radii;
2) Calculation of the center-of-mass radial distribution function (RDF, equation (8.3)) for solute-solute, solvent-solvent and solute-solvent;
3) Calculation of the radial distribution function (RDF, equation (8.3)) for either selected or all pairs of atomic species, solute-solute, solvent-solvent and solute-solvent. This is useful to investigate the coordination sphere of specific atom species. In the output, these RDF data are labelled by the corresponding atomic species code numbers detailed in Table 1.1, without spaces between them. For example, 305 means that the RDF pair is formed by atom types 3 and 5, that is, by aliphatic H’s and hydroxyl H’s; 323 by aliphatic H’s (3) and –O– (23) atoms; and 1010 means the RDF of carbonyl oxygens (10) with themselves.

For the evaluation of RDF's, consider a pair of atomic species (atom–atom RDF), or pairs of molecular centers (center of mass RDF). N_i is the number of distances in a spherical distance bin of radius dR (usually 0.1–0.2 Å) and volume V_i, N is the total number of distance points and V is the total volume of the system sphere; the radial density function $g(R_i)$ for each distance R_i of the ith bin is:

$$g(R_i) = \left(\frac{N_i}{V_i}\right) \cdot \left(\frac{N}{V}\right)^{-1} = \frac{N(R_i)}{4\pi R_i^2 dR} \cdot \left(\frac{N}{V}\right)^{-1} \tag{8.3}$$

In equation (8.3), N/V is the total number density of distances, corresponding to uniform and random distribution; $g(R)$ is thus normalized and $g(R_i) > 1$ indicates a significantly high frequency of distances at R_i. For this reason, RDF is the main tool for structural analysis in liquids.

CAUTION: While a quick impression can be gathered from analysis on the last frame, averaging over *.mcc* or *.mdc* frames after steady state is highly recommended, and is indispensable for crystal simulations. If the system is at equilibrium, the last 100-500 frames of the production stage are usually enough for averaging. Module ***Distrib*** (Section 8.2) can do this using an anl*.pri* output file produced by ***Analys***, which contains the analysis of more than 1 frame.

CAUTION: For periodic-box simulations, 'total volume' means the volume of the computational box. The volume of an isolated cluster can be estimated as the sum of the molecular volumes of the constituting molecules divided by an approximate packing coefficient of 0.5 for a liquid or 0.7 for a crystal (see parameter cpack below). For very small clusters, the RDF gradually loses physical significance.

CAUTION: The RDFs are meaningful only for homogeneous systems, not for example for clusters with solute nucleus surrounded by solvent.

Running command:

run.analys name1 name2

Where the labels have the following meaning:

name1	name1.*top*, forcefield file;
name2	name2.*dat*, coordinate (trajectory) file to be analyzed;

run.analys module (Unix/Linux)

```
cp $1.top analys.top
cp $2.dat analys.dat
~/programs/MiCMoS/exe/analys
rm $2anl.pri
mv analys.pri $2anl.pri
rm analys.top
rm analys.dat
```

The output is written on a name2anl.*pri* file.

Answer the dialog mode, which asks for:

(1) nfrmi,nfrma	Start and final frame numbers. The requested analysis (see below) will be repeated for each frame between nfrmi and nfrma. Give "1 1" if your *.dat* file contains only 1 frame.
(2) PERINT	Prints all intermolecular contacts shorter than PERINT percent of the sum of intermolecular radii. 90 could be a good choice. = 0.0 to skip.

In the *.pri* file, short contacts are reported as a list of entries labelled as "`short inter, at-mol-types`". For each entry: number of the atom in the first molecule (same order as in the *.top* file), number of the first molecule (same order as in the *.dat* file), atomic species code number (as in Table 1.1), number of the atom in the second molecule (same order as in the *.top* file), number of the second molecule (same order as in the *.dat* file), atomic species code number (as in Table 1.1), actual distance (Å), van der Waals contact distance (Å).

(3) IGR	Switches on / off the evaluation of radial distribution functions. = 0: *g*(*R*)'s are computed; proceed with instruction (3). = 1; *g*(*R*)'s are not computed, end of job.
(4) cpack	If IGR = 0, an approximate packing coefficient must be given (see above). Acceptable values are 0.5 for liquids or amorphous solids and 0.7 for crystalline solids. This paremeter is important only if the simulation is not periodic (*e.g.* isolated clusters, no box), otherwise it can be safely set equal to 0.0.
(5) rminc,stepc,nbinc	Parameters to compute the centre–of–mass radial distribution function. Give zeros to skip this RDF calculation.

	rminc: minimun distance (R_{min}) value for starting the distribution.
	stepc: bin widths. Usually 0.1-0.2 Å.
	nbinc: number of bins (maximum 300). The maximum R of the distribution will be R_{min} + nbinc·stepc.
(6) rmin,step,nbin	As above, for the individual radial functions of specific atom types. Give zeros to skip all atom–atom RDF calculations. If nbin ≠ 0, rmin is the minimum distance (R_{min}) value for starting the distribution; step is the bin width; nbin the number of bins (max 300).
(7) ncpu	Number of specie pairs for the solute. Individual RDF will be computed for each of the ncpu pairs of atomic species.
	= –1: Calculate RDF's for all pairs of atomic species (solute). = 0: skip (proceed to instruction (8)). > 0: Will ask for ncpu pairs (instruction (7).
(8) ncpu ISP indicators	If ncpu is greater than zero, the program will ask for ncpu pairs of ISP specie indicators. These are the solute atom-atom species for which g(R) is to be computed, according to the codes given in Table 1.1.
(9) ncpv	Number of atom species pairs for the solvent.
	= –1: Calculate RDF's for all pairs of atomic species (solvent). = 0: skip (proceed to instruction (10)). > 0: Will ask for ncpv pairs (instruction (9).
(10) ncpv ISP indicators	ncpv pairs of solvent atom-atom species, for the solvent. Refer to Table 1.1 for the meaning of ISP indicators.
(11) ncpuv	number of atom species pairs for solute-solvent.
	= –1: Calculate RDF's for all pairs of atomic species (only solute–solvent pairs). = 0: skip (proceed to instruction (12)). > 0: Will ask for ncpuv pairs (instruction (11).
(12) ncpuv ISP indicators	ncpuv pairs of solute-solvent atom-atom species. Refer to Table 1.1 for the meaning of ISP indicators
(13) ISMO	Activates radial function smoothing. = 0: $g(R)$'s are smoothed (usually recommended) = 1: $g(R)$'s are not smoothed (*e.g.* for systems with sharply peaked g(R)'s, like perfect crystals)

CAUTION: Computing the RDFs for all the atom–atom pair types is very time–consuming, especially when large or complex molecules are studied. It is advisable to focus just on the centre of mass, and/or possibly on 1 or 2 key interactions. The centre of mass RDF is often enough to characterize the average molecular environment.

8.3 The *Distrib* module

The analysis of the MD/MC output should be made on an average of the last frames of the MD/MC run that can be obtained using module ***Distrib***. This program averages up to 5000 distributions present in a NAMEanl.*pri* output from ***Analys*** (Section 8.2), "NAME" being any valid compound id. Preliminary information on short intermolecular atom–atom contacts is always skipped, if present in the NAMEanl.*pri* file. If the system is in equilibrium, averaging over 40–200 ps of MD is usually enough to produce meaningful distributions. This corresponds to some hundreds of frames, typically 100–500, depending on parameters *timestep* and *nwbox* in MD input file *.mdi*, Section 7.6.2.

According to (8.4), each *i*th step of the *n*th distribution *P*, $P_i(n)$, is averaged:

$$\langle P_i \rangle = \frac{\sum_n P_i(n)}{N} \tag{8.4}$$

where the summation runs over a total of *N* frames for which *P* has been computed, and $P_i(n)$ is the value of the *i*th bin in the *n*th frame. In Molecular Dynamics, this operation provides time–averaged distributions over the *N* frames included into the ***Analys*** analysis (Section 8.2). For each $\langle P_i \rangle$ step of the average distribution, the corresponding estimated standard deviation (ESD) of the mean is computed according to (8.5):

$$\sigma\langle P_i \rangle = \frac{1}{\sqrt{N}} \sqrt{\frac{\sum_n (P_i(n) - \langle P_i \rangle)^2}{N-1}} \tag{8.5}$$

Averaging is mandatory in crystals to avoid conducting the analysis over a one-sided displacement along lattice vibrations. This danger is less prominent for liquids where each frame is much more homogeneous, but averaging is always a better choice. Moreover, averaged distributions are considerably smoother than non–averaged ones, and thus less prone to show instantaneous fluctuations due to background noise.

Running command:

run.distrib NAME

The program reads a NAMEanl.*pri* file produced by ***Analys*** (Section 8.2) and prints a NAMEdistr.*pri* output that contains only the averaged distributions in X, Y format. No input instructions are required. If the NAMEanl.*pri* output refers just to a single frame, *i.e.* it contains only one distribution per type, ***Distrib*** stops with a warning message, as obviously no averages can be performed in this case.

run.distrib module (Unix/Linux)

```
rm $1distr.pri
cp $1anl.pri distrib.inp
~/programs/MiCMoS/exe/distrib
mv distrib.pri $1distr.pri
rm distrib.inp
```

8.4 The *Correl* module

This module reads a trajectory mdc.*dat* or mcc.*dat* file and calculates rotational correlation functions (equation (8.6)) and root-mean-square displacements (equation (8.7)) for self-diffusion coefficients. This applies to MD output files rather than MC output trajectories.

The rotational correlation function $\tau(t)$ and the diffusion coefficient $D(t)$ and at any specific time t are estimated by the standard formulas:

$$\tau(t) = \frac{\sum_{k=1}^{N_{mol}} \mathbf{u}_k(t) \cdot \mathbf{u}_k(0)}{N_{mol}} = \langle \mathbf{u}_k(t) \cdot \mathbf{u}_k(0) \rangle \qquad (8.6)$$

$$D(t) = \frac{1}{6t} \cdot \frac{\sum_{k=1}^{N_{mol}} |\mathbf{r}(t) - \mathbf{r}(0)|^2}{N_{mol}} = \frac{1}{6t} \cdot \langle |\mathbf{r}(t) - \mathbf{r}(0)|^2 \rangle \qquad (8.7)$$

where $\mathbf{u}(t)$ is an orientation unit vector within the molecule defined as in Figure 8.1 below, and $\mathbf{r}(t)$ is the position of the center of mass at "time" t. The average runs over all the molecules in the simulation box. $\mathbf{u}(0)$ and $\mathbf{r}(0)$ are the same quantities at $t = 0$, that is, for a chosen reference, that can be any frame along the trajectory. The τ functions are dimensionless numbers between 1 (complete correlation) and 0 (no correlation), or for more clarity, between 100 and 0. The rotational correlation time is estimated as the time for $\tau(\mathbf{u}, t)$ to decay from 100 to about 20, and should be of the order of 5–20 ps for organic liquids.

The D functions are averaged over molecules within a radius of usually 30–40 Å from the overall center of the box. The quantity $\langle |\mathbf{r}(t) - \mathbf{r}(0)|^2 \rangle$ is the mean square displacement, msd; thus, from eq. (8.7), it is clear that the slope of a plot of msd vs. time, divided by 6, with distances in Å units and time in ps, gives the translational diffusion coefficient. For organic liquids, D should be of the order of 10^{-8} m^2 s^{-1} (or Å^2 ps^{-1}).

CAUTION (known bug): in periodic–box runs, the value of D *can be affected by periodic displacements of the molecule*. For vary small clusters, say $N_{mol} < 100$, care should be taken in attaching a physical meaning to the results, especially when rmsd^2 exceeds the cluster radius.

Running command:

run.correl name1

where name1 means the full trajectory file name1.*dat*. The output is found on a file called name1cor.*pri*.

run.correl module (Unix/Linux)

```
cp $1.dat correl.dat
~/programs/MiCMoS/exe/correl
rm $1cor.pri
mv correl.pri $1cor.pri
rm correl.top
rm correl.dat
```

Answer the dialog mode, which asks for:

(1) NREF, NTOT, TSTEP	number of the reference frame, total number of frames in *.dat* trajectory file, time step (in ps). All frames up to NREF-1 will be skipped.
(2) n1, n2, n3, n4, n5	Atom sequence numbers to define the reference molecular vector for rotational correlation. See Figure 8.1 below for some illustrative examples.

a) n1, 0, n3, 0, 0	vector is atom n1 to atom n3
b) n1, n2, n3, 0, 0	vector is midpoint of n1-n2 to n3
c) n1, 0, n3, n4, 0	vector is n1 to midpoint of n3-n4
d) n1, n2, n3, n4, 0	vector is midpoint of n1-n2 to midpoint of n3-n4
e) n1, n2, n3, n4, 1	vector is perpendicular to vector of case d) (useful for example for 6-fold axis in benzene)

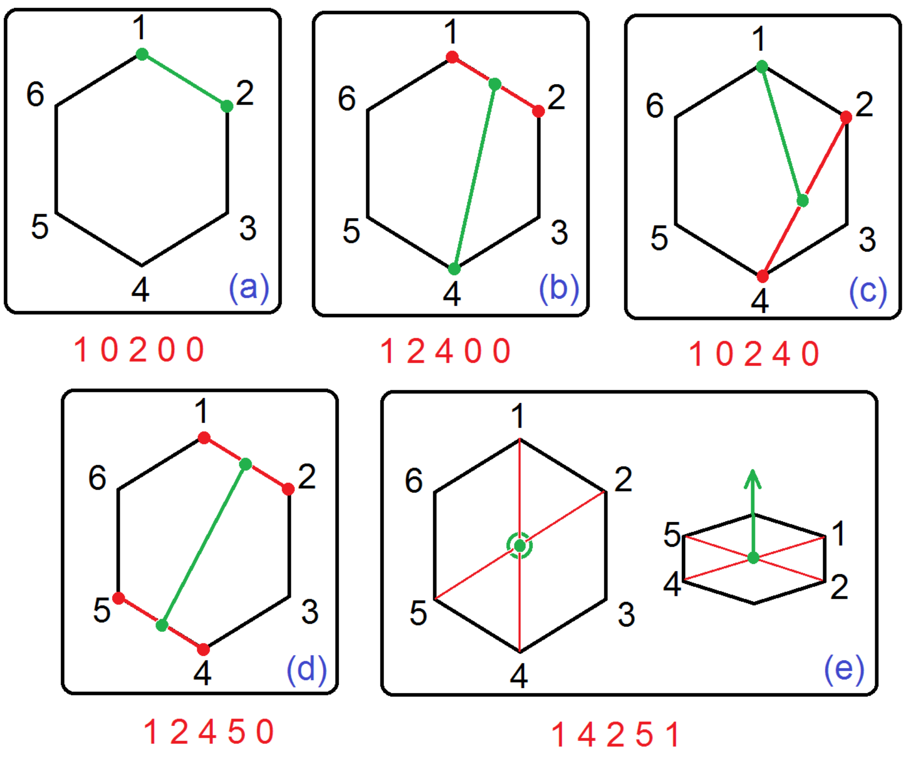


Figure 8.1. Examples for the meaning of the five designators for the definition of the *intramolecular* vector (green) for the *intermolecular* rotational correlation. Red lines represent ancillary vectors, necessary to define the green ones that are then used in equation (8.6) (see point (2) above). In the case (e), the vector is the vector product of the first two.

(3) NDIFSU, DIMAXU	NDIFSU is an atom sequence number (refer to the atom list in the *.dat*/*.top* file) for squared rms displacement in equation

(8.5); if not zero, calculate rmsd^2 of that atom, if zero calculate rmsd^2 of center of mass.

DIMAXU is a threshold distance between com's for rmsd^2 calculation; should be as large as to include all cluster or box molecules (e.g. the max dimension of the computational box) but not for example evaporated molecules.

8.5 The *Redene* module

This module reads an *.ene* file produced by a MC or MD run and interpret the energy results, providing an energy evolution profile and averages (with estimated standard deviations) over defined simulation periods.

Running command:

run.redene name1

“name1” is the name of the name1.*ene* file coming from a MC or MD run.

run.redene module (Unix/Linux)

```
cp $1.ene redene.inp
~/programs/MiCMoS/exe/redene
rm $1ene.pri
mv redene.pri $1ene.pri
rm redene.inp
```

Answer the dialog mode, which asks for:

(1) FREPA, FREPB, FREPC	Multiples of crystal cell dimension along *a*, *b* and *c*. These are the number of cells along each direction that define the supercell of the simulation box (see Sections 5.1 and 5.2) and are also reported in the MC and MD output printfiles. Set 1 for liquids.
(2) TIMST	Time step (ps) for MD runs, or equal to 1 for counting steps in a MC run.
(3) MOVE LIMITS	Min and max move. You have to specify the starting and ending moves (MC) or times (MD) to be included in the analysis. A “move” here corresponds to any frame actually written on the trajectory. Thus, for example, if you have a trajectory 100 ps long and you want to average over the last 50 ps, you should input “50 100” here. Ensure however that both $t = 50$ ps and $t = 100$ ps are included in your frames. In general, the n frames corresponding to the last t ps can be obtained $\mathbf{n=t/(nwbox \cdot dt)}$, where d$t$ is the timestep (Section 7.6.2, line 3) and nwbox the writing frequency on the trajectory (Section 7.6.2, line 12).

The output consists of 4 main tables, summarizing various energy contributions. Entries are given in kJ/mol per molecule units, and have the following meaning:

(1) Rounded time (ps), E(Lennard–Jones + Polarization), E(Coulomb), E(intramolecular, solute), E(intramolecular, solvent), E(total), E(total electrostatic+LJ, solute-solute), E(total

electrostatic+LJ, solute-solvent), E(total electrostatic+LJ, solvent-solvent). If only “solute” is present, entries corresponding to “solvent” will be identically 0.

(2) Focus on electrostatic and dispersion contributions. For solute–solute, solute–solvent and solvent–solvent, E(Lennard–Jones + Polarization) and E(Coulomb) are given, for a total of 6 columns.

(3) Intramolecular stretching, bending, torsion and nonbonded interactions from solute (columns 1–4) and solvent (columns 5–8).

(4) Interaction energies of solute and solvent with barriers, if present: Lennard-Jones and Coulomb solute-barrier; Lennard-Jones and Coulomb solvent-barrier; total Lennard-Jones; total Coulomb; total interaction energy (solute + solvent) with barrier. Note that if no confinement is applied, such as for example in MC calculations, these values will be identically zero.

(5) Time–evolution of crystal density, plus a, b, c, α, β, γ cell edges and angles.

8.5.1 Format of the *.ene* file

For each simulation step that is printed in the *.ene* output, according to what is specified in the *.mci* or *.mdi* command file (see the nwre parameter described in Sections 6.6.2 and 7.6.2), the following quantities are present:

LINE 1: nmsolu, nasolu, nmsolv, nasolv, wemolu, wemolv

Number of molecules (solute); number of atoms in each molecule (solute); number of molecules (solvent); number of atoms in each molecule (solvent); molecular weight (solute, au); molecular weight (solvent, au)

LINE 2: nstep

Number of the simulation step

LINE 3: a, b, c, alf, bet, gam, Vbox

Overall dimensions of the simulation box (Å, deg, Å^3). Box edge lengths, angles, and volume.

LINE 4: ELP(uu), EQ(uu), ELP(vv), EQ(vv), ELP(uv), EQ(uv), ELP(ubar), EQ(ubar), ELP(vbar), EQ(vbar)

Intermolecular potentials, all in kJ/mol.
ELP(uu): Total solute–solute Lennard–Jones + Polarization energy (if present);
EQ(uu): Total solute–solute Coulomb energy;
ELP(vv): Same as ELP(uu), for solvent–solvent interactions;
EQ(vv): Same as EQ(uu), for solvent–solvent interactions;
ELP(uv): Same as ELP(uu), for solute–solvent interactions;
EQ(uv): Same as EQ(uu), for solute–solvent interactions;
ELP(ubar): Solute-barrier Lennard-Jones + Polarization energy (if a barrier is present);
EQ(ubar): Solute-barrier Coulomb energy (if a barrier is present);
ELP(vbar): Same as ELP(ubar), for solvent-barrier interactions;
EQ(vbar): Same as EQ(vbar), for solvent-barrier interactions;

LINE 5: [Estr, Ebend, Etors]u, [Estr, Ebend, Etors]v

Intramolecular potentials, all in kJ/mol. These quantities are defined only from MD calculations; files from MC have a -1 marker instead.
[Estr, Ebend, Etors]u: Total stretching, bending and torsional energies for the solute molecules, in this order.

[Estr, Ebend, Etors]v: Same as above, for the solvent molecules.

LINE 6: ELP,tot EQ,tot, Eintram,u, Eintram,v, Etot

Total energies, all in kJ/mol.
ELP,tot: Total intermolecular Lennard–Jones + Polarization energy (if present).
EQ,tot: Total intermolecular Coulomb energy.
Eintram,u: Total intramolecular energy, solute.
Eintram,v: Total intramolecular energy, solvent.
Etot: Total energy of the whole simulation box, computed as ELP,tot + EQ,tot + Eintram,u + Eintram,v

8.6. The *Datgro* module

This module reads a trajectory *.dat* file containing any number of frames, and converts it into a corresponding trajectory file in Gromacs–compatible *.gro* format (see http://www.gromacs.org/ and http://manual.gromacs.org/documentation/2018/user-guide/file-formats.html#gro). Visual analysis of the *.gro* trajectory can be carried out by available graphic software. Free VMD (Visual Molecular Dynamics) from the NIH Center for Macromolecular Modeling & Bioinformatics, Theoretical and Computational Biophysics Group, University of Illinois, USA is very good to this purpose (see https://www.ks.uiuc.edu/Research/vmd/).

