MACSIMUS manual
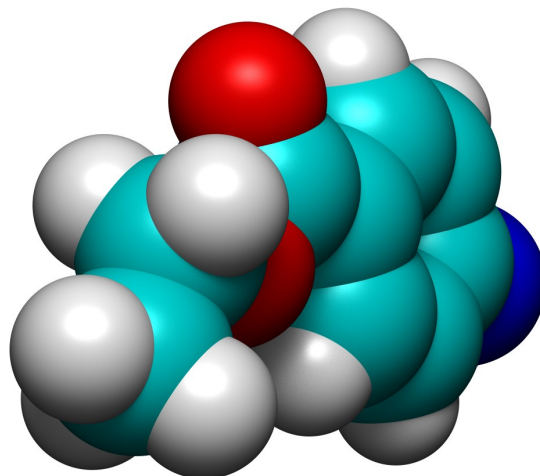
```
benzocaine (ethylaminobenzoate)
parameter_set = charmm21


    HA
     |
HA-CT-HA
     |
 HA-CT-HA
     |
    OSn.2
     |
    Cp.7=On.5
     |
    C6R--C6R-HA
     |     |
HA-C6R    C6R-HA
     |     |
HA-C6R--C6R-NPn.5^-Hp.25
              |
           Hp.25
```

Most often used links:

2.2 Blend synopsis and options

9.2 Cook synopsis and options

9.2.5 Cook input data

© Jiří Kolafa 1993–2025

With the exception of (some of the code or data in):

- ray, Mark VandeWettering "reasonably intelligent raytracer"
- CHARMM force field (files: blend/data/charmm*.par, blend/data/charmm*/*.rsd)
- GROMOS force field (files: blend/data/gromos*.par, blend/data/gromos*/*.rsd)
- other data and methods as referenced in particular files
- traces of DOS mouse library by unknown author (files: mouselib.[ch])
- amoeba implementation by Z. Wagner

all the code and data are covered by the GNU General Public Licence V3

Credits:

- All items mentioned above
- Several bug fixes by T. Trnka
- Users (discovering bugs, feedback; chronologically): Jan Picálek, Linda Viererblová, Karel Matas, Nikita Parfenov, Victor Kwan, Jiří Janek, Martin Klíma, David Celný

Contents

IV Appendixes   300

Foreword

MACSIMUS is essentially a "one man software". Its development started in 1991 in the John Perram's group in Odense, Denmark, under project PROSIS (PROtein SImulation Software). The first sufficiently general version is from 1993. In 1999, PROSIS was renamed to MACSIMUS (MACromolecule SIMulation Software), the name invented by John Perram. However, the development has declined from macromolecular simulations and has focused rather on fluids, ionic systems, polarizability, surface and two-phase phenomena, etc.

MACSIMUS makes use of the traditional Unix (now Linux) front-end of command prompt. Graphics is based on hot keys and old-style buttons for most common task. Midnight Commander is supported.

Basic components of MACSIMUS with the most important file types follow:
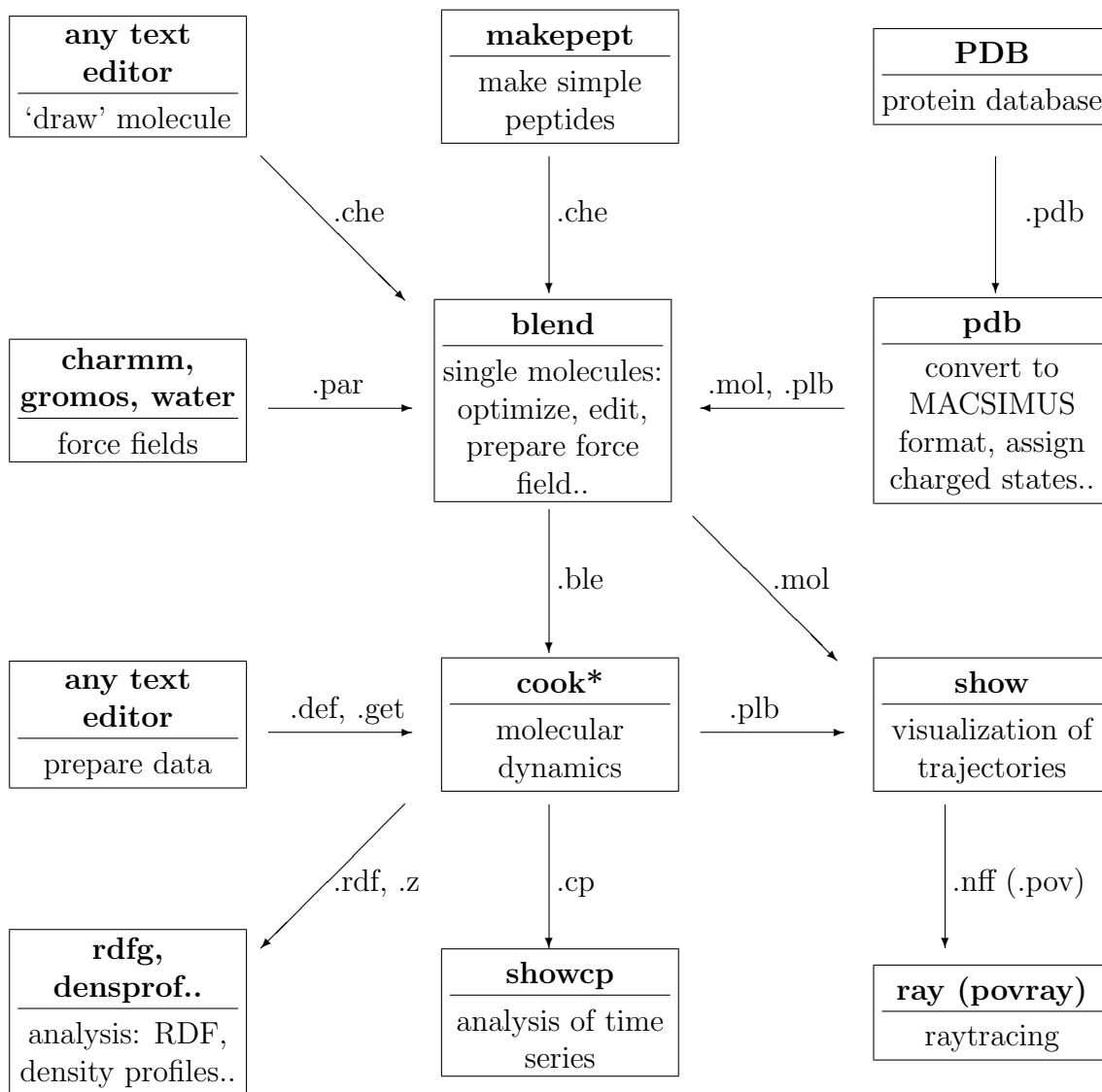
Part I

`blend` version 2.4e and the force field parameter file

Odense, October 1993 (V0.663), November 1995 (V1.5), August 1996 (V1.6g)
Guelph April 1996 (V1.6d)
Evanston December 1995 (V1.6c), July 1996 (V1.6f)
Prague 1996 (V1.7)
1999 PROSIS renamed to MACSIMUS
More updates: 2001–2025

Program `blend` assigns a force field to (a mixture of) molecules and prepares thus data in the `ble`-format to be used as input in molecular dynamics simulation programs of the `cook` family. The input to `blend` is either in a `che`-format (2D ASCII pseodogragraphics) or `mol` format (ASCII-based table of molecular topology).

Other `blend` features include: essential dynamics, 2nd virial coefficient, normal modes of vibration (more general code is available in `cook`, editing etc.

Unfortunately, not all models are implemented; e.g., Gaussian charges are not, see Sect. 15.6.1. Consequently, there is more work with obtaining `ble` and `mol` files needed for `cook` and `show`.
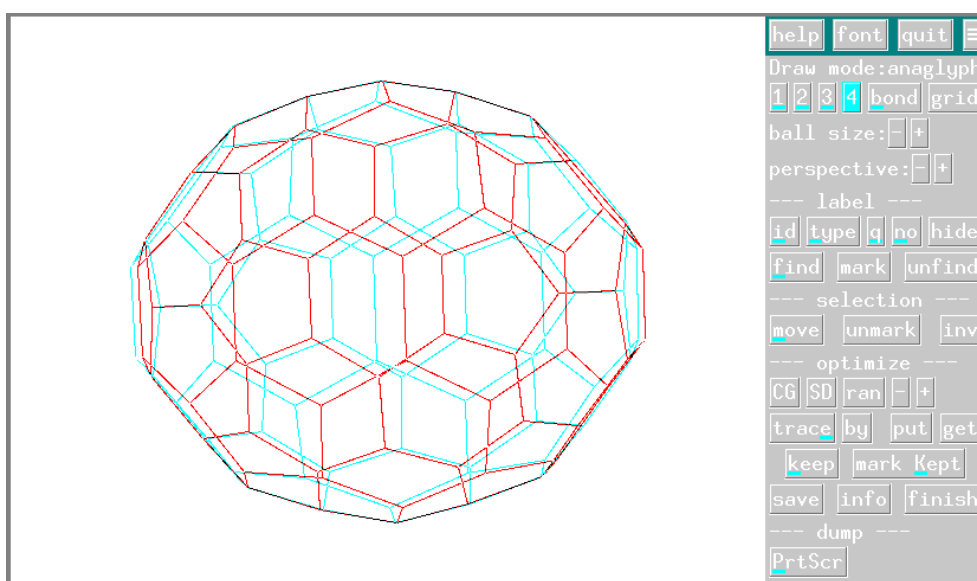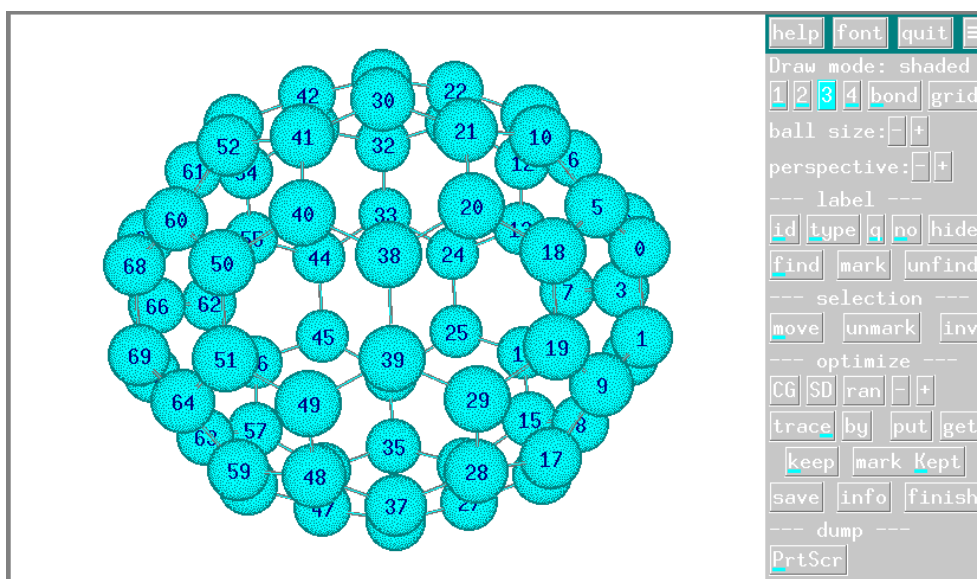
Chapter 1

Introduction

1.1   Force fields

Force fields describing complex organic molecule are composed of many elementary energy terms, typically including Lennard-Jones and Coulomb interactions for non-bonded forces, and bond, bond angle, dihedral and improper torsion potentials for intramolecular bonded forces.

Our task is to assign a force field to a molecule given by a chemical formula without additional information with the exception of partial charges attached to atoms which cannot be easily derived from the chemical formula only and must be defined for each residue or calculated by quantum chemistry. This requires a comprehensive analysis of the molecular structure. Since the rules of assigning a force field to the molecular structure differ from one package to another, and even from one version of the same package to another, and are often on purpose obscured by the authors to prevent any other use of the force fields than buying their expensive and inefficient programs, program `blend` has become much more complicated than was originally intended.

1.2   'blend' overview

Program `blend` assigns a force field to (a mixture of) molecules and prepares thus data for molecular dynamics simulation program `cook` (better said, many versions of this).

`blend` analyses the structure of a molecule and finds the required energy terms (bonded and non-bonded) by looking in tables. Partial charges are usually not part of the force field and are given separately.

The molecules are described internally in a table-based format (originally produced by graphical Molecule Editor `MEDIT` which unfortunately has died)[1]. They can be input by a user in a text format close to the chemical structure formulas or via external sources (PDB interfaced by utility `pdb`).

Using the constructed force field, `blend` can find the (local) energy minimum of the molecule (from given configuration or from random input) and also fill missing atom coordinates (typically hydrogens).

Several molecules can be blend-ed together and the prepared mixture can be simulated by the MD program `cook`. In other words, `blend` creates a file that contains a full description of the

force field of given molecules and this file is used by `cook`.

Although `blend` was developed originally with the CHARMM force field in mind, an attempt was made to be more general if possible.

During years, many features have been added to `blend`: support of polarizability, molecular editing features, essential dynamics, normal mode vibrations, inertia matrices and radii of gyration, internal coordinates, etc.

1.3   Versions

See `blend/metamake.mmk` for compilation instructions.

Major versions of 'blend' are:

`POLAR` This version supports scalar and tensor polarizabilities of atoms. With the Busing (exp-6) potential, it supports the 'shell-core' model. The polarizability may also be saturated (see Sect. 25). Use #define POLAR for compiling.
BUG: Gaussian charges and BK3 water is not supported (however, it is in `cook`).

`site-site potential` The used site–site potential: Lennard-Jones, exp-6, WCALJ, Busing, RGL (metal) and more.

Chapter 2

Running `blend`

2.1 Environment

Environment variable `BLENDPATH` can be set to point to the path that contains the parameter files (with extensions `.bin` and `.par`). If it is empty, the parameter files are looked for in the working directory.

Example (sh, bash):

```
export BLENDPATH=/home/jiri/blend/data
```

Other environment variables are:

```
BLENDGEOMETRY = window geometry, see also option -g
BLENDINIT = initial keystroke, see also option -I
GUI = common GUI setup, see page (g)ui after running blend without a parameter
```

2.2 Synopsis

WARNING: This may be somehow out-of-date, run `blend` without any parameter to obtain the up-to-date (but brief) info!

```
blend \
[-o file | -s] \
[-lnumber] [-vnumber] [-znumber] [-gnumber] \
[ par-options ] [ parset.par | parset.bin ] \
[ charges.pch ] [ reactions.rea ] \
[ mol-options ] { species | species.mol | species.che } \
[ mol-options { species | species.mol | species.che } ] \
...
```

The options are parsed from left and thus, e.g., `-s` must precede any parameter that generates output. Option `X` is switched on (i.e., set to value 1) by one of `-X -X+ -X1` and switched off (cleared to value 0) by `-X-` or `-X0`. The options of `blend` are case sensitive (most

other MACSIMUS programs use case-insensitive options). Numerical options may be given in decimal, octal (number begins with 0) or hexadecimal (number begins with 0x).

If `blend` is run without any parameter, a brief description of options is printed.

2.2.1 Global options

These options apply to the whole run and should be placed before any molecule is processed.

-g<u>number</u> Graphical show of minimization (by <u>number</u> steps). Valid for X-Windows or DOS Turbo C versions. The default is `-g3` for DOS but `-g0` (no graphics) for UNIX (X-Windows)

-l<u>number</u> Size of the line buffer for reading files (default=256). You may find this option useful if you have a complicated formula in the chemical format.

-o <u>file</u> This option sets the batch mode. Output from `blend` is written to <u>file</u>; if <u>file</u> does not have extension, `.ble` is appended. No keyboard input is requested; in the case of error, the calculations exit with a nonzero return code. The space between `-o` and the file name is optional. Cannot be combined with option `-s`.

-s<u>number</u> (Deprecated) Enable scrolling (screen buffer = <u>number</u> kB). If no <u>number</u> is given, the default is 31 kB (this is also maximum for DOS). Cannot be combined with option `-o`. `blend` must be compiled with option `-DSCR` to enable this feature.

-v Sum of (note that the option may be octal or hexadecimal, e.g., `-v11 = -v013 = -v0xb`)

  -v0 Neither force field information nor the table of sites are printed. This is the default for the interactive mode (no output file specified by `-o` option).
  -v1 List sites and their parameters. Tables of bonds, angles, dihedrals and impropers are not printed.
  -v2 Print tables of bonds, angles, dihedrals, impropers etc.
  -v3 The same as `-v1` + `-v2`: Print the full force field information (as required by `cook*`). This is the default for the batch mode (<u>file</u>`.ble` is created).
  -v4 Verbose (info on topological analysis, selection of terms, etc.).
  -v8 List the used pair site–site parameter terms.
  -v16 POLAR only: print convergence rates.
  -v256 List all pair site–site parameter terms (even for atoms not present in the molecules), may be very lengthy.
  -v4 -J1e-11 -v0 (Debugging Jacobi diagonalization: shows the diagonalization progress and turns off verbose mode otherwise. See `-J`, `-N`, `-E` for details.)

-z<u>number</u> (0) Seed for random number generator. If `-z` is not specified, the seed is derived from time so that it is different for each run.

-_<u>digit</u> (6) (Underscore) Output precision of force constants. Most of the force field terms (like force constants K) are printed with <u>digit</u> decimal digits, some of them (to have comparable relative precision) with precision <u>digit</u>`-1` or <u>digit</u>`+1`. Must be $0 < \underline{digit} < 9$. In addition, negative values of <u>digit</u> force fine adjustment of rounding of partial charges when a mol-file is (re)created so that their sum for a molecule is an exact integer. Useful for charges pasted to a mol-file from a quantum-mechanical calculation.

2.2.2   par-options **and parameter files**

Normally, the parameter set to be used is defined in the mol-file (keyword `parameter_set`). If not defined, it can be specified as a parameter (but note that all molecules blended together must use the same parameter set!). The options that control the force field assignment logic have their defaults defined in the parameter file and may be changed using a command line parameter. See Sect. 19.1.4, for more information. A list of par-options follows:

`parset.par` Name of the parameter file in text format.

`parset.bin` Name of the parameter file in binary format. The default is `charmm21.bin`. The binary file is created from the text file if option `-b` is given. It is faster to use binary files for parameters (well, in 21st century the difference is negligible, but on 386 it used to make a difference). WARNING: binary files of full and TINY versions are not compatible.

`-anumber` For number ≥ 0: scale all angle force constants number% times. For number < 0: set all angle force constants to $-$number kcal/mol. For number $= 0$: constrain angles (inefficient minimization algorithm).
See also `-b`. Note that the changed `-a0` value is exported to a ble-file. `-a0` does not work accurately enough in some cases like planar molecules (e.g., TIP4P water); a large value may help, but the algorithm becomes inefficient.

`-b` `parset.par` (Deprecated) Make binary image `parset.bin` from `parset.par`. The name of the source parameter file `parset.par` must be specified after option `-b`. `blend` version number is written to the header of `parset.bin` and two major digits (e.g., 1.2) are tested whenever `blend` reads the binary file.

`-bnumber` For number > 1: scale all bond force constants number times. For number < 0: set all bond force constants to $-$number kcal/mol/Å^2. For number=0: constrain bonds (inefficient minimization algorithm).
Warning: `-b1` is reserved for making binary files (see above).

`-fnumber` Fabricate (make up) bond or angle terms that are missing in the parameter file.

> `-f0` Missing bond or angle terms issue a warning and no term is added.
>
> `-f1` A missing bond parameter is added with length 1.5 Å and force constant $K = 50$ kcal/mol.
>
> `-f2` A missing angle parameter is added with equilibrium angle 120° and force constant $K = 20$ kcal/mol.
>
> `-f3` Both `-f1` and `-f2`.

`-inumber (default=5)` Calculate the maximum angle potential energy (over all angle potential terms in a molecule) and print a warning if this energy exceeds number kcal/mol. Such a warning for "normal" molecules means that the configuration is wrong, e.g., methane is in a wrong "pyramidal" conformation:

The default `-i5` should detect such cases.
Use `-i0` if it is OK and the warnings bother you.
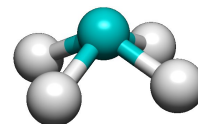
`-xnumber` Lennard-Jones scaling; the default is `-x1000100` (all scaling factors are unities).

-xEEE (three digits) All Lennard-Jones energy terms ($\epsilon = -E_{\min}$) are multiplied by factor EEE/100.

-xRRRREEE (more than 3 digits) In addition, all Lennard-Jones radii (σ or R_{vdW} are multiplied by factor RRRR/1000.

-x-DDD Energy terms are scaled $x = \exp(\mathtt{DDD}/1000)$ times and radii $x^{-1/3}$ times. This means that density is scaled x-times with the structure unchanged.

-@number POLAR only: Output multiplication factor in % for the value of `shell` (number of shell electrons for auxiliary charges for MD with polarizabilities). Only exported to MD because `blend` uses exact point dipoles. Default=100%.

-^number

-~number POLAR only: polarizability multiplication factor in %. Default=100%.

-[number POLAR only: saturation energy multiplication factor in %. Default=100%. Rarely used.

-[number Active with -n-1 only: reorder sites in molecules of number sites in increasing site mass. E.g., -[3 -n-1 will reorder TIP3P water to HHO.

2.2.3 mol-options **and molecular files**

These options can be changed before each molecule is processed. Once an option is set or cleared, its new value remains active until it is cleared or set again (i.e., it is not re-set to the default).

species.mol Molecule file (*mol-file*) generated by `blend`, `pdb` (and originally by the molecule editor `MEDIT`) is read and processed.

species.che File in the chemical format (*che-file*) is read and processed. The corresponding file species.mol is created or rewritten (in this case the old species.mol is backed-up as species.mol (species.mox for DOS). See Sect. 4.2.

species If no extension is specified, species.mol is tried first. If it exists, it is read and processed, otherwise species.che is processed instead and the corresponding species.mol is created. Thus, if you call `blend` for the second time, species.mol and not species.che is read.

-cnumber This option affects how chirality information on CH1E and other chiral atoms is treated. See Sect. 3.8.6.

-c0 Chirality info is taken from the .mol file or determined by using z-coordinates in the .che file (default). If a 3D configuration is available and the calculated chirality differs, a warning is printed.

-c1 Calculate unknown (i.e., not given in the .mol file) chiralities from the configuration (.plb, .che, etc.) and write them to the .mol file. The old species.mol is backed-up as species.mol (species.mox for DOS).

-c2 Calculate all chiralities from the configuration (`.plb`, `.che`, etc.) and write them to the `.mol` file. The old species.mol is backed-up as species.mol (species.mox for DOS). Warning is printed if the calculated chirality does not match the former chirality from the `.mol` file.

-dnumber Automatic bond generation. For positive number, a bond is added if the atom-atom distance is less than number% of the bond equilibrium distance. For negative number, all bonds are erased first. Useful, e.g., if some bonds are missing in the PDB file but the coordinates are good, or if ionic bonds are created (cation-anion strong force) and these should be turned into explicit bonds. Bond info must be present in the par-file. Useful range of —number— is between 110% and 120%; values larger than 130% may lead to extra bonds when atoms are accidentally close together. Values less than 100% issue a warning. Note: bonds are not created if there is a record in the PAR file for the bond, but the force constant is zero. This is the case of fake "bonds" as H-H in water to define a rigid molecule.

-e [no 2D input] Selects editing (adding/removing atoms/bonds). Cannot be combined with a 2D input (see below). If file species.edt exists then the edit information is read from this file, otherwise from standard input.

Note that any editing is performed BEFORE the molecule is minimized and shown. The recommended style of interactive work is to run two instances of `blend`, one for showing atom ID's, and the second for editing. After editing, kill the first `blend`.

Available commands are

? Print brief help

aa id type charge [id1 [id2 [...]]] Add Atom and connect it to atoms id1, id2, ... It will be marked as 'missing' and its position will be calculated later (cf. option -k). Note: thus, after short minimizing, you should start minimizing again with `keep=0` or re-run `blend`.

af id type charge [id1 [dist]]

af id type charge = X Y Z Add Free atom (not connected). It will be placed dist from atom id1 on the opposite side than id1's neighbors (or randomly if id1 does not have neighbors) and will not be marked as 'missing'. The second form adds the atom with given absolute coordinates X,Y,Z. Blend should be called with -y0 or -y2

aw [id1 [dist [#]]]

aw = X Y Z Add Water molecule (TIP3P model). Its oxygen will be placed dist from atom id1 on the opposite side than id1's neighbors (or randomly if id1 does not have neighbors) and will not be marked as 'missing'. Both water hydrogens will be marked as 'missing'. The atom ID's are `W#-O`, `W#-H1`, and `W#-H2`; if `#` is missing, it is advanced by 1 from previous `aw` command (or is 1 for the 1st `aw` command). The default dist is 2. See the note for `aa` statement. The second form places the water oxygen to X,Y,Z specified. Note: Equivalent to the following sequence of commands:

```
af W#-O OW -.834 ID1 DIST
aa W#-H1 HT .417 W#-O
aa W#-H2 HT .417 W#-O
```

ra id Remove Atom.

rm id Remove whole Molecule. id is the ID of any atom in the molecule.

ab id1 id2 Add Bond.

rb id1 id2 Remove Bond.

i id newid Change ID (rename atom)

q id charge Change charge.

t id type Change type.

p idPrint info on atom(s) id.

pf id type charge [grid [shell [NforPLB]]]

pw [grid [shell [NforPLB]]] Probe using Free atom / Probe using Water. Given probes (atom for `pf`, TIP3P water for `aw`) are placed to a grid around the molecule and the energy is calculated; in the case of water, the water molecule is first rotated to give the minimum energy. The minimum energies are then printed and optionally the playback files containing the molecule and NforPLB minimum energy probes are generated. This may be optimized by a special patch: compile-time option causing omitting all atoms kept on place from energy calculations (#ifdef OMITKEPT in blendmin.c)

react rea-file where rea-file is a reaction-file (extension `.rea`, see Sect. 2.3). Thus, 'reaction' can be performed during editing. Example:

```
af prot HP 1 16-OAC .8
react p.rea
.
```

ff parset Sets the force field (parameter set), also `parameter_set`=parset. If set, the editing commands are performed only if the current force filed matches the ff command. Turned off by empty parset (i.e., editing is on). This allows to have different editing statements for different force fields in one `.edt` file. Example:

```
ff charmm22
! this applies for charmm22 only
t CYS13aSG S
ff charmm21
! this applies for charmm21 only
t CYS13aSG ST
ff
! this applies for any force field
q CYS13aSG -0.8
```

! Any comment.

.

end End of editing (not needed for input from species`.edt`)

Blank lines are simply ignored. Wildcards **?** (matches exactly one character) and ***** (only at the end of id, matches any number of characters) can be used in id's. Example: if your mol-file has been created for the che-file (in some previous `blend` run), it has atom ID's of the form `12-CH2E`. Specify `12-*` if you know the number but not the atom type.

-e*number* [2D input] Scaling for 2D input given in % of the default values. If <u>number</u> is positive, the plane configuration is "waved" to prevent numerical problems with pure 2D starting configuration. This can be switched off by using negative <u>number</u>. Active only if 2D input has been specified (option `-r2` or `-r12`, or `.che` file as the only source of information: in this case option `-e` does not mean editing!).

-h*number* Constraint bond angles containing light atoms (typically hydrogens). This information is written to a ble-file applies for a subsequent `cook*` simulation, not the optimization performed by `blend`!

abs(<u>number</u>)+0.5 is the mass limit (in atomic units) to treat bond angles constrained. Thus, `-h-` (default) sets the limit to 0.5 (all bond angles with hydrogens can bend), `-h` or `-h1` sets the limit to 1.5 (bond angles with hydrogens are fixed but all other atoms not). This applies only to output passed to `cook` where the bonds corresponding to constrained angles are marked by K=0.0 in the table of bonds.

If the argument of `-h` is negative, all angles containing light atoms are constrained, irrespective of overdetermination and singularities.

Normally the argument of `-h` is positive. Then the program removes the overdetermined bond in groups like `-CH3` and constraints only one bond of two possible in a chain like `-C-NH-C-` and does not constraint the `H-N-H` angle in `-NH2`. No general algorithm detecting overdetermination or singularities is used but the used algorithm constraints correctly hydrogens in proteins and TIP3P water.

Using `-h` improves generally energy and constraint conservation with a marginal impact on the generated trajectory. Higher values of `-h`<u>number</u> are not recommended.

The new bond force constant exported to a blend-file is equivalent to the angle bending (the same vibrational frequencies) provided that the adjacent bonds are constrained. If also the bonds are vibrating (e.g., `-u9999` on cook), the vibration spectrum differs.

-j*number* This option has two purposes:

1. Allows adding special bonds (harmonic springs between atoms). Useful e.g. for keeping certain distances (nearly) constant (if the force constant is high). These bonds are used while energy minimization and, if option `-o` is used simultaneously, exported to `cook`. These 'bonds' are specified either in file species.`jet`, or, if this file does not exist, may be given from keyboard. The format of data is similar to the format of the table of bonds, See Sect. 3.6, only atom IDs replace atom types. Example:

   ```
   !REMARK : Distances to keep tail from folding up:
   ! id       id       K length
   ASN3CA   TYR159CA  1   61
   LEU22CA  TYR159CA  1   52
   ALA26CA  TYR159CA  1   84
   end
   ```

   where K is in [kcal/mol] and length is in [A]. Single statement `end` or `.` marks the end of data (not needed if data are read from file).

   HINT: if you want to use these constraints only within `blend` and not in `cook`, re-run `blend` with these options: `-m- -w- -o` <u>file</u>, but without `-j`.

2. Dihedrals to constrain can be also given in this file. Example:

```
! ID ID ID ID ANGLE/degree(0,180)
5-CT 6-CT 7-CT 8-CT 120
```

BUGs (in V2.1k): cos(angle) is used internally, 0 and 180 are not allowed and the sign of ANGLE is not distinguished.

-knumber This options allows to keep some atoms in fixed positions during minimization.

> **-k0** All atoms (incl. those marked by `*` in the 1st column in species.`mol`) are free and can move during minimization.
>
> **-k1** The *marked atoms are kept fixed; however, atoms with missing coordinates are always free to allow energy minimization.
>
> **-k2** Atoms with missing coordinates are free, known ones are kept fixed (marking is irrelevant).
>
> **-k3** If there is any missing atom, `-k2` applies, otherwise `-k1`. This is the default.
>
> **-k4** Read keep info from file species.`keep`. Should be combined with `-k1` to be active (`keep=1`), i.e., use `-k5` to read the keep info and minimize using it.
>
> **-k8** Read mark info from file species.`mark`.
>
> **-k-1 -k-2 -k-3** As above with the exception that single atoms are always free. Particularly, option `-k-3` is intended to initial minimizing a protein with charges compensated by counterions: protein skeleton is kept, unknown hydrogens and counterions are free (note that hydrogens are filled by `blend` while counterions have been exported from `pdb` and are inaccurate and may clash with hydrogens).

-mnumber Energy minimization (optimization). If number is positive, then number steps of the conjugate gradient method is performed. If number is negative, -number/4 steepest descent steps are performed and then -number steps of the conjugate gradient method. The default is `-m-100`.

-nnumber Number of molecules. Not used by `blend`, only passed via a ble-file to program `cook`.

-n-number If the argument is negative, `blend` tries to split a 'molecule' into submolecules (clusters). The format of output ble-file is changed and understood by `cook` as a configuration. PDB-file with crystal water or insulin consisting of two identical chains are typical examples.

BUG: The algorithm for determining whether two clusters are identical is simplified. Clusters having the same numbers and types of atoms and bonds on each atom are considered identical. Thus, clusters written with different order of atoms are not recognized as identical. On the other hand, the following two clusters will be incorrectly considered as identical:

```
A--B        A--B
|  |  and   \/
|  |        /\
D--C        D--C
```

WARNING: If the mol-file has been generated from a che-file, it may happen that numbers in atom ID's do not match the real numbering because sites might have been renumbered to have consecutive numbering of clusters.

TECHNICAL NOTE: since `blend` does not know the final number of species, it first writes `nspec=<number> !?` to the ble-file. After all molecules are processed, and if `-o` has been specified, it re-reads the ble-file and writes the correct final number of species in the form `nspec=<number> !fixed`. This does not work if `-o` has not been specified (i.e., you use option `-v0` and either `-s` with `$w` command or redirection of standard output).

`-p`number Write configuration in the playback-compatible format. Optional number > 0 means that number% of the van der Waals radii of atoms will be used as the radii of atom spheres (default is 70%). Two files, species`.plb` and species`.gol`, will be created.

`-p-`number As above with reversed endian of species`.plb`. To be used for transferring binary data between computer architectures using the same format of floating point numbers (e.g., IEEE) but different order of bytes (endianness, sex). E.g., x68 (IBM-compatibles) and Dec Alpha are little endians while Sun, SGI are big endians. If this does not help, ASCII data should be transfered (utility `plb2asc`).

`-q`number Scaling of charges in %. The default is 100%. Note that scaling is done on input, i.e., the scaled charges are written to output. At the same time, warning on fractional charges is suppressed. Hint: use `-q100` to suppress the warning and keep charges unchanged

`-r`number Read configuration option:

   `-r0` Initial configuration will be read from available configuration (tried in the order of types species`.plb`, `.pla`, `.3db`, `.3dt`), if not available then derived from 2D data in species`.che` (or deprecated species`.2db`, species`.2dt`), and if this does not help, then filled with random numbers. This is the default.

   `-r1` Initial configuration is random. No action is taken to prevent atoms overlap. Note that certain force fields (Busing) may fail for atoms very close together.

   `-r2` Initial configuration will be obtained from 2D screen coordinates. If the molecule is given by species`.mol` file, then species`.2db` (binary) file will be read, if the molecule is given by species`.che`, then the 2D coordinates are derived from it. See also option `-e`.

   `-r12` (Deprecated) If the molecule is given by species`.mol` file, then species`.2dt` (text) file containing screen coordinates will be used to determine the initial 3D configuration. See also option `-e`.

   `-r3` Initial configuration will be read from species`.3db` (binary) file. Note that format `.3db` is deprecated since V2.1a and replaced by `.plb`.

   `-r13` Initial configuration will be read from species`.3dt` (text) file.

   `-r4` Initial configuration will be read from the playback file species`.plb`. Special sub-options can be specified:

      `-r4:`frame Selected frame is read from the playback file. frame $= 1$ is the 1st frame, frame $= -1$ denotes the last frame, frame $= -2$ the second last, etc.

-r4:<u>from</u>:<u>to</u>:<u>by</u>[:<u>file</u>] For special functions (see options `-A -E -G -H -R`) as well as the PDB output (options `-w10, -w20, -w30`), the range of frames analyzed is specified. The default is [`-r4:1:-1:1`]. Parameter <u>file</u> must be given with extension (usually `.plb`); if omitted, <u>species</u>.`plb` is the default.
NOTE: for PDB output, see also `pdb` `-pFROM:TO:BY`.

-r-3 Initial configuration will be read from deprecated <u>species</u>.`3db` file with reversed endian.

-r-4 Initial configuration will be read from the playback <u>species</u>.`plb` file with reversed endian. Use for transfering data between machines with the same format of floating point numbers (now usually IEEE) but different order of bytes (endian, sex).

-r5 Alpha helix, old algorithm

-r6 (Added in version 1.6d, changed in 1.7k; see also `-k7` and `-k5`.) The initial configuration is alpha-helix. The molecule must be a protein (or peptoid, i.e., N-substituted glycine; the N-backbone may need finer minimization). The backbone is recognized by atom types (see Sect. 3.4.10). The alpha-helix configuration is generated and the backbone atoms are marked as to be kept during minimization (cf. option `-k2`). The side-chains are very approximate and thus the first step should be energy minimization which should not include the backbone: the keep status is set to `-k1` (see option `-k`. As the second step, the backbone atoms should be included in the minimization (use `keep=0` or re-run `blend`). Example:

```
blend -r6 -g -t10 polyval.che # creates alpha-helix
                              # + optimizes side-chains
blend -o xxx -g polyval       # optimizes whole config.
```

-r7:<u>phi</u>:<u>psi</u>:<u>omega</u> (Added in version 2.0k.) The initial configuration is described by dihedral angles <u>phi</u> (-C-N-Ca-C-), <u>psi</u> (-N-Ca-C-N-), and <u>omega</u> (-Ca-C-N-Ca-) (in degrees). If no <u>phi</u>, <u>psi</u>, <u>omega</u> are given, the default is <u>phi</u>=<u>psi</u>=-51, <u>omega</u>=180, which is approximate alpha-helix (to be optimized). If more molecules are processed, the default is the values of previous molecules.

-t<u>number</u> The range [C1,C2] for the cutoff switch function will be C1=<u>number</u>-1 and C2=<u>number</u>+1. The default is no cutoff (infinite range). See Sect. 3.5.

-t-<u>number</u> The range [C1,C2] for the cutoff switch function will be C1=int[<u>number</u>/100]/10 and C2=C1+(<u>number</u> mod 100)/10. E.g., -t-12345 gives C1=12.3, C2=12.3+4.5.

-u<u>number</u> All non-bonded pair energies (i.e., site–site [typically Lennard-Jones] and Coulomb ones) are recorded and first <u>number</u> highest terms are printed. Thus, possible atom overlaps are easily visible. If <u>number</u> is negative then first <u>-number</u> terms with lowest energy are printed.

-w Write configuration in some special formats:

-w Write configuration to <u>species</u>.`3db`. If this file already exists, the old one is backed up as <u>species</u>.`3db` (<u>species</u>.`3dx` for DOS). This was the default prior V2.1a; the 3db format is now deprecated—plb is recommended instead.

-w- -w0 Do not write configuration; does not apply to the plb format (use `-p0` to suppress writing <u>species</u>.`plb`). This is the default.

-w-1 As option -w with reversed endian.

-wdigit If digit > 1 then write configuration to ascii species.3dt. digit is the number of decimal digits printed. If this file already exists, the old one is backed up as species.3dt (species.3dx for DOS).

-w10[:Hstyle] Creates PDB file species.pdb using backbone analysis. To be used for che-file based peptides created, e.g., by `makepept`. BUGS: This is a simple code that may fail in complex cases. Does not recognize PRO as the 1st residue. Uses non-standard numbering of H and perhaps also other atoms. None or very limited support for residues like HEM, etc. Knows only some termini. Some problems can be solved by hand-editing the resulting PDB file.

-w10:0 Hydrogen on beta C is ' HCB1'

-w10:1 Hydrogen on beta C is ' HB1 ' (C as Carbon omitted). Hydrogen on the backbone N is always NH1 (N is not omitted).

-w10:2 Hydrogen on beta C is '1HCB '

-w10:3 Hydrogen on beta C is '1HB '

-w10:4 Hydrogens are omitted (default)

Can be used with option -r4 to create a series of PDB files.

-w20 Re-creates PDB file species.pdb using id names in mol-file, to be used for files created by the pdb converter. BUGS: 1 backbone only; changes order of atoms (in this case, try utility `pdb2pdb`, see Sect. 23.1.) Can be used with option -r4 to create a series of PDB files.

-w30 Selects -w10 or -w20 automatically

-w40 Write file species.atm with a header containing the number of atoms, a blank line, and then 1 line/atom in 4 columns At x[A] y[A] z[A].

-w80 Write a cfg-file in the `cook` format (V2.7a and newer. Cf. `plb2cfg`.

-w160 Write a cfg-file in the `cook` format (older than V2.7): deprecated.

The -w options may be combined (-w125 = .3dt in 5 digits + .atm + .cfg); however, .3db and .3dt cannot be combined (only one is written).

-ynumber Center molecule or create box. Sum of flags:

1   Center molecule before minimizing; box size (if contained in the plb-file) is not changed. Recommended to show a molecule.

2   Center molecule after minimizing and set box size (written to the .plb file) to zero (free boundary conditions). Note that cook with `init<0` reads only the contents from such a plb-file (the box size is derived from cook initialization).

4   Make all coordinates positive, incl. Lennard-Jones sigma. This means that min and max in all coordinates are determined and the molecule is moved so that all coordinates are positive.

8   Box molecule after minimizing, incl. Lennard-Jones sigma. Maximum of all coordinates is written as box size to the plb-file. Recommended with -y4. Note that cook with `init<0` reads such a plb-file including the box size.

16   Modifies -y4 and -y8: do not add Lennard-Jones sigma to coordinates (may lead to overlaps in periodic b.c.). (To modify the box size, use `plbbox`).

The default, suitable for making a blend-file, is `-y3`. To work with a configuration in periodic b.c., use `-y13`

2.2.4   Extra-options

`-A` Generate species.ang of independent internal coordinates.   File species.plb (and species.mol) will be read (see option `-r4`) and the following files will be generated:

species.ang protocol about angle analysis

species.1, species.2 ... angle files

Option `-A` implies `-w0,-p0` (no write), `-m0` (no minimization), and `all_dihedrals=0` (one dihedral per bond).

The angle analyzing algorithm takes into account only the following structures:

`3- and 4- bonded atoms` 1 dependent angle is removed, 2 angles and 1 improper are left

`6-cycles` of 6 dihedrals and 6 angles, 6 are removed

`5-cycles` of 5 dihedrals and 5 angles, 6 are removed

`merged rings` 1 additional improper removed

`NOT SOLVED` other cycles than 5- and 6- (like if there are CYS-CYS bridges); complex merged rings (like in the heme)

Actually, the topological analysis to get only independent angles is not easy. The formula 2*ns-5 works for trees only (no cycles). WARNING (?): tested with CHARMM21 only.

`-C`number `(-1)` Reference atom for dipole and quadrupole calculations, and for the second virial coefficient (see `-V`; in this case, both molecules must have the same atom as the reference). Note that the dipole value depends on the reference point for ions only, and similarly the quadrupole value depends on the reference point for dipolar molecules.

number`>=0` Atom number.

`-1` The reference point is the centroid of atomic charges (default).

`-2` The reference point is the center of mass.

`-3` The reference point is the geometric center.

`-D` File species.msd of mass-weighed MSD (mean square displacement) is calculated. Implies `-w0,-p0` (no write), `-m0` (no minimization).

`-E` Essential dynamics. The essential dynamics (ED) algorithm analyzes the trajectory (in playback file, see option `-r4`) and by diagonalizing the covariance matrix calculates the "essential motions". Implies `-w0,-p0` (no write), `-m0` (no minimization).

`-EA` use all atoms in ED

`-EH` use all heavy atoms (not hydrogens)

`-EC` use only C-alpha (cheapest but usually sufficient)

`-EB` use all atoms in the backbone (omits all sidechains)

It is assumed that atom IDs have been derived from a PDB file.

Algorithm:

1. Reads frames and matches frames with the 1st frame which means that r^N is rotated and displaced so that $\sum(r_i - r_{0i})^2$ is minimum (simple MC minimization).

2. Calculates the covariance matrix $\mathrm{Cov}(r_i, r_j)$ (all coordinates separate, i.e., rank=3*#_of_atoms).

3. Calculates its eigenvalues.

More about ED:

- Essential dynamics files with extension `.ess####.plb` use the average frame as the center

- This average frame is also written as `essr0.plb` (if `-F#` is odd, this is the same is the central frame)

- The amplitude of essential motion in `.ess####.plb` is the same as to reproduce the stdev (mean square amplitude), i.e., cos-amplitude is sqrt(2)*stdev and linear /\/\/\ amplitude=sqrt(3)*stdev (if the essential motions were harmonic or /\/\/\ , respectively, the `ess####.plb` would show the correct motion). This holds for `-M1` (the default), `-M#` just multiplies the motion by #

- File `.ess.cp` is written containing development of coordinates in the basis of eigenvectors. There are `-P#` items (columns) in `ess.cp`. Use `showcp -a` or `showcp -p` etc. to analyze `ess.cp`, ignore bad headers of columns 0 (Etot) and 1 (T)

- the matched trajectory (with rotations and translations removed) is written to `essmatch.plb` if option `-P`. `blendess.c` must be compiled with `#define WRITEMATCH`.

  Note: try `blend/esstest.c`

-F<u>number</u> For essential dynamics and normal mode analysis. Number of frames in the output playback files used to visualize the motion. Positive <u>number</u> denotes harmonic motion, negative denotes linear motion between min and max. Cheap: use `-F2`, better: `-F7` or so. See also `-M` and `-P`. Of course, `-F-2=-F2`, `-F-3=-F3`.

-G Calculate eigenvalues of the gyration (`-G`) matrix (see below)

-H Calculate eigenvalues of the inertia matrix. Three files are produced (for `-G` they are `gyration.cp`, etc.)

  `inertia.cp` Convergence profile of 7 items: I_{xx}, I_{yy}, I_{zz}, R_x, R_y, R_z, R_g.

  `inertia.mol` Mol-file file for `show` to visualize `inertia.plb`. 'inertia' is viewed as a 'molecule' of 6 atoms

  `inertia.plb` Playback file for 'inertia'. The following Cartesian cross or 'molecule':

```
           O
           | O
           |/
   O----+----O
          /|
         O |
           O
```
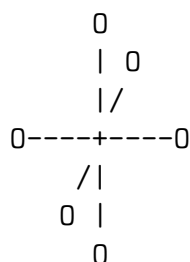
where the masses of 'atoms' O are 1/2 and the lengths of axes (O–center) are R_x, R_y, R_z, has the same tensor (matrix) of inertia as the original molecule

BUG: The sign of both vectors in each O–O 'bond' is undefined. This does not cause any problems while showing, but might cause problems in further calculations

Theory: The gyration or inertia matrix is calculated first

$$\overset{\leftrightarrow}{G} \;=\; \sum_i m_i \Delta\vec{r}_i \Delta\vec{r}_i \tag{2.1}$$

$$\overset{\leftrightarrow}{I} \;=\; \sum_i m_i \left[\Delta\vec{r}_i^{\,2} \overset{\leftrightarrow}{\delta} - \Delta\vec{r}_i \Delta\vec{r}_i \right] \tag{2.2}$$

where $\overset{\leftrightarrow}{\delta}$ is the unit tensor (matrix) and

$$\Delta\vec{r}_i = \vec{r}_i - \vec{r}_{\text{center of mass}} \tag{2.3}$$

The diagonalized tensor is

$$E = U(I \text{ or } G)U^{-1} = \begin{pmatrix} E_x & 0 & 0 \\ 0 & E_y & 0 \\ 0 & 0 & E_z \end{pmatrix} \tag{2.4}$$

where U is a unitary matrix. In the same spirit of "radius of gyration", the following "partial radii of gyration" R_a are defined from diagonalized G (and similarly I, if this makes sense).

$$R_a = \sqrt{E_i/m_{\text{tot}}}, \tag{2.5}$$

where m_{tot} is the total mass of the molecule. Finally,

$$R_{\text{gyr}} = \sqrt{R_x^2 + R_y^2 + R_z^2} \tag{2.6}$$

For diagonalized I, the principle moments of intertia vs. axes are $E_y + E_z$, $E_z + E_x$, $E_x + E_y$.

-J*number* Essential dynamics and normal mode analysis: accuracy for the diagonalization (Jacobi method)

Essential dynamics: sqrt(*number*)*0.2 is used as the error threshold for configuration matching when the configurations are translated and rotated to match best each other (before the covariance matrix is calculated)

-M*number* For essential dynamics and normal mode analysis. Amplitude of the visualized motions. The units are Å for normal mode analysis, and standard deviation of the essential motion for the essential dynamics. The default is 1 (Å or standard deviation).

-M-*number* For normal mode analysis, the same as *number* $\cdot$ `ns`$^{1/2}$. `-M-0.2` gives a reasonably good amplitude irrespective of molecule size.

-Nnumber Calculate the normal mode frequencies (vibrational spectrum). If number is given, it denotes dr for numerical calculation of second derivatives; default `dr=1e-5` Å. File species.`nmf` is created. See also `-M`, `-P`, and `-F`. See Sect. 14.9 for the theory.

The input configuration should be well minimized: use `-mBIG_NUMBER` or `cg=BIG_NUMBER` and wait until the minimization stops; to be on a very safe side, repeat once more with `-m-100` or `sd=100`. The output file species.`nmf` contains frequencies ν in THz and wave numbers in cm^{-1}, ordered.

If the eigenvalue is negative (saddle point—not enough minimized or numerically imprecise), then ν printed is negative, too.

Accuracy: If the step $\Delta r \approx$ (machine precision)$^{1/3} \approx 10^{-5}$, then U' is known with precision of 10 digits. Six (for a general molecule) eigenvalues corresponding to rotations and translations should be zeros $\Rightarrow$ they will be calculated with relative (to other eigenvalues) precision of about 1e-10. Since the frequencies are square roots of eigenvalues, they will be given with relative precision of 1e-5. Consequently, frequencies slower than 1e-5 of the fastest ones, i.e., slower than about 1 GHz, cannot be calculated.

Diagonalization method: threshold Jacobi method

Example (especially didactic): Create file `biphenyl.che`:

```
  biphenyl

             HA      HA^
              |       |
    HA-C6R--C6R    C6R^--C6R^-HA^
        |    |      |      |
    HA-C6R  C6RP--C6RP  C6R-HA
        |    |      |      |
    HA-C6R--C6R    C6Rv--C6Rv-HAv
              |       |
             HA      HAv
```
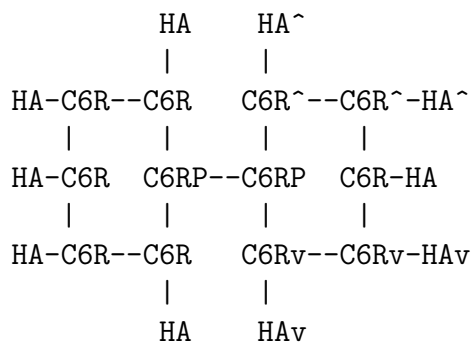
then do the following:

```
  blend -p -N -M.4 -P99 -F7 -m-9999 biphenyl
  (biphenyl.nmf and biphenyl.nm0000.plb, ... created)
  show biphenyl biphenyl.nm0006
    rotate the molecule (left mouse button)
    type hot keys: $iwwsss
```

You can also use script `nmf.sh biphenyl`, then (from `blend`) click button [finish] or hot key $\boxed{.}$. Use buttons `<` or `>` or hor keys $\boxed{\text{PgUp}}$, $\boxed{\text{PgDn}}$ to switch modes.

BUG: Cannot be used for models with zero masses of some particles (as TIP4P water)

-P<u>number</u> For essential dynamics and normal mode analysis. Number of playback files generated. <u>number</u> of the most significant eigenvalues is considered (-<u>number</u> least significant if <u>number</u> is negative). Files have extensions `.ess####.plb` or `.nm####.plb`, where #### is 0000, 0001, ... Use also 'blend -p' to get the gol-file, then `show` to view the motion. The default is `-P0` (no playback files).

-R<u>number</u> Core size for the second virial coefficient (see `-V`). The center of this spherical core is given by `-C`.

-S<u>number</u> POLAR ONLY: maximum number of self-field iterations before an error is printed. The default is 200. With virial calculation (see `-V`), divergence is treated as infinity potential. Since the internal accuracy is 1e-8, this means that the convergence rate must be better than 1e-8$^{1/200}$ = 0.91; increase `-S` if you have such an ill-defined model. On the other hand, you may decrease `-S` if you are sure the convergence is fast enough. Also, the number of SHAKE iteration. Was `-I` before V2.2h

-T<u>number</u> Temperature (in K) for the second virial coefficient (see `-V`).

-V<u>number</u> Number of random samples for the second virial coefficient. Negative value means that twice as many samples are used and every second is generated with molecule 1 reflected. Use this if both molecules are dipolar.

Caveat: both molecules are treated as rigid (after separate optimizations).

To calculate the second virial coefficient, use 1, 2, or 3 mol-files as blend arguments:

1   The second virial coefficient of pure compound is calculated.

2   The cross second virial coefficient (for a mixture) is calculated.

3   For development purposes: the third argument must correspond to a mol (or che) file with both molecules merged (in the same order of sites).

An example of the 2nd virial calculation at 373 K is:

```
blend -C1 -V-1000000 -T373 spce
```

The algorithm is based on the formula:

$$B_2 = \frac{2\pi}{3}R^3 - 2\pi \int_0^{1/R} \langle f(1/x, \Omega)\rangle_\Omega 1/x^4 \mathrm{d}x \tag{2.7}$$

where $f(r,\Omega) = \exp(-u(r,\Omega)/k_\mathrm{B}T) - 1$, Ω, stands for all angles, and R is the core size (option `-R`; the potential can be assumed infinity inside the core).

First, the configuration is split into molecules. Both are minimized separately (in two steps, with the other molecule shifted far away).

Then, the above integral is calculated by the naive Monte Carlo (random shooting) If the `-V` option has a negative value, every second configuration is created instead by inverting ($r \to -r$) the first molecule. This is necessary for both molecules dipolar because the integrand is then limiting to a constant as $r \to \infty$ (otherwise it would be infinity and the integral would be unsuitable for the MC method).

Works with polarizability, too; note that a diverging self-field is treated as infinity potential. (See `-I` for the number of iterations.)

2.3   File extensions

File names of all molecule-related files are derived from species or species.mol or species.che specified at command line. A list of extensions follows. '(i)' means that the file is can be read by blend, '(o)' that it is created or rewritten.

Suffix ~ denotes a backup (previous file before rewriting).

.2db (i) DEPRECATED, do not use! 2D screen configuration (binary). Produced by MEDIT (?).

.2dt (i) DEPRECATED, do not use! 2D screen configuration (2 column x-y ASCII file). Produced by MEDIT (?).

.3db (io) DEPRECATED, rather use .plb instead. 3D configuration (binary file). It consists of number_of_atoms vectors, each consisting of three single precision floating-point numbers containing x-y-z.

.3dt (i) 3D configuration in ASCII (3 column x-y-z). Deprecated.

.che (i) Chemical format. See Sect. 4.2.

.edt (i) Edit molecule file. See option -e.

.dep Dependant info file, with lines of the form dependants : base sites. See Sect. 5.10.

.keep Info on atoms kept (frozen) while minimizing. Can be written by hot key $\boxed{@k}$, read by $\boxed{@K}$ or option -k4. The file format is simple:

```
! comment
2 ! atom number
3 ! atom number
```

.mark Info on marked atoms. Can be written by hot key $\boxed{@m}$, read by $\boxed{@M}$ or option -k8. The file format is the same as for .keep

.pch (i) Partial charge replacement file, must be in the current directory or in the directory given by BLENDPATH. If present in the input line, all patterns for groups of atom types replace the respective patterns found in the molecule, overriding any charge specified in the .che or .mol files. Mol-file will be rewritten with the new charges. Example of the pch-file:

```
! >C=O
C=0.55      O=-0.55

! -NH2
NT=-0.3     H=0.15      H=0.15
```

The first atomtype=charge item on line is the 'central' atom, all other atoms are connected by bonds. Ordering of atoms but the first one defining the center is irrelevant. The charge assignment algorithm takes sites in a molecule one by one and for each tries to match all possible groups of this site and its nearest neighbors to patterns in the .pch file. If there are several matches possible, the first one applies; thus, the following trick is possible:

```
CT=-0.4 HA=0.1 HA=0.1 HA=0.1 HA=0.1
CT=-0.3 HA=0.1 HA=0.1 HA=0.1
```

Using this, the first line will be used for (full-atom) methane and the second one for ethane.

If an atom is a member of several groups, its charges is calculated as a sum over all group contributions and info is printed.

`.pro (o)` Results of probing. See 2.2.3, option `-e`, commands `pa` and `pw`. See also utility `showpro` (**??**).

`.rea (i)` Chemical reaction file. Performs a special form of 'chemical reaction' on groups defined by reaction patterns. Can be given also by command 'react' during editing. Example: to perform the following reaction

```
         O                   OT--H
        /                   /
  HP + -C          ->     -C
        \                   \
         OAC                 OA
```
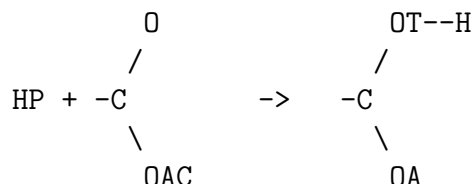
the rea-file should contain a line like this:

```
HP + O <1.2 *C OAC += H-.6 OT+.35 C+.05 OA+.2 > -100
```

- `<`   the reaction takes place if the distance of HP and O is less than the distance given by the '<' command (here 1.2 Å, default = previous line, if nothing given = 1 Å)

- `*`   *C means that C is the central atom of the group (if no * is given, 1st atom after '+' is central)

- `+=`   += means that partial charges (numbers after atom types) will be added to already present partial charges

- `=`   = instead of += means assignment of partial charges

- `>`   energy of the reaction follows (negative=exothermic). This energy is written to the mol-file as the 'zero_energy' field.

- `\`   EXTRA lines: if the charge redistribution extends over one group, 'continuation' lines may be added. They start with '\', e.g.

  ```
  \C5RE + N5R H C5R += C5RE+.05 N5R+.1 H+0.05 C5R+0.05
  ```

  Here, '+' does not denote a reaction but an atom that has been marked in previous reaction step (=line without '\' + any number of lines with '\'); it must be bonded to the next atom (here C5RE–N5R) which itself must not be marked. Marking of other atoms in the group is irrelevant. Note that any such marking is canceled by a new reaction (=line without '\'). The above example changes charges in the group:

  ```
  H--N5R---C5R
         /
  C5RE
  ```
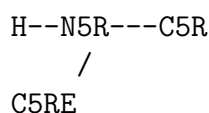
  where C5RE has been affected by a reaction but N5R not

`.geo (i)` If this file is present, `blend` prints selected distances and angles on hotkey 'E' and after minimizing is finished. species`.geo` is just file of lines with sites of bonds or angles

```
! comment
1 2 # bond 1-2
1 2 3 # angle 1-2-3
```

The comments are printed, too.

BUGS: very ugly implementation, should unify with 'clicking', should add dihedrals...

`.gol (o)` DEPRECATED. File containing a 2-line header followed by colors and radii of atoms. Used by `show`. Originally the goal file `goal.fil` for `PlayBack`. Generated by `blend` prior V2.3, removed since blend V2.4a and show V2.2a. (Show still uses gol-file if one is present).

`.jet (i)` File determining additional constraints. See Sect. 2.2.3, option `-j`.

`.mol (io)` Molecule description file. See Sect. 4.1.

`.plb (io)` 3D configuration (binary file), See Sect. 21.1 for the format. The recomended format in MACSIMUS.

Note that `blend` normally uses the free (vacuum) boundary conditions, other format options are used by `cook` and `show`, etc.

`.sym (i)` File defining some symmetry conditions to be kept while minimizing. Very primitive. Turned off if `-N` specified. Example:

```
z 0 2
z 3-4
z 5
.
```

`z 0 2` means that sites 0 and 2 are kept symmetric with respect to z=0 plane.
`z 5` means that site is kept in z=0 plane.
`z 3-4` means that sites 3 and 4 are kept in z=0 plane." . stops
Instead of x,y,z, one may use X,Y,Z or 0,1,2.

Other extensions (not related to individual species):

`.par (i)` Parameter file in text format, typically in directory `blend/data` (pointed to by environment variable BLENDPATH).

`.bin (io)` Its binary image, deprecated.

`.ble (o)` Default extension for the output file.

2.4   Run-time control

If no option `-o` is given, `blend` runs in the *interactive mode.* The user can interactively control the course of minimization and respond to error messages. The output is written to `stdout`.[1] (Do not confuse with key and mouse control in the graphical window!)

If option `-o` is given, only few messages are printed to `stderr`. The only user interaction is typing Ctrl-C (see Sect. 2.4.4).

2.4.1   "get data" format for input

The "get data" format resembles the `GET DATA` statement in the old good PL/1 or `NAMELIST` in FORTRAN. It is used as user-friendly input format in several places.

The input data consist of assignments of the form <u>variable</u>=<u>numeric expression</u>.

A set of such assignments is ended by a semicolon.   The <u>numeric expression</u> may contain numbers, variables (qualified as `a.b`, but not indexed arrays), operators `+ - * / ^ ( ) # %` and elementary functions, where symbol `^` denotes exponentiation and `#` the result of previous assignment. Operators `+= -= *= /= ^= %=` are also allowed if `#` is not used.

If used interactively, some additional commands are available so that you can use the "get data" module as a calculator.

Example:

```
  ran=1/3   ! this is comment
  by *= 2;  ! by is doubled (WARNING: not allowed in cook's
            ! def-file because it is read twice)
! x = L[2]  ! indexed variable not allowed in the r.h.s
```

List of commands:

`?`   Get help

`??`   List all variables.

<u>expression</u> Evaluate expression and store it into `#`

<u>variable</u>=<u>numeric expression</u> Evaluates expression and assigns the result to <u>variable</u>

`?`<u>variable</u>=<u>numeric expression</u> Evaluates expression, assigns the result to <u>variable</u>, and prints the result

`?=`   Toggle auto echo mode (NOTE: this is useful in case of errors during input, as wrong identifier or syntax)

`?`<u>format</u> Set output format (C-style), e.g. ?%.15g is g-format with 15 digits

`!`   Comment to the end of line

`;`   End of data block

`$`<u>command</u> To enter scrolling (see below)

---

[1]DEPRECATED: If option `-s` is also given, a user can watch several last screens of output.

2.4.2   Scrolling

Deprecated feature from the old times of simple 80x25 terminals. It is activated by option `-s#`, where # is the buffer size in kiB.

Scrolling is called from the "get data" module by $ and a command. Then, prompt $ appears; it is not repeated in next scroll commands. Scrolling is also available if an error occurs. Standard screen of 25 rows and 80 columns is the default screen.

A list of scroll commands follows:

`$?` `$h` Get help

`$u` Scroll by one screen up

`$number` Number lines down (up if negative)

`$-` One line up

`$t` `$g` Go to the top of stored screens

`$d` `$SPACE` Scroll by one screen down

`$b` `$G` Go to the bottom of stored screens

`$w`file Write whole buffer to file file

`$i`file Include file as if typed in

`$/`string Find string

`$/` Repeat last find

`$L`number Set number of lines on screen

`$C`number Set number of columns on screen. Lines longer than this limit are truncated; if you see garbage on the screen, try to re-set the number of columns. WARNING: Some screens have that bad habit of eating ends of lines longer than number of characters/line which does not fit the scroll's assumptions. If this is the case, set C to a large number.

`$!`number Set clear screen mode (try if the screen clears badly or slowly)

`$Q` Emergency quit (=exit(1))

`$.`   `$q` Return control to the "get data" module or error handling

`$exit` Kill program (nothing saved!)

2.4.3   Error handling

Error messages (Warnings, Errors and Disasters) are self-explanatory and contain also the name of the source file in C and the line number. They are printed both to `stderr` and the file defined by option `-o` (or `stdout` if not specified); the latter contains more information needed for analyzing the source of the error.

When an Error occurs in the batch mode (option `-o` file), the program exits with a nonzero return code. When a Warning occurs, message is printed and execution continues.

When either an Error or Warning occurs in the interactive mode, the user is prompted for an appropriate action (by typing a letter followed by RETURN).

`(a)bort` Abort (with a core dump if configured).

`(e)xit` Exit without a core dump (see above for the return code).

`(c)ontinue` (Warnings only) If you wish, it is safe to continue.

`(i)gnore` Ignore the Error and continue. **Your risk** – it is not guaranteed that it will not crash later.

`(s)croll` Use scroll (see above). Active only if option `-s` has been specified.

2.4.4   Interrupts

You can interrupt energy minimization by pressing Ctrl-C ; if the course of minimization is shown graphically, then ESC is preferred (in the X11 environment, ESC must be typed in the graphical window).

In the interactive mode (no option `-o`) you will get back to "get data".

In the batch mode (option `-o` file) just the current minimization is interrupted; type Ctrl-C twice to break the program.

Under UNIX, the interrupt is caught immediately and you only have to wait until one minimization step is finished. Under DOS, you must wait until the first I/O; for some extenders and Windows no Ctrl-C is accepted at all.

2.5   Showing molecules graphically

`blend` generates output mol- and plb-filed for program `show`[2]

2.5.1   X11 Graphics

`blend` since V2.3 cannot be compiled without graphics support, but it runs without graphics unless option `-g` is used.

---

[2]Its precessor was `PlayBack` written by Niels Jacobsen; then show in DOS/Turbo C followed. Later, `moil` by C. Simmerling, University of Chicago, was ported via `tomoil`.

If `blend` runs with option `-gnumber`, the course of minimization is shown each number-th step. Pressing ESC interrupts the minimization and you can watch the molecule, rotate it, etc. From the interactive input for minimization, graphics is called by entering `gr=1;` (bonds only, fast), `gr=2;` (balls-and-sticks, slower), `gr=3;` (shaded balls, slow), or `gr=4;` (anaglyph – blue-red glasses needed).

On-line help is obtained e.g. by pressing `F1`, context help on buttons by right-click.

Mouse:

`drag left button` Rotate molecule (or marked atoms in the 'move' mode) around x and y axes

`drag middle button` Move molecule (or marked atoms in the 'move' mode) in x and y directions

`drag right button` Rotate molecule (or marked atoms in the 'move' mode) around z axis

`wheel` Resize (rescale)

`click left button` Print info on the selected atom and mark the atom (e.g., for moving). Second click on any atom gives atom-atom distance, third click gives the angle. Does not work in the stereo (anaglyph) mode (gr=4).

`click middle button` Flood-mark, i.e. marks all atoms connected by bonds until any other marking. See also hot key M . To mark a group, left-click first the nodal atom(s) (which connects the group to the rest of the molecule), then middle-click any atom in the group.

`click right button` Unmarks the atom.

Hot keys   NB: may be somehow out-of-date because the current GUI uses buttons, although all of them are also accessible by hot key. Some less common functions are accessible by hot keys only.

`cursor arrows` Move the molecule in x and y directions; if the 'move' function is active (see yellow info bar in the bottom of the screen), the marked atoms are moved.

Home End Move the molecule in z direction, i.e., move the eye for central projection

x X y Y z Z Rotate molecule around given axis. Lowercase rotates in positive direction (anticlockwise), uppercase in negative direction.

w Write (remember) molecule position (in fact, erases a list of translations and rotations)

W Undo all translations and rotations (but those made with parts of a molecule) W and w are intended for editing a configuration which can be pasted back to `cook`. Example:

```
blend -o SYS A B C                # prepare SYS.ble
cook SYS                          # simulate+write SYS.plb
molcfg A B C SYS                  # generates SYS.mol
blend -o SYS -n-1 -g -y0 -r4:-1 SYS # move molecules, then type W .
```

NOTES: If original molecules A B C consist of non-connected parts, file `SYS.def` (variable `N[]`) will have to be changed. Option `-y0` to blend is essential. Option `-n-1` is required in the periodic b.c. These are not treated properly in blend so that `-m0` or `-mSMALL` is needed. Then, `-tSMALL` is recommended. **BUG:** Inefficient implementation, memory consuming (problems with DOS version)

`+` `-` Resize (rescale) the screen. Do not confuse with `Home` `End`!

`*` Double step size for the above functions

`/` Halve step size for the above functions

`r` `R` Decrease/increase radii of atoms (draw styles 2 and 3 only)

`1` Show bonds only (fastest)

`2` Stick-and-ball model

`3` Stick-and-ball model, spheres are shaded (slowest)

`4` Anaglyph, bonds shown in blue and red for the right and left eye, respectively. To view in 3D, red (left) and cyan (right) glasses are needed

`g` `G` Toggle among the above four graphics modes

`i` Label atoms by atom identifiers. For proteins taken from PDB database, this is something like `ALA12CA`

`t` Label atoms by atom types

`n` Label atoms by atom numbers

`q` Label atoms by (partial) atomic charges

`l` Toggle labeling style

`f` Find atom. One of the above labeling styles must be set. For atom identifiers and types, wildcards `?` and `*` are accepted, however, `*` works only as the last character. For charges, you must enter two numbers, charge value and inaccuracy. Thus, e.g., `0.3 0.05` will find all charges in the range `[0.25,0.35]`.

`F` Cancel finding, i.e., label ALL atoms again

`d` POLAR only: Show induced dipoles in eA

`D` POLAR only: Show induced dipoles in Debye

`p` POLAR only: Show polarizabilities in $\mathring{A}^3$

`s` POLAR only: Show saturations (in the saturated polarizability model only)

`SPACE` Cancel any labeling and redraw

`m`   Toggles the 'move' function. Atoms to be moved must have been marked: click the atom centers by mouse or use the 'find atom' function (hotkey `f`), alternatively use hot keys `M` and `u`. The selected atoms are moved either by cursors and hot keys `x` `X` `y` `Y` `z` `Z` or by the mouse. Rotations twist around the least clicked atom (=in green) as the center. Type `m` again to leave the moving mode; hot keys like `.` `ESC` `,` cancel moving. **BUG:** There is no 'undo'.

`L`   Mark all labeled atoms (see hot key f)

`M`   Flood mark. Marks all bonded atoms connected by bonds until an already marked atom is found. Groups can be readily marked. Equivalent to middle button click.

`u`   Unmark all marked atoms

`I`   Invert marking (swap marked ↔ not marked)

`k`   Show keep status. Atoms kept while minimizing (cf. option `-k`, variable 'keep', and `*` in the 1st column of the mol-file or in front of atom id in the che-file) are marked.

`K`   Inverse to hot key k: Will keep the marked atoms while minimizing. In addition, "get data" is forced after `ESC`.

`A`   Average charges of marked atoms and unmark. Useful, e.g., for editing quantum-mechanical charges: mark a group of equivalent atoms (e.g., all three hydrogens in a -CH$_3$ group), then press `A`.

`b` `B`   Recolor bonds, cycles through 9 preferably light colors: yellow (default), white, gray, dark gray, blue, green, cyan, red, magenta

`c`   Recolor carbons, cycles through 3 colors: cyan (default), gray, dark gray.
      BUG: Dump screen functions fail for dark gray.

`#` `=`   Toggle none/ 1 Å / 0.1 Å grid (graphic modes 2 and 3 only).

`'`   Cycles through different styles of showing atom centers (graph modes 2,3 only) or turns centers off.

`e`   Erase/draw-over toggle. Useful while minimizing to see conformational changes. The default is writing over with gradual shading off old positions, first value is 'animation' (old molecule is erased, new drawn), second value is writing over (without shading, fastest option).

`P`   Printscreen. A menu appears with further options:

   `p`   PPM (raw Portable PixMap, P6) dump of screen to file `blend000.ppm`; further dump is `blend001.ppm`, etc. Colors are inverted for draw styles (graphic modes) 1 and 4, with the exception that the background (black on screen) is always white. Bug: dark gray for carbons (see hotkey c) does not work well.

   `P`   PPM on black background.

   `e`   Color EPS on white.

   `E`   Color EPS on black.

`c`   Color PS on white.

`C`   Color PS on black.

`o`   Grey PS on white. Colors are translated using `0.3R+0.59G+0.11B`.

`O`   Grey PS on black.

NOTE: for fancy pictures, use `show` and `ray`.

`E`   Calculate Energy, dipole moment, quadrupole moment, and geometry (according to the `.geo` file). The quadrupole moment tensor Q is defined by

$$Q = \frac{1}{2} \sum_i q_i (3\vec{r}_i \vec{r}_j - r^2 I) \tag{2.8}$$

where I is the unit tensor. Tensor Q is then diagonalized.
NOTE: This form is common in chemistry. In physics, it is used without factor 1/2.
NOTE: the diagonalized tensor is not ordered and the molecule is not rotated into the principal axes. To get also the non-diagonalized tensor, use option `-v` (verbose).

`ESC`   Escape showing: if option `-s` has been specified, you may enter data and continue interactively.

`,`   Minimize again (conjugate gradients); if cg=0, then cg=1 is used. The same as `ESC` and ; `Enter`

`;`   As above with preceding steepest descent minimizing (with sd=cg, or `sd=1` if `cg=0`)

`'`   Toggle center points (when atoms are shown). (The center points help clicking.)

`Q` `Ctrl-Q`   Kill blend (nothing saved!)

`F1` `h` `?`   Display help for hot keys

2.5.2   Playback output

Output in the playback-compatible format is generated by default. Option `-p`number means that number% of the van der Waals radii of atoms will be used as sphere radii (the default is 70%).

Playback files with long trajectories are generated by `cook` and can be also read in by `blend`.

To get technical info on the format of the playback file, run

```
plbinfo +t
```

For more about available utilities, see Sect. 21.

2.6   Energy minimization

Energy minimization (optimization) module implements two common methods: the greedy (steepest descent) gradient method and the conjugate gradient method.

The steepest descent method simply moves by certain step in the direction of forces; the step size increases in the case of success and decreases on failure. The conjugate gradient method is the Polak-Ribiere version [2]; the algorithm to find a minimum along a line was optimized so that in most cases only two evaluations of forces are required by one conjugate gradient step.

The steepest descent method is usually less efficient than the conjugate gradient method. It is recommended for initial high energy configurations with atom overlaps. Once energy drops, the conjugate gradients should be used.

The force field used by this minimization is the full force field including bond vibrations which may be replaced by fixed bond-length constraints in simulations. Options `-a0` `-b0` to contrain bond lengths in `blend` are not reliable.

The energy minimization is called by `-m` option. If the molecular configuration is to be generated from random input, `-r1` is used. There is no guarantee that the resulting configuration is close to global energy minimum, a locked high-energy configuration may arise for complicated molecules; then, button [ran] or hot key |:| will randomize the coordinates before another cycle of optimization.

In the interactive mode, the user is after one minimization cycle prompted for entering new data. (See Sect. 2.4.1, for the data format.) On-line help is obtained by entering

```
help=1;
```

2.7   Missing coordinates

The initial configuration can have missing atom coordinates. These appear as blank lines in input text files or as values bigger than 9e5 in the binary input files. `blend` calculates approximately the missing coordinates by using known positions of bonded atoms; if it has not got enough information, a random position is generated. The configuration should be energetically minimized, preferably with the atoms with known coordinates kept in place (see option `-k`). Useful to fill missing hydrogens.

Chapter 3

Force field and the parameter file

Program `blend` first reads in the parameter file (or its binary image); while reading the file, various checks (e.g., on duplicated entries) are made.