Running command:

run.datgro name1

Here "name1" is any valid file with extension *.dat*, coming from either dynamics or Monte Carlo simulations. Velocities, if present in the *.dat* file, are always skipped to save disk space. The program produces a name1*.gro* output compatible with specs detailed in the Gromacs user manual.

run.datgro module (Unix/Linux)

```
rm $1.gro
cp $1.dat trajectory.dat
~/programs/MiCMoS/exe/datgro
mv trajectory.gro $1.gro
rm trajectory.dat
```

Answer the dialog mode, which asks for:

ni, nf	Initial and final frame numbers for the conversion. The program converts only those frames whose sequence numbers lie between ni and nf.

Major differences of the *.gro* file with respect to the *.dat* one (Section 5.1.3) include: (i) *.gro* coordinates and cell edges must be expressed in nm; (ii) each *.gro* frame ends with cell vectors expressed as Cartesian components in a specific global reference system (see the Appendix, Section A4); (iii) molecular id numbers and residue label must be always specified in *.gro* format. ***Datgro*** numbers atoms in each molecule in ascending order, following the same sequence as in *.oeh* and *.top* files. Labels SOLU and SOLV are assigned by default to "solute" and "solvent" molecules.

CAUTION: To use the file produced by ***Datgro*** as a suitable input in Gromacs, care should be paid in making atom and molecule numbering, order and labels fully compatible with those in Gromacs topology file and Force Field libraries. It is impossible to do this automatically.

8.7. The *Naverag* module

This module reads multiple frames in a MD/MC trajectory (MiCMoS *.dat* format) to produce average structures, referred to either the crystallographic cell or the simulation box. It also estimates average anisotropic thermal parameters of all the atoms in the crystallographic unit cell; obviously, this makes sense only for MD runs.

Naverag can be useful to evaluate the average crystal structure that emerges from a MD/MC run at equilibrium, while it is pretty useless if applied to non-equilibrium structures (*i.e.* changing in time) or disordered systems, like glasses, liquids and solutions. The reason is that not all the possible averages correspond to meaningful observable states. For example, consider a dynamic system in which some tens of identical jugglers are rapidly passing a ball from their right hand to the left one and *vice-versa*. The spacetime average of this system will produce a unique (average) juggler, whose average ball will be likely frozen in midair between his/her hands. Obviously, this does not correspond to an equilibrium configuration and hardly bears physical meaning – is it a transition state? Or does it flag an intrinsic disorder, either static or dynamic? It is always recommended to pay attention in attributing physical relevance to average structures.

The ***Naverag*** module can handle a maximum of 1,000 frames and 2,500 molecules, each composed by 100 atoms at most, for a total of max 10,000 atoms in the simulation box. It recognizes “solute” and “solvent” molecules. Admitted atoms are H, B, C, N, O, F, P, S, Cl, Br and I; if other species are present, the program stops with a warning message. In agreement with the purpose of MiCMoS, the program is not designed to work with polymeric structures - only molecular crystals are allowed.

Naverag performs two tasks. First, it reads the whole trajectory and does a time average of all the frames. This produces a time-averaged frame in *.dat* format. Second, it shrinks the average frame back to the original crystallographic cell, performing a space average of all the molecules in the simulation box. This produces a shelx *.res* file with a spacetime average crystallographic structure. As no crystallographic symmetry is explicitly considered in MD, the final structure is always treated as P1, with the number of independent molecules coincident with the number of molecules in the cell. Each atom is also provided with an estimate of its thermal parameters (in the U_{ij} form).

Running command:

run.naverag name1

“name1” is the name of the name1mdc.*dat* or name1mcc.*dat* file coming from a MC or MD run.

run.naverag module (Unix/Linux)

```
cp $1mdc.dat trajectory.dat
~/programs/MiCMoS/exe/naverag
rm $1ave.res
rm $1ave.dat
rm $1service.out
mv average.dat $1ave.dat
mv service.out $1service.out
mv saverage.out $1ave.res
rm trajectory.dat
```

Answer the dialog mode, which asks for:

(1) NI, NF	Initial and final frames of the trajectory to be included, expressed as frame numbers (not MD moves). Type 0 0 if you want to include all the frames in your trajectory. If more than 1000 frames are present, the program will work just with the first 1000 ones.

The output consists of the following files:

(i) name1ave.*dat*. This is the time-averaged frame, with coordinates in Å.

(ii) name1ave.*res*. Space-time averaged crystallographic structure in the basic reference cell (P1 symmetry assumed), in shelx format. Average cell edges and angles, as well as their standard deviations, come from the average of the corresponding parameters listed in the trajectory file; thus, they are generally slightly different from those provided by ***Redene*** (Section 8.5). For each atom, the corresponding thermal parameters are also listed in the form of U_{ij}'s.

(iii) name1service.*out*. This is a service file that contains information for checking purposes. It includes a table of time-averaged Cartesian coordinates of all atoms in the simulation box, with estimated standard deviations (ESD's) plus anisotropic displacement parameters (ADPs) in the form of Uij (Å^2). The latter are partitioned into contributions from internal degrees of freedom and translations of the molecular centre of mass. Then, molecules that are space-averaged are listed. Those that were discarded from the mean due to conformational rearrangements or significant rotational motion are indicated with negative sequence numbers (see below).

The philosophy of ***Naverag*** is briefly summarized below.

The algorithm starts by setting each molecule in its local inertial system, with molecular centre of mass as the origin. Taking the first frame read as a suitable reference, each molecule in the next frames is rotated so that its inertial axes coincide with those of the corresponding molecule in the first frame. Then, atomic coordinates are time–averaged throughout the whole trajectory. At the same time, time–average variance–covariance matrix elements of atomic coordinates are computed. This gives the contribution of internal degrees of freedom to ADP's. An approximate estimate of the rigid body contribution is given by computing the variance–covariance components of the molecular centre of mass. These two are summed together with equal weights to give full atomic ADP's (in Cartesian form, Å^2: high–frequency internal motion plus low–frequency molecular rigid displacements). Cell edges and angles of the simulation box are averaged as well. Then, average molecules are back–rotated to their original orientations, and a time-average .*dat* frame (output (i)) is produced.

The next step defines the space-time averaged crystallographic cell (output (ii)). First, the origin of the reference system of the simulation box is translated so that all the molecular centres of mass become positive. This means that the final, averaged crystallographic cell might have a different origin with respect to that specified in the original .*cif* file. The enlargement factors NREPA, NREPB and NREPC specified at the beginning of each .*dat* frame (see Sections 5.1 and 5.1.3) are used to scale down the box edges and estimate the crystallographic shrunk cell. Atomic coordinates in the average simulation box are also shrunk, so that molecular centres of mass lie all within the crystallographic boundaries; then, very close (d_{CM} < 3 Å) centres of mass flag molecules that are going to be averaged. The total atomic thermal motion parameters are obtained by averaging the total variance-covariance matrix elements computed above throughout the simulation box.

CAUTION: In doing averages, ***Naverag*** traces individual atoms, not individual coordinates. This means that rapidly varying conformations in flexible portions of the molecule might result in odd geometries. The typical case is a rapidly rotating methyl group, where individual hydrogens exchange their positions several times during the trajectory. This usually produces a correct average backbone structure, but the terminal C–H bonds of the rotating methyl are unnaturally short. For any subsequent meaningful discussion of the crystallographic results, it is a good idea to get rid of wrong hydrogens and to rebuild them from scratch using the correct geometry – for example, with Mercury (C. F. Macrae *et al.*, J. Appl. Cryst., 53, 226-235, 2020) or the MiCMoS ***Retcor*** module. Obviously, thermal ellipsoids computed for disordered groups should be considered with care.

To alleviate this problem, ***Naverag*** scans molecular pairs that are going to be averaged to see whether all the corresponding atoms are reasonably close to each other. If it is found that any pair of chemically identical atoms lie farther than 0.8 Å apart, it is assumed that one translation-dependent molecule has undergone a significant conformational or rotational rearrangement, and it is therefore skipped when the averaged structure is computed. The program takes note of what molecules are used for averaging and flags those that are skipped in the name1service*.out* file by assigning negative sequence numbers to them. If too few molecules survive to perform reliable averages, the module ask for user's intervention: either the calculation is stopped before producing the *.res* file, or the scan algorithm is switched off (at your own risk!). In any case, the time average *.dat* file is always printed.

CAUTION: Remember that accurate experimental U_{ij} come from the information collected on much larger time and length scales. If you want to compare simulated thermal ellipsoids with experiment, ensure that (i) your structure is fully equilibrated; (ii) the trajectory is long enough to sample all the relevant conformers; (iii) the writing frequency of the *.dat* file is high enough to avoid missing dynamical information. In any case, *ad hoc* scaling factors should be usually applied to predicted U_{ij} components to bring them on the same scale as the experiment. A good strategy is to evaluate such factors as the average ratios between $(U_{eq})_{\text{experimental}}$ / $(U_{eq})_{\text{predicted}}$ for different atom chemical classes (*e.g.* aromatic carbons, aromatic hydrogens, aliphatic carbons, methyl hydrogens…).

Molecular translational and rotational motion might significantly alter the position of the molecular centre of mass and/or the backbone orientation, especially at high T or if the system is not fully equilibrated. This might result in odd thermal ellipsoids in some of the average molecules in the *.res* file. However, if some molecules in your average unit cell have correct thermal ellipsoids, that is, comparable with the experimental ones, you could probably ignore the ones that are clearly biased by translational or orientational disorder issues.

8.8. The *Debye* module

The ***Debye*** module reads a MiCMoS computational box in *.dat* format (see Section 5.1.3), with explicit coordinates of all atoms. A simulated diffraction pattern is evaluated by the Debye scattering equation:

$$I(\vartheta) = \langle F(\vartheta)^2 \rangle = \sum_{k=1}^{N} f_k^2 + \frac{1}{N} \cdot \sum_{k,n=1}^{m} f_k f_n \frac{sin(Qr_{kn})}{Qr_{kn}} \tag{8.8}$$

where $Q = 4\pi \sin\theta/\lambda$ is the wavevector transfer modulus and f_k, f_n are the atomic scattering factors of atoms k and n, which lie r_{kn} Å apart from each other. The first summation considers overlapping terms due to self-pairing, which cannot be included in the second one as r_{kk}=0. The second summation runs on all conceivable m atom pairs in the sample of N molecules contained in the MiCMoS simulation box. Note that atoms k and n do not need to be chemically bonded.

The Debye scattering equation (its author's name, Petrus J. W. Debije, was originally pronounced "deb-ee-a" but became Peter Debye, deb-ah-ee, in the US) is of paramount importance in modern nanotechnologies: the interested reader may find many good reviews and books on this subject in the scientific Literature (for example, see P. Scardi *et al*., Acta Cryst. (2016). A72, 589–590 and references therein). In brief, it can be used to calculate the diffraction pattern from any specimen, either gaseous or liquid, if molecules are randomly oriented. It accounts for the total scattering output, which includes disorder and thermal effects (if properly modelled). It can be used also to estimate the diffraction pattern of a crystalline specimen in the form of a finely ground powder, where small crystallites are oriented at random with respect to the incoming radiation. Thus, the resulting $I(\vartheta)/I(2\vartheta)/I(Q)$ profile can be used to evaluate the degree of ordering in a liquid, or to simulate the powder pattern for a crystal. If scattering at very small Q is considered, a Small Angle X-ray Scattering (SAXS) signal can be predicted and possibly compared with experiment.

CAUTION. If applied to a perfect crystal, the Debye scattering equation should in principle reproduce the experimental X-ray powder diffraction output but the resolution of the pattern from the Debye formula will be much lower. Equation (8.8) is a brute force summation, which in principle requires an infinite number of terms up to very high interatomic distances. When the pattern is simulated using calculated structure factors, the periodical lattice interference condition is incorporated in the expression for the structure factors, and thus the infinite lattice periodicity is implicitly considered. The added value of (8.8) is that a perfect periodicity is not needed at all, as the total scattering signal is simulated.

The atomic scattering factor of the k^{th} atom is computed as a function of the scattering vector module using analytical fitting exponential functions:

$$\begin{aligned} f(k,x) = {} & f_e(1) \cdot e^{-f_e(2)\cdot x^2} + f_e(3) \cdot e^{-f_e(4)\cdot x^2} + \\ & + f_e(5) \cdot e^{-f_e(6)\cdot x^2} + f_e(7) \cdot e^{-f_e(8)\cdot x^2} + f_e(9) \end{aligned} \tag{8.9}$$

Where $x = \sin\theta/\lambda$ and the $f_e(i)$ are fitting coefficients taken from Table 6.1.1.4. of International Tables for Crystallography, Volume C, *Mathematical, Physical and Chemical Tables*, Third Edition, Editor E. Prince, Kluwer Academic Publishers, Dordrecht/Boston/London, 2004. They come from the fitting of the form factors as predicted by quantum simulations on isolated atoms.

Application of the Debye scattering equation can be very time consuming in serial codes like MiCMoS, as the summation must be carried out across all the atom pairs. This means that for a system with N

atoms, the cost scales as N^2. To speed up the calculation is possible to exclude hydrogen atoms, whose scattering power is small, especially when they are a minor part of the molecule. Another time-saving strategy is to ignore periodicity of the simulation box and to apply equation (8.8) just to one isolated simulation box.

Running command:

```
run.debye name1 name2 name3
```

“name1” is the name of the input parameter file (name1.*inp*), which must have *.inp* extension and collects all the steering commands needed for the calculation (see Section 8.8.1 below). “name2” is the name of the name2.*oeh* file. This file (see Section 1.4) is used only for the assignment of atomic species indicators, which are not present elsewhere. Eventually, “name3” indicates the MiCMoS simulation frame name3.*dat* on which the calculation is to be performed. The user must specify in the input stream (see Section 8.8.1) on what frame interval the calculation will be done (a maximum of 1,000 frames are allowed). It is thus possible to follow the time evolution of the Debye scattering curve, or to merge different curves into a time-average signal.

run.debye module (Unix/Linux)

```
cp $1.inp debye.inp
cp $2.oeh debye.oeh
cp $3.dat debye.dat
rm $3deb.pri
rm $3prof.pri
~/programs/MiCMoS/exe/debye
mv debye.pri $3deb.pri
mv profiles.pri $3prof.pri
rm debye.inp
rm debye.oeh
rm debye.dat
```

The program produces two output files in tabular form; name3prof.*pri* contains the intensity *vs.* 2θ (or *Q*) profiles for each frame analyzed, while name3deb.*pri* summarizes the *average* intensity *vs.* 2θ (or *Q*) profile. These data can be easily adapted to be plotted with any graphic utility.

CAUTION (current program limitation). The input box can have both solutes and solvents; in a run without periodic boundary conditions (PBC) the solvents are ignored, in a run with PBC the solvents are not allowed.

8.8.1 Description of the debye.*inp* file

An example of the input stream (debye.*inp* file) for ***Debye*** is given below. The format is free.

```
# nstart nend (first and last frame number, 0 0 to analyze all)
    1      5
# lambda, theta min, theta max, step
   1.54     2.0       30.0      0.25
# ilp (=0 apply, =1 not apply LP correction), thm(theta monochrom.)
                  0                                 13.3
# ibox (0=aperiodic, 1=periodic), cutoff
                  0                 0.0
# ihyd (0=include,1=exclude H)
                  0
# iprint (0=normal, 1=extended printout)
                  0
```

Note that the steering parameters are interspersed by comment lines, each starting with a hashtag symbol "#", which can be used to summarize the meaning of the various quantities. The program ignores the comment lines.

nstart, nend (line 2)	Starting and ending frames in the trajectory. The Debye scattering equation will be applied only to frames in the *.dat* file that are included in this interval. If you want to analyze the whole trajectory, write "0 0". A maximum of 1,000 frames is allowed; if the trajectory contains more than 1,000 frames, only the first thousand will be processed.
lambda, thmin, thmax, step (line 4)	Wavelength of the X-ray beam (Å); min and max values required for the Bragg angle θ (deg); the step of θ (deg) at which the $I(\theta)$ curve is computed. Typical values for these parameters might be 1.54 (Cu Kα radiation), 2.0, 30.0 (2θ = 4-60 deg) at steps of 0.25-0.5 deg (0.5-1 deg in 2θ). The total number of steps in the intensity profile can be estimated as 1+(thmax–thmin)/step. A maximum of 5,000 steps is allowed.
ilp, thm (line 6)	ilp controls whether the Lorentz-polarization factor is applied; in case, thm is the Bragg angle θ_M (deg) of the monochromator crystal. =0 the Lp factor is applied; a valid thm is required. If thm is lower than 0, the program adds 360 deg to make it positive. =1 the Lp factor is not applied. In this case, the thm value is irrelevant. If ilp=0, each intensity step $I(\theta)$ of the total scattering profile is multiplied by the factor Lp:

$$\mathrm{Lp} = \frac{1+(\cos 2\vartheta_M)^2(\cos 2\vartheta)^2}{[1+(\cos 2\vartheta_M)^2]\sin 2\vartheta}$$

This expression assumes (i) that an ideally imperfect crystal monochromator is used and (ii) that polychromatic, monochromatized and diffracted beams are all coplanar, as for example they all lie in the equatorial plane of the Ewald sphere. Typical values for θ_M are 13.3 deg for Cu Kα and 6.2 deg for Mo Kα radiations, if a graphite monochromator is used. If the correct value is unknown, or you don't want to apply the correction for a crystal monochromator, you should select thm=0.0 deg. This excludes the contributions for the monochromator but still applies the [1+cos^{2}2θ]/[2sin2θ] factor due to the scattering from the sample. In that case, the program prints a reminder.

ibox, cutoff (line 8)	ibox specifies whether the calculation is periodic or not. =0 non periodic calculation; =1 periodic calculation. When ibox=1 (periodic), cutoff selects the cutoff distance for including molecules in the calculation. It should be something below 1.5 the average box dimension. When ibox=0, cutoff is irrelevant. Note that activating periodic boundary conditions **is often very time consuming**.
ihyd (line 10)	ihyd selects the treatment of hydrogen atoms. =0 all hydrogens are included; =1 all hydrogens are excluded.
iprint (line 12)	Controls the amount of printout. =0 normal printout (normal option) =1 extended printout (for checking purposes)

In summary, the procedure can be carried out in four different conditions:

a) no periodic box conditions (PBC), include hydrogen atoms: in this case the summation runs only on atoms in the original box. ibox=0, cutoff=0.0, ihyd=0.
b) as in a), but excluding hydrogen atoms. ibox=0, cutoff=0.0, ihyd=1.
c) PBC with generation of 26 identical boxes surrounding the central one, but including only molecules whose distance from the center of coordinates of the central box is below a given threshold. ibox=1, cutoff>0, ihyd=0.
d) as in c) but excluding hydrogen atoms. ibox=1, cutoff>0, ihyd=1.

For example, the input above requires evaluating the total scattering of the first 5 frames of the trajectory using Cu Kα radiation. The diffraction profile is predicted for $2.0 \leq \theta \leq 30.0$ deg, that is, $4.0 \leq 2\theta \leq 40.0$ deg at steps of 0.25 deg (θ) or 0.50 deg (2θ). No periodicity is exploited, but the hydrogen atoms are included, and the Lp factor is active. The prontout is normal: the ***Debye*** module will produce a deb.*out* file with a single, time averaged *I vs.* 2θ (or *Q*) profile, plus a *prof*.out file with individual profiles from the various frames analyzed.

8.9. The *Nanocut* module

Nanocut reads the information on atom identities, atom fractional coordinates and crystal packing from an input *.oeh* file (Section 1.4). Then, it cuts the periodic structure by applying user-defined boundary conditions in terms of surface lattice planes (*hkl*). The module produces a MiCMoS *.dat* frame file, interpretable by both the MC and MD engines, which contains an isolated, ordered molecular nanocluster with the desired shape. This nanocluster can be used as a starting point for subsequent MD or MC calculations.

Running command:

run.nanocut name1 name2

"name1" is the name of the input ASCII text file that includes the steering parameters for this calculation. This latter file must have *.inp* extension. Section 8.9.1 below contains a full description of the required ASCII instructions. "name2" is the name of the name2*.oeh* file for a crystal structure, *i.e.* with specified the crystallographic unit cell and coordinates expressed as fractional vectors. No input data are required from keyboard.

run.nanocut module (Unix/Linux)

```
rm $2cut.dat
rm $2cut.xyz
rm $2cut.out
rm clustercom.xyz
cp $1.inp nanocu.inp
cp $2.oeh struct.inp
~/programs/MiCMoS/exe/nanocut > nanocut.out
mv frame.dat $2cut.dat
mv coordi.xyz $2cut.xyz
mv ccom.xyz clustercom.xyz
mv nanocut.out $2cut.pri
rm nanocu.inp
rm struct.inp
rm serv1.txt
rm serv2.txt
```

The main printout is given on screen (the macro redirects it automatically to name2cut*.pri*), while the cartesian coordinates of molecules belonging to the computed nanoparticle are written in two files, name1cut*.xyz*, and name1cut*.dat*. The latter is a standard MiCMoS *.dat* frame file (Section 7.6.3). The program produces also another file, clustercom*.xyz*, where the centre of mass coordinates of all the molecules of the original array are displayed as dummy C, S or O atoms, depending whether they are included (C: solute, S: solvent) or not (O) within the nanoparticle boundaries. The *.xyz* files can be visualized by all graphical programs able to interpret the *xyz* format, including Mercury (C. F. Macrae, I. Sovago, S. J. Cottrell, P. T. A. Galek, P. McCabe, E. Pidcock, M. Platings, G. P. Shields, J. S. Stevens, M. Towler and P. A. Wood, J. Appl. Cryst., 53, 226-235, 2020). Moreover, the name1cut*.dat* file can be converted into VMD-compatible format using ***Datgro*** (Section 8.6).

The nanoparticle produced by ***Nanocut*** can be solvated. In practice, the simulation box of the solvated nanoparticle is produced by ***Nanosolv***, which merges the name1cut*.dat* file with a second *.dat* file containing an equilibrated liquid, which is used as solvent. See Section 5.6 for more details.

CAUTION. Even though is possible to create a nanoparticle from a crystal with two independent molecules in the asymmetric unit, such as solvates and co-crystals, it is not generally possible to put such a bi-component molecular cluster into a solvent with ***Nanosolv***. The reason is that MiCMoS can handle no more than two different compounds: solvating a co-crystal will result necessarily in a simulation box with three chemical species, which is forbidden by the current program limitations. The only exception is a solvated nanocluster where the external solvent is identical to either component in the nanoparticle. In that case, ensure that the two identical chemical species have the same atom sequence in both *.dat* files.

An example of a cluster produced by ***Nanocut*** is given in Figure 8.2 below.

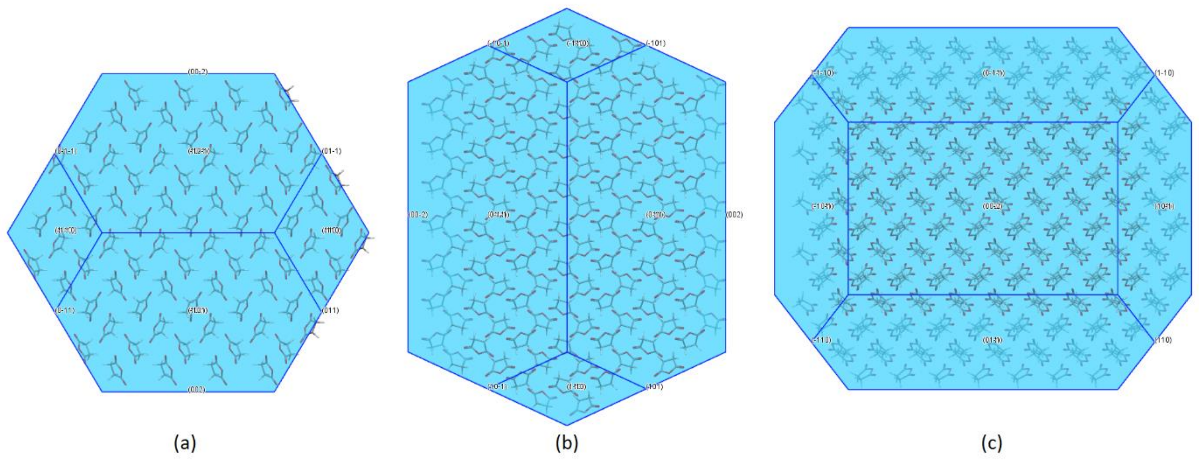


Figure 8.2. Projections along the three cell edge directions of a nanocluster of 651 molecules of succinic anhydride (7161 atoms) generated by ***Naverag***. The cluster is bound by the following lattice planes and distances (Å) (see Section 8.9.1 for details) and comes by cutting the polyhedron from a 11x11x11 slab of unit cells. (0 1 1), 21.18; (0 1 -1), 21.18; (0 0 2), 21.66; (1 0 1), 25.78; (1 0 -1), 25.78; (1 1 0), 29.64; (1 -1 0), 29.64. The transparent polyhedron shows the superposition of the corresponding BFDH morphology with the nanoparticle, as computed by Mercury on the parent SUCANH15 structure.

8.9.1 Description of the nano.*inp* file

An example of the input stream (nano.*inp* file) for ***Nanocut*** is given below. The format is free.

```
# ipack1, ipack2, ipack3 (initial cluster)
    2      2      2
# iprint (=0 normal, 1=extended printout)
    0
# nplanes (number of (hkl) planes for boundary conditions)
    3
# h,  k,  l,  dhkl
  1   1   1   10.0
  0   1   0    8.0
  0   0   1   11.54
```

Note that command lines are interspaced by comment lines, each beginning with a hashtag ("#"). Such lines are ignored by the program and can be used to pin up comments on the meaning of the steering parameters.

ipack1, ipack2 and ipack3 (line 2)	An initial cluster will be produced from the crystallographic unit cell, by exploiting all the elementary translations ranging from ±ipack*n*. For example, "`2 2 2`" means that translations of –2, –1, 0, +1 and +2 are used along all the *a*, *b* and c cell unit vectors. Overall, the initial cluster will consist of 5x5x5 = 125 unit cells in this case. Note that a maximum of 8,000 unit cells can be handled by the program.
iprint (line 4)	The parameter iprint controls the amount of information print in the output file. If iprint=1, all the centre of mass coordinates generated according to coefficients ipack*n* (see above) are printed on screen, together with the corresponding acceptance conditions. For normal use, leave iprint=0.
nplanes (line 6)	Number of (*hkl*) planes defining the boundary conditions. This parameter must be equal or higher than 3 to ensure that a closed polyhedron is defined; otherwise, the program stops with a warning message.
h, k, l, dhkl (lines 8ff)	Add nplanes (see above) rows, each specifying *hkl* and *dhkl* quantities. *hkl* are the reciprocal space coordinates of the crystallographic planes employed as boundaries of the nanoparticle. For each plane, *dhkl* is the corresponding distance from the origin (in Å). The program automatically generates also the centrosymmetric faces (–*h* –*k* –*l*), keeping them at the same distance from the origin. Overall, a total of 2·nplanes boundary conditions are used; this should ensure that the polyhedron is always closed.

For example, the input specified above generates a starting ordered box of 5x5x5 unit cells. Only those molecules whose centres of mass projections along the reciprocal space vectors (111), (010), (001) (plus the centrosymmetric ones (-1-1-1), (0-10) and (00-1)) lie within 10, 8 and 11.54 Å apart from the origin will be included in the nanoparticle.

CAUTION. The algorithm is developed so that the nanoparticle is always a convex closed polyhedron. If any of the defined (*hkl*) planes is set too far from the origin to act as an effective boundary, the elementary facets of the ipack1 x ipack2 x ipack3 original prism are used instead.

8.10. The *Trajedit* module

Trajedit reads a *.dat* trajectory file produced by either the MC and MD engines and modifies the simulation box and/or the frame sequence according to the user's specifications. For example, *Trajedit* can get rid of velocities and forces if present, it can cut the trajectory by selecting only a subset of frames, or it can print an edited trajectory with a lower frame frequency. Also, operations on the reference system are allowed: the program can change the unit cell system, including the origin. Finally, it is possible to produce an edited trajectory which contains either the solute or the solvent, or any number of user-selected molecules.

Running command:

```
run.trajedit name1 name2
```

"name1" is the name of the input ASCII text file that includes the steering parameters for the editing. This latter file must have *.inp* extension. Section 8.10.1 below contains a full description of the required ASCII instructions. "name2" is the name of the name2*.dat* file, without extension, with a MD or MC trajectory of any length. The new trajectory will be print in a name2edit*.dat* file, which can undergo any further analysis. The program produces an output on screen, that by default is redirected to a trajedit*.pri* output file.

run.trajedit module (unix/linux):

```
rm $2edit.dat
cp $1.inp trajedit.inp
cp $2.dat trajectory.dat
echo 'Working...'
~/programs/MiCMoS/exe/trajedit > trajedit.pri
echo 'Done. Output on trajedit.pri'
mv trajectory.out $2edit.dat
rm trajedit.inp
rm trajectory.dat
```

The operation requested are applied with the following order.

- First, the trajectory is scanned to select the desired bunch of frames.
- Then, a specific set of molecules within the desired frames is selected and conserved; the other molecules are erased. Selection is carried out based on either the molecular specie (solute or solvent) or a set of molecule id numbers, depending on the user's request.
- The next step is to change the origin; this is done by applying a rigid user-specified translation to the whole set of atomic coordinates. The translation vector must be specified in crystallographic coordinates.
- Then, the axes of the simulation box are transformed according with a user-defined matrix. Any transformation matrix can be used, provided it is not singular. The reason is that the transform of the contravariant coordinates in the real space is carried out by the corresponding inverted matrix. If the determinant of the transformation matrix is not unitary, a warning is issued: be aware that in that case you are changing the cell volume. If the box is not periodic, no further changes are made; if periodicity is exploited, the program fills the transformed box with the

appropriate number of molecules to avoid the occurrence of void regions. This is done by expanding the original box into the usual 3x3x3 supercell, and including into the transformed box all the molecules whose center of mass lies within the new boundaries. Two tolerance parameters (*apar1* and *apar2*) must be specified in the input to set the coordinate limit for center of mass inclusion (if origin is not changed, *apar1* should be ~ –0.5 and *apar2* ~ +0.5). This implies that the molecule count changes in the transformed box: the program updates the *nmsolu* and *nmsolv* counters in the transformed trajectory accordingly. The final cell parameters that will be written on the output are enlarged by a user-defined factor. This allows avoiding steric clashes on the new boundary of the transformed simulation box.

- Finally, if requested, the whole molecular array can be renumbered.
- In the printout, velocities and forces (if originally present) are kept or discarded, according to the user's preference. Note that they are automatically discarded if a change of the reference system is required. This ensures that, if the edited trajectory is further evolved by MD or MC, no forces are constrained to be equal by translation in the simulation box.

8.10.1 Description of the *edit.inp* file

An example of the input stream (*.inp* file) for ***Trajedit*** is given below. The format is free.

```
# itrim   iskip    icell    iextr   ivel   icentre
   1        0        1        0      1        0
# First and last frame to print (only effective if itrim = 1)
   10      50
# Cell transform matrix (only effective if icell = 1)           apar1    apar2    enlarge
  -0.5  0.5   0.5    0.5   -0.5   0.5   0.5   0.5   -0.5        -0.500    0.500     1.2
# label -1: take all; label n1,n2,n3..., take only n1,n2,n3... mols.
SOLU  12 5 45 100 105 6 7 10 27 107 99 500  508
SOLV  -1
# irenu (renumber molecules, 0:no; 1:yes) only effective if iextr/icell.ne.0
  1
# coordinates shift  (to change the origin, only effective if icentre = 1)
   -0.5  -0.5  -0.5
```

Note that command lines are interspaced by comment lines, each beginning with a hashtag ("#"). Such lines are ignored by the program and can be used to pin up comments on the meaning of the steering parameters.

1) #line -------------------------------

2) *itrim*, *iskip*, *icell*, *iextr*, *ivel*, *icentre* (line 2)

itrim: If active, only frames within the interval *istart-iend* will be saved (see below).
0: option inactive (all frames will be processed)
1: option active; only frames within the limits specified below will be passed to the new trajectory.

iskip: Selects the number of frames that are to be skipped after any valid read. The program keeps all frames for which the ratio frame number / (iskip+1) gives a zero remainder.

icell: Activates a change in the reference frame.
0: The box coordinates frame is left unchanged.

1: The coordinate frame is changed according to the transformation matrix given below. If the simulation box contains an isolated cluster (i.e. no periodic boundary conditions), the transform matrix just affects the atomic coordinates. If a periodic box is present, the reference box is rotated and filled again with those molecules whose centre of mass falls within the box. Depending on fluctuation of cell edges during the simulation and on the chosen tolerances *apar1* and *apar2* (see below), the number of molecules may vary along the trajectory.

iextr: Activates the molecule selection stream. If this option is active, only a subgroup of molecules will be selected, according with the user's choice. Note that the *iextr* task is performed before any coordinate system adjustment (*icell* option); this implies that, after the coordinate change, only the selected molecules that fall within the new box are kept.
0: No molecules will be extracted from the trajectory; equivalently, all the original molecules will be kept.
1: An user-specified group of molecules will be extracted from the trajectory (see *nextru* / *nextrv* below).

ivel: Flag to decide whether velocities and forces are to be kept. It is effective only if the original trajectory contains these information (see aldo the parameter *irvel* in the description of the *.dat* file).
0: Velocities and forces, if present, are kept;
1: Velocities and forces, if present, are erased.

icentre: Specifies if a rigid translation of the origin is to be applied.
0: No origin shift is applied;
1: An user-defined shift vector is applied to all the crystallographic coordinates (see below).

3) #line --------------------------------

4) *istart*, *iend* (line 4)

These values are active only if *itrim* = 1 (see above); otherwise, they are ineffective. For *itrim* = 1, istart flags the first frame to be processed and iend the last one. All frames in the *istart*-*iend* interval are processed; others are skipped and will not be saved into the edited trajectory.

5) #line --------------------------------

6) 9 transform matrix components, *apar1*, *apar2*, *enlarge* (line 6)

These are the components of the transform matrix, to change the reference system. They are read only if *icell* = 1 (see above), otherwise they are ignored. The transform matrix is intended to operate in the crystallographic reference system and the components are organized in a row according to tra_{11}, tra_{12}, tra_{13}, tra_{21}, tra_{22}, tra_{23}, tra_{31}, tra_{32}, tra_{33}. For example, the sequence 0.707 0.707 0 -0.707 0.707 0 0 0 1 corresponds to the matrix

$$\begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

for $\alpha = 45$ deg, i.e. it rotates the crystallographic reference system by 45 degrees around *c*. Any other transform matrix is accepted, even if the volume of the unit cell does change, provided that the transform matrix is not singular. For example, 0.5 0.0 0.0 0.0 0.5 0.0 0.0 0.0 0.5 corresponds to

$$\begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \end{pmatrix}$$

and implies that the cell edges are halved. The number of molecules in the simulation box is consistently reduced (note that this transform is ineffective for isolated clusters, where periodicity is meaningless). In general, using the correct transform matrix the user may shape the simulation box according to their needs.

Caution. If the trajectory refers to an isolated cluster (*i.e.* no box is present) and a coordinate change is required (*icell* = 1), the program prompts the user to give from keyboard an estimate for the (isotropic) dimension of the cell edge. This information is necessary to perform the required coordinate conversion.

apar1, *apar2*: These floating-point parameters set the tolerances to consider a molecule included or not in the new simulation box. Only molecules whose centre of mass crystallographic coordinates fall within these limits, i.e. $apar1 < x_{CM}, y_{CM}, z_{CM} < apar2$, will be included in the edited trajectory. Usually, values like –0.495 and +0.495 or similar (*i.e.* close to –0.5 and +0.5) provide reasonable results. The reason is that MiCMoS sets the origin at the centre of coordinates of the simulation box, and this interval allows to fill correctly the transformed box. However, the user can adjust the limits according to their needs, for example if they are dealing with liquids or droplets.

enlarge: This floating-point parameter defines the enlargement factor that is applied to rescale the transformed cell edges. The rescaling is applied just before writing the new cell in the output frame. Experience shows that *enlarge* values of 1.1-1.2 are usually enough to avoid steric clashes on the new boundaries of the transformed box. Note that, after this operation, the transformed liquid needs to be equilibrated again before any meaningful results may be deduced from the dynamics. Unitary *enlarge* factors are accepted, but the program stops if one tries to load *enlarge* < 1.0.

7) #line --------------------------------

8) *nextru*, molecule ids... (line 8)

This option is active only if *iextr* = 1 (see above). The parameter *nextru* quantifies how many solute molecules are to be kept. *nextru* = –1 is special option that means that all solutes are kept; *nextru* = 0 implies that no solutes are left. If *nextru* > 0, then the program

	expects to find a list of molecule id numbers, corresponding to their sequence number in the original trajectory *.dat* file. These can be given in any order; only the molecules with the specified id numbers will be kept in the edited trajectory.
9) *nextrv*, molecule ids... (line 9)	Same as *nextru* above, for solvent molecules. Note that contradictory requirements (for example, *iextr* = 1 and both *nextru* and *nextrv* = –1) are recognized by the program, which issues a warning or stops, depending on the severity of the error.
10) #line --------------------------------	
11) *irenu* (line 11)	Flag renumber molecules. If it is not active, the same molecule id numbers as in the original trajectory are kept. If a change of coordinate is required, some molecule id numbers may be equal: these molecules were originally translation-related in the original supercell. 0: Original molecule id numbers are retained; 1: Molecules in the edited trajectory are renumbered in sequence.
12) #line --------------------------------	
13) *shift1*, *shift2*, *shift3* (line 13)	Coordinates shift along the *a*, *b*, *c* edges of the simulation box (in crystallographic coordinates). These shifts are applied only if *icentre* =1 (see above).

You may write your comments in lines 14ff.

8.11. The *Vanhove* module

Vanhove reads a *.dat* trajectory file produced by either the MC and MD engines, and computes the isotropic van Hove distribution function either for the molecular centre of mass or for any pair of atoms, or elements, the user wants.

Running command:

run.vanhove name1 name2

"name1" is the name of the input ASCII text file that includes the steering parameters for the editing. This latter file must have *.inp* extension. Section 8.12.2 below contains a full description of the required ASCII instructions. "name2" is the name of the name2mdc.*dat* file with a MD or MC trajectory of any length. The resulting van Hove distributions are printed on name2van.pri, where they are organized in parallel columns as a function of time.

run.vanhove module (Unix/Linux)

```
cp $1.inp vanhove.inp
cp $2mdc.dat trajectory.dat
rm $2_isot.out
~/programs/MiCMoS/exe/vanhove
mv vanhove_isot.out $2_isot.out
rm vanhove.inp
rm trajectory.dat
```

8.11.1 Background

The van Hove correlation function, $G(r,t)$, allows to follow the correlated motion of particles both in space and in time, an information that may be also obtained experimentally, for example from inelastic neutron scattering experiments.

For a homogeneous system like a liquid or a glass, $G(r,t)$ depends only on the relative distance:

$$G(r,t) = \frac{1}{N}\langle\sum_{i=1}^{N}\sum_{j=1}^{N}\delta(\mathbf{r}-[\mathbf{r}_i(t)-\mathbf{r}_j(0)])\rangle \qquad (8.11.1)$$

Where N is the number of particles (atoms or molecules) included in the calculation, δ is the Kronecker delta and $\mathbf{r}_i(t)$, $\mathbf{r}_j(0)$ are the corresponding vector coordinates of the particles i and j at times t and 0, respectively. The parentheses $\langle\ldots\rangle$ imply an ensemble average over the whole set of sampling directions $\mathbf{r}$; thus, $G(r,t)$ at time t expresses the probability to find any pair of particles at distance $|\mathbf{r}| = |\mathbf{r}_i(t) - \mathbf{r}_j(0)|$. In other words, $G(r,t)$ follows how the average relative distances among the particles change in space and in time.

It is easy to see that, at t = 0,

$$G(r,0) = \frac{1}{N}\langle\sum_{i=1}^{N}\sum_{j=1}^{N}\delta\left(\mathbf{r}-\mathbf{r}_i(0)+\mathbf{r}_j(0)\right)\rangle = \delta(r)+\rho\cdot g(r) \qquad (8.11.2)$$

That is, $G(r,0)$ is proportional to the pair distribution function $g(r)$; $\delta(r)$ is the Dirac delta function and ρ is the distance number density. More interestingly, $G(r,t)$ can be separated into a self- and distinct part:

$$G(r,t) = G_s(r,t) + G_d(r,t) \tag{8.11.3}$$

Where

$$G_s(r,t) = \frac{1}{N}\langle\sum_{i=1}^{N}\delta(\mathbf{r} - [\mathbf{r}_i(t) - \mathbf{r}_i(0)])\rangle \tag{8.11.4}$$

The self-part $G_s(r,t)$ is the probability density that a particle *i* has moved by *r* in a time *t*. Equivalently, $G_s(r,t)$ can be seen as the probability density of finding at *r* a particle *i* at time *t*, knowing where the same particle was at time 0.

$$G_d(r,t) = \frac{1}{N}\langle\sum_{i=1}^{N}\sum_{j\neq i}^{N}\delta\left(\mathbf{r} - \mathbf{r}_i(t) + \mathbf{r}_j(0)\right)\rangle \tag{8.11.5}$$

Let's assume that we know that at *t* = 0 a generic particle *j* was located at $\mathbf{r}_j(0)$. Then, the distinct part of the van Hove distribution, $G_d(r,t)$, is related to the probability of find any other (different) particle at distance *r* from that place. Moreover, by definition, $G_d(r,0) = g(r)$ at *t* = 0, excluding possible normalization factors.

In normal usage, both functions are normalized so that $\int G_s(r,t)dr = 1$ and $\int G_d(r,t)dr = N - 1$.

If desired, the user can select normalization conditions analogue to those employed by ***Analys*** (Section 8.2) to calculate the radial distribution function. However, note that the ***Vanhove*** does not perform distinct calculations of the centre of mass for the solute or the solvent, as it is intended to look for long-range correlations in both space and time, rather than to investigate the local average coordination. Thus, the whole information (solute-solute, solute-solvent and solvent-solvent) concur to define a van Hove distribution. It is still possible to extract information on the solute / solvent parts, though; for example, the user may edit the trajectory with ***Trajedit*** (Section 8.11), so that only the desired information is left. For example, if only solute-solute (solvent-solvent) contributions are looked for, solvent (solute) molecules must be erased frame by frame. The solute-solvent cross information may be obtained by subtracting the solute-solute and solvent-solvent contributions from the whole van Hove.

8.11.2. Description of the *vanhove*.inp file

An example of input stream for *vanhove* is given below. The format is free.

```
# ifirst(first frame)  ilast(last frame)   iuse(skip frames)   max_bin (of the van Hove distribution)
        10                  14                     1                     250
# delta_r (bin width of the van Hove)   delta_t (time step, in ps)
        0.5                                   0.002
# Type of van Hove (ivH, 0:com, 1:atom count)  Normalzation(norm, 0:particle count; 1:particle density)
                              1                                   1
# Atoms to be included
SOLU   8  -1 1  15
SOLV   0  0 0  0
```

Note that command lines are interspaced by comment lines, each beginning with a hashtag ("#"). Such lines are ignored by the program and can be used to pin up comments on the meaning of the steering parameters.

1) #line --------------------------------

2) ifirst, ilast, iuse, max_bin

ifirst	First frame to consider.
ilast	Last frame to consider.
iuse	Starting from *ifirst*, the analysis is done every *iuse* frames.
max_bin	Number of bins for the calculation of the discrete van Hove distributions. The maximum *r* (the extension) of the distribution is governed by the product max_bin · delta_r (explained below).