Then, `blend` analyses the structure of the molecule and finds both bonded and non-bonded energy terms. It looks in the tables to assign the potential parameters. The rules that apply as well as the format of the parameter file are described in this section.

3.1   Structure of the parameter file

The parameter file is recognized by extension `.par`. Its format is close to the format provided by CHARMM. Its general structure is:

```
<options>
<table "atoms">
<table of site-site terms>
<table "bonds">
<table "angles">
<table "dihedrals">
<table "impropers">
<table "nbfixes">    [obsolete name: <table "NBFIX">]
<table "polaratoms"> [POLAR only]
<table "polarbonds"> [POLAR only]
<table "waters">
<table "backbone">
```

The exclamation mark ! denotes a comment.

Each table begins with a header (single keyword on a line) and ends by a blank line; hence, blank lines are not allowed within tables (any lines without a header after a blank line are omitted). Detailed explanation is given below.

3.2   Force field generation options

Example of table `<options>` is:

```
all_dihedrals=0    ! one dihedral per central bond
ar_dih_limit=1     ! inclusion limit for dihedrals in aromatic rings
all_impropers=0    ! one improper per central atom
column_X=0         ! column X disabled (all atoms match = all X=1)
comb_rule=0        ! no sqrt combining rule for LJ energies
LJsigma=0          ! vdW radii used
factor14=0.5       ! multiplicative factor for 1-4 interactions
distance14=4       ! 1-4 interactions are really 1-4
ar_14_limit=0      ! 1-4 also for aromatic rings (use 3 for GROMOS)
polar=0            ! polarizability not included
;                  ! end of data set
```

The assignments may be in any order and must end by ; (see Sect. 2.4.1).

Description of options:

all_dihedrals Number of dihedrals generated.

> all_dihedrals=1 All possible dihedrals are generated for each central bond. Thus, ethane has 9 dihedrals. Used in newer versions of CHARMM. (This is the default if all_dihedrals is missing).

> all_dihedrals=0 Only one dihedral is generated for each central bond. Thus, ethane has one dihedral. Required by old versions of CHARMM. The rules which dihedral from generally several possibilities to select are mainly aesthetic. It is preferred that the four atoms of the dihedral be part of a longer chain.

> all_dihedrals=-1 No dihedrals nor aromatics are generated.

> all_dihedrals=2,3 As all_dihedrals=0,1 and dihedrals with multiple periodicities are shifted (i.e.m absolute term K[0] is set) to give Emin=0.

ar_dih_limit Maximum aromaticity of dihedrals. The default is ar_dih_limit=-1 which turns off special treatment of dihedrals in aromatic rings. See Sect. 3.8.7, for detailed explanation.

ar_14_limit Controls 1–4 interactions in aromatic rings (see Sect. 3.8.7). Implemented with distance14=4 only.

> ar_14_limit=4 The 1–4 interactions of atoms that are all in an aromatic ring are excluded (their nonbonded interactions are zero).

> ar_14_limit=3 As above and also atoms attached to the aromatic ring do not interact with the aromatic atoms (X does not interact with B but X and Y do interact).

```
      A
     / \
    B    F-X
    |    |
    C    E-Y
     \ /
      D
```

> ar_14_limit=2 As above and also atoms attached to the aromatic ring do not interact themselves (X and Y do not interact). This is used for GROMOS96.

ar_14_limit=0 No special treatment of aromatic atoms. A and D, X and B, and X and Y in the above molecule interact by 1–4 interactions. This is the default.

all_impropers   all_impropers=-1 No impropers are generated.

all_impropers&1 = 1 All three impropers are generated for each central atom.

all_impropers&1 = 0 Only one improper is generated for each central atom (default). The rules which improper from three (or six if all_impropers&2) possibilities to select are mainly aesthetic—symmetry and bond length criteria are applied. See the comments in the blendgen.c for details.

all_impropers&2 = 2 Swapping of central and ending atom is allowed when generating the impropers. Thus, one line C I J K is enough in the table of impropers for both these cases:

```
  I               I
   \               \
    C--K           K--C
   /               /
  J               J
```

This is used for CHARMM21.

all_impropers&2 = 0 Swapping of central and ending atom is not allowed. Thus, the second improper in the above figure will not match C I J K: a separate term for K I J C (or K J C I or K C I J) is required.

column_X Enable/disable column X in the atom table in parset.par.

column_X=0 Any atom can match wildcard atom X in the tables of dihedrals and impropers (default).

column_X=1 Only the atoms marked by 1 in column X in the table of atoms can match X. Rarely used option.

comb_rule The combining rule, See Sect. 3.4.2

distance14 Normal value for this option is 4. If, however, distance14=3 or distance14=5, then everything written in this manual about 1–4 interaction (incl. factor14) applies for 1–3 or 1–5 interactions, respectively.

distance14=0 turns all 1–4 etc. exceptions. This can be set in the generated ble-file if all molecules blended are without 1–4 to supress unnecessary 1–4 calculations and prints. NB: cook checks the presence of 1–4, but does not turn the calculations automatically.

factor14 1–4 interactions are interactions for atoms connected by three bonds (1–2–3–4). All Coulomb interactions for 1–4 atoms are multiplied by factor14. Lennard-Jones interaction for which no explicit 1–4 terms or nbfixes are available are also multiplied by factor14. The default is factor14=0.5.

LJsigma If LJsigma=1 then the RvdW column in the Lennard-Jones table is interpreted as Lennard-Jones sigma and the sigvdW column in NBfix table also as Lennard-Jones sigma. Affects reading the .par file only.

polar polar=1 denotes a polarizable force field; POLAR version of blend must be used. polar=0 is the nonpolarizable default; do not confuse with CHARMM old 'polar' force field, it is NOT polarizable!
polar=2 no longer active (see table "shellrep" as a more flexible replacement)
polar=4 (in addition to 1, i.e., polar=5) (1+4) requests ADIM (induced dipole–induced dipole terms within a molecule)

3.3 Table of atoms

The first table in the parameter file lists all atom types. Example:

```
atoms
!i name  X A    mass   Z  col description
  1 H     0 0   1.00800 1   H  Hydrogen bonding hydrogen (neutral group)
  2 HC    0 0   1.00800 1   H  Hydrogen bonding hydrogen (charged group)
...
 10 CT    1 0  12.01100 6   C  Aliphatic carbon (tetrahedral)
 11 CH1E  1 0  13.01900 7   C  Extended atom carbon with one hydrogen
...
```

Description of columns follows:

i   Number of atom, extended atom or interaction site. The atom numbers must be positive. They need not be ordered but must not duplicate.

name Symbolic name of atom, extended (united) atom or interaction site. It must be composed of max. 4 UPPERCASE letters and digits (1st char must be letter). (Maybe the length limit is 6, I am not sure now :-()

X   1 if the atom will match wildcard X in the tables of dihedral and improper torsions, otherwise 0 (as for hydrogens). Active only if column_X=1.

A   1 if the atom is used in aromatic rings.

mass Atomic mass in g/mol.

col (version 1.7 or higher) This is color type. Valid values are:

> C   Carbon (cyan or gray)
>
> H   Hydrogen (white or light gray)
>
> O   Oxygen (red)
>
> N   Nitrogen (blue)
>
> S   Sulphur (yellow)
>
> M   Metal (magenta)
>
> X   Halogen and other gases (green)

Z   Atomic number. Used for determining the atomic charge centroid serving as the reference point in dipole and quadrupole calculations (cf. hot key $\boxed{\text{E}}$).

description Any text.

3.4   Non-bonded forces

Non-bonded forces consist of the site–site and Coulomb forces. Typical site–site forces are the Lennard-Jones forces, `blend` however supports any forces that are written in a separate module. (`sim/`MODULE`/sitesite[ch]`). The following MODULEs are available:

`lj` Lennard-jones

$$u(r) = -\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - 2 \left(\frac{\sigma}{r} \right)^6 \right] \tag{3.1}$$

where eps = EvdW = ϵ < 0, and sig = 2*RvdW = $2^{1/6}$*LJsigma = van der Waals diameter

`wcalj` WCA-cut-and-shifted Lennard-Jones potential

$$\begin{aligned} u(r) &= -\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - 2 \left(\frac{\sigma}{r} \right)^6 + 1 \right], \quad r < \sigma \\ &= 0, \quad r \geq \sigma \end{aligned}$$

In cook* data, LJcutoff = $-2^{1/6}$ and corr = 0 is needed.

`buck` Exp-6 (with #define REP12, Exp-6-12) potential:

$$\begin{aligned} u_{\text{rep}}(r) &= A_{ij} \exp(-B_{ij}r), \\ u(r) &= u_{\text{rep}}(r) - \frac{C_{ij}}{r^6} + \frac{D_{ij}}{r^{12}} \end{aligned}$$

Several combining rules are available, incl. the Busing aka Kong combining rule (cf. below).

`busing` Exp-6 (with #define REP12, Exp-6-12) potential in the Busing form (Busing combining rule, equivalent to the Kong rule) is written as follows; note that parameter C_i has a different combining rule than usual. R_i is a parameter with rough meaning of molecule's radous and ρ_i is the hardness width.

$$\begin{aligned} u_{\text{rep}}(r) &= f(\rho_i + \rho_j) \exp \left(\frac{R_i + R_j - r}{\rho_i + \rho_j} \right), \\ u(r) &= u_{\text{rep}}(r) - \frac{C_i C_j}{r^6} + \frac{D_i D_j}{r^{12}} \end{aligned}$$

where $f = 0.05e^2/\text{Å}^2$ (in CGS), equivalent to $f = 1.15353878\text{e-}09$ N = 69467.7288 J/mol/Å = 16.6031857 kcal/mol/Å (using CODATA 2019).
Note 1: nbfixes replace $\rho_{ij} = \rho_i + \rho_j$ and $R_{ij} = R_i + R_j$ by another values.
Note 2: Conversion from Buckingham $u_{\text{rep}}(r) = A \exp(-Br)$: $\rho = 1/2B$, $R = \rho \ln(A/2f\rho)$.

`metal` Rosato, Guillope, Legrand (RGL) potential, tight binding potential (2nd order), see `blend/data/metals.par`

Parameters for the site–site potentials are calculated from the atom data in the parameter file using *combining rules*. The values of partial charges needed to calculate the Coulomb forces are not (and cannot be easily) given in the parameter file and must be specified with the molecule.

The following text assumes the Lennard-Jones potential as an example.

3.4.1   Selection of site–site and Coulomb energy terms

The following rules, ordered from high to low priority, apply for selection of the site–site (e.g., Lennard-Jones) interactions between two atoms:

1. All non-bonded forces between directly bonded atoms (1–2) and between second neighbors (1–3) are zero.

2. If the pair is separated by three bonds (1–4 interaction), energy terms are looked for in the following order:

   (a) 1–4 value from table `nbfixes`

   (b) General value from table `nbfixes` with energy multiplied by `factor14`

   (c) Atom values are taken from 1–4 columns of table `Lennard-Jones` and combined according to combining rules. If no 1–4 value is available in table `Lennard-Jones`, it is derived from the general value by multiplying the energy `Emin` by `factor14`.

3. Other interactions than 1–2, 1–3, 1–4 (both intra- and intermolecular): If the pair of atoms is listed in table `nbfixes`, these values for the Lennard-Jones parameters are used, otherwise table `Lennard-Jones` and the combining rules are applied.

The following rules apply for calculating the Coulomb interactions between two atoms:

1. All Coulomb forces between directly bonded atoms (1–2) and between second neighbors (1–3) are zero.

2. Coulomb forces between atoms separated by three bonds (1–4 interaction) are multiplied by `factor14`.

3. In all other cases (intra- and intermolecular), normal Coulomb forces are used.

3.4.2   Combining rules for the Lennard-Jones parameters

The combining rules depend on the site–site module used and you should read the comments of the respective module (files `sim/*/sitesite.[ch]`)

For the most usual Lennard-Jones module (MACSIMUS`/sim/lj/sitesite[ch]`), they are controlled by variable `comb_rule` (old still understood name `sqrt_rule`):

1. `comb_rule & 1`: 0=polarizability rule for Emin (see below), 1=geometric mean for Emin (TIPS style)

2. `comb_rule & 2`: 0=additive diameters (see below), 2=geometric mean for diameters (TIPS style)

The Lorentz–Berthelot combining rule (`comb_rule=1`) and geometric mean square rule (`comb_rule=3`) are typical for modern force fields. Older versions of CHARMM used the 'polarizability rule' (`comb_rule=0`) based on the Kirkwood-Slater formula. The Lennard-Jones interaction of two atoms i and j is then given by:

1. If the values of both polarizabilities `alpha` in table `Lennard-Jones` are nonzero and neither `comb_rule` nor option `-\` is specified:

$$u(r_{ij}) = \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \tag{3.2}$$

where the combining rules for the coefficients are

$$A_{ij} = \frac{B_{ij}}{2}(R_{\mathrm{vdW},i} + R_{\mathrm{vdW},j})^6 \tag{3.3}$$

$$B_{ij} = \frac{\alpha_i \alpha_j}{\gamma_i + \gamma_j} \tag{3.4}$$

$$\gamma_i = \frac{-\alpha_i^2}{256 E_{\mathrm{min},i} R_{\mathrm{vdW},i}^6} \tag{3.5}$$

2. If any of polarizabilities `alpha` is zero then `comb_rule=1` applies, i.e., geometric mean of energy and arithmetic mean of diameters (Lorentz–Berthelot rule).

$$B_{ij} = (-2E_{\mathrm{min},i})^{1/2}(-2E_{\mathrm{min},j})^{1/2}(R_{\mathrm{vdW},i} + R_{\mathrm{vdW},j})^6 \tag{3.6}$$

and A_{ij} is as above.

NOTE: The original article [3] contains the effective number of outer shell electrons N instead of E_{min} and a different formula. The conversion (see also `alphatoe.c`) is:

$$E_{\mathrm{min}} = -\frac{K}{4}\alpha^{3/2}N^{1/2}(2R_{\mathrm{vdW}})^{-6} \tag{3.7}$$

where

$$K = \frac{3}{2}(4\pi\epsilon_0 m_e)^{-1/2}e\hbar = 362.337468\,\mathrm{kcal\,mol^{-1}\mathring{A}}^{3/2} \tag{3.8}$$

3.4.3   Table of site–site parameters

`Lennard-Jones` is the second table in the CHARMM and GROMOS parameter files. For other site–site interactions, `Lennard-Jones` is replaced by the corresponding name, e.g., `Busing`, `Busing-12` – see file `metamake.mmk` in the `blend` directory. `blend` checks the table name not to use a wrong force field. Example:

```
Lennard-Jones
!       alpha     Emin       Rmin       These columns used for
!name   [A^3]   (kcal/mol)    (A)       1-4 interactions
H       0.044   -0.04983    0.800
HC      0.044   -0.04983    0.800
HA      0.100   -0.00447    1.468
CT      0.980   -0.09027    1.800
...
```

Description of columns follows:

name Symbolic name of atom, extended (united) atom or interaction site.

alpha Polarizability α in Å^3 to be used by some combining rules

Emin Minimum energy $E_{\min}$ of the Lennard-Jones (LJ) atom-atom potential in kcal/mol

RvdW Van der Waals radius R_{vdW} (half of the atom-atom distance of the LJ potential minimum), in Å. Note that if `LJsigma` is set than this column contains the Lennard-Jones sigma instead!

1-4 interactions If another three columns are given, they contain `alpha Emin Rmin` for 1–4 interactions

For other site–site interactions, the `Emin` and `RvdW` columns may have different meaning and more columns may be added; all columns but `atom` are repeated for 1–4 interactions.

The `alpha` column is used for combining rules and does not mean the polarizability in the POLAR version. Often it is irrelevant.

3.4.4 Non-bonded fixes

Site–site forces between certain types of atoms cannot be accurately calculated using the combining rules and must be listed separately in table `nbfixes` (old name `NBFIX`). If the 1–4 columns are missing, `factor14` multiple of the normal value is taken for the 1–4 interactions. WARNING: not true, also 1–4 must be present (to be fixed). If `sigvdW=0` for the normal interaction (not 1–4), the interaction is calculated from site–site (e.g, Lennard-Jones) tables using the combining rule; thus, only 1–4 terms can be given separately in the `nbfixes` table while other terms are regular. Example (from `charmm22.par`):

```
nbfixes
! names    Emin       sigvdW
! i    j [kcal/mol] [A]
! OT   OT    -0.152073  3.5365 not needed
HT   HT    -0.04598   0.4490 -0.04598   0.4490
HT   OT    -0.08363   1.9927 -0.08363   1.9927
```

where `sigvdW` is twice the van der Waals radius. Note that if `LJsigma` is set than this column contains the Lennard-Jones sigma instead!

WARNING (BUG/FEATURE): Emin (or other energy-like parameter, as A for Buckingham) is in the same units as in the Lennard-Jones (or Buckingham) table, typically kcal/mol. If there are additional parameters (as C and D for the Buckingham force field), they are in kcal/mol in the Buckingham table. For cook* version V2.7v and older, they are in the internal units (K) in the nbfixes table. It holds 1 kcal/mol = 503.219565 K (CODATA 2010). In version V2.8a and newer, uniform units are used.

3.4.5 Table of polar atom parameters

This applies for the POLAR version only.

```
polaratoms
!atom  alpha shell  Esat       arep
!       A^3    e    kcal/mol  1/e
```

```
CL      2   -1000  0          0.22439024
I       4.5 -1000  0          0.37037037
```

atom Atom name

alpha Dipolar polarizability[1] α in Å^3

shell Number of outer shell electrons in e (normally negative because the electron shell is negative). This value is copied to the output atoms table and used by the polar version of cook as the charge of auxiliary site mimicking polarizability by a dipole.

Esat Saturation energy of the induced dipole. `Esat=0` turns off the saturation (the same as infinity saturation energy)

arep Parameter for the shell-core model (repulsive antipolarization), in e^{-1}. Also denoted κ.

3.4.6 Table "shellrep" of repulsive counterparts

This applies for the POLAR&1 version only. The "repulsive antipolarization" or "shell-core" term, equation (25.14), is not symmetric and and consists of a polarizable ion (usually anion), i, and its "repulsive" counterpart (usually cation with none or small polarizability), j. MACSIMUS uses this term if:

1. the atoms are more than `distance14` bond apart (normally `distance14=4` and 1–4 interactions cannot have the shell-core term),

2. the anion i has nonzero parameter `arep` alias κ_i,

3. the cation j is listed in table "shellrep" (one atom type per line).

Example:

```
shellrep
!atom
NA
LI
```

(In principle also anions (with $\kappa > 0$) may appear in this table. In this case there may be two shell-core terms per a pair of atoms. This is not recommended.)

3.4.7 Table of axially polar bonds

This table describes the axial polarizability tensor of a pair of atoms connected by a bond. The tensor is located at the first atom of the pair. `alphazz` is the polarizability in the direction of the axis, `alpha = alphaxx =alphayy`.

```
polarbonds
! atom --> atom  alpha  alphazz Esat
!                A^3    A^3    kcal/mol
   H        CL    1      2      300
```

---

[1]More precisely, the polarizability volume

3.4.8 Table of 1–3 axially polar groups

The same as polarbonds but the second atom is a second connected neighbor (1–3). The middle atom does not enter the formulas for induced dipoles and is used only in topological analysis to assign the force field terms.

```
polarangles
! atoms     alpha  alphazz Esat
  S  C  N   1.778  5.670     0
  N  C  S   0.867  0.801     0
```

3.4.9 Water models

Supported water models are collected in the parameter file `water.par`. Other force fields may contain water, too.

Table `waters` defines water model(s). The reason is that water molecule is recognized from molecule topology and appropriate information is exported so that `cook` is able to use an optimized code for water which speeds up the calculations. However, this applies only to version without LINKCELL; i.e., makes sense for small systems only. Example:

```
waters
! must be in standard order recognized by cook
! name NS atom charge atom charge ...
TIP3P  3   HT .417     HT .417   OT -.834
```

`name` is the model name, NS number of sites. Atom types and partial charges follow.

More about waters in `blend`:

1. Command `aw` adds water sites in order H H O which is 'standard' and recognized by `cook` to be rigid TIP3P water model (it only is recognized if `blend -o SYSNAME` is called with `-n-1` option!)

2. To treat old files with water added by `aw` with order of sites O H H, option `-[3` should be used (in addition to `-n-1`). (Precisely, `-n-1 -[#` causes #-atom clusters to be reordered so that H go first.)

3. Then, in the result of `blend -n-1`, water line 'species xxx.1' should be edited to 'species TIP3P' unless the appropriate table `water` is present.

4. If the models are rigid option `-h` should be given to `blend` to prepare the force field ble-file, e.g.: `blend -o spce -h SPCE.che` Supported water models are collected in `water.par`. Example:[4]

   ```
   water SPC/E water model
   parameter_set = sea
   Hp0.4238 Hp0.4238
        \    /
        On.8476
   ```

5. In CHARMM21, the TIP3p water oxygen is called OW, in CHARMM19 and CHARMM22 it is OT[5].

6. CAVEAT: Crystal water in PDB files is not (now) recognized. If `blend` has been called with option `-n-1`, it is possible to edit the corresponding `species` command in the ble-file.

7. Che-file for the classical but now obsolete model by Stillinger et al., `ST2.che`, is:

```
ST2 water
parameter_set=water

! use blend -o st2 -h-1 ST2
! not real masses - not suitable for diffusivity etc.
! (real masses can be implemented by "lone dependants")

        OH2--LPn.2357
       / | \
H.2357  |  LPn.2357
       H.2357
```

Only `cook` versions without LINKCELL support this model; however, virial pressure reported is not correct – pressure by virtual volume change should be used. ERFCPLUS (see ewald.c) must be at least 1.8 for ST2.

(To be moved) In versions of `cook` newer than Sep 96 there are special optimized functions available for water–water interactions for standard rigid models. If option `-x` is given to `cook`, using of these functions is turned off and general slower functions are used (ST2 is not available with `-x`). The CHARMM version of TIP3P and any modifications with flexible angles and/or bonds are available with `-x` only.

3.4.10 Table defining the protein backbone types

This table contains three lines with atom types used respectively for N, C_α, and carbonyl C. Example:

```
backbone
! types of atoms for the backbone
! peptide N types:
NH1 NH2 NH3 N
! Calpha types:
CT1 CT2 CP1
! carbonyl (>C=O) C types:
C CC CD
```

This information is used for:

- Writing a PDB file 'from scratch' (option `-w10`), e.g., when the source of molecule was a che-file.

- Creating α helix conformation (option `-r6`).

3.5 Non-bonded potential cutoff

If option `-tnumber` is specified, the non-bonded interactions (both Lennard-Jones and Coulomb) within `blend` are calculated with cutoff. This speeds up the calculations.
NOTE: For cutoffs in `cook`, see see Sect. 11.4 and see Sect. 24 or see Sect. 15.3.

The cutoff is smooth between site–site distances C_1 and C_2 given by option `-t` (see Sect. 2.2.3). The switch function which multiplies the potential $u(r)$ is:

$$s(r) = \begin{cases} 1, & \text{for } r < C_1 \\ A[((r^2 - C)^2 - B)^2 - D]^2, & \text{for } C_1 < r < C_2 \\ 0 & \text{for } r > C_2 \end{cases} \tag{3.9}$$

where $C = (C_1^2 + C_2^2)/2$, $B = (C_2^2 - C_1^2)/2$ and constants A, D are given by the requirements that $s(C_1) = 1$, $s(C_2) = 0$. Then, $s'(C_1) = s'(C_2) = s''(C_1) = s''(C_2) = 0$ and the cut-off potential and the corresponding force are sufficiently smooth.

Note that this cutoff is used only within `blend` to increase performance for large molecules and is not exported to `cook`, which uses a different cutoff algorithm.

Recommended values:

- `-t0` (Default) No cutoff (infinite range), recommended for molecules smaller than 500–1000 sites.

- `-t11` Minimum reasonable value for larger molecules. Typical cutoff is then 3 LJ sigma, minimum (for larger atoms as triply bonded carbon) is 2.5 LJ sigma.

- `-t6` This is sufficient for filling missing hydrogens (in this case, all other atoms are kept fixed and the most important forces are bond angles).

3.6 Bond potential

The bond potential is:

$$U(r) = K(r - r_0)^2 \tag{3.10}$$

where r is the bond length, r_0 the equilibrium bond length, and K the force constant.

This potential is used for minimization in this program. In simulations by `cook`, bond-length constraints may be used instead.

The format of the table of bonds is given by the example:

```
bonds
!atom atom  K  length
C    C     450  1.38
C    CH1E,CH2E,CH3E  405  1.52
```

Description of columns follows:

`atom` Atom type. Comma separated lists of atom types are allowed. Both atoms may be lists, however, repeating terms are not allowed (e.g., `C1,C2 C1,C2 100 1.5` is incorrect because it expands into `C1 C2 100 1.5` and `C2 C1 100 1.5` which are equivalent.)

K Force constant K in kcal mol^{-1}Å^{-2}.

length Equilibrium bond length r_0 in Å.

An error condition is raised if blend cannot find a bond in the table.

3.7 Bond angle potential

The bond angle potential is

$$U(\theta) = K(\theta - \theta_0)^2 + K_{\text{Urey-Bradley}}(|r_1 - r_3| - s)^2 \qquad (3.11)$$

The algorithm to calculate the bond angle θ uses the scalar product of both bonds to calculate $\cos\theta$. There is a substantial (mechanical) singularity for θ close to 0 or π; for the equilibrium bond angle $\theta_0 = 0$ or $\theta_0 = \pi$ it becomes a numerical singularity only and is fixed. In the Urey-Bradley term, s is the 1–3 equilibrium constant

The table of bond angles in the parameter file looks like:

```
angles
!atom atom  atom        K     angle  [Kub  s]
C     C     C           70    106.5
C     C     CH2E,CH3E    65    126.5
C     C     CR1E         70    122.5

S     C     N            40    180    100   2.879
```

Description of columns follows:

atom Symbolic atom name.

K Force constant K in kcal/mol.

angle Equilibrium angle θ_0 in degrees.

Kub Force constant K in kcal/mol Å^2. If not present, Kub=0.

angle Equilibrium 1–3 distance in Å.

An error condition is raised if blend cannot find an angle in the table.

For planar groups (like Fe in the heme):

```
      B
      |
   A--X--C
      |
      D
```

where bond angles are 90°, it is necessary to suppress generating angle terms for angles A-X-C and B-X-D. This is not a problem if (approximate) coordinates are known: an angle larger than 150° is removed from the list of angle terms and a message is printed. If the coordinates are not known, a warning message is printed; it may help to run blend again.

3.8   Torsions

Torsions are force field terms defined by four atoms. Different types of torsions (dihedrals, impropers, and also *aromatics*) differ by bonding of the atoms and by energy terms used but not by the definition of the torsion angle.

3.8.1   Torsion angle

The *torsion angle* of four atoms I J K L (written in this order but not necessarily connected by bonds in this order) is the oriented angle of plane IJK from plane JKL. The orientation is defined as follows: let us watch the four atoms from the KJ direction so that atom K is closer to the eye than atom J. Then the angle is the oriented angle of line segment LK from IK:

```
        L                                        L
         .                       I                \   angle   I
          .                     -                  \ <---- /
           .                   -                    \     /
            .                 -                       \   /
             .               -                         \ /
  wiew ---->       K ------- J                          K
```

If the four atoms are connected by bonds as in I–J–K–L , the torsion angle is called *dihedral (torsion) angle* or *proper torsion angle*; it is zero for the *cis* conformation. If atom I is bonded to J, K, and L, the torsion angle is called *improper (torsion) angle*, atom I is thus the central atom.

3.8.2   Torsion potential

WARNING: The following text is probably imprecise. Note also that in the ble-file, ϕ_0 is not given. To be double checked: K is changed to $-K$ for $\phi = 180°$ for odd n in blend. Consequence: The torsion potential may be shifted by $2K$ with respect to the habit in other force fields.

The *torsion potential* for four atoms I J K L with the torsion angle ϕ, equilibrium torsion angle ϕ_0, force constant K and *periodicity* n is:

$$U(\phi) = \begin{cases} K(\phi - \phi_0)^2 & \text{for } n = 0 \\ K[1 + \cos(n\phi - \phi_0)] & \text{for } n > 0 \end{cases} \tag{3.12}$$

The torsion potential with $n > 0$ is sometimes called *dihedral (torsion) potential*. It is used to describe hindered rotation as e.g. of the methyl groups in ethane.

EXCEPTION: $n = 5$ denotes a special cis- and trans-dihedrals (see Sect. 3.8.4).

CAVEATS: Only $\phi_0 = 0°$ and $180°$ are supported. The case $\phi_0 = 180°$ is replaced by the following formula

$$U(\phi) = |-K| + (-K)\cos(n\phi) \tag{3.13}$$

which is equivalent as soon as $K > 0$. If, however, $K < 0$, which is the case of some CHARMM22 terms, this formula for $U(\phi)$ gives shifted values of energy.

The torsion potential with $n = 0$ is sometimes called *improper (torsion) potential*. It is used for three purposes:

1. With non-zero improper torsion angle it is used to define chirality of sp3 chiral atoms in the extended atom representation (e.g., CH1E in aminoacids).

2. With zero improper torsion angle it is used to keep the sp2 hybridized bonds planar (e.g., carbon in ketones).

3. In some force fields the improper torsion with zero dihedral angle is used misleadingly to keep 5- and 6- aromatic rings planar. The angle is in fact the dihedral angle of four atoms around the ring; for instance in benzene there are six such *aromatic dihedrals*. The problem is that the energy terms occur in the table of impropers and special hooks must be used to find them. See Sect. 3.8.7.

Note that the output from `blend` uses slightly different formulas for the torsion potentials (see Sect. 5.8).

3.8.3 Conversion of dihedrals

WARNING: Often $\psi = \pi - \phi$ is used as the dihedral angle (i.e., *trans* is zero).

The OPLS style formula for dihedrals is[6]

$$U(\phi) = V_0 + \frac{1}{2}[V_1(1 + \cos(\psi)) + V_2(1 - \cos(2\psi)) + V_3(1 + \cos(3\psi))]. \qquad (3.14)$$

In case of $\phi_0 = 0$, K_2 has **opposite sign**. Note also factor $1/2$. Several such terms must be written separately. For example:

```
!atom atom atom atom    K[kcal/mol] n angle
  CT   CT   CT   CT    0.87          1 0
  CT   CT   CT   CT    0.0785        2 0
  CT   CT   CT   CT    0.1395        3 0
```

is equivalent to $V_1 = 1.74$, $V_2 = -0.157$, $V_3 = 0.279$ (in kcal/mol). The MACSIMUS coefficients are positive indicating positive energy of the *cis* conformation.

The Ryckaert–Bellemans formula for dihedrals (good for alkanes) is

$$U(\phi) = \sum_n C_n \cos^n \psi, \quad \psi = \pi - \phi. \qquad (3.15)$$

The conversion is

$$
\begin{aligned}
C_0 &= V_0 + V_2 + \frac{1}{2}(V_1 + V_3) \\
C_1 &= \frac{1}{2}(3V_3 - V_1) \\
C_2 &= -V_2 \\
C_3 &= -2V_3
\end{aligned}
$$

Conversion of Ryckaert–Bellemans to MACSIMUS is (with $\phi_0 = 0$, upto $n = 6$, but note that K_5 means something else so that always use $K_5 = 0$):

$$
\begin{aligned}
K_0 &= C_0 + C_1 + C_3 - \frac{1}{4}C_4 + C_5 - \frac{3}{8}C_6 \\
K_1 &= -C_1 - \frac{3}{4}C_3 - \frac{5}{8}C_5 \\
K_2 &= \frac{1}{2}C_2 + \frac{1}{2}C_4 + \frac{15}{32}C_6 \\
K_3 &= -\frac{1}{4}C_3 - \frac{5}{16}C_5 \\
K_4 &= \frac{1}{8}C_4 + \frac{3}{16}C_6 \\
K_5 &= -\frac{1}{16}C_5 \\
K_6 &= \frac{1}{32}C_6
\end{aligned}
$$

and from MACSIMUS to Ryckaert–Bellemans:

$$
\begin{aligned}
C_0 &= K_0 + K_1 + K_3 + 2K_4 + K_5 \\
C_1 &= -K_1 + 3K_3 - 5K_5 \\
C_2 &= 2K_2 - 8K_4 + 18K_6 \\
C_3 &= -4K_3 + 20K_5 \\
C_4 &= 8K_4 - 48K_6 \\
C_5 &= -16K_5 \\
C_6 &= 32K_6
\end{aligned}
$$

In addition, constant positions of zero K_0 and C_0 are ignored (may be nonzero).

Scripts to convert Ryckaert–Bellemans to MACSIMUS and back (upto $n = 3$ only):

```
tabproc  "-A-C*.75" B/2 -C/4  # C1 C2 C3 -> K1 K2 K3
tabproc "3*C-A" "2*B" "-4*C"  # K1 K2 K3 -> C1 C2 C3
```

3.8.4   Cis and trans-dihedrals

Sometimes it is necessary to guarantee the cis and trans conformation around a bond. For instance, CHARMM22 distinguishes both amide hydrogens in amides:

```
    H   O    H              HA^p.09   Ovn.55      HCISvp.32
    |   ||  /                 |         |           /
  H--C--C--N        HA^p.09-CT2n.27--CCp.55--NH2n.62
    |      \                  |            \
    H       H              HA^p.09          HTRA^p.30
```

This is a problem of the initial configuration only—filling unknown hydrogens (heavy atoms are known). To do it correctly, both hydrogen types are distinguished and special cis dihedral terms are added:

$$U_{\text{cis}}(\phi) = K \cos^2 \phi \qquad \text{for } K \cos \phi < 0 \tag{3.16}$$

$$U_{\text{cis}}(\phi) = 0K \qquad \text{for } K \cos \phi > 0 \qquad\qquad (3.17)$$

$K < 0$ (or $K > 0$ and $\phi_0 = 180$) denotes the trans dihedral.

The cis-dihedral energy term is 0 is the atoms are in the correct cis position and large if cis is encountered.

These dihedrals are not exported (in a ble-file) to `cook` because they are not needed once the bond is in the correct conformation.

3.8.5   Implementation of the torsion potential

There are two different algorithms to implement the torsion potential. For $\phi_0 = 0$ or $\phi_0 = \pi$ the potential is even in ϕ and it is sufficient to calculate $\cos \phi$ (which requires evaluation of certain scalar product); both $(\phi - \phi_0)^2$ and $\cos(n\phi - \phi_0)$ (the former is implemented for the most usual case of $\phi_0 = 0$ only) is then expressed via $\cos \phi$.

The improper torsion used for keeping chirality has $n = 0$ and arbitrary ϕ_0 and in this case the sign of ϕ is important; another (and more complicated) algorithm based on $\sin \phi$ has to be used in this case.

3.8.6   Chirality

The chirality of an extended atom I (usually of type CH1E) with three bonds to atoms J, K, L (listed in this order) is defined as follows: if the triple of vectors (IJ,IK,IL) has the same arrangement as the standard right-hand orthogonal coordinate system *x-y-z*, the chirality is +1. In other words, let us watch the molecule from that side of plane JKL so that we see J–K–L arranged anti-clock-wise (in the positive direction). If then atom I lies below plane JKL, the chirality is +1.

The sign of the equilibrium improper torsion angle of four atoms I J K L is the same as chirality of atom I (see Sect. 3.8.1).

The chirality of a tetrahedral atom I (usually of type CT) with four bonds to atoms J, K, L, M (listed with this order in column `bound_atoms`) is defined in the same way. It is naturally assumed that I and M are not separated by plane JKL. Not implemented yet.

Normally, `blend` finds the requested chiralities in `.mol` file or calculates them from z-coordinates in `.che` file. If option `-c` is given, `blend` calculates chiralities from the 3D configuration. File species`.mol` is then rewritten (the old one is named species`.mox`).

Chirality calculations and checking are performed in two stages: first, 3D configurations around all three-bonded atoms are considered; second, tables of impropers are searched to find whether the atom is really chiral. It may happen in the first stage that an atom is considered chiral just because the planar configuration was distorted and this is fixed in the second stage.

Since the chirality depends on atom numbering, it is not suitable to human hackers. Thus, we define *alphabetical chirality* as the chirality with respect to alphabetical order of the three neighbors of the chiral atom, and *mass chirality* as the chirality with respect to ordering by atomic masses. All three chiralities are printed to output. The alphabetical chirality of chiral alpha CH1E atom in the main chain of naturally occurring aminoacids (but glycine) is +1 if the atoms are named as in CHARMM and united atom representation is adopted for aliphatic hydrogens. Our simple algorithm does not solve cases when some of the three neighbors of the

chiral atom are the same (the atom may still be chiral); in these cases, `?` is printed.

3.8.7   Dihedrals in aromatic rings

This section applies only to force fields using impropers (=terms listed in the table of impropers) also for aromatic dihedrals. Normally `ar_dih_limit=-1` and then all energy terms listed in table `dihedrals` apply to atom chains I-J-K-L (including case `n=0` for aromatic dihedrals) while all energy terms listed in table `impropers` are for atoms J,K,L bonded to I.

Let us define *aromaticity* of the dihedral angle I–J–K–L as the number of atoms that lie in the same aromatic ring, provided that bond J–K lies in the same aromatic ring too. For instance, benzene (full-atom model) has 6 dihedrals with aromaticity=4, 12 dihedrals with aromaticity=3, and 6 dihedrals with aromaticity=2.

Specifying `ar_dih_limit>0` has two effects:

1. Aromatic dihedrals are looked for in the table of impropers.

2. Variable `ar_dih_limit` controls the way in which dihedrals are assigned to bonds that are parts of aromatic rings. `ar_dih_limit` is the maximum aromaticity allowed for dihedrals to be included. `ar_dih_limit` must be in the range 1–4.
   `ar_dih_limit=1` means no dihedrals with central bond in aromatic ring; normally, the aromatic dihedrals should be enough to keep aromatic rings planar.
   `ar_dih_limit=4` means that aromaticity is irrelevant when counting dihedrals—all are included.
   `ar_dih_limit=3` means that if all atoms I, J, K, L lie in the same aromatic ring, they are not included in dihedrals (they have already been included in aromatic dihedrals). If, however, at least one of atoms I and L points out of the aromatic ring, the dihedral is included.

A 5-ring is called aromatic if it contains at least 3 aromatic atoms. A 6-ring is called aromatic if it contains at least 4 aromatic atoms. This cumbersome definition is necessary in older versions of CHARMM because non-aromatic (general) atom types are used sometimes in aromatic rings. In new versions of CHARMM, all atoms in aromatic rings are of aromatic type so that a simpler criterion could be used.

3.8.8   Tables of dihedrals and impropers

Example (from `charmm.par`):

```
 dihedrals
 !atom atom  atom  atom    K   n angle
 CH1E  C     N     CH1E   10   2  180
 CH2E  C     N     CH1E   10   2  180

... lines deleted ...

  X     S     S     X      4   2   0
```

```
impropers
! Improper torsions. See dihedrals above.
!atom atom  atom  atom    K    n angle
C     CR1E  C     CH2E    90   0    0
C     CR1E  CR1E  CH2E    90   0    0
```

... lines deleted ...

```
NR    X     X     CT      25   0    0

cisdihedrals
O    CC   NH2  HCIS     1000      5   0.00   !  cis
O    CC   NH2  HTRA     1000      5   180.00 !  trans
```

Description of columns follows:

atom Symbolic atom name

K   Force constant K in kcal/mol.

n   Periodicity number. Note that $n = 5$ is exceptional—see above.

angle Equilibrium angle ϕ_0 in degrees.

3.8.9   Atom matching rules for finding the energy terms

If `column_X=1` has been specified, wildcard atom `X` in the tables of dihedrals and impropers matches any atom but hydrogen (to be more precise, it matches all atoms with 1 in column `X` in the atom table); if `column_X=0` has been specified, `X` can match any atom.

If there are several matches possible, the match with less wildcards is selected; if this selection is not unique, an error message is printed.

If the dihedral or improper torsion is not found in the table, it is assumed that it is zero; if `-v` has been specified, a message is printed.

Example: In the improper torsion table [7], there are the following two entries:

```
!atom atom  atom  atom    K    n angle
C     OC    OC    CH2E  100.0 0  0.0
C     X     X     CH2E   60.0 0  0.0
```

Then, parameters for `C-OC-OC-CH2E` will be taken from the first line, while the parameters for `C-C-OC-CH2E` from the second line.

If there are several periodicities in the table of dihedrals, they are summed; note that the entries may have different numbers of wildcards. It is not allowed to combine the improper potential ($n = 0$) with the dihedral potential ($n > 0$).

Example:

```
!atom atom  atom  atom    K    n angle
CUA1  CUA1  CUA3  CUA3   0.43 1   0.0
X     CUA1  CUA3  X      0.76 2 180.0
```

In this case, both terms are summed for `CUA1-CUA1-CUA3-CUA3`.

Chapter 4

Description of molecules

The structure of molecules is described by listing all atoms and bonds. In addition, information to determine conformation of chiral atoms and partial charges are given.

`blend` supports two formats: The molecular format is intended for interaction with other programs. The chemical format allows simple molecules to be easily build up using a text editor only.

4.1 Molecular file (mol-file) format

This file with extension `.mol` describes the topology of a molecule and contains also partial charges. Program `blend` checks consistency of bonds, removes possible multiple bonds, and calculates the number of clusters. Thus, you can treat e.g. fulerene with an atom inside the cage or a water dimer as one "molecule" consisting of more clusters.

```
alanine - aliphatic hydrogens in united atom representation

! total charge = 0.00

parameter_set = charmm21
zero_energy = -1
number_of_atoms = 9

atoms
! i Crad/atom-id   a-type  charge   chir nb bound_atoms
  0 W095/H1      HC    0.3500   0    1    2
  1 W095/H2      HC    0.3500   0    1    2
  2 B130/N       NT   -0.3000   0    4    1 3 0 4
  3 W095/H3      HC    0.3500   0    1    2
  4 C139/CA      CH1E  0.2500  -1    3    5 2 6
  5 C139/CB      CH3E  0.0000   0    1    4
  6 C125/C       C     0.1400   0    3    7 4 8
  7 R126/O1      OC   -0.5700   0    1    6
  8 R126/O2      OC   -0.5700   0    1    6
```

The first non-comment (i.e., not beginning with `!`) line is the header.

Optional `parameter_set` defines the name of the parameter file without extension. Extension
`.bin` is added, and if the binary file does not exist, extension `.par` is tried. The default is
`charmm21`.

Optional `zero_energy` is the energy (in kcal/mol), or energy offset, which is to be added to the
intramolecular energy. It is typically a result of 'chemical reactions' – see group replacement
as defined in `.rea`. Missing `zero_energy` means zero.

`number_of_atoms` refers to the number of atoms in the following table.

`atoms` introduces the table of atoms. The table column are:

- **i** Order number.

- **Crad/atom-id** Crad: optional color (one of `RWCBYMGO` for Red, White, Cyan(gray), Blue,
  Yellow, Magenta, Green, Orange) and radius in 0.01 Å. Must be 4 characters followed by
  a slash. If not present, either a gol-file is used, or a guess is made from atom type.

  atom-id: Names that the user can give to the atoms. They have no special meaning for
  program `blend` but that they can label atoms in a picture of molecule.

- **a-type** Atom (extended-atom, interaction site) types.

- **charge** (Partial) charge of atom in e.

- **chir** Atom chirality. See Sect. 3.8.6.

- **nbonds** Number of chemical bonds.

- **bound_atoms** Indices of atoms bound by a chemical bond.

If there is an asterisk `*` in the 1st column of the `atoms` table, the corresponding atom is *marked*
and can be kept fixed during minimization. See option `-k` for details. Example:

```
! i a-id  a-type charge chir nbonds bound_atoms
* 4 CA     CH1E  0.2500  -1    3     5 2 6
* 5 CB     CH3E  0.0000   0    1     4
* 6 C      C     0.1400   0    3     7 4 8
```

4.2   Chemical file format

The chemical file format is again best seen from an example. The extension of the file must be
`.che`.

```
  dipeptide
  parameter_set=charmm21
        HC^.35
          |
  HCv.35-NTn.3:nter-HCv.35
          |
        CH1Ev.25---CH3E                                ! ALA
          |
        C.55-Ovn.55     H.3
          |               |
          NPn.35--------+-Hv.25
          |               |
          |             N5Rn.4==C5RE.3
          |             /         |
        CH1Ev.1-CH2E-C5R.1        |                    ! HIS
          |           \           |
        C.14:cter       C5RE.1==N5Rn.4
        /    \
    OC^n.57  OCvn.57
```

The first two lines have the same meaning as for the mol-file. They must not be separated by any blank line.

The symbolic atom names must be in uppercase and may contain digits. Numbers mean partial charges in electrons, negative charge is marked by `n` or `m` (because `-` denotes a bond), positive optionally (to distinguish the charge from a digit as a part of identifier) by `p` (e.g., `MNAp1`). The circumflex `^` means that the atom goes 1 Å up the plane, `v` that it goes down; you can use several `v` or `^`. The field after a colon means optional identifier (column `a-id` in the mol-file); if not present, the identifier is derived from atom number and type (e.g., `0-HC`).

The bonds cannot bend, they may, however, cross; character `+` should be used for the crossing (`x` is no longer allowed since V 2.1h: use `+` for crossing of bonds of any direction). Additional bonds can be added by statement #connect. Example:

```
  ...
    ---S.2:cys
  #connect 33-S 44-S  125-S cys
```

There may be any number of #connect statements, but the total number of id's must be even – the id's are interpreted in pairs. No wildcards are allowed.

Unless changed by option `-r`, the initial configuration is derived from the layout of atoms on the paper and the z-coordinates given by `^` and `v` used; this is not a good initial configuration, it is, however, sufficient to determine chirality for each chiral atom. (There are two chiral `CH1E`'s in the above example; try to check that the absolute chiralities of alpha-carbons in the above dipeptide are the same as in alanine given is above – see Sect. 4.1). Energy minimization must follow. Normally one line separation is 1 Å and the character pitch is 0.4 Å. This can be rescaled by option `-e`number (in %). The z-coordinates are not rescaled.

Statement `#include` file can be used in che-files: `file` should contain (a part of) molecule without header, just as included physically to the place of #include statement. The **BUG:** #include statements cannot be nested.

An asterisk `*` in front of atom type is exported to a mol-file and means that the atom will be kept in place while minimization. Examples:

```
blendbus -g5 -r2 -e200 AlCl3SCN.che
parameter_set=busing
        *ALp3


 CLn1:Cl1            CLn1:Cl2
        CLn1:Cl3


 LIp1


 #include "SCN.inc"
```

To create files in the chemical format, it is recommended to use a text editor that supports rectangular blocks. As complicated molecules as adamantane $C_{10}H_{16}$ or fulerene C_{60} with an atom in the cavity have been easily prepared using e.g. `QEDIT`.

If the molecule is given by this chemical format, the corresponding species`.mol` file is generated (the old one is backed up as species`.mox`).

Chapter 5

Output format (ble-file)

The output file contains full description of the mixture of molecules `blend`-ed together. It serves as input for MD programs `cook*`. In addition, it contains a lot of comments (marked by !) for humans. Alternatively, it may be created by hand to avoid `blend` stage; this is necessary for models which are not implemented in `blend`. The structure of a ble-file is:

```
<global data>
<table on site-types>
<table 'nbfixes' of non-bonded fixes>

<header of molecule 1>
<data of molecule 1>
<table 'sites' of molecule 1>
<table 'bonds' of molecule 1>
<table 'angles' of molecule 1>
<table 'dihedrals' of molecule 1>
<table 'impropers' of molecule 1>
<table 'aromatics' of molecule 1>

<header of molecule 2>
<data of molecule 2>
etc.
```

A user is encouraged to edit this file, like adding partial charges, deleting unnecessary potential terms, etc.

5.1   Global parameters

Global data are in the "get data" format (see Sect. 2.4.1). Example:

```
nspec=2  nsites=8  nnbfixes=3  factor14=0.500000 ;
```

where

distance14 Setting `distance14=0` means that any 1–4 terms are missing in the site–site and nbfixes tables of the ble-file. Note that the value of distance14 is not used by cook at all (except for distance14=0) – the topological analysis is performed by blend and coded in particular species as starred neighbor number; with distance14=0, no such term is allowed (cook* will report an error).

eunit Energy unit used in the ble-file, expressed in Kelvins (k_B K). The output of `blend` is in kcal/mol, which means that `eunit=kcal/Eunit=503.2195334987658`; this is the default. If you construct a ble-file by yourself, you may use (based on 05/2019 SI definitions, see `sim/units.h`)

>   eunit=1 For energies/k_B in K
>
>   eunit=1000/8.31446261815324 For energy in kJ/mol
>
>   eunit=1.380649e-23 For energy in J

factor14 polar comb_rule See Sect. 3.2.

nnbfixes Number of non-bonded fixes.

nparms Number of additional parameters (over two basic denoting atom size and energy) for the site–site interactions. For Lennard-Jones `nparms=0`. It must match `SS_PARMS` in header file `macsimus/sim/*/sitesite.h`, where * denotes the respoective potential version.

nsites Total number of types of interaction sites contained in all molecules.

nspec Total number of molecules blended together. (The name 'nspec' originates from 'Number of (molecular) SPECies'. Though here 'number of molecules' just means 'number of species', in molecular dynamics there are many *molecules* of the same *species*.)

polar Polarizability version, see also file `blend/metamake.mmk`. Flags can be summed in the polarizable versions.

>   polar=0 Standard non-polarizable version
>
>   polar&1 Basic dipolar polarizability support (incl. axial and saturated)
>
>   polar&2 : No longer used, the functionality can be mimicked by table `shellrep`
>
>   polar&4 : Also intramolecular 1-2 and 1-3 induced dipole–induced dipole terms are included. Note that then the molecular polarizability is not a sum of atomic polarizabilities!

The following **additional variables** are not present in the `blend` output, but are read and understood by `cook`:

a,b,c,x,y,z Auxiliary real (double) variables.

i,j,k,n Auxiliary integer variables.

5.2 Site types

This table lists all site types in all molecules processed. It's name depends on the site–site module linked; `Lennard-Jones` is most usual (subdirectory `macsimus/sim/lj`), other options are `Busing` (`macsimus/sim/bus`), are `Buckingham` (`macsimus/sim/buck`), etc. (The column headers are similar as in the parameter file. The number of columns depends on `nparms` (which should match `SS_PARMS` in `macsimus/sim/*/sitesite.h`). Example for a typical protein force field is (`nparms=0`, `polar=0`):

```
 Lennard-Jones
 !i atom    mass   alpha    Emin    RvdW  alpha[1-4] Emin[1-4] RvdW[1-4]
  2 HC     1.0080  0.044  -0.0498  0.6000    0.0440   -0.02490   0.6000
  4 HT     1.0080  0.044  -0.0498  0.9200    0.0440   -0.02490   0.9200
 11 CH1E  13.0190  1.350  -0.0486  2.3650    1.3500   -0.10000   1.8100
 13 CH3E  15.0350  2.170  -0.1811  2.1650    2.1700   -0.10000   1.7600
 14 C     12.0110  1.650  -0.1410  1.8700    1.6500   -0.07050   1.8700
 36 NT    14.0067  1.100  -0.0900  1.8300    1.1000   -0.10000   1.6300
 43 OC    15.9994  2.140  -0.1591  1.5600    2.1400   -0.07955   1.5600
 46 OW    15.9994  0.840  -0.1521  1.7680    0.8400   -0.07605   1.7680
```

The units are kcal/mol for `Emin` (if not changed by variable `eunit`), Å for RvdW, and Å^3 for `alpha`.

Site–site force fields with `nparms>0` have the columns `atom mass alpha Emin RvdW parm...parm alpha[1-4] Emin[1-4] RvdW[1-4] parm[1-4]...parm[14]`. GAUSSIANCHARGES version has the Gaussian width σ as the last additionalparameter `parm[nparms-1]`.

Since V2.8a, cook* accepts missing 1-4 terms if they are identical to the normal non-bonded ones. If more parameters follow (for POLAR), one = can replace all `nparms+2` 1-4 terms.

For POLAR force fields (`polar>0`), four columns are appended:

alphapol Polarizability in Å^3 to be used by polar version of `cook`. Do not confuse with `alpha` which serves for the Kirkwood-Slater combining rule; nevertheless, if this column is missing, `alpha` is used.

shell Auxiliary charge ('number of outer shell electrons') for replacing a point dipole by a finite-size dipole (in `cook`). See variable `shell` in the parameter file and option `-@`). If this column is missing, -1000 is used.

Esat Saturation energy, in the units defined by `eunit`, kcal/mol by default. For the model of 'saturated polarizability' only (see Sect. 25).

arep Parameter for the shell-core model (repulsive antipolarization), in e^{-1}. Also denoted κ.

5.3 Non-bonded fixes

This table has exactly the same format as table `nbfixes` in the parameter file. As it contains only non-bonded fixes applying to atoms in the blended molecules, it is usually shorter or even empty. See Sect. 3.4.4.

Since V2.8a, cook* accepts missing 1-4 terms if they are identical to the normal non-bonded ones.

5.4   Header of molecule

The header consists of keyword `species` followed by a name that is derived from the name of the molecular file (`species.mol` or `species.che`). At the same time, a summary of energy terms and a minimization protocol is printed as comments. Example:

```
!!!!!!!!!!!!!!
species valine
!!!!!!!!!!!!!!
! valine

! mass=117.1485 g/mol
! potential energy summary (in kcal/mol)
!                                            dihedrals:    2.818
! Lennard-Jones:   -0.085       bonds: 9197.366   impropers:   21.795
!       Coulomb:  -33.921       angles:  198.001  aromatics:    0.000
! sum nonbonded:  -34.006   sum bonded: 9419.979    total U:  9385.973
! 27 steps of steepest descent:
!   5 :  U=9095.6037           h=5.89786e-06
```

... lines deleted ...

```
!  95 :   U=-100.85253          h=0.000915236
! 100 :   U=-100.86482          h=0.000864632
!          U0=-100.865
! potential energy summary (in kcal/mol)
!                                            dihedrals:    1.245
! Lennard-Jones:    4.013       bonds:    0.358   impropers:    0.152
!       Coulomb: -113.163       angles:   6.530   aromatics:    0.000
! sum nonbonded: -109.150   sum bonded:    8.285    total U:  -100.865
```

Here, `U` is the potential energy in kcal/mol and `h` the step size used in minimization.

5.5   One species (molecule) data

The molecular data are in the "get data" format (see Sect. 2.4.1). Example:

```
i=0   N=5
ns=11  nc=10  nangles=15  ndihedrals=14  nimpropers=3  naromatics=0
ndependants=0;

! 0 dihedrals, 0 impropers, and 0 aromatics zero or not found
! 25 pairs excluded (1-2 and 1-3)   14 interactions 1-4   0 non-bond fixes
! total charge = 0.00
```

Where

- **i** Species order number

- **N** Number of molecules of the species. The value of option -n is thus passed to MD programs; it has no meaning to blend because blend is able to process only one molecule of each species.

- **config** config=1 denotes that table config will be present containing a configuration of all species. This is generated for blend -n-1, i.e., when one input molecule is split into clusters (molecules) and these are treated separately by cook. The configuration initializer of cook then reads this table and ignores the coordinates of table sites.

- **ns** Number of interaction sites (atoms) in the molecule

- **nc** Number of constraints. Unless -h has been specified, this is the number of bonds.

- **nangles** Number of bond angles (those that are not constrained, see option -h).

- **ndihedrals** Number of dihedral torsions; note that terms with different positive periodicities for the same dihedral angle are merged.

- **nimpropers** Number of improper torsions (does not include aromatics)

- **naromatics** Number of 'aromatic dihedrals' used to keep aromatic rings planar

- **ndependants** Number of 'dependants'. See Sect. 5.10.

- **naxials** POLAR only: number of atom pairs defining the axial polarizability tensor

Details on some tables follow

5.6 Table of sites

The table of sites consists of keyword sites followed by ns lines. An example follows:

```
sites
!i atom   charge     x        y        z     #  excluded *1-4   chir:nam
  0 HC     0.3500  -0.5158   2.4968   0.7781  0
  1 HC     0.3500  -1.7505   2.0381  -0.2218  1  0
  2 NT    -0.3000  -0.7538   2.3600  -0.2184  2  0 1
  3 HC     0.3500  -0.5890   3.1709  -0.8336  3  0 1 2
  4 CH3E   0.0000   2.1067   2.2152  -0.1489  1  *2
  5 CH1E   0.2500  -0.1139   1.1323  -0.6603  5  0 1 2 3 4 +++
  6 CH1E   0.0000   1.2201   0.9911   0.0936  6  *0 *1 2 *3 4 5 +??
  7 CH3E   0.0000   1.9662  -0.2889  -0.2953  4  *2 4 5 6
  8 C      0.1400  -1.1452   0.1554  -0.1754  8  *0 *1 2 *3 *4 5 6 *7
  9 OC    -0.5700  -2.1028   0.6459   0.4269  4  *2 5 *6 8
 10 OC    -0.5700  -1.0187  -1.0326  -0.4039  5  *2 5 *6 8 9
```

The columns are:

i   Order number. It is used in other tables to refer to the site.

atom Atom type

charge Partial charge in electrons

x y x Coordinates of the site

Number of exceptional site–site terms

excluded *1-4 List of exceptional site–site terms. Those marked by * are 1–4 interactions, those without * are exclusions (1–2 and 1–3 pairs that do not interact by bonded forces). Only sites with indexes lower than i are listed.

chir:nam This field appears for chiral atoms only. The first character is the chirality with respect to atom numbering (= the chirality that appears in .mol file), the second character is the alphabetical chirality, the third character is the mass chirality. This field is not used by MD programs. See Sect. 3.8.6, for details.

5.7   Tables of bonds and bond angles

Tables of bonds and angles have a similar format. Shortened examples follow:

```
 bonds
 !i atom    i atom K[kcal/mol/A^2]  r[A]    calc.      Upot
  10 OC      8 C         455.00    1.2300   1.2187     0.058
   9 OC      8 C         455.00    1.2300   1.2349     0.011
```

... lines deleted ...

```
 angles
 !i atom    i atom    i atom  K[kcal/mol] angle[deg]  calc.   Upot   [Kub s]
  5 CH1E    8 C      10 OC        85.00    118.5000   120.15  0.071
  9 OC      8 C      10 OC        85.00    123.0000   123.72  0.013
```

... lines deleted ...

i atom Refers to the notation of the table of sites

K   The force constant. See Sect. 3.

r   Equilibrium bond length

angle Equilibrium bond angle

calc. Calculated values of bond length or angle

Upot The energy of bond or angle potential in kcal/mol

If the angle potential contains the Urey-Bradley term, the force constant Kub and equilibrium length s appear as extra two columns in table angles.

5.8   Tables of dihedrals, impropers and aromatics

Tables of dihedrals, impropers and aromatics have the same format. Shortened examples follow (note that the table of aromatics is empty in this example):

```
dihedrals
!i atom  i atom  i atom  i atom nX  n  K|K[0] calc.angle  Upot  angle|K[1]
10 OC    8 C     5 CH1E  2 NT    2  3  -0.100  171.74     0.191
10 OC    8 C     5 CH1E  6 CH1E  2  3  -0.100   54.88     0.196
```

... lines deleted ...

```
impropers
!i atom  i atom  i atom  i atom nX  n  K|K[0] calc.angle  Upot  angle|K[1]
 8 C     9 OC   10 OC    5 CH1E  0  0  100.000   0.72     0.016  -0.0000
 6 CH1E  7 CH3E  4 CH3E  5 CH1E  2  0   55.000 -32.57     0.122 -35.2644
```

... lines deleted ...

```
aromatics
!i atom  i atom  i atom  i atom nX  n  K|K[0] calc.angle  Upot  angle|K[1]
```

Where:

i atom Refers to the notation of the table of sites

nX Number of matches of wildcard atoms **X** while looking for this torsion in the tables of dihedrals or impropers. For negative **n** (see below), the maximum number over all periodicities.

n The periodicity number. For nonzero **n** the potential form differs from that in the parameter file (see Sect. 3.8.2).

n=0

$$U(\phi) = K(\phi - \phi_0)^2 \tag{5.1}$$

n>0

$$U(\phi) = |K| + K\cos(n\phi) \tag{5.2}$$

n<0 Several dihedrals with different periodicities have been summed and the potential is

$$U(\phi) = \sum_{i=0}^{|n|} K_i \cos^i \phi \tag{5.3}$$

K|K[0] If $n \geq 0$ then the force constant K in kcal/mol, if $n < 0$ then the value of K_0.

calc.angle The calculated torsion angle

Upot The potential energy in kcal/mol

angle|K[1]   n=0 Equilibrium improper torsion angle

n>0 The field is empty

n<0 The values of K_1, K_2, ... $K_{|n|}$

5.9   Table of axial polarizability tensors

Applies to POLAR version only.

```
 axials
 !i atom-->i atom    arep     alpha  alphazz   Esat
   3 S      4 C     0.26998  1.7781  5.6704   0.000
   5 N      4 C     0.22019  0.8672  0.8005   0.000
```

`alphazz` is polarizability in the direction of the pair of atoms, `alpha` in both perpendicular directions. For other parameters see Sect. 5.2.

5.10   Dependants

5.10.1   Linear combination dependants

A dependant is a massless site which is a linear combination of two, three or four other sites: a typical example is site M in the TIP4P water model. `blend` automatically generates dependants in the following circumstances (`D`=massless dependant, `X`=any atom which may be bonded to more atoms):

```
    linear patterns:  X-D-X    X-X-D


                      X-D-X    X-X-X
    planar patterns:    |        |    (as e.g. TIP4P water)
                        X        D


                      X-D-X    X-X-X
    3D patterns:       / \      / \
                      X   X    D   X
```

`blend` itself does not use dependant for anything, these are only exported in the ble-file to be used by `cook`.

Caveats:

1. It is necessary to mark angles not containing the massless sites as constrainable by option `-h` or `-h-1` given to `blend`.

2. Badly determined multiple dependants are not checked. (?? - probably they are in newer versions)

3. For blend, it is generally necessary to add an improper or other terms to guarantee planarity/linearity

Field `err=` in the table of dependants (in ble-file) contains the deviation from linearity/planarity and should be small ($< $ 1e-12 for free simple molecules as e.g. TIP4P water, $<$ 1e-2 if the group is a part of more complex molecule). For 3D patterns, `err` should be always very small ($<$ 1e-12).

NOTES/BUGS/CAVEATS:

1. Dependants do not work well in certain 'nonstandard' cases like `Q-S-Q---C---Q-N-Q`, where Q is supposed to be a dependant.

   Bonds S-C and C-N are missing and should be added by hand from a ble-file generated with the masses of dependants nonzero (may be small; it is necessary to use parallelly this 'mass' version of the force field also if vibrations are calculated).

2. Any site dependent on another dependant is now removed (see #define REMOVEN-ESTED in `blenddep.c`).

Example:

```
dependants
!i atom # i atom weight...
 2 O5   4 0 H5 0.2041  4 E5 0.2959  3 E5 0.2959  1 H5 0.2041   err=5e-13
```

This table means that atom O5 (here of TIP5P water) is a linear combination of two H5 sites and two E5 sites, with weights given (their sum is 1).

`blend` uses a MC algorithm to determine the weights, the error is printed as `err=`.

It is also possible to specify dependants in a special file of extension `.dep`. This cannot be combined with automatic selection of massless dependent sites! Example: species`.dep`:

```
! this is comment
7 8 9 : 11 12 13 14 ! sites 7,8,9 depend on 11,12,13,14 (using site numbers)
H1 : CA1 CA2 CA3    ! site H1 depends on CA1,CA2,CA3 (using atom ID)
```

Another example – rigid molecule of benzene. The che-file `benzene.che` is

```
benzene
HA.1:b1   HA.1      HA.1:b2
 |         |         |
 C6Rn.1--C6Rn.1--C6Rn.1
 |                 |
 C6Rn.1--C6Rn.1--C6Rn.1
 |         |         |
 HA.1    HA.1:b3   HA.1
```

and using file `benzene.dep`

```
 * : b1 b2 b3
```

all sites but b1,b2,b3 are dependent on b1,b2,b3. `cook` then uses constrained dynamics of triangle b1,b2,b3 and all forces acting on other atoms are propagated to b1,b2,b3. See also utility `makedep.c`.

5.10.2 Lone (out-of-plane) dependants

`cook` now supports out-of-plane ("lone", from lone electron pair) dependants, based on 3 parents, see Sect. 11.3.5. The extended format of table "dependants" is below. Note that this must

be prepared 'by hand' because `blend` is not able to generate these dependants automatically.

```
dependants
M 3 M6NE 3 2 O6NE 0.60071493 1 H6NE 0.19964254 5 H6NE 0.19964254 e
L 0 L6NE 3 2 O6NE 1.86320976 1 H6NE -.43160488 5 H6NE -.43160488 -1.73602206 0.86801103
0.86801103 0.00000000 0.63064693 -0.63064693 -0.73718021 0.00000000 0.63988055 0.63988055
0.00000000 0.46490043 -0.63988055 e
L 4 L6NE 3 2 O6NE 1.86320976 1 H6NE -.43160488 5 H6NE -.43160488 -1.73602206 0.86801103
0.86801103 0.00000000 0.63064693 -0.63064693 0.73718021 0.00000000 -0.63988055 -0.63988055
0.00000000 -0.46490043 0.63988055 e
R 0 L5 3 2 O5 1.00000000 1 H5 -0.00000000 4 H5 -0.00000000 0.525 e
```

The "M" lines are the simple (old), "Middle" or linear, dependants

The "L" lines ("Lone") have another 13 numbers appended after the list of three parents and weights (See Sect. 11.3.5, for the notation):

x_1, x_2, x_3

y_1, y_2, y_3

w_z

$t_{x,1}, t_{x,2}, t_{x,3}$

$t_{y,1}, t_{y,2}, t_{y,3}$

Then, letter 'e' should follow to mark the end of the line.

The "R" lines ("Rowlinson") have another 1 number (the LO distance) appended; O should be the 1st site given.

Chapter 6

Problems

6.1 Bugs and caveats

1. Special hydrogen bonding potentials (as in old versions of CHARMM) are not supported.

2. Output tables are printed in reverse order than is natural. This is because of a simple implementation of linked lists.

3. No valence tests are performed on molecules. For instance, a missing hydrogen in "ethane" CH2–CH3 passes unnoticed.

4. Chirality is explicitly supported only for CH1E etc. in extended atom representation. Bad configurations around four-bonded tetrahedral atoms pass unnoticed.

5. If nbfixes for 1-4 interactions are present and `factor14=1` then 1-4 exceptions are always present even if it may happen in some cases that they are identical to normal interactions and some efficiency is lost.

6.2 Trouble shooting

Here we examine several possible sources of problems.

`Problem:` `blend` (UNIX, SGI) gives strange error messages which do not resemble anything described in this manual.

`Probable cause:` You run a system command of the same name. `blend` always prints a line like this:

```
! ------ MACSIMUS / BLEND 2.0k ------ (c) J.Kolafa 1993-2004 ------
```

`Solution:` Use system command `which blend` to determine which `blend` are you using. If this is not the correct `blend`, change your path (remember that if you want to run `blend` from the current directory, you must have the current directory `.` as the first item in your path) or rename `blend`.

Problem: I get error message "parameter file not found - try set BLENDPATH"

Solution: Set BLENDPATH (see Sect. 2.1). If it does not help, try to specify the parameter file name (with full path), see Sect. 2.2.2.

––––––––––––––––––––––––––––––––––––

Problem: I get error message: "bad coordinate - endian?"

Probable cause: You have binary file created on a computer with different endianness, i.e., with different order of bytes (like Sun and PC).

Solution: Try option `-r` with negative argument (see Sect. 2.2.3).

––––––––––––––––––––––––––––––––––––

Problem: I get error message: "atom coordinate out of range" while minimizing.

Probable cause: High-energy initial configuration, atoms overlap

Solution: Try to run more steps of the steepest descent method before switching to the conjugate gradients. This is done by giving option `-m` a negative argument, or by variable `sd` when running `blend` interactively.

––––––––––––––––––––––––––––––––––––

Problem: I have changed `.che` file but `blend` ignores my changes.

Problem: I have changed `.che` file but `blend` suddenly reports strange errors.

Probable cause: `blend` reads `.mol` and/or `.plb` files generated in previous runs.

Solution: Try

```
blend -r2 species.che
```

Option `-r2` forces `blend` to use 2D configuration from `.che` file. species`.mol` file will be ignored because extension `.che` is specified.

––––––––––––––––––––––––––––––––––––

Problem: I minimize a molecule from random initial configuration (or from rather complicated 2D screen coordinates) and I get unexpectedly high energy though the molecule configuration does not seem to be squeezed or locked.

Probable cause: Check arrangement of bonds around tetrahedral atoms. There is a local energy minimum consistent with the values of bond angles in which the four atoms form a square (instead of vertices of tetrahedron) and the "central" atom lies above or below this square, see Sect. 2.2.2.

Solution: (1) Run `blend` interactively and enter `ran=1;` to randomize the positions; if it does not help, try higher values, but be careful. Hot key `:` works in a similar way. (2) Edit the problematic groups in the graphic environment (mark atom, press `m`, move the atom)

6.3 Frequently asked questions

Q: I want to add several water molecules (protons, ...) to a specified place on a protein.

A: I suppose you have already .mol and .plb files (obtained either via .che file, or using `pdb`). Run `blend` again interactively with `-e` option. Example:

```
blend -e -g -s myprotein
```

You will be prompted for editing commands. Type `?` `Enter` to get brief help and then add atoms like in this example which places a water molecule close to atom `O-HC`:

```
af W-O OW -.834 O-HC 2
aa W-H1 HT .417 W-O
aa W-H2 HT .417 W-O
end
```

If you need to add more waters, try input from .edt file instead of typing the commands interactively.

Now you may need to re-optimize the whole structure again. Either type `keep=0` interactively, or re-run `blend` without `-e` option.

In fact, there is command `aw` available so that the above three command can be simplified to:

```
aw O-HC 2
```

Q: I have a protein with several water molecules (or ions...) adsorbed. How to prevent them to 'evaporate' to infinity while minimizing?

A: Create the .jet file with artificial 'bonds' binding water to protein (See Sect. 3.2, option `-j`). Use `blend` with option `-j`. If you wish to have these forces (with much smaller force constant!) also in molecular dynamics, use `-j` option with `-o`, otherwise remove `-j` when the .ble file is generated by option `-o`.

Q: Is it better to split a 'molecule' consisting of parts not connected by bonds using option `-n` or treat it as one big 'molecule'

A: Probably yes. But you must have the corresponding .gol file for `show` obtained WITHOUT option `-n-1` given to `blend`.

Chapter 7

Examples

For simulation examples, see Sect. 17.

7.1 Cluster Na4Cl4

Let us study local energy minima of a cluster of 4 cations Na+ and 4 anions Cl-. Note that `blend` can treat this cluster as one molecule.

Create text file copy `examples/old/Na4Cl4.che` or prepare file `Na4Cl4.che` with:

```
cluster Na4Cl4
parameter_set=sea
NAp1   CLn1
     CL^n1   NA^p1
CLn1   NAp1
     NA^p1   CL^n1
```

First try one of

1. `blend -g Na4Cl4`

2. click `Na4Cl4.che` in the Midnight Commander and select 1

3. `start Na4Cl4.che` and select and select 1

This statement reads `Na4Cl4.che`, derives 3D configuration from it, and minimizes it energetically. You should get a cube.

Then, try:

```
blend -r -g Na4Cl4
```

Now, the starting configuration is random and you will get different local energy minima, like a strip 2×4, a ring (octagon), etc.

You can also randomize and optimize by button [ran] or key $\boxed{:}$, click an atom then after button [move] or $\boxed{m}$ move the marked atom(s) by the middle mouse button (wheel), and optimize by [sd] or $\boxed{,}$.

To calculate normal modes of vibration, use one of

1. `nmf.sh Na4Cl4`

2. click `Na4Cl4.che` in the Midnight Commander and select 2

3. `start Na4Cl4.che` and select and select 2

then switch the modes by $\boxed{\text{PgDn}}$, $\boxed{\text{PgUp}}$ or buttons [<] and [>] around the slider in the right bottom corner of the `show` window.

7.2   Protein in water

This example is obsolete, MACSIMUS is no longer developed with biosimulations in mind.

We want to simulate a protein like crambin in water. The structure and 3D coordinates of crambin are taken from the PDB database (file `examples/old/crambin.pdb`). By running the PDB-converter to the charmm21 force field,

```
pdb -fcharmm21 crambin
```

we get the following two files:

`crambin.plb` 3D coordinates of atoms; coordinates of hydrogens are missing.

`crambin.mol` Molecular file (see Sect. 4.1) in `charmm21`. Chiralities of extended tetrahedral atoms are missing.

In addition, we must have a file describing water copied to our working directory: `blend/data/charmm21/hoh.che` (or `hoh.mol` which is the sane as `blend/data/charmm21/hoh.rsd`). It will simplify later work if this file is renamed to `TIP3P.che`.

```
cp ~/macsimus/blend/data/charmm21/hoh.che TIP3P.che
```

We have to run `blend` twice. In the first step (optional `-g` selects graphics)

```
blend -c -t6 -g crambin
```

the following calculations are performed:

1. Files `crambin.mol` and `crambin.plb` are read.

2. Missing chiralities are calculated (this step is empty for full-atom force fields like charmm22). Note that (unknown) coordinates of hydrogens are not required to do this unless for tetrahedral NP groups, where a WARNING is issued – it can be safely ignored (continue by `c Enter` because H of -NH$_2$ are equivalent.

3. Coordinates of missing hydrogens are filled by (not very good) values.

4. Energy is minimized by 25 steps of steepest descent and 100 steps of conjugate gradient method. This is the default (see option `-m`) which is fully sufficient for obtaining good hydrogen positions. During minimization, positions of atoms with known coordinates are kept fixed; this is the default action when an unknown atom is found (see option `-k` for details). Short value of cutoff `-t6`, i.e., C1=5, C2=7, is sufficient for filling missing hydrogens.

5. If graphics is on (-g), stop showing by pressing $\boxed{.}$ or button [finished] in the graphics window.

6. Files `crambin.mol` and `crambin.plb` are rewritten by new ones with chiralities and hydrogen positions; the old files are backed-up.

In the second step,

```
blend -o crambinw -m30 -n crambin -h -n999 TIP3P
```

the following calculations are performed:

1. Files `crambin.mol` (with correct chiralities) and `crambin.plb` (with hydrogens) are read. Since all atom coordinates are now defined in `crambin.plb`, all atom coordinates are subject of minimization. 30 steps of the conjugate gradient method are performed, which is sufficient to fix inaccurate bond lengths and angles from the PDB file.

2. Unless `TIP3P.mol` and `TIP3P.plb` are already present, `TIP3P.che` is read and the water molecule is energetically minimized (by max 30 steps of the conjugate gradient method). Files `TIP3P.mol` and `TIP3P.plb` are created.

3. File `crambinw.ble` containing a description of the force field for both crambin and water is created. It can be used by program `cook`. Options `-n` (= `-n1`) and `-n999` define the numbers of molecules of crambin and water, respectively (these may be omitted and defined later in MD by `cook`). Option `-h` specifies that the TIP3P water molecule will have the H–O–H angle constrained, that is, it will be treated as hard body.

NOTE: this example is continued by MD simulation in the manual of `cook`, see Sect. 17.2.

Part II

MD simulation: `cook` version V3.7b

`cook` stands for a family of sequential and parallel programs for molecular dynamics of mixtures of molecules described by force fields based on site–site (Lennard-Jones etc.), Coulomb and bonded interaction. `cook` understands the description of molecules generated by `blend`.

Chapter 8

Overview

8.1 Features of `cook`

- Vibrating bonds or constraint dynamics to keep bond lengths constant (Lagrangian formalism or SHAKE)

- Free or periodic boundary conditions, rectangular box

- Linked-cell list method or all-pairs method for non-bonded interactions

- Ewald summation (highly optimized classical r- and k-space sums) or cut-off electrostatic

- Attention paid to errors of integration as well as Ewald cutoff errors

- Convergence profiles of important quantities

- Measuring of many quantities, e.g.:

  - Electrostatic, internal, kinetic etc. energies
  - Virial pressure and pressure calculated by virtual volume change
  - Site–site correlation functions
  - Dihedral angle distribution
  - Bulk conductivity, autodiffusion coefficients
  - Shear viscosity

- Autocorrelation and block analysis of measured data

- Nosé–Hoover canonical thermostat, friction (Berendsen) thermostat, Maxwell–Boltzmann thermostat

- Friction-like isobaric ensemble, Martyna–Tobias–Klein NPT ensemble

- Polarizability – original method with 2nd order predictor (ASPC)

- Parallelization – shared memory with Linux threads

- Batch and interactive modes

- Sophisticated input data unit including a calculator

- Scrolling windows emulation in interactive mode

- Flexible description of force field via program `blend`

- Normal modes of vibration also for models with constraints

For history, see `man/history.txt`

8.2   Compile-time versions of `cook`

The `cook` package allows customizing the executable according to various user requirements. This is accomplished by editing file `simopt.h`: see file `cook/generic/simopt.h` with comments.

Customizing of the most common versions is simplified by the *configurator*, script `configure.sh`, residing in the `cook` subdirectory. The projects created by `configure.sh` must be placed in subdirectories of `cook`. Example:

```
cd ~/macsimus/cook
./configure.sh
```

The following major versions are available:

Ewald summation Standard 3D periodic boundary conditions with Ewald summation to treat electrostatic interactions (of point charges). See Sect. 11.2. Requested by `#define` COULOMB `-1` or -2 in `simopt.h` or during configuration.
WARNING: if LINKCELL is not used, the cutoffs for "optimized water models" must be shorter than minimum half box minus the O–H distance. See option `-x`.

NIBC Nearest-image boundary conditions, no cutoff, all pairs considered. Bug: cutoff corrections not calculated. Suitable for small systems only. NOT TESTED RECENTLY.

Cutoff electrostatic Coulomb forces smoothly cut off to zero. Requested by `#define` COULOMB `0` (or `2`), also the Fennell–Gezelter version (`#define COULOMB 3`). See Sect. 15.3. Recommended executable name: `cookcut`, `cookfg`.

Gaussian charges Standard 3D periodic boundary conditions with Ewald summation to treat electrostatic interactions of Gaussian charges. See Sect. 11.2. Requested by `#define COULOMB -3` (also GAUSSIANCHARGES and QQTAB needed). Tested only for polarizable models.

LINKCELL Instead of the "all pair" method to calculate the pair (r-space) sums, the linked-cell list method is used. Compatible with Ewald and cutoff electrostatics, cannot be combined with FREEBC or POLAR. Since V2.7e, three versions are available:

> **LINKCELL=0** : Each particle control structure (list item) contains also forces; after the r-space calculation they are summed to the contiguous arrays where they normally reside.

> **LINKCELL=3** : Each particle control structure contains pointer(s) to forces in the contiguous arrays. This was the only method available in V2.7d and older.

> **LINKCELL=1** : For serial and PARALLEL=2 this is the same as above. For PARALLEL=1 the array of forces is duplicated to avoid summing to the same forces in parallel; the first instance of forces is indirected and the second one placed in the list item (as for LINKCELL=0).

Which linked-cell version is more efficient depends mainly on the processor and cache architecture, data size, and parallelism.

PERSUM Cutoff longer than half the box (minimum of all sizes) is allowed. Variable `No.molspan` must be set. For NPT, `No.molspan` must include possible change in box size. Inefficient code, incompatible with LINKCELL, NIBC. Optimized water models not supported, the default option `-x0` set. Using `-x1` is only WARNING because the error is caused by wrong treatment of Ewald intramolecular corrections; in case of the r-space Ewald potential small enough at $r = $ box size, the results are accurate. (It would not be too difficult to fix it, but I'm too lazy.)

FREEBC Free (vacuum) boundary conditions, Coulomb forces $1/r$, all-pairs method to calculate pair interactions. Recommended executable name: `cookfree`.

STARS The sign of electrostatic forces is changed so that gravity is simulated instead of charges. FREEBC required. See Sect. 15.4. Recommended executable name: `cookstars`.

GAUSSIAN Calls Gaussian instead of using a force field. FREEBC required. See Sect. 15.5. Recommended executable name: `cookgaussian`.

WALL REMOVED (?), see SLIT and SLAB

SLIT System is periodic in x,y only; i.e., there is no pair interaction between images in the z-direction. Not that the Ewald k-space (if used) is always z-periodic.

SLAB Slab geometry and z-density profiles measurement. Also forces needed to keep atoms in slabs. SLAB=1 turns on some additional forces (incl. walls).

GOLD Simulation at vicinity of metal (ideal conductor) surface. WALL required. See Sect. 15.11. **Not tested recently.**

POLAR Polarizable dipoles supported. Can be combined with FREEBC. See Sect. 15.6.

PARALLEL At this moment, two versions parallelized by 'pthreads' are available, (1) linked-cell list site–site interactions + Ewald and (2) real-space and Ewald running in separate threads. **A MPI version was removed.** (Other versions, as parallel code in FREEBC, have not been rewritten from older platforms.) See Sect. 10.

MARK Special to analyze pair energies. See Sect. 15.13. Not tested recently.

ANCHOR Fixes (anchors) one site and measures forces on it. See Sect. 15.12.

PRESSURETENSOR Direct measurements of the pressure tensor. See Sect. 15.7.

SHAKE, VERLET Several versions of the SHAKE algorithm and kinetic energy formula. See Sect. 11.3.1.

site-site The site–site potential (Lennard-Jones, exp-6, Busing, ...) is selected in the makefile (better said `metamake.mmk`). The respective code is in (from MACSIMUS home) `sim/XXX/sitesite.h` and `sim/XXX/sitesite.c`, where XXX is the name (e.g., `lj`, `exp6`, `busing12`...). The same code is used also by `blend`. The formats of the ble-file changes accordingly.

LOG In direct pair versions (not LINKCELL), allows for separate recording of Lennard-Jones and similar terms and elecrostatic terms. Also, keyword `No.first` is enabled (some functionality of former option `-j`). (Called MONITOR in older versions).

LIBRARY Library for energy and forces that can be called from other programs. Removed.

In addition, a lot of compile-time options define "minor" versions, which may improve efficiency for particular systems and add/remove some extra features—see `cook/generic/simopt.h`.

8.3   Disclaimer

No warranty is provided on anything. If you do not like this program, just do not use it.

Chapter 9

Running `cook`

9.1 Where is

After successful instalation, the executables are in `bin/`.

9.2 Synopsis

The command line to run any version of `cook` is:

> `cook` [options] [sysname[.ble]] simname[.get] [plbname[.plb]] [options]

If no parameter is given, a brief help on options is printed.

The parameters can be given in any order, the only exception is that sysname must precede simname.

Option `X` is switched on (i.e., set to value 1) by one of `-X -X+ -X1` and switched off (cleared to value 0) by `-X-` or `-X0`. The options are case insensitive.

9.2.1 File parameters

sysname Name of the system to simulate. If not given (which is deprecated), sysname=simname is assumed. Refers to file sysname`.ble` generated by `blend` and containing full description of the force field and, if simname`.def` does not exist, also to the definition file sysname`.def`

simname Simulation name. Names of files used in the run are obtained by appending extensions, see Sect. 9.2.3.

plbname.plb Optional name for the input playback file (used with `init="plb"` or `init<0`). If not given, plbname=simname. This parameter comes with extension `.plb` and may refer also to file `plbname.vlb` and configurations plbname.1, plbname.2, ...

9.2.2 Options

-number Overrides `no` for the 1st cycle (e.g., if input data are `no=3;;no=5;` and `-10` is specified, the result is the same as `no=10;;no=5;`).

-@WHO@WHERE Send e-mail to WHO@WHERE if the job finishes or crashes. Utility `mail` must be configured. (The old meaning of `-@` is now in the data as `No.occup`)

-anumber POLAR version: all polarizabilities are scaled by number (in % or on the value of option -_). The default is 1 (100%). Useful for initializing the simulation.

-b Beep, progress, and stop option.

In the batch mode (no option -s):

-bnumber Checks for existence of simname.stp every number-th cycle. If found, finishes cycle and stops (as if ⎡Ctrl-C⎤).

-b-number Checks for existence of simname.stp every [100000*number/(No.s*noint)+1]-th cycle. The default is -b-2 (stop in roughly a minute).

-b0 Does not check for existence of simname.stp.

In the interactive mode (option -s):

-b0 Does not beep

-b1 Beeps (sends `\a` to `stderr`) when finished

-b2 As above and after every "get data" block processed (the default)

-b3 As above and when SIGINT (⎡Ctrl-C⎤) or SIGTERM caught

-b4 As above and prints progress indicator to stderr.

-b-number The same as -bnumber

-c number Method for correcting constraints and measuring constraint errors. See Sect. 11.3. Sum of flags; the default is -c9 = 1+8:

-c1 This flag applies to the Lagrangian constraint dynamics only (irrelevant for SHAKE). The conjugate gradient method `conjgrad` is used to correct constraints after every integration step (default; denoted as -c4 in versions prior V2.4a).
If not set, a SHAKE-like direct iteration method `Scorrect` is used to correct constraints of real configuration after every integration step (denoted as -c3 in versions prior V2.4a).

-c2 If set, constraint errors are measured before every integration step. Denoted as "constr err before" in statistics.

-c4 If set, constraint errors are measured after integration step and before `Scorrect`. Meaningful with Lagrangian constraint dynamics and `Scorrect`. Denoted as "constr err mid" in statistics.

-c8 If set, constraint errors are measured after every integration step. Denoted as "constr err after" in statistics (default).

-dnumber Check site–site distances or initializer bias.

   -d1 Check once/"get data" set (ended by ;)

   -d2 Check once/cycle (and at load)

   -d3 Check once/step (and at load)

   -d ≤ 0 FREEBC: stop simulation if a distance of any atom from $(0,0,0)$ exceeds |number|. The affected molecule is not removed (unlike using `remove.n` or `slab.out` in SLAB version). SLAB: no action
   not SLAB and periodic b.c.: the same as `slab.geom` in the initializer (the default is droplet).

   The default is **-d0** = no action.

-e number Number of items in the convergence profile. The default depends on compiling switches and file simname.`cpi` but can be reduced, or completely turned off by **-e0**.

-f In the playback mode (options **-m0**, **-m1**): Calculate energy and forces from each configuration read. If POLAR, the precision is controlled by `scf.epsx,scf.omegax`.

   WARNING: Pressure for models with constraints (bonds) is incorrect because the virial of constraint forces, (15.12), is missing. Use molecule-based virtual volume change instead.

   The kinetic part of pressure is replaced by the averaged value given by temperature `T`, including the kinetic pressure correction, see (15.15). (The kinetic pressure correction applies for `corr&4`; in turn, the kinetic pressure is simply given by the ideal gas EOS of 1 molecule.)

   Thus, in the *NVT* ensemble for models without constrained bonds, pressure is correct (although not the same as in the original simulation because of kinetic energy fluctuation).

   The virtual volume change method (see `dV`) is applicable with the same kinetic term and correction.

   Pressure tensor is not calculated (the value printed is incorrect).

-g number REMOVED IN V3.0

-h number Centers all atoms of given valence to its neighbors. To be typically used as **-h4** to remove wrong pyramidal configurations of methane and other sp^3 carbons, or similarly as **-h6** for PF$_6$.

-i number $\boxed{\text{Ctrl-C}}$ interrupt handler option. (__EMX__: $\boxed{\text{ESC}}$ hit handler)

      -i-1 Disable $\boxed{\text{Ctrl-C}}$ handler.

      -i0 Interrupt and ask for the appropriate action, this is the default for the interactive run (see option **-s**) in serial versions. Not recommended for the PARALLEL version.

      -i1 Finish the running cycle and read the next set of data. This is the default in interactive run in PARALLEL versions.

      -i2 Finish cycle, then save all and stop. This is the default in the batch run.

      -i5 Finish step, then save all and stop. Recommended only if the next run will be with `init=2` because some measurements are affected by interrupting the measurement cycle.

   WARNING: OK for unix/linux. System-dependent and sometimes unexpected behavior otherwise. Can be disabled by deleting `#define SIG` in `main.c`. See Sect. 9.2.7.

-jnumber `(100)` Changed in V3.3r.

number ≥ 0 The potential energy minima (Emin, EvdW) of non-bonded potentials (*e.g.*, Lennard-Jones), are multiplied by factor number (in % or on the value of option -.)

number < 0 The atom sizes (*e.g.*, van der Waals radii or Lennard-Jones σs), are multiplied by factor |number (in % or on the value of option -.).

Meaning in versions prior V3.3r: see variable `No.first` and version LOG.

-knumber Force constant for keeping ("fixing") selected sites in place. The units are K/Å² (not kcal/mol/Å²). See Sect. 15.1.

-k0 Not ANCHOR: selected sites are fixed in place exactly (constraint dynamics). Cf. ANCHOR with more options. (In old versions, this functionality needed ANCHOR version and -k-1.)

-k-number ANCHOR version. See Sect. 15.12. **Negative** sum of flags:

0 Anchoring is performed without measurement/recording.

1 Constraint forces marked as `A00x, A00y, A00z, A01x`, etc. (see the prt-file), are measured using the standard statistics module.

2 Enables recording of the above constrained forces in the cp-file. Write the needed keywords (not necessarily all) to SIMNAME.cpi. **Both** `A00x` etc. in the cpi-file **and** -k-2 are required for recording.

4 ASCII file SIMNAME.anc is printed. This was the only option (and default for -k0) in V3.6a and older.

8 Several anchored sites in one molecule is warning only (nor error).

-lPEGFAVC ASCII dump. EGFAVC stand for decimal digits denoting:

C writes the configuration to file SIMNAME.asc, in Å

V writes the velocities to file SIMNAME.vel, in Å/ps

A writes the accelerations to file SIMNAME.acc, in Å/ps²

F writes the forces to file SIMNAME.for, in kK/Å= 1.3806485e-13 N

G writes the gradient of energy (numerical derivative) to file SIMNAME.gra (to compare with SIMNAME.for) unit as above (SLOW!)

E The sum of F and G (should be numerical zero)

P POLAR: the Drude amplitude (separation of the Drude charge)

All dumps are in the internal program units, as indicated. The digits denote: digit=0 do not write, digit=1 denotes the maximum precision in the g-format, digit=2..9: number of decimal digits in the f-format.

EXAMPLE: -l1003 will dump the configuration (to 3 dec. digits) and forces (in max-precision g-format)

The six columns of the output file are:

i Consecutive number

`mol` Species number

`x,y,z` Vector

`|vector|` Its absolute value

-m`mode` Mode of calculation and integration method.

-m0 Configuration files `plbname.#`, (where # is a decimal number, see `dt.cfg`) previously recorded are read and (partly) analyzed. Cf. `-m1 below`.

-m1 Frames from `plbname.plb` are read instead of simulating, cf. `dt.plb`. See variables `reread.from`, `reread.to`, and `reread.by` for control which frames to read and analyze. For both `-m0,-m1`, it is recommended to change the simulation name to avoid overwritting files of the original simulation; also, `-w0,-y0, -r1` are recommended. See also `-f` (forces).

-m2 Verlet integrator (with SHAKE if constraints are present). Requires #define SHAKE in `simopt.h`. This is the default.

-m3,..,-m9 The order-value Gear predictor-corrector method is used with Lagrangian constraint dynamics (if constraints are present). See also see `gear.init`. The value of `-m` is called `GearOrder` in the source code.

-n Playback options, sum of flags:

1   Write also velocity playback (`plbname.vlb`). All parameters for the playback (`plbname.plb`) apply: `dt.plb`, option `-y`. Note that the velocities with `-m2` (Verlet/SHAKE) are at times $t - h/2$, which is not in sync with the positions unless flag 2 is selected.

2   Positions in `plbname.vlb` are shifted by $-h/2$ (i.e., $h/2$*velocity is subtracted). To be used with `-m2` (Verlet/SHAKE) to provide `plbname.vlb` and `plbname.plb` in sync. (A WARNING is printed if they are not in sync.)

4   Positions in `plbname.plb` are centered. This trick increases the accuracy of 1 bit, but is not fully supported by utilities. The second float in the header is -6.

8   POLAR only: Write also the positions of Drude charges (absolute, not relative to the atom) in file (`plbname.dlb`). All parameters for the playback (`plbname.plb`) apply: `dt.plb`, option `-y`.

-o`number`   0   Simulation is started without asking even if the lock file sysname`.loc` exists. Use with caution!

1   If the lock file sysname`.loc` exists, the simulation does not start and an error message is printed. This is the default.

-p`KPC` (for POLAR version: K,P,C are decimal digits)

-p`C` (for nonpolar version) Selects various predictors.

C   **With Verlet+SHAKE** (option `-m2`): C is the length of the predictor for calculating the velocities. Applies for the Nosé–Hoover thermostat, otherwise not used. The higher order, the better time-reversibility and total Hamiltonian conservation. C=0 is no extra history ($v(t) \approx [r(t) - r(t - h)]/h$). The default is C=2.

With Gear + Lagrangian constraint dynamics (option `-m > 2`): Order of predictor for Lagrange multipliers (0=no prediction, default=9=value of `-m`). Affects efficiency only, not accuracy. The best results are probably obtained with the same order as Gear provided that parameter `eps` is optimized.

P POLAR only: the predictor order (or method):

   0 Car-Parrinello-like (extended Lagrangian) method, unfinished implementation, do not use. See Sect. 15.6.2.

   1 No prediction (previous value).

   2 Always stable predictor-corrector (ASPC). The predictor length is given by K, the default is K=2. This is the default.

  3,4 Higher-order predictors, may be useful with Gear integration. Only P=3 is alowed with Verlet (but not recommended).

K Predictor length. For P=2 (ASCP), the higher K, the better energy conservation.

`-q`number Charges are rescaled by |number| (in % or on the value of option `-_`). The default is 1. Negative number removed in V3.1i, see `el.bg` instead.

`-r`order Record configurations for further analysis (see option `-m`) up to (order − 1)-th derivative: `-r1`means positions only, `-r2` (the default since V3.5i) denotes positions and velocities, etc.; the value of option `-m` must not be exceeded. The configurations are recorded in files simname`.1`, simname`.2`, simname`.3`, etc., with the frequency `dt.cfg` ps. `dt.cfg` should be an integer multiple of `h*noint`.

`-s`number This option sets the interactive mode: input is keyboard (instead of a get-file), output the screen (instead of a prt-file). If `SCR` has been #defined at compile time, you can use scrolling to watch the previous output. Type `$?` from "get data" to get help or see the `blend` guide for details. number is the buffer capacity in kbytes, missing number or number=1 means 31 kbytes (DOS) or 80 kbytes (other).

`-t` Enables detailed runtime measurements (real time, i.e., it depends on the system load). In the PARALLEL versions sampling via `clock()` is also available; it reports critical portions of parallel times.

(In case of compatibility problems, use compile-time option -DCHEAPTIME, and the time will be in 1 s resolution only. Even if the total time is long, there may be grid/interference errors if the resulting times are obtained using several cycles.)

`-u`number Select which bonds will be constrained (rigid) and which will be vibrating. See the `blend` manual for details.

Artificial bonds written with K=0 are always constrained. These are typically created by `blend` from bond angles with hydrogens and it does not make sense to combine constrained angles with vibrating bonds.

   `-u0` All bonds are rigid (constrained). This is the default.

   `-u`K_{lim} Bonds with the force constant K [in kcal/mol/Å^2, formula $u = K(r - r_0)^2$] less than $K_{\text{lim}} + 0.5$ will be vibrating, bonds with a higher force constant will be constrained.

-u-$\tilde{\nu}_{\max}$ (New in V3.4d.) Bonds with wavenumber $\tilde{\nu}$ [in cm^{-1}] less than $\tilde{\nu}_{\max} + 0.5$ will be vibrating, bonds with a higher frequency will be constrained. The frequency is calculated simply from both atoms bonded by K given in the table "bonds" (no normal vibrations are calculated). In addition, any equalization is ignored. Typically, `-u-2500` is good to constrain bonds containing hydrogens. Use option `-v4` to print a full protocol about vibrating/rigid bonds.

-v*number* Verbosity level. Sum of powers of 2. The default is 3

   -v0 Minimum verbosity. Only basic system+run info printed

   -v1 Runtime info, initialization protocols

   -v2 Brief statistics (was detailed prior V2.4a; use `staprt` for detailed statistics), message 'playback written', more initialization details; dihedral distribution and similar (if compiled so)

   -v4 Verbose details on site–site potentials and energy terms, copy of input ble-file, constraint optimization (SHAKE), detailed statistics: for debugging purposes

   -v8 POLAR: convergence

   -v16 = -v0x10 POLAR: file .pol with some summary

   -v32 = -v0x20 POLAR with `dV`: extensive files with induced dipoles:

      `.run.pol` Running induced dipole moments in program units (0.0117501 D)
      `.ex.pol` Exact (iterated) induced dipole moments in program units. NOTE: this is an average of the induced dipoles for `V+dV` and `V-dV`. `dV` thus must be small enough.
      `.err.pol` Errors in induced dipole moments in program units

   -v64 = -v0x40 Protocol on momentum and angular momentum and their setting to zero. See also variable `drift`, codes for .cpi and see Sect. 32

   -v128 = -v0x80 Raw dump of configuration, charges, and forces (incl. POLAR, after every evaluation, i.e., during iterations). Huge files of form SIMNAME.#.dump are created. To avoid problems if used unintentionally, # is limited to 0..9, then the program stops.

-w*number* Write configuration (simname.cfg).

   -w0 Don't write.
   Note that if several sweeps (;;; in the data) are performed, **they are started from the same configuration** because a single ; means restart from simname.cfg.
   BUG: With (any) predictor (ASPC or Nosé-like), the predicted values are not reset. This means that the trajectory actually is not exactly the same.

   -w1 Always write (default). Before the first write, a backup (simname.cfg˜) is made.

   -w*number* As `-w1` and the available disk space is checked before selected files are written. If not enough space is available, it waits several minutes (more for the second time) and then checks again. The available space is compared to a rather pessimistic estimate of space needed plus underline number kB added (to be even more pessimistic). -w2 should be enough unless your co-workers often fill the disk in a very greedy way. The algorithm is not 100% reliable, though. It relies on a system call to df; if this fails for any reason, it is considered as no space and the program keeps waiting. A warning is printed after space has appeared again; for apparent reasons, nothing is printed when the no-space condition is encountered.

Note: `-w2` used to mean write only if the configuration has changed, but it become difficult to determine this condition so that this option was removed.

-x<u>number</u> `(5; 0 for PERSUM version)` Changed in V2.7b. Sum of flags:

1   Use optimized code for registered rigid water models. Not available with POLAR and LINKCELL and for models TIP5P, NE6. May give inaccurate results with PERSUM,see Sect. 8.2. Currently supported models are:

`TIP3P geometry` : TIP3P, HOH (=alternate name for TIP3P), SPC, SPCE

`TIP4P geometry` : TIP4P, TIP4P05 (=TIP4P/2005), TIP4PEW (=TIP4P/Ewald), TIP4PICE

`ST2` : ST2 (with the switch function, not available with LINKCELL, no pressure nor pressure tensor available – use the virtual volume change, cf. `dV`)

Note that a model with name TIP4P will be treated as TIP4P geometry irrespective of the parameter values (cf. option `-x4`)
If not set, the general code is used.

2   Keep H-O and H-H water-water Lennard-Jones terms. Missing flag 2 will cause clearing these interactions. This option is intended for TIP3P and HOH (= other name for TIP3P) leading to the original version of TIP3P. A warning is printed if TIP3P is used with H-O and H-H LJ terms, however, no automatic clearing of these parameters is performed.

4   Failed doublecheck of registered water models is ERROR (WARNING if `-x4` is unset)

Examples: use `-x5` for rigid optimized TIP3P model without H-H and H-O LJ terms; use `-x0 -u9999` to make a normally rigid model of water flexible.
WARNING: if LINKCELL is not used, the cutoffs for "optimized water models" must be shorter than minimum half box minus the O–H distance. If you cannot guarantee this, use `-x0`.

-y<u>number</u> Write playback files option.

<u>number</u>=0 Don't write (even if `dt.plb` is specified).

<u>number</u>>0 Write playback file(s), <u>number</u> is the number of molecules to write. The default is "big" which means that all molecules are written (in version prior 2.4g, this was `-y-1`). File names are always <u>simname</u>`.plb` (in version prior 2.4g, there were single-molecule files <u>simname</u>`.p00`, etc.). At the same time, `molcfg` is called to produce mol and gol files for `show` unless `init=0,1`.

<u>number</u><0 As above but `molcfg` is not called. (Support for layers of water was removed; see sim/old+misc/simils-plb.c and man/option-y.tex).

-z<u>number</u> Seed for random number generator. If `-z` is not specified, the seed is derived from time so that it is different for each run. Random numbers are needed for the configuration initializer and cross section measurements (certain modes only).

-^<u>number</u> (0) POLAR only: The value of `scf.omega` (in % or on the value of option `-_`). The default is 0, which means that the value guaranteeing ASPC stability for any converging SCF is chosen. (Different meaning in older versions.)

-\number (0) Removed in V2.7f. Use environment variable NSLOTS instead.

- number (100) The denominator for scaling factors defined by -q, -j, -a.

+number (0) Special for debugging with -DCHECKHEAP=-2: AllocRange parameter, see gen/alloc.h for details.

9.2.3   File extensions

Two names are specified at the command line. The system name (sysname) refers to sysname.ble generated by `blend` and containing full description of the force field. The simulation name (simname) refers to the specific MD run.

A list of extensions appended to simname follows. For instance, if simname=crambin1 then file crambin1.cp contains the convergence profile.

.1 .2 .3 ... Recorded configurations, see option -r and variable tcfg

.anc ANCHOR: measured forces. See Sect. 15.12.

.asc Ascii dump of the configuration, see option -l

.box File for controlling the box size during simulation, see variable tau.rho.

.cfg Configuration. Contains the whole configuration and some additional information.

.cp Convergence profile. Record of total, potential, and bonded energies, pressure, temperature, and optionally more. Note that the term "convergence profile" here means just time dependence of quantities, not running averages of any kind. See Sect. 14.2 for user-info, see Sect. 16.2 for file format.

.cpa Ascii dump of (block-averaged) .cp file. Generated by `showcp`, in some special cases directly by `cook`.

.cpi Input file of optional additional items to be recorded in the convergence profile (.cp) and statistics (.sta).
Example:

```
! ATOM-ATOM DISTANCES will be recorded:
! atom1 atom2 [name]
  1     123   CC1 (this is comment, too)
  1     124   CC1_nbr
! ADDITIONAL VARIABLES will be recorded:
elst
LJ
+Lz
A00x
```

In the convergence profile, only the first 4 letters of the name are used (e.g., CC1_nbr is truncated to to CC1_). If the third column is missing, names ss1, ss2, ... are used.

If + is placed in front of the name or site–site pair, the variable will be also recorded in the statistics (this is redundant because this statistics can be always calculated by `showcp`).

Names starting with A apply for anchoring (#define ANCHOR) for the respective names.

More keywords are listed below. To program a new variable of type `double`, see `sim/simmeas.c CPItab[]`.

`Epnc` potential energy without cutoff corrections [J/mol]

`Eusr` user potential energy (see userforces.c) [J/mol]

`elst` electrostatic energy (POLAR: excl. self-energy) [J/mol]

`bond` bonded energy [J/mol]

`Eext` extended degrees of freedom energy (Nose-Hoover, MTK barostat) [J/mol]

`Ekin` kinetic energy [J/mol]

`U` Internal energy = Epot + Ekin with cutoff corrections [J/mol]
(It differs from the total energy, provided in the 1st column `Etot` of the convergence profile, so that it does not contain the extended degrees of freedom, Eext)

`Unc` Internal energy = Epot + Ekin without cutoff corrections [J/mol]
(It differs from the total energy, provided as the 1st column of the convergence profiles, so that it does not contain the extended degrees of freedom)

`H` enthalpy = Epot + Ekin + pV with cutoff corrections [J/mol] see Sect. 14.10.8

`Hnc` REMOVED Enthalpy without cutoff corrections [J/mol]

`fix` The potential keeping selected sites in place [J/mol]

`Pref` pressure as defined by variables virial, corrected [Pa]

`Pvir,PECC,Psc,Pcor` ECC only (to be updated) [Pa]

`Pdv` pressure by virtual volume change (also PdVm/PdVa) w. cutoff corr. [Pa]
PdVm= molecule-frame, PdVa=atom frame

`Pevc` pressure (based on el. virial) w. LJ cutoff corr. [Pa]

`Pevn` pressure (based on el. virial) w/o LJ cutoff corr. [Pa]

`V` volume [p.u.=Å^3]

`CMdr` corrected center-of-mass drift [Å=p.u.], abs. value of vector calculated from corrected components as given by `drift&7`. Use use `CMx,CMy,CMz` for CM in not corrected coordinates.

`vdr` Corrected velocity drift [p.u.=Å/ps], abs. value of vector calculated from corrected components as given by `drift&56`. Use `LMx,LMy,LMz` for not corrected coordinates.

`AMdr` corrected angular momentum drift [p.u.], abs. value of vector calculated from corrected components as given by `drift&448`. Use use `AMx,AMy,AMz` for not corrected coordinates.

`omdr` corrected angular velocity drift [p.u.=1/ps], abs. value of vector calculated from corrected components as given by `drift&448`.

`CMx,CMy,CMz` center-of-mass [Å], use `lag.CM` to request, cf. `slab.sp`.

`LMx,LMy,LMz` linear momentum [p.u.], use `lag.LM` to request.

`TLM` cluster translational temperature (from CoM)

`AMx,AMy,AMz` angular momentum [p.u.], use `lag.AM` to request.

`TAM` cluster rotational temperature [K]

POLAR only variables to be included in CP:

`pstd` 1st iteration stderr [p.u.] = running error of induced dipoles (predicted minus corrected/iterated), standard deviation; the statistics is called "polar 1st iter maxerr"

`pmax` 1st iteration max err. [p.u.] = running error of induced dipoles (predicted minus corrected/iterated), maximum; the statistics is called "polar 1st iter stderr"

`plst` last iteration stderr [p.u.] = running error of induced dipoles (predicted minus corrected/iterated), standard deviation; the statistics is called "polar last iter maxerr" (zero if there is only 1 iteration)

`plmx` 1st iteration max err. [p.u.] = running error of induced dipoles (predicted minus corrected/iterated), maximum; the statistics is called "polar last iter stderr" (zero if there is only 1 iteration)

`Pstd` Polar stderr (vs. iterated solution, scf.test=1) [p.u.] (in older versions in virtual volume change method)

`Pmax` Polar max err. (vs. iterated solution, scf.test=1) [p.u.]

`rate` averaged convergence rate, with scf.test=1 or virtual volume change method

`iter` number of iterations (per MD step, not testing notor virtual volume change method)

`X_hf` high-frequency susceptibility by the direct response to an external field

`Tpol` for Lagrangian (Car-Parrinello-like) polarizability, kinetic energy of the additional degrees of freedom [K], DEPRECATED

`Pvxx,Pvyy,Pvzz,Pvyz,Pvzx,Pvxy` PRESSURETENSOR: Components of the virial part of the pressure tensor, in Pa

`Hz` Enthalpy = Epot + Ekin + p_zz V, without cutoff corrections [J/mol]

`Pkxx,Pkyy,Pkzz,Pkyz,Pkzx,Pkxy` Components of the kinetic part of the pressure tensor (site-based)

`PKxx,PKyy,PKzz,PKyz,PKzx,PKxy` Components of the kinetic part of the pressure tensor (molecule-based)

`Ptxx,Ptyy,Ptzz,Ptyz,Ptzx,Ptxy` Total pressure tensor

`Lx,Ly,Lz` Sizes of the simulation box [p.u. = Å]

`Jx,Jy,Jz` Current density (x, y, z components separately) in $A\,m^{-2}$, `lag.J` or `lag.M` must be set.

`Jx0,Jy0,Jz0` Current density (x, y, z components separately) in $A\,m^{-2}$ for group 0 (use `group[*]=0` in the data)

`Jx1,Jy1,Jz1` As above for group 1

`Jx2,Jy2,Jz2` As above for for group 2

`Mx,My,Mz` Simulation cell dipole moment (x, y, z components separately), `lag.M` or `lag.J` must be set. The dipole moment of ions is recalculated to their centroid (point ions have zero dipole moment). Thus, there are no jumps in $M_\alpha(t)$ if an ion moves to another periodic image. Their statistics is available with `lag.M`. These dipole

moments are calculated independently on (optional) Ewald summation after a cycle. Note that the dipole moments recorder every step start during Ewald summation are not available here.

Unit: p.u. $= 0.0117501021272\,\mathrm{D} = 3.9194121835\times10^{32}\,\mathrm{C\,m}$

`Mx0,My0,Mz0` Simulation cell dipole moment in p.u., group 0 (use `group[*]=0` in the data). This is the raw dipole moment $\sum_i q_i \vec{r}_i$. Thus, function $M_\alpha(t)$ has jumps whenever an ion moves to another periodic image

`Mx1,My1,Mz1` As above for group 1

`Mx2,My2,Mz2` As above for group 2

Caveat: For a mixture of dipolar molecules, it holds `Mx = Mx0+Mx1+···`. However, for brine with group 0 = point ions and group 1 = water, it holds `Mx = Mx1` instead.

`NHxi` Nosé–Hoover degree of freedom $\xi = \log(s)$, dimensionless

`NHdx` time derivative of the Nosé–Hoover degree of freedom $\dot\xi$, in 1/ps. This is the velocity-Verlet value from the final trajectory $\dot\xi(t) = [\xi(t + h) - \xi(t - h)]/2h = [\dot\xi(t + h/2) + \dot\xi(t - h/2)]/2$

`MTlx,MTly,MTlz` MTK thermostat/barostat [8] degree of freedom λ (diagonal x, y, z components of the λ tensor)

`MTdx,MTdy,MTdz` time derivatives of the MTK thermostat/barostat [8] degree of freedom $\dot\lambda$ (diagonal components of the tensor). This is the TRVP-predicted value (not the velocity-Verlet value from the final trajectory)

SLAB version variables to be included in CP:

`cx,cy,cz` SLAB: drop,trickle,slab,cavity,tunnel,bals(inverted slab) center as determined by "autocenter" [Å]

`symx,symy,symz` center of species $\leq$ `slab.sym` w.r.t. "autocenter" [Å]

`Pt` tangential pressure (Ptxx+Ptyy-2*Ptzz)/3 [Pa]

`Ptv` virial part of tangential pressure (Pvxx+Pvyy-2*Pvzz)/3 [Pa]

`Pw0,Pw1` pressure to the bottom and top wall, respectively [Pa]

`Wcl` $\mathrm{d}G/\mathrm{d}\sigma$, reversible work per unit interface area [Pa]

`clz0,clz1` centers of cleaving walls, in the units of L_z

`clf0,clf1` forces to the cleaving walls [p.u.=K/Å]

`Rgxx,Rgyy,Rgzz` RGYR: averaged diagonalized moment of inertia of species 0

These are for the ECC version only (to be fixed after renaming):

`Pvir` ? Modification of `Pvir` above: virial pressure except epsf(V) dependence [Pa]

`PECC` The ECC pressure, the same as `ECC P [Pa]` in statistics

`Psc` Conventional pressure without any epsf-based ECC terms, the same as `ECC Pscaled [Pa]` in statistics

`Pcor` The ECC correction (to be added to Pvir) [Pa]

These are applicable in direct evaluation of forces (not LINKCELL); with water `-x0` needed. WARNING: not tested recently, #define LOG needed. Deprecated variable `No.first` = number of molecules in the first group should be in input data. `No.first` needs also `cache=1`, no LINKCELL. With Ewald only r-space is supported.

Ep0 potential energy first `No.first` molecules [J/mol]

EpX potential energy `No.first` vs. rest molecules [J/mol]

Ein intramolecular potential energy [J/mol] (NEW in V2.8c)

el0 electrostatic energy first `No.first` molecules (w/o self) [J/mol]

elX electrostatic energy `No.first` vs. rest molecules (w/o self) [J/mol]

LJ   non-bonded (LJ) energy excl. elst. [J/mol]

LJ0 non-bonded energy excl. elst. first `No.first` molecules [J/mol]

LJX non-bonded energy `No.first` vs. rest molecules [J/mol]

bon0 bonded energy first `No.first` molecules [J/mol]

`.cpz` Convergence profile in packed format (see variable `CPnbit`). Use utility `cppak` to convert it to `.cp` (and vice versa). Utility `showcp` understands the cpz-files as well so that the analysis of convergence profiles is transparent.

`.def` System `definition` (or `def`aults). If simname.`def` is not found, sysname.`def` is tried. The file should contain 1 set of input data (see Sect. 9.2.5). Note that numbers of molecules (species) `N[*]` can be given here, but not in `.get` file or by keyboard input. In addition, this file is read twice so that you cannot modify the program default here (e.g., the cook default is `h=0.001`, but `h*=2` in this file would lead to `h=0.004`).

`.dih` DIHedral angle distributions, binary.

`.dia` DIhedral angle distributions, Ascii dump.

`.dcp` Dihedrals Convergence Profile. The dihedral values, in degrees, are printed every cycle to simname.`dcp` as ASCII. Which dihedrals are dumped is selected by simname.`ddh` and variable `dih.dcp` in input data. Must be compiled with DIHHIST=-1, applies to species=0 only.

`.ddh`

`.mar` Define DiHedrals (species=0 only). Must be compiled with DIHHIST=-1. Selects dihedrals to dump to simname.`dcp` and record (in simname.`dih` and `.dia`). If does not exist, all dihedrals are dumped (lot of data!). File of lines as, e.g.:

```
1  156   phi = -154.04  C5 - N6 - Ca10 - C15
2  145   psi = -124.77  N6 - Ca10 - C15 - N16
0  141 omega = -177.79  Ca10 - C15 - N16 - Ca20
```

The first column is irrelevant provided that it is nonzero (lines with 0 are ignored), so are lines beginning by '!'. The second column contains the dihedral number, as from the sysname.`ble` file, numbered from 1 to `ndihedrals`. The rest of the line is irrelevant. For the peptide backbone, angles phi,psi, the output of `ramachan` (the sysname.`mar` file) has exactly the correct format and can be renamed to simname.`ddh`. (`ramachan` uses the ble-file to derive this information).

`.dpr` Density profile, binary. Requires SLAB defined.

`.fix` To fix certain atoms. See Sect. 15.1. Note that the file format depends on ANCHOR and option `-k`.

`.for` Ascii dump of the forces, see option `-l`

`.g` For `.g` extension, see `rdfg` in the utilities manual.

`.g1, .g2, ...` Alternate names of input files (instead of <u>simname</u>`.get`). See options `-g` and `-f` (see Sect. 9.2.2).

`.get` Input data. <u>simname</u>`.get` contains one or more sets of input data (see Sect. 9.2.5) ended by 'quit=1;' or EOF. Irrelevant if `-s` is specified (input from keyboard).

`.loc` A lock-file to prevent running two instances of `cook` with the same <u>simname</u>. Normally removed when `cook` finishes by any controlled way (incl. ERRORs and $\boxed{\text{Ctrl-C}}$). Remains if `cook` (or computer) crashes or is killed by `kill -9`. It should be then removed before restarting `cook`.

`.pol` POLAR: Summary info, requires option `-v16`

`.run.pol`

`.ex.pol`

`.err.pol` POLAR: Extensive tables of induced dipoles, see option `-v32`

`.prt` Output protocol. Irrelevant if `-d` or `-s` are specified (output to display), changed to `.prtx` if option `-f` is given

`.prtx` Output protocol for measurements while reading playback (typically autodiffusion, conductivity, clusters, cross sections, structure factor). Prior V3.0 called `.sfd`

`.rdf` Radial distribution function(s), binary, see `rdfg` in the utilities manual.

`.s-s` List of atom types and groups for measuring radial distributions functions. See Sect. 14.7.

`.sfr` Structure factor (radial or sphericalized), see option `-f`. Note that the k-vectors are 'circular', i.e., they mean a number of waves in 1 Å multiplied by 2π.

`.sfsd` Structure factor (full 3D), see option `-f`. Note that the k-vectors are 'circular', i.e., they mean a number of waves in 1 Å multiplied by 2π.

`.sfd` Prior V3.0: Output protocol with option `-f` (Structure Factor and autoDiffusion)

`.sta` Statistics. Contains measurements recorded for autocorrelation and error analysis (module `statics`)

`.stp` See option `-b`

`.*~` Backups. Most output files (.cfg,.sta,.rdf,.dih...) are backed up before writing; this is done every data set (; in get-file). The .prt file is backed up once when `cook` is started. Files .cp, .plb, .pol ... are not backed up if they are appended. Backup names are created by appending ~.

`.wid` Results for the Widom method (WALL only). Not tested recently.

`.z` Density profiles in ASCII, as generated from .dpr.

9.2.4   Program flow

1. Options are analyzed.

2. Files sysname`.ble` and sysname`.def` (or simname`.def`) are read and force field tables are constructed.

3. A set of input data (each set is ended by `;`) is read from simname`.get` or from console. If `key="quit"` is specified, or EOF is reached, the program stops (see variable `key` for other options).

4. The initial configuration is either generated from scratch (for `init>=3`) or read from simname`.cfg` (if `init<=2`); see variables `init` and `load` for more options.

   simulation mode If option `-m` is 2 or more (default = `-m2` = leap-frog + SHAKE), the simulation starts.

   reread mode If option `-m0` or `-m1`, previously saved configurations are re-read instead of simulating. See options `-m`, `-f`, and variable reread for more info.

5. Ewald only: Unless `el.test=0` or `el.test=-10`, the control switches to another module that tests the Ewald summation and sets the parameters. This module accepts another data (see Sect. 11.2) but one of them is `el.test`: once `el.test=0` is specified, this module is abandoned and the control continues by the next step.

6. The specified number of steps (given by variables `no` and `noint`) is performed unless interrupted by pressing Ctrl-C or presence of file simname`.stp`. This is called "a job".

7. The configuration is stored to file simname`.cfg`. Files with statistics, convergence profile, etc., are closed. The protocol file `.prt` is flushed.

8. In the simulation mode, control continues by step 3. Most variables retain their values, the notable exception being `init` which is changed to `init="cont"` after a job. In some cases (e.g., calculation of normal vibrations or in the reread mode), continuing (beyond `;`) is undefined.

9.2.5   Input data

The input data are in the "get data" format, see Sect. 2.4.1. Briefly, it consists of a set of assignments ended by a semicolon; expressions can be written at the right hand sides of the assignments. A string with a mnemonic value can be used instead of an integer switche; e.g., `thermostat="Nose"` is the same as `thermostat=2`.

The following list contains all variables that can be used either in sysname`.def` (one set of data containing the defaults), or in simname`.get` or interactive input from keyboard. The values in parentheses are the default values.

WARNING: sysname`.def` is parsed twice so that you cannot modify the program default here (e.g., the cook default is `h=0.001`, but `h*=2` in sysname`.def` file would lead to `h=0.004`).

Example (of simname`.get`):

```
thermostat="friction"
no=10 tau.T=0.01;               ! 10 cycles of fast cooling
no=20 tau.T*=5;                 ! 20 cycles of 5*slower cooling
```

Note: in case of errors as e.g.:

```
ERROR data:bad identifier or syntax
ERROR data:bad number or expression
```

it may be difficult to say where the error occurred. Command ?= in the input data may help. Note also that there may be wrong data in the ble-file. E.g., WARNING in the ble-file prevents its correct reading.

Any declared variable except array can be used on the right-hand side in an expression. To enable calculations, there are the following auxiliary variables declared:

```
double a,b,c,aux,x,y,z;
int i,j,k,n;
```

In addition, $pi = \pi$ is available.

Example:

```
x=0.01                  ! auxiliary, to mean cycle in ps
noint=6                 ! steps/cycle
h=x/0.001               ! timestep in ps
no=20/(h*noint)+0.5     ! simulation length 20 ps
```

Note that no is integer while expressions are calculated in double. In conversion double to integer, the result is rounded down (truncated), and that is why +0.5 is added.

AllocSizeLim (0x40000000) Maximum size allocated at once (one array), default 0x40000000 = 1073741824 B = 1 GiB, maximum is 2147483647 (2 GiB − 1 B).

bj.eps (1e-6) BJERRUM version only: precision of Newton-Raphson minimization for finding the Bjerrum defects, in Å. There is pre-scan in a 2σ-sphere from the position of the defect read from the oo-file and with grid $\sigma/4$.

bj.mode (0) BJERRUM version only, sum of flags:

  1 Read the oo-file, generated by iceplus. If not given, the search of defects starts from $\vec{r}_D = (0, 0, 0)$ and $\vec{r}_L = (L_x/2, L_y/2, L_z/2)$. One line printed, example:

  ```
  bIcQ8-8-8.oo read: D-defect: D1=2593 D2=2594, L-defect: L1=770 L2=773
  !nD=0 nD1=0 nD2=0   nL=1 nL1=770 nL2=773 BJERRUM1
  ```

  where D1,D2 and L1,L2 are molecule numbers, nD = number of D-defects found, nD1,nD2=numbers of both molecules

  2 Scan several Gaussian σs (see below): For each σ, start from the oo-based positions of one D- and one L-defect and find the respective maximum and minimum of the charges smoothed by the σ-Gaussian. Lines printed are:

```
! sigma         q              x            y            z      #it
 3.363586 -0.187483 [ 19.291970  31.205150   3.244432] 166 BJERRUM2
 3.668016 -0.189113 [ 19.312199  31.215377   3.276372] 142 BJERRUM2
 4.000000 -0.190715 [ 19.333204  31.224358   3.309127] 170 BJERRUM2
```

columns: σ = in a loop from `bj.from` to $V^{1/3}$`*bj.to` with quotient `bj.q`, q = calculated charge, $(x, y, z) = r_{\text{Bj}}$ = calculated position, #it = number of iterations needed

4   `bj.mode&4=0` The Bjerrum charges are calculated for every configuration at the end of the `noint`-cycle, as usual.

`bj.mode&4=4` All configurations are everaged and the Bjerrum charges are calculated for the averaged configuration at the end of job (given by `;` in the data). The averaging takes into account the periodic boundary conditions; i.e., it is performed with the configuration scaled to box 1^3 and the averaged box is rescaled back to the averaged box sizes.

128 Search based on a configuration; DO NOT USE (broken/unreliable). (Was `bj.mode=1` in V3.3b and older.)

`bj.from` BJERRUM version only: The first Gaussian σ; see above for the loop.

`bj.to (0.25)` See above.

`bj.q (1.09050773266525766)` Gaussian σ's multiplied by q after step.

BJERRUM version requires `#define BJERRUM` in `simopt.h` and module `minimize` included in `metamake.mmk`.

The following function is minimized or maximized, to determine $\vec{r}_{\text{Bj}}$:

$$q_{\text{bj}} = \sum_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\vec{r}_i - \vec{r}_{\text{Bj}})^2}{2\sigma^2}\right] q_i \tag{9.1}$$

Experimantally, including more images of the basic cell have been tested, but it was commented out in thecode because too slow.

`box.max14 (0)` LINKCELL version only:
The linked-cell list algorithm searches for 1–2, 1–3, and 1–4 exceptions only in radius `box.max14` around sites. No check is made whether this distance is sufficient: if not, the affected 1–2, 1–3, 1–4 interactions will be calculated incorrectly! The default `box.max14=0` means that an automatic setup will be used instead, see variable `box.over14` below. However, `box.max14` should be set manually if numerical derivates are needed; e.g., for normal mode analysis.

`box.over14 (8)` LINKCELL version only:
The linked-cell list algorithm searches for 1–2, 1–3, and 1–4 exceptions only in a certain radius around sites. If `box.over14` is set, this radius is dynamically adjusted during the run: `box.over14` is the factor by which the estimate of the change (between MD steps) of the max 1–2, 1–3, 1–4 distance is multiplied. (It has more effects, see `lc.c` for details). The speed gain is small, though.
Negative `box.over14` selects the debug mode: also prints distances and speeds of their changes. `box.over14=3` is safe enough in most cases. If the algorithm runs close to its safety limits, a warning is printed, and `box.over14` is automatically increased. If this

happens too often (let us say more than twice during a moderately long run), use larger `box.over14`.

`box.over14=0` with unset `box.max14` will set `box.max14=cutoff` (backward compatibility).

WARNING: Not suitable for normal mode analysis.

(Called only `over14` in V2.7g and older.

box.rmin (0) For LINKCELL only: all coordinates are normalized to be $>$ `box.rmin`. (Without LINKCELL always > 0). This is to prevent unnormalized configurations with Gear or box change, or with "lone dependants", or when numerical derivatives are calculated (incl. the normal mode analysis). The default is `box.rmin` $= 0 =$ automatic setup (recommended, but working only during standard simulation). `box.rmin` < 0 forces `box.rmin` $= 0$ even with Gear or box change (not recommended).

bulkmodulus (0) An estimate of bulk modulus (inverse compressibility) in Pa. It is used in the Berendsen-like NPT simulation (see `tau.P`), van der Waals radius fitting and similar. If not set, an ideal gas approximation is adopted. Hint: for water `bulkmodulus=2.2e9`.

No bulk modulus applies for the MTK-style barostat, `thermostat="NPT"`.

If used for particle size adjustment (see `tau.sig`), and if the σ-like parameter has a meaning of inverse size (as in the Buckingham or Busing force fields), `bulkmodulus` must be negative. For Lennard-Jones `bulkmodulus` should be positive.

cache (64) This option rearranges the pair (r-space) sums. Void with the LINKCELL version.

> **cache=0** Recursive triangulation (fractal) algorithm, efficient for large systems of polyatomic molecules (with long cutoff, otherwise LINKCELL may work better).
>
> **cache=1** The simplest pair forces in 2 nested loops, good for small systems.
>
> **cache>1** The pair (r-space) sum is rearranged into pair sum of blocks over `cache`$\times$`cache` molecules. In case of a simulation with a lot of small molecules, the program runs faster because the number of cache-memory transfers is decreased. The typical speed gain is 20% on x86 for 1000 water molecules.

center.K (0) FREEBC only: removed in version 2.9f (see below)

center.r0 (0) FREEBC only: removed in version 2.9f (see below)

center.K[*] (0) Harmonic (and optionally shifted) central force constants, separately in x, y, z coordinates. The force is to the center of the box (in periodic b.c.) or to $(0, 0, 0)$ (if FREEBC).
BUG: In FREEBC, the number of degrees of freedom as well as drift corrections are not set automatically. Use variables `conserved` and `drift`!

center.r0[*] (0) Periodic b.c.: offsets from the center of the box, separately in x, y, z coordinates. The potential is:

$$u = \sum_{w=x,y,z} h_w(w), \text{ where } h_w(w) = K_w \begin{cases} (w + \texttt{r0}_w)^2, & \text{for } w < -\texttt{r0}_w \\ 0, & \text{for } |w| < \texttt{r0}_w \\ (w - \texttt{r0}_w)^2, & \text{for } w > \texttt{r0}_w \end{cases} \quad (9.2)$$

(x, y, z) is the distance from the respective center, $\vec{K} =$ `center.K[]` is in the internal program units; i.e., in K.

If `center.r0=0`, this is the harmonic oscillator: A free molecule will exhibit harmonic vibrations with period

$$\tau = C \times \sqrt{\frac{m}{nK_w}}, \tag{9.3}$$

where m is the mass in g/mol, n is the number of sites, and $C = 2\pi \times 0.776 \text{ ps} = 4.88 \text{ ps}$.

Example: for a water molecule and $K_w =$ `center.K = 1` we get from $m = 18$ g/mol and $n = 3$ that the period is approximately 12 ps.

These forces are intended with nonzero `center.r0` to keep molecules in the selected area; outside this area, accelerations acting to molecules of different masses differ and motion is affected by a torque. Consider also `center.cmK` below.

WARNING: the potential is not spherically symmetric unless all `center.r0` are zero and all `center.K` are the same. Set variables `conserved` and `drift` manually!

`center.cmK[*]` `(0)` The same harmonic force (pointing to $r_c = (0, 0, 0)$ in case of FREEBC, or to $r_c = (L_x/2, L_y/2, L_z/2) =$ center of the box in the periodic b.c.) is added to **all** particles. The potential of the force is

$$\sum_{i=x,y,z} K_i (r_{\text{CM},i} - r_{c,i})^2 \tag{9.4}$$

where CM is the center-of-mass. This force causes the whole configuration (typically the slab for $K_z =$ `center.cmK[2]` > 0) to move to the center of the box, but there is **no force to keep the slab together**. The force applies to the first `center.cmn` molecules only (excl. those fixed by option `-j`). Centering and removing velocity drift should be turned off for the selected coordinates (see `drift`); e.g., `drift&4` and `drift&32` should be zero if `center.cmK[2]>0`). The typical usage is to keep a configuration (slab) in place if other forces (fixed atoms) are present. A suitable force constant `cmK` may be from 100 to 10000. The correlation time in ps is $\tau_i = \sqrt{m/2/K_i} = 0.7755\sqrt{M/K_i}$), the period is $2\pi\tau$. (The mass is in the program units; $m = 0.831446\,M$, where M is the total mass of **all molecules** in g/mol.)

`center.cmn` `(0)` Number of molecules to which the central harmonic force (see `center.cmK`) is added. A number greater than the number of molecules N is equivalent to N.

`cl.format` `(1)` The same as keyword `format` in file <u>simname</u>`.cli` (see Sect. 14.8.3).

`cl.maxn` `(20)` The same as keyword `maxn` file <u>simname</u>`.cli` (see Sect. 14.8.3).

`cl.maxcluster` `(12)` The same as keyword `maxcluster` in file <u>simname</u>`.cli` (see Sect. 14.8.3).

`cl.mode` `(0)` `cl.mode` is a sum of bits:

  `cl.mode=1` Turn cluster calculations on.

  `cl.mode=2` The same as keyword `clusters` in file <u>simname</u>`.cli` (see Sect. 14.8.3).

  `cl.mode=4` The same as keyword `configurations` in file <u>simname</u>`.cli` (see Sect. 14.8.3)

  `cl.mode=8` The same as keyword `bonddynamics` in file <u>simname</u>`.cli` (see Sect. 14.8.3)

NOTE: All `cl.*` variables require #define CLUSTERS at compile time.

`conserved (-1)` The number of conserved degrees of freedom (e.g., momenta and angular momenta or additional constraints with ANCHOR, but not energy). If `conserved=-1` (default), it is determined from `drift`. WARNING: the automaic setup does not work with ANCHOR. See also `drift`.

`corr (3+16; with PERSUM 3+16+32)` Selects which corrections will be included by summing up the following flags. For slab corrections, see `slab.K` and `slab.range` instead. Do not confuse with `drift`.

1   Homogeneous cutoff corrections (of Lennard-Jones and similar interactions) are included in the final results for pressure, energy, enthalpy, etc.; for the exception, see `corr=2` below. Some quantities (components of the pressure tensor, quantities requested in a cpi-file) are still recorded without these corrections.
   If this bit is 0, the cutoff corrections are not included; nevertheless, they are calculated and reported (so that they can be added manually).

2   Homogeneous cutoff corrections are included in the pressure used by barostat. Turning off this feature is useful for maintaining a selected pressure component as `Ptzz` $= P_{zz}$.

4   Finite-size correction $N_{f0}/N_f\times$ for the kinetic part of the pressure, see (15.15).[1] This applies to the atom-based pressure (tensor) and atom-based virtual volume change calculation. The value can be combined with 1 or 2. Cannot be combined with `tau.CM`.
   This correction should be used for NVE and momentum-conserving thermostats (Nosé–Hoover, Bussi, Berendsen). It should not be used with particle-based stochastic thermostats (Andersen, Langevin). It is not suitable for pressure components in the slab geometry (e.g., vapor pressure from P_{zz}).
   In the NPT ensemble by Martyna-Tobias-Klein (`thermostat="NPT"`), N_{f0} is already modified so that this term is correct and `corr&4` should not be used (this is correctly set with `corr&16` – see below). This may be broken while switching thermostats – be careful and check N_{f0} (`No.f4P` in the prt-file) vs. N_f (`No.f`).

8   As above, except that the degrees of freedom are molecule-based (w.r.t. centers of mass). Suitable for pressure calculated via molecule-based rescaling.

16 The kinetic pressure corrections are set automatically according to the thermostat and barostat in use.
   WARNING: This automatic setting fails if the thermostat/barostat is changed within one run (after `;` in the data); then, `corr` should be set manually. See also `corr&4` above.

32 In calculating the cutoff correction and RDF, the number of pairs of equal atoms is set to $N^2/2$ instead of $N(N-1)/2$. This guarantees identical results when the configuration is periodically repeated (useful with PERSUM), but violates the strict NVT definition of RDF.

64 In the calculation of the number of the degrees of freedom with NVE (no-thermostat) and Berendsen thermostat, 1 is subtracted to account for energy conservation. This

---

[1]Since V2.6m this had been the default without a possibility to turn this correction off. Since V2.7a the correction is set by `corr&4` with a smart default.

was the default for V3.1m and older. See Table 12.1 for further explanation and discussion.

128 Suppress warning "The kinetic pressure correction seems to be inconsistent...".

For the cutoff, See Sect. 11.4.

NOTES: `corr=4,8` are new in V2.6m. These corrections are small for bulk liquids and solids but important for gases. Some finite-size effects $\approx C/N$ are still present, but these corrections decrease substantially the constant C.

`CPnbit (0)` If `5<CPnbit<25` is given, packed file `.cpz` will be created for the convergence profile instead of the normal `.cp` one. `CPnbit` is the number of bits for coding the min-max range of each recorded variable (i.e., there are `2^CPnbit` levels). Use utility `cppak` to convert it to `.cp` (and vice versa). Recommended values are from 12 to 20, with the file sizes shrunk to 30%–50%. Utility `showcp` understands the cpz-files as well so that the analysis of convergence profiles is transparent.

`cutoff (0)` Real-space cutoff in Å. Unless PERSUM is #defined in `simopt.h`, `cutoff` must not be shorter than `LJcutoff` and longer than half the box size (for the LINKCELL version this limit is weaker). The default of 0 means the automatic setup:

- With the Ewald summation, `cutoff` will be selected to half the final (reference) box for small systems (< 1000 charges), otherwise to $\mathrm{const} \times N^{1/6}$.
- With cutoff electrostatics, it is set to min(half the box,12 Å).
- With FREEBC and NIBC, it is set to 9e9 ("infinity")

Note that `cutoff` also defines the range for calculating the radial distribution functions (this does apply also for the FREEBC version) unless modified by `rdf.cutoff`.

`MSD.mode` (0)] Normally in the reread mode (options `-m0 or -m1`). See also `plb2diff`. Sum of flags:

`MSD.mode=1` Calculate mean square displacements (over species) and diffusivities.

`MSD.mode=2` Calculate square charge displacements and the conductivity.

(Called `diff.mode` in V3.7a and older.)

`dih.grid (0)` Grid for measuring the dihedral angle distributions. See also `#define` DIHHIST in `simopt.h`. The full angle of 360 deg is divided into `dih.grid` subintervals. `dih.grid=0` turns off measuring. The results are stored in binary file with extension `.dih`; at the same time, ascii dump `.dia` is printed.

`dih.res (0)` REMOVED in V3.4e.

`dih.cp (-1)`   `-1` Average of all gauche or cis counts recorded in the convergence profile

`>0` Dihedral # dih.cp is recorded. (For the numbers, see the output .prt file) NOTE: from V2.0j, dihedrals (and dih.cp) are numbered from 1

`dih.dcp (0)` Must be set (`dih.dcp=1`) to enable dihedral dump. See file extensions `.dcp` and `.ddh` for details. DIHHIST=-1 required, species=0 only.

`drift (4096)` This variable affects the way how the center of mass, linear momentum, and angular momentum are corrected for numerical errors. In addition, the number of conserved degrees of freedom is derived from `drift` as the default (with `conserved=-1`). It is a sum of (NB: leading 0 denotes octal number):

  `drift=1=01` Place the x-coordinate of the center of mass to center (FREEBC) or box center (periodic b.c.).

  `drift=2=02` Place the y-coordinate of the center of mass to center (FREEBC) or box center (periodic b.c.).

  `drift=4=04` Place the z-coordinate of the center of mass to center (FREEBC) or box center (periodic b.c.).

  `drift=8=010` Set the x-component of the linear momentum (velocity drift) to zero.

  `drift=16=020` Set the y-component of the linear momentum (velocity drift) to zero.

  `drift=32=040` Set the z-component of the linear momentum (velocity drift) to zero.

  `drift=64=0100` Set the x-component of the angular momentum (rotation) to zero.

  `drift=128=0200` Set the y-component of the angular momentum (rotation) to zero.

  `drift=256=0400` Set the z-component of the angular momentum (rotation) to zero.

  `(drift&1536)=0 (octal: (drift&03000)=0)` Corrections will be performed at start (init/load) and every cycle (default).

  `(drift&1536)=512 (octal: (drift&03000)=01000)` Corrections will be performed at start (init/load) only.

  `(drift&1536)=1024 (octal: (drift&03000)=02000)` Corrections will be performed at start (init/load) and every step.

  `(drift&1536)=1536 (octal: (drift&03000)=03000)` Corrections will be performed at start (init/load) and before saving.

  `drift=2048=04000` For molecules with 'dependants' (like site M in TIP4P water), the dependant positions are recalculated after every cycle (of `noint` steps). The default is that the dependants are calculated when necessary only (=before force calculation and saving cfg- or plb-files).

  `drift=4096=010000` Requests automatic determination of all components 1..256. This is the default. However, it may fail at certain combinations (like configuration anchored to a few atoms, combination of anchoring with walls, etc.)

`drmax (0.2)` Maximum displacement allowed in one integration step, in Å.
  Positive: applies with `init>=3` (initialization) and thermostat only, irrelevant for `init<3` and constant energy/enthalpy (no thermostat).
  Negative: `|drmax|` applies unconditionally (may be useful with `init<0` which ignores `drmax>0`; `drmax` should be then set to 0 after a sweep).
  Useful while initializing when forces are large and would cause the integrator to crash. Equilibration with this option turned off (`drmax=0` or `|drmax|` large enough) should follow. **Should not be used in productive runs!** For compounds containing hydrogen at ambient temperatures, `drmax=0.125` works fine, `drmax=0.2` OK with longer timesteps. If `drmax` is used, `Emax` can be larger (about 10000, but usually not more than 100000), which makes the initializer faster.
  (V3.0e: thermostat checks added)

`dt.cfg (0)` Frequency of writing configurations to files simname.1, simname.2, ..., in ps. The default `dt.cfg=0` switches off the function, so does `-r0`.

Option `-r` determines the number of derivatives written. The default is `-r1`, i.e., only positions (not velocities) are dumped; use `-r2` to dump positions and velocities (NOTE: with Verlet, $r(t) - r(t - h)$ are written, with Gear, $h\dot{r}(t)$). The dump for polarizable models includes positions of the Drude charges.

During a simulation (option `-m2` or higher), `dt.cfg` should be an integer multiple of `h*noint`. If started with `init=2`, simname.1 is saved at time `t=dt.cfg`; there is no simname.0 (unlike the first frame in simname.plb which is for $t = 0$).

New in V3.6k: During the re-read mode with `-m1` (*i.e.*, a plb-file is read instead of simulating), `dt.cfg` should be an integer multiple of `dt.plb*reread.by`. Files simname.# with # = `t/dt.cfg` (rounded if needed) are dumped. Note that frame 1 (`reread.from=1`) in a plb-file corresponds to `t=0` which is how it is typically written during simulation; hence, simname.0 is dumped (unlike during simulation where dumping starts with `t=dt.cfg`). The numbering of the remaining simname.# is the same is if obtained during a simulation with the same `dt.cfg`. Option `-r1` should be used because velocities are not available. With polarizable models, use also option `-f` and set `scf.omegax` and `scf.epsx` to the required accuracy of SCF; the positions of Drude particles are then written to the cfg-files dumped. Example:

```
dt.plb=1 dt.cfg=1
reread.from=2 reread.to=11 reread.by=1
scf.omegax=0.88
scf.epsx=0.01
! run by: cookgcp0slc -m1 -f -r1 -w0 forcefield simname source.plb
```

will create files numbered simname.1 .. simname.10, corresponding to times 1 .. 10 ps.

See utility `cfg2asc` to get the ASCII image of the saved files.

`dt.plb (0)` How often to write playback, in ps. See also option `-y` (the default is to record the whole configuration).

> `dt.plb>0` : Writes once every `dt.plb` ps. `dt.plb` should be an integer multiple of the timestep `h`, otherwise the configurations will not be written in regular time intervals.

> `dt.plb=0` : Writes once after every sweep (data set ended by ;) finishes or is interrupted.

In both cases the initial configuration (at t=0) is included (even if t=0 is a consequence of `init=2`).

CAVEAT: `dt.plb` for post-processing (as diffusivity and conductivity calculations), should be written in the def (not get) file.

`dt.prt (0.001)` Frequency of printing protocol (in ps). `dt.prt=0` switches off printing, `dt.prt=`tiny value causes every cycle to be printed.

`dV (0)` Fool-proof independent pressure calculation by virtual volume change, also for area-change for surface tension 15.8. For debugging and checking purposes—normally the virial of force as calculated in `cook` from all pair forces is more accurate. Uses the formula:

$$P_{\mathrm{dV}} = k_{\mathrm{B}} T \mathcal{N} - \langle \mathrm{d}U/\mathrm{d}V \rangle \tag{9.5}$$

where the derivative in the ensemble average is calculated numerically by scaling the volume exp(dV) and exp(-dV) times and taking the difference.

Turned off by dV=0 (default).

With `rescale&8` set ("CM" in the mnemonics, e.g. `rescale="xyzCM"`), a molecule-based calculation is selected: Positions of centers of mass of the molecules are scaled by a factor given by the cube root of the volume ratio. The shape and size of the molecules are unchanged. In the above formula, $\mathcal{N} = N/V$ is the number density and $k_\mathrm{B}T\mathcal{N}$ becomes $2E_{\mathrm{kin-inter}}/3V$, where $E_{\mathrm{kin-inter}}$ is the translational kinetic energy of whole molecules (of their centers of mass). Can be used also with constraint dynamics, but fails if the molecules are of sizes larger or comparable with the box size/2.

If `rescale&8 = 0`, atom-based calculations are selected. Positions of all sites are scaled by a factor given by the cube root of the volume ratio. In the above formula, ρ is the atom number density and $kT\rho$ becomes $2E_{\mathrm{kin}}/3V$. Should not be used with constraint dynamics, i.e., `cook -u99999` should be used.

Reasonable `dV` is around 0.001, according to the accuracy of calculations, which means that Ewald needs to decrease `el.epsr, el.epsk`. The pressure, called `PdVa` (atom-based) or `PdVm` (molecule-based), is recorded in column 4 in the convergence profile. Typical numerical errors for `dV=0.001` are tens of kPa for charged systems or units of kPa for argon. Note also the systematic difference between virial-based and pressure-tensor-based values for Ewald, which are around MPa for the default setup.

For Ewald summation, the atom-based pressure `PdVa` is the same as the true virial pressure with small numeric deviation caused by finiteness of `dV` and sampling at different time. The ensemble average of the molecule-based pressure `PdVm` is also the same (irrespective of the system size).

For cutoff electrostatic, see Sect. 15.3, this is no longer true because of the truncate-and-shift errors. However, both pressures are the same in the thermodynamic limit and large `cutoff`.

For POLAR, see `scf.epsx` determining the accuracy of the self-field. `dV` should be relatively large, perhaps up to `dV=0.01`, even though this may lead to systematic pressure error. Note that the pressure error is proportional to dV^2 because a second order difference method is used to calculate the energy over volume derivative.

For POLAR, the accurate (=iterated to precision `scf.epsx`) induced dipoles are compared to those generated by the integrator (predictor-corrector). However, to spare, the reference induced dipoles are calculated as average of the values for volumes `V+dV` and `V-dV` so that `dV` must be small enough.

Other boundary conditions (NIBC) have not been tested.

If rescaling by only some coordinates is set (e.g., `rescale="x"`), the box is scaled in selected coordinates only and the corresponding diagonal component of the pressure tensor is printed. However, the kinetic part is based on temperature (not on the respective kinetic part of the pressure tensor) so that the virtual volume pressure tensor component agrees with the virial-based only on average. Similarly, the virial pressure (or component) agrees with the molecular-scaling result only on average.

NOTE: the calculated pressure is written to column 4 of the cp-file – `tau.rho` should be unset.

`E (0)` Total (potential+kinetic) energy, i.e., the value of the Hamiltonian, to be kept constant, in Kelvin. If the Nosé–Hoover ensemble is used, then the extended Hamiltonian. If the isobaric ensemble is used, then the enthalpy. See also `tau.E`

`Eelst[*] (0,0,0)` Removed in V 3.4f, see `el.E[]`

`el.alpha` Ewald (and Fennell–Gezelter): The separation parameter, in 1/Å. See also `el.test`. It is set automatically if `el.test=-10`, otherwise `el.alpha=0.2`.

FREEBC: irrelevant.

Cutoff electrostatics: the cutoff is smooth in the interval `[alpha*cutoff,cutoff]`, `alpha=0.7` (the default) is recommended. (NOTE: larger values (> 0.8) cause artifact angular correlations for small molecules as water).

`el.B[*] (0,0,0)` Static external magnetic field in T. Not for POLAR. Magnetic force affects moving point charges vial the Lorentz force. Moving charges are not assumed to be a source of magnetic field. Implemented for Verlet/SHAKE only using velocities predicted by TRVP. Cf. `el.m.*`. For FREEBC, `drift` must be set manually.

`el.bg (0 or 1)` Must appear in the def-file. What to do in case of charged system (cf. `el.epsq`):

  `el.bg=0` Charged system is error. This is the default for COULOMB.

  `el.bg=1` Calculation proceeds with the charged system. If Ewald summation applies, the simplest energy term for "charged background" is added. This is the default for FREEBC and NIBC. SLAB: the background charge is not icluded in charge z-profiles.

  `el.bg=2` Make the system neutral by adding the same charge to all charged sites. OLD: Available by option `-q-#` prior V3.5i. ??? POLAR: "Drude charges are not changed since V3.1c" – double checked, the Drude charges are changed, too.

`el.centroid (0)` This option make sense only for ionic systems (charged molecules). If set to 1, the dipole moments of ions are calculated with respect to charge centroid (center of squared partial charges). The total dipole moment of the simulation cell then is not a sum of the partial dipole moments. (The total dipole moment calculate during the Ewald summation is anyway.) If set to 0 (default), no transformation is made; the dipole moments of individual molecules do not make sense. (To be changed in future.)

`el.corr (0)` Extended slab dipolar correction by Yeh–Berkowitz [9] or by Ballenegger–Arnold–Cerdà [10]. Sum of flags:

  `1="x"` Correct the x-component of forces and energy. Probably not useful.

  `2="y"` Correct the y-component of forces and energy. Probably not useful.

  `4="z"` Correct the z-component of forces and energy. Good for the z-slab geometry used by (the SLAB version of) `cook`. This is the slab correction by [9]: In the z-slab geometry, the interaction of the z-components of the slab dipole moments is compensated. For systems with no free charges, this is (almost) equivalent to a system periodic in x, y. `el.corr="z"` is recommended for slabs of dipolar fluids. It

cannot be used for ionic systems because of a jump in forces.
IMPORTANT: See also variable `corr`, see Sect. 15.9.

8 Corrected Measurement only. The above corection is not included in forces (the trajectory is the same as with `el.corr=0`) but affects the electrostatic energy, pressure, and the selected diagonal components of the pressure tensor (cf. Sec. 15.9). Energy conservation is thus violated. Useful, e.g., for determining vapor pressure of ionic liquids. Typically, `el.corr=12="zCM"` adds the correction to P_{zz} and the electrostatic energy.

NOTE: `el.corr=7="xyz"` is equivalent to `el.epsinf=0`, which can be interpreted as the zero electric displacement (because $\vec{D} = \epsilon' \vec{E} = 0$).