Caution: If the extension is insufficient to safely allocate all the bins, the program stops and a warning is issued.

3) #line --------------------------------

4) delta_r, delta_t — Control the bin width and the time step.

delta_r	Bin width of the van Hove distributions, in Å. All distances *r*'s that differ less than delta_r are placed into the same bin.
delta_t	Timescale of the Molecular Dynamics simulation, in ps. It corresponds to the timestep parameter of the mdi file (see Section 7.6.2).

5) #line --------------------------------

6) ivH, norm — Controls the type of the van Hove distributions.

ivH	= 0 : the van Hove distributions is based on molecular centre of mass;
	= 1 : the van Hove distributions is based on individual atom-atom distances. Specific atoms to be used are given on lines 8 and 9 below. Only distances relating different atoms (or atomic species) in different molecules are included in the distribution. In other words, intramolecular distances are always skipped.

norm	Selects the normalization factor for the van Hove distribution.
	= 0 : the particle count is used as a normalization factor. This implies that the whole van Hove distribution is normalized over the total number of particles, N, and at the same time $\int G_s(r,t)dr = 1$ and $\int G_d(r,t)dr = N - 1$.
	= 1 : the same normalization procedure for the calculation of the pair distribution function $g(r)$, as implemented in analys.for (Section 8.2), is employed. This means that the ith bin is divided by $4\pi R_i^2 dR \cdot N/V$, i.e. by $^4/_3\,\pi[(R_i + \text{delta\_r})^3 - (R_i)^3] \cdot N/V$. As usual, N/V is the total number density of distances, which sets the reference for a perfect random distribution.

7) #line --------------------------------

8) labl,izu1,natu1,izu2,natu2

These instructions are effective only if ivH = 1.

labl	This is a user-defined label. It can be any alphanumeric quantity (4 characters long). Usually, it is just a memo for solute ("SOLU"), as the following parameters on this line refer to the solute.
izu1	Atomic number of the first solute atomic specie to be included in the evaluation of the van Hove distribution.
natu1	Atom id number that is used to compute the distribution. natu1 correspond to the number position of the desired atom in the solute atom list written in both the topology and the trajectory. For example, "8 21" means that the #21 oxygen atom on the atom list is selected. If a "-1" is given, all solute atoms with izu1 atomic number are used. A "0" implies that no izu1 atoms are employed.
izu2	Atomic number of the second solute atomic specie to be included in the evaluation of the van Hove distribution.
natu2	Same as natu1, for the second solute atomic specie.

Caution: Only intermolecular distances are included in the calculation. Distances relating atoms within the same molecule are skipped.

9) labl,izv1,natv1,izv2,natv2

These instructions are effective only if ivH = 1. They are the same as in line 8) above, referring to the solvent.

labl	This is an user-defined label. It can be any alphanumeric quantity (4 characters long). Usually, it is just a memo for solvent ("SOLV"), as the following parameters on this line refer to the solvent.
izv1	Atomic number of the first solvent atomic specie to be included in the evaluation of the van Hove distribution.
natv1	Atom id number that is used to compute the distribution. natv1 correspond to the number position of the desired atom in the solvent atom list written in both the topology and the trajectory. For example, "8 21" means that the #21 oxygen

	atom on the atom list is selected. If a “-1” is given, all solvent atoms with izv1 atomic number are used. A “0” implies that no izv1 atoms are employed.
izv2	Same as natu1, for the second solvent atomic specie.
natv2	Atom id number of the izv2 atom that is used to compute the distribution. If a “-1” is given, all solute atoms with izv2 atomic number are used. As above, a “0” implies that no izv2 atoms are employed.

The user may decide which specific atoms, or atom classes, are to be included in the calculation. Suitable choice of natu1,2 and natv1,2 switches, allows to cover any case. For example,

```
# Atoms to be included
SOLU   8  -1 1  15
SOLV   0   0 0   0
```

The above instructions require to compute the van Hove distribution by considering the distance between the hydrogen atom number 15 of the solute and any of the oxygen atoms in the solute. It is implicit that only intermolecular distances are included in the statistics.

```
# Atoms to be included
SOLU   8  1 0  0
SOLV   7  4 0  0
```

Here, we want the van Hove distribution for the oxygen atom number 1 of the solute and the nitrogen atom number 4 of the solvent. Note that “`SOLV    0 0 7 4`” would have been totally equivalent.

```
# Atoms to be included
SOLU   0 0 0 0
SOLV   7 -1 7 -1
```

This calculation evaluates the van Hove distribution among all the intermolecular distances involving nitrogen atoms of solvent molecules.

8.12. The *Renergy* module

Renergy reads a *.dat* trajectory file produced by either the MC and MD engines, and re-computes the potential energies, both intramolecular and intermolecular, based on the atomic coordinates. If requested, the program also produces extended files with individual relevant (*i.e.* most attractive) molecule-molecule interaction energies.

Running command:

```
run.renergy name1 name2 name3
```

"name1" is the name of the input ASCII text file that includes the steering parameters. This file must have *.inp* extension. Section 8.13.1 below contains a full description of the required ASCII instructions. The label "name2" indicates the name2.*top* topology file, which must be read to retrieve the atom identities, the atom charges, and the intramolecular parameters of the force field. Finally, "name3" is the name of the name3.*dat* file with a MD or MC trajectory of any length; it also sets the output file names.

Note that the program is compiled in serial mode. If you want, you can compile ***Renergy*** in parallel mode following the instructions detailed below.

By default, the program prints two main files. The name3rene.*pri* output lists the total intramolecular and intermolecular energies, as well as the corresponding decomposition into dispersive and electrostatic contributions, plus an estimate for the cohesive energy of the system. The usual decomposition into solute-solute, solute-solvent and solvent-solvent contributions is maintained. The name3rene.*ene* output is equivalent to the *.ene* file that can be printed on the fly while the trajectory is computed (see the parameter nwre in Section 7.6.2; Section 8.5.1 describes extensively of the *.ene* format).

If requested by the user (see Section 8.12.1 below), for each frame in the trajectory ***Renergy*** can print detailed lists of molecule-molecule contacts, which include center of mass distance, intermolecular dispersive and Coulomb contributions, and total intermolecular energies. Distinct files are produced for solute-solute (*name3*uu.pri), solute-solvent (*name3*uv.pri), and solvent-solvent (*name3*vv.pri) interactions. Only molecule-molecule energies lower than a user-defined threshold on the total energies are printed.

Renergy is intended to perform one or more of the following tasks. It can be used to obtain information on specific molecule-molecule interactions, which are not usually printed during the normal trajectory analysis. It can be also employed to test the effect of different cutoffs, or force fields, on the estimated cohesive energies, getting rid at the same time of redistribution effects like those due to kinetic bias (Section 7.2.4), anisotropic pressure (Sections 7.3.2 and 7.3.3) and nanoconfinement (Section 7.2.5). Note, however, that ***Renergy*** does not recalculate forces and velocities; in other words, it only takes note of atom positions and gives back the corresponding intramolecular and intermolecular energies as a function of time. Finally, ***Renergy*** can also be used in conjunction with ***Trajedit*** (Section 8.10) *e.g.* to compute the potential energies of specific molecular subsets. For example, you may use ***Trajedit*** to select a group of molecules that is of some interest (*e.g.* a hydrogen bonded cluster, a droplet of solute…). Then, you can use ***Renergy*** to analyze in depth their contribution to the total energy of the system. Note that such a partitioning is possible formally as MiCMoS relies on pairwise atom-atom summations, which in turn sum up exactly to give molecular contributions. Clearly, ignoring multi-body interactions and correlations is an approximation. This is the cost for an exact partitioning.

Caution. Being equal the force field, the factin parameter and the various cutoffs, small numerical discrepancies, lower than 0.1 %, are to be expected in total energies when one compares the outcomes of *renergy* with those obtained on the fly from the original trajectory calculation. These are likely due to rounding of atomic coordinates in the written trajectory. We verified that cohesive energies are essentially not affected from these small (random) errors.

run.renergy module (Unix/Linux)

```
rm $3rene.pri
rm $3rene.ene
rm $3uu.pri
rm $3vv.pri
rm $3uv.pri
cp $1.inp renergy.inp
cp $2.top topology.top
cp $3.dat trajectory.dat
~/programs/MiCMoS/exe/renergy
rm renergy.inp
rm topology.top
rm trajectory.dat
mv renergy.pri $3rene.pri
mv renergy.ene $3rene.ene
mv emoluu.pri $3uu.pri
mv emolvv.pri $3vv.pri
mv emoluv.pri $3uv.pri
```

To be compiled, the program requires to have access to MiCMoS libraries. The following macro should be employed. Note that these instructions are already present in the standard compilation macro *run.compileB*, thus you are expected to do nothing to compile ***Renergy***. See also the installation notes at the very beginning of this manual.

Compilation instructions for ***Renergy*** (serial mode)

```
gfortran -O2 -mcmodel=medium -c -std=legacy ~/programs/MiCMoS/SourceB/alldat.for
gfortran -O2 -mcmodel=medium -c -std=legacy ~/programs/MiCMoS/SourceB/mdlibs.for
gfortran -O2 -mcmodel=medium -c -std=legacy ~/programs/MiCMoS/SourceB/mcmdpo.for
gfortran  -O2  -mcmodel=medium  -static  -std=legacy  ~/programs/MiCMoS/SourceB/renergy.for  alldat.o  mdlibs.o
mcmdpo.o -o renergy
[ -f "./renergy.exe" ]  && mv "./renergy.exe"  renergy
rm *.o
mkdir -p ~/programs/MiCMoS/exe
mv renergy ~/programs/MiCMoS/exe/.
```

By default, the program is compiled in serial mode. However, you can also produce by yourself a parallel version of ***Renergy*** by executing the following instructions. Please note that these will overwrite the serial ***Renergy*** executable produced by *run.compileB* during the installation process. To avoid this, you may change the name of the executable.

Compilation instructions for ***Renergy*** (parallel mode)

```
gfortran -O2 -mcmodel=medium -c -std=legacy ~/programs/MiCMoS/SourceB/alldat.for -fopenmp
gfortran -O2 -mcmodel=medium -c -std=legacy ~/programs/MiCMoS/SourceB/parallel/mcmdpo.for -fopenmp
gfortran -O2 -mcmodel=medium -c -std=legacy ~/programs/MiCMoS/SourceB/parallel/mdlibs.for -fopenmp
gfortran -O2 -mcmodel=medium -frecursive -std=legacy ~/programs/MiCMoS/SourceB/parallel/renergy.for alldat.o
mdlibs.o mcmdpo.o -o renergy -fopenmp
[ -f "./renergy.exe" ]  && mv "./renergy.exe"  renergy
rm *.o
mkdir -p ~/programs/MiCMoS/exe
mv renergy ~/programs/MiCMoS/exe/.
```

To execute the parallel version, you should add the following instruction `export OMP_NUM_THREADS=<N>` to the first line of the run.renergy module file (change <N> with the proper number of threads).

CAUTION. The parallel version of ***Renergy*** is not tested to date. You should check that it works properly before extensive use.

8.12.1. Description of the renergy.*inp* file

An example of input stream for ***Renergy*** is given below. The format is free.

```
Example CLP energy recalc
# idstr  timestep  euu   evv    euv  cutoffu cutoffv cutoffuv   ipots    factin    indiuu    indivv  indiuv
    1      0.001   -5.0 -15.0 -10.0   15.0    15.0     15.0        0       0.7        1         1       1
```

Note that command lines are interspaced by comment lines, each beginning with a hashtag (“#”). Such lines are ignored by the program and can be used to pin up comments on the meaning of the steering parameters.

1) Title line Use this line to sketch some information on your calculation. The format is free.

2) #line --------------------------------

3) idstr, timestep, euu,evv,euv, cutoffu, cutoffv, cutoffuv, ipots, factin, indiuu, indivv, indiuv

idstr Controls whether the energy/distance distribution analysis must be carried out (see Sections 7.5.3 and 7.6.2).

=0 No distribution analysis is done

=1 The distribution analysis of molecule–molecule pair energies (< Emolim) and their centre–of–mass velocities is carried out and written in the *.pri* file.

timestep Timescale of the Molecular Dynamics simulation, in ps. It corresponds to the timestep parameter of the mdi file (see Section 7.6.2).

euu If idstr = 1, this is the energy limit (<0) to store solute–solute energies and distances in the distribution (see Section 7.5.3). In other words, euu takes the place of the *emolim* instruction in the original *.mdi* input file (Section 7.6.2). This option is directly available only for the solute molecules. If interested in the solvent, you may run this program on a suitably edited trajectory with *Trajedit* (Section 8.10). If indiuu =1 (*vide infra*), individual solute–solute energies are stored in the *name3*uu.pri output file, provided that their total interaction energies are more negative than euu.

evv If indivv =1 (*vide infra*), individual solvent–solvent energies are stored in the *name3*vv.pri output file, provided that their total interaction energies are more negative than evv. As for euu, also evv < 0 is required.

euv If indiuv =1 (*vide infra*), individual solute–solvent energies are stored in the *name3*uv.pri output file, provided that their total interaction energies are more negative than euv. As for euu, also euv < 0 is required.

cutoffu	Distance cutoff in intermolecular sums (solute–solute). See Section 7.3.2 for more details. All cutoffs are expressed in Å. Note that cutoffu = 0.0 is incompatible with indiuu = 1.
cutoffv	Distance cutoff in intermolecular sums (solvent–solvent). Note that cutoffv = 0.0 is incompatible with indivv = 1.
cutoffuv	Distance cutoff in intermolecular sums (solute–solvent). Note that cutoffuv = 0.0 is incompatible with indiuv = 1.
ipots	Controls the energy functional of the Force Field. =0 use AA–CLP =1 use AA–LJC
factin	Damping factor for <u>intramolecular</u> nonbonded interactions (see Sections 6.4.2, 7.4.1 and 7.6.4).
indiuu	Flags whether individual molecule-molecule energies in the solute should be printed. =0 No explicit recording of molecule-molecule interactions. =1 Individual solute-solute interactions are recorded in the output file *name3*uu.*pri*.
indivv	Flags whether individual molecule-molecule energies in the solvent should be printed. =0 No explicit recording of molecule-molecule interactions. =1 Individual solvent-solvent interactions are recorded in the output file *name3*vv.*pri*.
indiuv	Flags whether individual cross-interaction energies between the solute and the slvent should be printed. =0 No explicit recording of molecule-molecule interactions. =1 Individual solute-solvent interactions are recorded in the output file *name3*uv.*pri*.

8.13. The *Denflu* module

Denflu reads a *.dat* trajectory file produced by either the MC and MD engines, and computes the local density fluctuations, as opposed to the average density of the simulation box. This can be useful, for example, to spot early aggregation phenomena or inhomogeneities in liquids and glassy states.

Running command:

run.denflu name1 name2

"name1" is the name of the input ASCII text file that includes the steering parameters. This latter file must have *.inp* extension. Section 8.14.1 below contains a full description of the required ASCII instructions. "name2" is the name of the name2mdc.*dat* file with a MD or MC trajectory, without extension. A maximum of 2000 frames can be processed at once. The output is an ASCII text file named name2denflu.*pri*.

run.denflu module (unix/linux)

```
rm $2den.pri
cp $1.inp denflu.inp
cp $2.dat trajectory.dat
~/programs/MiCMoS/exe/denflu
mv denflu.out $2denflu.pri
rm trajectory.dat
rm denflu.inp
```

In each frame, the program partitions the total volume of the simulation box into a grid according with the user's instructions. The grid must contain no more than 10^6 volume elements. Then, each molecule in the simulation box is associated with one and only one volume element, the one that is closest to the corresponding center of mass. For each volume element, the number of molecules it contains is evaluated. Thus, it is possible to compute the instantaneous density in each volume element of the grid. This value can be compared with the total density of the simulation box, as well as with the average density over the whole trajectory.

For each volume element, a time average value of the density is provided by averaging its point density throughout the trajectory. The averaged squared density difference is computed locally in each volume element i according to

$$\langle \Delta^2 \rho_i \rangle = \langle (\rho_i - \langle \rho_i \rangle)^2 \rangle \tag{8.13.1}$$

Where ρ_i is the density of the i^{th} volume element, $\langle \rho_i \rangle$ is the corresponding trajectory average, and brackets $\langle \dots \rangle$ denote the temporal average as well. An index can be defined, to quantify the average fluctuations with respect to the system density. Following Moynihan & Schroeder, *Journal of Non-Crystalline Solids* 160, **1993**, 52-59, we define an average density fluctuation $\langle \Delta^2 \rho \rangle$ by averaging the individual $\langle \Delta^2 \rho_i \rangle$ over all the volume elements in the grid, and then by normalizing by the squared total density. The fluctuations can be also related to thermodynamic response coefficients according to:

$$\frac{\langle \Delta^2 \rho \rangle}{\langle \rho^2 \rangle} = \frac{k_B T \cdot \Delta k}{\langle V \rangle}$$

Here, brackets $\langle \ldots \rangle$ indicate the spatial average through the grid elements, ρ^2 is the corresponding average squared density, and $\langle V \rangle$ is the average volume element. T is the absolute temperature and k_B the Boltzmann constant (1.380649·10^{-23} J·K^{-1}). Finally, Δk is the difference between the liquid and glass isothermal compressibilities.

The program also prints a table with the time-averaged parameters of all the grid elements. For each point volume i, the Cartesian coordinate of the volume centroid are given, together with: (i) the average density $\langle \rho_i \rangle$ (in g·cm^{-3}), (ii) the corresponding standard deviation of the mean ($\sigma\langle \rho_i \rangle$), (iii) the average density difference from the average density ($\langle \rho_i - \langle \rho_i \rangle \rangle$), (iv) the average squared density difference ($\langle (\rho_i - \langle \rho_i \rangle)^2 \rangle$), (v) the local fluctuation for that grid element ($\langle \Delta^2 \rho_i \rangle / \langle \rho_i \rangle^2$) and (vi) the density difference with respect to the macroscopic one ($\rho - \langle \rho_i \rangle$).

CAUTION. The program is equipped with internal checks for consistency. It may happen that the center of mass of a molecule falls exactly halfway between two volume elements and thus it is counted twice. In this case, the program stops and issues an error of "*Internal molecule count mismatch*". You may avoid the problem by changing the grid of volume elements (*i.e.* ibx, iby and ibz – Section 8.13.1 below).

8.13.1 Description of the *denflu*.inp file

An example of the input stream (denflu.*inp* file) for ***Denflu*** is given below. The format is free.

```
# bin numbers:  ibx   iby    ibz
                10    10     10
# Starting frame   Ending frame   iprint
     1               1000           0
# Temperature in K
   350
```

Note that command lines are interspaced by comment lines, each beginning with a hashtag ("#"). Such lines are ignored by the program and can be used to pin up comments on the meaning of the steering parameters.

1) #line --------------------------------	
2) ibx, iby, ibx	Number of grid bins along x, y and z. The cell edges are partitioned according to the ibx, iby and ibz factors to define the grid steps and the corresponding volume elements. Each grid element is a parallelepiped with edges *a*/ibx, *b*/iby and *c*/ibz long, where *a*, *b* and *c* are the box edges. Note that a maximum of 10^6 volume elements are allowed in the grid.
3) #line --------------------------------	
4) istart, iend, iprint	*istart* flags the first frame to be processed and *iend* the last one. For example, if you have a total of 1000 frames in your trajectory, 500 800 means that the analysis will be carried out from the frame 500th to the frame 800th. All frames in the *istart-iend* interval are processed; others are skipped. *iprint* controls the amount of output that is printed. 0 is the normal option; 1 implies full printout, which includes the orthogonalization matrices and frame-by-frame information on the box parameters.
5) #line --------------------------------	

6) atemp	Temperature of the simulation to which the trajectory refers, in Kelvin. It is used just to evaluate the final thermodynamic properties.

8.14. The *Clusters* module

Clusters reads a topology (.top) and a trajectory (.dat) and look for stable aggregates of molecules based on geometry or energy thresholds.

Running command:

run.cluster name1 name2 name3

where:
- name1: is the name of the *top* topology (without extension).
- name2: is the name of the *dat* trajectory to analyze (without extension).
- name3: is the name of the output files (without extension).

The program produces three different files:
- name3_xxx.*pri*: lists all the cluster found frame by frame with the geometric (xxx = geo) or energetic (xxx = ene) criterion.
- name3_histene.*pri*: contains the distribution of the number of interacting molecules vs the energy of interactions. This doesn't depend on the selected interaction criterion.
- name3_xxx_breaking. *pri*: contains the frequency of the persistence of molecule-molecule interactions as a function of the number of frames.

The two criteria for molecule-molecule recognition work as follows:

- Geometric mode: two molecules are considered interacting if (i) the distance between the donor and the acceptor atoms is lower than the sum of their van der Waals radii, scaled by a user-defined factor ($d_{H\text{-}A} \leq k^*(r_{vdw,H} + r_{vdw,A})$), and (ii) the D-H···A angle is below a given threshold ($\alpha_{D\text{-}H\cdots A} \leq \vartheta$). The user has the possibility to define the list of donor and acceptor atoms, together with the distance scaling (default: k = 0.9) and the angle of the interaction (default: ϑ = 120 deg).