Former `el.slab` or negative `el.epsinf` was equivalent to `el.corr=4`

`el.diag (0)` (to be reconsidered, cf. `el.rshift&4`)

`el.diag=1` Calculate and include to the electrostatic energy the diagonal k-space correction [11]. Useful for molten salts.

`el.diag=0` Calculate, but do not include the diagonal correction for Ewald electrostatic energy [11].

`el.diag=-1` Do not calculate the diagonal correction at all.

`el.diff (0.05)` Relative error of box size change (any coordinate) before a warning "*** box has changed by >el.diff (log-scale) and k-vectors remain unchanged" is printed. Note that the number of k-vectors is not adjusted with the box because it would lead to a jump in energy.

`el.E[*] (0,0,0)` Called `Eelst[*]` before V3.4g.

External electrostatic field $\vec{E} = (E_x, E_y, E_z) \equiv (E_0, E_1, E_2)$, in V/m. It is recommended to use only the z-part, e.g., $E_z = $ `el.E[2]` $= $ `3e7`. See `el.f` and `el.phase[]` for time-dependent field. For FREEBC, `drift` must be set manually.

For dipolar systems:
If `el.E[`i`]` is set, $i = 0, 1, 2$, the component `M[i]` instead of total dipole moment `|M|` is reported in the convergence profile. Note that either `lag.M` or `lag.J` must be set to turn this calculation on. (There is an independent calculation of the dipole moment with the Ewald summation. This inefficency may be removed in future.)

MACSIMUS reports `Mz` (or `M[2]`) in "program dipole units" (1 p.u. $= 0.0117501$ D $= 3.9194115e-32$ C m). The dielectric constant (relative permittivity) is then computed from averaged $M_z = $ `Mz`, see Appendix 29, eq. (29.7).

For ionic systems:
Can be used for NEMD determination of conductivity. To measure the electric current density, you must specify `lag.J` (e.g., `lag.J=lag.err`). Thermostat (e.g., `thermostat="Berendsen"` must apply. MACSIMUS reports the current density in A/m^2 so that the conductivity in S/m is simply $\kappa = J_z/E_z$.

`el.ecc (0)` with `#define ECC`, EXPERIMENTAL, See Sect. 33. Turns on the Electronic Continuum Correction (ECC) code aka Molecular Dynamic with Electronic Continuum (MDEC).

0   Standard no-ECC version (default).

1   Ion solvation model. The ble-file contains real (unscaled) charges, these charges are then multiplied by scaling factor $\epsilon_f^{-1/2}$ (see `el.epsf` below).

2   Dipole solvation model. The ble-file contains real (unscaled) charges, these charges are then multiplied by scaling factor $\epsilon_f^{-1/2}(2 + \epsilon_f)/3$.

-1 Ion solvation model. The ble-file contains already scaled charges. Useful for pretending that a model has ECC-like scaled charges, for using `el.epsf` calculated by another way than from the Clausius–Mossotti (CM) formula, etc.
    Note that in some calculations, the original charges are reconstructed (using the CM formula) and scaled again by a different factor (in V3.4d, only virtual volume change determination of pressure is implemented in this way).

-2 Dipole solvation model. The ble-file contains already scaled charges.

Must be present in the def-file. NOTE: replaces `el.fast` of V3.3h and older

`el.epsf (0)` #define ECC needed, EXPERIMENTAL, with `el.ecc`. The dielectric constant of the electronic continuum (not simulated optical or fast modes of the material dielectric constant), ϵ_f. If not given (i.e., `el.epsf=0`), it is calculated from the polarizabilities present in the ble-file in the column (originally) intended for a combining rule using the Clausius–Mossotti formula:

$$\epsilon_f = \frac{1 + 2a}{1 - a}, \quad a = \frac{4\pi}{3V} \sum_i \alpha_i \tag{9.6}$$

where α_i are the polarizability volumes. Must be present in the def-file.

WARNINGs:

- `el.epsf` is calculated once and does not change if volume changes (e.g., if density is requested to change by `tau.rho`). If you wish to change it after volume change, use `el.epsf=0` and equilibrate.

- No support for NPT (`el.epsf` remains constant)!.

- Final pressure must be calculated from individual contributions printed, see script `Pk.sh` and file `notes.pdf`.

`el.epsinf (3e33 for Ewald, 1 for cutoff electrostatics)` The relative dielectric constant ϵ_r' of surrounding continuum for the Ewald summation.
WARNING: since V2.7f, `el.epsinf=-1` does not mean the dipole slab correction (cf. `el.corr`).

`el.epsk (0.5)` Ewald only: Accuracy of k-space Ewald sums, in K/Å. `el.epsk` is the maximum allowed k-space cutoff error in force on one charge. See Sect. 11.2. (Prior 2.4e called only `epsk`)

`el.epsq (0)` Must appear in the def-file. Detection limit for the total charge expressed as a relative error. If

$$\sqrt{\frac{\left(\sum_i q_i\right)^2}{\sum_i q_i^2}} > \texttt{el.epsq}$$

then further actions are taken according to the value of `el.bg` (0=error, 1=background, 2=redistribute to make neutral). The default translates into a reasonable heuristic threshold (different for POLAR and nonpolar).

`el.epsr (0.05)` Ewald summation: Accuracy of r-space Ewald sums (see Sect. 11.2) [? or approximation of $1/r$ in cutoff electrostatics (see Sect. 15.3)], in K/Å. `el.epsr` is the maximum allowed cutoff error in force on one charge. (Prior 2.4e called only `epsr`)

`el.f` Frequency of the external electric field (`el.E`) in Hz. See also `el.phase` and `el.E`.

`el.grid` Periodic b.c. only.
Grid to calculate functions (splines) needed for real-space Ewald sums (also for deprecated `COULOMB=2`). The grid means that independent variable $x = (\alpha r)^2$ is split into spline intervals so that there are `el.grid` subintervals per $\Delta x = 1$. Typically, the maximum x corresponding to the cutoff is around 3.5–4, but the tables are a little bit extended.

The default `el.grid` depends on the splines used:

`SPLINE=3` `el.grid=32`. In cook V3.6w and older, finer grids (larger `el.grid`) were numerically imprecise due to numerically unstable procedure used in the calculation of the spline coefficients. This was fixed in V3.7a.; however, finer grids are usually not needed anyway.

`SPLINE=±2`
For COULOMB=-2 (recommended, intramolecular charge-charge terms calculated by the library function): `el.grid=256`.
For COULOMB=-1 (intramolecular charge-charge interactions calculated by splines – not recommended): `el.grid=512`. For water, the error in pressure caused by intramolecular Ewald r-space terms calculated via splines is 2e13/el.grid^3 Pa. Similarly, the surface tension error is about 200000/el.grid^3 N/m.

`el.kappa (0.2 [Ewald], 0 [Cutoff electrostatics])` The reciprocal space cutoff parameter in $[1/\text{Å}]$: maximum integer k-vectors are `Kx=el.kappa*L[0]`, `Ky=el.kappa*L[1]`, `Kz=el.kappa*L[2]` in the respective directions. More precisely, the absolute value of vector $(K_x/L_x, K_y/L_y, K_z/L_z)$, $\vec{K} \in Z^3$, is bound by `el.kappa`. Note that these k-vectors mean the number of waves in a unit of length (1 Å). The k-vectors used in the structure factor calculations are multiplied by 2π. See also `el.test`. For the Cutoff electrostatics version, `el.kappa=0` should be used.
HISTORY: in old cubic-box versions, dimensionless parameter K=el.kappa*L was used. In non-rectangular-box versions prior 2.4e, this K meant Kz and Kx,Ky were scaled accordingly.

`el.m.*` Simulation of molecules with a classical magnetic moment. The magnetic moment is represented by two opposite magnetic monopoles. It interacts with the external magnetic field `el.B` only. Magnetic dipole moments of different molecules do not interact mutually nor with moving charges Not implemented for POLAR. Cf. `el.B`. **BUG:** Virial of forces not calculated.

Note that the energy of two Bohr magnetons 1 Å apart divided by k_B is in interval $[-0.62, 1.25]$ K, and negligible if angle averaged in a fluid (as Keesom). For comparison, one Bohr magneton in a field of 1 T has energy 0.67 K (but this is not averaged to zero). The code includes also the interaction of magnetic field with point charges. A water molecule at ambient temperature can be roughly approximated by a loop of radius $r = 1$ Å and an elementary charge with speed $v = 1$ km/s. This generates a magnetic moment of $\mu = IS = \frac{ve}{2\pi r}\pi r^2 = ver/2 = 0.00086\mu_\text{B}$. This small and likely negligible, too.

`el.m.sp (0)`

> `el.m.sp`≥ 0 Species containing the magnetic moment.
>
> `el.m.sp`< 0 All species $|$`el.m.sp`$|$, $|$`el.m.sp`$|+1$ etc. contain the magnetic moment.

`el.m.plus (0)` Site number of the above species where the positive magnetic monopole is placed.

`el.m.minus (0)` Site number where the negative magnetic monopole is placed. It is recommended that bond `el.m.plus`–`el.m.minus` is constrained.
BUG: Validity of `el.m.plus` and `el.m.minus` is not tested.

`el.m.m (0)` Magnetic moment in $A\,m^2 = J\,T^{-1}$. Note that this is a large unit, use multiples of the Bohr magneton $\mu_B = 9.2740100783(28) \times 10^{-24}\,A\,m^2$.

`el.minqq (1)` Minimum charge-charge distance, in Å. This value serves for two purposes:

1. For point charges: The error of splines for the r-space part of the Ewald sums is reported for this distance, see Sect. 11.6. (GAUSSIANCHARGES: always from zero)

2. With POLAR: ERROR is reported if the distance of the Drude particle from the central atom exceeds `el.minqq`.

(Called `minqq` prior version 2.4e)

`el.Perr (1e6 for Ewald, 1e7 otherwise)` Maximum difference between pressure calculatd from the virial theorem (elst. energy $= -$virial) and trace of the pressure tensor before a warning is printed. For Ewald of point charges, this indicates the Ewald precision. Does not apply for Gaussian charges (the virial theorem does not work here).

`el.phase[*] (0,0,0)` Phase for time-dependent external electric field, vector:

$$E_i(t) = \texttt{el.E[i]}\cos(2\pi(\texttt{el.f} * \texttt{t} - \texttt{el.phase[i]})) \qquad (9.7)$$

where $i = x, y, z$. Example:

```
el.f=2.45e9 ! [Hz]
el.E[0]=1e8 ! [V/m]
el.phase[0]=0 ! Ex=cos(2*pi*el.f*t)
el.E[1]=1e8 ! [V/m]
el.phase[1]=0.25 ! Ey=sin(2*pi*el.f*t)
```

In turn, this field rotates around the $\hat{z}$-axis.

`el.rplus (2.5)` Additional range to erfc splines (former ERFCPLUS): the real range usable without index out of range is `cutoff+el.rplus`. Needed for optimized water models, does not hurt if unnecessarily large. Not needed for Drude charges (POLAR version) because the cutoff test is based on the distance of central atoms.

`el.rshift (3)` Sum of flags:

1   The r-space spline [of eru$(r) =$ erfc$(r)/r$ for point charges] for the r-space electrostatic energy is shifted to avoid the jump at the cutoff.

2   As above for the forces

    4 Hammonds–Heyes correction added [12], see Sect. 11.7. It means that $\alpha' = \alpha \operatorname{erf}(\pi\kappa/\alpha)$ will be used for the r-space calculations in the Ewald summation and the self-term, to account in some averaged way for the missing k-space terms.

    The performance of this approach is not clear. Cf. `el.diag`. In V3.6s and older, this meant detailed test (as `-v4`).

`el.rshift=3` is recommended at very low temperatures and especially for normal mode calculations.

`el.sat (0)` The desired saturation of dipole moment of the cell. Void if `tau.sat=0`. See Sect. 29.3.

`el.sf (0)` Applies for `cook* -m0,-m1` (re-read mode).

    `el.sf=1` : Calculate a sphericalized structure factor (for a cubic box box only)

    `el.sf=3` : Calculate the full 3D structure factor.

`el.test (Ewald: -10, cutoff electrostatics: 0)` Unless `el.test=0` or `el.test=-10`, the control switches to a module that sets and tests the electrostatic forces accuracy and Ewald parameters. See Sect. 11.2.

Ewald only: The default value `el.test=-10` means automatic selection of `el.alpha` and `el.kappa` based on default accuracies `el.epsk` and `el.epsr`.

BUG: Out of order for polarizable molecules with no or small charges (yet it makes sense to calculate, e.g., in elst. field).

`el.test=0` means no automatic selection (the values of `el.alpha` and `el.kappa` remain unchanged).

`Emax (10000)` The energy limit for inserting one molecule when the configuration is initialized (`init>=3`), in Kelvin. See also `drmax`.

`eps (1e-6)` Accuracy for calculating the Lagrange multipliers, See Sect. 11.3. With option `-l`, the step for numerical gradient. (the default was 1e-5 prior V 2.0d)

`epsc (1e-7)` Accuracy of correcting constraints, See Sect. 11.3. At least 1e-9 (better 1e-10) needed for adiabatic simulation! The default is changed to `epsc=0.05` for conjugate gradient method of correcting constraints (unset bit 1 of `-c`).
Special: For `|SHAKE|=1`, `epsc>=1` determines a constant number of iterations; `omegac` must be set by hand.

`epsp,epspx,epspq` Changed into `scf.eps`, `scf.epsx`, `scf.epsq` – see there.

`equalize.cfg (0)` Redistribute masses of atoms globally for species given by `equalize.sp`. The molecular masses of species may change because the masses may be redistributed across species; use `-v7` for a verbose list of all masses. Should be used in the def-file only and is applied only once. Does not apply to dependants. Normally should be in interval $[0, 1]$. If both `equalize.cfg` and `equalize.mol` are given, then molecule-based equalization (`equalize.mol`) is applied first. It is not allowed if the number of molecules changes from that given in the def-file (as may happen for `init="crystal"` or loading a cfg-file with `load.N` allowing changing numbers of molecules: use `equalize.mol` or put the correct numbers of molecules.

> `equalize.cfg=0` Means preserving the original masses (the default).
>
> `equalize.cfg=1` Means that all atoms in the selected species will have the same (averaged) mass.

See below for the formula.

`equalize.mol (0)` Redistribute masses of atoms by molecules for all molecules. The molecular masses of species do not change. This is equivalent to `equalize` in cook prior V3.3k. Should be used in the def-file only and is applied only once. Does not apply to dependants. Normally should be in interval $(0, 1)$.

> `equalize.mol=0` Means preserving the original masses (the default).
>
> `equalize.mol=1` Means that all atoms in individual molecules will have equal masses; however, these masses may differ for different species.

The formula:

$$m_i := \frac{1}{\texttt{ns}} \sum_{j=1}^{\texttt{ns}} m_j * \texttt{equalize.mol} + m_i * (1 - \texttt{equalize.mol}) \qquad (9.8)$$

Equalizing masses (especially of hydrogens) enables longer timesteps, however, kinetic quantities are affected. *E.g.*, `h<0.0015` ps is recommended for water (with normal hydrogens) to guarantee equipartition error less than 0.5 K, `h=0.002` ps is acceptable if the masses are equalized. Dependants are not equalized (if there is a dependant with mass, it is distributed to nearby sites.)

CAVEAT: Since `equalize.mol` is read from data (def-file) before the ble-file is read, not all variables are available to be used in formulas. The following variables are available: `no, noint, h, E, rho, L[], tau.T, T`. Examples of data:

```
noint=5 h=.01/noint equalize.mol=0.8*(h>0.0015) ! allowed, equalize.mol=0.8
center.K=5 equalize.mol=center.K>1 ! equalize.mol=0: WRONG - no error reported
```

BUG/FEATURE: cannot use constructs as `rho+=1` in the .def file (if `rho` is the default value)—would be executed twice!

`equalize.sp (−big=all species)` Species control for equalization.

> `equalize.sp>=0` The species number for which equalization applies.
>
> `equalize.sp<0` Equalization applies for species in range `0..|equalize.sp|`. Molar masses of species change, the mass of the whole configuration is unchanged.
>
> − `big number` changes into `-(nspec-1)`; i.e., whole configuration is equalized.

`gear.C[] (-9e9,-9e9,..)` Gear's coefficients to be changed, applies only with `gear.init>0`. The array is initialized to large negative values indicating that the coefficient should be left as set by option `-m` and `gear.init`.

`gear.init (3)` Gear integrator version:

> `gear.init=1` The original Gear's corrector coefficients for the right-hand side not containing velocities.

> > `gear.init=2` The original Gear's corrector coefficients for the right-hand side containing velocities.
>
> > `gear.init=3` Corrector coefficients by [50] (lower order but better energy conservation).
>
> > `gear.init=0` Not to be set by a user: `init=0` is set after initialization with positive `init`.

`gear.P[] (1,1,..)` Removed in V3.6h

`group[3] (0,1,2...)` Groups of species for convergence profiles and analysis of cell dipole moment and current density, see Sec. 9.2.3, file simname.`cpi`. Do not confuse with groups of sites in simname.`fix`!
Example: We have brine with species $0 = Na^+$, species $1 = Cl^-$, species 2 = water. Then

```
group[0]=0 ! Na+
group[1]=0 ! Cl-
group[2]=1 ! SPC/E water
```

will define both ions as group 0 and water as group 1. To monitor the current density for ions only, use keyword `Jx0` in simname.`cpi`. To monitor the current density for water, use keyword `Jx1`.

`h (0.001)` The integration step in ps. Example (1 cycle = 0.01 ps):

```
noint=6 h=.01/noint
```

`init (0="continue")` The initial key:

> `0 "continue" or "cont"` Continue the run from the point stored on disk, all measurements the convergence profile and playback will be appended. Some measurements like CPU times are not stored and are initialized for each run. This is the default for every cycle (after every cycle init is set to 0 with the exception of `tau.sat<0`).
>
> `1 "append"` As above but the measurements are initialized; only the convergence profile and the playback files will be appended.
>
> `2 "start"` As above but also the convergence profile will be initialized.
>
> `3 "random"` Everything will be initialized included the configuration. Random shooting algorithm used. See also `MC`. FREEBC: the first molecule (molecule 0) is always centered, other molecules are shot according to Gaussian distribution. See Sect. 13.
>
> `4 "bias" or "slab"` . SLAB version: Cheap way to make a liquid slab, trickle, droplet, cavity, tunnel. The same as `init=3="random"` with a bias to/out of the center of the box in the respective coordinates. With #define SLAB, the geometry is given by `slab.geom`. Without #define SLAB and in the periodic boundary conditions, use option `-d` instead. (In the FREEBC boundary conditions, a Gaussian droplet applies for both init=3 and 4 and `-d` has a different meaning.)
> The initial slab (trickle, droplet) is quite diluted, but it condenses. The box should be longer in the respective directions and the initial cooling should be intensive (otherwise the slab expands instead of condensing to the center). Tested for water with `thermostat="Berendsen" tau.T=0.1`.

Without #define SLAB, only a simple slab is prepared.

(In versions V2.4n–V3.5h, `init=4="slab"` created a slab only both with #define SLAB and without. In versions prior 2.4n `init=4="rawrandom"` meant not correcting constraints of input configurations.)

5 `"crystal" or "lattice"` Everything will be initialized included the configuration. Initial configuration is a regular lattice. See `pins` for available crystal structures. Cf. `No.rotatefrom`.

10 `"asc"` Reads configuration from file simname.asc (see option `-l` for its format.) incl. box size.

This option is not not equivalent to standard read of binary .cfg file (init=0,1,2). Some information is lost, most load options not available).

11 As above + also velocities are read from simname.vel.

12 As above + also accelerations are read from simname.acc. Useless for normal runs (because accelerations are calculated), usable for debugging Ewald summation.

`init<0` Reads playback, frame `|init|`, then as if `init=2`. Symbolically `init=-1="plb"`. Velocities are assigned randomly (as if `initvel=`big_number). If the plb-file contains zero box size (any of Lx,Ly,Lz), the box size is copied from cook initialization (given by values of `rho` and `L[]`). If the box size in the plb-file is nonzero, it overrides the cook initial values; however, some initializations (Ewald, cutoff corrections) are kept from cook initializations. Thus, it is recommended to save the configuration (after `no=0`) and then continue or restart cook.

(Since version 2.6b it is possible to rewrite the plb-file after reading. Utilities `plb2cfg` and `cfg2plb` can be also used for mutual conversion.)

999 `"convert"` Obsolete. Reads simname.1, simname.2 ... simname.no and writes frames to the playback file(s); nothing is calculated. The last configuration treated has number `no`

Any init except 0,1 sets the running simulation time `t` to zero.

`initrho (0)` The initial density in kg/m^3. Applies for initializing the configuration (`init>=3`). The value of 0 (default) means that `initrho=rho`. Example of input data to obtain a random initial configuration for, e.g., water solution:

```
initrho=700 rho=1000 tau.rho=1
init="random" pins=0 Emax=1e5
noint=5 h=0.01/noint no=100 ! 1 ps
thermostat="Berendsen" tau.T=0.1;
```

`initvel (0)` Re-assign the velocities of the first `initvel` molecules according to the Maxwell–Boltzmann distribution with current `T`. It is done just once (`initvel=0` is set after assignment). (Formerly option `-j` with negative arg).

`key (0)` A "command" to execute immediately (i.e., do not use `;` after the command). Useful if `cook*` is called interactively (`cook -s`)/

0=`"run"` No command. This is the default which is re-set after any other command is finished. The simulation is (re)started after data input is ended by `;` as usual.

> `1="sort+x"`
>
> `2="sort+y"`
>
> `3="sort+z"`
>
> `-1="sort-x"`
>
> `-2="sort-y"`
>
> `-3="sort-z"` OUT OF ORDER, use `sort="z"` and similar. The molecules (of each species separately) are sorted according to increasing values of given coordinate of the center-of-mass; the versions with + in the increasing order, version with - with the decreasing order. May be useful, e.g., for selecting some molecules.
>
> `4="cp"` Show the convergence profiles: calls `showcp -p99`
>
> `5="show"` Show the trajectory: calls `show`
>
> `6="rdf"` Show the rdfs: calls `rdfg -g -p`
>
> `7="cn"` Show the coordination numbers: calls `rdfg -c -p`
>
> `8="shell"` Call the shell
>
> `9="quit"="exit"` Exit cook
>
> `97="rv"` `key=99`, then `key=98`.
>
> `98="v"` Paste velocities v_x, v_y, v_z in lines from cfg start, $\boxed{=}$ instead of a number means leave the value unchanged. Single period . quits this module, also ; (in this case, the rest of line is ignored).
>
> `99="r"` As above for the positions.

`L[*] (0)` Box sizes (3D array). If `rho=0` is set, these values are taken literally and are used in configuration initializer as well as target size (if `tau.rho` is specified).

With nonzero `rho`, the box is rescaled to reach given `rho` (i.e., `L[*]` specifies only ratio `L[0]:L[1]:L[2]`). If one, two or three of `L[]` are missing (i.e., = 0), and `rho` is given (> 0), the missing `L[]`s are calculated to reach given `rho`; if more than one `L[]` are, they are the same. E.g., if all `L[*]` are missing and `rho` is given, a cubic simulation cell is created.

`L[]` and `rho` are ignored if the configuration is read (`init<=2`) from file and `tau.rho=0` is specified. This default behavior can be changed by variable `load.L[]`.

`LJcutoff (-3)` Cutoff for the Lennard-Jones interactions. If negative, then (except sign) in the "reference molecule size", which is the potential minimumdistance, sometimes called "van der Waals diameter" or "van der Waals `sigma`", denoted `vdWsig`. For Lennard-Jones, it holds `vdWsig` $=\sigma_{\mathrm{LJ}} \times 2^{1/6}$, where σ_{LJ} is the σ of Eq. (3.1). The minimum possible absolute value of negative LJcutoff for the Lennard-Jones potential is `LJcutoff=-1.5875` (theoretically $2^{2/3}$). The minimum `LJcutoff`$/\sigma_{\mathrm{LJ}}=1.7819$ (theoretically $2^{5/6}$).

`lag.AM (0)` Lag for angular momentum w.r.t. center of mass. Variable `drift` should not contain removing the angular momentum drift. In the periodic b.c., box center is used. In SLAB version, use `slab.sp` for autocenter; then, the configuration is shifted so that the cluster is centered (now for droplet only).

`lag.CM (0)` Lag for center of mass measurement. Variable `drift` should not contain removing the angular momentum drift. In the periodic b.c., box center is used. In

SLAB version, use `slab.sp` for autocenter.

BUG: In the periodic b.c., the jumps are ignored. Statistics is wrong. However, storing CMs in the cp-file (keywords `CMx,CMy,CMz`) in the cp-file may allow to restore the trajectory.

`lag.dim (3)` RGYR only: Number of coordinates (v_x, v_y, v_z) recorded.

`lag.err (32)` Lag for the statistical analysis of time-correlated data, in the number of cycles. In every cycle, one item of data is recorded. See Sect. 14, for details. (One cycle is `noint` integrations steps `h`). Applies only to certain more important variables. Some variables (cell dipole moment and Eel for Ewald) are recorded every step. For them, the lag of `lag.err*noint` is used so that the maximum time of the time correlation function is the same. In the same spirit, `lag.n` is increased.

`lag.J (0)` Lag for the time autocorrelation functions of the current density (in cycles). If `lag.J=1` or `-1`, no blocking calculations are performed, otherwise `lag.n` applies. From these functions the conductivity can be calculated. (There is an equivalent method based on the Einstein relation, see `plb2diff.c` 21.20.) The names of variables are `Jx,Jy,Jz` (can be specified in `SIMNAME.cpi`) Negative `lag.J` means that partial currents (by groups) are not recorded in statistics.
NB: The module caculating also the cell dipole moment is called, cf. `lag.M`.
Called `lag.cond` before V3.5a.

`lag.LM (0)` Lag for linear momentum of the system. Variable `drift` should not contain removing the linear momentum drift.

`lag.M (0)` Lag for the time autocorrelation functions of the cell dipole moments (in cycles). If `lag.M=1` or `-1`, no blocking calculations are performed, otherwise `lag.n` applies. If selected (or `lag.J` is selected), the module calculating also the currents and molecular dipole moments is called (the latter using shorted lag). The dipole moments `Mx,...` are in the program units (1 p.u. = 0.0117501021272 D). Negative `lag.M` means that partial currents (by groups) are not recorded in statistics. The dipole moments are denoted `Mx[#]` (for given group) or `Mx[sum]`. Notes:

- For one group (see `group[]`, use negative `lag.M`, otherwise the same dipole moments would be reported twice (no problem, just inefficient).

- These measurements are performed at cycle end (by `noint` steps) irrespective of boundary conditions and Ewald summation. Possible background (see `el.bg`) terms are not added.

- With Ewald summation (negative `#define COULOMB`), the cell dipole moments denoted `Mx,My,Mz` and calculated every step (*before* the step is performed) are recorded independently with lag `noint*lag.err`. These may contain finer short-lag time correlation functions.

- If the dielectric constant is of interest and calculated in the Ewald b.c. (as recommended), it is still recommended to use at least `lag.M=1` to calculate the dipole moments of individual moelcules to be able to report the saturation.

`lag.n (14)` Blocking (sub-averaging) by the factor of 2 to the block length of $2^{\texttt{lag.n}}$. For the blocked data, maximum lag=2 is used.

`lag.nv (big number)` Velocity-velocity time autocorrelation functions recorded for `lag.nv` molecules.

`lag.v (0)` Lag for the velocity-velocity time autocorrelation functions. Active for #define RGYR only. From these functions the diffusivities can be calculated. (There is an equivalent method based on the Einstein relation, see `plb2diff.c` 21.20.)
WARNING: slow for too long `lag.v` and/or too many molecules `lag.nv`.
NB: With RGYR, module calculation also the cell dipole moment and current density is called, cf. `lag.M, lag.J`.

`lag.Pt (0)` Lag for the time autocorrelation functions of the off-diagonal components of the pressure tensor. From these functions the viscosity can be calculated. Small `noint` is recommended, and the total lag `noint*h*lag.Pt` at least several ps for common non-complicated liquids. See Sect. 14.4.3 for EMD viscosity calculations. Called `lag.visc` before V3.5a.

`load.L[3] (0,0,0)` Box size strategy: the box may be specified in the input data (.def or .get files, either directly, or calculated from `rho` and `N[]`) or loaded (from .cfg, not .plb):

  `load.L[i]=0` The loaded box size is used (i=0,1,2). This is the default.

  `load.L[i]=1` L[i] is taken from data only if the data value is larger (so that no overlap may occur)

  `load.L[i]=2` L[i] is taken from data only if the data value is smaller (i.e., only if the box shrinks, which may result in overlaps).

  `load.L[i]=3` L[i] is taken from data unconditionally (the box may shrink or swell)

Example: You have a periodic cubic box of size $31 \times 31 \times 31$ Å and want to use it as an initial configuration for a slab of size $30 \times 30 \times 75$ Å: while in the input data you should have

```
drift=4+8+16+32    ! center slab in z, remove drift in x,y,z (8+16+32)
el.epsinf=-1       ! slab dipole correction - Ewald only
L[0]=30 L[1]=30 L[2]=75 rho=0 ! box size fixed
load.L[2]=1        ! L[2]=75 will be used
tau.rho=1          ! how fast to reach the box
```

L[0]=30 L[1]=30 will be slowly reached because of `tau.rho=1`

HINT: if you wish to recenter the configuration, use array `shift[]` in the input data.

`load.N (0)` Change the number of molecules. Safe for pure species, limited for mixtures.

  `load.N=0` Default: If the specified (in the def- or ble-files) number of molecules does not match the loaded one (from the cfg-file), this is an error and the program stops.

  `load.N=1` If the specified number of molecules > the loaded one, the missing molecules are inserted randomly; the opposite inequality is an error. Cannot be used for mixture. It is recommended to use `pins=0` to avoid box scaling.

  `load.N=2` If the specified number of molecules < the loaded one, the overflow molecules are omitted; the opposite inequality is an error. In mixtures only the last species can be removed.

> `load.N=3` Either of `load.N=1,2` applies.

`load.n[3] (1,1,1)` Periodically repeat the cell while loading. Useful for simulating fluids of small molecules. To use this feature, prepare first a small configuration of the same name (unless you use a plb-file: see variable `init<0`). Then prepare a def-file of the new configuration (its `N[]` should contain the NEW [large] numbers of molecules) and run with `load.n[]` and and `init=2` in the input data. It is then recommended to randomize velocities by specifying `initvel=999999`, otherwise all the copies will develop in the same way (they are identical incl. velocities) until tiny rounding errors cause the trajectories to diverge. See also utility `plbreplicate`.
WARNING: Automatic cutoff determination is incorrect!

`load.reverse (0="no")` Removed in V2.7a.

`load.tr (0)` Specifies a transformation (swapping, rotation) of the initial box while loading.

> `1="xy" x <-> y`
>
> `2="yz" y <-> z`
>
> `3="zx"="xz" x <-> z`
>
> `4="zyx" z -> y -> x -> z`
>
> `5="xyz" z -> x -> y -> z`

WARNING: may interfere with `L` specified in the data. Example: to immerse a crystal of NaCl into water, prepare the crystal (use order of species Na Cl water; e.g., N[0]=108, N[1]=108), add water (e.g., N[2]=800), then specify a larger box size in the data (may be done indirectly via `rho=1050`) and run cook with `load.L[0]=1 load.L[1]=1 load.L[2]=1 load.N=1`. The original (smaller) crystal of NaCl will be placed into a large box and surrounded by water molecules.

`maxscale (1.03)` Maximum allowed scaling of box or R. If larger scaling is requested, warning is printed and scaling is reduced to range [1/maxscale,maxscale].

`MC (0="no")` WARNING: deprecated unless special requirements. It is recommended to use MD minimization with `drmax` turned on, and large `Emax`.

For `init=3` only. Adds `MC` Metropolis Monte Carlo sweeps (attempted moves for every molecule) at temperature `T` after initializing the configuration. Pressing `^C` (and selecting 1, see option `-i`) interrupts MC and (after finishing a sweep) continues by MD. `MC=-1="forever"` selects running MC until all pair energies fall below `Emax` (or until `^C`). In addition, molecules are not checked for mutual overlaps (given by the energy limit `Emax`) when they are inserted; however, if option `-j`<u>number</u> is specified, then overlaps with first <u>number</u> molecules are checked. One MC step is with probability 1/3 a shot to a random place, with probability 1/3 a random displacement (the length is adjusted during the run) and with probability 1/3 a random rotation (the angle is adjusted during the run). Note that the k-space Ewald contribution (if any) is not included in the MC energy.

FREEBC: central force (see `center.K`) is included into the MC energy.

WARNING: uses a table of pair energies and therefore requires a lot of memory for many small molecules.

mirror (0) For `mirror=1`, mirror inversion is added to a random orientation of a molecule. Applies for the Widom insertion and random initial configuration.

N[*] Number of molecules. Available in simname.def only. The format is:

N[species number]=number of molecules

The number of molecules passed from `blend` (option `-n`) and written to sysname.ble can be thus changed. Note that species are numbered from zero: the 'first' species is referred to by `N[0]`, e.g., `N[0]=500` to simulate 500 molecules.

nm.ampl (0.3) Maximum amplitude of harmonic motion in visualization. Applies if also `nm.modes` given.

nm.dr (0) Normal mode analysis, see Sect. 14.9. Step to calculate the numerical derivative of forces.

> **nm.dr=0** No analysis (default).
>
> **nm.dr>0** Second-order central formula for the 1st numerical derivative of forces. Two evaluations of forces are needed for one gradient.
>
> **nm.dr<0** Fourth-order central formula for the 1st numerical derivative of forces. Four evaluations of forces are needed for one gradient.

The calculation is performed *after* a simulation cycle is finished. The configuration must be well minimized (simulated annealing with `T=0`). With polar, `scf.epsx=0` is recommended (maximum precision). The Ewald calculations should be accurate (esp. small `el.epsr` and longer `el.grid`), in addition, `el.rshift=3` is strongly recommended to avoid small jumps in the potential and forces at cutoff.

Recommended values of `nm.dr` are around 1e-4 to 1e-5 or -1e-3 to -1e-4 without polarizability and Ewald. The accuracy of frequencies is better than 1e-8.

With both dipolar polarizability via large Drude charges (shell $= -1000$) and Ewald, careful choice of parameters (`el.epsr<0.001, el.grid>1000, scf.epsx=0`) and `nm.dr` in the range 1×10^{-3} to 3×10^{-3} guarantees the final relative accuracy about 3×10^{-5} THz $(0.001\,\mathrm{cm}^{-1})$ and the precision of T_ν better than 1×10^{-5}. Since the error $\propto$ `nm.dr`^{2}, the δ^2 extrapolation may improve the precision by one order of magnitude. Negative `nm.dr` may be larger (in abs. value), but the calculation is slower.

WARNING: with LINKCELL, set `box.rmin` to a bit more than needed (for models without "lone dependants" at least `nm.dr` and set `box.max14`.

BUG: Not correct for general constraint models with bond constraints connected by a non-fixed angle. Good for rigid models (as water).

nm.eps (1e-9) Normal mode analysis: accuracy (of the Jacobi method). Negative `nm.eps` selects the verbose mode (iterations and some other technical info is printed).

If the accuracy is finer than the numerical noise, the additional code is usually able to interrupt iterations without too much loss of efficiency (especially for the non-constraint variant where the convergence is quadratic); however, if the numerical noise (with constraints) is larger than sqrt(eps), the algorithm would run forever.

Pressing Ctrl-C or signalling SIGINT (`kill -2 PID`) breaks the iterations. WARNING: The CPU time scales as N^4; the estimated CPU time for about 1000 atoms may be hours.

Typical relative errors in the averaged vibrational temperature are 1×10^{-6} for nonpolarizable models and 1×10^{-5} for polarizable models.

nm.frames (7) Normal mode analysis: number of frames in the generated trajectories to visualize the harmonic motions. Positive = cos-like motion, negative = linear motion (not recommended). The higher value, the smoother motion.

nm.key (0) Additional control of convergence, use in case of problems. For verbose mode, use `nm.eps<0`.

   0   Standard end of iteration criterion used (checks speed of convergence and one-step error).

   >0 In addition, stops if the error during `key` consecutive iterations increase. Useful to avoid infinite loop in case of problems (note also that the iterations are interruptable by SIGINT (Ctrl-C).

   -1 Stops if the first eigenvalue becomes negative. The accuracy of the results is somehow compromised, but in case of problems, the iterations are interrupted as soon as possible.

nm.mem (13) Maximum memory which can be requestes for matrices (only the largest terms terms $\propto N^2$ are counted), in GiB (1 GiB = 1024^3 B). The default 13 GiB is suitable for a computer with 16 GiB RAM provided that no other memory-consuming programs are running. Note that this is the peak; during diagonalization unused arrays can be swaped out.
BUG: The arrays are freed at the end of the routine, not when no longer needed.

nm.method (0) Version of the method to calculate normal modes of vibration.

   0   For models without constraints. The matrix to diagonalize is symmetric, the Jacobi method is used.

   1   For rigid molecules (or molecules with rigid parts connected by flexible bonds). The constraint matrix is constant. The matrix to diagonalize is not symmetric, the generalized Jacobi–Schur method is used.

   2   For general bond constraints. The constraint matrix is not constant. The matrix to diagonalize is not symmetric, the generalized Jacobi–Schur method is used.

   3   As above, slow fool-proof algorithm (for debuging purposes). Use `nm.method=2` instead.

   8   Flag to be added to methods 1,2,3: call Octave to calculate the eigenvalues instead of the Jacobi–Schur method. Faster but memory consuming.

nm.modes (0) Normal mode analysis: number of fundamental modes visualized. Positive = modes are counted from the lowest ones (corresponding to translations/rotations), negative from the highest frequencies, 0 means that the eigenvectors are not calculated.
BUG: The algorithm is very slow for larger systems (a few hundred of atoms).

nm.zero (3 or 6) Normal mode analysis: number of modes which are treated as zero for the calculation of the Helmholtz energy of a crystal. The default is 3 for the periodic b.c. (three zero translations) and 6 for free b.c. (3 translations + 3 rotations; this is incorrect for a linear molecule!).

no (10000) Number of cycles to be performed. Note that you can interrupt the calculations before `no` cycles gracefully by [Ctrl-C] or signaling `kill -2`. See also option -number.

No.cell[*] (0) Only for LINKCELL version: number of subdivisions of the simulation box in each dimension. Default `No.cell=0` means automatic selection according to the number of sites in one simulation cell, `No.percell`.

No.first (0) DEPRECATED, #define LOG needed.
Energy of `No.first` molecules (without Ewald k-space part, if any) used as the 4th column of the convergence profile in NVT. (REMOVED in 2.8c: When `No.first=0` (default) then the 'intramolecular' energy of all molecules is recorded.) Not supported for LINKCELL. `cache=1` required (from V3.0h, `cache>1` no longer supported!).

No.molspan (0) Version PERSUM only: maximum span (min–max if any rotation) of the molecule (and something more for NPT). This value serves for determining the number of images around an atom that guarantees that no interaction within cutoff is lost. The default is for monoatomic molecules! The formula is (for each coordinate separately):

$$n_x^{\mathrm{img}} = 1 + \mathrm{int}\left[\frac{\texttt{cutoff} + 2 * \texttt{No.molspan}}{L_x}\right] \tag{9.9}$$

To enhance speed with a small decrease of precision (some interactions close to the cutoff lost), `No.molspan` maybe a bit less than the maximum span; for monoatomic molecules it may be negative.

No.occup (0) Applies to the LINKCELL version only: Histogram of cell occupancies is calculated and printed in every step. If `No.occup>1`, then max number atoms in a cell are recorded, for `No.occup=1` certain default is chosen. For debugging purposes.

No.percell (5) Only for LINKCELL version: Calculates `No.cell[*]` so that there are about `No.percell` sites per each cell and the cells are (approximately, if not possible exactly) cubic.

No.rotatefrom (0) For initialization (`init="random"` etc.): rotate molecules randomly from molecule of given number. Default $= 0 =$ rotate randomly all molecules.

noint (10) Number of steps per one cycle. There are (`noint-1`) steps performed without measurements, and 1 step with the measurements included. Note that measurements are quite expensive so that do not use too low `noint`.

nomax (0) The simulation stops if the total number of measured cycles reaches or exceeds `nomax`. The default (`nomax=0` turns off the check (the simulation runs until all data sets are processed or `key="quit"` is specified or EOF reaached). See also `stop`.

norm (0) REMOVED: see `drift` and `conserved`.

nshift (No.N) With `shift` and `vshift`: number of affected molecules. The default is a large number = all molecules.

> **>0** The shift (push) applies to the first `nshift` molecules only.
>
> **<0** The shift (push) applies to the first `No.N+nshift` molecules, the remaining `-nshift` molecules are shifted (pushed) in the opposite direction.

omega (0) Not implemented in the current version! The relaxation parameter for the `directiter` method for calculating the Lagrange multipliers.

omegac (-1.2) The relaxation parameter (mixing iteration parameter) for the `Scorrect` method for correcting constraints or for SHAKE. If negative, automatic optimization is performed (use option `-v&4` to get detailed protocol). See Sect. 11.3, for details.

P (1e5) Pressure to be kept constant (if `tau.P` is nonzero), in Pascal.

pins (0.01) Insertion probability and more.

> init=3,4 0<pins<1 Lowest acceptable probability for inserting molecules while the configuration is initialized. If the insertion probability falls below `pins`, the density is decreased (box size increased) and a new attempt to fill the simulation box with molecules is made.

> init=3,4 -1<pins<0 As above with `|pins|`; in addition, if the insertion probability is higher than `sqrt(|pins|)`, a new attempt is made with a higher density. Thus, after several iterations, a density with an insertion probability in `(|pins|,sqrt(|pins|))` is found.

> init=3,4 pins=0 Does not change density while initializing, i.e., continues in shooting molecules even if very unsuccessful; it's user's responsibility to choose a suitable `initrho`.

> init="crystal" pins="auto"

> init=5 pins=0 Automatic selection of the lattice (minimum number of vacancies)

> init="crystal" pins="sc"

> init=5 pins=1 Simple cubic lattice (with possible vacancies at random positions)

> init="crystal" pins="bcc"

> init=5 pins=2 Body-centered lattice (with possible vacancies)

> init="crystal" pins="fcc"

> init=5 pins=3 Face-centered lattice (with possible vacancies)

> init="crystal" pins>3

> init=5 pins>3 (NOT TESTED RECENTLY, See Sect. 17.1) Structure contained in file with name `cfg`ns`.`pins, where ns is the number of sites per molecule and pins the number of molecules, will be used and, if necessary, periodically copied; if the total number of molecules is not an `integer`[3] multiple of pins, then vacancies will be present. The box size will be derived from the box size stored in the file. The configuration file can be created using option `-l` by `cook`, it only must be renamed. May be combined with option `-j`

poteps (-1e-5) Accuracy for fool-proof internal check of site–site (non-bonded) potentials and forces. Note that internally cook macroizes these functions in two versions – with and without measurement (of virial and energy). This test compares all these versions and also the derivative of minus the potential energy and forces. The active accuracy is a sort of relative accuracy times `poteps` for numerical derivative, otherwise `poteps`[2] applies.

If `-v4`, tables of potentials and detailed debug info are printed. (Was switch SS_DEBUG in V2.7t and older.)

poteps=0 The test is turned off.

poteps<0 The test is done only once at job start.

poteps>0 The test is done every batch (data set), and if potential adjusting is on
(`tau.sig`, see Sect. 12.3.4), then also whenever σ changes. (WARNING: heavy
output with option `-v4`!)

quit (0="no") REMOVED in V3.6j, see `key` for some functionality.

reread.by (1) In the read mode (options `-m0` or `-m1`), the stride of the loop. (New in V3.0a.)
You should change the simulation name to prevent rewriting your simulation results!
Example of usage:

```
cook SYSNAME SIMNAME            # simulation - SIMNAME.plb generated
cp SIMNAME.def SIMNAME2.def   # not if SYSNAME.def used
cp SIMNAME.cfg SIMNAME2.cfg
cp SIMNAME.cpi SIMNAME2.cpi   # if needed
NPLB=`plbinfo +f SIMNAME.plb` # number of frames
cat > SIMNAME2.get <<EOF
reread.from=1
reread.by=1
reread.to=$NPLB
init="start";
EOF
cook -m1 -f SYSNAME SIMNAME2 SIMNAME.plb # recalculate energies, pressure, RDF, etc
```

reread.from (1) In the read mode (options `-m0` or `-m1`), the first frame or configuration
read; the first frame is 1. (New in V3.0a.)

reread.to (0) In the read mode (options `-m0` or `-m1`), the last frame read (incl.). New in
V3.0a.

removemol.n (-1) At job end (delimited by ; in the data – further semicolons are ignored),
remove molecule number `removemol.n` (if it exist).

To restart such a simulation from the saved cfg-file (using `init=2`), either change the
number of molecules in the def file (`N[0], N[1],...`), or use `load.N=4`.

The saved cfg-file has the molecule removed, in the plb-file (if requested), it is present
because the last frame in the plb-file is save before removing the molecule (new in V3.6q).

Cf. `slab.out`.

rescale (15="xyzCM") Mode for box rescaling used by a barostat (see `tau.P`), density/box
rescaling (see `tau.rho`), and virtual volume (area) method (see `dV`), and the MTK
barostat (see `thermostat="NPT"`). Sum of powers of 2:

1   x coordinates are rescaled; `"x"` in the mnemonics below.

2   y coordinates are rescaled; `"y"` in the mnemonics below.

4   z coordinates are rescaled; `"z"` in the mnemonics below.

8   Rescaling is based on the molecular center-of-mass (default), `"CM"` in the mnemonics
below. This is the preferred choice for models with fixed (constrained) bonds unless
the MTK barostat (`thermostat="NPT"`) is used.

16 Pressure tensor components are calculated and used separately; (#define PRES-SURETENSOR=3 is required). All specified dimensions of the box fluctuate independently; thus, this choice is suitable for crystals and inapplicable to fluids. Denoted by uppercase `"X","Y","Z"` in the mnemonics below.

If this bit is not set, and a barostat is used, the isotropic virial pressure is used to calculate the rescale factor or rescale dynamics; #define PRESSURETENSOR is not required. Consequently, all specified dimensions of the box fluctuate synchronously (their aspect ratio is kept, a cube with `rescale=7` is always a cube). This is the default suitable for isotropic systems (fluids).

This flag is irrelevant with box control (see `tau.rho`).

New in V3.1i: with barostat (`tau.P`), the affected box sizes are included in statistics. In order to include the respective convergence profiles, Lx,Ly,Lz should be given in the .cpi file (do not use +Lx,+Ly,+Lz, the box statistics would be diplicated).

32 x and y scalings are made the same; with a barostat this is equivalent to using $(P_{xx} + P_{yy})/2$ to calculate scaling in both the x- and y-directions. Denoted by `"XX"` in the mnemonics below. Useful only with `rescale=16` because otherwise the scaling is already isotropic.

64 All x, y, z scalings are made the same; `"XXX"` in the mnemonics below. Useful only with `rescale=16` because otherwise the scaling is already isotropic[2]

In principle, 1+2+4 and 1+2+4+16+64 denote the same uniform box rescaling. In the latter case, $\text{Tr}(\overset{\leftrightarrow}{P})/3$ (as calculated from the pressure tensor) is used instead of the virial pressure. Although both values should be the same, they are not exactly the same because of inherent algorithm inaccuracy. With Ewald summation the error is small (usually a few MPa), however, it is large with cutoff electrostatics, the pressure-tensor value being usually more accurate.

Examples:

- `rescale="ZCM" tau.P=10 P=1e6 corr=3` will maintain $P_{zz} + P_{\text{cutoff corr.}}/3 = 1$ MPa by changing L_z only. Time constant of the Berensen barostat would be 10 ps in the ideal gas approximation, shorter in condensed phases (see `bulkmodulus`).

- `rescale="zCM"` or `rescale=12` selects molecule-based rescaling (good for small molecules) of z-coordinates only

- `rescale="xyzCM"=15` selects molecule-based rescaling (all coordinates).

The full list of available mnemonic codes is:
`"x", "xCM", "y", "yCM", "xy","xyCM", "z", "zCM",`
`"xz","xzCM", "yz","yzCM", "xyz", "xyzCM",`
`"X","XCM", "Y","YCM",`
`"XY","XYCM", "Z","ZCM", "XZ","XZCM", "YZ","YZCM",`
`"XYZ", "XYZCM", "XX","XXCM", "XXZ","XXZCM", "XXX","XXXCM"`

`rdf (structure)` Variables controlling measurements of the radial distribution functions (`rdf.grid`, `rdf.cutoff`, `rdf.onefour`), See Sect. 14.7.

`rg.cp (0)` #define RGYR required. Warning: radii of gyration are calculated for molecules, this should be extended.

---

[2]That is why there is no `rescale="xxx"` mnemonic.

rg.cp >= 0 Radius of gyration and end-to-end distance of molecule number `rg.cp` recorded in convergence profile. `rg.cp=0` is recommended for e.g. a protein in water: only Rgyr of the protein will be recorded.

rg.cp = -1 Averaged radius of gyration and end-to-end distance over all molecules recorded in convergence profile. `rg.cp=-1` is recommended for bulk pure molecular fluid.

The value of `rg.cp` does not affect statistics which is recorded for each species separately (if possible by the values of `rg.end`)

rg.end[0 (0)] Head atom for measuring the "end-to-end" distance. Negative value means `ns+rg.end[0]`, where `ns` is the number of sites. For instance, `rg.end[0]=-1` is the last atom in the molecule.

rg.end[1 (-1)] Tail atom for measuring the "end-to-end" distance. See above.

rho (0) The reference and target density in kg/m^3. If `tau.rho` is specified, this density will be reached and kept constant. The box is rescaled so that its shape (`L[0]:L[1]:L[2]`) is the same. See also `L[*]`. If one or two of L[] are missing, and `rho` is given, the remaining L[] are calculated. If `rho=0`, all three `L[0],L[1],L[2]` must be given and are used without rescaling.
See also `initrho`.

scf.E (0) POLAR only: Virtual electrostatic field that is applied to a configuration in order to calculate the optical permittivity of a polarizable model. The SCF accuracy and convergence are controlled by `scf.epsx,scf.omegax`. See `scf.Estride` below. Note that the electrostatic field changes sign and coordinate every sample. The result is printed as `chi`; in the tin-foil b.c., $\epsilon^\infty = $ `eps(high) = 1+chi`.

scf.domega (0) POLAR only: Automatic setup of the optimum `scf.omega`. Set `scf.omega` to a low enough value and `scf.domega` to a small increment and `no` to a big enough number. `scf.omega` will be incremented by `scf.domega` every cycle (of `noint` steps). If divergence is detected (number of iterations needed to reach given precision `scf.eps` increases), the value of `scf.omega` is set to `scf.omega+scf.margin` and the sweep is interrupted (data after ";" are read and executed). Also, `scf.domega` is set to zero. (NB: `scf.margin` is negative.)

Caveat: Negative `scf.eps` (number of steps) can be used, too; in this case, the calculation will likely crash but the limiting `scf.omega` is available in the protocol.

scf.Estride (0) POLAR only: How often to perform the sampling of the optical permittivity, in the unit of cycles by `noint` timesteps (1=every cycle, 2=every second, etc.). Zero turns off the feature. See `scf.E`.

scf.eps (1) POLAR only: Controls SCF iterations/accuracy during the simulation.

scf.eps> 0 The accuracy of induced dipoles (in one iteration), in program units (1 p.u. = 0.0117501 Debye = 0.0024463 eÅ). This value is used during MD simulation. If ASPC is used, `scf.eps` should be large enough to guarantee one iteration per MD step. (Called `epsp` for V<2.6f). See Sect. 15.6.

scf.eps ≤ -1 Fixed number of iterations given by |`scf.eps`|.

For more (less useful here) options see `scf.epsx` below.

`scf.epsx (3e33)` POLAR only: Accuracy of induced dipoles outside MD (as P measured by virtual volume change, normal mode frequencies, recalculation with options `-m1 -f`, etc.).

> `scf.epsx=3e33` This default implies certain value which can be reasonable for the virtual volume/area change algorithms. Generally it is recommended to try several values to resolve the speed/accuracy tradeoff.
>
> `scf.epsx` ≤ -1 Fixed number of iterations given by `|scf.epsx|`.
>
> `scf.epsx` > 0 Accuracy (of one iteration) in the internal program units (1 p.u. = 0.0117501 Debye = 0.0024463 eÅ).
>
> $-1 <$ `scf.epsx` < 0 "Maximum precision" algorithm, useful with normal modes calculations and similar. The iterations are stopped when the following three conditions are met:
>
> > 1. at least `1+scf.maxit/20` iterations have been calculated
> > 2. the one-step error is below `|scf.epsx|`
> > 3. for the first time the one-step error *increases*
>
> The last condition happens when the SCF error is about the same as the numerical error. However, convergence problems may pass unnoticed.
>
> `scf.epsx=0` Similar as above with the following two conditions:
>
> > 1. at least `2+scf.maxit/5` iterations have been calculated
> > 2. for the first time the one-step error *increases*

The optimum `scf.epsx` depends on `dV` or `nm.dr` and Ewald setting. For predicted force field and one iteration/step, one should have `scf.epsx` $\ll$ `scf.eps`. For integration methods with full iteration of the self-field, both `scf.eps` and `scf.epsx` should be small enough.

The course of iterations can be monitored by flag 4 in option `-v` (`-v7` incl. the default `-v3`).

(Called `epspx` for V<2.6f).

`scf.epsq (0.8)` POLAR only:

> `0<scf.epsq<1` `scf.eps*(1-scf.epsq)` will be used as the actual error limit EPSP in the 1st step, instead of `scf.eps`. In subsequent steps,
>
> > `EPSP := EPSP*scf.epsq + scf.eps*(1-scf.epsq)`
>
> is performed after every step so that after some time `scf.eps` is reached. After every data set (ended by ;) read, `scf.epsq` is made negative and this function is turned off. This is because the history for the predictor IS saved across batches.
>
> `scf.epsq<=0` turns off this function and EPSP=`scf.eps` is always used.
>
> `scf.epsq>=1` is invalid

`scf.margin (-0.03)` Polar only: See `scf.domega`.

`scf.maxit (30)` (POLAR only) Maximum number of iterations to calculate the induced dipoles. Applies both to the simulation and virtual volume change. If not enough to reach the predefined accuracy, it is doubled with verbose output on. (To use a constant number of iterations, use negative `scf.eps`)

`scf.omega` (POLAR only) The mixing iteration parameter for iterations of the self-consistent field during simulation. Can be also given in % as option `-^`. (Called `omegap` for V<2.6f).

`scf.omegax (0.9)` The mixing iteration parameter for iterations of the self-consistent field for virtual volume change. The optimum values are slightly below unity.

`scf.test (0)` If nonzero, the accuracy test of SCF is performed every cucle with respect to the fully iterated SCF (see `scf.epsx`, `scf.omegax`). Variables `Pstd`, `Pmax` in the CP contain the respective standard deviation and and maximum errors, the averages are `Polar stderr` and `Polar maxerr`. With option `-v32`, files with the errors are also recorded.

`shear (0)` For measuring the shear viscosity, $shear = \dot{T}\eta$ (estimated using ideal gas heat capacite). Only if compiled with `#define SHEAR`. See Sect. 15.14.

`shift[*]` The whole configuration (or a part of it – see `nshift`) is shifted by this 3D vector before the first MD step. The vector is given in Å. This vector is then set to zero so that no shift is performed in the next data set (unless specified again). The maximum allowed shift in periodic b.c. is ±L (no check!).

`SI (1="yes")` REMOVED in V3.6g, all energies are in J/mol.

The following forces are active in the SLAB version only: (in version 2.6h called center.*)

`slab.ext.center (0)` The slab center (denser part) extended by `slab.ext.center` bins.

`slab.ext.span (0)` Turns on the feature, the number of bins around the extended parts which are replicated to fill both phases.

`slab.ext.zero (0)` The slab outer part (gas) extended by `slab.ext.zero` bins.

WARNING: `slab.ext.*` features are likely out of order because the default slab position has changed (centered at $z = 0$).

For removing the stacking error of the surface tension, see Sect. 31.3. In the postprocessing stage, the cutoff corrections are calculated from the calculated z-profiles. In addition, the z-profiles are extended by replicating some bins in both phases and the corrections recalculated. Autocenter (see `slab.sp`) is required. This feature is turned on by a positive `slab.ext.span`.

```
    Example of bin replication scheme for 26 bins with slab.ext.span=2,
    slab.ext.zero=4, slab.ext.center=6. Spaces are added for readability.
        ABCDEFGHIJKLM        NOPQRSTUVWXYZ ->
    ZABY ABCDEFGHIJKLM MNOLMN NOPQRSTUVWXYZ
```

`slab.geom (2)` Slab geometry extended in V3.5h to:

   `slab.geom=0="drop"="droplet"` Droplet (ball)

  slab.geom=1="trickle" Trickle (cylinder) along the x-direction

  slab.geom=2="slab" Slab (film) perpendicular to the z-direction

  slab.geom=3="cavity" Spherical cavity

  slab.geom=4="tunnel" Tunnel (cylindrical cavity) along the x-direction

  slab.geom=5="bals" Slab of gas perpendicular to the z-direction (shifted/inverted slab)

The choice of geometry affects the measured distribution functions. Autocentering (see `slab.sp`) is recommended.

slab.grid (0) SLAB version only (see Sect. 15.8). The meaning of `slab.grid` depends on variable `slab.max`:

  slab.max is not specified: number of histogram bins in the range $[0, z_{max})$ (where z_{max} may fluctuate).

  slab.max is specified: number of histogram bins per 1 Å (not suitable for changing box size)

May be a real number (`slab.grid=0.5` without `slab.max` gives the histogram bin width of 2 Å).
(Called `densprof.grid` in versions 2.6h and older.).

slab.K (0) Fourier transform slab cutoff correction (FTSCC), new in V2.8h. Incompatible with the homogeneous cutoff corrections, i.e., `corr=0` is needed in input data[3]. See also `slab.range`, see Sect. 31.1. Wave number `slab.K` is not included (C-style); thus, `slab.K=1` is error. Recommended values are from 3 to about 10.
Uses the initial box size to set up the correction coefficients. Thus, NVT (or NPT with small fluctuation of box sizes) is supported.
BUG: Cannot be combined with any method which changes the initial box; e.g., `load.L[]`, `load.n[]`.

slab.Kz[*] (0) Harmonic potential constant in the z-direction, see `slab.n[]`.

slab.Lx[4] (0)

slab.Ly[4] (0) Box sizes for the "zone melting" ensemble, given as polynomials in temperature:

$$\text{Lx} = \sum_{i=0}^{3} \text{slab.Lx}[i]T^i \tag{9.10}$$

SLAB & 1 must be #defined (8 prior V2.8c). See `tau.L` for details, see Sect. 14.10.5,

slab.max (0) Max. z or r recorded in the density profiles.

  slab.max=0 The maximum range of the density profiles is determined dynamically from the current box size (fluctuating in the NPT ensemble). `slab.grid` is the number of division point per this variable range. This is the default.

---

[3]Since specifying `corr&3` does not affect the trajectory, it is still possible to extract the uncorrected results for many variables.

slab.max>0 Fixed range, `slab.grid` is the number of division point per 1 Å. Points beyond this range are lost.

CAVEATS: `slab.max<Lz` for a slab is strongly discouraged because function modulo is used in calculating the indices. For a droplet or trickle geometries and cavities, the last histogram bin contains the sum of all further molecules/atoms.

(Called `densprof.zmax` in V2.6h and older, `slab.zmax` in V2.6h–V3.5g.)

slab.mode (0) (Called `densprof.slab` in V2.6h and older.). Sum of flags:

1   Deprecated. The system is assumed to be in the slab geometry and surface tension is calculated from previously recorded components of pressure tensor and virial pressure. Slab-based cutoff corrections are calculated from the z-density profile (post-processing), see Sect. 31.2. It is recommended to use `corr=0`, although the correct results can be extracted even if `corr&3` is set. This method is deprecated, use the Fourier transform slab cutoff correction, see `slab.K`.

2   If `slab.mode&2`, and `dV` and `rescale` are set appropriately, the test area method is used instead of virial pressure (see Sect. 15.8). Also `slab.mode&1` must be set.

4   (Used internally for calculating the stacking correction to surface tension – see `slab.ext.*`, **do not set manually!**).

8   POLAR only: Drude-dipole is rescaled to a dipole of z-length given by the z-profile grid. To be used for too short Drude dipoles.

slab.n[*] (0) Number of molecules selected for adding forces in the z-direction.

Example:

```
slab.n[0]=800      slab.n[1]=200
slab.Kz[0]=100     slab.Kz[1]=100
slab.z[0]=20       slab.z[1]=-20
slab.z0[0]=10      slab.z0[1]=2
                   slab.ns[1]=2
```

Molecules 0..799 (the first 800 molecules: we number from 0 as in C) are kept in a slab centered at z=Lz/2+20 and 20=10+10 wide. First 2 sites of molecules 800..999 are kept in a slab centered at z=Lz/2–20 and 2 wide.

slab.ns[*] (0) Affected sites are: `slab.ns[*]=0` all, `slab.ns[*]>0` only first `slab.ns[*]` sites, `slab.ns[*]<0` only last `|slab.ns[*]|` sites. See `slab.ns[]`.

slab.sp (0x7fffffff) Automatic slab centering for the measurements of the density profiles. It affects the measurement only, not trajectory. In the slab, centering is based on formula

$$c_{\text{center}} = \frac{L_z}{2\pi} \arg\left[\sum_i \exp\left(\frac{2\pi i z_i}{L_z}\right)\right] \qquad (9.11)$$

and similarly for x and y if needed for other shapes. In Eq. (9.11), arg is the phase (argument, angle) of the complex number in [] and the sum is over centers of mass of molecules of species defined below. Note that the "center" of the density profiles is always at $z = 0$ or $r = 0$; particularly, to see the slab "nicely centered", you should periodically shift the slab; e.g.:

```
    tabproc "(A+78/2)A\`78" B C D < sss.cm.AA-3.z | jksort -f > aux.z
    plot aux.z:A:B :C :D
```

Variable `drift` should not contain a request for centering the same coordinates.[4]

> `slab.sp`≥ 0 specifies the species number used for centering (from 0 to `nspec-1`).
>   Example: `slab.sp=0` selects species 0 for centering.
>
> `slab.sp`< 0 means that all species from number 0 to |`slab.sp`| (incl.) are used.
>
> `0x7fffffff` density profiles w.r.t. box center [default] (?)
>
> `0x7ffffffe` (any big number except `0x7fffffff`) density profiles w.r.t. origin $(0, 0, 0)$.

`slab.sym (big number = off)` Monitoring slab asymmetry for given species coded in the same way as slab.sp. Distance from the center calculated using `slab.sp` is calculated and printed to the cp-file of entry `zsym` is given on the cpi file.

`slab.out (0)` SLAB required. The default `slab.out=0` turns the function off.
If the distance of the center of mass of a molecule from the slab center plane / trickle center axis / droplet center (see `slab.geom`) exceeds `slab.out`, the molecule is removed from the configuration and the simulation stops immediately. Only one "evaporated" molecule can be removed. If also `slab.outc` is set (see below), this distance is actually `slab.out-slab.outc` at job start to catch a possible second molecule if the farthest molecule was removed previously.
The test is reversed for cavities (`slab.geom > 2`; WARNING: not tested).
The test is performed once every cycle (`noint` steps). You should guarantee that the CoM of a molecule cannot travel across the respective periodic boundary. It is recommended to use the slab/trickle/droplet centering, see `slab.sp`. Example:

```
a=35
L[0]=a          ! x-box (the trickele is periodic in x)
L[1]=2*a        ! y-box
L[2]=2*a        ! z-box
tau.rho=1
noint=30 h=0.05/noint ! with speed 1 km/s = 10 AA/ps, max distance = 0.5 AA
corr=0          ! no corrections at all
slab.geom="trickle"
slab.grid=1     ! density profile grid = 1 bin/AA
slab.max=a      ! DPR range (for NVT: slab.grid is in bin/AA)
slab.prt=2      ! only molecular density profile in kg/m3 is determined
slab.sp=-1      ! automatic centering for all species
epsc=1e-10      ! for adiabatic simulation
slab.out=a-1    ! 1 AA before half box
```

In the prt-file, the positions of all sites of the molecule are printed; the number of higher derivatives is determined by option `-r`; the default `-r2` will dump also velocities (precisely, $\vec{r}_i(t) - \vec{r}_i(t - h)$ for Verlet and $h\dot{\vec{r}}_i$ for Gear) are printed in Å).

To restart such a simulation from the saved cfg-file (using `init=2`), either change the number of molecules in the def file (`N[0]`, `N[1]`,...), or use `load.N=4`.

---

[4]Called `densprof.sp` in versions 2.6h and older.

The saved cfg-file has the molecule removed, in the plb-file (if requested), it is present because the last frame in the plb-file is save before removing the molecule (new in V3.6q).

See also `removemol.n` and option `-d` with negative argument.

slab.outc To be used with `slab.out`: The size of a cluster that may evaporate at once. This is a cheap imperfect trick how to remove a cluster that has evaporated because only one molecule can be removed at once. Note that if a molecule of a fast rotating dimer approaches distance `slab.out`, it is removed. The remaining molecule may travel backwards. However, if the distance `slab.out` is decreased to `slab.outx=slab.out-slab.outc` at (re)start, such a molecule is caught, too. After a cycle (of `noint` steps), `slab.outx:=slab.outx*0.75+slab.out*0.25` is performed so that after a while the limit is `slab.out`.

slab.prt (31=all) Controls which density profiles will be printed. The printouts are generated from the data stored in `SIMNAME.dpr`. Sum of flags:

 1   Molecule density profiles (for centers of mass) and the sum in the number density units Å^{-3}, ext=`.cm.AA-3.z`.

 2   Molecule density profiles (for centers of mass) and the sum in the mass density units kg m^{-3}, ext=`.cm.kgm-3.z`.

 4   Site density profiles and the sum in the number density units Å^{-3}, ext=`.site.AA-3.z`.

 8   Site density profiles and the sum in the mass density units kg m^{-3}, ext=`.site.kgm-3.z`.

16   Charge density profile (molecule-based and the sum) in the units of $e\text{Å}^{-3}$, ext=`.q.eAA-3.z`.
     Since V3.4n, it is calculated correctly for Gaussian charges using the true distribution.
     BUG: Before V3.4n, the charge profile was calculated incorrectly as for point charges.

Hint: use `slab.prt=21` for all in Å^{-3} (not mass units)

slab.range (8) Range of integration over z-periodic slab, in L_z. Another useful value is 0.5. See also `slab.K`, see Sect. 31.1.

slab.T (-1) Separate thermostat for particles with z-coordinates in the range [`slab.Tz0*Lz`, `slab.Tz1*Lz`). Requires Verlet/SHAKE and Andersen/Maxwell thermostat (incl. center-of-mass versions). For instance, to partly melt a crystal in a slab, you may try `slab.T` $\approx$ 2T and `tau.T` in the range 0.1 to 1. Turned off by `slab.T<0`

CAVEAT: Not fully implemented with the Berendsen thermostat. The algorithm just changes the velocity scaling factors in both slabs using the global temperature (it does not determine the temperatures separately in both slabs). In turn, if the heat conduction is too slow (compared to `tau.T`), the difference of both temperatures grows to more than T vs. `slab.T`. Nevertheless, even this partial algorithm may be useful. For instance, to melt a crystal in a slab, you may try `tau.T` $\approx$ 2T and `tau.T=1` or more (depending on the system size) and watch the melting process.

slab.Tz0 (0.25)

slab.Tz1 (0.75) Defines range of z-coordinates (in the units of L_z) to apply a separate thermostat (see `slab.T`).

slab.wall.sig (3)

slab.wall.epsrho (0) SLAB==4 required. DEPRECATED, WILL BE REMOVED, use
SLAB + wall.* instead.
WARNING: drift not set automatically!!!
Rather inconsistent LJ wall force at z=0. slab.wall.epsrho = energy density in K/Å^3 of
the Lennard-Jones ϵ:

$$U_{\mathrm{LJ-wall}}(\vec{r}) = -2\pi\epsilon\mathcal{N}\sigma^3 \left[\frac{1}{3}\left(\frac{\sigma}{z}\right)^3 - \frac{2}{45}\left(\frac{\sigma}{z}\right)^9 \right] \tag{9.12}$$

where $\mathcal{N}$ is the particle number density (in Å^{-3}). All particles interact in the same way.
The minimum is at $z = \sigma/\sqrt[6]{2.5}$.

slab.z[10] (0) The center of the force, with respect to the box center in the z-direction
(L[2]/2)

slab.z0[10] (0) The added energy term is zero for $z \in$ [z-z0,z+z0]. (NOTE: in older
versions, this was called slab.r0)

slab.z1[10] (0) A 'bias function' is added:

```
    ---------        ----------
        |\          /|       ^
        | \        / |       |
        |  \      /  |       Kz
        |   \_____/  |      _|_
        A   B  C  D  E
```

This function (actually quadratically smoothed) is selected by slab.z1!=0, where
|BC|=|CD|=slab.z0 and |AC|=|CE|=slab.z1, and slab.Kz = height of the outside part
from bottom (bottom=potential 0).
Notes:
With slab.Kz<0, and a slab placed at B..D, there are molecules with a higher probability
$\propto \exp(-$slab.Kz$/RT)$ outside the slab. Note that the meaning of slab.Kz differs from
the standard (U-shape) case for slab.z1=0.
Since V2.8c always active for SLAB (used to be #define SLAB 1).

sort (0="none") If one of 1="x", 2="y", 3="z" is specified, the molecules (of each species
separately) are sorted according to increasing values of given coordinate of the center-of-
mass. If one of 1="-x", 2="-y", 3="-z" is specified, the sort order is decreasing. Sorting
is performed once before writing a cfg-file (or asc), then sort=0 is set. May be useful,
e.g., for selecting some molecules.

stop (0) Stop simulation if the time since (re)start, i.e., last init>=2, exceeds stop ps.
Note that nomax applies to the number of measurement, i.e., also last init=1. If init=1
is not used and the cycle length does not change (see h and noint), then nomax and
stop=nomax*noint*h are equivalent.

T (300) Temperature for the thermostat in K. tau.T is the correlation time to keep the
temperature constant

T is also the approximate initial kinetic temperature (init>=3).

`t (0)` The simulation time (in ps) can be changed directly. Normally do not use; may be useful with `rho<0` (instead of shifting time in simname.box).

`tau.CM (0)` Experimental patch to sample the center of mass of a periodic simulation cell. Good with Nose and MTK (NPT), maybe also with Bussi NVT. These algorithms provide what is called [14] $(N-1)VT$ or $(N-1)PT$ ensembles because one degree of freedom, the motion of the center of mass, is not sampled. With this patch, NVT or NPT ensembles are simulated. The implementation is simple and the correlation time `tau.CP` must be long enough because the equations of motion are affected up to term $\propto$ `h/tau.CM`.

  `tau.CP < 0` Langevin thermostat for the center of mass with the correlation time `-tau.CM` ps.

  `tau.CP > 0` Maxwell–Boltzmann thermostat for the center of mass with a new CoM velocity drawn every `tau.CM` ps.

  Added in `cook` V3.6s. NOTES/CAVETATS/BUGS:

  - Static quantities in the NVT ensemble are not affected (in the $h \to 0$ and `tau.CM`$\to 0$). All velocities contain the following correction term added to all particles at time t:
  $$\Delta v_{\mathrm{CM}} = \sqrt{k_{\mathrm{B}}T/M}\,\xi(t) \tag{9.13}$$
  Here, $\xi(t)$ is a time-correlated normalized Gaussian random number evolving according to `tau.CM`.

  - Quantities in the NPT (MTK) ensemble have now the additional measure

  - Cannot be used when the periodic b.c. are violated by walls (of any type), fixed atoms etc.

  - The interaction with other parameters as `drift`, `conserved`, `corr&16` is not automatically resolved and is left to user's responsibility. Some errors and warnings are printed, though.

  - Implemented for Verlet/leap-frog only.

`tau.E (0)` The correlation time to keep the energy (Hamiltonian, extended Hamiltonian, or the enthalpy) constant, in ps. `1/tau.E` is the friction coefficient. `tau.E=0` turns off the function (infinitive correlation time or zero friction coefficient). Cannot be combined with any thermostat. Implemented via velocity rescaling once a cycle.

`tau.i (0)`

`tau.j (0)` Atom type indices (numbered from 0) for for Berendsen-like Lennard-Jones σ_{ij} (or similar) adjustment, see Sect. 12.3.4.

`tau.L (0)` For SLAB & 1 #defined: Berendsen-style correlation time to maintain the values of x and y box sizes (see `box.Lx[]`, `box.Ly[]`, and see Sect. 14.10.5).

`tau.P (0)` Time constant for a barostat, cannot be combined with `tau.rho`. The configurational pressure used in the barostat is given by variable `virial`, see there (usually, the default works fine). See also variables `dV` and `corr`.

  `tau.P=0` Isochoric ensemble.

> `tau.P>0` Constant pressure.
>
> > `thermostat="NPT","MTK","Hoover","HooverNPT"` The approximated oscillation time of the extended-Lagrangian-type barostat.
> >
> > `Other thermostats` The exponential correlation time of the Berendsen friction-style barostat (see Sect. 12.3). The Berendsen-like barostat uses the ideal-gas bulk modulus unless `bulkmodulus` is set (see there). The box is rescaled before every step, the scaling factor is calculated at the end of the preceding cycle and is kept the same for all steps in the cycle (some bias is introduced, cf. `tau.P<0` below). Variable `noint` should not be too large (let us say, max. 10).
>
> `tau.P<0` Legacy barostat.
>
> > `thermostat="NPT","MTK","Hoover","HooverNPT"` Not supported.
> >
> > `Other thermostats` The box is rescaled at the end of every cycle using the current value of virial pressure (this is the old version). Interaction with `dV` is unknown (= not recommended unless checked).
>
> With `PRESSURETENSOR=3` (or more) and appropriate value of `rescale`), applies to diagonal components of the pressure tensor independently in each coordinate (suitable for crystals). Virtual volume/area is not supported.

`tau.rho (0)`   `tau.rho=0` The box size (density) does not change.

> `tau.rho>0` The correlation time to reach the desired density (given by `rho`) or box sizes (`rho=0` and `L[]`; see `rho` and `L[]` for details). In case of initialization the initial density is `initrho`. Time needed to reach `rho` is `abs(ln(rho/initrho))*tau.rho`. The number of cycles needed is then `no=1+abs(ln(rho/initrho))*tau.rho/(noint*h)`; this formula can be directly written to input data.
>
> `tau.rho=-1` The box size is controlled by file simname.`box`, which contains two columns: time in ps and the density in $\mathrm{kg\,m^{-3}}$, see Sect. 12.3.3.
>
> `tau.rho=-3` The box size is controlled by file simname.`box`, which contains four columns: time in ps and the box sizes in Å, see Sect. 12.3.3.
>
> NOTE: column 4 of the cp-file shows the density.

`tau.sat (0)` Turns on the module for automatic calculation of the electrostatic field `el.E` (if one component set to a positive value) or the dielectric constant of the surrounding continuum (if `el.E=0`) $\epsilon'_r =$`el.epsinf`, see Sect. 29.3.

> `tau.sat=0` Autoset is off.
>
> `tau.sat > 0` The correlation time to reach the desired saturation `el.sat` of the cell dipole moment.
>
> `tau.sat < 0` As above with |`tau.sat`|; in addition, after the sweep is finished, the averaged value of `el.epsinf` (if no field) or `el.E` (if field) is assigned, `tau.sat` is cleared, and `init=1` is set (instead of usual init=0). Note that you must first perform at least one (shorter) sweep with `tau.sat`>0 and then one (longer) with `tau.sat`<0; just starting (`init=2`) a simulation with `tau.sat`<0 will cause convergence profile error (missing header).

(Warning if there are free ions)

`tau.sig (0)` Correlation time for Berendsen-like Lennard-Jones σ (or similar) adjustment, see Sect. 12.3.4.

`tau.T (0)` The correlation time for the thermostat. See `thermostat` and see Sect. 12.2. Recommended values are 0.2–5 ps, in most cases 0.5–1. For simulated annealing (minimizing energy as $T \to 0$), much longer times are needed to cool slowly to escape from local minima. However, close to the minimum, shorter times (comparable to the periods of typical motions) are more efficient; e.g., for water ice, the fastest relaxation to zero is obtained with `tau.T=0.12`.

For extreme rates of cooling or heating using friction thermostat, the friction term is bounded by `+-1/tau.T`. Therefore `T=0` can be safely used.

`thermostat` The selection of thermostat. See Sect. 12.2.

   `0 "none"` No thermostat—the MD NVE ensemble.

   `1 "friction"`

   `1 "Berendsen"` Simple friction thermostat, see variables `tau.T` and `T`.

   `2 "Nose" "Nose-Hoover"` The Nosé–Hoover canonical ensemble.

   `3 "Andersen"` Velocities of randomly chosen atoms are replaced by those drawn from the Maxwell–Boltzmann distribution[15]. The mean time of updating the same atom is `tau.T`. Not good with constrained dynamics. The averaged kinetic temperature difers from `T` by an error of the order of $\mathcal{O}(h^2)$ unless VERLET=3 version is used. Issues have been reported (see `history.txt`).

   `4 "Maxwell"` Velocities of all atoms are replaced by those drawn from the Maxwell–Boltzmann distribution in the regular period of `tau.T`. The same conditions as above.

   `5 "AndersenCM"` Velocities of randomly chosen molecules (their centers of mass) are replaced by those drawn from the Maxwell–Boltzmann distribution. The mean time of updating the same atom is `tau.T`. May be used with constrained dynamics. Note that the actual relaxation time is longer because energy must be equipartioned with internal degrees of freedom. Not suitable for fast initial cooling/heating. The averaged kinetic temperature difers from `T` by an error of the order of $\mathcal{O}(h^2)$ (irrespective of the VERLET version); this is the consequence of equipartition error proportional to $\mathcal{O}(h^2)$. Issues have been reported (see `history.txt`).

   `6 "MaxwellCM"` As above, in regular period of `tau.T`.

   `7 "Langevin"` Poor Langevin thermostat for atoms, Gear only. Not good with constrained dynamics. Perturbs the Gear predictor and leads to equipartition errors (and averaged kinetic temperature error) on the order of $\mathcal{O}(h)$. Should be used with long enough `tau.T` only. This problem is solved[16], but not implemented here.

   `8 "LangevinCM"` Poor Langevin thermostat for molecules (centers of mass), Gear only. Can be used with constrained dynamics. Perturbs the Gear predictor, therefore only long enough `tau.T` are acceptable.

   `9 "Bussi"` CSVR (Canonical Sampling through Velocity Rescaling)[17], for Verlet only. (Inefficient implementation, tested for Ar only.) Variance (T) gives error on the order of $\mathcal{O}(h)$. I do not know whether a fix (like for Langevin) exists here.

   `11 "BerendsenCM"` The friction thermostat applies for the translational motions of the centers of mass only.

12 `"BerendsenIN"` The friction thermostat applies for the intramolecular (vibrational+rotational) motions only.

13 `"frictions"` The translational and intramolecular+rotational degrees of freedom are thermostated separately. See also `T_tr_in`.

22 `"MTK"` NPT ensemble: thermostat + barostat [8], via velocity predictor (see Nosé–Hoover) and box predictor (for constraints). Also `"NPT"`. See Sect. 12.3

23 `"Hoover"` NPT ensemble: thermostat + barostat [8], via velocity predictor (see Nosé–Hoover) and box predictor (for constraints). Also `"HooverNPT"`. See Sect. 12.3

See also variable `corr`: NPT requires `corr&16` unset. The default is correct, however, it does not change automatically if you also change the thermostat!

`Tstop (0)` If nonzero, the simulation stops when the kinetic temperature reaches `Tstop` (whether while cooling or heating)

`T_tr_in (1)` Applies to `thermostat=13="frictions"` only. The translational and intramolecular+rotational temperatures to be kept constant are different with ratio `Ttr/Tin=T_tr_in`. The total kinetic temperature is kept to `T`.

`unsplit (0)` If the input molecules (e.g., from `init="plb"`) are split over the periodic boundary conditions, they are normalized to the MACSIMUS style (periodic b.c. do not apply within a molecule). This un-splitting is done by the coordinates indicated by bits (1=x,2=y,4=z); use `unsplit=7` for all coordinates.

`virial (0)` Formula for the (configurational) pressure to be used for other purposes, as in NPT. Other flavors of pressure are reported.

`virial=0` Automatic selection of a suitable version.

`virial=1` Symbol `P(vir)`: The virial of electrostatic force is taken from minus the electrostatic energy. This is theoretically exact for point charges (with the Ewald summation and zero cutoff errors), in practice it is usually less accurate but a bit faster (if full pressure tensor is not calculated). Not suitable for Gaussian charges, quite inaccurate for cutoff electrostatics.

`virial=2` Symbol `P(dV)`: Pressure is calculated by the virtual volume change method. Variables `dV`, and `rescale` must be set appropriately.

`virial=3` Symbol `P(tens)`: Pressure is calculated from the trace of the full pressure tensor. This is most accurate.

`vshift[*]` Velocities of the whole configuration (or a part of it – see `nshift`) are changed by this 3D vector (before the first MD step). The vector is given in Å/ps = 100 m/s. This vector is then set to zero so that no push is performed in the next data set (unless specified again).

`xs.freq (0)` How often to measure the cross section (CHANGED in V2.7o).

0   Do not measure.

1   Measure every cycle (of `noint` steps) during simulating. Measure for every frame in the reread playback mode,

> > 1 Measure every `xs.freq`-th cycle during simulating (data in the convergence profile will repeat the last calculate value). Not recommended in the reread playback mode (use `reread.by` instead and `xs.freq=1`).

> < 0 Measure every cycle |`xs.freq`|-times. Makes sense with modes using random orientations, `xs.mode=0,1,4`. More samples = higher precision.

`xs.grid (10)` Grid for calculating the cross-section, in number of points per 1 Å. The method places the molecule on a rectangular mesh, projects all atoms and counts the number of mesh points. Note that there is a limit for the size of this mesh (see `xs.sizelimit`), the default is enough with `xs.grid=10` for most cases but choosing a finer mesh could cause exceeding this limit.

`xs.maxs (-1)` Molecule mode: number of SPECIES included.
Whole configuration mode (`xs.mode&8`): number of SITES included.
0=all SPECIES or SPECIES
Does not apply for clusters (`xs.mode&16`): see `xs.mincluster`.

`xs.mincluster (0)` CLUSTERS only: Minimum cluster (number of molecules) for which cross-section will be calculated.

`xs.mode (0)` Only if compiled with #define XSECTION: method of calculating the cross sections.

> 0 `"gauss"` Uses 16 sample projection directions for each measurement, corresponding to 32 of the vertices of a regular dodecahedron and inscribed icosahedron with weights giving the Gaussian integration formula[5] The orientation of the polyhedron is random.

> 1 `"one"` One random projection direction per measurement.

> 2 `"shear"` Three directions measured separately: $(x + y)$, $(x - y)$, and z. Suitable for testing possible orientational order caused by shear stress.

> 3 `"xyz"` Three directions measured separately: x, y, z. Suitable for monitoring melting a crystal and similar.

> 4 `"three"` Three sample orthogonal projection directions for each measurement, rotated by a random orientational matrix as in `xs.mode=0`.

> 5 `"x"` Direction x measured separately.

> 6 `"y"` Direction y measured separately.

> 7 `"z"` Direction z measured separately.

> 8 If combined with the above: The cross-section will be calculated for the whole configuration (otherwise for all molecules separately, which is not what you want for a water cluster).
>
> Use uppercase mnemonic codes: `"GAUSS"`, `"ONE"`, `"SHEAR"`, `"XYZ"`, `"THREE"`, `"X"`, `"Y"`, `"Z"`. (`"SHEAR"`, `"XYZ"` do not have too much sense.)

> 16 CLUSTERS only: calculate cross-section for clusters. Exclusive with bit 8. In this mode, clusters $\geq$ `xs.mincluster` are analyzed and their cross-sections (in Å^2) printed. The maximum cluster cross-section (by the number of molecules) is recorded in variable `Xsection[max.cluster]` and in column Xsec of convergence profile.

---

[5]probably from book A.H. Stroud, The Approximate Calculation of Multiple Integrals, (Prentice Hall, Englewood Cliffs, New Jersey, 1971), see gen/sphint.c

Note that modes 0,1,4 give the angle-averaged cross-section while modes 2,3 give cross-sections in specified directions.

`xs.Rscale (2^(-1/6))` For cross section measurements, the atoms are represented by spheres of radii given by `xs.Rscale*(RvdW+xs.Rvdw)`, where `RvdW` is the van der Waals radius of the atom being measured and xs.RvdW of the testing particle in a scattering experiment. The default is the same as using half Lennard-Jones sigma. (NOTE: old name = `xs.scaleR`).

`xs.RvdW (1.4)` Van der Waals radius of the testing particle in a scattering experiment. 1.4 corresponds to Helium.

`xs.sizelimit (`2^{28}`)` Max number of bytes that can be used for the cross-section grid. (This is the upper limit, only the necessary memory is allocated.) If exceeded, a warning is printed and outer parts of the molecule are ignored; the parts which are farthest from the origin are more likely to be cut off. Thus, if this happens for one molecule in free boundary conditions (FREEBC) which contains parts which can be released and fly away (like water molecules), these water molecules will be simply ignored. Cf. `AllocSizeLim`.

`wall.g (0)` Gravity in the direction of $\hat{z}$, in the unit of $\text{Å}/\text{ps}^2$ (internal program units, see `sim/units.h`). Note that the "gravity" is usually negative (lower potential for smaller values of z). May be useful for initializing the adsorbtion of molecules on a wall. Practical range is up to 10 with intensive thermostatting, otherwise much less.

`wall.LJ[nsites].sig`

`wall.LJ[nsites].neps` Optional replacement of the wall–atom Lennard-Jones (12–6) parameters (the wall-atom potential is integrated 9–3). Must be in the simname`.get` file (not simname`.def`.

> `wall.LJ[i].sig` [Å] is the Lennard-Jones (combined) σ of the "wall atom" vs. atom number `i` (i goes from 1 to `nsites-1`, see sysname`.ble`).

> `wall.LJ[i].neps` [k_{B} K Å^{-3}] is the Lennard-Jones (combined) ϵ multiplied by the wall number density (in Å^{-3}).

NOTE: As the default, the wall–atom Lennard-Jones parameters are determined from the respective (not necessarily Lennard-Jones) atom–atom potential. The last atom in the "Lennard-Jones" or "Buckingham" or "Busing" table is the material of the walls.

`wall.n (0)` The walls. Sum of flags and sign:

> `|wall.n|&1` Wall 0 at $z = $ `wall.z[0]`L_z.

> `|wall.n|&2` Wall 1 at $z = $ `wall.z[1]`L_z.

> `wall.n>0` Walls are repulsive (the attractive part of the Lennard-Jones potential is set to zero, only a z^{-9} term is used.

> `wall.n<0` Walls are attractive (full Lennard-Jones potential).

Example:

```
rho=0 z=60 wall.n=-1 wall.z[0]=1/z L[2]=z
```

defines one bottom ($z = 1$ Å) attractive wall in a box with $L_z = 60$ Å.

`wall.rho (19320)` Mass density of wall atoms, in kg/m^3 (the default is Au).

`wall.z[2] (0,1)` The positions of the respective walls in the units of z-box size. `wall.z[0]<wall.z[1]` is required.

9.2.6   Interactive and batch control

`cook` runs in both batch and interactive modes. To turn the interactive mode on use option `-s`. This also enables scrolling (if compiled so).

The 'get data' input module, the use of scrolling and error handling are the same as in program `blend` – see the manual for `blend`!

9.2.7   Interrupt

If not turned off by option `-i-1`, pressing Ctrl-C (or signalling `kill -2` or `kill -INT`, on some systems also `kill -15` or `kill -TERM`) causes the program to be interrupted and you can select (in interactive mode and a serial version from keyboard, in PARALLEL version and/or batch mode by option `-i`):

`0 = (r)esume:  continue calculations` Continue running as without interrupt.

`1 = (i)nterrupt:  finish cycle then save all and read next data` In interactive mode just the running cycle will be finished and you will be prompted for other data. Not so useful for the batch mode because the data are read from a file.

`2 = (.)stop:  finish cycle then save all and stop` Stop `cook` gracefully (both in interactive and batch modes). Default for the batch mode.

`5 = (j)ump to stop:  finish step then save all and stop` (incomplete cycle!) Stop `cook` less gracefully. The configuration will be saved but measurements will not be recorded in proper intervals (one per a cycle). To be used if you want to start `cook` again with `init=2`.

`-1= (e)xit immediately (nothing saved!)` Kill `cook`.

`9 = (s)croll (then type ?  for help)` Enter scrolling; once prompt `$` appears, type a scroll command or `?` for help.

NOTE: You can disable the Ctrl-C handler totally by deleting `#define SIG` from `main.c`. See comments there!

Chapter 10

Parallelization

MPI support was removed at cook V2.5a. Now there are two shared memory versions implemented using POSIX threads ("pthreads").

10.1 Compiling

The version is determined by compile-time switch PARALLEL (`#define PARALLEL`) in `simopt.h`. Several options are possible (see below). The pthread library is linked by option `-lpthread`.

Script `configure.sh` takes care about all settings.

10.2 Running

The number of threads is given by the environment variable NSLOTS. See more below.

If cook is submitted using SGE (PBS) command `qsub`, the number of requested cores ("slots") is specified by parameter `-pe shm NUMBER`; it is passed in the form of environment variable NSLOTS to the node. Example:

```
qsub -cwd -b y -q sq-8-16 -pe shm 2 ./cookewslcp0P2 polwater slab -t
```

10.3 Linked-cell list and Ewald parallelized

#define PARALLEL 1 : This version is suitable for simulation of large periodic systems.

The Ewald k-space part is parallelized in the natural way by splitting the sums over charged sites into parallel threads.

The linked cell list method is parallelized by slabs (2D arrays of cells) in the x-direction. There are `No.cell[0]` such slabs; it should be an integer multiple of the number of threads, otherwise a bad load-balancing occurs.

If you are running cook on your own machine, it may be advantageous to specify `No.lcth`. It will split the Ewald k-space part into given number of threads while the r-space sums into `No.cell[0]` threads (more than processors).

10.4   Ewald k-space and r-space running in parallel

#define PARALLEL 2 : This version is suitable for smaller systems. It uses two processors, in one the Ewald k-space part is calculated and in the other and the other the r-space part (of Ewald plus Lennard-Jones). It is good to set the Ewald parameters (in the standard situation for `el.test=-10` this means trying several values of `cutoff`) so that the time spent by both parts is the same, otherwise load-balancing will be lost. The r-space part may be either all-pair or linked-cell.

10.5   Pair sums for a single big molecule parallelized

WARNING: not implemented now

#define PARALLEL 3 : This version is suitable especially for 1 big macromolecule in free boundary conditions (version FREEBC). All pairs in the pair-sum are considered and parallelized as in the following scheme:

```
-----------------------------------------
 1-0

 2-0 2-1
                            Thread 1
 3-0 3-1 3-2

 4-0 4-1 4-2 4-3
-----------------------------------------
 5-0 5-1 5-2 5-3 5-4
                            Thread 2
 6-0 6-1 6-2 6-3 6-4 6-5
-----------------------------------------
  7-0 7-1 7-2 7-3 7-4 7-5 7-6  Thread 3
-----------------------------------------
```

The amount of "lines" is determined in such a way that the amount of work is approximately the same for each thread. The algorithm allocates a copy of forces (though not of the full length in all cases) for each thread.

BUG: Measurements of radial distribution functions and dihedral angle distributions are not supported.

Chapter 11

Algorithms and parameters

There are many parameters that control the accuracy end efficiency. Most of them have reasonable defaults that will work at least fairly well in most typical cases. But one is never watchful enough. You must be able to recognize that something is getting wrong, so do not skip this chapter completely!

11.1 Accuracy

From the technical point of view, there are four sources of inaccuracies in the generated configurations:

- Integration errors. They are controlled by the integration method used (option `-m`) and the timestep `h`.

- Cutoff errors. These (in periodic b.c.) consist of the site–site (Lennard-Jones) cutoff errors and electrostatic cutoff errors. For Ewald summation, the latter consist of the k-space and r-space errors.

- Constraint errors. These are caused by different phenomena and must be corrected to some low value.

- Inaccurate calculation of charge–charge interactions (r-space sums). Normally negligible.

- Inaccurate ensemble. E.g., serious problems may arise when the friction thermostat with short correlation time is used for small molecules (they will rotate faster than corresponds to the temperature).

From the physical point of view, there may be many reasons why things go wrong. A few typical reasons follow:

- Long correlation times of crossing large energy barriers. For instance a convergence to equilibrium between cis and trans conformations may be slow.

- Small coupling between different degrees of freedom, e.g., slow coupling between fast bond vibrations and translations/rotations. That is why constrained bonds may be more accurate/efficient than simulating a fully flexible model. A slow convergence of the Nosé–Hoover thermostat is of similar nature.

- Finite size effects. Especially important for systems close to a critical point. Increasing system size not only slows down the simulation algorithm but also increases the correlation times.

The following quantities serve to observe inaccurate sampling of the phase space:

- Conservation of integrals of motion. The most important is the total energy (Hamiltonian), in free b.c. the angular momentum

- Translational and intramolecular (+rotational) temperatures (called Ttr and Tin) can reveal a possible wrong equipartition (e.g., the "flying ice cube problem" of the Berendsen thermostat)

- Errors of constraints

- (POLAR only) Errors in self-consistent field

11.1.1   Errors of constraints

The following quantities are available to observe the accuracy of the constraints:

`cerr.r1` Error of constraints of predictor; reported as "`constr err 1`" in statistics.

`cerr.r2` Error of constraints after integration step.

`cerr.r3` Error of constraints (of true configuration, i.e., after correcting constraints).

`cerr.v1` Error of velocity constraints of predictor; reported as "`v constr err 1`" in statistics.

`cerr.v2` Error of velocity constraints after integration step

`cerr.v3` Error of velocity constraints (of true configuration)

All these quantities are normalized to be dimensionless.

Generally, `cerr.r1` should be several times higher than `cerr.r2` which should be several times higher than `cerr.r3` and similarly for `cerr.v`'s (with the exception of `cerr.v1/cerr.v2` which is rather high anyway). If all `cerr.r`'s are comparable (ratios less than 2), the constraint dynamics is solved too inaccurately and `epsc` and/or `eps` should be decreased. If `cerr.r2/cerr.r3` or `cerr.v2/cerr.v3` is too high (say, > 100), `epsc` is unnecessarily low and efficiency is lost.

11.1.2   Energy conservation

The total energy should be in principle constant. The exception is the friction thermostat (and friction-like isobaric ensemble and similar) which spoils in principle the energy conservation – even if you wish to use the friction thermostat, try once a while a short microcanonical run to check the energy conservation! But even in the true microcanonical ensemble, various inaccuracies cause both statistical fluctuations and systematic secular drift in the total energy (the Hamiltonian).

1. Integration errors cause cooling (energy decrease) for the 4-value Gear method for 2nd-order equations which is the case of `cook` with the default value of option `-m`. In other cases they cause heating.

2. Cutoff errors (i.e., small jumps or peaks in the potentials and forces) cause heating (energy increase)

3. Inaccurate constraint dynamics causes heating

The first reason usually is (and should be) most important.

For the Nosé–Hoover canonical ensemble the drift does not matter because it means only some rescaling. The drift, of course, must not be too high: the good criterion is not the drift during the entire (possibly very long) simulation but the drift during a typical correlation time (say, 1 ps) which should be set according to the demanded accuracy. The same holds true for the statistical fluctuations of the total energy.

The situation is not so simple for the microcanonical ensemble where energy drift causes cooling or heating. To avoid this, set `tau.E` to a typical correlation time (at least 1ps) and energy `E` will be conserved; the difference of `E-<measured Etot>` as well as `sqrt(Var Etot)` (both quantities are measured and statistically analyzed) is then the equivalent measure of the quality of the simulations.

The problems are usually smaller for the Verlet integrator (with SHAKE for constraints) which is time reversible so that (if there are no other sources of errors) there is no energy drift. In addition, the Verlet integration can be treated as exact integration of a perturbed Hamiltonian so that the errors are bound. (To be exact, this holds true only for certain class of continuous potentials with derivatives. Typical potentials like Lennard-Jones with singularities for particle overlaps exhibit energy increase. This phenomenon is for typical timesteps usually negligible.)

11.1.3   Self-consistent field accuracy

Applies to POLAR version only. The reported errors, `selffield maxerr` (maximum found in the configuration) and `selffield stderr` (standard deviation) of induced dipole moments are in the program units (one program unit is 0.0117501 D, or see `sim/units.h` for details). See also the explanation of `scf.eps` variable. see Sect. 27.

11.2   How to set Ewald parameters α and κ

Both $\alpha =$ `el.alpha` and $\kappa =$ `el.kappa` depend on the r-space cutoff (`cutoff`). It must not exceed half the box size L/2 (to be precise, a tiny overflow is accepted; in addition `cutoff` may exceed half box if the linked cell list method is used, but this method becomes efficient for large systems so that using `cutoff>L/2` is not recommended. For small systems (say, number of sites < 1000), half the box size is the optimum value. For larger systems a lower value is optimum. If you select `cutoff<0` (this is the default), `cook` calculates a value that should work fairly well in typical cases (it uses the final density `rho` to calculate the final box size). But if you want to optimize the run, you should try several values of `cutoff` and measure the time.

11.2.1   Simple way

It follows from the formulas in Appendix 24 that the main term of the r-space error is erfc(αcutoff) $\propto \exp[-(\alpha\text{cutoff})^2]$ whereas the k-space error is $\exp[-(\pi\kappa/\alpha)^2]$. Let ϵ be the desired accuracy. Then the estimated parameters are

$$\alpha = \frac{\sqrt{-\ln\epsilon}}{\text{cutoff}}, \qquad \kappa = \frac{\alpha}{\pi}\sqrt{-\ln\epsilon}. \qquad (11.1)$$

Usually $\epsilon = \exp(-\pi^2) \approx 5.2 \times 10^{-5}$ is a reasonable choice. Then

$$\kappa = \alpha = \pi/\text{cutoff} \qquad (11.2)$$

Example of code:

```
el.test=0            ! no automatic setting of alpha,kappa
el.alpha=pi/cutoff   ! simple estimate
el.kappa=el.alpha    ! simple estimate
```

11.2.2   More accurate way

As the default (parameter `el.test=-10`), `cook` implements approximate formulas [11] based on the assumption that charges in the system are distributed randomly. The value of the r-space `cutoff` is here a free parameter and `el.alpha` and `el.kappa` are calculated to satisfy the accuracy requirements given by errors `el.epsr` and `el.epsk`.

But be aware that *approximate* error formulas are used. They tend to overestimate the k-space cutoff errors for large systems without free charges (i.e., if all charges are parts of small electroneutral groups). In addition, the k-space cutoff errors are both positive and negative and do not cause any violation of energy conservation. Hence, `el.epsk` may be several times higher than `el.epsr`.

11.2.3   Most accurate way

Since only approximate formulas are used for the cutoff errors, I recommend to check them against the actual errors. This is done in a special module that is entered by specifying `el.test` different from `0` and `-10`. The following data are available:

`el.alpha` As in the main get data module

`el.kappa` As in the main get data module

`cutoff` As in the main get data module; must be positive.

`eps` As in the main get data module; the original value is restored automatically when the Ewald testing module is left.

`el.epsk` As in the main get data module

`el.epsr` As in the main get data module

`el.grid` As in the main get data module

`el.test` Control:

> `el.test=0` Leaves Ewald testing module Ewaldtest and calculates (additional get data are not read, must be prepared before el.etest is invoked).
>
> `el.test=1` Calculates forces and energy and, if reference is set, errors.
>
> `el.test=2` Calculates forces and energy and store them as a reference for evaluating errors in next steps, sets `el.test=1`.
>
> `el.test=-1` Calculates `el.alpha` and `el.kappa` from `cutoff el.epsr el.epsk` and performs `el.test=1`.
>
> `el.test=-2` Calculates `el.alpha` and `el.kappa` from `cutoff el.epsr el.epsk`, performs `el.test=2` and sets `el.test=-1`.
>
> `el.test=-3` Calculates `el.alpha` and `el.kappa` from `cutoff el.epsr el.epsk` and waits for further data.
>
> `el.test=-10` Calculates `el.alpha` and `el.kappa` from `cutoff el.epsr el.epsk`, works silently and quits Ewaldtest afterwords.
>
> `el.test=-9` Exit `cook*` immediately.

`el.minqq` As in the main get data module

`No.cell` As in the main get data module

`x` Auxiliary variable

Testing should not be performed using the initial configuration which is very artificial – it should be at least a little bit condensed and equilibrated. An example follows:

```
eps=1e-9    ! sufficiently high accuracy for calculating the Lagrange
            !  multipliers (not to introduce other inaccuracies!)
epsc=1e-9   ! sufficiently high accuracy for calculating constraints
LJcutoff=6  ! cutoff in the module should not fall below cutoff
scf.eps=1e-8 scf.omega=0.95 ! if POLAR: iterate to high precision
el.test=-2 ;! switch to the Ewaldtest routine (step 7. of 'program flow'
            !   above) and select automatic setting of alpha and K
            !
el.epsr=1e-4! sufficiently high accuracy for real-space errors
el.epsk=1e-4! sufficiently high accuracy for $k$-space errors
el.test=-2; ! 1) alpha,kappa determined from epsr,epsk
            ! 2) accurate forces and energies are evaluated and stored as
            !     "reference" benchmark values
            !
el.epsr=.05 ! working accuracy
el.kappa=.5 ! working accuracy
el.test=-1; ! 1) alpha,kappa determined from epsr,epsk
            ! 2) errors (from the reference) calculated
            !
el.test=0;  ! abandon Ewaldtest; remember to return the changed variables!
```

11.3   Constraint dynamics

The constraint dynamics comes in two versions, one using Lagrange multipliers for calculating the constraint forces, and the SHAKE algorithm. The SHAKE algorithm is based on the second-order Verlet integration method while the multiplier method allows higher-order integrators (see option `-m`). The multiplier method requires a step correcting the constraints.

11.3.1   The SHAKE algorithm with Verlet integration

Several versions of SHAKE are available differing in efficiency only marginally.

`undefined` Shake is not implemented (only Gear+Lagrangian constraint dynamics is available).

`#define SHAKE 1` Simple update in sweeps. This is the default in `configure.sh`.

`#define SHAKE 2` Information on moved sites is kept and only the bonds between sites that have moved in the previous step are updated. May be slightly more efficient in some cases

`#define SHAKE -1`

`#define SHAKE -2` As above and a more complicated algorithm taking into account the angle between the old and new constraint is used. Some more tests are added. Might be slightly better for diatomics. Not tested recently.

The velocity algorithm comes in four compile-time flavors. The trajectory is the same and given by the Verlet algorithm with SHAKE. The velocity algorithm affects the kinetic energy and kinetic part of the pressure tensor. It does not affect the .vlb file which always corresponds to VERLET=0 (shifted by $h/2$). The differences in speed are small.

`#define VERLET 0` The simplest, fastest, and least accurate $O(h)$ version, $v(t) = [r(t+h) - r(t)]/h$. Essentially, velocity is shifted by $h/2$ from positions. Correct averaged energy of the harmonic oscillator. Good enough with the friction (Berendsen) thermostat. (Added in V2.4a)

`#define VERLET 1` Compromised speed, velocity accurate to $O(h^2)$: $v(t) = [r(t+h) - r(t-h)]/(2h)$. Averaged energy of harmonic oscillator has error $O(h^2)$. (Added in V2.4a) This is equivalent to the velocity Verlet algorithm.

`#define VERLET 2` Best energy conservation (exact for harmonic oscillator but with $O(h^2)$ error), slowest, $v(t)^2 = [r(t) - r(t-h)]/h \cdot [r(t+h) - r(t)]/h$. (The off-diagonal components of the pressure tensor are approximated as average of both possible $h/2$-shifted terms).

`#define VERLET 3` Compromised speed, velocity and energy conservation accurate to $O(h^2)$ (slightly worse than for VERLET=1), but with correct averaged energy of harmonic oscillator. Energy and pressure tensor components are averages of both shifted values $v(t) = [r(t+h) - r(t)]/h$ and $v(t) = [r(t) - r(t-h)]/h$. Many quantities are more accurate then for VERLET=1. This is the recommended default.

`#define VERLET 4` With Nosé-Hoover thermostat only: velocity and energy from the TRVP predictor. Not included in the distribution.

`#define VERLET 5` Mixed, best equipartion for a rotating diatomics (?).

`#define VERLET 6` On average equivalent to Beeman up to $O(h^4)$.

`#define VERLET 30` Energy=3/pressure components mixed, not included in the distribution.

SHAKE is controlled by two parameters, `omegac` and `epsc`. The first one is the relaxation (mixing iteration) parameter: `omegac=1` means direct iterations, `omegac>1` overrelaxation, and `omegac<1` underrelaxation (mixing with the previous iteration). In most cases, values around `omegac=1.2` are optimum. ANCHOR version (see Sect. 15.12) may require `omegac<1`. Parameter `epsc` is the required relative accuracy of constrained bonds.

Since V2.4a, a negative value of `omegac` causes automatic determinantion of the optimum `omegac`, separately for each species. The optimization starts from the absolute value, but it is not stored (is done again even if `init=0` is selected). Use option `-v&4` to get detailed protocol on the process of optimization.

The numbers of needed SHAKE iterations are recorded and statistically analyzed in variable `nitc` and reported as `corr constr it` separately for the 1st molecule and averaged rest.

11.3.2   Constraint dynamics with Gear integrators

An alternative to SHAKE is the Lagrangian formulation[1] of constrained dynamics [18] implemented with the Gear integrators. The algorithm follows; see also the comments in `constrd.c`. The Newton equations of motion are modified by adding the constrained forces:

$$\ddot{\vec{r}}_i = \frac{1}{m_i}(\vec{f}_i + \vec{f}_i^{\,c}). \tag{11.3}$$

The constrained forces are in the direction of constraints,

$$\vec{f}_i^{\,c} = \sum_a g_a \frac{\partial c_a}{\partial \vec{r}_i}, \tag{11.4}$$

where c_a denotes the a-th constraint. For a bond constraint between atoms i_a and j_a it holds

$$c_a = \frac{1}{2}[(\vec{r}_{i_a} - \vec{r}_{j_a})^2 - l_a^2], \tag{11.5}$$

where l_a is the bond length. In order to write equations for the unknown multipliers g_a, let us take the second time derivative of c_a's:

$$\dot{c}_a = \sum_i \dot{\vec{r}}_i \cdot \frac{\partial c_a}{\partial \vec{r}_i}$$

$$\ddot{c}_a = \sum_i \ddot{\vec{r}}_i \cdot \frac{\partial c_a}{\partial \vec{r}_i} + \sum_{i,j} \dot{\vec{r}}_i \cdot \frac{\partial^2 c_a}{\partial \vec{r}_i \partial \vec{r}_j} \cdot \dot{\vec{r}}_j.$$

The constraints are fixed, so $\ddot{c}_a = 0$. By inserting $\ddot{\vec{r}}_i$ of (11.3) into the above $\ddot{c}_a$, we arrive at equation (in matrix form)

$$Mg + G = 0, \tag{11.6}$$

---

[1]The older versions of `cook` supported also the Hamiltonian formulation, however, this procedure suffered from growing momenta.

where M is an $n_c \times n_c$ matrix ($n_c =$ No.c is the number of constraints)

$$M_{ab} = \sum_i \frac{1}{m_i} \frac{\partial c_a}{\partial \vec{r}_i} \cdot \frac{\partial c_b}{\partial \vec{r}_i}, \tag{11.7}$$

$$G_a = \sum_i \frac{\vec{f}_i}{m_i} \cdot \frac{\partial c_a}{\partial \vec{r}_i} + \sum_{i,j} \dot{\vec{r}}_i \cdot \frac{\partial^2 c_a}{\partial \vec{r}_i \partial \vec{r}_j} \cdot \dot{\vec{r}}_j. \tag{11.8}$$

To evaluate the above formulas for bond constraints (11.5), we easily calculate

$$\frac{\partial c_a}{\partial \vec{r}_i} = (\vec{r}_{i_a} - \vec{r}_{j_a})(\delta_{i,i_a} - \delta_{i,j_a}). \tag{11.9}$$

The diagonal elements are:

$$M_{aa} = \left(\frac{1}{m_{i_a}} + \frac{1}{m_{j_a}} \right) (\vec{r}_{i_a} - \vec{r}_{j_a})^2. \tag{11.10}$$

A nondiagonal element is nonzero only if constraints c_a and c_b share a common site; for bond constraints, at most one such site exists. E.g., if $i_a = i_b$ then

$$M_{ab} = \frac{1}{m_{i_a}} (\vec{r}_{i_a} - \vec{r}_{j_a}) \cdot (\vec{r}_{i_a} - \vec{r}_{j_b}). \tag{11.11}$$

Matrix M is sparse and coded as a list (type M_t in simglob.h). The set of linear equations is solved by the conjugate gradient method[2]; the multipliers g are predicted between time steps. Matrix M is also needed for normal modes calculation, See Sect. 14.9.2.

11.3.3   Constraint forces by Lagrange multipliers

The conjugate gradient method is used to calculate Lagrange multipliers and then the constraint forces. The algorithm is described in constrd.c. The method is controlled by one parameter, eps, which is the maximum relative (dimensionless) error. If eps is too high, you will get into troubles with correcting constraints (see Sect. 11.1.1).

Another guide to optimization is the number of iterations: nit, or Lagr mult it, (the number of iterations of the conjugate gradient method needed to calculate the Lagrange multipliers) should not exceed (too much) the number of bonds; if it does, eps is unnecessarily small (or there is something seriously wrong like bad cutoffs etc.).

11.3.4   Correcting constraints

Integration and cutoff errors and inaccurately calculated Lagrange multipliers cause errors in constraints. It is necessary to correct these errors, otherwise they cumulate and the algorithm explodes. The constraint errors are—and must be—corrected every integration step.

The correction to be added to particle positions is

$$\Delta \vec{r}_i = \sum_a \epsilon_a \frac{1}{m_i} \frac{\partial c_a}{\partial \vec{r}_i} \tag{11.12}$$

---

[2]Optionally, Cholesky preconditioning can be turned on by #define PRECOND, with a marginal impact on efficiency

where ϵ_a solves equation

$$M\epsilon + E = 0, \quad E_a = c_a \tag{11.13}$$

Similarly, the velocities are corrected by adding the following velocity components perpendicular to the constraints

$$\Delta \vec{v}_i = \sum_a \nu_a \frac{1}{m_i} \frac{\partial c_a}{\partial \vec{r}_i} \tag{11.14}$$

where ν_a solves equation

$$M\nu + V = 0, \quad V_a = \sum_i \dot{\vec{r}}_i \cdot \frac{\partial c_a}{\partial \vec{r}_i} \tag{11.15}$$

Two methods are available to correct the constraints.

`Scorrect` is a SHAKE-like iteration method. The inverse-mass weighting ensures that the total momentum is conserved (within machine precision). This method is selected by unset bit 1 in option `-c` (was `-c3` prior V2.4a). It is controlled by two parameters, `epsc` and `omegac`. `epsc` is the maximum relative error of the constraints and `omegac` is the relaxation parameter: `omegac=1` are pure SHAKE-like iterations. Usually, small overrelaxation (`omegac=1.1-1.2`) is more efficient. The number of iterations is recorded and printed (separately for the 1st molecule and average for the rest). `Scorrect` is more efficient than the following method for simple constraints provided that good values of the parameters are set. It may become inefficient for molecules with many small cycles and/or bond angle constraints.

`conjgrad` uses the same conjugate gradient method and the same (inverse-mass weighted) matrix that has been used for calculating Lagrange multipliers for the constraint dynamics, hence, the total momentum is conserved as above. The method is selected by bit 1 in option `-c` (`-c4` prior V2.4a). It works fairly well for both simple and complex constraints. It is controlled by one parameter `epsc` and it seems that the value `epsc=0.05` (default) is OK if the order of multiplier predictor is the same as of the method (see options `-m` and `-p`). However, it requires more memory than `Scorrect` above.

For as complex systems as proteins a pessimistic choice of `eps`, `epsc`, or `omegac` affects the efficiency only slightly because calculating the force field is most time consuming anyway. See the discussion on `cerr` above.

Note that for `conjgrad` the conjugate gradient method in the Lagrange formalism is called twice (once for length constraints and once for velocity constraints).

For `Scorrect`, the number of iterations depends on `omegac`. If it is too high, `conjgrad` should be used instead of `Scorrect`.

Summary:

- `eps` about 1e-5..1e-6 should work (for `-m4` and `-p4`)

- use `Scorrect`, `epsc=eps` and `omegac=1.2` for simple molecules

- use `conjgrad` and `epsc=0.05` for complex molecules (`omegac` is irrelevant)

- `conjgrad` requires more memory than `Scorrect`. If `Scorrect` is to be used also for complex molecules, it is recommended to adjust `omegac` to get the minimum value of the number of iterations. (This is automated since V2.4a)

11.3.5   Dependants

A dependant is a massless site which is calculated from other sites (parents). The configuration of the parents must be rigid (constrained). Forces on dependants are distributed back to parents before constraint dynamics.

There are two types of dependants:

M   "Middle" dependants are a linear combination of the parents. The forces are distributed back using the same coefficients. Example: site M in the TIP4P water model.

L   (New in cook V2.5f): "Lone" dependants are based on three atoms, but they lie out of the plane defined by these three atoms. Example: site L in ST2 or NE6 water models.

There may be several mechanically separated dependants (of both types) in one molecule.

Data for the dependant mechanics are collected in table "dependant" in the ble-file. The "Middle" dependants are calculated by blend. For water models, utility `waterdep` is available.

Lone dependant mechanics: Rigid reference

A position of this "Lone"-type dependant is derived from a rigid triangle (for water HOH). The lenghts and angles must be constrained! The dependant may be in any relative position to this fixed triangle.

Let us denote the positions of the three parents as $\vec{r}_a$, $a = 1, 2, 3$. The calculations need a local orthonormal coordinate system $(\hat{x}, \hat{y}, \hat{z})$, where vectors $\hat{x}, \hat{y}$ are in the plane of the parents and $\hat{z}$ is perpendicular:

$$
\begin{aligned}
\hat{x} &= \sum_a x_a \vec{r}_a \qquad (\sum_a x_a = 0), \\
\hat{y} &= \sum_a y_a \vec{r}_a \qquad (\sum_a y_a = 0), \\
\hat{z} &= \hat{x} \times \hat{y}.
\end{aligned}
$$

Constants x_a, y_a are defined in the ble-file, table "dependants". Particularly for water (models POL4D, NE6) it holds (as calculated by `waterdep`):

$$
x_L = -1/c, \quad x_{H_1} = 0.5/c, \quad x_{H_2} = 0.5/c \quad \text{for} \quad c = |OH|\cos(HOH/2)
$$
$$
y_L = 0, \quad x_{H_1} = 0.5/c, \quad x_{H_2} = -0.5/c \quad \text{for} \quad c = |OH|\sin(HOH/2)
$$

The dependant position is

$$
\vec{r} = \sum_a w_a \vec{r}_a + w_z \hat{z} \qquad (\sum_a w_a = 1) \tag{11.16}
$$

where again w_a and w_z are defined in the table "dependants". Particularly for water it holds:

$$
\begin{aligned}
w_z &= \pm |LO| \sin(LOL/2) \\
w_{H_{1,2}} &= \frac{-|LO|\cos(LOL/2)}{2|OH|\cos(HOH/2)} \\
w_L &= 1 - 2w_{H_{1,2}}
\end{aligned}
$$

To distribute force f on the dependant back to the parents, we first calculate its components in the local coordinate system,

$$f_x = \hat{x} \cdot \vec{f}, \quad f_y = \hat{y} \cdot \vec{f}, \quad f_z = \hat{z} \cdot \vec{f}. \tag{11.17}$$

The force on parent a, $a = 1, 2, 3$, is

$$\vec{f}_a = w_a \vec{f} + (t_{x,a} f_x + t_{y,a} f_y)\hat{z}. \tag{11.18}$$

The 'torque constants' $t_{x,a}$, $t_{y,a}$ are again defined in the table "dependants". For water:

$$t_{x,\mathrm{L}} = \frac{\pm|\mathrm{LO}|\sin(\mathrm{LOL}/2)}{|\mathrm{OH}|\cos(\mathrm{HOH}/2)}, \quad t_{x,\mathrm{H}_{1,2}} = -t_{x,\mathrm{L}}/2$$

$$t_{y,\mathrm{L}} = 0, \quad t_{y,\mathrm{H}_1} = \frac{\pm|\mathrm{LO}|\sin(\mathrm{LOL}/2)}{2|\mathrm{OH}|\sin(\mathrm{HOH}/2)}, \quad t_{y,\mathrm{H}_2} = \frac{\mp|\mathrm{LO}|\sin(\mathrm{LOL}/2)}{2|\mathrm{OH}|\sin(\mathrm{HOH}/2)}$$

Pressure tensor calculation The force-calculating module also calculates the virial of force and optionally also components of the pressure tensor, see Sect. 15.7. For the "Lone" dependants, a correction must be made because of force redistribution,

$$V\Delta\overset{\leftrightarrow}{P} = \sum_a \vec{f}_a(\vec{r}_a - \vec{r}) = \sum_a (t_{x,a} f_x + t_{y,a} f_y)\hat{z}(\vec{r}_a - \vec{r}) - w_z \vec{f}\hat{z} \tag{11.19}$$

where we used (11.16) to simplify the expression. The virial of force correction is the trace,

$$V\mathrm{Tr}\Delta\overset{\leftrightarrow}{P} = \sum_a (t_{x,a} f_x + t_{y,a} f_y)\hat{z} \cdot (\vec{r}_a - \vec{r}) - w_z f_z = -w_z f_z \tag{11.20}$$

because the torque part (in the sum) has a zero trace.

Note that the pressure tensor corrections are zero for the "Middle" dependants.

Rowlinson-type dependant mechanics: Flexible reference

This site is perpendicular to a (generally flexible) tringle at a point. The example is a "Rowlinson" site L in water models: LO is perpendicular to the HOH plane and at a fixed distance. The triangle may be flexible.

Because of flexibility, the weights cannot be pre-calculated; an example for water follows. To calculate the dependant position, we must calculate a vector perpendicular to the molecule plane

$$\vec{l} = \vec{h}_1 \times \vec{h}_2 \quad (\vec{h}_i = \vec{r}_{\mathrm{H}_i} - \vec{r}_\mathrm{O}, \ i = 1, 2) \tag{11.21}$$

Then $\vec{r}_{\mathrm{L}_{1,2}} = r_\mathrm{O} \pm |\mathrm{LO}|\vec{l}/|l|$.

To distribute forces, we calculate the torque (moment of force):

$$\vec{M} = \vec{l} \times \vec{f}_\mathrm{L} \tag{11.22}$$

We have to decompose $\vec{M}$ to components perpendicular to vectors $\vec{h}_{1,2}$; it is easier to decompose a perpendicular (but sign) vector to vectors $\vec{h}_{1,2}$. Thus we calculate a perpendicular vector proportional to M lying in the HOH plane:

$$\vec{M}^p = \vec{l} \times \vec{M} \tag{11.23}$$

The decomposition should be

$$\vec{M}^p = a_1 \vec{h}_1 + a_2 \vec{h}_2 \tag{11.24}$$

so that

$$
\begin{aligned}
a_1 &= \frac{(\vec{M}^p \cdot \vec{h}_1) h_2^2 - (\vec{M}^p \cdot \vec{h}_2)(\vec{h}_1 \cdot \vec{h}_2)]}{h_1^2 h_2^2 - (\vec{h}_1 \cdot \vec{h}_2)^2} \\
a_2 &= \frac{(\vec{M}^p \cdot \vec{h}_2) h_1^2 - (\vec{M}^p \cdot \vec{h}_1)(\vec{h}_1 \cdot \vec{h}_2)]}{h_1^2 h_2^2 - (\vec{h}_1 \cdot \vec{h}_2)^2}
\end{aligned}
$$

Then the equivalent forces to both hydrogens, H_i, $i = 1, 2$, are $a_i \vec{l}/l^2$, $i = 1, 2$, to H_i (and subtract both from the oxygen). The total force on oxygen is thus $\vec{f}_O = \vec{f}_L - (a_1 + a_2)\vec{l}/l^2$.

11.4   Site–site potential cutoff

Let $u(r)$ be an interatomic potential at atom-atom separation r. This non-bonded potential normally includes repulsive and attractive (London) forces, but not Coulomb interaction of (partial or full) charges, which are treated elsewhere, see Sect. 24 or 15.3. In the simulation, we use a truncated and smoothed (but not shifted) potential $u_{MD}(r)$,

$$u(r) = u_{MD}(r) + \Delta u(r), \tag{11.25}$$

where the correction $\Delta u(r) = 0$ for $r < C_1$ and C_1 is the inner cutoff. Note that the cutoff truncating scheme for blend differs, see Sect. 3.5. In `cook` it holds:

$$u_{MD}(r) = \begin{cases} u(r) & \text{for } r < C_1 \\ A(r^2 - C_2^2)^2 & \text{for } C_1 < r < C_2 \\ 0 & \text{for } C_2 < r \end{cases} \tag{11.26}$$

where $C_2 = \texttt{LJcutoff}$ is the real cutoff (so that $u_{MD}(r) = 0$ for $r > C_2$). Constants A and C_1 are calculated from the potential and the cutoff C_2 so that function u_{MD} and its derivative are continuous, see Fig. 11.1.

The most typical site–site potential is that of Lennard-Jones,

$$u_{LJ}(r) = 4\epsilon \left[(\sigma/r)^{12} - (\sigma/r)^6 \right] = \epsilon \left[(R_{vdW}/r)^{12} - 2(R_{vdW}/r)^6 \right], \tag{11.27}$$

see Sect. 3.2. If $C_2 \gg \sigma$ (for Lennard-Jones or any potential with van der Waals r^{-6} limiting behavior) it holds $C_1 \approx 0.775\,C_2$.

The homogeneous cutoff corrections are calculated by numerical integration (hence, any potential can be easily implemented) using the assumption that $g(r) = 1$ for $r > C_1$. If i, j are two sites, then the contribution is

$$\Delta U_{ij} = 4\pi \mathcal{N} \int_{C_{1,ij}}^{\infty} \Delta u_{ij} dr \tag{11.28}$$

where $\mathcal{N} = N/V$ is the number density of all sites. The total energy correction is

$$\sum_{i<j} \Delta U_{ij} \equiv \frac{\texttt{En.corr}}{V}, \tag{11.29}$$

Figure 11.1: The enlarged tail of the original and truncated potential (for the Lennard-Jones argon and $C_2 = 7\,\text{Å}$)

where the sum is over all site pairs. `cook` calculates ΔU_{ab} for all site types a, b and sums them with weights $N_a(N_a - 1)/2$ for like sites and $N_a N_b$ for unlike sites, multiplied by volume. This quantity (independent on volume) is called `En.corr` and it is reported; the cutoff energy correction is `En.corr`$/V$.

Since it holds

$$\int_{C_1}^{\infty} r\Delta u' \mathrm{d}r = \left[r\Delta u(r) \right]_{C_1}^{\infty} - \int_{C_1}^{\infty} \Delta u \mathrm{d}r = -\int_{C_1}^{\infty} \Delta u \mathrm{d}r \tag{11.30}$$

(note that $\Delta u(C_1) = \Delta(\infty) = 0$), the correction of pressure is simply

$$\Delta p = \frac{\texttt{En.corr}^2}{V}. \tag{11.31}$$

Variable `corr` determined whether the above corrections are included in the final values of energy and pressure.

About 3 LJ sigma for `LJcutoff` seems to be sufficient for liquid phase (but not for fluids close to the critical point!). This is the default: `LJcutoff=-3` in input data means that the real cutoff for each pair of interacting sites will be just 3 times their LJ sigma. For small systems `cutoff=LJcutoff=L/2` is OK. (`L` is the smallest box size).

11.5   The timestep

The errors (total energy drift and fluctuation, with caution also `cerr`'s) are proportional to higher powers of the timestep `h` (with the exception of the Verlet drift, see Sect. 11.1.2). If they do not scale with these higher powers once `h` is changed it means that other errors (e.g., r-space cutoff errors) spoil the energy and constraint conservation.

For Ewald, the r-space cutoff errors (see Sect. 11.2) should be comparable with the integration errors. Check it: an increase of `el.epsr` (or explicitly decrease `cutoff` or increase `el.alpha`) or an increase of `h` should worsen the conservation of energy while a decrease of `el.epsr` or decrease of `h` should not significantly improve it.

The optimum timestep size generally depends on the complexity of molecules and the temperature and pressure. Good `h` in normal conditions is around 0.002 ps if there are no explicit hydrogens but `h=0.001` if there are hydrogens; the latter pessimistic value is the default. Vibrating bonds (see option `-u`), especially for hydrogens, require `h=0.0005`.

11.6   Functions for r-space Ewald sums

The following functions are needed to calculate the r-space part of Ewald sums for point charges (for Gaussian charges see Sect. 24.2):

$$
\begin{aligned}
\mathrm{eru}(x) &= \alpha \mathrm{e}(\alpha\sqrt{x}), \\
\mathrm{erd}(x) &= -\alpha^3 \frac{\mathrm{e}'(\alpha\sqrt{x})}{\alpha\sqrt{x}}, \\
\mathrm{erud}(x) &= \mathrm{eru}(x), \text{ the corresponding macro also returns byerd} = \mathrm{erd}(x)
\end{aligned}
\tag{11.32}
$$

where

$$
\begin{aligned}
\mathrm{e}(y) &= \frac{\mathrm{erfc}(y)}{y}, \\
\mathrm{e}'(y) &= \frac{\mathrm{de}(y)}{\mathrm{d}y}, \\
\mathrm{erfc}(y) &= \frac{2}{\sqrt{\pi}} \int_y^\infty \exp(-t^2)\mathrm{d}t
\end{aligned}
$$

Functions $\mathrm{eru}(x)$ and $\mathrm{erd}(x)$ are approximated by hyperbolic splines (functions $A + B/(x+C)$, no jumps in the derivative). These simple functions are fast to evaluate, however, large tables are required to reach a good accuracy. The accuracy is controlled by `el.grid` which means that unity in the argument is divided into `el.grid` subintervals; i.e., there is `el.grid` subintervals for $r \in [0, 1/\alpha]$. The spline table size is $[(\texttt{range}\alpha)^2 + 1]$, where $\texttt{range} = \texttt{cutoff} + \texttt{el.rplus}$.

The maximum error occurs at the minimum distance `el.minqq` (the value of this variable affects the reported accuracy only, you can call `eru` and `erd` for any distance from 0 to $\texttt{cutoff}^2$). By default, cook has `el.minqq=1.0`. Since this minimum distance occurs normally for constrained bonds, the error in eru/erd in `el.minqq` is constant as well and cannot affect energy conservation. A minimum intermolecular distance is longer, errors lower but more relevant.

If requested so by `#define COULOMB -2` in simopt.h or -2 from "Select algorithm for ELECTROSTATIC calculations", cook uses a more accurate erfc formula for 1–2 and 1–3 distances (and auxiliary charges in POLAR version) so that larger `el.minqq` applies. Also POLAR implies COULOMB=-2.

Low `el.grid` may include some systematic errors into calculations, especially with COULOMB=-1 (which is not recommended). High `el.grid` may slow down the calculations (consider the cache memory!). The default is 256 for point charges (32 for Gaussian charges, see Sect. 24.2).

For ambient TIP4P water, a typical error with the default `el.grid=256` caused by using `COULOMB=-1` is 10 kPa in pressure and 1 J/mol in energy per particle. These errors are constant (for the same trajectory). A trajectory is not affected because all imprecise forces are constrained and are removed.

11.7   Hammonds–Heyes correction

The k-vectors beyond cutoff can be to some extent taken into account by changing α[12]. This is called α' or AP method. The k-space is run as usual using parameters $\kappa = $ `el.kappa` and $\alpha = $ `el.alpha`. The r-space including the "self-term" then uses a modified α' (reported as `alphar`).

$$\alpha' = \alpha \operatorname{erf}\left(\frac{\pi\kappa}{\alpha}\right).$$

Figure 11.2 moderate shows improvements for both pressures. To keep the accuracy high, cutoff=20 is recommended, which is close to the Ewald optimum speed. Decreasing κ to about $0.05/\text{Å}$ could give a considerable speedup here. Note also that usually a higher k-space error is acceptable. But to be on a safe side, a lot of tinkering is needed before productive runs.

Figure 11.2: The differences of pressure from an accurace reference for molten salt $Cs_{850}I_{850}$ (force field by Young–Cheatham, $T = 1550.3$ K) in dependence on κ. Parameters estimated from `el.epsr=el.epsk`=0.1 K/Å are:
top: cutoff= 20 Å, alpha=0.169269, kappa=0.180637,
middle: cutoff= 15 Å, alpha=0.227104, kappa=0.243901,
bottom: cutoff= 10 Å, alpha=0.343618, kappa=0.372304.
Shifted energy and forces (`el.rshift=3` or `7`) in the r-space. Three configurations shown.

Chapter 12

NVT and NPT ensembles

12.1 Kinetic temperature

The kinetic temperature is

$$T_{\text{kin}} = \frac{E_{\text{kin}}}{\frac{1}{2}kn_f} \tag{12.1}$$

where n_f is the number of the degrees of freedom,

$$n_f = 3N - n_{\text{bonds}} - n_{\text{cons}}, \tag{12.2}$$

where n_{bonds} is the number of bonds and similar mechanical constraints (as keeping selected atoms in place). n_{cons} is the number of conserved degrees of freedom (as momenta in periodic b.c.), optionally including energy conservation.

12.1.1 Should we subtract 1 from n_f for energy conservation?

In the Nosé–Hoover thermostat (and combined methods), there is one additional degree of freedom (n_f+=1) and one Hamiltonian conservation (n_{cons}+=1) which cancel out; then, it is proven[19, 20] that the system is canonical. There is a question whether to include consistently the energy conservation also in the NVE ensemble (and similarly the Berendsen thermostat) into the number of conserved quantities, n_{cons}. The NVE and Berendsen results are subject to $\mathcal{O}(1/N)$ errors anyway so that the question actually is, which option is more accurate? But this question is not well formulated either: Should we prefer the fastest convergence to the thermodynamic limit, or the best match with NVT (with the same N)?

Based on simulations with $N = 32$, see table 12.1, we conclude for NVE that:

- For liquid and solid argon, $n_f = 3N - 3$ (energy conservation not included in n_{cons}) is better for both kinetic temperature and pressure.

- For dense gas, $n_f = 3N - 3$ is slightly better (both $n_f = 3N - 4$ and $n_f = 3N - 3$ are likely the same for ideal gas), but for pressure $n_f = 3N - 4$ is better.

Similarly, the averaged temperature for Berensen differs from the thermostat temperature, so that it is not clear which temperature should be used for comparison (cf. the last lines of Table 12.1. Anyway, we observe that

Table 12.1: Argon simulations; unless given otherwise, $N = 32$, $h = 4$ fs, cutoff = LJcutoff = 7.85 Å, VERLET = 3. Pressure correction (corr&4) is used, $n_f = 3N - 3$ unless **marked***. `cookewps` is used because cutoff > (half box size). Potential and internal energies are per atom.

method	parms	T_{kin}/K	E_{pot}/ J mol^{-1}	U/ J mol^{-1}	P/kPa
gas $T = 150$ K, $\rho = 500$ kg m^{-3}					
Nose	as below, $N = 256$	149.994(9)	−1872.46(15)	−9.08(21)	7500(4)
Nose	$T = 150$ K, $\tau = 0.2$ ps	149.981(11)	−1777.87(13)	34.20(21)	7307(6)
NVE*	$U = $ Nose	150.914(8)	−1769.53(9)	34.21(0)	7352(5)
NVE	$U = $ Nose	149.304(8)	−1769.67(10)	34.21(0)	7175(5)
Ber	$\tau = 1$ ps	149.770(4)	−1772.41(12)	37.10(11)	7247(5)
Ber*	$\tau = 10$ ps	149.653(4)	−1774.73(13)	13.93(12)	7114(5)
Ber	$\tau = 10$ ps	149.662(4)	−1769.28(11)	38.94(10)	7237(5)
Ber	Gear $m = 4$	149.661(3)	−1769.29(11)	38.91(12)	7244(4)
Nose	$T = T_{\mathrm{Ber}}$	149.654(12)	−1779.37(12)	28.75(21)	7248(6)
liquid $T = 100$ K, $\rho = 1350$ kg m^{-3}					
Nose	as below, $N = 256$	99.983(9)	−4905.10(5)	−3663.02(12)	34213(12)
Nose	$T = 100$ K, $\tau = 0.2$ ps	99.985(11)	−4926.79(8)	−3718.78(14)	27251(15)
NVE*	$U = $ Nose	100.984(6)	−4925.70(8)	−3718.74(0)	27844(15)
NVE	$U = $ Nose	99.903(5)	−4925.76(6)	−3718.74(0)	27531(11)
Ber	$\tau = 1$ ps	99.606(5)	−4928.39(7)	−3724.96(5)	26914(14)
Ber*	$\tau = 10$ ps	99.570(5)	−4937.44(9)	−3747.38(6)	25090(18)
Ber	$\tau = 10$ ps	99.568(5)	−4928.52(10)	−3725.55(7)	26878(19)
Ber	Gear $m = 4$	99.570(5)	−4928.53(8)	−3725.53(6)	26880(16)
Nose	$T = T_{\mathrm{Ber}}$	99.576(12)	−4930.53(8)	−3727.46(17)	26396(17)
fcc crystal $T = 50$ K, $\rho = 1851.27$ kg m^{-3}					
Nose	as below, $N = 256$	50.000(5)	−6992.84(4)	−6371.69(3)	229780(6)
Nose	$T = 50$ K, $\tau = 0.2$ ps	50.003(5)	−7010.80(5)	−6406.66(2)	225278(10)
NVE*	$U = $ Nose	50.526(4)	−7010.52(5)	−6406.64(0)	225565(10)
NVE	$U = $ Nose	49.993(4)	−7010.66(5)	−6406.64(0)	225327(10)
Ber	$\tau = 1$ ps	49.769(4)	−7013.14(5)	−6411.83(1)	224650(11)
Ber*	$\tau = 10$ ps	49.746(5)	−7019.29(6)	−6424.73(1)	223167(11)
Ber	$\tau = 10$ ps	49.755(3)	−7013.45(4)	−6412.31(1)	224566(8)
Ber	Gear $m = 4$	49.745(4)	−7013.36(5)	−6412.34(1)	224585(10)
Nose	$T = T_{\mathrm{Ber}}$	49.763(4)	−7013.57(4)	−6412.34(2)	224525(8)

* $n_f = 3N - 4$ (additional 1 degree of freedom "for energy conservation" deducted from "standard" $n_f = 3N - 3$ in periodic boundary conditions.)

- $n_f = 3N - 3$ is better except for gas potential energy where $n_f = 3N - 4$ wins.

Therefore, from `cook*` version V3.2, the default is not to deduct 1 for energy conservation. The old option is available by `corr&64` flag.

12.2   Constant temperature simulations

The thermostat is turned on by selecting variable `thermostat`. The value of `tau.T` is then the typical correlation time to keep the temperature constant, irrespective of the method.

There are three kinds of thermostats available, the friction (Berendsen) thermostats, the Nosé–Hoover canonical ensemble, and the Maxwell–Boltzmann thermostats.

Special effects are obtained by decoupled inter- and intramolecular friction thermostats, selected by non-zero `thermostat>10`.

12.2.1   The Berendsen (friction) thermostat

The friction method, attributed to Berendsen, is based on the idea of velocity rescaling. In its crudest version the velocities are rescaled in every step so that the desired kinetic temperature is recovered or gradually approached. The differential form is equivalent to differential equations with friction term:

$$\ddot{\vec{r}}_i = \frac{\vec{f}_i}{m_i} - \frac{1}{2\tau_T} \dot{\vec{r}}_i \ln \frac{T_{\text{kin}}}{T}, \tag{12.3}$$

where

$$T_{\text{kin}} = \frac{1}{n_f kT} \sum_i m_i \dot{r}_i^2 \tag{12.4}$$

and T is the desired temperature and τ_T the typical time (it equals the correlation time for ideal gas or when the heat capacity is given by that of ideal gas with n_f degrees of freedom).

For `tau.T=h` the crude rescaling method is approximately recovered. In order not to spoil trajectories, `tau.T` should be comparable to the typical correlation time in the system and much longer than `h`, at least 1 ps for common biochemical systems.

The above formulation can be used also for constraint dynamics.

Advantages of the friction thermostat:

- Exponential convergence, for reasonable choices of `tau.T` usually fast.

- Simplicity, suitable also for Verlet/SHAKE

Disadvantages:

- Flying icecube problem – artifacts for rotating molecules not sufficiently coupled to the rest of the system. In this case, the rotational degrees of freedom are systematically hotter. Thus, problems are expected for fluids of small and weakly interacting molecules (hydrocarbons) while large molecules and with care (`tau.T>=1`) also water is OK. Note: this affects also one molecule in free boundary conditions, however, the system sets the angular momentum to zero. These artifacts can be observed via variables `Ttr` and `Tin`,

which should be the same if the system is well equipartitioned. (They may both differ from the kinetic temperature `Tkin` by not more than $1/n_f$, where n_f = number of degrees of freedom).

- Does not generate the canonical ensemble. If `tau.T` is long enough, then some distribution with `T=<Tkin>` is obtained, the fluctuations, however, differ from the canonical values (heat capacity cannot be obtained from fluctuations of internal energy). If `tau.T` is not long enough, the results are undefined.

- The energy is no longer constant – this strong test that our simulation is OK is lost.

I would like to stress that the above method is something different than the method used to *correct* energy drift introduced by the integrator (by using `tau.E`) because errors of h^(order of the method) are corrected. The velocities are rescaled by values that are of the order of the integration errors that occur anyway so that no additional errors are introduced. The energy conservation criterion is replaced by the difference between E and measured total energy but it is not lost.

The friction thermostat is selected by `thermostat=1` or `thermostat="friction"` or `thermostat="Berendsen"`.

12.2.2   Decoupled translational and intramolecular thermostats

The translational thermostat affects only motions of molecular centers-of-mass, the intramolecular thermostat affects only relative motions with respect to the center of mass (incl. rotations).

The decoupled thermostats should be used for smaller compact molecules with intramolecular motions weakly coupled with the rest of the system. Be aware that the improvement is to some extent only optical because the coupling remains week. You will get the correct average kinetic temperature, but the distribution of intramolecular kinetic energy may be incorrect.

12.2.3   The Nosé–Hoover canonical ensemble

The second approach by Nose [19, 21] adds one additional degree of freedom to the system is such sophisticated way that (provided that the system is ergodic) the true canonical ensemble is obtained.

The equations of motion follow. This form is directly used in the program (only constraint dynamics has been added). It was obtained from [19] by using his 'time transformation' and $\xi = \ln s$ instead of s, [20, 21].

$$\ddot{\vec{r}}_i = \frac{\vec{f}_i}{m_i} - \dot{\vec{r}}_i \dot{\xi} \tag{12.5}$$

$$\ddot{\xi} = \frac{1}{\tau_T^2}\left(\frac{T_{\mathrm{kin}}}{T} - 1\right) \tag{12.6}$$

Here, ξ is the additional degree of freedom. Note that $f = 3N - 3$ for N atoms in periodic boundary conditions: there are $3N + 1$ degrees of freedom (ξ included) and 4 constraints, 3 for conserved total momentum and 1 for the (generalized) Hamiltonian; for constraint dynamics the number of constraints has to be subtracted.

If f_i/m_i is omitted and the equations are linearized, we get a harmonic oscillator of the form (schematically)

$$\ddot{v} = -\frac{2}{\tau^2}(v - v_0) \tag{12.7}$$

I wrote "If f_i/m_i is omitted" and you may ask whether this is a good approximation. Unfortunately, it is—note that absolute and linear terms in the forces cannot change the oscillatory behavior and small varying positive/negative contributions behave like a noise. The more complex and denser liquid, the higher coupling between the auxiliary degree of freedom s and the system is provided, and the faster convergence occurs.

The Nose method is unsuitable for fluids at low densities and for soft potentials like exp-6.

The typical convergence profile of the kinetic temperature in the Nose simulation is:

```
      *
   *  *  **            **              ** **                **                          *
  *         *         *   **          *        *          **   ***    ** **      ** * *
 *             *      *      *        *          **      *           ***     ** **       **
*                *       *        *  ***              ** **
          ***                 *
```

The typical length of one cycle is `tau.T`. The time to get a good canonical distribution is longer and depends on `tau.T` in a way which is not straightforward – the convergence profile should be observed (using 20.1).

The convergence problems occur also when starting from unequilibrated state. The initial oscillations of `Tkin` may be as high as 1:10 and it takes a long time to damp them. It is recommended to use another thermostat first.

The Nose thermostat can be also used for calculating one molecule in gas phase using the `cookfree` version. (The molecule must not be too simple to guarantee ergodicity).

The Nosé-Hoover thermostat is selected by `thermostat=2` or `thermostat="Nose"`. The implementation is straightforward for the Gear integrator. With the Verlet integrator, there is a problem that the standard Verlet method with SHAKE does not allow velocities in the right-hand side. MACSIMUS solves this problem by using predicted velocities (of both particles and $\dot{\xi}$)[1] The predictor length is selected by option `-pC`; the default is C=2, which gives time reversibility of $\mathcal{O}(h^7)$. See Sect. 26.

12.2.4   Maxwell–Boltzmann thermostat

The third method, suitable for systems without constraints (see option `-u`), is to replace once a while the velocities by new ones drawn from the Maxwell–Boltzmann distribution. It can be done in two ways: periodically (in time intervals given by `tau.T`, all velocities are replaced: `thermostat="Maxwell"`), or to update randomly chosen atoms (`thermostat="Andersen"`). Advantages

- Fast convergence.

---

[1]Other methods include iterations to obtain self-consisteny in velocities and RATTLE, methods based on the Trotter decompositions[21].

- True canonical ensemble

Disadvantages:

- Problematic for constraints and Gear integration

- Spoils time development of trajectories

- The Andersen thermostat erases any energy conservation information

This thermostat is fully and correctly implemented for vibrating bonds and Verlet/SHAKE only. For weak coupling, (long tau.T), it can be used, with some loss of accuracy, for constrained bonds or Gear integration. It works worse for constrained bonds and Gear integration (Lagrangian dynamics).

The same effect as the the Maxwell thermostat, but executed just onece, can be obtained by specifying `initvel`.

12.2.5   Langevin thermostat

Random force and friction are added to the equations of motion,

$$\ddot{x} = -\frac{\dot{x}}{\tau_T} + \sqrt{\frac{2k_{\mathrm{B}}T}{m\tau_T}}X \tag{12.8}$$

where X is uncorrelated Gaussian random noise, $\langle X(t)X(t')\rangle = \delta(t - t')$, x stands for one coordinate (of atom or center of mass), and m is the atom or molecule mass. The time constant $\tau_T = $ `tau.T` in the data (in ps). The random force is implemented (with respect to time step h) as normalized Gaussian random number divided by $\sqrt{h}$.

Bugs: Implemented for Gear only, the correlation time is twice as long as should. Cf. `tau.CM`.

12.2.6   Which thermostat

For easy systems there is no difference between thermostats. Problems start with systems with fast and slow motions not coupled together: e.g., the full-atom model of methane. When the C-H potentials are harmonic springs, then the energy transfer between the fast C-H vibrations and other degrees of freedom is very slow (and the integration errors accumulate to violate the thermal equilibrium between different degrees of freedom). Then, the Maxwell–Boltzmann thermostat is recommended because it samples all degrees of freedom. In less problematic cases (e.g., constrained bonds but not angles) the decoupled thermostats may be a choice. In all these cases the Nose and simple friction thermostats will be inefficient. Sometimes it may help to start with the Maxwell–Boltzmann thermostat for fast equilibration and continue by, e.g., Nose or friction (the latter with a long `tau.T` to avoid its artifacts). In any case, it is strongly recommended to monitor the 'internal' and 'external' (translational) temperatures.

12.3   Constant pressure simulations

Nonzero value of `tau.P` allows for a variable box size. The barostat pressure to keep is given by variable P. See also `rescale` and `bulkmodulus`.

12.3.1   Friction (Berendsen) barostats

Unless `thermostat="NPT"`, the barostat is a simple Berendsen-style one. Pressure is calculated onece every cycle (`noint` steps) and either the configuration is rescaled once after every cycle (for `tau.P<0`) or after every step by a proportionally smaller value (for `tau.P>0`).

Unless `bulkmodulus` is set, the friction term is derived using ideal-gas compressibility (more exactly with the number of degrees of freedom instead of the number of particles). In dense systems the compressibility is several times smaller (16 times for water, roughly 100 times for ambient-temperature solids), therefore the values of `tau.P` must be increased in the same proportion; `tau.P=10` to 1000 is a useful range. Alternatively, variable `bulkmodulus` can be set; `tau.P` is then a realistic relaxation time. This (realistic) `tau.P` should be several times longer than `tau.T` anyway. Setting `bulkmodulus` is recommended for solids at very low temperatures where the ideal gas approximation is very bad indeed; the bulk modulus of solids is of the order of 1e10 Pa.

12.3.2   MTK thermostat and barostat

The Martyna-Tobias-Klein (MTK) barostat [8] provides the NPT ensemble defined by the following semiclassical partition function

$$Z_{NPT} = \int Z_{NVT} \mathrm{e}^{-\beta PV} \mu(V) \mathrm{d}V, \tag{12.9}$$

where Z_{NVT} is the canonical partition function and the "volume measure", "density of volume states", or "scale function" μ is

$$\mu = \begin{cases} 1/V & \text{for } \texttt{tau.CM=0} \\ \text{const} & \text{for } \texttt{tau.CM} \text{ set, see } 9.2.5 \end{cases}$$

The standard MTK version without sampling the momenta (`tau.CM=0`) in the periodic boundary conditions is sometimes referred to as $(N-1)PT$ ensemble [14]. The NPT ensemble is selected by `thermostat="NPT"` or `thermostat="MTK"`. The pressure tensor (virial pressure in the isotropic case) is needed and calculated at every step. Temperature `T`, pressure `P`, and time constants `tau.T` and `tau.P` must be set.

The periodic simulation box with all three box sides fluctuating independently is described by tensor $\overset{\leftrightarrow}{\lambda} = \ln(\overset{\leftrightarrow}{L})$ (by components). MACSIMUS currently supports only diagonal components, i.e., a rectangular box; however, we keep the tensor notation for consistency. The code is in `constrd.c`, function `Shake()`, in parts/files indicated below. Tags with @ refer to the same tags in the code. For a version without barostat (Nosé–Hoover only), see Sect. 26.