- Energetic mode: molecules are considered bonded if their interaction energy is lower than a given threshold (E_{ij} < threshold). No default value is set.

run.cluster module (unix/linux)

```
#!/bin/bash
if [ -f "$3_geo.pri" ]; then rm $3_geo.pri ; fi
if [ -f "$3_ene.pri" ]; then rm $3_ene.pri ; fi
rm "histene.out"
rm "breaking.out"
cp $1.top cluster.top
cp $2.dat cluster.dat
~/programs/MiCMoS/exe/cluster
if [ -f "cluster_geo.out" ]; then crit="geo" ; fi
if [ -f "cluster_ene.out" ]; then crit="ene" ; fi
mv cluster_${crit}.out $3_${crit}.pri
mv histene.out $3_${crit}_histene.pri
mv breaking.out $3_${crit}_breaking.pri
rm cluster.dat
rm cluster.top
```

The program works interactively with the following flow:

i1	modinter: (1 = geometric, 2 = energetic)	**modinter** Select which interaction criterion you want to use: - 1: geometric. Two molecules interact if their donor-acceptor distance is lower than the sum of van der Waals radii scaled by a scaling tolerance factor, and the D-H···A angle is lower than the given threshold. - 2: energy threshold. Interactions occur when two molecules have interaction energy lower (more negative) than the given threshold
i2	Which potential? (0 = clp, 1 = ljc)?	**ipot** Select one of the two available force field for the energy calculation in *modinter* = 2 and for the *name3_histene.out* output. Refer to *Section 2.1* of the manual for more details about the force fields. - 0: CLP (Coulomb-London-Pauli force field. - 1: LJC (Lennard-Jones-Coulomb) force field.
i3	Initial and final frames to analyze:	**first_frame last_frame** - *first_frame*: first frame to analyze from the *dat* trajectory - *last_frame*: last frame to analyze. All the frames *i* < *first_frame* & *i* > *last_frame* are skipped.
i4	Cutoff for molecule-molecule interactions (0 = default 15.0 Å)	**cutoff** The user selects the maximum molecule-molecule distance taken into consideration when interactions are searched.
i5	Is there some confinement? (1 = yes, 0 = no)	**iconf** There is the possibility to read confined simulation trajectories. This parameter controls the periodic boundary conditions. Expected inputs: - 0: unconfined trajectory - 1: confined trajectory

	[if confinement is active (iconf=1)]	
i5.1	Insert confining planes XY, XZ, YZ (1 = on, 0 = off (default))	**iplane(1) iplane(2) iplane(3)** a triad of integers defines if a barrier is placed along the indicated directions. - Nanolayers: “1 0 0” or any permutation; - Nanotubes: “1 1 0” or any permutation; - Nanocavities: “1 1 1”

	[if geometric mode is active:]	
i6	-- Donor-Acceptors atoms (D-H ... A) --	
i7	How many Donor D-H couples?	**ndonors** number of D-H donor atom pairs the user wants to insert
i7.1	[*for i=1,ndonors*] Donors #i, D and H =	**idonors(i,2) idonors(i,1)** indexes of the D-H atoms (according to the topology list). The program asks for the same information *ndonors* time
i8	How many Acceptor A atoms?	**nacceptors** number of A acceptor atoms the user wants to insert
i8.1	[*for i=1,nacceptors*] Acceptor #i, A =	**iacceptors(i)** indexes of the A atoms (according to the topology). The program asks for the same information *nacceptors* time
i9	Insert damping for H-bonds distance (0 = default 0.90)	**tolerance** the scaling parameter used to scale the sum of the van der Waals radii when geometric criteria are used to detect interaction. Interactions between molecules occur when $d_{H-A} \leq tolerance*(r_{vdw,H} + r_{vdw,A}))$
i10	Insert threshold for D-H---X angle (0 = default 120 deg)	**anglehb**

		the threshold for the angle criterion. Interaction between molecules occur when $a_{D\text{-}H\cdots A} \leq anghehb$

	[if energetic mode is active]	
i6	Insert interaction energy reference (in kJ/mol)	**eneref** read the energetic threshold. Two molecules *i* and *j* belong to the same cluster if $E_{ij} \leq eneref$

Here are three examples for the *clusters* steering parameters: (1) geometry mode, LJC force field, no confinement; (2) geometry mode, CLP force field with confinement on XZ plane; (3) energy mode, LJC force field, with no confinement. For the geometry mode, the selected frame range is 100-200, the cutoff is 15 Å with a single D-H pair of atoms (indexed 14 and 15) and two possible acceptors (13 and 14). For the energy mode, the threshold value in Example 3 is set to -25 kJ/mol, while the frame range and cutoff are the same as the previous examples.

Example 1	Example 2	Example 3

1
1
100 200
15
0
1
14 15
2
13
14
0.9
120
	1
	0
	100 200
	15
	1
	0 1 0
	1
	14 15
	2
	13
	14
	0.9
	120
		2
		1
		100 200
		15
		0
		-25.0

The output files are organized as follows:

cluster_ene.*pri* or **cluster_geo.*pri***

This is the main output file of the cluster program. It contains all the clusters found frame-by-frame, with some valuable information about energy, shape and internal symmetry.
The output format and meaning is the following:

```
#1    #2   #3           #4           #5           #6          #7             #8             #9             #10  #11  #12       #13  #14
frame 100
100   1    -130.336506  -96.827425   -127.363919  0.000000    4067.112110    3829.274731    282.317075     F    L    0.187076  2    1 361
100   2    -107.644271  -102.728180  -126.663329  0.000000    24291.858917   19257.277544   10918.135224   F    L    0.881904  6    2  13  23  371  373  382
100   3    0.000000     -122.094045  -127.839255  0.000000    510.299627     413.363036     127.929495     F    X    0.000000  1    3
```

The columns have these meanings:

#1: index of the current frame. This value ranges from *first_frame* to *last_frame*, as indicated in the input.

#2: progressive index of the cluster. Each frame has a different number of clusters, but the ordering index always start from 1.

#3: cluster internal cohesive energy, normalized by the number of molecules belonging to the cluster. This value is the interaction energy between the molecules belonging to the cluster. In the first line of the example above, the third column reports the normalized interaction energy between molecule no. 1 and 361. To note that for cluster no. 3, this energy is null. This is because the third cluster is composed by one isolated molecule.

#4: cluster-environment cohesive energy, normalized for the number of molecules forming the cluster. This is the interaction energy between the molecules belonging to the cluster and all the surrounding molecules that belongs to the environment.

#5: environment cohesive energy, normalized for the number of molecules not belonging to the cluster.

#6: kinetic energy of the cluster. Currently not active.

#7-9: eigenvalues of the inertia tensor. These values give an idea of the shape and extension of the cluster in the three dimensions. If all the three values are similar, the cluster has a spherical-like shape, while asymmetric values indicate oblate or prolate forms.

#10: Flag that signals whether the cluster is infinite, that is, whether it connects two opposite boundaries of the simulation box. If True, the cluster is repeated indefinitely by periodic boundary conditions and has as an infinite number of molecules. If False, the cluster is finite.

#11: Flag for cluster type. L: linear, C: cyclic, M: mixed linear and cyclic branches, X: not applicable. By default, cyclic dimers are considered linear in our program. To be flagged as "C", cluster must be cyclic and contain more than 2 molecules.

#12: Global asymmetry index G, as proposed by Gavezzotti and Lo Presti (New J. Chem., 2019,43, 2077-2084). Please refer to this paper for numerical and physical interpretation.

#13: No. of molecules forming the cluster.

#14: List of molecule indexes forming the cluster.

histene.*pri*

This file contains information on the number of molecules that have intermolecular interaction equal or below the given values. It gives valuable information on the interaction energy distribution.
The output format and meaning is the following:

```
#1   #2
1.0  58906
2.0  26616
3.0  9980
4.0  5284
5.0  3351
```

The first column reports the interaction energy bin value, in kJ/mol. Each energy bin takes the reported value as the maximum, while the minimum of the range is the energy value of the previous line. As an example, the bin *7.0* includes all those molecules that have $6.0 < E_{ij} \leq 7.0$ *kJ/mol*. The second column report the number of molecules that satisfy that requirement.

breaking.*pri*

This file gives information on the average lifetime of the interactions.
The output format and meaning is the following:

```
#1  #2
1   199
2   84
3   45
4   27
5   22
```

The first column indicates the molecule-molecule interaction persistence in terms of number of frames. The second column reports the number of molecule-molecule interactions that are broken as a function of frame persistence.

8.15. The *Conta* module

Conta reads a cluster output file produced by *Clusters.f90* (see 8.14) and computes some key quantities of the various independent clusters. All individual clusters with the desired size are organized in a table and their lifetimes are computed.

Running command:

run.conta name1

“name1” is the name of the main *Clusters* output (*name3*_xxx.pri, xxx=geo or xxx=ene, Section 8.14), without extension. The program produces three files: name1_timespan.*pri* lists all the lifetimes of the detected independent clusters; name1_dimension.*pri* lists the corresponding size; and name1_ordered.*pri* produces the full data, organized by independent clusters.

run.denflu module (unix/linux)

```
rm $1_timespan.out
rm $1_dimension.out
rm $1_ordered.out
cp $1.pri clusters.out
~/programs/MiCMoS/exe/conta
mv timespan.out $1_timespan.pri
mv dimension.out $1_dimension.pri
mv ordered.out $1_ordered.pri
rm clusters.out
```

The program prompts the user to give the following information:

nmin, nmax	nmin, nmax: minimum and maximum cluster size to analyze. Obviously, nmax > nmin is required. nmax = nmin is allowed, though, to analyze only a specific cluster size.
ntol	This is the frame tolerance to consider a cluster as “persistent”. A cluster persists if and only if is found without compositional changes in frames N and N+ntol at least. The lifetime of the cluster is determined by counting how many frames host a bound cluster, within the ntol tolerance. Thus, if for example ntol = 2, the cluster is considered as bound even though it is found in frame N and not in N+1, provided that is found again in frame N+2 at least. In other words, an interruption of 1 frame in its lifetime is ignored. Similarly, ntol = 2 allows to ignore a 2 frames long interruption to define the cluster as “persistent”, and so on. This parameter can be used to get rid of noise, especially when you are using a very short timestep and a high sampling frequency in your trajectory. However, unless you know exactly what you are doing, it is safer to put ntol = 1 in most cases.
delt	Gives the time span of 1 frame, in ps. You can retrieve this quantity by checking your timestep in the MD output, and by multiplying it by your sampling / writing frequency. For example, if you have time step of

0.002 ps (=1 fs) and you write a frame every 500 steps, you should specify delt = 0.002 · 500 = 1.0 ps here.

The output files are organized as follows.

dimension.pri

```
1       3000        17          2
2       3002        17          1
3       3003        17          1
4       3004        17          1
```

Here, the first and second columns give the order number of the frame. In this example, the first frame analyzed corresponds to the #3000 one in the original trajectory, the second one to #3002, and so on. The third column specifies the cluster size that is being analysed (N = 17 molecules in the present case), and the last one the number of clusters of that size that are found in the corresponding frame. In this example, the first frame contains 2 clusters with 17 molecules, while in the subsequent frames the number of clusters with that size reduces to 1.

timespan.pri

```
Frame,n(cluster),Dimension,molecules:   11    1    4      0.500      0.125      0.125    2   31   82   94
Frame,n(cluster),Dimension,molecules:   11    2    4    133.500     23.950      7.845  142  160  184  187
```

This output summarizes the timespan of independent clusters. The first entry is the order number of the frame that is being analyzed; the second one is the order number of the cluster in that frame. the third column expresses the cluster size, *i.e.*, the number of molecules it contains. The fourth entry is the maximum time span of that cluster (in ps). The next two numbers are the corresponding mean timespan, with its estimated standard deviation of the mean. Finally, the order number of molecules in each cluster are given.
In this example, the frame #11 contains 2 clusters made of 4 molecules. The first has a maximum lifetime of 0.5 ps, and an average lifetime of 0.1(1) ps. The second one is much more persistent, with a maximum lifetime of 133.5 ps and an average lifetime of 24(8) ps.

ordered.pri

```
Frame,n(cluster),Dimension,molecules:    5    3   16    2   21   62   74   85   95   96  123  146  148  158  161  208  260  263  338
     Frame        N        Euu        Euv        Evv        Ekin         A            B            C            symm        Ray        DEexc       time
     3000.        2.    -54.363    -65.576    -84.776     61.881  319486.688  298706.344   49379.848       0.958       0.846      11.213      0.000
     3014.        2.    -54.457    -60.168    -84.661     67.411  316168.312  300312.938   44257.133       0.990       0.883       5.710      0.000
```

This file is organized in blocks. Each block specifies an independent cluster (first row). The first two numbers are the order number of the frame (5 in this case) and the order number of the cluster (3 in this case). Thus, here we are looking at the third cluster in the fifth frame. The next entry is the cluster size (16 molecules), followed by the molecule ID's of its building blocks. The next line specifies the energy, symmetry and mechanical parameter of the cluster in the first row, as they evolve in those frames where it is present. Euu is the molecule-molecule energy within the cluster; E_{uv} the interaction energy with the surrounding molecules and E_{vv} the contribution of surrounding molecules only. Ekin is the kinetic energy of the cluster, while A, B and C the corresponding eigenvalues of the inertia tensor. Symm is the symmetry indicator by Gavezzotti & Lo Presti (New J. Chem., 2019,43, 2077-2084), Ray is the Ray's asymmetry parameter for the rotational dynamics of the cluster (B. S. Ray, "*Über die Eigenwerte des asymmetrischen Kreisels*," Zeitschrift für Physik, vol. 78, no. 1-2, pp. 74–91, 1932) and ΔE_{exc} is the cohesion excess energy with respect to the surrounding liquid (Sironi, Macetti, Lo Presti, *submitted*).

Finally, time is the cumulative surviving time of the cluster, whose estimation depends also on the *ntol* and *delt* parameters described above.

PART C

Appendix: Reference Materials and Technical Details

A1. The *Retcif* procedure: atom type recognition and assignment of atom type codes

The ***Retcif*** module extracts crystallographic data from a Crystallographic Information File (*.cif* file). See Section 1.1 for operational information. This Appendix explains in detail how the algorithm works.

A1.1 Determination of average bond distances and average bonding radii for various atomic species.

Key to ***Retcif*** operation is a recognition of atomic connectivity. Atomic species are recognized from the symbol in the *.cif* file, but then different subspecies are identified. For that purpose, a survey of about 18000 crystal structures in the CSD has been made to determine average bond distances and ranges (Table A1.1), from which atomic bonding radii could be derived. An atom pair is considered bound if the distance is below the sum of the bonding radii allowing for some tolerance.

Table A1.1

Average bond lengths and variability ranges from about 18000 crystal structures in the CSD. These parameters are used by ***Retcif*** to determine average bonding radii. See Table 1.1 in the main text for atomic species code numbers. C_{ar} stands for aromatic carbon, a question mark "?" means any atom sub-species and "–" denotes unavailable values due to poor statistics.

Bond	**Atomic species code numbers (Table 1.1)**	**Min–Max / Å**	**Average / Å**
–C≡C–	C11–C11	1.11–1.23	1.208
≡C–C<	C11–C12	1.39–1.50	1.433
≡C–C<	C11–C13	1.39–1.53	1.474
>C=C<	C12–C12	1.25–1.45	1.390
>C–C<	C12–C13	1.40–1.60	1.510
>C–C<	C13–C13	1.40–1.64	1.530
C_{ar}–C_{ar}	C14–C14	1.34–1.50	1.428
>C–C_{ar}	C12–C14	–	1.417
≡C–C≡	C11–C11	1.35–1.39	–
>C–C<	C12–C12	1.45–1.54	–
C–F	C?–F41	1.27–1.42	–
C–Cl	C?–Cl42	1.70–1.85	–
C–Br	C?–Br43	1.90–2.00	–
C–I	C?–F44	2.10–2.20	–
=C–O–	C12–O23	1.30–1.45	1.373
>C–O–	C13–O23	1.32–1.54	1.439
–O–O–	O23–O23	1.45–1.52	1.475
–C=O	C13–O27	1.14–1.30	1.216
>C–O(H)	C13–O29	1.34–1.48	1.424
>C–O(H)	C12–O29	1.30–1.40	1.356
>C–N^+	C13–N16	1.45–1.55	–
>C–NR_4	C13–N17	1.40–1.54	–
>C–N=	C12–N18	1.22–1.44	–
>C–NO_2	C12–N20	1.40–1.52	–
–N–O(–)	N?–O30	1.18–1.28	–
–C≡N	C11–N19	1.10–1.18	–
>C–S	C13–S?	1.70–1.90	–

⋗C–S	C12–S?	–	1.80

Distances involving H atoms have been retrieved from experimental neutron diffraction estimates (Table A1.2).

Table A1.2
Average bond lengths for hydrogen atoms, as retrieved from neutron diffraction experiments.

Group	**Chemical specie**	**Average / Å**	**Nº structures**
(R)–O–H	Alcohols	0.970	35
(CO)–O–H	Acids	1.000	7
(CO)–N–H	Amides	1.014	18
>N–H	Amines	1.020	11

Covalent bonding radii estimated from parameters in Table A1.1 and Table A1.2 are shown in Table A1.3 and are loaded into the Block Data `Alldat.for` (double precision) and `Alldas.for` (single precision).

Table A1.3
Estimated covalent bond radii for specific atomic species in ***Retcif***.

Atom type	**Atomic species code numbers (Table 1.1)**	**Covalent radius / Å**
H	1–9	0.30
=C–	10, 12, 14	0.70
≡C–	11	0.60
⋗C–	13	0.77
F	41	0.65
Cl	42	1.00
Br	43	1.20
I	44	1.35
P	45	1.05
–O–	23, 24, 28	0.68
=O	27	0.55
–O(H)	29	0.68
N=O	30	0.50
P=O	32	0.40
S=O	31	0.40
S	34, 35, 36, 37	1.05
$(R_nH_{3-n})N^+$	16	0.75
$(R_nH_{3-n})N$	17	0.75
>N–N<	17	0.75
=N–	18	0.65
≡N	19	0.55
$(O)_2N$–	20	0.72
(CO)N– (Amide)	21	0.70

Retcif assumes that two atoms at distance R_{ij} with covalent radii R_i and R_j, are bonded if $\Delta = \left|R_{ij} - \left(R_i + R_j\right)\right| < 0.35$.

A1.2 Stage 1: Reading structure

An input *.cif* (Crystal Information File) file is read to retrieve the Cambridge Structural Database (CSD) refcode (if present), space group symbol and symmetry operations, brute formula, cell parameters, number of chemical units, atom type and coordinates for all atoms. The atomic coordinate list is rearranged with non-hydrogens first, the connectivity matrix is computed using the covalent radii, separate chemical units are recognized and each atom is assigned to one of them. Hydrogen atom assignment and renormalization is performed (see below). The final atom count is then compared with the retrieved brute formula; if the count is incorrect (some hydrogen coordinates not present) an error flag is hoisted. Other error conditions arise from the many inconsistencies that may be present in *cif* files, mostly from valence errors or missing essential information (see below).

A1.3 Stage 2: Scan non-hydrogen atoms

There are separate procedures for each atom type (see also Figures A1.1–A1.2).

Carbon atoms. ***Retcif*** work if original H-atom coordinates are present or missing. In the latter case, the number of hydrogens attached to each C-atom is guessed from standard valence considerations. Xray H-atom positions are anyway discarded unless the forced retrieval option is adopted.

See Figure A1.1 for the flow diagram followed by the algorithm. Hydrogen atom assignment is considered for C≡CH acetylenic, C=CH_2 terminal methylene, R-CH_2-R saturated methylene, R=CH-R unsaturated or aromatic, R_3CH methine, R-CH_3 methyl, R being any non-hydrogen atom compatible with the rules of valence. No action is of course required for quaternary R_4C carbons. All C–H distances are renormalized to 1.08 Å; then: a) acetylenic H is located along the C-C bond direction; b) terminal methylene H's are located assuming a planar configuration at the double bond, all angles 120°; c) methylene H's are located in a plane perpendicular to the RCR plane and bisecting the RCR angle, with a tetrahedral configuration; d) unsaturated H's are located in the RCR plane on the bisector of the RCR angle; e) methine H's are located assuming a C-H direction from the center of coordinates of the three basal atoms to the apical atom; f) methyl atoms are located with tetrahedral CCH angles, as close as possible to the original set of R'-R-C-H torsion angles, if available from the cif file; otherwise, staggered configurations are assumed.

The problem is the blindfold determination of which of the above is the case for each C atom in a molecule. If the *.cif* atom count is correct, the problem is solved by using the *.cif* connectivity to determine the type and number of hydrogen atoms to be assigned to the carbon atom under examination (Figure A1.1). Otherwise the approximate valence saturation, *V*, is calculated on the basis of average C-R bond distances, the number of hydrogen atoms to be assigned being then 4-*V*, with appropriate roundoff. This last step is obviously not 100% safe, due to valence vagaries, especially when R is nitrogen, and to errors or misinterpretations in the original X-ray coordinates.

Nitrogen atoms. See Figure A1.2 for a schematic description of the algorithm. Since various degrees of pyramidalization are possible, the ab initio prediction of missing H-atom locations is impossible. Structures are accepted only if approximate positions for the H atoms are available in the *.cif* files, which is the case in recent X-ray structure determinations. The *.cif* connectivity is then used and an error flag is set if the atom count does not match (as above). Quaternary nitrogen $R_nN^+H_{4-n}$ is assigned 4-*n* H atoms. No action is taken for R_3N groups (this causes an error if the nitrogen atom is actually quaternary and H positions are not in the original *.cif* file). Nitrogen is H-bond acceptor in R-NH-R or RNH_2 groups; amide hydrogens are recognized and marked as H-bond donors. No action is taken when the R=N-R connectivity is detected, as distinguished from the R-NH-R connectivity based on the presence of H-atom coordinates in the original *cif* file. Sometimes a R-N-R connectivity is detected on the basis of R-N bond distances, even in absence of H-atom coordinates; in such case, a H atom is assigned on the bisector of the RNR angle. This identification is sometimes uncertain. In the two latter configurations, nitrogen is considered as a H-bond acceptor and hydrogen as a H-bond donor. One hydrogen is assigned for the R=N-H terminal connectivity. Terminal nitrogen atoms are not H-bond acceptors. N-H distances are renormalized to 1.00 Å (1.03 Å for quaternary ammonium ions).