@1 `constrd.c`	$\overset{\leftrightarrow}{L}$	$= \exp(\overset{\leftrightarrow}{\lambda})$, by components; in the code $\lambda := \ln(L)$	(12.10)
@2 `constrd.c`	V	$= L_x L_y L_z$	(12.11)
@3 `forces()[,SCF]`	f_i	$= -\dfrac{\partial U}{\partial \vec{r}_i}$	(12.12)
@4 `trvpscale.c`		TRVP: predict all velocities at t from $t - \frac{h}{2}, t - \frac{3h}{2}, \ldots$	(12.13)
@43 `trvpscale.c`	V_{f}	$= \dot{\xi} + \dfrac{\mathrm{Tr}\, \overset{\leftrightarrow}{\dot{\lambda}}}{f}$ in the code: $\xi = \texttt{logs}, V_{\mathrm{f}} = \texttt{Vfactor}$	(12.14)
@53 `trvpscale.c`		scaling predictor, see (12.29) and (12.30)	(12.15)

@6 `verlet.c`
$$\dot{\vec{v}}_i = \frac{\vec{f}_i}{m_i} - \left(V_{\mathrm{f}} \overleftrightarrow{I} + \dot{\overleftrightarrow{\lambda}} \right) \cdot \vec{v}_i \quad \text{\& leap-frog} \tag{12.16}$$

@7 `constrd.c`
$$\text{SHAKE with predicted scaling of bond lengths} \tag{12.17}$$

`shakev#.c`
$$\overleftrightarrow{E}_{\mathrm{kin}} = \frac{1}{2} \sum_i m_i \vec{v}_i \vec{v}_i, \quad E_{\mathrm{kin}} = \mathrm{Tr}(\overleftrightarrow{E}_{\mathrm{kin}}) \qquad = \texttt{En.Pkin} \tag{12.18}$$

@93 `thermo.c`
$$\overleftrightarrow{P}_{\mathrm{cfg}} = \frac{1}{V} \left(2\overleftrightarrow{E}_{\mathrm{kin}} + \sum_i \vec{r}_i \vec{f}_i + \cdots \right) + P_{\mathrm{corr}} \overleftrightarrow{I} \tag{12.19}$$

@A3 `thermo.c`
$$\ddot{\xi} = \frac{2E_{\mathrm{kin}} + \sum_\alpha M_{P,\alpha} \dot{\lambda}_{\alpha\alpha}^2 - (f + f_+)k_{\mathrm{B}}T}{M_T} \tag{12.20}$$

@B3 `thermo.c`
$$\text{leap-frog for } \xi \tag{12.21}$$

@C3 `thermo.c`
$$\ddot{\overleftrightarrow{\lambda}} = \frac{1}{M_P} \left[V(\overleftrightarrow{P}_{\mathrm{cfg}} - P\overleftrightarrow{I}) + \frac{2E_{\mathrm{kin}}}{f}\overleftrightarrow{I} \right] - \dot{\xi}\dot{\overleftrightarrow{\lambda}} \tag{12.22}$$

@D3 `thermo.c`
$$\text{leap-frog for } \lambda \tag{12.23}$$

@E3 `thermo.c`
$$\text{box rescaling} \tag{12.24}$$

where f_+ is the number of independently fluctuating box sides and $\overleftrightarrow{I}$ is the unit tensor. Note also the different definitions of $E_{\mathrm{kin}} = \mathrm{Tr}(\overleftrightarrow{E}_{\mathrm{kin}})$ and $P_{\mathrm{cfg}} = \mathrm{Tr}(\overleftrightarrow{P}_{\mathrm{cfg}})/3$.

The ellipsis in (12.19) stands for other terms (constraint forces, polarizable terms, etc.; see Sect. 15.7). The cutoff correction is assumed to be isotropic (see Sect. 11.4):

$$P_{\mathrm{corr}} = -\frac{\partial U}{\partial V} = \frac{\texttt{En.corr}}{V^2} \tag{12.25}$$

The masses of the extended degrees of freedom are related to the thermostat and barostat characteristic times

$$M_T = fk_{\mathrm{B}}T\tau_T^2, \tag{12.26}$$
$$M_P = f_P(f + 3)k_{\mathrm{B}}T\tau_P^2, \tag{12.27}$$

where f_P is the number of fluctuating (not constant) box sides. (Note the difference between f_P and f_+. E.g., if x and y fluctuate synchronously and z is independent, then $f_P = 3$ and $f_+ = 2$.)

For ideal gas and $\tau_T \ll \tau_P$ (which is recommended), it holds

$$\tau(E_{\mathrm{kin}}) = \frac{\tau_T}{\sqrt{2}}, \ \tau(V) = \sqrt{3}\tau_P, \tag{12.28}$$

where $\tau quantitiy$ is the inverse circular frequency obtained from the time autocorrelation function[?]. Table 12.2 contains the pseudoexperimental characteristic times.

WARNING: the above M_P differs from the M_P in [8]; in addition, the equations of motion do differ by factors of 3 inside the algorithm as well. Effectively for the isotropic barostat, formula (12.27) gives τ_P which is $1/3$ of the characteristic time τ_P of [?]. However, out choice of tau_P is closer to the experimental correlation times in condensed systems.

These equations do contain a correction to the conserved total momentum [8] (term $\mathrm{Tr}\dot{\overleftrightarrow{\lambda}}/f$ in (12.16) or $3\dot{\lambda}/f$ in (12.39)) so that E_{kin} should not be rescaled by $(f + 3)/f$, cf. (15.15).[2] A warning is printed on incorrect use.

---

[2]In the case of this correction added anyway, $3/f$ in (12.39) would be replaced by $6/f$, so in fact the correction would be applied twice.

Table 12.2: The thermostat and barostat characteristic times, Eq. (**??**), and the corresponding real characteristic times of the controlled quantities determined from the respective time correlation functions. All data are in ps.

system	τ_T	$\tau(E_{\mathrm{kin}})$	τ_P MACSIMUS	τ_P literature	$\tau(V)$
ideal gas, Eq. (12.28)	0.2	0.7071	3	9	5.196
SPC/E	0.2	0.227	3	9	1.79
RTIL-flex	0.2	0.1965	3	9	1.406
RTIL-rigid	0.2	0.1962	3	9	1.256
liquid nitrogen	0.2	0.1371	0.5	1.5	too damped
liquid nitrogen	0.2	0.1575	2	3	0.56
liquid nitrogen	0.2	0.1613	2	6	1.155

The MACSIMUS implementation uses a time-reversible velocity predictor (TRVP, see Sect. 26). This is not sufficient if there are constraints treated by SHAKE because the scaling (box size) is needed to know the bond lengths at time $t + h$. A sufficiently accurate workaround predicts the value of $\lambda(t + h)$ as

$$\overset{\leftrightarrow}{\lambda}(t + h) = \overset{\leftrightarrow}{\lambda}(t) + h\overset{\dot{\leftrightarrow}}{\lambda}(t + h/2) \tag{12.29}$$

with a second-order predictor for the difference

$$\overset{\dot{\leftrightarrow}}{\lambda}^{\mathrm{pred}}(t + h/2) = 2\overset{\dot{\leftrightarrow}}{\lambda}(t - h/2) - \overset{\dot{\leftrightarrow}}{\lambda}(t - 3h/2). \tag{12.30}$$

The Verlet-based values of $\vec{r}_{ij}$ to be constrained are then multiplied[3] by the predicted anisotropic scaling $\exp[h\overset{\dot{\leftrightarrow}}{\lambda}^{\mathrm{pred}}(t+h/2)]$. After $\overset{\leftrightarrow}{\lambda}(t+h)$ is calculated by Verlet, the configuration is rescaled by $\exp[\overset{\leftrightarrow}{\lambda}(t + h) - \overset{\leftrightarrow}{\lambda}(t)]$. The constraints are subject to error on the order of $h^2\overset{\ddot{\leftrightarrow}}{\lambda}(t) \propto (h/\tau_P)^2$.

General independent scaling of all box sides is selected by `rescale="XYZ"`, scaling of x and y (keeping L_z constant) by `rescale="XY"`, etc.

If a synchronous scaling, i.e., constant ratio $\lambda_{xx} : \lambda_{yy}$ or $\lambda_{xx} : \lambda_{yy} : \lambda_{zz}$, is requested (by `rescale="XX"` or `rescale="XXX"`), the value of f_+ changes to 2 or 1, and the second derivative in (12.24) is modified by

$$\ddot{\lambda}_{xx} = \ddot{\lambda}_{yy} := \frac{\ddot{\lambda}_{xx} + \ddot{\lambda}_{yy}}{2} \quad \text{or} \quad \ddot{\lambda}_{xx} = \ddot{\lambda}_{yy} = \ddot{\lambda}_{zz} := \frac{\ddot{\lambda}_{xx} + \ddot{\lambda}_{yy} + \ddot{\lambda}_{zz}}{3} = \frac{\mathrm{Tr}\overset{\ddot{\leftrightarrow}}{\lambda}}{3}, \tag{12.31}$$

respectively.

The conserved total energy is

$$E_{\mathrm{tot}} = E_{\mathrm{kin}} + \frac{M_T}{2}\dot{\xi}^2 + \frac{M_P}{2}(\overset{\dot{\leftrightarrow}}{\lambda})^2 + U + PV + (f + f_+)k_{\mathrm{B}}T\xi \tag{12.32}$$

It can be easily proven by taking $\dot{E}_{\mathrm{tot}}$. Symbol $(\overset{\dot{\leftrightarrow}}{\lambda})^2 = \dot{\lambda}_{xx}^2 + \dot{\lambda}_{yy}^2 + \dot{\lambda}_{zz}^2$.

---

[3]Actually $\vec{r}_{ij}$ inside the SHAKE algorithm are rescaled by $\exp[h\overset{\dot{\leftrightarrow}}{\lambda}(t + h/2)]$

For NPT in a cubic box there is simply $L = \exp(\lambda)$, $\lambda = \text{Tr}\overset{\leftrightarrow}{\lambda}$. This is selected by `rescale="xyz"`. This option option differs from `rescale="XXX"` because the isotropic pressure (not pressure tensor) is used. This is faster (if PRESSURETENSOR=3 is not configured); however, for cutoff electrostatic less accurate. Real positions are $\vec{r}_i = L\vec{\rho}_i$ ($\vec{\rho}_i$ are the scaled dimensionless coordinates), velocities with respect to the "background" are $\vec{v}_i = L\dot{\vec{\rho}}_i$ (i.e., not dragged with if the whole box shrinks/swells). "Real" (not useful) velocities including the "drag" caused by fluctuating box would be $\dot{\vec{r}}_i$. Number of degrees of freedom $= f$; for simple fluid $f = 3N - 3$ (minus 3 for periodic b.c.).

For isotropic fluctuations the algorithm reads as:

$$\texttt{@1 constrd.c} \qquad L = \exp(\lambda) \tag{12.33}$$

$$\texttt{@2 constrd.c} \qquad V = L^3 \tag{12.34}$$

$$\texttt{@3 forces()[,SCF]} \quad f_i = -\frac{\partial U}{\partial \vec{r}_i} \tag{12.35}$$

$$\texttt{@4 trvpscale.c} \qquad \text{TRVP: predict all velocities at } t \text{ from } t - \tfrac{h}{2},\, t - \tfrac{3h}{2},\, \dots \tag{12.36}$$

$$\texttt{@41 trvpscale.c} \quad V_\text{f} = \dot{\xi} + \left(1 + \frac{3}{f}\right)\dot{\lambda} \qquad f = \texttt{No.f},\, 1 + \frac{3}{f} = \texttt{No.NPT} \tag{12.37}$$

$$\texttt{@51 trvpscale.c} \qquad \text{scaling predictor} \tag{12.38}$$

$$\texttt{@6 verlet.c} \qquad \dot{\vec{v}}_i = \frac{\vec{f}_i}{m_i} - V_\text{f}\vec{v}_i \;\;\text{\& leap-frog} \tag{12.39}$$

$$\texttt{@7 constrd.c} \qquad \text{SHAKE with predicted scaling of bond lengths} \tag{12.40}$$

$$\texttt{shakev\#.c} \qquad E_\text{kin} = \frac{1}{2}\sum_i m_i v_i^2 \tag{12.41}$$

$$\texttt{@91 constrd.c} \qquad P_\text{cfg} = \frac{1}{3V}\left(2E_\text{kin} + \sum_i \vec{r}_i \cdot \vec{f}_i + \cdots\right) + P_\text{corr} \tag{12.42}$$

$$\texttt{@A1 thermo.c} \qquad \ddot{\xi} = \frac{2E_\text{kin} + 3M_P \dot{\lambda}^2 - (f+1)k_\text{B}T}{M_T} \qquad f + 1 = \texttt{No.fx} \tag{12.43}$$

$$\texttt{@B1 thermo.c} \qquad \text{leap-frog for } \xi \tag{12.44}$$

$$\texttt{@C1 thermo.c} \qquad \ddot{\lambda} = \frac{1}{M_P}\left[V(P_\text{cfg} - P) + \frac{2E_\text{kin}}{f}\right] - \dot{\xi}\dot{\lambda} \qquad \frac{1}{f} = \texttt{No.invf} \tag{12.45}$$

$$\texttt{@D1 thermo.c} \qquad \text{leap-frog for } \lambda \tag{12.46}$$

$$\texttt{@E1 thermo.c} \qquad \text{box rescaling} \tag{12.47}$$

Terms in red are zero in the algorithm by Hoover[22], available by `thermostat="HooverNPT"`.

12.3.3   Simulation along given $V(t)$ time dependence

Simple box rescaling at every step according to defined function $V(t)$ leads to poor energy conservation. The following algorithm uses ideas of the extended Lagrangian barostat of Andersen:

$$\texttt{@3 forces()[,SCF]} \quad f_i = -\frac{\partial U}{\partial \vec{r}_i} \tag{12.48}$$

$$\texttt{@4 trvpscale.c} \qquad \text{TRVP: predict all velocities at } t \text{ from } t - \tfrac{h}{2},\, t - \tfrac{3h}{2},\, \dots \tag{12.49}$$

$$\texttt{@4b trvpscale.c} \qquad \overset{\leftrightarrow}{V}_{\mathrm{f}} \;=\; \frac{1}{h}\ln\left[\frac{\overset{\leftrightarrow}{L}(t+h/2)}{\overset{\leftrightarrow}{L}(t-h/2)}\right] \qquad \text{is scalar for } V(t) \qquad (12.50)$$

$$\texttt{@5b trvpscale.c} \qquad\qquad \text{scaling predictor} \qquad\qquad\qquad (12.51)$$

$$\texttt{@6 verlet.c} \qquad \dot{\vec{v}}_i \;=\; \frac{\vec{f}_i}{m_i} - \overset{\leftrightarrow}{V}_{\mathrm{f}}\vec{v}_i \;\; \text{\& leap-frog} \qquad (12.52)$$

$$\texttt{@7 constrd.c} \qquad\qquad \text{SHAKE with predicted scaling of bond lengths} \qquad (12.53)$$

$$\texttt{shakev\#.c} \qquad E_{\mathrm{kin}} \;=\; \frac{1}{2}\sum_i m_i \vec{v}_i^{\,2} \qquad (12.54)$$

$$\texttt{@D1 thermo.c} \qquad P_{\mathrm{cfg}} \;=\; \frac{1}{3V}\left(2E_{\mathrm{kin}} + \sum_i \vec{r}_i\cdot\vec{f}_i + \cdots\right) - P_{\mathrm{corr}} \qquad (12.55)$$

$$\texttt{@Eb thermo.c} \qquad\qquad \text{box rescaling, cf. @A–@E in the above algorithms} \qquad (12.56)$$

This algorithm is requested by negative `tau.rho`:

`tau.rho=-1` for a cubic box (with `rescale="xyz"`). In this case, $\overset{\leftrightarrow}{L}$ and $\overset{\leftrightarrow}{V}_{\mathrm{f}}$ become scalars. Text file <u>simname</u>`.box` should contain lines of pairs $\{t, \rho\}$, where t is time in [ps] and ρ is the density in $\mathrm{kg\,m^{-3}}$. The file must be ordered by t. Note that `init>=2` sets $t := 0$, otherwise the time continues.

`tau.rho=-3` for controlling all three box sides separately (with `rescale="XYZ"`). Similarly, text file <u>simname</u>`.box` should contain lines of $\{t, L_x, L_y, L_z\}$, where box sides are in Å. (NOT TESTED).

The interaction with thermostats is unknown, Berendsen will likely work, for models without constraints also Andersen and Langevin. The code for Nose should be checked.

12.3.4 Adjusting force field parameter to pressure

Instead of scaling the box, it is possible to change a selected "atom size" (size-like parameter of the force field, as the Lennard-Jones σ, or a sum of R_{min} radii, or similar) and to rescale it in the Berendsen (friction) way so that pressure reaches the predefined value (P).

To do this, there must be a line in the ble-file in the `nbfixes` table for given pair of site types; do not forget to set variable `nnbfixes`. In the `nbfixes` table, the first two columns are the respective sites coded as symbolic names (e.g., KR AR), the third parameter is an energy-like parameter (e.g., Lennard-Jones ϵ), and the fourth column is the size-like parameter (e.g., Lennard-Jones σ), to be scaled. In the cook input data, the respective sites are `tau.i` and `tau.j`, coded as numbers. These numbers correspond to the line of the table of non-bonded interaction (e.g., `Lennard-Jones` or `Buckingham`) in the C-style (1st line has index of 0, the last line `nsites-1`; the i column in the table does not apply). 1–4 term is not supported here.

Similarly as for the Berendsen barostat, `bulkmodulus` should be set to define the sensitivity of pressure to particle size. Variable `tau.sig` then defines the correlation time (like `tau.P` for Berendsen NPT ensemble). For a good stability, `tau.sig` should be several times longer than `tau.T` (e.g., `tau.T=1` and `tau.sig=10`). If the sigma-like parameter has a meaning of inversed size (e.g., parameter B in the Buckingham force field), `bulkmodulus` must be negative.

During simulation, the selected value of σ (or B) fluctuates so that the instantenous pressure on average equals parameter `P`. The final value of σ is stored in the cfg-file for further steps/restart.

The value of σ is also recorded in column 4 of the cp file (where normaly density is recorded) and statistics is calculated.

To start the σ-fitting mode, specify `tau.sig` (and `bulkmodulus`); if there is nonzero σ in the cfg-file, it is used, otherwise the initial value of σ is taken from the ble-file. On the other hand, for `tau.sig=0` the value possibly stored in the cfg-file is ignored and the standard ble-file value applies; in addition, `sigma=0` is set for subsequent write to the cfg-file.

BUG: Smooth cutoff is always recalculated if σ changes. However, the cutoff corrections are not recalculated, which is a bug in principle, although in practice the error is small.

In V2.7t and older, only single atom (not pair) σ was supported, there was `tau.R` instead of `tau.sig`, and `tau.j` was not used.

Chapter 13

Initial configuration

There are several ways how to generate initial configurations. Small molecules can be placed in vertices of a lattice, bigger either to enlarged box (to be compressed lated) or shot to an (enlarged) lattice at random. A small lattice with a periodic configuration can be repeated several times. A problem of immersing a big molecule into a liquid (of small molecules) can be solved by removing overlapping small molecules.

13.1 Random-shooting algorithm

If `init=3` or `init="random"` is specified in input data, the initial configuration is generated by the following algorithm suitable especially for small or moderately large molecules.

1. For given initial density (variable `initrho`), an attempt is made to fill the simulation box by randomly distributed and rotated (if `No.rotatefrom` permits) molecules. Molecules are inserted one-by-one and there is an energy limit (variable `Emax`); if it is exceeded, the insertion of the molecule is rejected.

2. If step 1. is not successful (i.e., if the insertion probability exceeds the limit of `pins`), the density is lowered and step 1. is repeated. Ignored if `pins=0`.

3. If `pins<0` only: If 1. is successful, however, the system is too dilute (i.e., the insertion probability of step 1. has been greater than `sqrt|pins|`), step 1. is repeated with a higher density.

4. Velocities are assigned (approximately according to temperature `T`) and both velocity and length constraints are adjusted.

A configuration obtained by this algorithm can be dilute and should be shrunk. The final density is named `rho` (and is in $\mathrm{kg\,m^{-3}}$). The speed of changing the density is given by typical time `tau.rho` (in ps). Recommended values are 0.1–1 ps; too low values may cause a crash. If you meet problems, we recommended to run first several steps with `tau.rho=0` (this means in fact infinity, i.e., changing density switched off). Use the friction thermostat to cool the heat of compression.

13.2   Crystal initial configuration

Alternatively, small molecules may be placed at vertices of a regular lattice which is selected by `init=5`. The lattice type is defined by variable `pins` either explicitly (values 1,2,3) or left to program's decision (`pins=0`) to obtain lattice with minimum number of vacancies. There is no overlap check and it is user's responsibility to choose, if necessary, density low enough not to have overlapping molecules; see also option `-q` which offers possibility to turn off random rotation of molecules.

If `pins>3` then the configuration is obtained by repeating an elementary cubic cell containing a prepared configuration of `pins` molecules; see the explanation of `pins` for details.

13.3   Immersing a large solute into solvent

Rapid changes in density can destroy a structure of a big macromolecule. For building an initial configuration consisting of (for example) a big protein in water, there are two possibilities:

METHOD 1

1. Prepare the `.ble`-file with the macromolecule as the first molecular parameter given to `blend`. Thus, it has index 0.

2. Select `cook` option `-j1` to freeze the atom positions of the protein.

3. Run the initializer `init=3` with `initrho=rho` (or only slightly lower if necessary) and very high `Emax` and `pins=0` (or so low so that the initializer succeeds);

4. Use the friction thermostat (set `tau.T` and `T`). You may have to specify a rather low timestep `h` if the initial `Emax` was very high. In normal cases, `tau.T` should not be (much) lower than the typical relaxation time in water which is about 1ps; you may have to use longer values for complex proteins with cavities.

5. Run `cook` again without option `-j`.

METHOD 2

1. Prepare the `.ble`-file with the macromolecule as the first molecular parameter given to `blend`. Thus, it has index 0, and the solvent has index 1.

2. Prepare independently a configuration (may be small, but normally at least about 50 molecules) of the solvent (water) and export it using `-l` option. Your protein will be immersed into this configuration (possibly periodically repeated) so that you must precalculate the appropriate size of the system to allow this.

3. Rename the ascii dump to `cfgns.pins`; e.g., `cook/cfg3.50` contains such configuration of 50 TIPS water molecules.

4. Run `cook` with option `-j1` to freeze the positions of the protein.

5. Run the initializer `init=5` with `pins` equal the number of molecules from step 1. `Emax` should be quite high.

6. Some of the solvent molecules are removed. Record the changed numbers of molecules and edit the `.def` file accordingly.

7. Run `cook` again without option `-j`. As in METHOD 2, equilibrate, preferably first with the friction thermostat.

Chapter 14

Measurements

14.1 Units of measurements

I'm sorry, but three systems of units of measurements are used by program `cook`:

- File `sysname.ble` defining the force field uses kcal/mol as the energy unit and Å as the length unit because these units are used by CHARMM. See the documentation of `blend` for details.

- Internally, `cook` uses Ångström to measure lengths, ps to measure time, Kelvin to measure temperature and $k \cdot$K (where k is the Boltzmann constant) to measure energy (loosely, energy is measured in Kelvins). See `units.h` for details. The values of constants are based on CODATA 2019.

- `cook` communicates with the user in SI units: density in kg/m^3, pressure in Pa, energy in J/mol; however, energy is often given in Kelvins (I have a better feeling on Boltzmann probabilities if the Boltzmann constant is defined as unity).

14.2 Convergence profile

The convergence profiles are important for observing whether the system is in equilibrium. Many numerical or physical problems are also discernible from convergence profiles. Note that the term "convergence profile" here means just time dependence of quantities, not running averages of any kind.

`cook` records several quantities every cycle as the default, more quantities can be specified in file `sysname.def`, See Sect. 9.2.3. **The default quantities have changed, those given below do not apply in all versions of `cook`.**

`column 1 = Etot` Total energy in Kelvin (more precisely, Etot/k_B is in Kelvin). For NVE simulation or Nose-Hoover thermostat or MTK thermostat+barostat or simulation with given box vs. t dependence it includes the extended degrees of freedom; therefore, it should be constant (with noise and only a small drift). See Sect. 11.1.2, for details.

`column 2 = Tkin` Instantaneous (kinetic) temperature in K. From `showcp`, the total kinetic temperature is combined with the translatioal part (column 7 = `Ttr`) and the rest (rotational and vibrational, column 6 = `Tin`).

column 3 = `Epot` Total potential energy, in J/mol (of the whole configuration).
 Note: `Epot = LJ + Ecorr + bond + Eel`

 LJ Non-bonded (LJ) energy, separately available with direct pair forces algorithm (not LINKCELL), without cutoff correction.

 Ecorr Cutoff energy correction.

 bond All bonded forces (bonds, angles, torsions), separately available with direct pair forces algorithm (not LINKCELL) (?).

 Eel (also `elst`) = electrostatic energy (POLAR: incl. self-term)

column 4 The meaning depends on the simulation ensemble. Possible quantities are:

 `rho` Density in $\mathrm{kg\,m^{-3}}$, for NPT and other simulaion with changing density.

 `PdV` Pressure obtained from the virtual volume change, see variable `dV`.

 `Ep+k` Sum of potential + kinetic energies, in J/mol (differs from `Etot` because no extended degrees of freedom are included).

 `Ep0` No.first molecules [J/mol], not tested recently.

column 5 = `P` Pressure in Pa; in special cases replaced by `rho` or `PdV` (see above) or `svdW` (see `tau.sig`).

column 6 = `Tin` Rotational + vibrational kinetic energy, in K.

column 7 = `Ttr` Translational kinetic energy, in K. In a well equipartioned system `Tin` = `Ttr`. Note that for the Berendsen thermostat generally `Tin` may slightly differ from `Tkin`.

`|M|` Total simulation cell dipole moment [p.u.=0.0117501021272 Debye]

`|J|` $\sqrt{J^2}$, simulation cell current density in $\mathrm{A\,m^{-2}}$

`self` self-field in [p.u.]

Additional quantities to be recorded can be given in file simname.`cpi`, see Sect. 9.2.3.

For technical information on cp-file format, see Sect. 16.2.

14.3   Analysis of statistical errors

Let Q be an arbitrary quantity measured in the simulations and $Q_i, i = 1..n$, its values obtained in consecutive measurements. Then the expected error of the average $\langle Q \rangle$ is given (under some assumptions) by the formula

$$\delta Q = \sqrt{\mathrm{Var}Q(1 + 2\tau)} \tag{14.1}$$

where the (discrete) correlation time is defined by:

$$\tau = \sum_{t=1}^{\infty} c_t \tag{14.2}$$

and the autocorrelation coefficients are

$$c_t = \frac{\mathrm{Cov}(Q_{i+t}, Q_i)}{\mathrm{Var}Q}, \quad \mathrm{Cov}(Q_{i+t}, Q_i) = \langle Q_{i+t}Q_i \rangle - \langle Q_{i+t} \rangle \langle Q_i \rangle, \quad \mathrm{Var}Q = \mathrm{Cov}(Q_i, Q_i) \tag{14.3}$$

c_t should go to zero and the sum for τ should converge (if it does not or does slowly, another methods should be used). In practice, the sum for τ is truncated at a certain $t = t_{\max}$; if $t_{\max}$ is too low, τ is usually underestimated; if $t = t_{\max}$ is too high then the inaccuracies in c_t may cause too large errors in τ.

It is possible to extrapolate c_t to infinity, usually by $c_t = A \exp(-Bt)$ (one bottleneck process or barrier) or $c_t = At^{-3/2}$ (hydrodynamic tail) and thus to correct τ.

Another way is to use blocking or sub-averages and the same method with only few first autocorrelation coefficients, or perhaps none (just blocking)m but it is better to include at least c_1.

To understand the output of `statics.c`, let us consider the following example produced by testing program `staticst.c`:

```
-----------------------------------------------------------------------
r               No = 100000     range = <0.78591037,4.2648801> = 3.4789698
 Mean = 2.50087886     Var = 0.232787747     StDev0 = 1.5257e-03
 _t__c[t]_1+2tau__StDev_ _t__c[t]_1+2tau__StDev_ _t__c[t]_1+2tau__StDev_
   1 0.8020 2.604 2.5e-03  2 0.6440 3.892 3.0e-03  3 0.5170 4.926 3.4e-03
   4 0.4143 5.755 3.7e-03  5 0.3311 6.417 3.9e-03  6 0.2654 6.948 4.0e-03
 * 1 0.7225 2.445 3.2e-03  2 0.4652 3.375 3.8e-03  3 0.2981 3.972 4.1e-03
 * 1 0.5652 2.130 3.9e-03  2 0.2402 2.611 4.3e-03  3 0.1096 2.830 4.5e-03
 * 1 0.3662 1.732 4.4e-03  2 0.0763 1.885 4.6e-03  3 0.0114 1.908 4.7e-03
-----------------------------------------------------------------------
```

`1+2tau` and `StDev` that appear in the table use truncated sums up to the lags of `t`. The first line marked by `*` refers to the analysis of data averaged in blocks of length 2, the second line marked by `*` in blocks of length 4, etc. The result (rounded) is `<r> = 2.501+-0.005`, to be compared with the following exact results for the example:

```
  <r> = 2.5
  standard deviation of <r> = 4.564e-03
  Var r = 25/108 = 0.2315
```

NOTE: in newer versions, a summary determined from maximum blocked StDev with c_1 included is printed.

Try to run `staticst.c` several times for different **n**'s to see how the method works!

See also utility `autocorr`, 20.10.

14.4   Kinetic quantities from equilibrium molecular dynamics

14.4.1   Diffusivity

Green–Kubo

Diffusivity can be calculated from the velocity-velocity time autocorrelation function (more precisely the covariance) using the Green–Kubo formula

$$D_i = \int_0^\infty \langle \dot{x}_i^{\mathrm{CM}}(0)\dot{x}_i^{\mathrm{CM}}(t)\rangle, \tag{14.4}$$

where $x_i^{\mathrm{CM}}(t)$ is the x-coordinate of the center-of-mass of molecule i. MACSIMUS offers a direct, but (sometimes) costly, calculation of the covariance $\langle \dot{x}_i^{\mathrm{CM}}(0)\dot{x}_i^{\mathrm{CM}}(t)\rangle$ by setting variable `lag.v` (time range in the units of `noint*h`), see also `lag.dim` (1 to record the sum over coordinates x,y,z or 3 to record the all separately) and `lag.nv` (number of molecules). By averaging these covariances (over all molecules of the same species and all three coordinates) and integrating them (use utility runint), the diffusivity can be obtained.

The program unit of diffusivity is $\text{Å}^2/\text{ps} = 10^{-8}$ m^2/s $= 10^{-4}$ cm^2/s.

Einstein

It is possible to calculate the diffusivity from recorded configurations using the Einstein relation. The mean square displacement is

$$\mathrm{msd}(t) = \frac{1}{6}\frac{\sum_i \left\{ m_i[\vec{r}_i(t) - \vec{r}_i(0)]\right\}^2}{\left(\sum_i m_i\right)^2} \stackrel{t\to\infty}{=} \mathrm{const} + D_i t, \tag{14.5}$$

where the sums are over all sites of one molecule.

MACSIMUS calculates $\mathrm{msd}(t)$ if `MSD.mode=1` is set. It is recommended to calculate it from a stored playback file via option `-m1` and variables `reread.*`. The algorithm follows the path of the particles in the periodic boundary conditions so that `dt.plb` in the simulation must not be too long (particles must not travel more than half the box within one frame recorded, which is checked by the algorithm).

Whole calculations should be repeated with different blocks, averaged and possibly corrected for the hydrodynamic tail and finite size effects. Utility `plb2diff` automates the calculations. See Sect. 21.20.

Finite-size effects

The diffusivity obtained by either method is subject of large finite-size errors [23]. These errors are caused by the periodic images traveling in the same direction. The MD diffusivity is thus underestimated. The corrected diffusivity is [23, 24]

$$D = D_{\mathrm{MD}} + \frac{2.8373 k_{\mathrm{B}}T}{6\pi\eta L} \tag{14.6}$$

where η is the shear viscosity. This formula applies to cubic boxes L^3 only.

This correction is large. In order to calculate an accurate value of the diffusivity, both the periodic (MD) diffusivity and the viscosity must be calculated, and typically the uncertainty in viscosity has a higher impact on the result. Since the viscosity calculation itself is not subject to large finite-size effects[24], and the statistical accuracy is low for large numbers of particles, it is recommended to determine the viscosity using a smaller system[1] and the diffusivity using a larger system.

Recently, we tested the above correction for argon and water [25]:

```
Ar
^^

EvdW=-0.2380684 kcal/mol, RvdW=1.910992 AA
T=143.76 (T*=1.2)
rho=1344.2582 kg/m3 (rho*=0.8)

viscosity (Green-Kubo): eta=0.00017543 Pa.s
D is in 1e-9 m^2/s
Dcorr = Dsim + 2.837*k*T / (6*pi*eta*L)
===============================
N method tau/ps  Dsim   stderr Dcorr
-------------------------------
250   B  0.2     4.217 0.019 4.954
250   B  1       4.229 0.022 4.966
250   N  0.2     4.210 0.021 4.947
250   N  1       4.220 0.022 4.957
2000  B  0.2     4.560 0.012 4.928
2000  B  1       4.567 0.011 4.935
2000  N  0.2     4.568 0.013 4.936
2000  N  1       4.578 0.010 4.947
===============================
2000: L=46.21296 AA
250: L=23.10648 AA
N=Nose+Gear
B=Berendsen(+Shake)
```

```
SPCE water
^^^^^^^^^^
T=298.15 K

==================================
N method tau/ps  Dsim   stderr Dcorr
----------------------------------
250   B  1       2.30   0.06    2.84
250   B  1       2.26   0.07    2.80
2000  B  1       2.49   0.10    2.76
2000  B  1       2.56   0.09    2.83
==================================

viscosity (N=250):  0.00058(6) Pa.s
L=19.575161 AA (N=250)

NB: later results, N=300
viscosity=0.00073(4) Pa.s
Dsim=2.390(8), D=2.80(2) [1e-9 m^2/s]
```

14.4.2   Conductivity

Partial molar conductivity is proportional to the diffusivity

$$\lambda_n = \frac{z_n^2 e^2}{k_\mathrm{B} T} D_n = \frac{z_n^2 F^2}{RT} \tag{14.7}$$

where z = charge in e, F = Faraday constant, R = gas constant, and V_m = molar volume. Equivalently we may define the partial conductivity as

$$\kappa_n = \frac{z^2 e^2 N_n}{k_\mathrm{B} T V} D_n = \frac{z^2 F^2 c_n}{RT} D_n \tag{14.8}$$

where N_n is the number of molecules of species n and c_n is the concentration (in mol/dm^3). In program units ($k = 1$) this formula becomes

$$\kappa_n = \frac{N_n D_n z_n^2}{TV}, \tag{14.9}$$

---

[1]In addition, the Green–Kubo route requires a short `noint`.

where the program unit of conductivity is 111.26501 S/m (but the user interface uses SI units). The Green–Kubo formula for the (total) conductivity is

$$\kappa = \frac{V}{k_{\mathrm{B}}T} \int_0^\infty \langle J_x(0) \cdot J_x(t)\rangle \tag{14.10}$$

and the same in the y and z directions (three independent values). The current density $\vec{J}$ is

$$\vec{J} = \frac{1}{V}\sum_i q_i \dot{\vec{r}}_i \tag{14.11}$$

Three covariances $\langle J_a(0)J_a(t)\rangle$, $a \in \{x,y,z\}$, are calculated by cook for `lag.J` set. In addition, a guide how to finish the calculations is printed.

For a successful calculation, `DT=h*noint` must be fine enough and the lag `lag.J` long enough, which is best checked using graphs of covariances and their running integrals.

The corresponding Einstein relation (derived from the current, not current density) reads as

$$\mathrm{mscd}(t) = \frac{1}{6}\left\{\sum_i q_i[\vec{r}_i(t) - \vec{r}_i(0)]\right\}^2 \overset{t\to\infty}{=} \mathrm{const} + k_{\mathrm{B}}TV\kappa t \tag{14.12}$$

The calculations should be performed in the reread mode, see option `-m` and variables `rered.*`. Similarly as for diffusivity, `plb2diff` can be used to calculate mscd and in turn the partial and total conductivities, see Sect. 21.20.

In the non-equilibrium molecular dynamics, the conductivity is given by

$$\kappa = \frac{J_z}{E_z} \tag{14.13}$$

where we assume that only the z-component of the electrostatic field, $E_z =$ `el.E[2]` (in V/m), was set. Since the $J_z =$ `Jz [A/m2]` is reported in SI units, κ will be in S/m.

14.4.3   Viscosity

The Green–Kubo equation for viscosity is

$$\eta_{ab} = \frac{V}{kT}\int_0^\infty \langle P_{ab}(t)P_{ab}(0)\rangle \mathrm{d}t, \ a \neq b \tag{14.14}$$

where the pressure tensor components are for pairwise interactions given by (15.10). Three covariances are calculated by cook for `lag.Pt` set. cook prints a guide, too, and similar rules as for the conductivity apply. If `lag.Pt` is too short, the hydrodynamic tail ($\propto t^{-3/2}$) should be used for extrapolation. A support for off-diagonal pressure tensor components is needed (PRESSURETENSOR=7, see See Sect. 8.2).

It is also possible to calculate viscosity from traceless diagonal components[26]

$$\eta_{aa} = \frac{3}{4}\frac{V}{kT}\int_0^\infty \langle P'_{aa}(t)P'_{aa}(0)\rangle \mathrm{d}t, \quad P'_{aa} = P_{aa} - \frac{1}{3}\sum_{b=x,y,z} P_{bb} \tag{14.15}$$

The following mix of both formulas is recommended[26]

$$\eta = \frac{3}{5}\eta_{\mathrm{off}} + \frac{2}{5}\eta_{\mathrm{trless}}, \quad \eta_{\mathrm{off}} = \frac{1}{3}\sum_{ab=xy,yz,zx}\eta_{ab}, \quad \eta_{\mathrm{trless}} = \frac{1}{3}\sum_a \eta_{aa}. \tag{14.16}$$

Note that η_{ab} is symmetric. MACSIMUS since V1.7p prints also these components (full off-diagonal support and `lag.Pt` are needed).

Note: it is (probably) not possible to calculate viscosity from the Einstein relation obtained by the following "naive integration" of the Green–Kubo formula (see [27], p. 86+).

$$P_{ab} = \frac{1}{V}\left(\sum_{i=1}^{N}\frac{p_{i,a}p_{i,b}}{m_i} + \sum_{i=1}^{N}r_{i,a}f_{i,b}\right) = \frac{1}{V}\left(\sum_{i=1}^{N}\frac{p_{i,a}p_{i,b}}{m_i} + \sum_{i<j}^{N}r_{ij,a}f_{ij,b}\right) \qquad (14.17)$$

$$2t\eta = \frac{V}{kT}\langle [L_{ab}(t) - L_{ab}(0)]^2\rangle \qquad (14.18)$$

where

$$L_{ab} = \frac{1}{V}\sum_{i}r_{i,a}p_{i,b} \qquad (14.19)$$

Limited support present in `plb2diff` is thus meaningless. For a work-around, see[28] (not implemented in MACSIMUS).

14.5   Kinetic quantities from the Einstein relations

Conductivity and autodiffusion measurements by Einstein relation See `plb2diff.c`, 21.20, and `sim/sfdx.c`

NOTE: The Green–Kubo-based conductivity is usually more precise, but the grid has to be short enough, sometimes `2*h`, and the lag long enough, which may be a problem for one-particle diffusivities. The analysis may be rather subjective because a graph has to be analyzed. The Einstein version is very similar if the same lag is used. It becomes safer (less systematic error) but less accurate (larger statistical error) as the lag increases.

14.5.1   Requirements

- Periodic b.c.

- *NVT* or *NPT* ensemble (for *NPT*, see below)

- Stored configurations:

  – either in a plb-file (simname.plb, generated by `cook*` with `dt.plb` placed in the def-file (not get-file)

  – or simname.1,simname.2,... (generated by `cook -r`)

14.5.2   Usage

Copy/link files to another name, then re-run `cook`:

- either with option `-m1` (to read simname.plb); `dt.plb` should math that of the productive run,

- or `-m0` (to read simname.1,...), `dt.cfg` should match that of the productive run,

Input data are:

reread.from First frame (stored configurations) read, default=1

reread.to Last frame read (hint: use `plbinfo` simname.plb to get the max frame!)

reread.by Stride (`reread.by=1` reads every configuration).

Example:

```
for ext in cfg def ; do ln -s SIMNAME.\$ext NEWSIMNAME.\$ext ; done
\# edit NEWSIMNAME.def, create NEWSIMNAME.get
cook* -m SYSNAME.ble NEWSIMNAME SIMNAME.plb
```

with `reread.from=5, reread.to=9, reread.by=2` placed in `NEWSIMNAME.get`, will read frames 5,7,9 from NEWSIMNAME.plb

14.5.3 Results

simname.prt Protocol (is the screen if option `-s`)

simname.m.cp Mean square displacements of the center of mass:

COLUMN 1 mean square displacement for species 0:

```
          1                                    2           2
    msd = ----- SUM {SUM m  [r (t) - r (0)]}  / {SUM m }
          6 N_0  n   i   i   i        i            i   i
```

where m_i = mass of site i, SUM_i is over all sites of species 0, SUM_n is over all molecules of species 0, N_0 = SUM_n 1 = number of molecules of species 0

COLUMN 2 as above, species 1

...

COLUMN nspec+1 The sum over the whole simulation box.

```
          1                      2
   mdif = - {SUM m  [r (t) - r (0)]}
          6   i   i   i        i
```

Note that `mdif=0` but rounding errors because of momentum conservation!

simname.q.cp mean square charge displacement:

COLUMN 1 cumulative mean square charge displacement for species 0:

```
          1                          2
   mscd = - SUM {SUM q  [r (t) - r (0)]}
          6  n   i   i   i        i
```

where q_i = charge of site i, SUM_i is over all sites of species 0, SUM_n is over all molecules of species 0,
NOTE: mscd is SUM while msd is AVERAGE (divided by N_0)

COLUMN 2 as above, species 1

...

COLUMN nspec+1 total mean square charge displacement:

```
          1                           2
   mscd = - {SUM q  [r (t) - r (0)]}
          6   i   i   i        i
```

where the sum is now over the whole simulation box. This is related to the conductivity, see below.

14.5.4   Analysis of results

See also `plb2diff` which makes time averages and automates the following algorithm.

DIFFUSION COEFFICIENTS   Run:

```
  showcp -p SIMNAME.q.cp
```

and estimate the time derivatives

```
  D = d msd(t) / dt
```

Hints:

- Frames are separated by dt.plb*# or dt.cfg*# ps (# denotes the -f option, `-f#` or `-f-#`).

- Several first data should be skipped

- Estimate the slope from from the plots or by linear regression of; the linear regression should use weight 1/t because the error of msd(t) is proportional to sqrt(t).

- And all this should be done for several blocks to have statistics!

D is the diffusion coefficient of given species (column 1 = species 0) in program units [A^2/ps]; in usual units it holds:

- D*1e-8 = diffusion coefficient in m^2/s

- D*1e-4 = diffusion coefficient in cm^2/s

Make also sure that the last column in simname`.q.cp` (`mdif`) is small.

NOTES: 1st 2 columns of simname`.q.cpa` and the plots are incorrectly labeled as `Etot` and `Tkin` instead of `0` and `1`. In addition, the last column in simname`.q.cpa` just repeats the 1st column, thus the "last column in simname`.q.cp`" is actually the second-last in simname`.q.cpa`.

CONDUCTIVITY Run:

```
showcp -p SIMNAME.q.cp
```

and the LAST graph (named `cond`) should be approximately linearly increasing function of time, `cond(t)`; of course, it is not and averaging should be done over different runs or blocks of one run! The conductivity (per unite volume) in program units is:

$$\kappa = \frac{1}{TV}\frac{\mathrm{d cond}(t)}{\mathrm{d}t} \tag{14.20}$$

where T = average temperature in K and V is the volume of the simulation box. In MACSIMUS units:

```
kappa*111.26502 = conductivity in S/m
kappa*1.1126502 = conductivity in S/cm
```

PARTIAL CONDUCTIVITIES kappa_n are obtained from columns 0,1,..nspec-1 in the same way. Note that the total bulk conductivity is a sum of the partial conductivities only if the ions are uncorrelated which holds true in special cases and only approximately. It holds

$$\kappa_n = \frac{D_n z^2 F^2}{RTV_\mathrm{m}}, \tag{14.21}$$

where z = charge in e, F = Faraday constant, R = gas constant, and V_m = molar volume, which in program units ($k = 1$) becomes

$$\kappa_n = \frac{N_n D_n q_n^2}{TV}, \tag{14.22}$$

where N_n = number of molecules of species n, V = volume of the simulation box, q_n = charge of species n.

Note that for monoatomic ions, mscd = $N_n q^2$ msd. For molecules generally mscd(t) $\neq N_n q^2$ msd(t) although the time derivatives should be the same (but statistical noise).

FINAL NOTES See also `util/plb2diff.c`, 21.20!

For NPT ensemble (`tau.P` is nonzero), `cook` produces scaled (by powers of V^(1/3)) quantities and the above factors should be recalculated! However, `plb2diff` does it for you.

14.6 Structure factor

14.6.1 Structure factor for pure simple fluids

The structure factor is in principle the Fourier transform of a radial distribution function, see see Sect. 30. Here, we use the atomic masses to weight different atoms. We define a k-vector by $\vec{k} = \vec{n}/\lambda$, where λ is the wavelength and $\vec{n}$, $|\vec{n}| = 1$, its direction. (This k-vector is common in crystallography, whereas in physics a 'circular' k-vector is more common, $2\pi\vec{n}/\lambda$. In cook V2.6h and older, the circular definition was used, from V2.6i the above 'crystallographic' definition

is used. The definition is noted in the output files.) In a periodic box only some k-vectors are available, $\vec{k} = \vec{k^i}/\vec{L} \equiv (k_x^i/L_x, k_y^i/L_z, k_x^i/L_x)$ and $\vec{k^i}$ is an integer vector. The structure factor of a simple fluid is:

$$S(\vec{k}) = \frac{1}{N}|Q(\vec{k})|^2, \quad Q(\vec{k}) = \sum_j \exp[-2\pi i \vec{k} \cdot \vec{r}_j] \qquad (14.23)$$

and the sum is over all atoms j.

The structure factor is used to detect crystallization and glassy state of liquids.

14.6.2 Structure factor for mixtures

Theory

A sensitivity of atoms to scattering is described by weights, in neutron diffraction 'coherent scattering lengths' b_j. They are generally complex numbers (LIMITATION: MACSIMUS supports only real values). The mixture structure factor is then

$$S(k) = 1 + N\frac{\langle |Q(\vec{k})|^2\rangle - \sum_j b_j^2}{\left(\sum_j b_j\right)^2} \qquad (14.24)$$

where

$$Q(\vec{k}) = \sum_j b_j \exp[-2\pi i \vec{k} \cdot \vec{r}_j] \qquad (14.25)$$

and the sums are over atoms.

Implementation

The scattering lengths b_I should be written 'by hand' to the ble-file to the table of pair potential parameters (Lennard-Jones, Busing-12, ...) after the last column. (If this column is missing, atomic masses m_I are used instead, however, do NOT write b_I instead of masses to the 2nd column!)

Cook has to be run again with option `-f` (see there) to calculate the structure factor from stored configurations. Variables `init` and `no` are the first and last frames processed, respectively. Variable `el.kappa` is the maximum k-vector calculated.

Sphericalized structure factor

The structure factor of fluids is isotropic. A sphericalized structure factor is

$$S(k) = \sum_{k=|\vec{k}|} S(\vec{k}) / \sum_{k=|\vec{k}|} 1 \qquad (14.26)$$

LIMITATIONS: MACSIMUS can calculate a sphericalized structure factor for a cubic simulation cell only.

Sphericalization is requested `el.sf=1` in the data (while running cook with option `-m`). `el.sf=3` in the data specifies the full 3D structure factor.

It is not possible to calculate both sphericalized and 3D structure factors at the same run. Note that a sphericalized structure factor can be obtained from the 3D one provided that the numbers of vectors $\vec{k}$ giving the same k are known (the last column, cf. utility avdata).

See also utility `sfourier`.

14.7   Radial distribution functions

If `rdf.grid` is selected, the program measures the site–site correlation functions, also called radial distribution function (RDF). The results are stored in binary file with extension `.rdf` and may be printed and viewed using program `rdfg`. Site types used for RDF are listed in table `Lennard-Jones` (or similar according to the force field used) in the ble-file in 2nd column (denoted as `atom`). By default there is one RDF for each pair of site types even if this site type appears several times in a molecule (molecules).

WARNING: program `rdfg` (and also `harmg`) uses for site–site correlation functions formulas giving 'incorrect' limits

$$\lim_{r \to \infty} g_{IJ} = 1 \tag{14.27}$$

and not the correct (conforming the NVT definition)

$$\lim_{r \to \infty} g_{IJ} = \begin{cases} 1 - 1/N_I & \text{for } I = J \\ 1 & \text{for } I \neq J \end{cases} \tag{14.28}$$

Formula (14.27) is probably a better approximation of the thermodynamic limit, however, in certain cases (e.g., when integrals of RDFs are calculated), (14.28) should be preferred. In most cases the difference between (14.27) and (14.28) is irrelevant.

Options for RDF are controlled by variables and by file simname.`s-s`.

rdf.grid (0) Grid for calculating the site–site radial distribution functions, in $1/\text{Å}$(i.e., the number of histogram bins per 1 Å). If negative, there is one summary (equally weighted) function for all combinations of site types. If positive, there is a separate function measured for each combination of site types (if simname.`s-s` is missing) or only selected site–site pairs are measured (if simname.`s-s` exists, see below). If `rdf.grid=0`, the radial distribution functions are not measured.

rdf.cutoff (0) Max range for which the site–site radial distribution functions are measured, in A. If `rdf.cutoff=0` then the value of `cutoff` is assumed (default).

rdf.onefour (1) Controls whether 1-4 (or 1-5 if `distance14=4` in the ble-file) are to be included to measured radial distribution functions. E.g., in a united-atom model of butane, CH3-CH2-CH2-CH3, there is a peak on the CH3-CH3 radial distribution function caused by intramolecular CH3-CH3 pairs. Setting `rdf.onefour=0` will eliminate this intramolecular peak.

More options can be specified in file simname.`s-s`. Data in this file are active if `rdf.grid>0`.

Site types are listed in table Lennard-Jones (or similar according to the force field used) in the 2nd column (denoted as `atom`).

File format:

```
 ! define group of atoms of given TYPE:
 ID TYPE SPECIES.ATOM [SPECIES.ATOM ...]
 ID ...
 ! include all pairs of types and stop reading TYPE1-TYPE2 data:
 *
 ! include RDF of TYPE1-TYPE2:
 TYPE1 TYPE2
 ...
```

where

- **ID** Lowercase letter or a decimal digit. The full group identifier is a concatenation ID+TYPE (e.g., aCH3). It is not possible to have different atom types in one group.

- **TYPE** Atom type, see above

- **SPECIES** Species number (numbered from 0 in the order given to `blend` and used in the ble-file)

- **ATOM** Atom (site) number, see the mol-file or table sites (in the corresponding species section) of ble-file.

One group of atoms should correspond equivalent (with respect to symmetry) atoms. All atoms of the same type not listed in any group comprise a 'default' group (with identifier = TYPE without prefix).

Examples of [simname].s-s for a simulation of a united-atom model of pentane CH3E-CH2E-CH2E-CH2E-CH3E (species 0) follow.

To measure only RDF of CH3E-CH2E and CH3E-CH3E pairs:

```
 !type type
 CH3E CH2E
 CH3E CH3E
```

To measure all RDF's so that the central CH2 is distinguished, use:

```
 c CH2E 0.2
 *
```

If, in addition, individual atom pairs are to be selected, the ID's are prepended, e.g.:

```
 c CH2E 0.2
 CH2E CH2E
 CH2E cCH2E
 cCH2E cCH2E
```

14.8   Cluster (oligomer) analysis and bond kinetics

Let us consider bulk fluid simulation of atoms (ions) (or small molecules – not fully implemented) interacting via strong attractions so that it is reasonable to say that two atoms

are bonded if their distance is less than certain limit. Typical example is molten salt like $AlCl_3$ consisting of individual ions. In the melt, the ions are bound to relatively very stable molecules like calculates the clusters (molecules).

14.8.1   Cluster overview

1. Reads stored configurations (simname.plb or simname.1..., see option -f)

2. Calculates bonds between molecules (species), bonds are defined by a distance criterion between selected sites (atoms) in molecules

3. Analyzes connectivity of molecules and splits each configuration into clusters

4. Makes statistical analysis of clusters. Clusters are distinguished by size, stoichiometry, and topology (with the exception of very large clusters)

5. Calculates time development of selected clusters, prints summary of all clusters in all configurations

14.8.2   Compilation and synopsis

cook must be compiled with #define CLUSTERS: can be specified while running configure.sh or directly in simopt.h. The calculation can be performed both during simulation and from stored trajectory. For the latter case, see options -m0,-m1 and variablesreread.from,reread.by,reread.to.

14.8.3   Input data

Cluster calculations are requested by setting cl.mode (see there). You can perform cluster calculation during simulation or from playback, see option -m1 and variables reread.*.

Commented example of sysname.cli:

```
 bonds     ! section defining bonds, end the section by an empty line
 AL CL 2.1 ! create bond between AL and CL in different molecules if |AL-CL|<2.1
 AL S 3    ! AL CL S are ATOM names

 maxn=20 ! If given, cluster counts up to size  maxn  are printed to the
         ! prt-file for each frame analyzed in the format defined by
         ! variable  format  (see below).
         ! Cluster topology is not distinguished here, only sizes matter.

 ! keyword  format  is equivalent to cl.format in input data
 ! formats can be combined; e.g., format=15 will print all four lines
 ! WARNING: there must be integer after =, not a formula (NOT format=2+16)
 format=1  ! (default):
   ! 1st column = number of single molecules in the frame
   ! 2nd column = number of pairs
   ! ...
```

```
       ! maxn-th column = number of clusters of size maxn and bigger
       ! last column before keyword = size of the largest cluster
       ! keyword CLUSTERCOUNT1
       ! number of clusters

   ! format=2  ! as above with different order:
       ! 1st column = size of the largest cluster
       ! 2nd column = number of single molecules in the frame
       ! 3rd column = number of pairs
       ! ...
       ! last column before keyword = number of clusters of size maxn and bigger
       ! keyword CLUSTERCOUNT2
       ! number of clusters

   ! format=4 ! lists individual cluster sizes (from largest to smallest), e.g.:
       ! 222 111 44 44 3 2 1 1 1 CLUSTERCOUNT4 number_of_clusters

   ! format=8 ! lists individual cluster sizes with counts, e.g.:
       ! 222*1 111*1 44*2 3*1 2*1 1*3 CLUSTERCOUNT8 number_of_clusters

   ! format=16 ! print simulation time in front of each line (as column 1)
       ! WARNING: the columns are shifted by 1 to the right

   ! format=32 ! print all clusters (molecule numbers), one line for each cluster:
       ! 1st column = cluster number (numbered from 0)
       ! 2nd column = cluster size (number of molecules in the cluster)
       ! 3rd column = molecule 1
       ! ...
       ! keyword CLUSTERCOUNT32
       ! number of clusters

  maxcluster=15 ! larger clusters are not topologically distinguished
               ! (would be too slow)
               ! equivalent to cl.maxcluster in input data

  mincluster=0  ! ignore all smaller clusters in detailed output (format&32)

  ! topology of named (registered) clusters is in the che-format
  ! (see blend for details):
  ! blank line required after the molecule

  cluster AlCl3 ! named cluster for convergence profile, empty line after section
  Cl    Cl
    \  /
     Al
     |
     Cl


  cluster AlSCN ! another named cluster, Al CL SCN are MOLECULES
```

```
  Al--SCN

$i extra.cl  ! include file (cannot nest)

clusters ! mol- and plb-files will be created for clusters, showing
         ! the connectivity of molecules which are represented by
         ! site[0] only.  Use 'show' to view the cluster structure.
         ! Good for monoatomic molecules, equivalent to bit cl.mode&2 set

configurations ! mol- and plb-files will be created for whole
               ! configurations (frames), full molecules are shown,
               ! but intermolecular bonds are shown between site[0] only
               ! Good for monoatomic molecules, equivalent to cl.mode&4 set

bonddynamics ! record created/broken bonds
             ! may be memory consuming in some cases
             ! equivalent to cl.mode&8 set

colors ! Good for monoatomic molecules only:
       ! data used to make a gol-file when "clusters" specified
       ! The first column is species name (not atom)!
Al MAGENTA 1.1
Cl GREEN 1.5
I YELLOW 1.8
```

Comments:

- A bond is created between different molecules if there exists a pair of sites less than the limit (keyword `bond`) apart.

- Of course, molecules may consist of one atom only. Molecule named "Al" may contain only atom "AL" (note case sensitivity!). "Al" refers to "species Al" in a sysname.ble and thus to files `Al.mol` etc. for blend.

- `maxcluster` is the maximum cluster size (number of molecules) for which the topological analysis is performed. Larger clusters of the same stoichiometry but different structure are not distinguished (even if they are named). Limited by 1024 (more than 20 too slow anyway).

- `maxn` is limited by 65536

- `cluster NAME` defines a named cluster in simplified CHE-format. There must not be any blank line after the `cluster NAME` line.

- If `colors` is given, gol-files will be created for each cluster shown.

- No gol-file is created for `configurations`. If you wish to have proper colors, use `molcfg` and link or copy simname.gol to simname.#.gol

- Equal sign = is optional between a keyword and its value, e.g., `$i=FILE` (bug: FILE must not contain '='). For better compatibility, also $iFILE (without space) is allowed

14.8.4   Results

simname.prtx Final statistics of clusters. The clusters are sorted by size and stoichiometry (we call this "standard order"). If the cluster is named, the name is printed. Then, a list of bonds is printed: atoms are numbered consecutively in the order of the stoichiometry formula. E.g., $Al.Cl_4$ has bonds 0-1 0-2 0-3 0-4 because Al has number 0 and four Cl are numbered 1,2,3,4.

simname.cl.cpa Convergence profile of counts of named clusters. Columns correspond to the named clusters in standard order (not in the order in which the clusters appear in simname.def after ;). Each line corresponds to one configuration analyzed.

simname.#.mol, simname.#.plb #=1,2,... denotes frame number. This applies only if keyword configurations has been specified in sysname.def. To view the configuration with molecule-molecule connectivity, use e.g.:

    show simname.1

*.mol, *.plb To view the clusters. Applies only if keyword clusters has been specified. For instance, Al3.Cl9b.mol or , Al3.Cl9b.plb. The last configuration (position of atoms) encountered is used to show the cluster, each molecule is represented by the 1st atom.

simname.rm.sh Shell script which will remove all files created with keyword clusters: run it as

    sh simname.rm.sh

14.8.5   Bugs and caveats

- The topology analyzer is inefficient—it requires n1! n2!...nN! operations for a cluster with stoichiometry MOL1_n1 MOL2_n2...MOLN_nN. Hence, the practical maximum maxcluster is around 15.

- Molecule-molecule bonds in *.mol are always created between site[0] irrespective of the actual site that passed the distance condition (no problem with one-atom molecules).

14.8.6   Bond kinetics

Files simname.created.cpa and simname.broken.cpa are created:

column 1 time

column 2 created[0] or broken[0] = 2*(# of broken/created bonds since the previous frame)

column 3 created[1] or broken[1], as above but the bond is not counted if the same bond has been or will be created/broken immediately before/after

column 4 created[2] or broken[2], as above, one other change allowed between creation/breakage of the same bond

more etc., according to compile-time switch #define BONDHIST

ALGORITHM:

- For bond definition see the 'cluster analysis' above

- The algorithm works on the basis of a list of bonded molecules attached to each molecule. The current and previous lists are compared and the differences determine the broken and created bonds (column 2).

- In addition, the list of created and broken bonds (at each molecule) is searched for time-ordered sequences created-broken, broken-created, created-broken-created, etc.; sequences with even number of changes (=the bond returns to the same state) are omitted, odd sequences are included as one creation or breakage (for time of the middle change, e.g.. for created-broken-created for the time of the middle breakage). These sequences for created[1] or broken[1] must not be interrupted by creation or breakage of another bond (CreateBondTo77-CreateBondTo13-BreakBondTo77 is not recognized as bond to molecule 77 created and immediately broken).

- For created[2] or broken[2], one change in between is allowed (the above example is recognized and does not count for created/broken bonds to molecule 77)

- Etc., up to created[BONDHIST] or broken[BONDHIST]: BONDHIST-1 changes in between are allowed

- It may happen that created[i] or broken[i], $i > 0$, are odd since the 'does not count' condition may occur for one molecule from the bonded pair, e.g. (A-B C) $\rightarrow$ (A B-C) $\rightarrow$ (A-B C). This looks strange but is not a bug—is a feature.

14.9 Normal modes of vibration

14.9.1 Without constraints

For notation simplicity, let us assume that the local minimum occurs at $\vec{r}_i = 0$ and $U(0) = 0$. We shall also drop the vector symbol ($\rightarrow$): in r_i, index i will thus run over $3N$ components (N particles $\times$ three coordinates). The potential can be expanded in

$$U = \frac{1}{2} \sum_{i,j} r_i A_{ij} r_j, \quad A_{ij} = \frac{\partial^2 U}{\partial r_i \partial r_j}. \tag{14.29}$$

In accord with the notation, A is a $3N \times 3N$ matrix and its values are calculated at the minimum. The Newton equations of motion are

$$\ddot{r}_i = \frac{f_i}{m_i} = -\frac{1}{m_i} \sum_j A_{ij} r_j \tag{14.30}$$

A vibration can be written as (**i** denotes here the imaginary unit)

$$r_i(t) = R_i e^{\mathbf{i}\omega t} \tag{14.31}$$

where R_i are numbers and $\omega = 2\pi\nu$ is the circular frequency. By inserting (14.31) to (14.30) we get

$$-\omega^2 R_i e^{\mathbf{i}\omega t} = -\frac{1}{m_i} \sum_j A_{ij} R_j e^{\mathbf{i}\omega t} \tag{14.32}$$

and hence in the matrix notation

$$\omega^2 R' = A' R' \tag{14.33}$$

with

$$R'_i = \sqrt{m_i} R_i, \quad A'_{ij} = \frac{A_{ij}}{\sqrt{m_i m_j}}. \tag{14.34}$$

Matrix A' is symmetric. We diagonalize it by the threshold Jacobi method. The normal mode calculation is available both in `blend` (see options `-J`, `-M,-N,-P`) and `cook` (see variables `nm.*`).

14.9.2 With constraints

We have to expand the right hand side of the equations o motion (11.3),

$$\ddot{\vec{r}}_i = \frac{1}{m_i}(\vec{f}_i + \vec{f}^c_i), \tag{14.35}$$

at minimum to the first order in the amplitude $\vec{r}_i$.

The forces are no longer zeros at minimum because they may have nonzero components in the direction of constraints. The expansion of forces is then

$$\vec{f}_i = \vec{f}^0_i - \sum_j \overleftrightarrow{A}_{ij} \cdot \vec{r}_j, \quad \overleftrightarrow{A}_{ij} = -\frac{\partial \vec{f}_i}{\partial \vec{r}_j} = \frac{\partial^2 U}{\partial \vec{r}_i \partial \vec{r}_j}, \tag{14.36}$$

where we have adopted the tensor notation: Matrix A is treated as an $N \times N$ matrix of tensors (3×3 matrices); we will use consistently the dot-product symbol $\cdot$ for any sum over components while $\vec{a}\vec{b}$ is a 3×3 tensor. Matrix A is symmetric (whether a matrix of tensors or numbers).

To expand the constraint forces

$$\vec{f}^c_i = \sum_a g_a \frac{\partial c_a}{\partial \vec{r}_i}, \tag{14.37}$$

we have to expand $\partial c_a / \partial \vec{r}_i$ as well as g_a given according to (11.6) by

$$g_a = -\sum_b M^{-1}_{ab} G_b, \quad G_b = \sum_i \frac{\vec{f}_i}{m_i} \cdot \frac{\partial c_b}{\partial \vec{r}_i}, \tag{14.38}$$

To expand g_a we have to expand both G_a (composed of $\vec{f}_i$ and $\partial c_a / \partial \vec{r}_i$) and generally also M. Here we make a simplification valid for rigid models (e.g., water): We will assume that matrix M does not depend on the configuration.

The expansion of constraint derivatives is

$$\frac{\partial c_a}{\partial \vec{r}_i} = \frac{\partial c^0_a}{\partial \vec{r}_i} + \sum_j \frac{\partial^2 c_a}{\partial \vec{r}_i \partial \vec{r}_j} \cdot \vec{r}_j. \tag{14.39}$$

Now we insert expansions (14.36) and (14.39) into (14.38). The absolute terms represent the equilibrium condition, $\vec{f}_i + \vec{f}^c_i = 0$. Linear terms can be collected into equation

$$\ddot{\vec{r}}_i = -\sum_j \overleftrightarrow{B}_{ij} \cdot \vec{r}_j, \tag{14.40}$$

where

$$B = B^0 + B^1 + B^2 + B^3 \tag{14.41}$$

Component B^0 comes directly from the expansion (14.36) of $\vec{f}_i$,

$$\overset{\leftrightarrow}{B}{}^0_{ij} = \frac{1}{m_i} \overset{\leftrightarrow}{A}_{ij}. \tag{14.42}$$

Component B^1 arises from expanding $\partial c_b / \partial \vec{r}_i$ in G_b,

$$\overset{\leftrightarrow}{B}{}^1_{ij} = \frac{1}{m_i} \sum_{a,b} M_{ab}^{-1} \frac{\partial c_a}{\partial \vec{r}_i} \sum_k \vec{f}_k^0 \cdot \frac{\partial^2 c_b}{\partial \vec{r}_k \partial \vec{r}_j} \tag{14.43}$$

Component B^2 is from the expansion of forces in G_b,

$$\overset{\leftrightarrow}{B}{}^2_{ij} = -\frac{1}{m_i} \sum_{a,b} M_{ab}^{-1} \frac{\partial c_a}{\partial \vec{r}_i} \sum_k \frac{1}{m_k} \frac{\partial c_b}{\partial \vec{r}_k} \cdot A_{kj} \tag{14.44}$$

and finally expanding constraints in g_a leads to

$$\overset{\leftrightarrow}{B}{}^3_{ij} = -\frac{1}{m_i} \sum_a g_a \frac{\partial^2 c_a}{\partial \vec{r}_k \partial \vec{r}_j}. \tag{14.45}$$

For bond constraints (11.5) we have formula (11.9) and

$$\frac{\partial^2 c_a}{\partial \vec{r}_i \partial \vec{r}_j} = (\delta_{i,i_a} - \delta_{i,j_a})(\delta_{j,j_a} - \delta_{j,j_a}) \overset{\leftrightarrow}{I}, \tag{14.46}$$

where $\overset{\leftrightarrow}{I}$ is the diagonal unit (identity) 3×3 tensor.

The contributions to B^1 are calculated in a double loop over constraints a, b with contributions

$$\overset{\leftrightarrow}{B}{}^1_{i_a i_b} \;\; += \;\; \frac{1}{m_i} \sum_{a,b} M_{ab}^{-1} (\vec{r}_{i_a} - \vec{r}_{j_a}) \left(\frac{\vec{f}_{i_b}}{m_{i_b}} - \frac{\vec{f}_{j_b}}{m_{j_b}} \right)$$

$$\overset{\leftrightarrow}{B}{}^1_{i_a j_b} \;\; -= \;\; \frac{1}{m_i} \sum_{a,b} M_{ab}^{-1} (\vec{r}_{i_a} - \vec{r}_{j_a}) \left(\frac{\vec{f}_{i_b}}{m_{i_b}} - \frac{\vec{f}_{j_b}}{m_{j_b}} \right)$$

$$\overset{\leftrightarrow}{B}{}^1_{j_a i_b} \;\; -= \;\; \frac{1}{m_j} \sum_{a,b} M_{ab}^{-1} (\vec{r}_{i_a} - \vec{r}_{j_a}) \left(\frac{\vec{f}_{i_b}}{m_{i_b}} - \frac{\vec{f}_{j_b}}{m_{j_b}} \right)$$

$$\overset{\leftrightarrow}{B}{}^1_{j_a j_b} \;\; += \;\; \frac{1}{m_j} \sum_{a,b} M_{ab}^{-1} (\vec{r}_{i_a} - \vec{r}_{j_a}) \left(\frac{\vec{f}_{i_b}}{m_{i_b}} - \frac{\vec{f}_{j_b}}{m_{j_b}} \right).$$

The components of B^2 are calculated in a triple loop over constraints a, b and index j:

$$\overset{\leftrightarrow}{B}{}^2_{i_a j} \;\; -= \;\; \frac{1}{m_{i_a}} \sum_{a,b} M_{ab}^{-1} (\vec{r}_{i_a} - \vec{r}_{j_a})(\vec{r}_{i_b} - \vec{r}_{j_b}) \cdot \left(\frac{\overset{\leftrightarrow}{A}_{i_b j}}{m_{i_b}} - \frac{\overset{\leftrightarrow}{A}_{j_b j}}{m_{j_b}} \right)$$

$$\overset{\leftrightarrow}{B}{}^2_{j_a j} \;\; += \;\; \frac{1}{m_{j_a}} \sum_{a,b} M_{ab}^{-1} (\vec{r}_{i_a} - \vec{r}_{j_a})(\vec{r}_{i_b} - \vec{r}_{j_b}) \cdot \left(\frac{\overset{\leftrightarrow}{A}_{i_b j}}{m_{i_b}} - \frac{\overset{\leftrightarrow}{A}_{j_b j}}{m_{j_b}} \right)$$

The contributions to B^3 are calculated in a loop over constraints a

$$\overset{\leftrightarrow}{B}{}^3_{i_a i_a} \quad -= \quad \frac{1}{m_{i_a}} g_a \overset{\leftrightarrow}{I}$$

$$\overset{\leftrightarrow}{B}{}^3_{i_a j_a} \quad += \quad \frac{1}{m_{i_a}} g_a \overset{\leftrightarrow}{I}$$

$$\overset{\leftrightarrow}{B}{}^3_{j_a i_a} \quad += \quad \frac{1}{m_{j_a}} g_a \overset{\leftrightarrow}{I}$$

$$\overset{\leftrightarrow}{B}{}^3_{j_a j_a} \quad -= \quad \frac{1}{m_{j_a}} g_a \overset{\leftrightarrow}{I}$$

From (14.40) it follows that the fundamental frequencies are given by eigenvalues of matrix B and the normal modes by its eigenvectors. There are two problems to solve. First, matrix B is not symmetric nor symmetrizable. Yet it has (for physical reasons) all eigenvectors real. Second, some modes correspond to motion along the constraints.

The second problem is solved by constructing a linear space perpendicular to the constraints. It is done by orthonormalizing the space of constraints (by the Gram–Schmidt algorithm), adding random vectors and orthonormalizing (if the random vector is almost parallel to the already obtained orthonormal basis, a new one is generated). The resulting space perpendicular to the constraints is represented by a matrix P composed of $3N - n_c$ row $3N$-vectors. The projected matrix (of rank $(3N - n_c) \times (3N - n_c)$) is $B^P = PBP^T$; if its column eigenvectors are R', then the original eigenvectors are $R = P^T R'$.

The first problem is solved by using the Jacobi method to convert B^P into the Schur normal form[2]. This leads to orthonormal matrix U such that $S = U^T B^P U$ is upper triangular matrix. The eigenvalues are then on the diagonal, $\lambda_i = S_{ii}$. However, the columns of U are no loger eigenvectors (as for symmetric B). Let us denote the j-th column of U as u^j (with components $u^j_i = U_{ij}$). The first eigenvector of B^P is $R'_0 = u^0$ (indices start from 0),

$$B^P R'_0 = \lambda_0 R'_0. \tag{14.47}$$

Then the i-th eigenvector can be written using either u^j, $j < i$, or equivantly using previous R'_j, $j < i$:

$$R'_i = u^i + \sum_{j<i} x_j R'_j. \tag{14.48}$$

Let us multiply equation $B^P \cdot R'_i = \lambda_i R'_i$ from left by $R'_k \cdot$, $k < i$. We arrive at a set of i linear equations for x_j, $j < i$:

$$R'_k \cdot B^P \cdot R'_i - \lambda_i R'_k \cdot R'_k = \sum_{j<i} (\lambda_i - \lambda_j) R'_k \cdot R'_j x_j \tag{14.49}$$

or in matrix form

$$\sum_{j<i} C_{kj} x_j = P_k, \quad C_{kj} = (\lambda_i - \lambda_j) R'_k \cdot R'_j, \quad P_k = R'_k \cdot B^P \cdot R'_i - \lambda_i R'_k \cdot R'_i. \tag{14.50}$$

Finally $R = P^T R'$.

---

[2]Alternatively, an `octave` script can be called. This is faster, but more memory consuming.

14.9.3   Harmonic Verlet correction

While numerical verification of the above formulas, it is useful to consider the frequency error of the Verlet ($\equiv$ leap-frog) integrator.

Let us consider a harmonic oscillator

$$\ddot{r} = -\omega_0^2 r, \tag{14.51}$$

where $f_0 = \omega/2\pi$ is the frequency.

Let us insert

$$r(t) = e^{i\omega t}, \quad v(t) = A e^{i\omega t} \tag{14.52}$$

into the leap-frog integration of (14.51),

$$r(t+h) = r(t) + hv(t+h/2), \quad v(t+h/2) = v(t-h/2) - h\omega_0^2 r(t). \tag{14.53}$$

After some algebra we get the equation for ω:

$$1 - \cos(\omega h) = \frac{\omega_0^2 h^2}{2} \quad \text{or} \quad \sin\left(\frac{\omega h}{2}\right) = \frac{\omega_0 h}{2}. \tag{14.54}$$

Approximately for the wavenumber $\tilde{\nu} = f/c$

$$\tilde{\nu} = \tilde{\nu}_0 (1 + \frac{\pi^2}{6} c^2 \tilde{\nu}_0^2). \tag{14.55}$$

Maximum frequency and wavenumber that can be (inaccurately) integrated are

$$f_{\max} = \frac{1}{2\pi h}, \quad \tilde{\nu}_{\max} = \frac{1}{2\pi c h}. \tag{14.56}$$

14.10   Thermodynamic integration from a harmonic crystal

14.10.1   Consistent and inconsistent models

Any model – as classical or quantum molecular dynamics or Monte Carlo simulation based on forces between molecules – is only a better or worse description of reality. A model that can be used in one simulation – as determining a phase equilibriuum using the slab geometry – is a "consistent" model.

On the contrary, there are models which describe different phases using different approximations, for instance: (i) including the Berensen "missing term" in water/vapor equilibrium calculation, (ii) using the electronic continuum approximation (which appears as scaled charges) plus correction terms in condensed phases and no scaling in the gas phase, (iii) using quantum mechanics for a crystal and (semi)classical mechanics for the fluid phases, or even (iv) using the experiment for one phase and calculations for the other. Such models are "inconsistent" and one cannot directly simulate both phases in one box (as droplets, bubbles, nucleation, interfaces).

Here we deal with classical consistent models. Using classical mechanics implies that the results must not depend on the value of the Plack constant nor masses of particles.

14.10.2   Reference state

Although calculations provide the "absolute" chemical potential of a model, it is convenient to use a combination of all constituent molecules, ions and atoms at the ideal gas state at the same temperature and some standard pressure p^{std} (typically 1 bar) as the reference,

$$\tilde{\mu}(p, T; p^{\text{std}}) = \mu(p, T) - \sum_i \nu_i \mu_{\text{id},i}(p^{\text{std}}, T) \qquad (14.57)$$

where $\mu(p, T)$ is the calculated chemical potential (of a crystal) per a molecular formula with stoichiometric coefficients ν_i and subscript $_{\text{id}}$ refers to the ideal gas state (of the same model and the same approximation).

We use classical mechanics, therefore $\tilde{\mu}_{\text{cr}}(p, T)$ depends neither on the value of the Planck constant nor atomic masses; this is actually a strong test of calculations. Particularly, the spectrum does depend on massed of particles so do the partition functions of molecules in gas, but their difference according to (14.57) must be the same. Note that equalizing masses of particles (light hydrogens become heavier, heavy oxygens lighter) not only allows for longer timesteps in simulations, but also decreases the width of the spectrum and increases the precision of normal modes calculations.

For many purposes, the reference chemical potential is taken "inconsistently" from the experiment as the ideal gas value at given T and a standard pressure p^{std}. The chemical potential with respect to this reference is

$$\tilde{\mu}^{\text{expt ref}}(p, T; p^{\text{std}}) = \tilde{\mu}_{\text{cr}}(p, T) + \sum_i \nu_i \left[\mu_{\text{id},i}(p^{\text{std}}, T) - \mu_{\text{id},i}^{\text{expt}}(T, p^{\text{std}}) \right], \qquad (14.58)$$

where $\mu_{\text{id},i}$ is now calculated from (14.71) with real masses and the experimental Planck constant. The difference between both references should be small for a good model.

14.10.3   Thermodynamic functions

Let Q be the semiclassical canonical partition function of a system of N structureless (point) and identical particles,

$$Q = \frac{1}{h^{3N} N!} \int \exp\left[-\frac{E_{\text{tot}}(r^N, p^N)}{kT} \right] \mathrm{d}r^N \mathrm{d}p^N, \qquad (14.59)$$

where h is any constant of dimension energy $\times$ time, k is the Boltzmann constant, T the absolute temperature, $E_{\text{tot}}(r^N, p^N) = E_{\text{pot}}(r^N) + E_{\text{kin}}(p^N)$ is the sum of the potential and kinetic energy, and r^N, p^N denote concisely N vectors of atom positions and momenta, respectively. The integration in positions is over volume V and in momenta over infinite range. No observable quantity (e.g., the Gibbs energy of a crystal with respect to ideal gas reference) must depend on the value of h.

If h is the Planck constant, the above Q is called the semiclassical partition function; it represents a good approximation of the true quantum partition function provided that the products of typical energies and times are much less than h.

It is straightforward to generalize the above formula to molecules described by internal coordinates (e.g., rotating rigid bodies) treated via the classical mechanics as well as to mixtures of such molecules.

The Helmholtz free energy and Gibbs energy are then

$$A = -kT \ln Q, \quad G = A + pV. \tag{14.60}$$

For the sake of the thermodynamic integration, it is advantageous to work with ratio $-A/T$ (called the Massieu potential) because the differential contains quantities easily measurable in simulations, internal energy $U = \langle E_{\text{tot}} \rangle$ and pressure p,

$$\mathrm{d}(-A/T) = \frac{U}{T^2}\mathrm{d}T + \frac{p}{T}\mathrm{d}V, \tag{14.61}$$

This differential form is easy to integrate using simulation-based integrands unless we are close to two singular cases, a low-temperature state limiting to harmonic crystal and a low-pressure state limiting to ideal gas.

14.10.4 Classical crystal

Harmonic crystal

First, let us consider a classical harmonic oscillator in one dimension defined by force constant K. Its potential energy is

$$u(x) = \frac{K}{2}(x - x_0)^2. \tag{14.62}$$

The semiclassical partition function of this oscillator is

$$q = \frac{1}{h} \int_{-\infty}^{\infty} \mathrm{d}x \int_{-\infty}^{\infty} \mathrm{d}p \exp\left[-\frac{u(x) + p^2/2m}{kT}\right] = \frac{kT}{h} 2\pi \sqrt{\frac{m}{K}} = \frac{kT}{h\nu}, \tag{14.63}$$

where ν is the frequency of motion. The semiclassical partition function

$$q = \frac{T}{T_\nu}, \quad T_\nu = \frac{\nu h}{k}, \tag{14.64}$$

where T_ν is the vibrational temperature.

For comparison, the quantum partition function is

$$q = \frac{\mathrm{e}^{-T_\nu/2T}}{1 - \mathrm{e}^{-T_\nu/T}}. \tag{14.65}$$

A crystal in the harmonic approximation is characterized by a set of N_{f} fundamental frequencies; for N sperically symmetric molecules in periodic boundary conditions with zero total momentum it holds $N_{\text{f}} = 3N - 3$. The crystal partition function is then

$$Q_{\text{harm}} = \prod_{i=1}^{N_{\text{f}}} \frac{T}{T_{\nu_i}} \tag{14.66}$$

and the Helmholtz energy is

$$A_{\text{harm}} = -kT \ln Q_{\text{harm}} = N_{\text{f}} kT \ln \frac{\overline{T}_\nu}{T}, \quad \overline{T}_\nu = \left(\prod_{i=1}^{N_{\text{f}}} T_{\nu_i}\right)^{1/N_{\text{f}}}, \tag{14.67}$$

Table 14.1: Harmonic sinjarite crystal $(CaCl_2.2H_2O)_{96}$ at 365 K. eq is the factor for equalizing masses within a molecule (for eq = 1 the masses of H and O are the same), $\overline{T}_{nu}$ is the geometric average of vibrational temperatures, a_{harm} is given by eq. (14.67), $a_{harm}^* = a_{harm}(N_f + 3)/N_f$, g_{id} is the sum on the right hand side of (14.57). The calculation uncertainties are on the order of the last given digit.

eq	$\overline{T}_{nu}$	a_{harm}	g_{id}	$a_{harm} - g_{id}$	$a_{harm}^* - g_{id}$
1	K	J/mol	J/mol	J/mol	J/mol
0.0	370.6504	977.6	−269304.5	270282.1	270283.5
0.2	338.6379	−4770.5	−275052.7	270282.2	270275.1
0.4	322.6019	−7857.6	−278140.1	270282.5	270270.8
0.6	312.9599	−9788.6	−280071.0	270282.4	270267.8
0.8	307.0343	−11005.0	−281287.4	270282.4	270266.0
1.0	303.9410	−11649.4	−281931.8	270282.4	270265.0
0.8	307034.8	428573.6	158946.2	269627.4	270266.1

where $\overline{T}_\nu$ is the geometric mean of the vibrational temperatures. This potential diverges as temperature approaches zero.

No divergence occurrs if the vibrations are treated by quantum mechanics, as in the Debye model. One would be tempted to use quantum mechanics to calculate the partition function. However, such an approach is not consistent with classical simulatios: Equilibria calculated from quantum-based crystal would not be consistent with direct simulations of crystals in a solution or melt.

In the ideal gas reference the number of the degrees of freedom will be $3N$ not decreased by 3 for the periodic boundary conditions. This introduces an inconsistency because the final result in the classical mechanics must not depend on the value of Planck constant h and in the classical Gibbs energy we always have term $N_f kT \ln h$. Finite-size error on the order of $\mathcal{O}(1/N)$ is inevitable, but in this inconsistency the error would depend on the value of h which is "arbitrary" in the classical mechanics. A natural solution is to multiply A_{harm} by factor $(N_f + 3)/N_f$. However, an $\mathcal{O}(1/N)$ dependence on the masses of the atoms is introduced, which we consider as a minor problem.

Since the calculation of $\overline{T}_\nu$ is relatively easy, we extrapolate (see later) $\overline{T}_\nu$ to the thermodynamic limit and calculate the Helmholtz energy per one molecule as

$$a_{harm} = n_f kT \ln \frac{\overline{T}_\nu}{T}, \tag{14.68}$$

where now n_f is the number of degrees of freedom per a chemical formula; for models without constraints it holds $n_f = 3 \sum_i \nu_i$, for rigid water $n_f = 6$.

Real crystal

In order to calculate the Helmholtz energy of real crystal with simulation-based energy, we will start from the harmonic crystal at temperature T, integrate down to zero, and then integrate

back with the simulation internal energy[29]. (A similar trick is used to calculate the fugacity of real gas.)

$$
\begin{aligned}
A_{\mathrm{cr}} &= A_{\mathrm{harm}} + E_0 - cT \int_0^T \frac{E_{\mathrm{tot}} - E_0 - E_{\mathrm{tot,harm}}}{T^2} \mathrm{d}T \\
&= cA_{\mathrm{harm}} + E_0 - T \int_0^T \frac{E_{\mathrm{pot}} - E_0 - N_{\mathrm{f}}kT/2}{T^2} \mathrm{d}T
\end{aligned}
$$

because $\langle E_{\mathrm{tot,harm}} \rangle = N_{\mathrm{f}}kT$, $\langle E_{\mathrm{kin}} \rangle = N_{\mathrm{f}}kT/2$ for both crystals, and $\langle E_{\mathrm{pot,harm}} \rangle = N_{\mathrm{f}}kT/2$. The potential energy at zero temperature, denoted as E_0, is added.

The integrand is no longer singular. However, precision is lost close to $T = 0$ because the standard deviation in the potential energy is proportional to T so is the statistical error (with the same trajectory length). Consequently, the statistical error in the integrand is proportional to $1/T$. This is the same behavior as for the gas at pressure approaching zero (unless we know the second virial coefficient). Finally, the Gibbs energy of the crystal is

$$
G_{\mathrm{cr}} = A_{\mathrm{cr}} + pV. \tag{14.69}
$$

14.10.5 Gas and liquid

Ideal gas

The classical partition function of (generally molecular) ideal gas is

$$
Q = \frac{1}{N!} \left(\frac{Vq}{\Lambda^3} \right)^N, \quad \Lambda = \frac{h}{\sqrt{2\pi mkT}}, \tag{14.70}
$$

where q is the internal partition function. The Gibbs energy is then

$$
\begin{aligned}
G_{\mathrm{id}} &= NkT \left[\ln \frac{ph^3}{N(kT)^{5/2}(2\pi m)^{3/2}q} + \frac{\ln N!}{N} + 1 \right] \\
&= NkT \left[\ln \frac{ph^3}{(kT)^{5/2}(2\pi m)^{3/2}q} + \frac{\ln(2\pi N)}{2N} + \mathcal{O}(N^{-2}) \right],
\end{aligned}
$$

where we used the ideal gas equation of state, $pV = NkT$. Term $\ln(2\pi N)/2N$ vanishes in the thermodynamic limit, thus the chemical potential (per molecule) is

$$
\mu_{\mathrm{id},i} = kT \ln \frac{ph^3}{(kT)^{5/2}(2\pi m_i)^{3/2}q_i} = kT \ln \frac{p\Lambda_i}{kTq_i}. \tag{14.71}
$$

where subscript i denotes species.

For instance, for rigid clasical water the semiclassical partition function includes only rotations,

$$
q = \frac{\sqrt{\pi}}{\sigma} \prod_{a=\{x,y,z\}} \left(\frac{2I_a kT}{\hbar^2} \right)^{1/2} = \frac{(2\pi)^{7/2}}{h^3} (kT)^{3/2} \sqrt{I_x I_y I_z}, \tag{14.72}
$$

where the symmetry number is $\sigma = 2$).

Mixture of ideal gases

In the Boltzmann statistics, the partition function of a mixture is a product of partition functions of molecules. Thus

$$G_{\mathrm{id}} = \sum_i N_i \mu_{\mathrm{id},i} + NkT \sum_i x_i \ln x_i + \frac{kT}{2} \sum_i \ln(2\pi N_i), \tag{14.73}$$

where N_i is the number of molecules of species i, $N = \sum_i N_i$, and $x_i = N_i/N$ is the mole fraction. The last term is the estimated finite-size error.

Real gas

The corection for nonideal behavior is given by the integral of the volume difference from the ideal gas mixture over pressure at constant T,

$$G = G_{\mathrm{id}} + \int_0^p \left(V - \frac{NkT}{p} \right) \mathrm{d}p. \tag{14.74}$$

The integrand is not singular; since it limits to BN, where B is the second virial coefficient, in simulation we may avoid precision loss by calculating B and using it as the constraint in the low-pressure limit.

Crystal–liquid equilibrium

For calculating melting temperatures of pure crystals (ice, rock salt, etc.), the chemical potentials equal the Gibbs energy divided by the number of molecules. In general mixtures (e.g., to calculate solubilities of crystals in water), we need the chemical potentials of species,

$$\mu_i = \mu_{\mathrm{id},i} + kT \ln x_i \int_0^p \left(\overline{V}_i - \frac{N_i kT}{p} \right) \mathrm{d}p, \tag{14.75}$$

where $\overline{V}_i$ denotes the partial molar volumes of species. Particularly, having crystal with composition given by stoichiometric coefficients ν_i, we need combination

$$\mu_{\mathrm{sol}} = \sum_i \nu_i \mu_i \tag{14.76}$$

and therefore $\sum_i \nu_i \overline{V}_i$. It means to run (at least) two NpT simulations with different compositions. Alternatively, the Kirkwood–Buff integrals can be used.

To reach liquid, we have to add thermodynamic integration over temperature and pressure around the critical point to avoid gas–liquid phase transition (there are other options, eg.g, to start from Lennard-Jones liquid and transmute Lennard-Jonesium into molecules we need). The integration over temperature requires partial molar enthalpies,

$$\frac{\mu_{\mathrm{sol}}(T_2)}{T_2} - \frac{\mu_{\mathrm{sol}}(T_1)}{T_1} = \int_{T_1}^{T_2} \frac{\sum \nu_i \overline{H}_i}{T^2} \mathrm{d}T. \tag{14.77}$$

Since the composition of the liquid (solution) may be different, we equate the chemical potential of crystal

$$\mu_{\mathrm{cr}} = \sum_i \nu_i \frac{G_{\mathrm{cr}}}{N} \tag{14.78}$$

and the chemical potential of the solution, μ_{sol}, integrated to the same temperature and pressure. Note that $x_i = N_i/N \neq \nu_i / \sum_i \nu_i$ unless we are in an eutectic point.

14.10.6   Finite-size effects

One dimension

To derive the functional form of the finite-size effects of A_{harm}, let us first consider a periodic chain of N atoms (one-dimensional crystal) which may vibrate by N modes with frequencies approximately given by $\nu = cn/Na$ (double degenerate), $n \in \{1, \ldots N/2\}$, where a is the lattice constant and c the speed of sound. We assume that c is a constant, N is even, and neglect momentum conservation. The Helmholtz energy (14.67) is

$$
\begin{aligned}
A_{\text{harm}} &= -2kT \sum_{n=1}^{N/2} \ln \frac{NakT}{hcn} & (14.79) \\
&= -kTN \ln \frac{akT}{hc} - 2kT \sum_{n=1}^{N/2} \ln \frac{N}{n} & (14.80) \\
&= -kTN \ln \frac{akT}{hc} - 2kT(N \ln N - \ln(N/2)!) & (14.81) \\
&\approx -kTN \ln \frac{2akT\text{e}}{hc} - \ln(\pi N), & (14.82)
\end{aligned}
$$

where we used the Stirling asymptotic expansion of factorial with terms $\mathcal{O}(1/N)$ and higher neglected. The finite-size correction per particle is then $\ln(\pi N)/N$, as a direct consequence of term $\ln \sqrt{2\pi N}$ in the Stirling formula.

A more accurate calculation should take into account dispersion. To the second order it holds $\nu = (1 - C(n/N)^2)cn/Na \approx cn/Na \exp(C(n/N)^2)$, where C is a constant independent on N. The term to be added to A_{harm} is

$$
\sum_{n=1}^{N/2} C \left(\frac{n}{N} \right)^2 \approx \frac{C}{8} \tag{14.83}
$$

for large N ($1/N$ neglected), leading to the correction on the order of N^{-1}.

In addition, the number of degrees of freedom is $N - 1$ because of momentum conservation. This changes the estimated Helmholtz energy by a constant and the finite-size error by an $\mathcal{O}(1/N)$ term.

Three dimensions

Let us estimate the finite-size correction for a cubic crystal in periodic boundary conditions. The key term of the Helmholtz energy, A_{harm}, contains the sum over integer vectors (positive coordinates) up to certain maximum $n = |\vec{n}| < m \propto N^{1/3}$,

$$
S = \sum_{\nu} \ln \frac{1}{\nu} = \text{const} N - \sum_{\vec{n}} \ln n. \tag{14.84}
$$

The above formula is based on two approximations: (i) the speed of sound does not depend on the direction (this is not generally true even for cubic crystals), and (ii) we have to sum over both longitudal and transversal waves. While enlarging N and m (and dividing by N) we get finer grid; in turn, the finite-size correction equals the sum S minus the integral of the same argument and in the same bounds.

In order to calculate the correction, we first shift the integration bounds by 1/2; in fact, the upper bound is slightly less because of the conserved momenta. This will produce terms proportional to less than $m^2 \ln m$ (for $n_x, n_y, n_z = 1$) and to $m^2 \ln m$ (for $n = m$). The remaining error is estimated using formula

$$\int_{-1/2}^{1/2} f(x) \mathrm{d}x - f(0) \approx \frac{f''(0)}{24} \tag{14.85}$$

valid for small higher derivatives. After applying this formula in three coordinates (the second derivative leads to a Laplacian Δ), replacing the sums into integrals, and using spherical coordinates, we get a term proportionala to

$$\int_{\mathrm{const}}^{m} \Delta \ln r \, 4\pi r^2 \mathrm{d}r = 4\pi m - \mathrm{const.} \tag{14.86}$$

The leading terms are then $m^2 \ln m / N$ and smaller m/N.

This is in disagreement with numerical tests where such large terms have not been detected. The formula consistent with data is

$$\mu_{\mathrm{harm}}(N) = \mu_{\mathrm{harm}}(\infty) + \frac{a \ln N}{N} + \frac{b}{N}, \tag{14.87}$$

where constants a, b are fitted by the least-square methods with weights proportional to N^2. In addition, we extend this formula to non-cubic crystals provided that the shape (aspect ratios) is preserved. Interestingly, a lattice model of ice exhibits the same finite-size behavior [J. Kolafa, JCP 140, 204507 (2014)].

14.10.7 Miscelaneous notes

Massieu and Planck potentials

Slightly simpler expressions can be obtained using the Massieu and Planck potentials, respectively,

$$\Phi = S - \frac{U}{T} = -\frac{A}{T} = k \ln Q, \quad \mathrm{d}\Phi = \frac{U}{T^2}\mathrm{d}T + \frac{p}{T}\mathrm{d}V, \tag{14.88}$$

$$\Xi = \Phi - V\frac{p}{T} = -\frac{G}{T} = k \ln Q - \frac{pV}{T}, \quad \mathrm{d}\Xi = \frac{H}{T^2}\mathrm{d}T - \frac{V}{T}\mathrm{d}p. \tag{14.89}$$

Possible extension to integration over pressure

We define the reference as a limiting behavior for $T \to 0$ which can be then extended to a harmonic crystal with linear compressibility. Its Planck potential is

$$\Xi_{\mathrm{harm}} = -\frac{E_0 + pV_0(1 + \alpha_0 T)}{T} + N_{\mathrm{f}} k \ln \frac{T}{\overline{\overline{T}}_\nu}, \tag{14.90}$$

where $H_0 = E_0 + pV_0$ is the enthalpy at $T = 0$, $\alpha_T = (1/V)(\partial V/\partial T)_p$ is the thermal expansion coefficient, and $\overline{T}_\nu^{N_{\mathrm{f}}} = \prod T_{\nu_i}$. The full Planck function is then given by integration over T at constant p

$$\Xi = \Xi_{\mathrm{harm}} + \int_0^T \frac{H - H_0 - N_{\mathrm{f}} k T - pV_0 \alpha_0}{T^2} \mathrm{d}T, \tag{14.91}$$

where H is available from an NpT simulation. Note that the integrand is not singular.

or per mole

$$\mu_{\mathrm{m}} = \mu_{\mathrm{id,m}} + \int_0^p \left(V_{\mathrm{m}} - \frac{RT}{p} \right) \mathrm{d}p. \tag{14.92}$$

14.10.8   Enthalpy

The enthalpy is

$$H = E_{\mathrm{pot}} + E_{\mathrm{kin}} + pV + \text{cutoff corrections} \quad [\mathrm{J/mol}]. \tag{14.93}$$

In cook version V4.3m and newer, hints are also printed in the prt-file how to calculate enthalpy from other columns in the cp-file.

In cook version V4.3l and newer:

NPT For `tau.P` $\neq 0$, $p = $ P (barostat parameter). For `thermostat="MTK"`, the enthalpy variance is related to the isobaric heat capacity, $C_P = \mathrm{Var} H / RT^2$.

NVT and NVE For `tau.P` $= 0$, p is the configurational pressure.

BUGS: In cook version V4.3k and older, always $p = $ configurational pressure and cutoff corrections have been added twice. In this case, calculate the enthalpy by.

```
showcp -a -b1 SIMNAME.cp
tabproc C+a*B+b*D < SIMNAME.cpa
where
  a = R*No.f/2 # for No.f, see SIMNAME.prt, and R=8.314462618
  b = P*RHOX*6.02214076e-07,
where RHOX is the number from the following formula found in
SIMNAME.prt in the following line (example):
  rho = 16981814.9518 / V  (for rho in kg m^-3 and V in AA^3)
```

Chapter 15

Special versions

I should write more about boundary conditions in `cook`. Now, see Sect. 8.2 for an overview. Also comments in file `simopt.h` may be useful. Only a few particular versions and special terms are listed below; some details may be out-of-date.

15.1 Fixing positions of selected atoms

Positions of selected atoms can be fixed via harmonic springs, which is requested by option `k#` with a nonnegative argument. See also see Sect. 15.12. The potential is

$$U_{\text{fix}} = \sum_i \frac{K}{2}(r_i - r_i^{\text{fix}})^2. \tag{15.1}$$

Note that a harmonic potential elsewhere is written without factor $1/2$. File <u>simname.fix</u> defining sites to be kept fixed is needed:

```
# this is comment
! this as well
! the following line specifies that site # 2 will be fixed:
2
! the following line specifies that sites 5 6 7 8 will be fixed:
5-8
! the following line specifies that site 12 will be fixed
! to the position x,y,z given (if no x,y,z are given, the positions
! are taken from the initialization)
12  5.5 6.6 7.7
! then the initial file written while init>=3 is not needed.
! * Site numbers for the first molecule can be found in mol-file or
!   ble-file (table sites), otherwise have to be calculated
! This is incorrect line:
2 3
```

This file has the same format as <u>molname.keep</u> or <u>molname.mark</u> from `blend` and can be copied (however, atom positions are not present).

Note that the format of file <u>simname.fix</u> for ANCHORing (negative option `-k`) is different!

- Force constant K = option `-k` (if positive). K is in units K/Å². Reasonable values are probably in range 100–10000. Too low values do not keep atoms in positions firmly enough, too large values cause bad energy conservation and other artifacts.

- (New in V3.6k) Option `-k0` fixes the atom positions exactly in the place given. It is similar to anchoring (version ANCHOR) except that ANCHORing also measures the force and (thus) has several limitations.

- A single atom of mass M g/mol and `-kK` will oscillate with period $\sqrt{m/K} \times 6.89$ ps.

- Value `-k-16` [default] turns off any fixing and file simname.`fix` is ignored.

- For negative values, ANCHOR version is required, see Sect. 15.12.

- If a line containing keyword `fix` is present in simname.`cpi` and option `-k` is positive, energy U_{fix} is recorded in the convergence profile.

Initialization (was changed in V2.4j):

- If simname.`fxc` exists, it is read and the positions saved there are used as fix locations.

- If simname.`fxc` does not exist, the positions are taken from the configuration at job start.

- If, in addition, atom positions are present in file simname.`fix`, they replace the positions given above.

In any case, file simname.`fxc` is written to be used next time. In the next start of `cook -kK`, $K > 0$, simname.`fxc` is used; it may be modified by coordinates specified in simname.`fix`

Conserved quantities: Presence of non-symmetric potential causes the momentum and angular momentum (for free b.c.) not to be conserved. For general 3 or more fixed atoms, this is correctly taken into account in calculating the number of degrees of freedom (needed to evaluate `Tkin`). In addition, correcting the momentum and angular momentum for numerical errors is turned off (unless `drift` is specified)

Bugs: for 1 or 2 atoms to be fixed (or linear set of atoms), the number of degrees of freedom calculated and correcting the angular momentum is incorrect and must be set using `drift` and `conserved`.

Tested with FREEBC only, but in principle should work with other b.c. too

15.2   Notes on water models

There are THREE versions of TIP3 water:

1. Flexible TIP3 water, as used in `blend`, with small intermolecular H-O and H-H Lennard-Jones terms to prevent H-O charge singularity. Because of flexibility, the H-bond energies are lower (dimer energy = -6.8322 kcal/mol). To use this model also in `cook`, do not use option `-h` in `blend` and run `cook -u9999 -x`.

2. Rigid TIP3 water, with the same intermolecular H-O and H-H Lennard-Jones terms as above. This is used in `cook` when *.ble file has been created by `blend -h` AND `cook -x` (and no `-uK` option selecting flexible terms is used). Dimer energy = −6.5920 kcal/mol. It is recommended to use this model with the random initializer (`init=3="random"`) to prevent singularities.

3. Rigid 'original' TIPS3 water: `blend -h` needed and no `-x` nor `-uK` is used. If possible, optimized code is used (using the optimized code is suppressed by `cook -x0`; the numerical results should be the same). Dimer energy = −6.5390 kcal/mol.

As regards other water models, they are usually defined without H-H and H-O Lennard-Jones terms. There are, however, optimized and unoptimized versions of TIP4P and ST2. (Note: TIP4P dimer energy = −6.2345 kcal/mol).

- `cook` prints a warning when it recognizes water and `blend -h` has not been used.

- `blend` without `-h` and `cook -x -u9999` will use the flexible water also in MD. This is less efficient.

- `cook` need not recognize water (to use optimized code) that has been added to the system in other way than via 'blend hoh' and consequently uses the general code (with H-O and H-H terms). The molecule is still rigid (if `blend -h`)

15.3   Cut off electrostatic forces

Instead of the Ewald summation (which can be regarded as more accurate, but is more costly), the simulation in periodic boundary conditions can neglect electrostatic forces beyond certain `cutoff`. This version of `cook` is requested by `#define COULOMB 2` or `#define COULOMB 0` in `simopt.h`.

The $1/r$ term in the Coulomb energy is replaced by

$$\frac{1}{r} \approx \begin{cases} 1/r - \texttt{shift}, & \text{for } r < \alpha\texttt{cutoff} \\ (r - \texttt{cutoff})^3 (A\,\mathrm{FA}(r) + B\,\mathrm{FB}(r)), & \text{for } \alpha\,\texttt{cutoff} < r < \texttt{cutoff} \\ 0, & \text{for } \texttt{cutoff} < r \end{cases} \qquad (15.2)$$

where $\mathrm{FA}(r) = 1$ and $\mathrm{FB}(r) = r$ are the base functions.

The electrostatic force is thus neglected beyond the cutoff, shifted at short separations, and smoothly interpolated in between.

The default value of $\alpha = 0.7$; the optimum values are around 2/3, which comes from a comparison with the method by Fennell and Gezelter[51][1]. The former default 0.9 is not good [30]. The `cutoff` must be less than one half of the box size.

For `#define COULOMB 2` this cutoff electrostatic is implemented by quadratic splines, for `#define COULOMB 0` directly by formulas. In fact, the COULOMB=2 version it makes use of the structure of the Ewald summation, where the k-space part is turned off (K=0) and the erfc functions (implemented by rational splines, `#define COULOMB -1`) are replaced by quadratic splines that are more suitable. Both the COULOMB=2 and COULOMB=0 are approximately of the same speed, but this may depend on the architecture.

---

[1]Available by #define COULOMB 3

Table 15.1: Comparison of the Ewald and cut-off electrostatics for SPC/E water, cubic box with 500 molecules, temperature 300 K (Berendsen `tau.T=0.5`), density $997\,\mathrm{kg\,m^{-3}}$, time 110 ns, and cutoffs `LJcutoff=cutoff=12.3318` Å. Pressure P is based on the virial theorem whereas P_tens uses the corresponding force.

method	ϵ_r	T_kin	E_pot	P	P_tens
	[K]	[J/mol]	[MPa]	[MPa]	
Ewald	72.79(58)	299.9373(4)	−46714.40(83)	−2.835(88)	−2.743(87)
cut-off	72.01(59)	299.9373(4)	−46756.51(86)	−13.896(88)	−4.904(89)

The cutoff electrostatic should not be used for systems with free charges unless the cutoff is longer than the Debye screening length. It is acceptable for systems with partial charges but neutral groups, possibly also for charged systems solvated e.g. by water. Table **??** shows a comparison of both methods for SPC/E wateat ambient conditionsr. Note that P and P_tens would converge to the same value with higher Ewald precision irrespective of the calue of `cutoff` (especially more terms in the k-space are needed), for the cut-off electrostatics we need `cutoff` $\to \infty$. It is seen that the dielectric constants are indistinguishable, the potential energy and pressures are lower for the cut-off version, and the pressure based on the pressure tensor (not using the virial theorem) is more accurate.

The pressure calculation assumes that the virial of electrostatic forces is the same as the electrostatic energy. This holds exactly for the Ewald summation (and therefore the virial pressure calculated by the virtual volume change [see variable `dV`] is the same as the virial calculated directly from pair interactions). However, this is an approximation only for the cut and shifted electrostatic. The virial pressure *is not consistent* with the used potential; this need not mean that it is worse approximation of the full electrostatic system, though.

15.4   Gravity simulation (STARS)

In this version, requested by `#define STARS`, the sign of the Coulomb interactions is changed to attraction of equal 'charges'. To be used with FREEBC. For instance, a globular star cluster can be simulated.

15.5   Forces by Gaussian

FREEBC required, under development, some quantities are not supported, unfinished guide.

1. Prepare your simulation (initial configuration = cfg-file) using standard force-field simulation in the free boundary conditions. The atom names must be understandable by GAUSSIAN input, otherwise the ble-file must be edited. For water, SPC/E is OK.

2. File simname`.com0` must contain the following lines, in this order:

   1   #-line (with keywords) for calling Gaussian without measurements, single-point calculation, NoSymmetry, forces. The log-files (Gaussian output) will be erased.

    2   #-line for calling Gaussian with measurements (the last step of the noint-cycle). Typically, Prop may be present. These log-files will be numbered and stored.

    3   Line with charge and multiplicity (two integers, no leading space).

    4   %-line, typically with the %mem= keyword. The predictor requires guess=(Read,TCheck) and %chk=SIMNAME.

  5-8 Optionally more %-lines (but not %Nproc=, see below), not more than 8.

  more @-lines define the predictor (see cook/gaussian.c).

Using this information, the Gaussian input com-files are generated. The NSLOTS environment variable is used to generate the %Nproc= statement. Example:

```
# B3LYP/6-31G* SP guess=(Read,TCheck) NoSymmetry force EmpiricalDispersion=GD3BJ
# B3LYP/6-31G* SP guess=(Read,TCheck) NoSymmetry force EmpiricalDispersion=GD3BJ Pr
0 1
%mem=16000MB
%chk=xnewtest
@2
```

3. The following environment variables must be exported: `GAUSS_EXEDIR` and `GAUSS_EXE`; also, `LD_LIBRARY_PATH` (the same as `GAUSS_EXEDIR`). Optionally scratch. Example of a job for the "Grid Engine" batch system:

```
#/bin/bash
#$ -q "nq-16-8@n29.vscht.cz"
#$ -cwd
#$ -V
#$ -j y
#$ -pe shm 4
export GAUSS_SCRDIR=/scratch/$USER/gaussian_$$
mkdir -p $GAUSS_SCRDIR
export GAUSS_EXEDIR=/SOMEWHERE/G16/g16
export LD_LIBRARY_PATH=$GAUSS_EXEDIR
[ -e gaussian ] || ln -s $GAUSS_EXEDIR/g16 gaussian
[ -e formchk ] || ln -s /usr/local/programs/custom/PHOTOX/bin/formchk formchk
[ -e unfchk ] || ln -s /usr/local/programs/custom/PHOTOX/bin/unfchk unfchk

echo NSLOTS=$NSLOTS
./cookgaussian spce -x0 -u9999 spce3
```

Note that `cookgaussian` calls `./gaussian`, `./formchk`, `./unfchk`. The latter two are needed with prediction (@-codes). This job can be submitted via command `qsub`. Parameters: `-q` requests a queue, `-cwd` changes the working directory, `-V` copies the environment, `-j y` concatenates stderr and stdout, `-pe shm 4` requests 4 threads (NSLOTS exported). Note that `-u9999` requests flexible bonds and `-x0` turns off standard optimized water models.

15.6   Polarizable dipoles

For a general theory and formulas, see Appendix 25.

While point induced dipoles are used in `blend`, they are approximated by macroscopic dipoles in `cook`. A polarizable site i (with an optional charge q_i) accepts and additional charge $-Q$ and there is an auxiliary site $\delta\vec{r}$ apart with charge $+Q$:

```
  q_i-Q     +Q            E
    o------o           ---->
       dr
```

It holds

$$\vec{\mu}_i = Q\delta\vec{r}_i = \overset{\leftrightarrow}{\alpha}_i \cdot \vec{E}_i = \overset{\leftrightarrow}{\alpha}_i \cdot \vec{F}_i/Q \tag{15.3}$$

or

$$\delta\vec{r}_i = \overset{\leftrightarrow}{\alpha}_i \cdot \vec{F}_i/Q^2 \tag{15.4}$$

where $\vec{E}_i$ is the electric field acting on site i, $\vec{F}_i$ is the electric force, $\vec{\mu}_i$ is the induced dipole, and $\overset{\leftrightarrow}{\alpha}_i$ is the polarizability tensor. (See also Appendix 25.) The larger Q, the better approximation of point dipoles, but lower numerical accuracy; $Q = -1000e$ seems to be a good tradeoff.

Particularly, equation (25.19) is approximated by the numerical derivative

$$\vec{f}_{\text{rep},i} = -\kappa_i Q \delta\vec{r}_i \cdot \vec{\nabla}_i \vec{f}_{ij} \approx -\kappa_i Q[\vec{f}_{ij}(\vec{r}_i + \delta\vec{r}_i) - \vec{f}_{ij}(\vec{r}_i)]. \tag{15.5}$$

Equation (15.3) should be valid for all polarizable atoms in the system. Since the field $\vec{E}$ depends on the induced dipoles on other atoms, we have a set of equations for the so-called self-consistent field (SCF). Let us write it shortly as

$$\mathbf{D} = \text{Iter}(\mathbf{D}) \tag{15.6}$$

where $\mathbf{D}$ is the set of induced dipoles (or equivalently $\delta\vec{r}$'s) and operator Iter() denotes one iteration. In this iteration, all electrostatic forces are calculated (using the fixed set of atomic charges and current induced dipoles) and from them the new induced dipoles.

If the iterations diverge, it is called the 'polarization catastrophe'. It may happen if the polarizability is too large and/or atoms are too close together (e.g., at too high temperature).

The SCF equation can be rewritten as

$$\mathbf{D} = \omega\text{Iter}(\mathbf{D}) + (1 - \omega)\mathbf{D} \tag{15.7}$$

where ω (in the program called `scf.omega`) is the 'mixing iteration parameter'. For the equations of motion for the induced dipoles, see also Appendix 27.

15.6.1   Polarizability models

Several models of polarizability are available. They are selected in the `simopt.h` file by `#define` POLAR <u>number</u>, where <u>number</u> is a sum of the following numbers (or via `configure.sh`):

  1   Repulsive antipolarizability (shell-core model [31]). All polarizable atoms can interact.

2   Saturated polarizability (see Sect. 25).

4   Makes some optimizations if there are also uncharged atoms. Use this option if there are more than about 1/4 of uncharged atoms in the system.

8   Axial polarizability supported. The polarizability is a tensor. The principal axis of this tensor is defined by a pair of atoms, such pairs are listed in table `axials` in the ble-file [see the blend manual, Section 5: Output format (ble-file)]. Bugs: tensor polarizability cannot be saturated. POLAR 8 has been checked with the shell-core model (POLAR 1) and it is not clear whether it can be used without it.

16   Special (see constrdaa.c and gear2polg.c)

32   Modifier of 1: repulsive antipolarizability for polar-nonpolar pairs only. (E.g., polarizable anions and nonpolarizable cations).

64   The ADIM model [32]: also all intramolecular induced dipole–induced dipole interactions are included

E.g., `#define POLAR 9` selects the shell-core model with a support for axial polarizability tensors; the code will be optimized for no (or a few) uncharged atoms in the system.

If `#define POLAR 0` is specified, only scalar linear polarizability is supported.

If `POLAR` is not #defined the polarizability support is turned off.

In addition, switch GAUSSIANCHARGES (to be used with COULOMB=-3 and POLAR=0) selects all charges to be of Gaussian distribution (of width given as `sigma` in the ble-file as the last parameter).

BUG: Gaussian charges are not supported in blend and via par-files. To use them, you should create the ble-file as well as the respective mol and gol files by a point-charge version, the ble-file must be edited "by hand" to add the sigmas.

15.6.2   Integration methods

There are three integration methods available.

- **Iterations** This method is most accurate but also most time demanding because electrostatic forces are calculated several times in one integration step. The self-consistent field is iterated until sufficient accuracy (given by variable `scf.eps`) of the induced dipoles is reached. The value of `scf.eps` is given in program units, 1 p.u. = 0.0117501 D. Values of `scf.eps` less than 0.01 prog. units are normally recommended for the iteration mode. Since prediction of the induced dipoles from previous integration steps is used at the same time, (cf. option `-p`) usually 2–4 iterations are sufficient.

  This method can be used with both Gear integration (option `-m#`, # > 2) and Verlet/SHAKE (`-m2` = default). For Gear, you may use either the ASPC predictor (default) with the mixing iteration parameter `scf.omega=1` or higher (superrelaxation), or a higher-order partially stable predictor (see option `-p`); here, `scf.omega` less than 1 may be more efficient[2]. By careful testing of available predictors and the values `scf.omega`, often very accurate induced dipole can be obtained for two iterations per step.

---

[2]`scf.omega` can be also set by option `-^`

- **ASPC** Method using a stable second-order predictor see Sect. 27. It should be used with Verlet/SHAKE. This method normally requires one evaluation of electrostatic forces per step, however, the self-consistent field is less accurate (and a little delayed behind the correct SCF); the time-reversibility (and energy conservation) is good, though.

  This method is selected by option `-p`. The default is the $k = 2$ predictor (equivalent to `-p229`). The value of `scf.omega` is set to the theoretical stability limit. However, (much) more precise integration can be obtained by careful optimizing the value of `scf.omega`. To do this, run a series of short simulations (let us say by 100–1000 steps) with increasing values of `scf.omega`, starting from the theoretical stability limit $(k+2)/(2k+3)$ by steps of about 0.01. Set `scf.eps` 10–100× grater than the expected error. Follow quantities called "polar one-step maxerr", "polar one-step stderr", "selffield rate", and "selffield iter"; in the `.cpi` file, use "pstd", "pmax", "Pstd", "Pmax", and "rate" (so they appear in the convergence profile file .cp). Use `scf.omega` by 0.02–0.05 smaller than the value where the iterations become unstable (the error increases and the number of iterations > 1).

- **Combined strategy** In practical simulation, if ASPC is not accurate enough, a much better accuracy (by one order and with a good energy conservation) can be obtained by using exactly two iterations per MD step. This is requested by `scf.eps=-2`. The value of `scf.omega` can be much larger than with one iteration, sometime even biger than 1; careful setup is needed, though, because there is no accuracy limit anymore to prevent the polarization catastrophe.

- **Car–Parrinello**-like approach (also called Lagrangian or extended Lagrangian) is a mechanical model of a polarizable dipole. The Lagrangian is:

$$L = L_0 + \sum_i (M/2)D\dot{\mathbf{r}}_i^2 - \sum_i (K/2)D\mathbf{r}_i^2 \tag{15.8}$$

  where L_0 is the unperturbed Lagrangian and $K = Q^2/\alpha$ (only scalar polarizability is supported). For a fixed configuration at zero temperature, the equilibrium values of the dipoles are the same as the SCF dipoles, during simulation they will oscillate around the SCF solution.

  This method is selected by option `-p`, zero 2nd position.

  The 'mass' of the additional degree of freedom is set indirectly using its typical correlation time `tau.dip`:

$$M = Q^2 \texttt{tau.dip}^2/\alpha \tag{15.9}$$

  (One free 'mass' M in a constant field exhibits small harmonic motion with period $2\pi\texttt{tau.dip}$.)

  BUGS:

  - **The current implementation does not support keeping the SCF subsystem at low temperature. Do not use it.**

  - In FREEBC, the algorithm removing the angular momentum drift is not correct and leads to energy drift (decrease). Use `drift` unless friction thermostat.

  Car-Parrinello-like method is new in version 2.0c.

  NOTE: name "Car–Parrinello" is rather misleading. It is not the quantum Car-Parrinello method.

15.7   Pressure tensor

New in version 2.4a, definition modified in 2.4m.   Pressure tensor (also called stress tensor[3]) is for a system of point interacting particles in free space given by

$$\overset{\leftrightarrow}{P} = \overset{\leftrightarrow}{P}_{\text{kin}} + \overset{\leftrightarrow}{P}_{\text{vir}} = \frac{1}{V} \sum_{i=1}^{N} \left(m_i \dot{\vec{r}}_i \dot{\vec{r}}_i + \vec{f}_i \vec{r}_i \right) \tag{15.10}$$

The scalar pressure is 1/3 of its trace, $P = \text{tr}(\overset{\leftrightarrow}{P})/3$.[4]

For models with rigid bonds (or rigid models described by sites and rigid bonds, see Sect. 11.3) it is given by

$$\overset{\leftrightarrow}{P} = \overset{\leftrightarrow}{P}_{\text{kin}} + \overset{\leftrightarrow}{P}_{\text{vir}} = \overset{\leftrightarrow}{P}_{\text{kin}} + \overset{\leftrightarrow}{P}_{\text{constr}} + \overset{\leftrightarrow}{P}_{\text{config}}, \tag{15.11}$$

where

$$\overset{\leftrightarrow}{P}_{\text{constr}} = \frac{1}{V} \sum_{a,b} \vec{f}_{ab}^{\text{constr}} \vec{r}_{ab}, \tag{15.12}$$

the sum is over all bonds a–b and $\vec{f}_{ab}^{\text{constr}}$ is the constrained force (fictitious force needed to maintain the bond, available readily in the MD code). The configuration al part in periodic b.c. has to be also expressed by pair interactions,

$$\overset{\leftrightarrow}{P}_{\text{config}} = \frac{1}{V} \sum_{i<j}^{N} \vec{f}_{ij} \vec{r}_{ij}. \tag{15.13}$$

For long-range forces the sums must run also over all periodic images. Formulas for the Ewald summation are summarized in excellent paper [33]; for the r-space cutoff less than half the minimum box size the the real-space part of $\overset{\leftrightarrow}{P}_{\text{config}}$ is evaluated in the same way as (15.13).

Notes:

- For atomic systems without rigid bonds (constraints) in the NVT ensemble it holds

$$\overset{\leftrightarrow}{P}_{\text{kin}} = \frac{N k_{\text{B}} T}{V} \overset{\leftrightarrow}{1} \tag{15.14}$$

  The atom-based kinetic part of the pressure tensor for anisotropic systems is no longer isotropic. Full pressure tensor must be therefore used (typical example is surface tension of rigid models of water).

- NEW in V2.6m:
  If some integrals of motion are preserved (e.g., 3 components of the momentum in the periodic b.c. with the Nosé–Hoover thermostat), then the kinetic part of (15.10) is subject to finite-size errors of the order of $\propto 1/N$. More accurate results, especially in dilute fluid (see Fig. 15.1), are obtained if $\overset{\leftrightarrow}{P}_{\text{kin}}$ is multiplied by a factor which restores the equation for the kinetic temperature,

$$\overset{\leftrightarrow}{P}_{\text{kin}} = \frac{N_{f0}}{N_f} \frac{1}{V} \sum_{i} m_i \dot{\vec{r}}_i \dot{\vec{r}}_i \tag{15.15}$$

---

[3]Sometimes the stress tensor is defined as minus the pressure tensor.

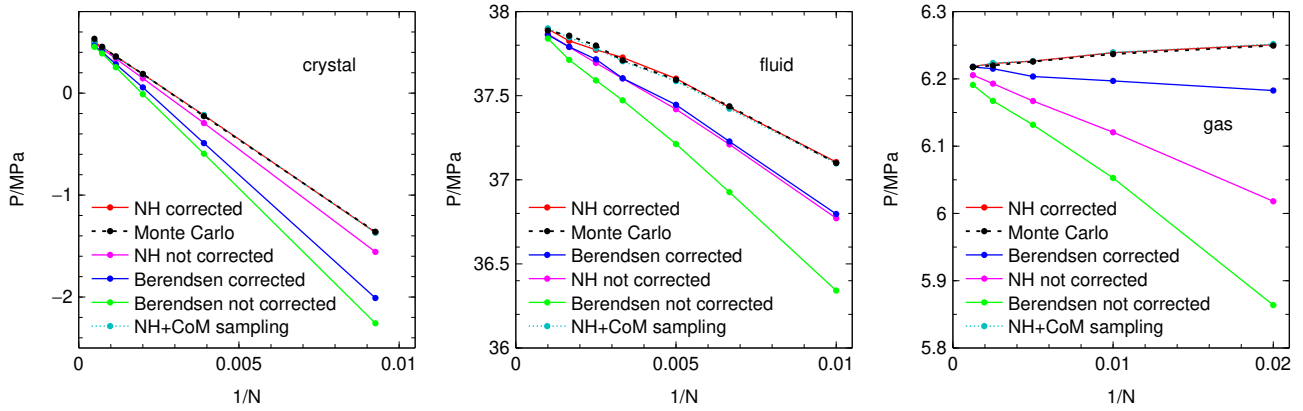[4]In versions 2.4a–2.4l the pressure tensor was defined with factor 1/3 so that $P = \text{tr}(\overset{\leftrightarrow}{P})$.

Figure 15.1: Pressure as a function of the number of atoms for Lennard-Jones argon ($\epsilon/k_{\mathrm{B}} = 119.8\,\mathrm{K}$, $\sigma = 3.405\,\text{Å}$, MACSIMUS style cutoff 7 Å) in the NVT ensemble. From left: fcc crystal ($T = 60\,\mathrm{K}$, $\rho = 1.62\,\mathrm{g/cm^3}$), fluid ($T = 160\,\mathrm{K}$, $\rho = 1\,\mathrm{g/m^3}$), gas ($T = 140\,\mathrm{K}$, $\rho = 0.4\,\mathrm{g/m^3}$). Results with the Nosé–Hoover and Berendsen thermostats are shown both with and without the kinetic pressure correction (for code debugging purposes, these are obtained by independent runs, although the correction is a simple term $k_{\mathrm{B}}T/V$). The NH+CoM sampling item uses the Langevin thermostat of the center of mass, $\tau = 3$ ps, see 9.2.5. The error bars are comparable to the symbol sizes.
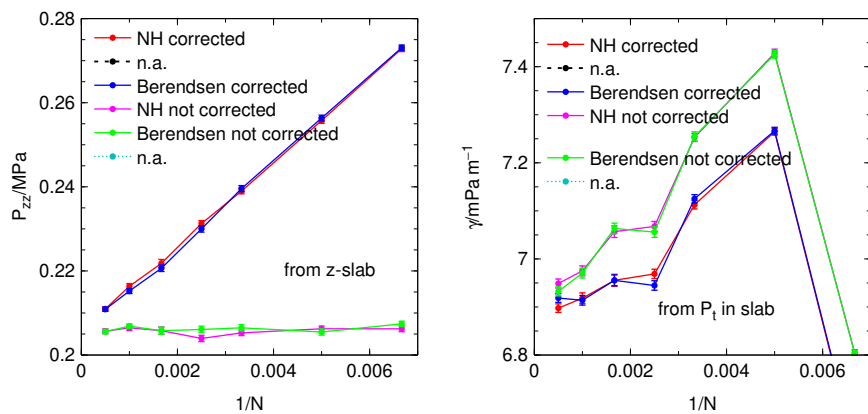

Figure 15.2: Left: The P_{zz} pressure component (vapor pressure) in the slab geometry ($L_z = 2.2L_x = 2.2L_y$). See Fig. 15.1 for the systems. Right: The surface tension; here, the "corrected" value corresponds to *ad hoc* correction $P_t \mathrel{+}= k_{\mathrm{B}}T/V$.

where N_{f0} is the number of degrees of freedom not taking into account conserved quantities but taking into account mechanical constraints (bonds: $N_{f0} = 3N -$ constraints), N_f is the effective number of degrees of freedom (with all conserved quantities subtracted); N_f is used in the formula for the kinetic temperature, T_{kin}. For atomic systems without constraints and with canonical thermostats, (15.14) is exactly reproduced; it is not for the Berendsen thermostat where a (smaller) residual finite-size error $\propto 1/N$ is inevitable. This correction should not be used with the true MTK barostat. In addition, it should not be used in the slab geometry because of thermodynamic inconsistency $\propto 1/N$ present: Test simulations show that the $P_z z$ component of pressure (*i.e.*, the saturated pressure over slab) converges faster with increasing N without this correction. See also variable `corr`.

- The diagonal components can be independently calculated by appropriate scaling of the box, see variable `rescale`; the typical application is the surface tension in the slab geometry, see Sect. 15.8. In the case of systems with constrained bonds, scaling based on center-of-mass has to be used. To obtain the full tensor, the corresponding kinetic part calculated from the centers of mass must be added. The result is in the thermodynamic limit (not numerically for every configuration) the same.

- Direct calculation of the center-of-mass-based configurational part of the pressure tensor is not supported (in V2.4a).

All pressure tensor components are given in Pa. Symbols used in statistic analysis are:

- `Pvir` $= \overset{\leftrightarrow}{P}_{\mathrm{vir}} = \overset{\leftrightarrow}{P}_{\mathrm{constr}} + \overset{\leftrightarrow}{P}_{\mathrm{config}}$

- `Pkin` $= \overset{\leftrightarrow}{P}_{\mathrm{kin}}$ (atom-based)

- `PKin` = molecular (center-of-mass) based kinetic pressure tensor

- `Pvirc` $= \mathrm{tr}(\overset{\leftrightarrow}{P}_{\mathrm{constr}})/3$

- `Pvir pair` $= \mathrm{tr}(\overset{\leftrightarrow}{P}_{\mathrm{vir}})/3$
  (without k-space contribution if Ewald summation is used)

- `Pvir xy-zz` $= (P_{\mathrm{vir},xx} + P_{\mathrm{vir},yy} - 2P_{\mathrm{vir},zz})/3$
  (good for surface tension of models without rigid bonds)

- `Pt` $= (P_{xx} + P_{yy} - 2P_{zz})/3$
  (good for surface tension of all models, needs PRESSURETENSOR=3)

To record components of the pressure tensor in the .cp files, their codes must be given in the .cpi-file (see Sect. 9.2.3. The four-letter codes are `Pvxx`, `Pvxy`, etc., for the site-based virial parts, `Pkxx` etc. for the site-based kinetic part, and `PKxx` for the center-of-mass based kinetic part, and finally `Ptxx`, `Ptyy`, `Ptzz`, `Ptxy`,`Ptyz`, `Ptzx` for the full pressure tensor components. All these values are without cutoff corrections and the units are Pa.

15.7.1 Pressure tensor for Drude oscillators

Let $\vec{r}_i^{\,\prime}$ be positions of the Drude charges and shortly $r = \{r_i\}_{i=1}^N$, $r' = \{r_i'\}_{i=1}^N$. Let us denote $U' = U'(r, r')$ the potential energy as a function of all point charges (normal and Drude). We are interested in the self-consistent potential energy,

$$U(r) = \min_{r'} \left[U'(r, r') + \sum_i \frac{K_i}{2} (\vec{r}_i^{\,\prime} - \vec{r}_i)^2 \right] \tag{15.16}$$

The minimum condition is equivalent to

$$\frac{\partial U'}{\partial r_i'} + K_i(r_i' - r_i) = 0 \tag{15.17}$$

Let us calculate $U(r + \mathrm{d}r)$:

$$U(r + \mathrm{d}r) = \min_{r''} \left[U'(r + \mathrm{d}r, r'') + \sum_i \frac{K_i}{2} (\vec{r}_i^{\,\prime\prime} - \vec{r}_i - \mathrm{d}\vec{r}_i)^2 \right] \tag{15.18}$$

Here $r'' = r' + \mathrm{d}r'$, where $\mathrm{d}r'$ is of the same order as $\mathrm{d}r$. However, the minimum is quadratic in r' and therefore replacing $r'' \approx r'$ in (15.18) will lead to a second order error which can be neglected. From the Taylor expansion of $U(r + \mathrm{d}r)$ (or simply taking $\partial/\partial \vec{r}_i$ of $[\ldots]$ in (15.16)) we get

$$\frac{\partial U}{\partial \vec{r}_i} = \frac{\partial U'}{\partial \vec{r}_i} - K_i(\vec{r}_i^{\,\prime} - \vec{r}_i) = \frac{\partial U'}{\partial \vec{r}_i} + \frac{\partial U'}{\partial \vec{r}_i^{\,\prime}} \tag{15.19}$$

The computer code (r- and k- space Ewald sums for both real and Drude charges) calculates the following components of the pressure tensor

$$\frac{1}{V} \sum_i \left(\vec{r}_i \vec{f}_i + \vec{r}_i^{\,\prime} \vec{f}_i^{\,\prime} \right) = -\frac{1}{V} \sum_i \left(\vec{r}_i \frac{\partial U'}{\partial \vec{r}_i} + \vec{r}_i^{\,\prime} \frac{\partial U'}{\partial \vec{r}_i^{\,\prime}} \right) \tag{15.20}$$

However, from (15.19) it follows that it should be

$$-\frac{1}{V} \sum_i \vec{r}_i \frac{\partial U}{\partial \vec{r}_i} = -\frac{1}{V} \sum_i \left(\vec{r}_i \frac{\partial U'}{\partial \vec{r}_i} + \vec{r}_i \frac{\partial U'}{\partial \vec{r}_i^{\,\prime}} \right) \tag{15.21}$$

Therefore, the following correction should be added

$$\frac{1}{V} \sum_i (\vec{r}_i^{\,\prime} - \vec{r}_i) \frac{\partial U'}{\partial \vec{r}_i} = -\frac{1}{V} \sum_i (\vec{r}_i^{\,\prime} - \vec{r}_i) \vec{f}_i^{\,\prime} \tag{15.22}$$

where $\vec{f}_i^{\,\prime}$ is the force acting on the Drude charge. Since in our model the force on the Drude charge is entirely electrostatic, the trace of the above term (contribution to the virial pressure), the above term is $-(2/V)E_{\text{self}}$.

15.8   Slab geometry and surface tension

To enable the *z*-density profile and surface tension calculations, SLAB must be `#define`d in the respective `simopt.h`, or selected while running `configure.sh`. In addition, `slab.mode=1` and `slab.grid` must be set to see the table of results with and without cutoff corrections. Slab centering (see `slab.sp`) is recommended.

For slab cutoff corrections, see Sect. 31.

To prepare a slab, define first the box; $L_x = L_y$ and $L_z = 2.5L_x$ or $L_z = 3L_x$ is recommended. Example of data in the def-file:

```
x=30
L[0]=x
L[1]=x
L[2]=x*3
```

Note that the efficiency of Ewald summation decreases somehow for a too elongated slab even if the linked-cell list method is used; cutoff electrostatics is not affected in this case.

15.8.1   Create a slab configuration

1. Use `init="slab"` with `tau.T=0.1` or so. Simple, not good for bigger molecules.

2. Prepare a periodic box first, then enlarge `L[2]` and load again with `load.L[2]=1`. You may use variable `shift[2]` to shift the configuration.

3. Use variables `slab.n`, `slab.Kz`, `slab.z0`, etc. These artificial forces should be turned off for measurement. Example:

   ```
   n=300
   N[0]=n
   slab.n[0]=n      ! will affect n molecules
   slab.Kz[0]=10    ! moderate force
   slab.z0[0]=x*0.6 ! slab width=1.2x
   ```

4. Edit a configuration using utilities `plbbox` and/or `plbstack`, then load it using `init="plb"`.

15.8.2   Slab simulation modes

Strictly periodic mode (default) is recommended for a liquid with a very dilute vapor or two layers of liquids. The system is fully periodic even in the the z-direction. If the the vapor pressure is not negligible, molecules moving in the z-direction cross the boundary and cause that the center of the slab moves slightly in the opposite direction. The z-density profile is then "smoothed out" and the cutoff corrections may be imprecise, too. These errors are acceptable if only a few molecules cross the z-boundary in this way. It is not recommended to use `drift=4+8+16+32` (recenter slab and all drifts after every step) and `el.epsinf=-1` (slab-dipole correction) unless *very few* molecules evaporate.

Periodic mode with centered measuring (NEW in 2.6c) The autocenter feature (see `slab.sp`) now shifts the slab automatically before measuring the z-profiles (not physically—the trajectory is not affected). Recommended for high vapor pressures/solubilities etc. May not be good for polymers (the centering algorithm ts molecule-based).

Almost periodic mode is recommended for liquid with moderately dense vapor. In addition, a very small harmonic force keeping the slab in the center is added (see `center.cmn`, `center.cmK[2]`). Particles can still cross the boundary (there is a jump in forces there, but very small) but the slab cannot drift too much. The force constant should be set so that the period of harmonic motion is larger than the evaporation rate; then this method is unsuitable for very low evaporation rates. Incompatible with `drift=4+32`.

Z-aperiodic mode is recommended for a slab with dense vapor. It employs two "caps" preventing particles to move in the z-directions. Example:

```
z=x*3
n=300
N[0]=n
slab.n[0]=n        ! will affect n molecules
slab.Kz[0]=300     ! large force
slab.z0[0]=z/2-3  ! 3 A thick repulsive wall
drift=4+8+16+32     ! recenter slab and all drifts after every step
el.epsinf=-1        ! slab-dipole correction (Ewald only)
```

15.8.3   Surface tension via pressure tensor

The recommended method, `#define PRESSURETENSOR=3`[5] and `#define SLAB` is required. The surface tension is calculated directly from

$$\gamma = -\frac{3}{4}L_z P_{\mathrm{t}}, \quad \text{where } P_{\mathrm{t}} = \frac{P_{xx} + P_{yy} - 2P_{zz}}{3}. \tag{15.23}$$

Variable P_{t} is available under name `Pt` [Pa]; for models with constraints (incl. rigid bodies), do not use variable `Pvir xy-zz` [Pa] because it does not contain the virial of constraint forces. For other info see above and see Sect. 15.7. The saturated pressure is given (without cutoff correction) by component $P_{zz} =$ `Ptzz`. (Note: in versions 2.4a–2.4l by three times `Ptzz`.) Do not forget `slab.mode=1` and `slab.grid`.

15.8.4   Surface tension via virtual area change

This method [34] is based on the formula

$$\gamma = \left\langle \left(\frac{\delta U}{\delta A} \right)_V \right\rangle \tag{15.24}$$

---

[5]For models without constaints (monoatomic, with vibrating bonds), the kinetic part is isotropic, therefore, `PRESSURETENSOR=1` is enough: use "`Pvir xy-zz`" in the output data for the calculations.

where $A = 2L_x L_y$ and the derivative is interpreted as scaling of x and y coordinates and simultaneous contra-scaling of z so that the volume is the same. For technical reasons, this is implemented as:

$$\left(\frac{\delta U}{\delta A}\right)_V = -\frac{3}{4}L_z(p_{\text{vir}} + p_{\text{cutoff correction}})$$

$$p_{\text{vir}} = -\frac{U(+\texttt{dV}) - U(-\texttt{dV})}{2\texttt{dV}\,V}$$

where

$$U(d) = U(qx, qy, z/q^2), \quad q = \exp(d/3)$$

This method is turned on by `slab.mode&2`; `dV` must be also set. Example of surface tension measurements:

```
dV=1e-3          ! step for numerical derivative; for water and 40Ax40A
                 ! box, the systematic error in eta is 1e-5 N/m
slab.mode=3      ! 1=calculate corrections
                 ! 2=use shape scaling V=const
                 ! (default=all coordinates scaled q-times)
```

Notes:

- This method is useful to verify (the accuracy of) the pressure tensor method.

- `rescale="xyzCM"` (molecule-based rescaling) is the default; use `rescale=0` for atom-based rescaling.

- Internally, `slab.mode&2` causes that `rescale&1024` is set; this flag then selects the the virtual area method.

15.8.5   Surface tension via virial pressure

The fastest route to the surface tension of liquids with **negligible vapor pressure** is via the virial of force in the slab geometry. PRESSURETENSOR=3 need not be #defined and the code is a bit faster (e.g., using the virial theorem for Ewald summation). Surface tension is

$$\gamma = -\frac{3}{4}pL_z \tag{15.25}$$

or more precisely with correction to vapor pressure p^{s}:

$$\gamma = -\frac{3}{4}(p + p^{\text{s}})L_z \tag{15.26}$$

where p^{s} can be estimated from the density (use density profile) and the ideal gas EOS. The value of $\gamma = $ `gamma` printed contains cutoff correction of LJ terms calculated by integration from the current density profile.

The following variables (structures) apply for slab calculations, see Sect. 9.2.5: `center`, `densprof`, `el.epsinf`, `drift`. Do not forget to set `slab.mode=1` and `slab.grid` and with Ewald `el.epsinf=-1`, check `drift`. Note that with cutoff electrostatics the virial of electrostatic forces is not equal to the electrostatic energy. Consequently, better values of the pressure and surface tension require the virtual volume change method, see below.

15.8.6 Surface tension via virtual volume change

For completeness, this method is equivalent to the above method, but it uses the virtual volume change to calculate the virial pressure. This makes sense with cutoff electrostatics (`COULOMB>=0`), where the directly calculated virial (replacing the virial of electrostatic force by the electrostatic energy) gives usually worse results. But there is no efficiency gain because scaling is used—the virtual area change is a better choice. To select this method, set `dV` and an isotropic `rescale`.

15.9 Slab geometry and vapor–liquid equilibrium

In the slab geometry, the saturated vapor pressure is given by the zz-component of the pressure tensor, P_{zz}. This quantity is a result of cancellation of large components and determining of low pressures is difficult. The following issues must be considered:

1. Do not use the homogeneous Lennard-Jones correction.

2. Do not use (thermodynamically inconsistent) kinetic correction; both points 1. and 2. are set by `corr=0`.

3. The Fourier transform slab cutoff correction is recommended, see Sect. 31.1.

4. Use short enough timestep, especially for models with constrained bonds (the virial of constrained forces is one of the components). Which VERLET version is the most accurate is not known.

5. For charged systems, use high enough Ewald precision. It can be checked by changing `el.epsk,el.epsr`; e.g., for one selected configuration run `cook` with option `-w0` to prevent writing the configuration, use `noint=1 no=1` in the data, and try different `el.epsk,el.epsr`.

6. P_{zz} includes the virial of attraction of dipoles of both slabs; therefore, the vapor pressure reported is lower (even negative). For the theory, see Sect. **??**.

   (a) For dipolar systems (without free charges), you can use the Berkowitz correction for the energy and forces directly: `el.corr="z"`.

   (b) For systems with free charges (molten salts), the Yeh–Berkowitz correction leads to jumps in forces and cannot be used. The following *a posteriori* correction for the attraction of the z-periodic "sandwich" of slabs is

$$P_{zz}^{\text{corrected}} = P_{zz} + \Delta_{\text{corr}} P_{zz} \tag{15.27}$$

   where

$$
\begin{aligned}
\Delta_{\text{corr}} P_{zz} &= \frac{\text{Var}(M_z)}{2\epsilon_0 V^2} && \text{(SI)} \\
&= 2\pi \frac{\text{Var}(M_z)}{V^2} && \text{(CGS)} \\
&= 2\pi \frac{\text{Var}(M_z/\text{p.u.})}{(V/\text{\AA}^3)^2} \cdot 13806490\,\text{Pa} && \text{(prog.units)}
\end{aligned}
$$

where the last version is directly applicable: use `staprt -nMz SIMNAME.sta` to get the variance (in p.u.) and the volume (in Å^3) in `SIMNAME.prt`.

The values of these corrections are printed to simname.prt, too.

For dipolar systems, both (a) and (b) can be used; it is not clear which approach is better (more accurate).

(c) If `el.corr=12="zCM"` (for "z-Corrected-Measurement") is given in the data, the above *a posteriori* correction is transparently added to P_{zz} (as well as total pressure and potential energy). The trajectory is identical to `el.corr=0` – forces **are not corrected** by Yeh–Berkowitz).

(d) If `el.corr=0`, the above corrections in pressure are printed but not included in the final pressure tensor.

15.10   Interfacial Gibbs energy by the cleaving method

15.10.1   The method

The implemented method is a modification of the method of [35]. It is turned on by specifying `#define SLAB 2` in the `simopt.h` file and recompiling. Also `PRESSURETENSOR=3` is needed (not necessary in NVT, but NVT is not suitable anyway – see below).

The Fourier transform slab cutoff correction are recommended, see Sect. 31.1.

Instead of two walls, we use the cleaving potential (a "knife") in the z-direction; for N interaction sites:

$$\Phi(\sigma) = \sum_i^N K\sigma \, \text{bell}\left(\frac{z_i - z_0}{\sigma}\right) \tag{15.28}$$

where $\text{bell}(x)$ is a bell-like function

$$\text{bell}(x) = \begin{cases} 0, & |x| > 1, \\ (1-|x|)^2/2, & 1/2 < |x| < 1, \\ 1/4 - x^2/2, & |x| < 1/2. \end{cases} \qquad \text{bell}'(x) = \begin{cases} 0, & |x| > 1, \\ x - \text{sign}(x), & 1/2 < |x| < 1, \\ -x, & |x| < 1/2. \end{cases} \tag{15.29}$$

If $\sigma > L_z/2$, the nearest-image conditions apply (there is only one term). This is not recommended because of possible artifacts.

Parameter σ is the half-width of the "knife" while $K/2$ is the maximum steepness. The potential maximum is $K\sigma/4$. The figure shows the function and its derivatives for one particle.

The force on atom i in the z-direction is

$$f_i = -\frac{\partial \Phi}{\partial z_i} = -K \, \text{bell}'\left(\frac{z_i - z_0}{\sigma}\right) \tag{15.30}$$

and hence the contribution to the virial of force (and the P_{zz}-component of the pressure tensor) is

$$W_i = (z_i - z_0) f_i. \tag{15.31}$$

From the semiclassical partition function in the isothermal-isobaric ensemble[6],

$$Z_{NPT} = \frac{\beta P}{N! \Lambda^{3N}} \int dV \int \exp(-\beta U - \beta \Phi - \beta PV) \tag{15.32}$$

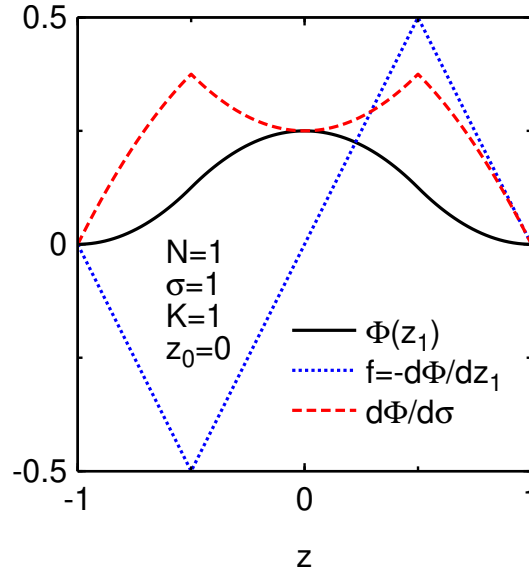[6]for a pure substance – an extension to mixtures is straightforward

Figure 15.3: The cleaving potential ("knife"), force f, and the derivative over σ

where $\beta = 1/k_{\mathrm{B}}T$ and $\Lambda = h/\sqrt{2\pi m k_{\mathrm{B}}T}$ is the de Broglie thermal wavelength, we get the Gibbs energy

$$G(\sigma) = -k_{\mathrm{B}}T \ln Z_{NPT}. \tag{15.33}$$

It is a function of knife width σ, thus by direct differentiation of (15.32) we get

$$\left(\frac{\partial G}{\partial \sigma}\right)_{p,T} = \left\langle \frac{\partial \Phi}{\partial \sigma} \right\rangle_{NPT}, \tag{15.34}$$

where

$$\frac{\partial \Phi}{\partial \sigma} = \sum_i^N K \left[\mathrm{bell}\left(\frac{z_i - z_0}{\sigma}\right) - \frac{z_i - z_0}{\sigma}\, \mathrm{bell}'\left(\frac{z_i - z_0}{\sigma}\right) \right] = \frac{1}{\sigma}\left(\Phi + \sum_i^N W_i\right). \tag{15.35}$$

The interfacial Gibbs energy is obtained by integration over $\mathrm{d}\sigma$.

The program stores (in the convergence profile) and analyses (calculates averages and errors) quantity `Wcleave` normalized by the surface area $L_x L_y$ and converted to Pa[7],

$$W_{\mathrm{cleave}}(\sigma) = \frac{1}{n L_x L_y} \frac{\partial \Phi}{\partial \sigma}. \tag{15.36}$$

where n is the number of cleaving walls ($n = $ `cleave.n`). In other words, $W_{\mathrm{cleave}}(\sigma)\mathrm{d}\sigma$ is the element of reversible work per surface area needed to increase σ by $\mathrm{d}\sigma$, i.e., a pressure; it thus includes the external z-pressure.

The surface cohesion or adhesion energy per surface area is

$$W_{\mathrm{coh,adh}} = \int_0^\infty \left[W_{\mathrm{cleave}}(\sigma) - \langle P_{zz} \rangle \right] \mathrm{d}\sigma, \tag{15.37}$$

where normally a z-direction barostat is used, thus, $\langle P_{zz} \rangle = P$ (simulation parameter, in Pa). Do not forget to convert σ in the integral from Å to m.

---

[7]Internally, the program units are used (K/Å^3, or more precisely kK/Å^3), which the same as the pressure unit, 1 p.u. = 13806485.2 Pa (Codata 2014), see `sim/units.h`

For liquid, the cohesion energy equals twice the surface tension, $W_{\mathrm{coh}} = 2\gamma_{\mathrm{l}}$, where γ_{l} can be obtained easily (see Sect. 15.8); note, however, that the current version of the cleaving method does not include the cutoff corrections.

The solid (crystal) surface energy and interfacial l/s energy are, respectively,

$$\begin{aligned}
\gamma_{\mathrm{l/s}} &= \gamma_{\mathrm{l}} + \gamma_{\mathrm{s}} - W_{\mathrm{adh,l/s}} \\
\gamma_{\mathrm{s}} &= \frac{1}{2} W_{\mathrm{coh,s/s}}
\end{aligned}$$

15.10.2 User interface

The following quantities are available:

cleave.sigma [default=0] Parameter σ, in Å. Typically should run from zero to several Å in steps by 0.01–0.1 Å.

cleave.K [0] Parameter K, in K/Å. Typical values are 10000–100000.

cleave.n [0] Number of cleaving walls, max. 2. Both walls have the same K and σ, but differ in z_0. cleave.n is intended for adhesion energy calculations.

cleave.init [0] If bit 2^0 is set (cleave.init=1) then the cleaving wall positions given below are used, otherwise the previous versions are loaded. To be used for simulation start. This bit is then unset so that next time the wall positions are not changed (similarly to init = continue with simulation).
If bit 2^1 is set (cleave.init=2) then the cleaving walls (for (cleave.n=2) are adjusted along with L_z according to the the z-components of the pressure tensor using the same tau.P and bulkmodulus; use rescale="ZCM" or similar.

cleave.z0 [0] The initial position (z_0) of the first wall, in the units of z-box size L_z.

cleave.z1 [0] The initial position of the second wall (for cleave.n=2).

If two walls are used, cleave.n=2, the Andersen (or center-of-mass Andersen) thermostat should be used to avoid mutual "convection" in both compartments.

A barostat (tested with the Berendsen barostat) should be used in the z-direction (e.g., rescale="ZCM"); thus, PRESSURETENSOR=3 configured is required. For cleave.n=2 also the wall positions are barostatted to maintain the same pressure in both compartments. Note that the wall positions are stored in the cfg file and used unless cleave.init is set.

The slab cutoff correction (see slab.K) is recommended with cleaving, see Sect. 31.1.

15.11   Simulations at walls

Setting `wall.n` in the SLAB version turns on one or two walls or gravity. The atom–wall potential is the Lennard-Jones integrated (9-3) potential).

The simulation can be combined with the SLIT version (system is not periodic in the z-direction). Both FREEBC and cutoff electrostatics (see Sect. 15.3) versions are available, the latter is periodic in x and y directions only and there is no problem with one (bottom) wall and large z. The Ewald k-space is always z-periodic. The SLIT version is compatible with LINKCELL; however, z must not exceed L_z.

Not tested recently: The GOLD version performs simulations at vicinity of metal (ideal conductor) surface.It is (since version 2.0e) treated as sub-version of WALL. The surface is located at $z = 0$, bulk metal at $z < 0$, studied configuration at $z > 0$. If `wall.n=2`, then another Lennard-Jones (non-conducting) wall is added at $z = L$.