Oxygen atoms. The considerations made for nitrogen on the a priori predictability of H-atom positions hold also for oxygen; in this case too, only the connectivity retrieved from the original *.cif* files is used for H-atom assignment. No action is taken for the R-O-R connectivity, and oxygen is considered not to act as H-bond acceptor (sometimes a questionable choice). Alcohol and acid functions are recognized and marked as H-bond donor and acceptor groups. C=O, N=O, S=O and P=O oxygen is always considered as H-bond acceptor. COO^- (ester) groups are recognized and assigned no hydrogen; except for the case of zwitterions (e.g. aminoacids and peptides) the carboxyl group is often a source of error because in many carboxylic acid crystals, there is disorder in the H-atom positions and C-O and C=O distances are apparently very similar. There is no possible (automatic) solution for this problem, thus input and output file must be carefully checked to avoid mistakes. S-OH and P-OH hydrogens are taken care of as alcohol hydrogens. All O-H distances are renormalized to 1.00 Å. Oxygen atoms not bound to any other atom are considered as water oxygens.

Table A1.4
Assignment of atomic species code numbers for O atoms, according to Table 1.1 in the main text. A stands for "any atom".

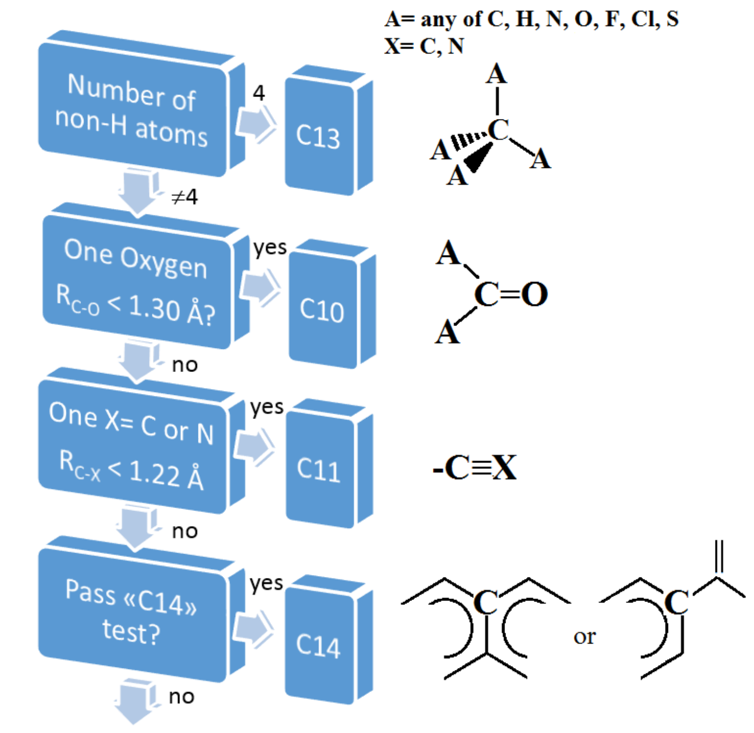
Specie	Atomic specie code number for O	Notes
C=O	27	Ketones, aldehydes, acids, amides, COO^- groups (keto oxygen). R_{CO} must be < 1.30 Å
N=O	30	Nitro or nitroso group
S=O	31	Sulfone or sulfoxide
A-O-A	23	Ether
A-O-H	29	Alcohol
H-O-H	24	Water
(O=C)–O–A	28	Acids and esters, single–bonded oxygen

Other atomic species. S-H thiol hydrogens are assigned as for O-H hydrogens preserving RSH angles and with S-H = 1.30 Å. No provision is made for hydrogen atoms attached to atoms other than C, O, N and SH, and an error message is generated when this happens, as well as in cases with erroneous, improper, or just unusual bonding situations.

Stage 3: Final checks. These mainly concern large databases retrieved from the Cambridge Structural Database, and seldom to single *.cif* files prepared by the user. In Stage 3, the total atom count after the H-atom assignment procedures is matched with the brute formula retrieved from the *.cif*, generating error flags when appropriate (see Section 1.1). In spite of this and all other sieves, the whole procedure is not absolutely safe, because of many combinations of casual errors and of their accidental cancellation. When retrieving from the CSD, a "statistical noise" of the order of 1-3% of wrong structures passes the checks and is introduced in the retrieved samples.

CAUTION: Errors at this stage mainly concern items retrieved from the Cambridge Structural Database, and seldom to single *.cif* files prepared by the user, in which case a good practice is to prepare directly a *.oih* file with hydrogen atom codes as necessary, skipping the ***Retcif*** stages.

Figure A1.1. Flow diagram for the assignment of atom types: Carbon atoms. Id numbers indicated in the Figure correspond to atomic species code numbers in Table 1.1 (main text). "C14 test" stands for the check against aromatic C atoms not bearing hydrogens: it is passed if the C atom bears 3 bonds, has neither H nor O attached atoms and does not belong to a nitrile group.


A= any of C, H, N, O, F, Cl, S
X= C, N
Number of non-H atoms
4
C13
≠4
One Oxygen $R_{C\text{-}O}$ < 1.30 Å?
yes
C10
no
One X= C or N $R_{C\text{-}X}$ < 1.22 Å
yes
C11
no
Pass «C14» test?
yes
C14
no
C=O
-C≡X
or

NVALT: number of H-atoms attached
NATTC: number of non-H attached

NVALT	NATTC	Atom type
1	1	**C11**
1	2	**C12**
1	3	**C13**
2	1	**C12**
2	2	**C13**
3	(1)	**C13**

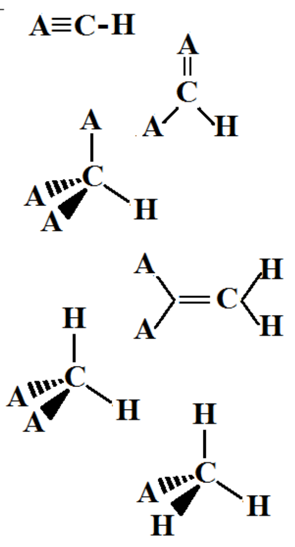
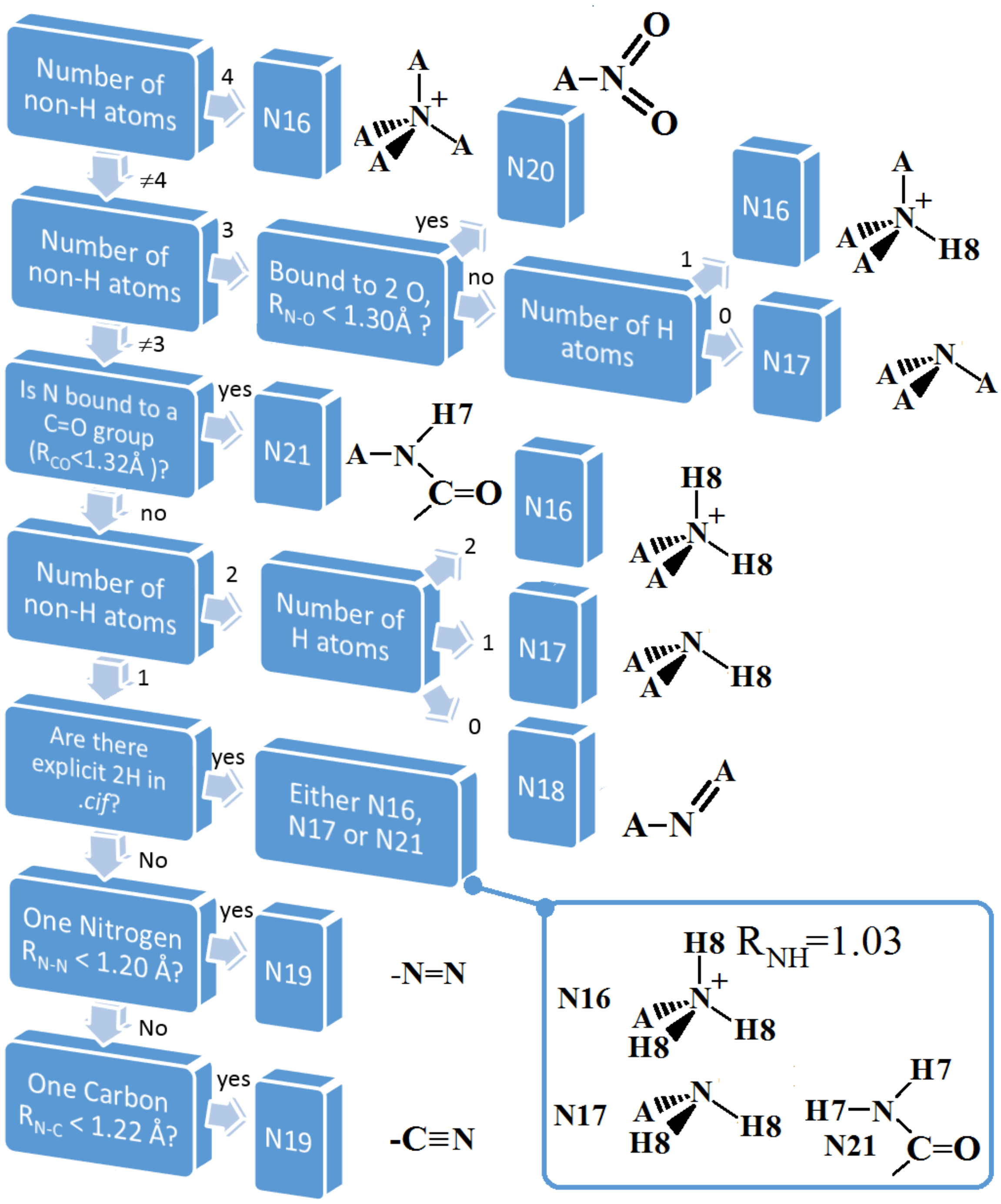


Figure A.1.2. Flow diagram for the assignment of nitrogen of atom types. Symbols have the same meaning as in Figure A1.1.



The ***Retcif*** procedure produces an intermediate file, with extension *.oih*, generated with only the un-flagged crystals structures, unless an explicit override command is given, in which case the structural information is anyway written on file, ready for perusal and manual correction. A total override option is also available (Section 1.1), in which atomic labels (*e.g.*, hydrogen bond acceptor or not, etc.) are issued, but the original H-atom coordinates are written on the *.oih* file without any attempt at hydrogen reassignment or normalization. This option is useful for neutron crystal structures. Deuterium is treated as hydrogen.

The *.oih* file contains all independent coordinates for non-hydrogen atoms, always including entire molecules if Z'<1 (Section 1.4.3). Hydrogen atom positions are represented in the form of sets of numerical codes that specify the connectivity and the desired bond lengths, bond angles, etc. These codes can be changed by the user by manually editing the files. In addition, the *.oih* format allows the introduction atoms other than hydrogens in desired position upon intevention by the user. A subsequent routine (***Retcor*** module, Section 1.2 in the main text and A2 in the Appendix) reads the *.oih* files and generates the extension *.oeh* files in which full H-atom *x, y, z* coordinates are stored. *oeh*-type files branch into all modules of the CLP package for crystal structure analysis, atom-atom lattice energies, crystal structure generation, and for the PIXEL calculations modules.

A2. The *Retcor* module

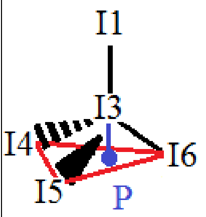
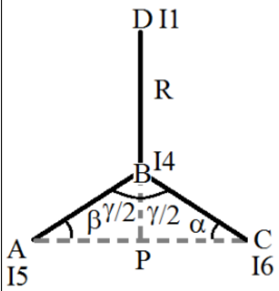
Refer to the main text (Section 1.2) for a general description of how to operate this module. The following diagrams show the geometrical procedures employed to generate new atoms from the coordinates of atoms present. These procedures allow the preparation of a molecular model from a minimum of 3 atoms whose coordinates are specified. The program generates coordinates sequentially, so that care must be taken to build new atoms only when the coordinates of atoms from which they depend have already been generated.

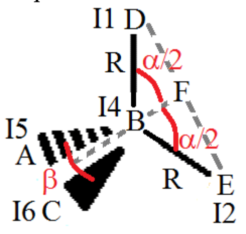
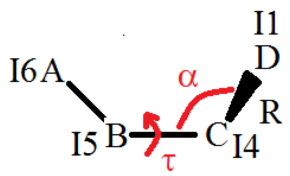
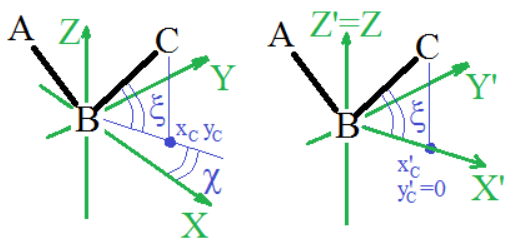
In crystal structure analysis and lattice energy calculations, these procedures are used exclusively to generate standard coordinates for hydrogen atoms whose X-ray positions are unreliable.

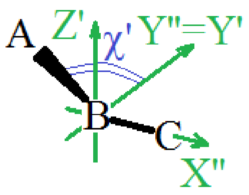
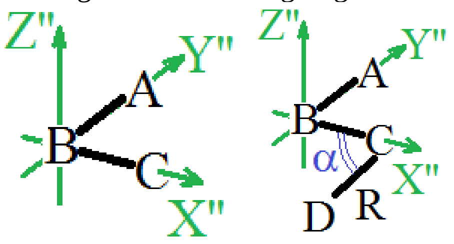
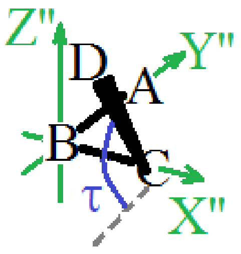
In the *.oih* file produced by ***Retcif***, 6 integer codes must be specified, called placement code lines (n_1, n_2, n_3, n_4, n_5, n_6: see Section 1.4.3, NHYD indicator, lines id 7–8 and following). They specify how new atoms are constructed from the available coordinates. Some integers in the sequence n_1, n_2, n_3, n_4, n_5, n_6 must be sequential atom id numbers, while others may be path indicators. The main options have been already presented in Figure 1.2; the full list of available options, together with detailed a description of the various algorithms, is shown in Table A2.1.

Some geometrical parameters must be also specified just after the placement code, that is, MLC, ISPEN, QRG, R, TORS, ALPH (Section 1.4.3, lines id 8ff). MLC is the number of the chemical fragment to which the new atom belongs; ISPEN the corresponding atomic species indicator (Table 1.1); QRG the estimated atom point charge (zero if further calculation by ***Retcha***, Section 1.3, is required); and R, TORS and ALPH are distance and angle parameters that define the new group (Figure 1.2 and Table A2.1). Table A2.1 specifies, for each case, what entries in the placement code are nonzero and what geometrical parameters in the R, TORS and ALPH sequence must be specified accordingly.

Table A2.1
Options for molecular reconstruction in ***Retcor***. Sets of six integers n1-n6, needed values of conformation parameters (distance, angle, torsion).

Operation	**Placement code sequence**	**R, TORS, ALPH**	**Construction procedure**; boldface entries denote vectors
Reset distance	I1 0 0 I4 0 0	R 0 0	Reset **I1**–**I4** vector length by moving **I1** according to: $\mathbf{I1} = \mathbf{I4} + \frac{R}{R_{old}}\lvert\mathbf{I1} - \mathbf{I4}\rvert$
Build an acetylene–like terminal atom	I1 0 0 I4 I5 0	R 0 0	A new atom I1 is added according to: $\mathrm{I5} - \mathrm{I4} \overset{\mathrm{R}}{-\!\!-\!\!-} \mathrm{I1}$ The new coordinates are: $\mathbf{I1} = \mathbf{I4} + (\mathbf{I4} - \mathbf{I5})\frac{\lvert\mathbf{I1} - \mathbf{I5}\rvert}{\lvert\mathbf{I4} - \mathbf{I5}\rvert}$
Build a “methine” group	I1 0 I3 I4 I5 I6	R 0 0	 A new atom I1 is created at distance R above the I3 apex. **P** is the centre of coordinates of the basal triangle I4-I5-I6.
Build a trigonal atom	I1 0 0 I4 I5 I6	R 0 0	For simplicity, be **A** the point of coordinates of the atom I5, **B** of I4, **C** of I6 and **D** of I1. A new atom I1 is placed at distance R from I4 along the *ABC* bisector. Note that, if *BA* and *BC* have different lengths, angles *DBA* and *DBC* are different as well.  D I1 R B I4 β γ/2 γ/2 α A I5 P C I6 $\sin(\gamma/2) = \sqrt{\frac{1-\cos\gamma}{2}}$; $T = \frac{\sin\beta}{\sin\alpha}$ The point **P** where the *ABC* bisector cuts **AC** is looked for. $AB = \frac{BP}{\sin\beta} = \frac{PA}{\sin(\gamma/2)};\ BP\sin(\gamma/2) = PA\sin\beta$ $BC = \frac{AC-PA}{\sin(\gamma/2)} = \frac{BP}{\sin\alpha};\ (AC-PA)\sin\alpha = BP\sin(\gamma/2)$ $PA = \frac{CA}{T+1}; \mathbf{PA} = \mathbf{CA}\frac{PA}{CA}$ $\mathbf{P} = \mathbf{A} + \mathbf{PA}$ $\mathbf{D} = \mathbf{P} + \mathbf{BP}\frac{DP}{BP}$

Operation	**Placement code sequence**	**R, TORS, ALPH**	**Construction procedure**; boldface entries denote vectors
Build a "methylene" group	I1 I2 0 I4 I5 I6	R 0 ALPH	If the ALPH parameter is set to 0, a default value of 108.0 deg is used. Be **A** the coordinates of the atom I5, **B** of I4, **C** of I6, **D** of I1 and **E** of I2. Two new atoms I1 and I2 are defined at the same distance R from I4, with I1–I4–I2 angle equal to α = ALPH deg.  I1 D R α/2 F I4 B α/2 I5 A β R E I6 C I2 $BF = R\cos\frac{\alpha}{2}$ $DF = R\sin\frac{\alpha}{2}$ A point **F** is defined from points **A**, **B** and **C** with the same procedure used for the trigonal case. The vector **BF** thus lies on the bisector of the *ABC* angle. A vector $\mathbf{P_v}$ is computed, orthogonal to the ABC plane: $\mathbf{P_v} = (\mathbf{B}-\mathbf{A})\wedge(\mathbf{B}-\mathbf{C}) = \det\begin{Vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_{BA} & y_{BA} & z_{BA} \\ x_{BC} & y_{BC} & z_{BC} \end{Vmatrix}$ With module $P_v = \sqrt{P_{vx}^2 + P_{vy}^2 + P_{vz}^2}$. $\mathbf{P_v}$ then defines the vector **DF**, which is orthogonal to the *ABC* plane and has the desired length *DF*, according to $\mathbf{DF} = \frac{\mathbf{P_v}}{P_v}DF$. Finally $\mathbf{D} = \mathbf{F} + \mathbf{DF}$ $\mathbf{E} = \mathbf{F} - \mathbf{DF}$
Build a generic group with "Z–matrix" specs	I1 –1 0 I4 I5 I6	R, TORS, ALPH	Be **A** the coordinates of the atom I6, **B** of I5, **C** of I4 and **D** of I1. A new atom I1 is generated at distance R from I4 and configurational parameters TORS (τ deg) and ALPH (α deg).  1) Origin is set at **B**. Being **C** known, the coordinates x_C and y_C of its projection in the *XY* plane are also known. A clockwise rotation on **Z** by χ deg is carried out, so that $y_C' = 0$.  A Z C Y B ξ x_C y_C χ X A Z'=Z C Y' B ξ x'_C y'_C=0 X' (continue in the next page)

Operation	**Placement code sequence**	**R, TORS, ALPH**	**Construction procedure**; boldface entries denote vectors
Build a generic group with "Z–matrix" specs	I1 –1 0 I4 I5 I6	R, TORS, ALPH	(from previous page) 2) The reference system is now rotated counterclockwise by ξ deg around the axis **Y'**, so that the direction of **BC** coincides with that of the new axis X'', as shown in the next graph. Be now χ' the angle made by the **AB** vector with respect the **Y'** axis.  3) A clockwise rotation by χ' deg is applied around **X''**, so that the point **A** is now orthogonal to the **Z** axis. At the end of this transform, **A**, **B** and **C** lie on the same *XY* plane. The fourth atom **D** is now added in the *XY* plane, at a distance R from **C** and making a α = ALPH deg large *BCD* angle.  Z'' Y'' A B C X'' Z'' Y'' A B C α X'' D R 4) Finally, D is rotated clockwise around X'' (i.e. around the **BC** axis) by τ = TORS deg.  5) Now the reference system is back–rotated to the original one. Being $\mathbf{R}_1(\chi)$, $\mathbf{R}_2(\xi)$ and $\mathbf{R}_3(\chi')$ are the matrices of rotations used to carry out the individual transforms, the overall rotation for steps 1→ 3 can be expressed as: $$\mathbf{R_{tot}} = \mathbf{R_1} \cdot \mathbf{R_2} \cdot \mathbf{R_3}$$ And the inverse transform, $\mathbf{R_{tot}^{-1}}$, can be applied to the atomic coordinates to restore the original axes. 6) The origin is back–translated to the original laboratory reference frame.

A3. Algebra for the generation of crystal coordinates, orthogonalization, inertial reference frame

Three main reference systems are used in MiCMoS: (i) a crystallographic reference frame; (ii) a crystallophysical (Cartesian orthogonal) reference frame; (iii) an internal inertial reference frame.

(i) The crystallographic frame has atomic fractional coordinates ($\mathbf{x}_{\mathrm{FC}}$). This reference system is useful to apply symmetry transformations and lattice translations, *i.e.*, to build the whole simulation box but is impractical for computing distances, angles and forces

(ii) A Cartesian orthogonal reference system, with coordinates in Å units, is sometimes referred to as a "crystallophysical" system. An orthogonalization matrix $\mathbf{O}$ transforms coordinates between (i) and (ii):

$$\mathbf{x}_{\mathrm{OC}} = \mathbf{O} \cdot \mathbf{x}_{\mathrm{FC}}$$
$$\mathbf{x}_{\mathrm{FC}} = \mathbf{O}^{-1} \cdot \mathbf{x}_{\mathrm{OC}}$$

$$\mathbf{O} = \begin{bmatrix} (a) & (b\cos\gamma) & (c\cos\beta) \\ 0 & (b\sin\gamma) & \left(c\dfrac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}\right) \\ 0 & 0 & \left(\dfrac{cf_V}{\sin\gamma}\right) \end{bmatrix}$$

$$f_V = (1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma)^{1/2}$$

$$\mathbf{O}^{-1} = \begin{bmatrix} \left(\dfrac{1}{a}\right) & \left(-\dfrac{\cos\gamma}{a\sin\gamma}\right) & \left(\dfrac{\cos\gamma\,(\cos\alpha - \cos\beta\cos\gamma)}{af_V\sin\gamma} - \dfrac{\cos\beta\sin\gamma}{af_V}\right) \\ 0 & \left(\dfrac{1}{b\sin\gamma}\right) & \left(-\dfrac{\cos\alpha - \cos\beta\cos\gamma}{bf_V\sin\gamma}\right) \\ 0 & 0 & \left(\dfrac{\sin\gamma}{cf_V}\right) \end{bmatrix}$$

Let now $\mathbf{S}_s$, $\mathbf{t}_s$ be a matrix-vector pair representing a symmetry operation within the crystal space group. Calling $\mathbf{x}_{\mathbf{FC},1}$ the vector of coordinates of the reference molecule "1", the coordinates of any other molecule *s* in the crystallographic ($\mathbf{x}_{\mathbf{FC},s}$) and crystallophysical ($\mathbf{x}_{\mathbf{OC},s}$) reference frames can be obtained through the following transforms:

$$\mathbf{x}_{\mathrm{FC},s} = \mathbf{S}_s\mathbf{x}_{\mathrm{FC},1} + \mathbf{t}_s$$
$$\mathbf{x}_{\mathrm{OC},s} = \mathbf{O} \cdot \mathbf{x}_{\mathbf{FC},s} = \mathbf{O} \cdot [\mathbf{S}_s\mathbf{x}_{\mathrm{FC},1} + \mathbf{t}_s] = (\mathbf{O} \cdot \mathbf{S}_s) \cdot \mathbf{x}_{\mathbf{FC},\mathbf{1}} + (\mathbf{O} \cdot \mathbf{t}_s)$$

Cartesian coordinates $\mathbf{x}_{\mathrm{OC},s}$ are used throughout the whole package for all calculations involving distances between atoms *k* and *m* in molecules *i* and *j*, the modules of the corresponding difference vectors :

$$R_{k,m}^{i,j} = \left|\mathbf{x}_{\mathrm{OC},i}^{k} - \mathbf{x}_{\mathrm{OC},j}^{m}\right|$$

(iii) Local (molecular) reference frames are useful for example whenever differently oriented molecules must be compared. When the local reference frames has origin at the molecular centre of mass, orthogonal *X*, *Y*, *Z* axes are rotated to lie along to the three principal moments of inertia, that is, the eigenvectors of the inertial matrix. The local reference frame may instead have its origin in the center of coordinates without any axis rotation.

To find the inertial $\mathbf{x}_{\mathrm{IC}}$ coordinates, the following procedure is applied.

(1) Being w_k the atomic weight (in atomic units) of the kth atom in molecule *i*, molecular centre of mass coordinates $\mathbf{x}_B$ are:

$$\left.\begin{aligned} x_B &= \frac{\sum_k w_k x_k}{\sum_k w_k} \\ y_B &= \frac{\sum_k w_k y_k}{\sum_k w_k} \\ z_B &= \frac{\sum_k w_k z_k}{\sum_k w_k} \end{aligned}\right\}$$

With x_k, y_k and z_k being the crystallophysical Cartesian coordinates of atom k. The center of coordinates is obtained by the same expression with w_k's=1.

(2) Atomic coordinates of every atom k are now referred to the centre of mass:

$$\mathbf{x}_{\mathrm{OC},B}^{k} = \mathbf{x}_{\mathrm{OC}}^{k} - \mathbf{x}_B$$

(3) The symmetric inertial matrix **I** is computed using orthogonal coordinates with origin in $\mathbf{x}_B$:

$$\mathbf{I} = \begin{bmatrix} \sum_k (w_k y_k^2 + w_k z_k^2) & -\sum_k w_k x_k y_k & -\sum_k w_k x_k z_k \\ -\sum_k w_k x_k y_k & \sum_k (w_k x_k^2 + w_k z_k^2) & -\sum_k w_k y_k z_k \\ -\sum_k w_k x_k z_k & -\sum_k w_k y_k z_k & \sum_k (w_k x_k^2 + w_k y_k^2) \end{bmatrix}$$

Diagonalization of **I** gives the main moments of inertia (eigenvalues) and three inertial eigenvectors. These are stored as column vectors into a matrix **W** to rotate all the orthogonal atomic coordinates $\mathbf{x}_{\mathrm{OC},B}^{k}$ into the inertial frame:

$$\mathbf{x}_{\mathrm{IC}}^{k} = \mathbf{W} \cdot \mathbf{x}_{\mathrm{OC},B}^{k}$$

The matrix **W** is unitary, that is $\mathbf{W}^{-1} = \widetilde{\mathbf{W}}$. Its eigenvalues are the same for any symmetry transformation on the $\mathbf{x}_{\mathrm{IC}}^{k}$ coordinates.

A4. Coordinate systems and transformation matrices in PIXEL

This procedure produces the translation–related replicas of molecules in the unit cell to evaluate the lattice energy by the PIXEL method (See also Section 3.2 in the main text). The $\mathbf{M}_1$, $\mathbf{M}_2$ matrices and $\mathbf{t}_1$, $\mathbf{t}_2$ vectors must be specified in the PIXEL input file (Section 3.2.7) to transform coordinates from the standard orientation of GAUSSIAN into the PIXEL reference frame. If the option *Nosym* is employed in the GAUSSIAN input (recommended), the molecular coordinate system is not re–oriented and $\mathbf{M}_1$ = unit matrix, $\mathbf{t}_1$ = [0 0 0] (otherwise, they must be found by inspection of the atomic coordinates by a difficult manual inspection, seldom if ever necessary). $\mathbf{M}_2/\mathbf{t}_2$ transform the molecular reference system into the actual PIXEL reference. $\mathbf{M}_2$, $\mathbf{t}_2$ are automatically computed by ***Pixmt2*** (see also Section 3.2.5).

Let $\mathbf{x}_{\mathrm{LG}}$ be the collection of the atomic coordinates in the standard reference frame (GAUSSIAN), and $\mathbf{x}_{\mathrm{LO}}$ the same in any other user-defined reference, *e.g.* the inertial reference frame, with origin at the molecular center of mass. Let $\mathbf{x}°_{\mathrm{FC}}$ be the fractional coordinates of the reference molecule (unit cell reference system), and $\mathbf{x}°_{\mathrm{OC}}$ the corresponding orthogonalized coordinates:

$$\left.\begin{array}{c}\mathbf{x}_{\mathrm{LO}} = \mathbf{M}_1\mathbf{x}_{\mathrm{LG}} + \mathbf{t}_1 \\ \mathbf{x}^0_{\mathrm{OC}} = \mathbf{M}_2\mathbf{x}_{\mathrm{LO}} + \mathbf{t}_2 \\ \mathbf{x}^0_{\mathrm{OC}} = \mathbf{O}\cdot\mathbf{x}^0_{\mathrm{FC}} \\ \mathbf{x}^0_{\mathrm{FC}} = \mathbf{O}^{-1}\cdot\mathbf{x}^0_{\mathrm{OC}}\end{array}\right\}$$

where $\mathbf{O}$ is an orthogonalization matrix (Appendix, Section A3). The $\mathbf{M}_2$, $\mathbf{t}_2$ pair are calculated by module ***Pixmt2*** and are automatically inserted into the PIXEL input file.

Let now $\mathbf{S}_s$, $\mathbf{t}_s$ be a matrix-vector pair representing a given symmetry operation within the crystal space group. Then:

$$\left.\begin{array}{c}\mathbf{x}^s_{\mathrm{FC}} = \mathbf{S}_s\cdot\mathbf{x}^0_{\mathrm{FC}} + \mathbf{t}_s \\ \mathbf{x}^s_{\mathrm{OC}} = \mathbf{O}\cdot\mathbf{x}^s_{\mathrm{FC}}\end{array}\right\}$$

The final goal is an expression for the orthogonal coordinates for the *s*-th molecule in the molecular cluster that represents the crystal, $\mathbf{x}^s_{\mathrm{OC}}$, in terms of the coordinates in the standard molecular reference system, $\mathbf{x}_{\mathrm{LG}}$. This transformation can be carried out as follows:

$$\mathbf{x}^s_{\mathrm{OC}} = \mathbf{\Omega}^s\cdot\mathbf{x}_{\mathrm{LG}} + \boldsymbol{\omega}^s$$

where a little algebra shows that the $\mathbf{\Omega}^s$ matrix can be computed as:

$$\mathbf{\Omega}^s = \mathbf{O}\cdot\mathbf{S}_s\cdot\mathbf{O}^{-1}\cdot\mathbf{M}_2\cdot\mathbf{M}_1$$

and the $\boldsymbol{\omega}^s$ vector:

$$\boldsymbol{\omega}^s = \left(\mathbf{O}\cdot\mathbf{S}_s\cdot\mathbf{O}^{-1}\right)\cdot(\mathbf{M}_2\cdot\mathbf{t}_1 + \mathbf{t}_2) + \mathbf{O}\cdot\mathbf{t}_s$$

In practice, a number of molecules surrounding the reference one in the crystal are generated by adding integer cell translations to the components of the $\mathbf{t}_s$ vectors. Both atomic coordinates and coordinates of the *e*-pixels are transformed accordingly. This method allows the packing of any molecular object specified by $\mathbf{x}_{\mathrm{LG}}$ coordinates, with location and orientation in the unit cell specified by vector $\mathbf{t}_2$ and matrix $\mathbf{M}_2$, respectively, into a crystal specified by cell dimensions (matrix **O**) and space group ($\mathbf{S}_s$, $\mathbf{t}_s$ pairs).

As an added bonus, if matrix $\mathbf{M}_2$ is interpreted as a rotation matrix, the orientation angles for the molecule in the crystal with respect to the local reference frame can be derived (see Section A6). These could then be used as parameters in the rigid-body lattice energy minimization.

A5. Structure of the *.den* density file

This is the GAUSSIAN "cube" format; each entry a record.

• Title	
• n_{atom}, x_{min}, y_{min}, z_{min}	number of atoms and min values of the coordinates of the density cube
• n_{x}, d_{xx}, d_{xy}, d_{xz}	number of points along x, components of the x step vector
• n_{y}, d_{yx}, d_{yy}, d_{yz}	same, for y
• n_{z}, d_{zx}, d_{zy}, d_{zz}	same, for z
• For each atom: Z, q, x, y, z	atomic number, number of electrons, x, y, z coordinates

All the above lines are written in format: i5,4f12

Then, n_{x} times n_{y} lines, each n_{z} long: (den(k),k=1,nz) density points, format 6e13.6. Only the d_{xx}, d_{yy}, d_{zz} terms have meaning. Each pixel point in the density grid has coordinates of $x_{\mathrm{min}}+n\cdot d_{\mathrm{xx}}$, $y_{\mathrm{min}}+m\cdot d_{\mathrm{yy}}$, $z_{\mathrm{min}}+p\cdot d_{\mathrm{zz}}$, with n, m and p integers. All quantities are in atomic units and the unit length is 1 bohr.

A6. Definition of Euler angles θ, φ, χ in rotation matrix E (*Boxcry* module)

Boxcry produces a *.bxi* file format (Section 5.1.1) that specifies, for each molecule in the simulation box, three Euler rotation angles θ, φ, χ. The overall rotation is the product of three rotations; for clarity, $\mathrm{c}\theta$, $\mathrm{c}\varphi$, $\mathrm{c}\chi$ are abbreviated forms for $\cos\theta$, $\cos\varphi$, $\cos\chi$, and $\mathrm{s}\theta$, $\mathrm{s}\varphi$, $\mathrm{s}\chi$ the corresponding forms for $\sin\theta$, $\sin\varphi$, $\sin\chi$.

$$\mathbf{E} = \begin{bmatrix} \mathrm{c}\chi & -\mathrm{s}\chi & 0 \\ \mathrm{s}\chi & \mathrm{c}\chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{c}\varphi & 0 & \mathrm{s}\varphi \\ 0 & 1 & 0 \\ -\mathrm{s}\varphi & 0 & \mathrm{c}\varphi \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\vartheta & -\mathrm{s}\vartheta \\ 0 & \mathrm{s}\vartheta & \mathrm{c}\vartheta \end{bmatrix} = \begin{bmatrix} \mathrm{c}\chi & -\mathrm{s}\chi & 0 \\ \mathrm{s}\chi & \mathrm{c}\chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{c}\varphi & \mathrm{s}\vartheta\mathrm{s}\varphi & \mathrm{c}\vartheta\mathrm{s}\varphi \\ 0 & \mathrm{c}\vartheta & -\mathrm{s}\vartheta \\ -\mathrm{s}\varphi & \mathrm{s}\vartheta\mathrm{c}\varphi & \mathrm{c}\vartheta\mathrm{c}\varphi \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{c}\varphi\mathrm{c}\chi & \mathrm{s}\vartheta\mathrm{s}\varphi\mathrm{c}\chi - \mathrm{c}\vartheta\mathrm{s}\chi & \mathrm{c}\vartheta\mathrm{s}\varphi\mathrm{c}\chi + \mathrm{s}\vartheta\mathrm{s}\chi \\ \mathrm{c}\varphi\mathrm{s}\chi & \mathrm{s}\vartheta\mathrm{s}\varphi\mathrm{s}\chi + \mathrm{c}\vartheta\mathrm{c}\chi & \mathrm{c}\vartheta\mathrm{s}\varphi\mathrm{s}\chi - \mathrm{s}\vartheta\mathrm{c}\chi \\ -\mathrm{s}\varphi & \mathrm{s}\vartheta\mathrm{c}\varphi & \mathrm{c}\vartheta\mathrm{c}\varphi \end{bmatrix}$$

Note that these matrix products are not commutative: this means that a different sequence of rotations produces a different **E** matrix.

In module ***Boxcry***, the inertial matrix **I** of each symmetry–related molecule is treated as an Euler rotation matrix that transform the global reference system into the local (inertial) one (see Appendix, Section A3). Thus, the three Euler angles θ, φ, χ to be written in the *.bxi* file are :

$$\left.\begin{array}{c}\vartheta = \mathrm{atan2}[\mathbf{E}(3,2), \mathbf{E}(3,3)] \\ \varphi = \mathrm{atan2}\left[-\mathbf{E}(3,1), \sqrt{\mathbf{E}(1,1)^2 + \mathbf{E}(2,1)^2}\right] \\ \chi = \mathrm{atan2}[\mathbf{E}(2,1), \mathbf{E}(1,1)]\end{array}\right\}$$

Where atan2 is a modified $\tan^{-1}$ function so that atan2(x, y) = $\tan^{-1}(y/x)$ if x >0. If x is negative, the function gives $\tan^{-1}(y/x)+\pi$ if y is positive, or $\tan^{-1}(y/x)-\pi$ if y is negative. In other words, the sign of both the arguments are used to determine the quadrant of the output. Singularities arise whenever x=0 and y=0, for which the function atan2 is undefined. This happens for molecules or atoms in special positions. To cope with this problem, you should lower the symmetry of your space group and supply to the MiCMoS system a corresponding *.cif*. To do so, the Bilbao Crystallographic Server is definitely your friend (https://www.cryst.ehu.es/).

A7. The *Pretop* routine

In MC/MD calculations stretching and bending potentials are defined by harmonic functions, R and θ being distances and angles and R^0/θ^0 the corresponding reference values:

$$E(stretch) = E_s = \frac{1}{2}k_s(R - R^0)^2$$
$$E(bend) = E_b = \frac{1}{2}k_b(\cos\theta - \cos\theta^0)^2$$

Program ***Pretop*** reads a *.oeh* file and generates a template topology file with all possible stretch and bend potential sites. Reference distances and angles are taken as they are in the geometry supplied by the *.top* file coordinates, so they are not always strain-free values. Stretching (k_s) and bending (k_b) force constants are estimated by a combination of data derived from high-quality *ab initio* stretch and bend energy profiles on sample molecules, and of fitting against 54A7 GROMOS force field parameters (see Gavezzotti, A. & Lo Presti, L. J. Appl. Cryst. 2019, 52, 1253–1263). To add flexibility, overall rescaling factors are read in by the module at running time and are applied to all constants. Preliminary experience shows that values of 1.2 to 1.5 are appropriate as the force constants seem a bit underestimated.

The above assumptions derive from the fact that in MiCMoS the purpose of stretch and bend potentials is to keep molecules undistorted, rather than to describe precisely thermal vibrations. The whole package is oriented to low-frequency intermolecular libration and diffusion, while high-frequency vibrations much less if at all relevant. This minimal loss of physical reality affects only marginally the accuracy of the simulation and is counterbalanced by the avoidance of complex algorithms to prevent molecular distortions.

A7.1 Stretching potentials

Table A7.1
Bond stretching force constants from ab initio MP2/6–31G** calculations, consistent with energies expressed in kJ/mol and distances in Å.

Bond	R_{expt}	$R°_{calc}$	k_{str}	system for $R°$ and k calculation
-C≡C-	1.183	1.223	9620	but-1-yne
>C=C<	1.346-1.360	1.353	5600	butadiene
≡C - C≡	1.378	1.383	4540	buta-1,3-diyne
Car---Car	1.382	1.397	4640	benzene
Csp^2 - Csp^2	1.439	1.457-1.463	3400	butadiene
≡C - Csp^3	1.467	1.472	3340	but-1-yne
Csp^2 -Csp^3	1.503	1.513-1.515	3120	toluenes
Csp^3 - Csp^3	1.523	1.517-1.536	2800	butane
Csp^3- H	1.085	1.093	3630	ethane
Csp^2 - H	1.077	1.087	3630	benzene
C≡N	1.139	1.180	11500	acetonitrile
Csp^3 - N<	1.461	1.460	3540	trimethylamine
Csp^2 - O	1.369	1.381	4320	methoxybenzene
Csp^3 - O	1.435	1.432	3630	dimethylether
C=O	1.214	1.227	8200	acetone
Csp^2 - F	1.346	1.358	4200	fluorobenzene
Csp^3 - F	1.367	1.397	3950	fluoroethane
Csp^2 - Cl	1.735	1.742	2580	chlorobenzene
Csp^3 - Cl	1.771	1.784	2410	chloroethane
Csp^2 - Br	1.892	-		
Csp^2 - I	2.095	-		
N = O nitro	1.218	-		
N-H	-	1.018	5300	urea
O-H	-	0.987	4250	acetic acid

For a generic molecule with generic bond distances, the actual force constants adopted in ***Pretop*** come from a fitting of distance/force constant plots from the above data; for example, a generic carbon-carbon bond stretching force constant comes from interpolation of the *k* vs. distance plot for all the C···C types in the above Table.

Table A7.2
Other averaged or guessed *k*/*R* values used to define stretching potentials. A question mark means uncertain or unknown entries.

Bond	$R°$ / Å	k_s
C-F	1.38	4000
C-Cl	1.75	2500
C-Br	1.89	?2000
C-I	2.10	?1500

For bonds not in the above list, $R°$ distance is taken from coordinates in *.oeh* file and k_s is set to zero.

A7.2 Bending potentials

Table A7.3
Bending force constants from ab initio MP2/6–31G** calculations, for energies expressed in kJ/mol and angles in deg.

Angle	$\theta°$	k_b, kJ mol^{-1}	sample system
C-C-C	112.4	880	propane, bend of CCC and CCH angles in the CH_2 group
C=C-C	124.5	1030	propene, bend of CCC and CCH angles
C-O-C	112.4	972	dimethyl ether
C-C=O	123	894	acetone simultaneous bending of two angles
(O)=C-O-H	104	475	-COOH (acid)
(O)=C-N-H	120	940	-$CONH_2$ (amide, bend of 2 CNH)
(Ar)C-O-H	107	550	alcohol (phenol)
CCH	120	890	benzene simultaneous bending
HCH	106.5	530	propane, scissor mode at the methylene group
CCH	110	680	Methyl
CCH	110	980	Methylene

As mentioned before, $\theta°$ the bond angles come from the supplied atomic coordinates. k_b of CCC, XCH (X=C, N, O, S, Cl), CNO, C_xN_{3-x} and C_xO_{3-x}, are assigned through the appropriate, if approximate, fitting of ab initio data and 54A7 force field values. Table A8.5 has some averaged values.

Table A7.4
Averaged bending potential constants.

Bond	$\theta°$ / deg	k_b
HCH	108.0	470
COH	110.0	450
HNH	120.0	445
CNH	115.0	460

A7.3 Torsional potentials

In MC/MD calculations, torsional potentials are defined as $(E(tors) = k_{tors}[1 + f\cos m\varphi]$, see Sections 6.4.1 and 7.4.1 in the main text). *f* is normally equal to +1 or –1, and *m* ranges between 1 and 3. In ***Pretop***, starting values for the torsion angles are obtained directly from the coordinates in the topology file with the procedure described in Appendix, Section A8, and the local geometry is exploited to determine tentative values for *f* and *m*.

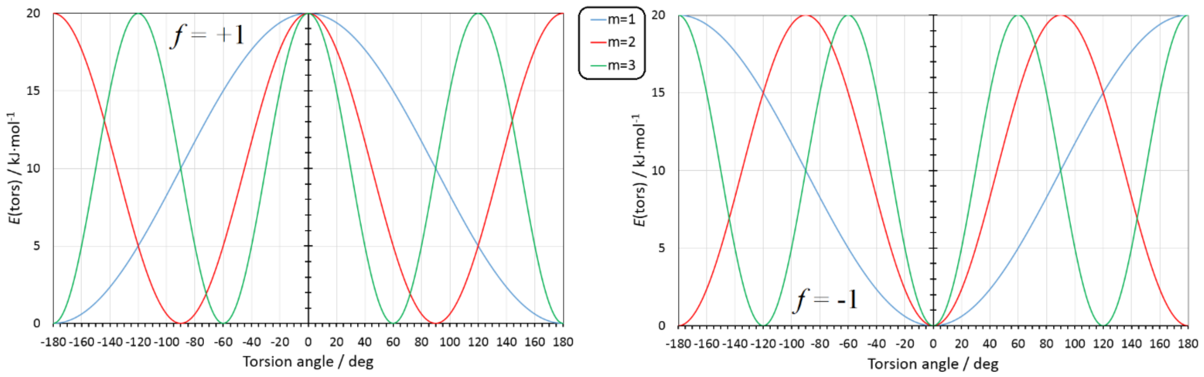
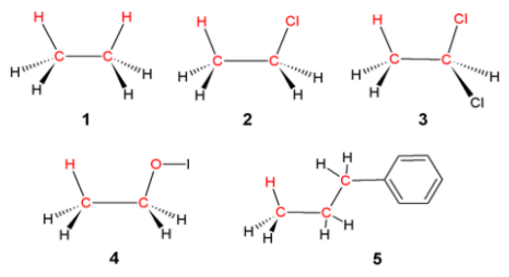
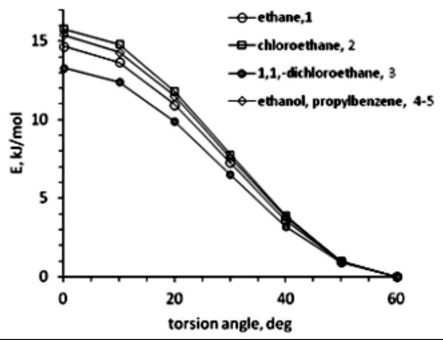
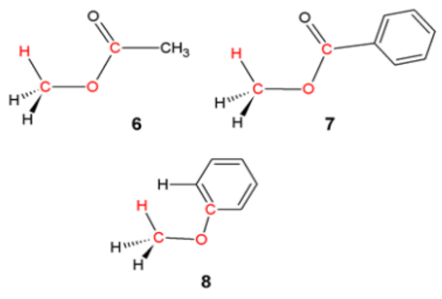
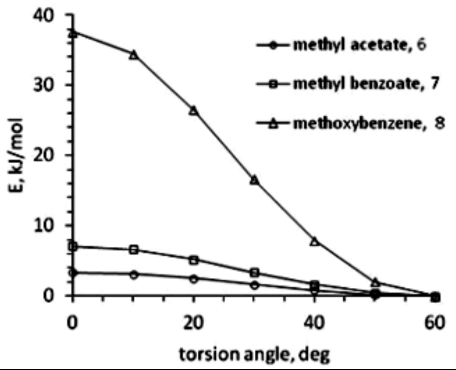
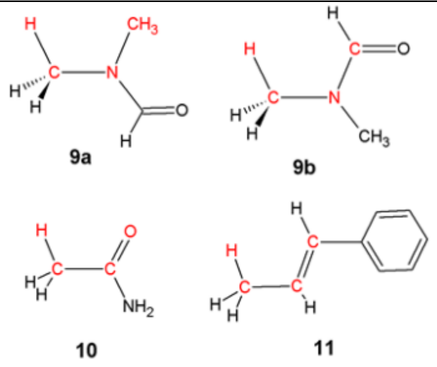
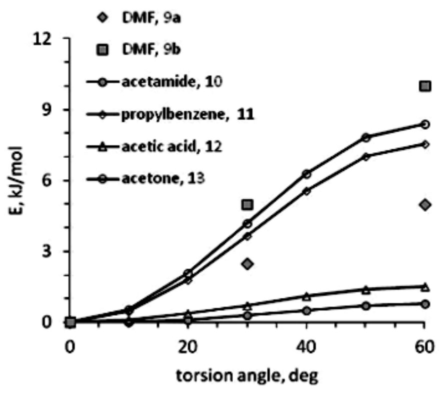
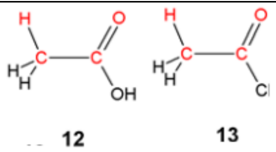


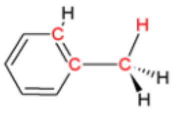
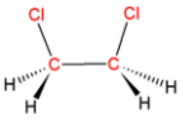
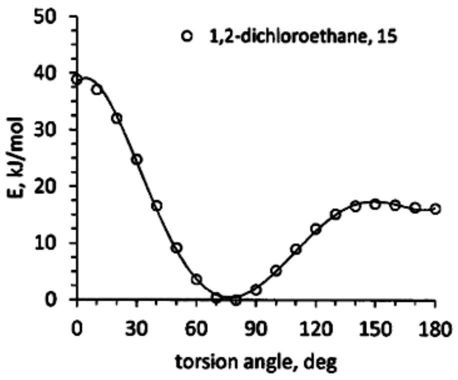
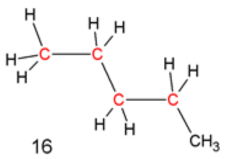
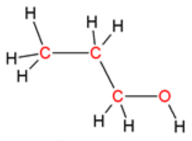
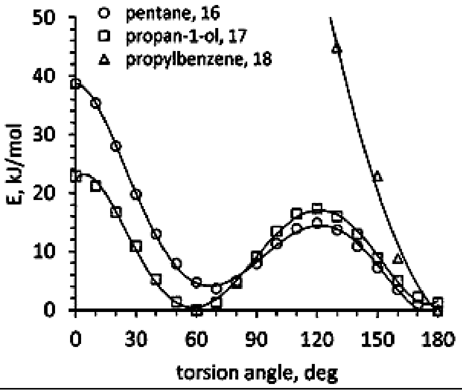
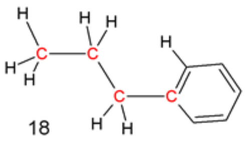
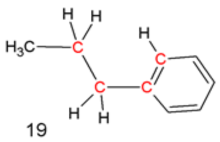
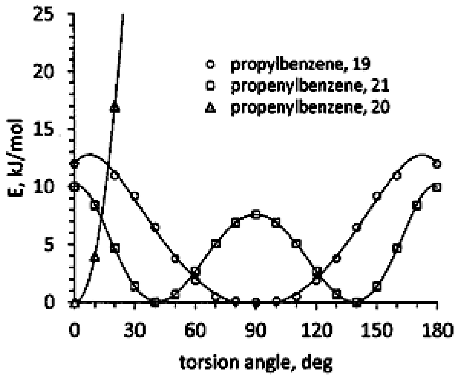
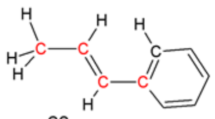
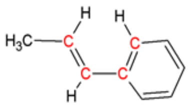
Figure A7.1. (a) Functional form of *E*(*tors*) for k_{tors} = 10, *f* = +1 and *m*=+1 (light blue), +2 (red) and +3 (green). (b) Same as (a), for k_{tors} = 10 and *f* = –1.

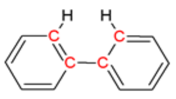
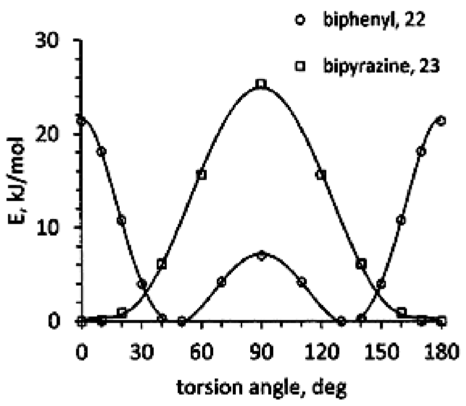
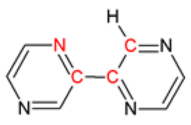
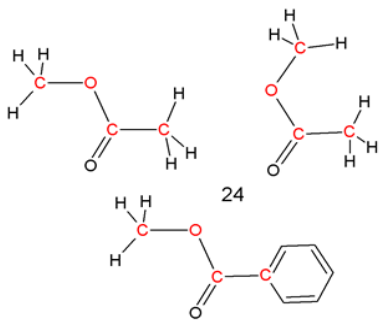
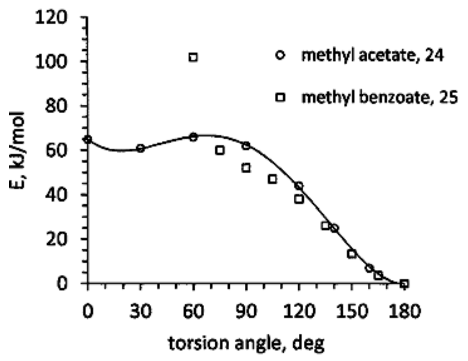
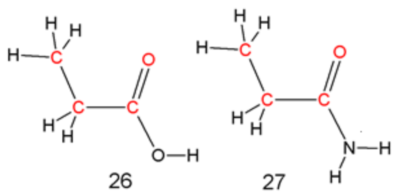
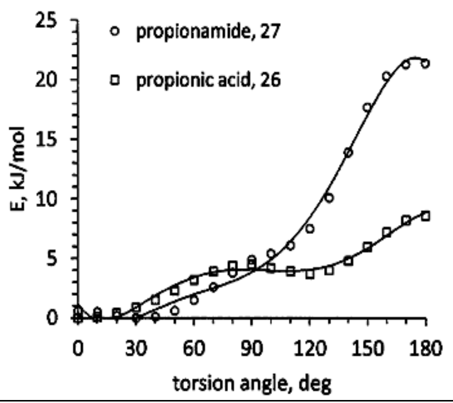
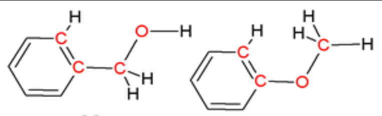
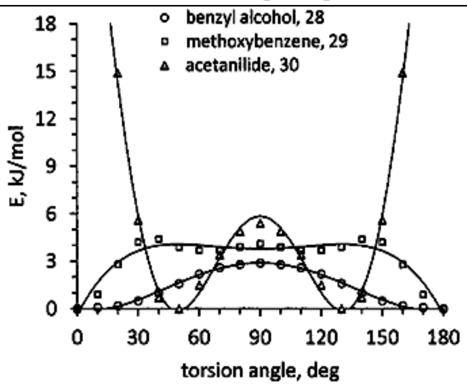
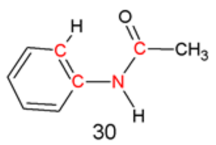
Proper torsions: Let N_a and N_b be the numbers of bonds out of bound atoms *a* and *b*. One torsion is assigned to any bond joining two atoms with N_a, $N_b > 1$. The corresponding assigned parameters must be checked and reset with actual values, many of which are given in Table A7.5, which summarizes the complete maps of torsional potential energies from post–HF quantum chemistry calculations.

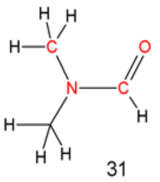
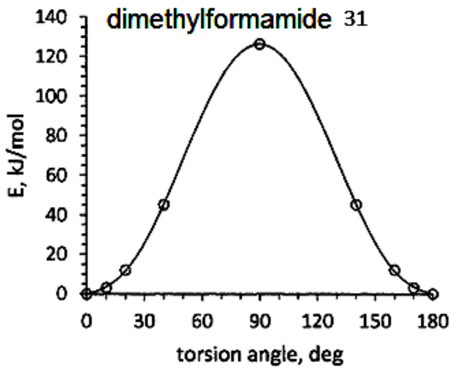
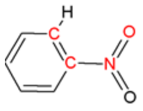
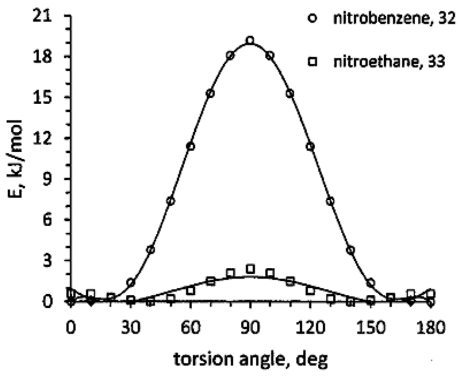
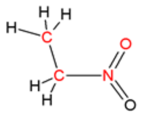
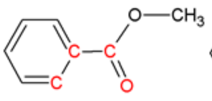
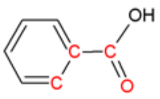
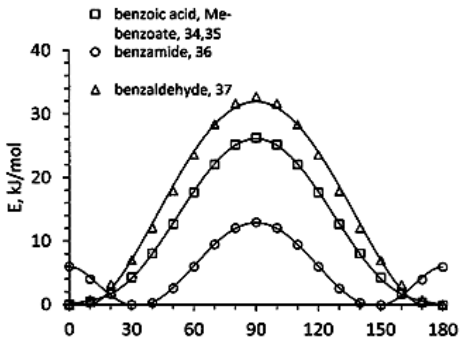
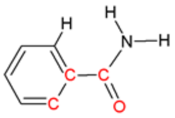
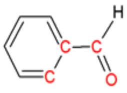
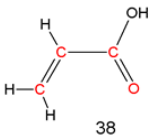
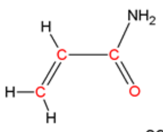
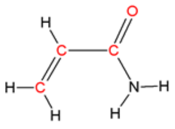
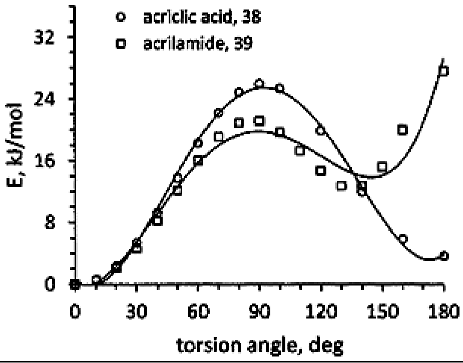
Improper torsions: The routine assigns one improper torsion to any trigonal center with k_{tors}=100, *f*=–1.0 and *m*=+1.

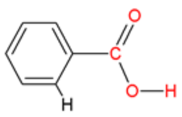
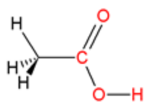
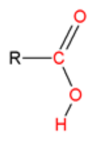
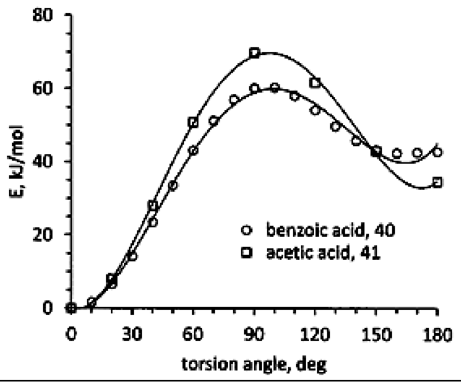
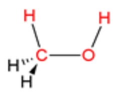
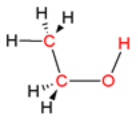
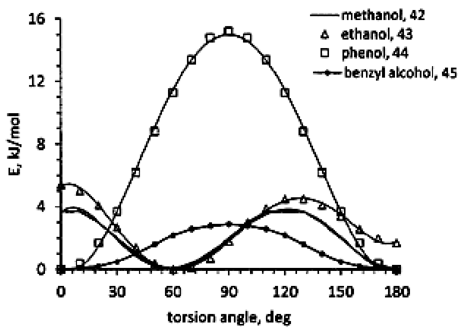
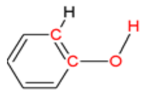
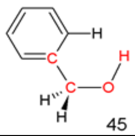
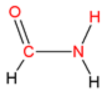
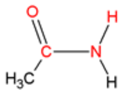
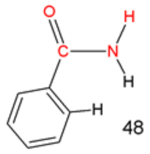
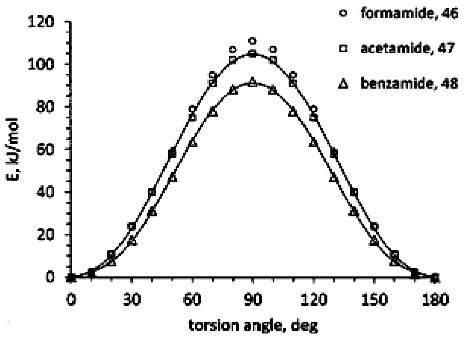
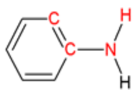
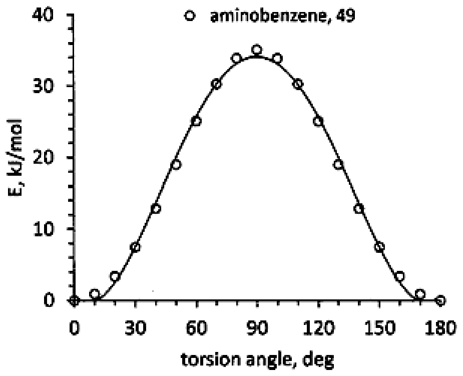
Table A7.5
Recommended values for parameters k_{tors}, f and m for MC/MD torsional potentials. "System" shows the chemical connectivity; atoms highlighted in red define the torsion. "Potential" shows the MP2/6-31G** results for single–point calculations of the total electronic energy as a function of the torsion angle, φ. "k_{tors}, f and m". k_{tors} is equal to ½ of the barrier height.

System	Potential	k_{tors}	f	m
 1 2 3 4 5		7.5	1	3
 6 7 8		2.5	1	3
 9a 9b 10 11		4	−1	3
 12 13		0.5	−1	3

System	Potential	k_{tors}	*f*	*m*
 **14**	Not available	2	–1	2
 **15**		7.5	1	3
 16  17		6	1	3
 18		50	1	1
 19		6	1	2
 20		50	–1	2
 21		10	–1	2

System	Potential	k_{tors}	f	m
 22		10	–1	2
 23		10	1	4
 24 25		40	1	1
 26 27		10	–1	1
 28 29		2	–1	2
 30		10	1	4

System	Potential	k_{tors}	*f*	*m*
 31		60	–1	2
 32		10	–1	2
 33		1	–1	2
 34  35		14	–1	2
 36		6	–1	2
 37		16	–1	2
 38   39		11	–1	2

System	Potential	k_{tors}	f	m
 40  41 		35	–1	2
 42  43		2	1	3
 44		7	–1	2
 45		2	–1	2
 46  47  48		50	–1	2
 49		17	–1	2

A8. Procedure to determine torsion angles according to standard conventions.

This procedure is exploited by Monte Carlo and Molecular Dynamics modules to compute torsion angles in the range $-180 \le \tau \le +180$ deg, to determine the intramolecular part of the potential (see Sections 5 and 6 in the main text). Given a 1–4 sequence of **i**, **j**, **k** and **l** bonded atoms (Figure A8.1a), the algorithm sets a local right–handed Cartesian reference frame, with origin on atom **j** (Figure A8.1b). Then, the whole atom sequence is rotated so that the **j**–**k** vector is aligned to the x axis, and the first atom **i** has $z = 0$. This way, the first three atoms now lie on the same (x,y) plane. This is always possible, as three atoms define a unique plane. To ensure the right handedness of the reference system, z must point upwards (Figure A8.1c).

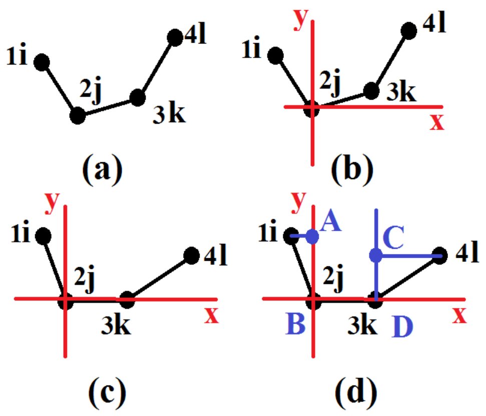


Figure A8.1. Procedure for the determination of torsion angles In MC and MD according to convention. $\tau > 0$ if looking from 1 down 2-3 atom 4 turns right.

Finally, *A*, *B*, *C* and *D* reference point (Figure A8.1d) are defined on the basis of the atomic coordinates in the local reference frame according to:

$$\mathbf{A} = [0, y_1, 0];\ \mathbf{B} = [0, 0, 0];\ \mathbf{C} = [x_3, y_4, z_4];\ \mathbf{D} = [x_3, 0, 0]$$

The following difference vectors and vector modules are thus computed:

$$\mathbf{A} - \mathbf{B} = [0,\quad y_1,\quad 0];\ |\mathbf{A} - \mathbf{B}| = |y_1|$$

$$\mathbf{C} - \mathbf{D} = [0,\quad y_4,\quad z_4];\ |\mathbf{C} - \mathbf{D}| = \sqrt{(y_4^2 + z_4^2)}$$

The scalar product between vectors **A**–**B** and **C**–**D** provides the angle between them, τ:

$$\cos\tau = \frac{(\mathbf{A} - \mathbf{B}) \cdot (\mathbf{C} - \mathbf{D})}{|\mathbf{A} - \mathbf{B}| \cdot |\mathbf{C} - \mathbf{D}|} = \frac{y_1 \cdot y_4}{y_1 \cdot \sqrt{(y_4^2 + z_4^2)}} = \frac{y_4}{\sqrt{(y_4^2 + z_4^2)}}$$

$$\tau = \cos^{-1}\left(\frac{y_4}{\sqrt{(y_4^2 + z_4^2)}}\right)$$

By the normally adopted convention looking from atom **i** down the **j**–**k** bond τ is positive if **l** turns to the right. The sign of τ is the same as that of the z_4 coordinate.