Not tested recently: The WIDOM code differs for a slab and provides a z-dependent profile of the chemical potential. WARNING: no cutoff corrections included for WIDOM + SLAB.

15.11.1   Atom–surface force field

The atom–surface interaction is based on averaged (integrated) Lennard-Jones, 11.4, of the interaction between a wall atom and a given atom. Let $\rho = $ `wall.rho` be the mass density of (smoothly distributed) wall atoms. Then the effective wall-atom potential is

$$E = 4\epsilon\rho\sigma^3 \left[\frac{\pi}{45}\left(\frac{\sigma}{z}\right)^9 - \frac{\pi}{6}\left(\frac{\sigma}{z}\right)^3 \right] \tag{15.38}$$

The atom–(wall atom) Lennard-Jones parameters are obtained from the position and depth of the minimum of the respective potential. For Lennard-Jones potentials the original parameters are re-derived, for exp-6-type potentials (version "buck" and "busing"), the equivalent LJ parameters are thus obtained.
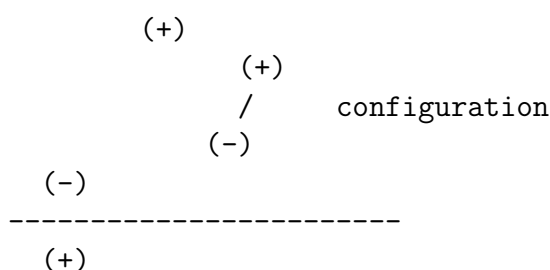
The wall atom is the the **last** atom in the atom types. This can be changed by parameters `wall.LJ[i].sig` and `wall.LJ[i].neps`.

Repulsive walls (for `wall.n>0`) have the $1/r^3$ term omitted.

15.11.2   Atom-metal force field

WARNING: Not tested recently!

The electrostatic forces of a charge with a conducting wall (gold) are obtained by the method of "inversion":

```
        (+)
             (+)
             /        configuration
            (-)
      (-)
   ------------------------
      (+)
```

```
        (+)
          \      mirror image with opposite charges
        (-)
     (-)
```

The potential energy:

$$U = \sum_{i<j} q_i q_j \left(\frac{1}{|r_i - r_j|} - \frac{1}{|r_i^* - r_j|} \right) - \frac{1}{2} \sum_i \frac{q_i^2}{|r_i^* - r_i|}, \qquad (15.39)$$

where $r_i = (x_i, y_i, z_i)$ and the mirror image is $r_i^* = (x_i, y_i, -z_i)$. In the cutoff electrostatics version all the $1/r$ terms are modified (see Sect. 15.3).

(To derive this formula, calculate the electrostatic energy of the 'double' system with reflected charged. You will get 2U. Use $|r_i^* - r_j| = |r_i - r_j^*|, |r_i - r_j| = |r_i^* - r_j^*|$ to simplify.) Note also that $|r_i^* - r_i| = 2z$.

Implementation:

Inside all site–site functions, the Coulomb term is replaced by Coulomb+inverted Coulomb. Module `interpot.c` is replaced by `gold.c`, with the same header file `interpot.h`. Boundary conditions macros in `intermac.h` are updated to treat periodic b.c. in x, y only and extra support for inversion is added. The SUM_i part (along with LJ, see below) is in `force.c`, function goldforces().

CAVEAT: optimized water potentials not (yet) implemented, `gold` must be run with option -x.

15.11.3   Using WALL and GOLD versions

Programs `wall/gold` derive `rho` from real mass density `wall.rho` in kg/m³ which is available in the data (the default `wall.rho=19320` corresponds to gold). The Lennard-Jones parameters of the wall material must be listed as the LAST atom in table Lennard-Jones (or tbale of given force field) in the system ble-file and at the same time it MUST NOT appear as normal atom in molecules. This can be accomplished by putting it the last position when using `blend` and specifying `N[LASTINDEX]=0` or blend option `-n0`, e.g.

```
blend MOLECULE(s) -n0 au
```

where `au.che` is just

```
gold atom
MAU ! for charmm21.par
AU  ! for charmm22.par or GROMOS (to be implemented)
```

WARNING: this requires that the correct data for gold are present in the parameter set (`charmm21.par`)! The current value is

```
!        alpha     EMIN         Rmin
!        [A^3]   (kcal/mol)     (A)
MAU       1       -0.039        3.293     ! UFF -- JK
```

based on UFF [J. Am. Chem. Soc. 114,10024 (1992)]. The value for alpha (used in the combining rule) is just guess. We need better data from literature, and/or adjust this for given task. This value affects the energy of adsorption.

NOT TESTED: interaction with `blend -n-1`

15.11.4   Wall versions and integrals of motion

Minor versions:

`FREEBC` free b.c. in x, y, and $z > 0$

`NIBC` nearest image periodic b.c. in x, y, free in $z > 0$ (NOT TESTED)

`Cutoff electrostatics` periodic b.c. in x, y, free in $z > 0$

`Ewald` Ewald summation not supported

Integrals of motion:

`x,y periodic and FREEBC:` momentum in x and y is conserved and is periodically reset to zero, z-momentum is unchanged; center-of-mass is adjusted to $(x, y) = (0, 0)$

`FREEBC:` angular momentum in z-direction is conserved and is periodically reset to zero, x, y are unchanged

Degrees of freedom to subtract for conserving quantities (see `siminit.c`):

- GOLD + FREEBC, no central force (option `-o0`): 3

- GOLD + FREEBC, central force (option `-o#`): 1

- GOLD + periodic: 2

- Fixed atoms (option `-k`) shrinks this to 0 (NOT GENERALLY CORRECT).

15.11.5   Initial configuration

Initializer:

- only `init=3` available, both with and without `config` option

- config (= option `-n-1` in `blend`): the configuration is shifted in the z-direction so that the atom with minimum z coordinate is `wall.minz` from the surface (`wall.minz` is available in the data, default=3). `wall.minz<=0` turns off this function. Suitable for a 'configuration' prepared by blend.

- standard (`-n-1` not used): when initializing by 'random shooting' condition, `min z > wall.minz` is satisfied

- CAVEAT: MC not implemented

15.11.6   Wall visualization and more

Option `-\`number was added to `show` to show the surface as number*number mesh. Version `-\`number`\` shows a slit pore, negative argument uses wall made of atoms. Note that `\` should be protected from the shell by doubling. The size is derived from the size of the system (if no box size is specified either in the plb-file or by option -l). Example:

```
molcfg MOLNAME1 [MOLNAME2 ...] SIMNAME
show -\\5 -I% SIMNAME
```

Full example (OPTIONS ARE OBSOLETE):

```
blend -o test -p ala -h hoh au
molcfg ala hoh test
goldcut test -y-1 -q-100 -x
```

Bugs/Plans:

- Unify interpot.c and gold.c into one module???

- POLAR version in some cases unnecessarily repeats calculation of sqrt()

WARNING: `LJcutoff<0` means that the effective LJ cutoff is —LJcutoff—*van der Waals diameter (not LJ sigma)

BUG to fix: some extra terms proportional to the shift of the cut-smoothed-and-shifted electrostatic potential are present (in functions XQQ, XQQM, and *QQ14). This affects internal energy and pressure, but not trajectory. Not fixed (yet) for gold!!!

15.12   Anchor sites and axes and measure forces

The ANCHOR version allows selected atoms or the center-of-mass to be fixed and at the same time to measure forces and torques acting on them. Run time option `-k-#` defining the output (bits of #: 1=statistics, 2=convergence profile, 4=ASCII file simname.anc), must be used. Cf. version with harmonic springs, see Sect. 15.1, using positive option `-k#`. The input data are in file simname.fix:

```
site  MOLECULE SITE X Y Z # keep site SITE of MOLECULE at (X,Y,Z)
                          # max 1 site per molecule allowed
                          # SITE must not be a dependant
cm    MOLECULE X Y Z    * # keep center-of-mass of MOLECULE at (X,Y,Z)
CM    X Y Z             * # keep center-of-mass of all not-anchored molecules
inert MOLECULE X Y Z      # keep principal axis parallel to direction (X,Y,Z)
axis  MOLECULE SITE X Y Z # keep vector (center-of-mass,SITE)
                          # parallel to (X,Y,Z)
pair  MOLECULE SITE1 SITE2 X Y Z
                          # keep vector SITE1-->SITE2 parallel to (X,Y,Z)
triple MOLECULE SITE1 SITE2 SITE3 X Y Z
                          # keep plane given by (SITE1,SITE2,SITE3)
```

```
                              # perpendicular to (X,Y,Z)
  group MOLECULE NS SITE_1 ... SITE_NS X Y Z
                              # keep center-of-mass of given group
                              # groups may repeat and even overlap, but no
                              # check of mechanical inconsistency is made
  # measurements only (no constraining), of center-of-mass:
  r MOLECULE                  # print position of MOLECULE
  v MOLECULE                  # print velocity of MOLECULE
  x MOLECULE                  # print acceleration of MOLECULE
  force MOLECULE              # measure force to (center-of-mass of) MOLECULE
  moment MOLECULE             # measure torque to MOLECULE (center-of-mass based)
```

where `MOLECULE` is the molecule number (1st molecule has n=0), `SITE` is the site number. `X Y Z` is the position or unnormalized direction to be fixed; if `*` is given, negative positions are not anchored. The keywords can be abbreviated to 1 letter. Line beginning with non-letter (e.g., space) is treated as comment and ignored, i.e., the keywords must start from column 0.

The last two commands, `force` and `moment`, measure the force and the moment of force (torque), respectively, acting to the given molecule. Nothing is constrained. Both quantities are related to the center-of-mass. These measurements are based on atom forces. All other commands constrain some quantity. At the some time, the constraint force is measured. All these forces are (in SI units) printed in file simname.anc with a self-explaining header.

Commands `site` and `cm` cause the constraint for position to be satisfied. `cm` is implemented after the Shake iteration because the whole molecule is is shifted to satisfy the constraint; at the same time, the constraint force is measured. `site` is implemented within the Shake iteration.

Commands `inert`, `axis`, `pair`, `triple`, cause the molecule to rotated around the center-of-mass to satisfy the constraint; this is exclusive with `site` option, but can be combined with `cm`. It is implemented after Shake because the center-of-mass is preserved.

If both `force` and `cm` are given, both the force on the center-of-mass and the constraint force (on the center-of-mass) are printed. The constraint equals minus the force.

If both `moment` and either of `inert`, `axis`, `pair`, `triple` are specified for a rigid molecule, the moment with subtracted projection to the given axis equals minus the constraint force with precision proportional to h^2. If the molecule is not rigid, both quantities differ, but in mechanically well-defined systems (plane defined by atoms far from each other, the principle moments of inertia differing sufficiently) the difference consists of small and fast oscillations.

Example:

```
  s 0 1  1 1 1
  c 1  5 5 5
  t 1  0 1 2  1 1 0
```

Site 1 of molecule 0 is (1,1,1). Center of mass of molecule 1 is (5,5,5). Plane given by sites 0,1,2 of molecule 1 is perpendicular to (1,1,0).

In the input data, `drift=0` must be specified and `omegac<1` is recommended. After these coordinates have changed, `drmax` should be set to a small negative value. Suggested value is `drmax=-0.03`, but the final temperature will be too low and equilibration with larger `|drmax|` or none must follow. If `init=3` and option `-k-KEY` are combined, the initial configuration satisfies these constraints and setting `drmax` is not needed (does not apply for group constraints).

Caveats:

- Dependants are not allowed as SITEs.

- Variable `conserved` and `drift` must be specified by user – the automatic setup is not implemented.

- Some combinations of constraints are illegal even if there is no apparent reason for it.

- Implemented for one version of Shake (SHAKE=1) only – check simopt.h

- Constraining axes for flexible molecules is not a well-defined task. Use with caution.

- Note also that rigid molecules generated by `blend` with the dep-files `species.dep` do not preserve the inertia tensor.

- If `init=3` is used, the generated configuration satisfies the positional constraints, but not the axes. Equilibrating with `drmax` must follow.

- The above point, or if the constraints are changed (with `drmax`), flexible molecules may adopt fast internal vibrations which (see see Sect. 12.2) are only weakly coupled to the rest. Using the Andersen thermostat may help; in case of molecules with rigid bonds, this is incorrect and can only be used temporarily.

- Inconsistent group constraints may lead to integrator crash or infinite SHAKE loop.

15.13   Analyze pair energies

Special `cook` version MARK:

- calculates pair energies between groups of atoms

- one group is the zeroth-group, there are `NG` (`NG<254`) other groups

- energy (in vacuum) of n-th group vs. zeroth-group is calculated and recorded in CP-file (note:1st 2 columns have bad heading)

Source of data:

- if option `-r`: files simname.1, simname.2, ..., simname.no

- if not option `-r`: playback file simname.plb, frames 1..no

Compiling:

- `#define MARK` needed in `simopt.h`

- link with module `mark` (`mark.h, mark.c`)

Bug: inefficient, calculates also pairs that are not needed

Usage:

- prepare file <u>simname.mkr</u> (option `-r` > 0) or <u>simname.mkp</u> (option `-r` ≤ 0) in the following format (ID is group ID derived from 1st atom ID):

```
NG
0 ID GROUP
1 ID GROUP
2 ID GROUP
...
```

  For example (abridged):

```
10
   0 ASN64aC      0
   1 ASN64aCA     0
   2 ASN64aCB     0
...
 633 PRO10cCG    10
 634 PRO10cN     10
 635 PRO10cOC1   10
 636 PRO10cOC2   10
```

  NOTE: in the playback version (no option -r), only those atoms actually stored in the playback may be listed.

Option -r summary:

 `-r0` reads playback <u>simname.plb</u>, uses <u>simname.mkp</u>

 `-r1` reads configs <u>simname.1</u> ..., uses <u>simname.mkr</u>

 `-r-1` reads configs <u>simname.1</u> ..., uses <u>simname.mkp</u>

NOTES (added 3/2000):

- IDs in the mkp,mkr files are used as info only, they do not select anything nor are checked

- The N[] in the def-file used must conform the original def-file.

- Atom not to be included in any group simply have the group field blank (e.g., "636 PRO10cOC2 ")

- <u>simname.mkp</u> and <u>simname.mkr</u> files may be shorter (i.e., not listing all atoms). Not listed atoms are simply ignored. Only if the actual plb file is shorter, a warning is printed.

- Typical usage is for protein in water. For energies without water (not solvated), mkr and mkp files can be created from the protein mol-files (water is ignored). To include water, use a mol-file generated by `molcfg`.

15.14 Viscosity by shear stress

Two methods for viscosity determination are supported. The equilibrium method is based on the Green–Kubo, See Sect. 14.5. It is generally more accurate.

The NEMD simulation requires compile-time switch #define SHEAR (or selecting it in `configure.sh`). It is performed in a rectangular box of sizes $L_x \times L_y \times L_z$, where $L_x = L_y$. The recommended $L_z = 3L_x$. The external acceleration acting on particles in the direction perpendicular to the z-axis is cos-modulated

$$\ddot{\vec{r}}_i = C_f \vec{n} \cos \frac{2\pi z_i}{L_z}. \tag{15.40}$$

Where unit vector $\vec{n}$ is perpendicular to $\hat{z}$; we use $\vec{n} = (1,1,0)/\sqrt{2}$. Then

$$f_{a,i} = n_a m_i C_f \cos \frac{2\pi z_i}{L_z}, \ \ a \in \{x, y\}. \tag{15.41}$$

We also add a small z-force to all atoms so that the sum of forces over all atoms is exactly zero—normally it slightly deviates because particles are not distributed ideally uniformly.

The Navier-Stokes equation for a steady Newtonian flow of an incompressible fluid reads as

$$\eta \nabla^2 \vec{v} + \vec{f} = 0, \tag{15.42}$$

where $\vec{f}$ is the (smoothed) external force per unit volume,

$$\vec{f} = \vec{n} \rho C_f \cos \frac{2\pi z}{L_z} \tag{15.43}$$

and $\rho = \sum_i m_i / V$ is the mass density. The solution is a steady flow proportional to the applied force,

$$\vec{v} = -\vec{n} \frac{C_f \rho L_z^2}{4\pi^2 \eta} \cos \frac{2\pi z}{L_z}. \tag{15.44}$$

To calculate $\vec{v}$ and its amplitude, we have to go from the continuum description back to the atoms. Let us define

$$S_f = \sum_i \vec{f}_i \vec{n} \cos \frac{2\pi z_i}{L_z}, \ \ S_p = \sum_i \vec{v}_i \vec{n} m_i \cos \frac{2\pi z_i}{L_z}, \tag{15.45}$$

The final formula for viscosity then adopts the form

$$\eta = \rho \frac{L_z^2}{4\pi^2} \frac{\langle S_f \rangle}{\langle S_p \rangle}. \tag{15.46}$$

During simulation, also the velocity amplitude

$$C_v = \frac{S_f}{S_p} C_f \tag{15.47}$$

is monitored (name `Cvel`) in the convergence profile to check for the steady state.

NOTE: The code (Shear() in simmeas.c) uses $\vec{n} = (1,1,0)$ and the following equivalent particular formulas: In order to calculate C_v and consequently η, we define sums

$$S_f = \sum_i f_{x,i} \cos\frac{2\pi z_i}{L_z}, \quad S_p = \sum_i (v_{x,i} + v_{y,i})m_i \cos\frac{2\pi z_i}{L_z}, \tag{15.48}$$

The final formula for viscosity then adopts the form

$$\eta = \rho\frac{L_z^2}{2\pi^2}\frac{\langle S_f\rangle}{\langle S_p\rangle}. \tag{15.49}$$

During simulation, also the velocity amplitude

$$C_v = \frac{S_f}{S_p}\frac{C_f}{2} \tag{15.50}$$

is monitored (name `Cvel`) in the convergence profile to check for the steady state.

Heating rate is an important control quantity in the simulation. In the steady state flow it is

$$\dot{E} = \frac{\mathrm{d}E}{\mathrm{d}t} = \frac{1}{2}\int \eta(\nabla v)^2 \mathrm{d}V = \frac{V}{\eta}\left(\frac{C_f\rho L_z}{4\pi}\right)^2. \tag{15.51}$$

It is useful to express it via expected increase in temperature because it will help us to set up the thermostat used to remove the heat. The crudest estimate of the isochoric heat capacity is that of the ideal gas, $C_V = N_f k/2$, where N_f is the number of degrees of freedom of the simulation box. Then

$$C_f = \sqrt{\frac{8\pi^2\dot{T}\eta N_f k}{V L_z^2\rho^2}} \tag{15.52}$$

and we use product `shear` $= \dot{T}\eta$ as the parameter of the simulation. If the Berendsen (friction) thermostat is used to remove the friction heat, the following quantity

$$\Delta T = T_{\mathrm{kin}} - T = \dot{T}\tau_T \tag{15.53}$$

a rough but still useful measure of the deviation of the system from equilibrium. Namely, the Berendsen thermostat also uses the ideal gas approximation to relate the correlation time `tau.T` to the friction term.

For larger systems it takes some time until a steady velocity profile develops. Relaxation to this steady state is exponential with the correlation time given by

$$\tau_v = \frac{\rho}{\eta}\left(\frac{L_z}{2\pi}\right)^2. \tag{15.54}$$

As soon as an estimate of viscosity is know, the recommended strategy to fast finding the stationary state is to simulate time $t = \tau_v \ln 2$ with doubled parameter χ and half τ_T.

The shear stress method is activated by the compile-time option `#define SHEAR` and by data `shear`. The value of `shear` $= \dot{T}\eta$ is given in SI units $\mathrm{kg\,m\,s^{-2}\,K} = \mathrm{PaK}$. Typical values are on the order of 1e9–1e10.

The viscosity value is a ratio of two variables denoted as `shear:num` and `shear:den`, see (15.46) and (15.49) (the factor are already embedded in these variables).

To obtain reliable viscosity estimates, a series of measurements with different values of `shear` should be conducted and extrapolated to `shear=0`. The theoretical dependence at the limit of `shear=0` is a linear dependence of η to `shear`.

15.15   Widom and scaled insertion particle method

Required: `#define WIDOM` in `simopt.h` or Widom requested in `cook/configure.sh`.

BUG: The Widom method cannot be used with Ewald (k-sum is ignored).

15.15.1   Pure liquid

The Widom method gives the residual chemical potential, i.e., the absolute chemical potential per a molecule is

$$\mu_{\mathrm{res}} = \mu - \mu_{\mathrm{id}}, \quad \mu_{\mathrm{id}} = kT \ln\left(\frac{\Lambda^3 N}{q_{\mathrm{in}} V}\right),$$

where Λ is the de Broglie wavelength, N the number of molecules, q_{in} the internal partition function (incl. rotations) at temperature T, and V the box volume. In the Widom method, one calculates

$$w = \exp(-\mu_{\mathrm{res}}/k_{\mathrm{B}}T) = \frac{1}{V}\frac{Q_{N+1}}{Q_N} \tag{15.55}$$

either in one step, or as product $w = w_0 w_1 \ldots w_k$ of the Boltzmann factors of inserting a small particle (w_0) and subsequent stepwise scalings of this particle. The corresponding variable names in sta- and prt-files are `Widom` for insertion and `XWidom` for the particle growth step.

In (15.55), Q_N is the configurational integral of N indentical particles. Therefore, quantities w and μ_{res} correspond best to number density $\mathcal{N} = (N + 1/2)/V$.

The chemical potential μ^* of the simulated fluid with repect to ideal gas at standard pressure p° is then

$$\mu^* = \mu - \mu^\circ = \mu_{\mathrm{res}} + k_{\mathrm{B}}T \ln\left(\frac{N k_{\mathrm{B}} T}{p^\circ V}\right), \tag{15.56}$$

where μ° is the chemical potential of ideal gas at T, p° (common standard state for gases).

The Widom method (with gradual insertion, see below) can be used to calculate the saturated vapor pressure, p^{s}, at low temperatures for which the direct vapor/liquid slab method is inaccurate. To do this, the density ρ_0 at an unknown (but low, in practice 0) pressure must be determined first; e.g., in an NPT simulation or in a series of NVT simulations. Remember that the the number density for the Widom insertion step (to determine w in one step or w_0) is then $\mathtt{rho} = \rho_0 N/(N + 1/2)$. The number of particles in the grow-steps to determine $w_1, w_2 \ldots$ is now $N_1 = N + 1$ and then $\mathtt{rho} = \rho_0 N/(N + 1/2) = \rho_0 N_1/(N_1 - 1/2)$.

The fugacity in the gas phase is

$$\exp(\mu^*/k_{\mathrm{B}}T)p^\circ = \varphi^{\mathrm{s}} p^{\mathrm{s}}$$

where μ is the chemical potential of both phases (the same because of the vapor/liquid equilibrium) and φ^{s} is the fugacity coefficient of the vapor. On comparing with (15.56)

$$p^{\mathrm{s}} = \frac{N k_{\mathrm{B}} T}{V w \varphi^{\mathrm{s}}},$$

where for low enough pressures it holds $\varphi^{\mathrm{s}} = 1$.

The methods for truncated argon are compared in Table 15.2. It is seen that small systems (box 300 particles, slab 1000 particles) are not sufficient because the difference between P_{zz} is about 5σ from the Widom value. The results using large N are better 1.5σ (and the difference is of opposite sign).

Table 15.2: Test simulations of vapor pressure for truncated argon ($\epsilon/k_{\mathrm{B}} = 119.8\,\mathrm{K}$, $\sigma = 3.405\,\text{Å}$, `LJcutoff=12`) at $75\,\mathrm{K}$. P_{zz} is the directly measured zz-component of the pressure tensor in the slab geometry, "from $\mathcal{N}(z)$" is calculated from the gas density in the slab geometry using the virial EOS (with B_2), The Widom method (6 steps) is based on the density calculated in a series of NVT simulations at the approximated target pressure, and Δp is the estimated pressure difference due to the difference between the above density and the liquid density at the slab center. Both the gas and liquid densities were obtained by fitting, see script `splitfitz.sh`. The small system results are averaged from $h = 2\,\mathrm{fs}$ and $4\,\mathrm{fs}$, the large system results are for $h = 4\,\mathrm{fs}$.

system $\rightarrow$	small		large	
method $\downarrow$	N	p^{sat} [Pa]	N	p^{sat} [Pa]
slab P_{zz}	1000	25299(250)	8000	25402(139)
from $\mathcal{N}(z)$	1000	25742($>$100)	8000	25600($>$33)
Widom	300	26558(80)	2500	25112(142)
Δp		250		20

15.15.2   Henry constant

The Henry constant is defined by

$$K_{\mathrm{H}} = \lim_{x_2 \to 0}(p_2/x_2),$$

where p_2 is the partial pressure of solute (2) above the solution and x_2 its molar fraction in the solution. In simulation, we virtually insert one particle ($N_2 = 1$) into N_1 molecules of solvent; thus, $x_2 = 1/(N_1 + 1) \approx 1/N_1$. We will assume that the solvent has a low vapor pressure so that it does not change the fugacity of solute and we may use the ideal gas approximation. The residual chemical potential of solute is

$$\mu_{\mathrm{res},2} = \mu_2(x_2) - \mu_2^{\mathrm{id}}(V)$$

In equilibrium:

$$\mu_2(x_2) \overset{\mathrm{eq.}}{=} \mu^{\mathrm{id}}(V_2) = k_{\mathrm{B}}T\ln\frac{\Lambda_2^3}{q_2 V_2}$$

Hence

$$\mu_{\mathrm{res},2} = k_{\mathrm{B}}T\ln\frac{N_2\Lambda_2^3}{q_2 V_2} - k_{\mathrm{B}}T\ln\frac{N_2\Lambda_2^3}{q_2 V_2} = k_{\mathrm{B}}T\ln\frac{V}{V_2}$$

where V_2 is the volume corresponding to $N_2 = 1$ particle of ideal gas at pressure p_2. Further

$$w = \mathrm{e}^{-\beta\mu_{\mathrm{res},2}} = \frac{V_2}{V} = \frac{N_2 k_{\mathrm{B}}T}{p_2 V} = \frac{k_{\mathrm{B}}T}{K_{\mathrm{H}} x_2 V} = \frac{k_{\mathrm{B}}T N_1}{K_{\mathrm{H}} V} = \frac{k_{\mathrm{B}}T \mathcal{N}_1}{K_{\mathrm{H}}}$$

where $\mathcal{N}_1 = N_1/V$ is liquid number density. Finally

$$K_{\mathrm{H}} = \frac{\mathcal{N}_1 k_{\mathrm{B}}T}{w}.$$

Once again, w may be a product of the initial insertion and Boltzmann factor of particle growth.

This should be equal the gas abs. chem. pot. of gas at pressure p (assumed to be ideal gas, Ng and Vg refer to the gas phase)

$$\mu = k_{\mathrm{B}}T\ln(\Lambda^3 N_g/V_g) = k_{\mathrm{B}}T\ln(\Lambda^3 p/k_{\mathrm{B}}T)$$

15.15.3 WALL/SLAB version

WARNING: this version has not been tested recently and may be out of order.

`#define WALL` or (?) `#define SLAB` required. In case of gradual insertion. also `#define ANCHOR`.

The insertions are performed for a series of z from `widom.z0` to `widom.z1` by `widom.dz` and a histogram of the Boltzmann factors is prinded. See more in the data below.

- The Widom method inserts a molecule to a place with defined z-coordinate with respect to the reference point as defined in the ble-file (with random rotation added). It is the CM (center-of-mass) by default if blend has been used (with the default option `-y`).

- Consequently, in the grow-molecule stages, the partial (scaled) molecule(s) should be placed at CM, too. To do this, cook* must be compiled with "anchoring" and run with option `-k0`. File SIMNAME.fix must contain the molecule(s) being fixed with keyword `cm`, like

  ```
  cm NUMBER X Y Z
  ```

- Equivalently, if the molecule in the blend stage has a selected site as the reference point (e.g., O for water), use either cook -k0 with SIMNAME.fix containing:

  ```
  site MOL_NUMBER SITE_NUMBER X Y X
  ```

  This fixes the site exactly (as a constraint). For fixing the site by a harmonic spring, use cook `-kK` (K=force constant) with SIMNAME.fix containing:

  ```
  SITE X Y Z
  ```

  CAVEATs: SIMNAME.fix has a different format with `-k-KEY` and `-kK`, $K > 0$. The former requires -DANCHOR, the latter does not.

- The slab should be centered. For `-kK`, $K > 0$, flag `drift&4` is OK. For `-k-#`, either use whole-slab harmonic force defined by (in input data)

  ```
  center.cmK[2]=100 ! tau=0.7755*\(M/cmK) [M=tot.mass in g/mol]
  drift=0
  ```

  or use centering of all not anchored molecules (keyword CM).

  (If appropriate, use `center.cmn` for # of molecules; default=all)

- For scaled water, -x must be used to suppress true water detection

- If the anchored (fixed) atoms are not in place (coordinates in SIMNAME.fix have changed or `-k-#` is turned on suddenly), `drmax=<negative value>` should be used in input data and then released. The recommended value is `drmax=-0.1` In case of problems, use lower (in abs. value) value, e.g., `drmax=-0.03`. However, too low `drmax` may cause too low a temperature and after `drmax=0` is reset, the system must be equilibrated. **Never use `drmax` in productive runs!**

The data are internally recorded by the statistical (statics.c) module (extension `.sta`) in variables of names "Widom<u>mode</u> z=<u>z</u>" The output is printed to file SIMNAME.wid.

NOTE: The Widom method inserts a virtual molecule into the configuration AFTER the last step in standard versions. However, in the linked-cell list version BEFORE the last step (previous linked-list is re-used). Therefore the results of non-linked-cell and linked-cell are not exactly the same even with the same random seed (option `-z`) specified. But if you advance the linked-cell output by 1 timestep, then the Widom results with `noint=1` are exactly synchronized with the non-linked-cell ones.)

BUG: Cutoff corrections are not calculated if SLAB (slab, wall, ..) is active. Final correction can be added.

WARNING: with slab-centering (`drift&4` set), it is generally illegal to have a molecule close to z=L[2]. The LINKCELL method may then detect coordinates out of range [0,L[2]]. Although (in the interactive run) it may be sometimes possible to continue, this is not recommended. A protective potential (see slab.Kz etc.) should be used.

15.15.4   Input data

`widom.sp (default=0)` Species to be (virtually) inserted (for the number, see the ble file, keyword species).

`widom.n (0)` Number of insertions per cycle (WALL version: and per slab)

`widom.spreal` Species to change in the gradual (scaled-particle-like) insertion. Instead of virtual insertion of species `widom.sp`, all molecules of species `widom.spreal` are changed into `widom.sp`. Since all the coordinates are unchanged, the molecules should differ in partial charges and Lennard-Jones terms only. Note that `widom.n` does not apply. The recommended strategy to calculate the chemical potential is to place the species into a ble-file several times with the extra instances edited by hand to describe the partial molecules during gradual insertions. If LJ terms are changed, it requires to create copies of atom types.
See also `TEMPLATE.sh` and `widomres.sh`.

`widom.mode` WALL version switch, use a sum of:

   1   also record symmetrized functions (makes sense for wall.rep=0 or wall.rep=3)

   2   record $\langle\exp(-[U_{\mathrm{pair}} + U_{\mathrm{wall}}]/k_{\mathrm{B}}T)\rangle$, leading to real mu

   4   record $\langle\exp(-U_{\mathrm{pair}}/k_{\mathrm{B}}T)\rangle$; useful for monoatomic molecules where the wall potential is simply added

   8   record $\langle\exp(-U_{\mathrm{wall}})/k_{\mathrm{B}}T)\rangle$ – the angle averaged Boltzmann factor of the molecule-wall potential

`widom.z0`

`widom.z1` Minimum and maximum z for which (in the wall/slab geometry) the insertion will be performed.

`widom.dz` The step.

15.16   Metals

15.16.1   Tight-binding potential

The interatomic potential [36] is decomposed to a sum over all pairs, $\sum_{i<j}$, and a nonadditive term, $\sum_i$:

$$U = \sum_{i<j} 2A\mathrm{e}^{p(r_{ij}/r_0-1)} - \sum_i \rho_i^{1/2}, \tag{15.57}$$

where

$$\rho_i = \sum_{j\neq i} \rho_{ij} = \sum_{j\neq i} \xi^2 \mathrm{e}^{-2q(r_{ij}/r_0-1)} \tag{15.58}$$

mimics the electron density. The algorithm to calculate the forces and energy proceeds in two passes. In the first one the density components ρ_{ij} are calculated and in the second one, which is again a loop over all pairs, the calculations of the additive and nonadditive forces are finished. From these pair and pair-like forces the pressure tensor is calculated. (Another way to calculate the pressure tensor is via the virtual volume and shape change method [37]. We used this method to test the algorithm.)

Forces

Let is write (15.57) as

$$U = U^{\mathrm{pair}} + U^{\mathrm{multi}} \equiv \sum_{i<j} u_{ij}^{\mathrm{pair}} - \sum_i \rho_i^{1/2}, \quad u_{ij}^{\mathrm{pair}} = 2A\mathrm{e}^{p(r_{ij}/r_0-1)} \tag{15.59}$$

The k-gradient of ρ_{ij} is

$$\vec{\partial}_k \rho_{ij} = \frac{\vec{r}_{ij}}{r_{ij}}(\delta_{kj} - \delta_{ki}) \tag{15.60}$$

where $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$. The pair part is standard,

$$f_{ij}^{\mathrm{pair}} = \frac{p}{r_0} 2A\mathrm{e}^{p(r_{ij}/r_0-1)} \frac{\vec{r}_{ij}}{r_{ij}} \tag{15.61}$$

The multi-body part:

$$f_k^{\mathrm{multi}} = -(-\vec{\partial} \sum_i \rho_i^{1/2}) \tag{15.62}$$

$$= \sum_i \frac{1}{2\rho_i^{1/2}} \sum_{j\neq i} \xi^2 \mathrm{e}^{-2q(r_{ij}/r0-1)} \frac{-2q}{r_0} \frac{\vec{r}_{ij}}{r_{ij}}(\delta_{kj} - \delta_{ki}) \tag{15.63}$$

Let's denote

$$\rho_{ij} = \xi^2 \mathrm{e}^{-2q(r_{ij}/r0-1)}), \quad q_{ii} = 0 \tag{15.64}$$

After summing up and replacing $k \rightarrow j$ we get

$$\vec{f}_k^{\,\text{multi}} = \sum_i \vec{f}_{ij}^{\,\text{multi}}, \quad \vec{f}_{ij} = -\frac{q}{r_0}(\rho_i^{-1/2} + \rho_j^{-1/2})\rho_{ij}\frac{\vec{r}_{ij}}{r_{ij}} \tag{15.65}$$

The algorithm:

1. pair loop, ρ_{ij} are calculated

2. sum, $\rho_i = \sum_j \rho_{ij}$

3. second pair sum where U^{pair} and contributions f_{ij} are calculated

4. $U^{\text{multi}} = -\sum_i \rho_i^{1/2}$ can be now finished

Chapter 16

File formats

This section is incomplete and occasionally obsolete...

16.1 Configuration

Files <u>simname.1</u>, <u>simname.2</u>, ..., and <u>simname.cfg</u> are binary files of variable record length. They consist of records of the following form

field	length/B	interpretation
header	4	length of data n as 4-byte integer
data	n	data in binary format
checksum	4	$19019 + \sum b_i$ cast to 4-byte integer, where b_i are
		bytes of data cast to unsigned char

The end-of-file is marked by a zero integer (because no record may have zero length) and by the system time (long integer, time in s since the standard time/date).

A list of record data follows. It applies to cook version $> 2.7a$; to read an old version, use `cfgconv` to convert. 'int' means 4-byte integer.

`int cfgkey` File version. Bits 0 and 1 mean:
  1 = cook version $\geq 2.7a$
  2 = POLAR version

`int optionlist[32]` The values of options, -@, -a, ...; run cook without a parameter for a list of them.

`int nspec` Number of species

`int spec[nspec]` Array [`nspec`] of species data:
  { int N; int ns; }
  where `N` = number of molecules and `ns` = number of sites in this molecule. The order of species is the same as in the ble-file.

`rec[*]` Records of form
   `{ int key; int intval; vector vecval; }`
   in any order, where the vector is
   `typedef double vector[3]`.
   The following keys are recognized:

    1 `intval=N` (total number of molecules)
     `vecval={t,h,Etot}` (running simulation time [ps], time step [ps], total energy [K])

    2 `intval=No.s`=total number of sites
     `vecval=L[3]` (box sizes, vector in [Å])

    3 (reserved for cell shape)

    4 `intval = thermostat`
     `vecval[0]=RvdW` (for van der Waals radius setup, see variable `tau.sig`)
     `vecval[1]=tau.T`
     `vecval[2]=tau.P`

    5 `intval` = number of cleaving planes
     `vecval[0]`,`vecval[1]` = positions of cleaving planes

    17 `vecval[0]=tau.dip` (POLAR only)

   Note that these values are mostly for checks and most of them are set while reading the simulation data.

`ToInt cfg[GearOrder+1]` The configuration, see `ToInt` in `simglob.h`. `cfg[0]` contains the size in bytes as integer first, then possible void bytes to satisfy alignment requirements, then the Nose variable `log(s)` as double, vector `lambda` for the barostat, vector `aux` reserved, and then `No.s` double vectors corresponding to all sites; for the POLAR version, the Drude positions (another `No.s` double vectors) follow. `cfg[1]` contains the first derivative of `cfg[0]`, multiplied by `h` (differences for Verlet/SHAKE), and so on as in Taylor expansion. For the Verlet/SHAKE simulation, `GearOrder=2`.

16.2 Convergence profile

For the main cp-file written by cook, see Sect. 14.2.

File simname.cp is a fixed record length file composed of n 4-byte words; n is also known as NCP. It must hold $1 < n < 65536$ (was smaller in old versions).

`1st record` {CPmark,NCP,COL3,COL4,..} defines the header:

   `CPmark` (float)-1.76398512e+37

   `NCP` integer n

   `COL3` (max 4 chars) name of field 3

   `COL4` (max 4 chars) name of field 4, etc.
    Note that COL1,COL2 are assumed to be COL1=Etot, COL2=T

   In old versions, NCP was used to determine sex (byte order), this is no longer supported. A very old version without this header assumed $n = 5$ with quantites {Etot,T,Epot,Ein,P}.

`control record` is recognized by the first 4 bytes interpreted as float; see `sim/cpmark.h` for the definitions:

`CPmark=(float)-1.755645e+37` 24 bytes follow with the ASCII write time (when the record was written); if $n < 7$, this string is truncated.

`CPmarkT=(float)-1.76398512e+37` For $n = 2$, 4 following bytes are interpreted as int; for $n > 2$, 8 following bytes are interpreted as long int = time_t. The number is the write time in s since 1970-01-01 00:00:00 +0000 (UTC). Use system function ctime() to convert to ASCII.

`CPmarkU=(float)-1.77232525e+37` For $n = 2$, 4 following bytes are interpreted as float; for $n > 2$, 8 following bytes are interpreted as double. The number is the simulation time of the previous record (or simulation start) in ps.

`CPmarkV=(float)-1.78066538e+37` For $n = 2$, 4 following bytes are interpreted as float; for $n > 2$, 8 following bytes are interpreted as double. The number is the time lag (simulation cycle DT=h*noint) in ps separating the following data (until another control record).

`CPmarkW(float)-1.7890055e+37` For $n > 4$, 16 following bytes are interpreted as two doubles, containing t and DT (see above). This is the preferred format.

`data record` is interpreted as n float numbers. Their meaning is defined in the header (except COL1=Etot, COL2=T). See Sect. 14.2.

16.3   Playback file

The playback file is a file of single precision floating point numbers (4 bytes long). It contains a header of two floats, the 1st one being the number of sites `ns` (in the float format) of the molecule/configuration, the 2nd contains

`L=0` free boundary conditions (old format)

`L>0` periodic b.c. - the box size L (fixed, old format)

`L=-3` New format enabling variable vector L. One frame consists of float vector `L[*]` first, then the configuration (`ns` float vectors) follows.

The body of the file consists of configurations recorded at regular time intervals.   Each configuration contains `ns` records of vectors of 3 floats `(x,y,z)`.

The extension of playback files is `.plb`.

Chapter 17

Examples

17.1 Crystals

Crystals must match to a rectangular box. Thus, monoclinic and triclinic crystals are not supported.

naclcryst Prepare rotated rock-salt crystals.

ice Prepare ice and clathrate crystals.

plbreplicate To replicate a crystal (not for mixtures)

cook Using `init=5` or `init="cryst"` with optional `pins` set can create basic crystal structures.

A more robust procedure will be explained using the wurtzite-type crystal (assuming that LiI crystallizes as wurtzite, which is the case in some force fields). A basic wurtzite cell must be replicated twice to fit into a rectangular box, see fig. 17.1. In the ATM-format, with ideal tetrahedral angle and atom-atom distance of 1 Å we have (file `wurtzite.atm`

```
8
 1.6329932  2.8284271  2.6666667    ! \(8/3) \8     8/3
I  0           0           0
I  0.81649658 0.47140452 1.3333333  ! \(2/3) \2/3   4/3
I  0.81649658 1.4142136  0          ! \(2/3) \2     0
```


Figure 17.1: Wurtzite structure viewed along the *c*-axix. Red: basic cell, blue: doubled cell matching a rectangular box

```
I  0          1.8856181  1.3333333  ! 0      \32/3  4/3
LI 0.81649658 0.47140452 0.3333333  ! \(2/3) \2/3   1/3
LI 0          0          1.6666667  ! 0      0      5/3
LI 0          1.8856181  0.3333333  ! 0      \32/3  1/3
LI 0.81649658 1.4142136  1.6666667  ! \(2/3) \2     5/3
```

You need also `I.che` and `LI.che`:

```
------- LI.che ------
ion Li+
parameter_set=JC+SPCE
LIp1
------- I.che -------
ion I-
parameter_set=JC+SPCE
Im1
--------------------
```

Possible script to make a crystal of Li180I180:

```
bonds wurtzite.atm                         # make wurtzite.plb (and .mol,.gol)
plbscale wurtzite.plb li180i180.plb "*2.8" # scale close to real Li-I distance
plb2cfg li180i180.plb li180i180.cfg        # convert to .cfg (accept WARNING)

blend -o LiI LI I # make force field

# sample def-file
cat > li180i180.def <<EOF
N[0]=180  ! Li
N[1]=180  ! I
rho=3280  ! approx. crystal density
tau.rho=2 ! density relaxation time [ps]

cutoff=11.2 ! approx. half box size
LJcutoff=cutoff

L[0]=5*\(8/3) ! box shape (not in scale)
L[1]=3*\8     ! box shape (not in scale)
L[2]=8/3*3    ! box shape (not in scale)

load.n[0]=5   ! cell will be 5x replicated in x
load.n[1]=3   ! cell will be 3x replicated in y
load.n[2]=3   ! cell will be 3x replicated in z
load.N=3      ! tell cookew to ignore N=0 in the input file
init=2        ! will read li180i180.cfg
thermostat="Andersen" tau.T=0.1
dt.plb=1
;
EOF
```

```
# run as
cookewslcP1 -s LiI li180i180

# enter
no=100;
then Ctrl-D

# NB: cannot be repeated because li180i180.cfg has been rewritten
```

After this step, all `load.*` should be deleted from `li180i180.def`. Suggested `li180i180.get` follows:

```
! cookewslcP1 LiI li180i180
no=10000
LJcutoff=10.95
cutoff=LJcutoff
init=2
thermostat="Berendsen" tau.T=1
bulkmodulus=2e10
tau.P=5
tau.rho=0
rescale="XYZCM"
rdf.grid=25
rdf.cutoff=cutoff
;
```

To run:

```
  cookewslcP1 LiI li180i180
```

Watch RDF and coordination numbers by one of

- `rdfg -g -c -p li180i180`

- click `li180i180.rdf` from the Midnight Commander

- `start li180i180.rdf`

Hint: right click to show color codes of curves.

After equilibrating, productive runs should use Nosé-Hoover thermostat or MTK thermostat and barostat for better quality.

17.2   Protein in water

This is a continuation of the example from the blend manual, see Sect. 7.2.

Let us choose a serial version with cheap electrostatics, `cookceslc`.

We have prepared (see Sect. 17.2) `crambinw.ble` containing one molecule of protein crambin and 999 water molecules. The protein has bond lengths constrained, water molecule is TIP3P treated as rigid body. Now we want to simulate this protein.

First, we must prepare the initial configuration. We will run `cook` interactively so that we need a default (.def) file only. We name our project `crambin` so that the default file will be `crambin.def`. A commented example of file `crambinw.def` follows:

```
  T=310                    ! temperature in K
  thermostat="friction" ! Berendsen thermostat
  tau.T=0.1                ! time constant (start only)
  rho=1050                 ! final density in kg/m^3 (this is the default)
  initrho=900              ! initial density
  tau.rho=1                ! typical time for initrho->rho condensing
  cutoff=15                ! electrostatic cutoff
  LJcutoff=12              ! Lennard-Jones cutoff
  equalize.cfg=0.8         ! equalize masses
  h=0.002                  ! [ps] timestep (use 0.001 without equalization)
  noint=5                  ! steps/cycle
  ;                        ! end of data set
```

Now, let us run `cook*` interactively:

```
  cookceslc -s crambinw # serial
or
  NSLOTS=4 cookceslcP1 -s crambinw # parallel
```

You will be prompted for data. Enter:

```
  init="random"     ! random initial configuration of molecules
  ;
```

You may see some warnings "dr in 1 timestep reduced" as a consequence of large initial temperature. They will disappear as soon as the system equilibrates. Ignore WARNINGS (type `c Enter`).

Time evolution of temperature (incl. graph), potential energy, total energy, density, pressure, and accuracy of constrained dynamics is printed. After several ps, interrupt the calculations by pressing Ctrl-C; in the serial version, select .   `eneter` = stop, in the parallek version, type ‾Ctrl-D‾ when new data are requested, or `key=''quit''`.

To generate a longer trajectory, create file `crambinw.get`:

```
init="start"       ! start simulation (from .cfg)
dt.plb=1           ! write playback every 1 ps
dt.prt=1           ! write line to prt every 1 ps
tau.T=1            ! thermostat time constant
tau.rho=0          ! turn off fixed box
bulkmodulus=2e9    ! [Pa] for Berendsen barostat
tau.P=5            ! barostat constant
no=1000            ! number of cycles (by h*noint=0.01 ps)
```

```
;                      ! end of data set
init="append"          ! restart measurements but append playback and .cp
no=1000                ! another 10 ps
;                      ! end
```

Start simulation by command:

```
  cookceslc crambinw
or
  cookceslcP1 crambinw
```

The output protocol is `crambinw.prt`. The simulation can be interrupted (with everything saved) either by signalling -2 (-SIGINT) to the process, or by creating file `crambinw.stp`.

To watch the convergence profiles, run

```
  showcp -p crambinw
```

if `plot` is installed properly, you should see the time development of quantities during 30 ps.

To watch the trajectory, check that files `crambinw.mol` and `crambinw.gol` have been created; if not, use command like

```
  molcfg crambin TIP3P crambinw
```

To show, use (may be used also while cook is running)

```
  show crambinw
```

Useful variants include

```
  show crambinw -Ob
  show crambinw -o396 -I%@ -l -x10 -y2 -z2
```

The first form omits all waters, in the second 396 is the number of atoms in crambin, the values of shift -x -y -z should be set according to the actual position.

Use mouse GUI or hot keys: $\boxed{\text{g}}$ $\boxed{\text{G}}$ to cycle the viewing mode, $\boxed{1}$ to start playback, $\boxed{0}$ to stop it, $\boxed{\text{i}}$ to start again.

Part III

Utilities

This manual page describes various utilities and supporting software. Some information may be incomplete. A brief but up-to-date help on most utilities can be obtained by running the utility without any parameter. In some cases (`ray`, filters), option `-h` may be needed. Most utilities also print names of related utilities as `See also:`.

Utilities marked by * in front of the *Source field need a make file (see Sect. 18.1), which is generated from `metamake.mmk` by utility `makemake`, otherwise they can be compiled by a command written in the first line of the source file.

Chapter 18

General utilities

18.1 `makemake`: Makefile and project interface

Projects are in MACSIMUS written in files called `metamake.mmk`. Utility `makemake` then finds dependencies and converts these files to standard makefiles (historically, also to Turbo C projects). See the source `c/makemake.c` for more info.

Normally, installing MACSIMUS starts by running `./install.sh`. For installing `cook*` versions, run `./configure.sh` from the `cook` directory. For customizing `blend`, see `blend/metamake.mmk`.

Without the above scripts, installing MACSIMUS software typically requires the following steps:

1. Edit the project `metamake.mmk`. The sample metamake.mmk files contain enough comments...

2. Running `makemake` may be as simple as

   ```
   makemake gcc
   ```

   or in case of project- and system-dependent switches something like

   ```
   makemake colorgcc polarlj
   ```

3. If there are no errors, run `make` on the requested target(s), e.g.

   ```
   make blend pdb
   ```

See comments in the source file for details!

Source: `c/makemake.c`

18.2 `plot`: Plot a graph (with formulas and mouse-rescaling)

Plots graphs of tabular ASCII data. This utility allows to plot formulas as well us use a mouse to rescale – it did so several years before `gnuplot` supported these features. It has developed

from the viewer of the NSK project and I consider it as the most useful general program I have written.

`plot` accepts input files in similar format as `gnuplot`: of white (SPACE, tab) separated numbers. Comment lines are marked by `#` or `!` in the 1st column. When plotting by lines, the comments as well as blank lines break the plotted curves.

Get help by running `plot` without parameters.

More features (run `plot` without parameters for help):

- Fitting (max 10 parameters) with error estimates and the variance-covariance matrix. Error estimates of integrals, derivatives, solutions of equations. High-precision arithmetics if needed.

- EPS and PS output – simple, but usable and editable.

- Re-read changed files after a script is called.

Examples:

Simple examples:

```
plot sim.g          # plots file sim.g: column 2 (y-axis) vs. col. 1 (x)
plot sim.g:2:3      # plots column 3 vs. column 2
plot sim.g:0:3      # plots column 3 vs. consecutive numbers 0,1,2,...
plot sim.g:0:3:p    # as above, using points
plot sim.g:0:3:p:4  # as above, using points w. error bars in col. 4
plot sim.g:A:B^2    # plots col. 2 squared vs. col. 1
plot :1:3 *.g       # plot all *.g files using cols. 2 and 3
plot "[0:10]" "[1000]:A:sin(A)" # plot function y=sin(x)
plot "[0:1]" FILE:A:B:o:C "[111]:A:(a+b*A^(1./4))/(1+c*A)":-:" # fitting
plot "[0:]" ":1/A:B+a/A:-o:C" data1.dat # use slider or aA to change a
```

To export PostScript file (`plot.ps` or `plot.eps`), use control data in `ps.def` (or defined by `-nNAME`); see a sample file `man/ps.def`. This file may contain the following commands. The command key letter must start from column 1. All sizes are in pt (only for command `w`, negative values denote cm).

`!` `anything`

`#` `anything` comment

`s` `SIZE` `[NAME`] Set font size (default=14) and name (default=Helvetica)

`m` `l` Set landscape mode

`m` `p` Set portrait mode

`m` `e` Set eps mode (encapsulated postscript, default)

`x` `LEFTMARGIN` `RIGHTMARGIN` `TEXT` `[TEXT`] Set margins (of the graph without axes labelling) of axis x and its label.

y BOTTOMMARGIN TOPMARGIN TEXT [TEXT] Set margins (of the graph without axes labelling) of axis y and its label.

w XSIZE YSIZE Set window size. Negative values denote sizes are in cm. Note that command `m p` used after `w` swaps XSIZE and YSIZE. Default is `w 576 432`, i.e., 8x6 inches.

t LINETHICKNESS FRAMETHICKNESS Set the default line thickness and the thickness of the frame and tics. The default is `t 1 0.5`.

f NOXTICKS NOYTICKS Sets the approximate number of ticks and corresponding labelling on axes. The default is `f 5 5`. The program determines the tick distance to be a round number of a power of 10 time 1, 2, or 5, and thus the actual number of ticks may differ.

 # R G B[,DASH,LINEWIDTH] Set the color (in RGB) of line number `#` (1st file=0, 2nd file=1, if not changed by hot keys $\boxed{c}$, $\boxed{C}$). DASH is a space-separated even number of numbers denoting DASH PAUSE DASH PAUSE... Example: `1 1 0 0,5 3 1 3,0.5`

r ANGLE Rotation angle for subsequent `l` and `y`, in degrees.

l X Y STRING

l X +[DY STRING] Print string at given position. The second form advances from the previous position down by DY (default DY=line feed).

STRING for commands `l, x, y` may contain the following characters:

****   Next character is symbol (Greek letter): `\a` is alpha, etc.

_   Next character is subscript

^   Next character is superscript

Examples: `t_c_o_r_r/ps`, `r^\a`

*Source: `show/plot.c`

18.3   `tabproc`: Command-prompt spreadsheet

Some people like `excel`, I like command prompt and pipes... Input data are processed using given formulas and formats and.

```
tabproc EXPR1[:FMT1] [EXPR2[:FMT2]] ... < INPUT > OUTPUT
```

EXPR Expression made of letters `A B C...` for columns 1,2,3..., and usual mathematical operators and functions. Instead of `A B C...`, `#1 #2 #3...` can be used. Consecutive numbers are coded by `@` or `n` and are reset to zero on a blank line. Examples of expressions: `A/2+sin(B)`, `#1/2+sin(#2)`, `@*pi^2`.

FMT Format in the C-style; the initial `%` may be omitted. Examples: `7.4f`, `%7.4f`, `err=%.2e`. The default format is that of the previous argument or `%g` if no format is given.

Lines beginning by # or ! are copied as comments. Max 26 columns are allowed. Environment variable NOLF suppresses line feeds (may be useful in scripts, see Sect. 18.13).

Use `tabproc` for more options.

*Source: `c/tabproc.c`

18.4   `field`: Select white-separated field(s)

Example:

```
SEP=",;" field 3 < FILE
```

Source: `util/field.c`

18.5   `filttab`: Select range from a table

Example:

```
cat DATA | filttab -1:-3:5 | filttab 2:0
```

Source: `util/filttab`

18.6   `avfiles`: Average files by coumns

Example:

```
avfiles -05%.9f sys[0-9].cpa > avsys.cpa
```

Source: `util/avfiles`

18.7   `blocktab`: Calculate subaverages over blocks

Example:

```
blocktab 10 2 < DATA # 2-column table by 10 lines
```

Source: `util/blocktab`

18.8   `sumetc`: Calculate sums, errors, variances...

Example:

```
sumetc -m -M -a -e < DATA # min max average stderr
```

Source: `util/sumetc`

18.9   `sumerr`: Average of data with standard errors

Example:

```
sumerr 1 <<EOF
1.34 0.15 100
1.42 0.21 50
EOF
```

Source: `util/sumerr`

18.10   `latextab`: Format a table for LATEX

Example:

```
latextab 1:s 2:\\Chem\{%s\} 3:.5f 4:.2f 5 < antoine.txt > antoine.tex
```

Source: `util/latextab`

18.11   `mergetab`: Merge columns of data

This utility merges columns (white-separated) of data. Lines beginning with `#` are ignored.
Call by:

```
mergetab [FILE1]:[-OMIT:][/STRIDE:]COL:COL...
    [FILE2]:[-OMIT:][/[-]STRIDE:]{COL|COL=COL1}:{COL|COL=COL1}...
    ...
```

`FILEi` File name, or `-` for stdin (max once allowed). Missing `FILEi` repeats previous, missing `FILE1`=stdin.

`OMIT` # of noncomment lines omitted from the top of file

`STRIDE` Consider only every `STRIDE`-th noncomment line

`-STRIDE` As above, starting with `(STRIDE-1)`-th line `OMIT` and `-STRIDE` may be combined, e.g., `:-2:/-10` is the same as `:-11:/10`

`COL` Column to extract (print)

`COL=COL1` Synchronizes with column `COL1` of `FILE1` but does not print. The data in columns to synchronize should be in increasing order.

Example:

```
mergetab st2.g:1:2 tip4p.g:1=1:2 | plot -:A:C-B
```

will pipe columns 1 and 2 of `st2.g` and column 2 of `tip4p.g` shifted so that it matches column 1. If `tip4p.g` has finer grid, extra data are omitted, if `st2` finer grid, missing data in `tip4p.g` are marked as n.a. The piped stream has thus three columns. Finally, the difference of tip4p and st2 is plotted.

Source: `c/mergetab.c`

18.12 `tab`: Column table of consecutive numbers

Call by:

```
tab FROM TO [BY [FORMAT]]
```

`TO` is included. If no `BY` is given, 1 is the default. If no `FORMAT` is given, "%g" is the default.

Example (Fourier transform of a saw-like function, 20.11)

```
tab 0 100 0.01 | \
  tabproc "A-int(A)" | \
  spectrum 10000 | \
  plot -:0:1
```

Source: `c/tab.c`

18.13 `ev` and `evu`: Calculators

Bash or prompt simple calculators, run `ev` = without units, `evu` = with units. Run without parameters for details.

Features:

- Variables, formulas, common functions in the usual syntax

- Embedded plot, sum, iteration, loop, continuous fraction

- Numerical integration, 1st derivative, solve 1 equation, min/max in 1D

- Macros

- Units (except units for light and radioactivity)

- Conversion from/to MACSIMUS internal units

- Predefined constants

- LaTeX output

Features/caveats/bugs:

- `-2*3` (with `-` as the 1st character) **subtracts 6 from the previous result**, this feature can be turned off by:

- option `---`
- leading space: `-2*3`
- parentheses: `(-2*3)`

- octal numbers: `047` = 39

- power is left-associated: `2^3^4` = $(2^3)^4$

- unary `+`,`-` have the same precedence as binary: `-2^2'` = $-(2^2)$

- multiple unary `+`,`-` are allowed: `--2` = 4

- leading space prevents commands to be recognized: try `deg=1;deg; deg`

- macros cannot be used with ;-separated statements (except the last one)

- cannot add/subtract different units: `0+1[m]` is error, `0[m]+1[m]` is correct

- `M(CuSO4.5H2O) = M(Cu)+M(S)+4.5*M(O)+M(H2O)`; for blue vitriol use: `M(CuSO4H10O5)`

Examples (from bash):

```
ev "pi^3*exp(2)"
evu "R*25 oC/1[bar]"
```

Example (prompt interface), note that `solve` must start the line:

```
$ evu --c
DH=-98[kJ.mol-1]
Cw=75[J/mol K]
Co=29 [J/mol K]
DT=35[K]
solve n=10 DH+DT*(n+1)*Cw+0.5*Co*DT
w=M(H2O2)/(M(H2O2)+n*M(H2O))
DvypH=42.5[kJ/mol]
solve n=10 DH+DT*(n+1)*Cw+0.5*Co*DT+0.25*DvypH
w=M(H2O2)/(M(H2O2)+n*M(H2O))
```

Source: `c/ev.C`

18.14   `start`: Start application according to file extension

This is a command-prompt analogue of the GUI mechanism of starting application according to the type of the associated file. On some versions of Windows there is command 'start' of the same function. Such a style of work may be considered strange by Windows folk. But my brain is not able to find a file of interest among more than ten icons in a graphical folder (or Norton Commander list) and I consider typing the file name much faster (especially with file completion and wildcards).

Simple examples:

```
start pig.jpg                --> display pig.jpg
start archive.zip            --> unzip -v archive.zip | less
```

Extended examples:

```
start -expand 2.5 pig.jpg   --> xv -expand 2.5 pig.jpg
start pig.jpg -expand 2.5       ERROR unknown ext
start pig.jpg "-expand 2.5" --> xv pig.jpg -expand 2.5
start mol cfg.plb -I%        --> show mol cfg -I%
```

Details:

1. Scans arguments given and determines the extension of the last file argument. (File argument is an argument not starting with - nor + and with an extension and not containing a space. The extension is the suffix after the last . in the file name provided that the name before this . is not empty.)

2. Looks up the associated application

3. Starts this application with arguments, using system()

File registration:

Edit the array of structures `reg[]` in `start.c` and recompile. The last line must be `NULL,NULL`.

Source: `c/start.c`

18.15 `sortcite`: Sort LaTeX citations

This utility sorts all \bibitem{ } statements in the thebibliography environment so that they appear in the order in which they are referenced using statements \cite{ }, \Cite{ }, and \Hide{ } (see `cite.sty` for the latter two).

Usage:

```
sortcite FILE [-]
```

`FILE.tex` is input, `FILE.ren` output. Optional `-` means that not referenced bibitems will be removed from output.

Source: `c/sortcite.c`

18.16 `lemon` Extension of `grep`

Example:

```
lemon NEEDLE -REPLACE FILE FILE.. # replace string
lemon NEEDLE +5 FILE # find and show +5 lines
lemon # help and a list of useful utitlities (not all them in MACSIMUS)
```

Source: `c/lemon.c`

18.17   `line` **Extension of** `head tail`

Example:

```
line 4-7 < FILE
```

Source: `c/line.c`

Chapter 19

Program 'pdb' version 1.5a

`pdb` reads protein in the PDB format and converts it to mol-format understood by program `blend`. See also `pdb2pdb`, see Sect. 23.1 (rearranges the lines of pdb-files to a 'more standard' order).

19.1 Running

19.1.1 Environment

Environment variable `BLENDPATH` can be set to point to the path that contains the needed residue-filed (`*.rsd` by default). If it is empty, the files are looked for in the working directory. A subdirectory of this directory, i.e., the used set of the residue files, may be specified by option `-r`.

Example for UNIX (csh, tcsh):

```
setenv BLENDPATH /home/jiri/macsimus/blend/data
```

Example for UNIX (sh, bash):

```
export BLENDPATH=/home/jiri/macsimus/blend/data
```

19.1.2 Synopsis

The command line to run `pdb` is:

```
pdb [options] pdbname [molname]
```

If molname is missing then it is assumed that molname=pdbname.

Run `pdb` without options to get a brief help.

19.1.3 Files

pdbname`.pdb` Source PDB-file

molname.`pdb` Output PDB-file (only if option `-p`)

molname.`####.pdb` Output PDB-file (only if option `-p` with range)

molname.`rep`

pdbname.`rep` Optional pattern replacement file. If molname.`rep` is not found, pdbname.`rep` is tried, if this file does not exist either, no patter replacement occurs. Example (to fix wrong OT type for C-O-C oxygens in sugar-sugar bond; `pdb` has been run with `-d2` option).

```
CH1E OT-0.65 CH1E      CH1E OE-0.5 CH1E
CH2E OT-0.65 CH1E      CH2E OE-0.5 CH1E
CH1E OE HO             CH1E OE DEL
CH2E OE HO             CH2E OE DEL
```

It means that all structures `CH1E-OT-CH1E` with partial charge on `OT` -0.65 will be replaced by `CH1E-OE-CH1E` with partial charge on `OT` -0.5, etc. `DEL` means deletion of the atom. The file is executed by lines so that all changes caused by one line are available when executing the following line. Currently only replacement in groups by 3 atoms defined by atom types is available. No wildcards available for atom types. If necessary, will be extended in future.

molname.`sel` Residue selection file. Example:

```
option -n0 ! should match with runtime option -n
! neutral versions of all negatively charged residue
!RSD # chain replacement
 ASP *   *   asph
 GLU 11  A   gluh
```

The format is free, `-` in the chain field means no chain, `*` in the chain field means any chain, `*` or `-1` as `#` matches any residue irrespective of its `#` in the PDB file. `RSD` MUST be uppercase, replacement refers to `.rsd` files and for UNIX is case sensitive, however, lowercase is recommended.

molname.`mol` Output mol-file (see `blend`) Default output from `pdb`.

molname.`3db` Molecular configuration in binary format `float[][3]`, deprecated (use .plb), see option `-b`.

molname.`plb` Molecular configuration in the playback format, the default since V1.4a (see the manuals for `cook` and `blend` and option `-b`).

molname.`3dt` Molecular configuration in text 3 column (x y z) format.

molname.`plt` Molecular configuration in text 3 column format with playback-like header line (one line of 2 numbers, 1st=number of sites, 2nd=0).

`*.rsd` Extension of the residue files, see also BLENDPATH and option `-r`.

19.1.4   Options

-a Some atoms in the pdb-file may be given alternative locations, i.e., different coordinates. They are numbered by letters A,B,...

> **-a0** Use location A (default)
>
> **-a1** Use average of all location weighted by occupancies
>
> **-a2** Use location with maximum occupancy number.

-b Write (if not **-p**) or read (if **-p**) binary file with the molecular configuration.

> **-b0 -b-** Don't read/write any binary coordinate file.
>
> **-b -b1** Binary coordinate file is molname.**3db**.
>
> **-b2** Use playback file molname.**plb** as the molecular coordinate file (default).
>
> **-b3** Both **-b1** and **-b2** (not for input).
>
> **-b-1** As **-b1** with reversed endian.
>
> **-b-2** As **-b2** with reversed endian.
>
> **-b-3** Both **-b-1** and **-b-2** (not for input).

-cnumber Calculate S-S (CYS-CYS) bonds from the configuration. number is the distance limit in Å, missing number means distance limit 3A (equilibrium bond length is 2.04A). Use if **SSBOND** and/or **CONECT** commands are missing.

-dkey Delete (do not include) the atoms which are in the residue files but are not present in the pdb-file. Note that typically hydrogens are NOT present in the pdb-file and should be included but sometimes a missing heavy atom should be removed because a chemical bond is present (hopefully described by CONECT statement). E.g., sugars are given with all -OH groups in the residue files but if they are chemically bonded then two -OH groups create one -O- bond and one O and two H should be removed. (See also molname.**rep**). The values of key are as follows:

> key>0 Deletions apply only to MOLECULES (1st keyword in the residue file must be **molecule**
>
> key<0 Deletions apply to all residues
>
> **1,-1** Deletions apply to hydrogens only
>
> **2,-2** Deletions apply to heavy atoms only; free hydrogens after removing these heavy atoms are also removed.
>
> **3,-3** Deletions apply to any atoms

-eRSD Use RSD.**rsd** as end (C-terminal patch) for ending peptide chains. The default is **cter.rsd** (COO$^-$). Extension **.rsd** can be changed by option **-f**.

-fPARSET PARSET.**par** will be the used force field (parameter_set). Information on the parameter set is written to the mol-file created so that **blend** can make use of it. In addition, file PARSET.**par** contains line **rsddir** rsddir with the name of the directory in which the residue files (extension *.**rsd**) are looked for. If no **-f** is specified, PARSET=RSDDIR is assumed (see option **-r**).

-g Gap in residue numbering terminates chain. If the difference of numbers of two consecutive residues is not 1 than the protein chain is terminated (normally by CTER) and started again (normally by NTER), as if there was a TER statement between them.

-hRSD Use RSD.rsd as head (N-terminal patch) for starting peptide chains. The default is nter.rsd (NH_3^+). Extension .rsd can be changed by option -f.

-ikey Ignore atoms which are present in the pdb-file but are not supported by the residue-files. Typical case is when all hydrogens are in the pdb-file (incl. aliphatic H) but the aliphatic H are not supported in the united atom representation. Typical key is 1 (ignore H in molecules) or -1 (ignore all H, in molecules and aminoacid residues). See option -d for details of key.

-m- -m0 Don't write .mol file

-n Use neutral residues:

-n0 Charged residue are used for acids (ASP and GLU) and bases (ARG, LYS, and HIS). This is the default. In V1.3c and older, HIS was neutral by default, now HIS is charged (protonated) by default and -n (see below) specifies its uncharged state.

-n Use neutral residues instead of 'standard' charged ones:

```
ARG  -> ARGN
ASP  -> ASPH
GLU  -> GLUH
LYS  -> LYSN
HIS  -> HISN
NTER -> NTERN
CTERH -> CTER
```

(In versions V1.3h and older did not work for the termini.)

-n-1 Use charged residues and compensate charges by counterions Na+ or Cl-. Also charges of NTER and CTER are compensated. NOTE: Positions of ions are calculated from known atoms of the side-chain (for ARG, ASP, GLU, LYS, HIS) or backbone (for terminals) so that the ion is placed in the direction of the chain off the molecule. E.g., for N-ter the formula is 2*r[N]-1*r[CA]. When counterions are used, blend should be called for the first time with option -k-3 to allow energy minimization with the molecule skeleton fixed and free ions and hydrogens.

-n-% As above where the distance of the counterion from the ion being compensated is -% percent of the previous bond., e.g.: (q/100+1)*r[N]-q/100*r[CA]

-oRSD Omit residue RSD. Typical use is such as -oHOH (or -oWAT—check the PDB file!) to remove all water molecules.

-pnumber[:FROM:TO] Paste a configuration from input file (typically the playback file molname.plb to a PDB-file. Both pdbname and molname must be specified on the command line (and must differ). File pdbname.pdb is the input PDB-file and molname.pdb the output. The source and format of input configuration is given by options -b and -t (only one of them may be active; since -b is default, it must be turned off if -t is to be used). -pnumber with a positive argument selects number-th configuration on the playback file (supported only for binary format), -p is the same as

-p1 (the 1st configuration), -p-1 means the last configuration in the playback file, -p-2 the 2nd last, etc. The version with range processes a range of frames. Option -b2 meaning playback file molname.plb as input is forced (also -t2?). The output files are numbered molname.####.pdb, where #### is the frame number. In this case, molname=pdbname is allowed. Cf. options -r4:FROM:TO:BY -w10 | -w20 of blend.

-qnumber Charge multiplication factor in %. Default is 100%.

-rRSDDIR The directory of residue files, relative to BLENDPATH (see Sect. 19.1.1). If also option -fPARSET is given, option -rRSDDIR overrides the value of rsddir stored in file PARSET.par. Slash or backslash after directory name is not allowed.

-snumber Scrolling enabled with number kB buffer, see blend for details.

-t Write (if not -p) or read (if -p) text file with the molecular configuration.

> -t -t1 Coordinate file is molname.3dt.
>
> -t2 Coordinate file is molname.plt with playback-like header line.
>
> -t3 Both -t1 and -t2.

-u Enable more relaxed atom matching algorithm when assigning PDB atoms to atom names in the residue files; for instance, with -u, HE3 in PDB-file matches HE1 in the residue file if there is only one (Unique) HE atom in the residue file. Without -u, they do not match. If the PDB-file contains aliphatic hydrogens (but no charged or hydrogen bonded hydrogen), the default -u0 must be used. Some PDB formats, however, may use different numbering and then option -u may help.

-v Verbosity level.

> -v -v1 Verbose (default).
>
> -v- -v0 Most of the info messages and warnings are suppressed.
>
> -v-1 Partly verbose. Only connection and re-connection reports are suppressed.

-x Wildcard A in atom names in the PDB file is enabled. This occurs when a group like -CNH$_2$O in GLN may rotate by 180 degrees so that it is not possible to distinguish both heavy atoms.

-z (new in V1.3n)

> -z0 Patched residue keeps name of the residue even for atoms added (i.e., the added atoms are treated as part of the residue).
>
> -z2 The patched part has always the name of the patch (i.e., the added atoms are treated as new 'residue' under the patch name). If the patch is specified by -h/-e options, first three letters of the patch name (in uppercase) are used.
>
> -z1 (default) Option -z0 applies for patches specified in the PDB file while -z2 for patches specified by -h and -e options.

19.2 Residues

The topology (chemical structure) along with partial charges are defined in the residue files. The name of a residue file is usually the same as the residue name (but is lowercase) and the extension is defined by the parameter set used (see option `-f`).

19.2.1 Format of residues

The format of residue files is almost the same as `.mol` format described in the manual for `blend`. Differences are:

1. First line must start by one of the following keywords:

   aminoacid Aminoacid residue, to be connected by a peptide bond

   patch A part of molecule that modifies another molecule, for C/N-termini deprecated

   nter Chain head (N-terminus). A peptide bond to the next residue is added. This residue keeps its name in the PDB file.

   nterp Patch for chain head (N-terminus), merged with the following residue. A peptide bond to the next residue is added even if not specified in the patch.

   cter Chain end (C-terminus). A peptide bond to the previous residue is added. This residue keeps its name in the PDB file.

   cterp Patch for chain end (C-terminus), merged with the following residue. A peptide bond to the previous residue is added.

   water Water model

   molecule Any other molecule

2. If the type is `patch`, `nter`, or `cter`, atomid field may be divided by a colon into 2 parts. The second part is atomid to be replaced in the residue and the first part is the new name. Example:

   ```
   patch NH3+ (N-terminal patch for all aminoacids but GLY and PRO)

   number_of_atoms = 5
   ! charge=1

   atoms
   ! i  id      type  charge chir nb bonded_atoms
     0  N:N      NT   -0.30   0   4  1 2 3 4
     1  NH1:H    HC    0.35   0   1  0
     2  NH2      HC    0.35   0   1  0
     3  NH3      HC    0.35   0   1  0
     4  CA:CA   CH1E   0.25   1   1  0
   ```

Since *Nothing is compatible*, the same atom can be coded in different PDB files by different id's. These aliases can be listed in a residue-file as a comma-separated list.

```
  4  C,CY    C     0.51   0   0   2 0 5
```

Only the first one is printed in info/error messages.

19.2.2 Termini

The residue files for aminoacids contain residues as they enter the peptide chain (-N-C$_\alpha$-CO-). The first aminoacid (N-terminus) must be modified as follows:

- +charged (=protonated): there is NH$_3^+$ instead of N, patch=NTER

- neutral (`pdb -n`) : there is NH$_2$ instead of N, patch=NTERN

- counterion (`pdb -n-1`) : there is NH$_3^+$Cl$^-$ instead of N, patch=NTERCL

This is accomplished by applying given patch (a patch is represented by a residue file with keyword `patch`, `nter`, or `cter` in the 1st line, e.g., file `nter.rsd` is the patch NTER for charmm21).

Residues PRO and GLY are exceptional and different patches must be used:

`PRO:`
- +charged (=protonated): there is NH$_2^+$ instead of N, patch=PROP
- neutral (`pdb -n`): there is NH instead of N, patch=PROPN
- counterion (`pdb -n-1`): there is NH$_2^+$Cl$^-$ instead of N, patch=PROPCL

`GLY:`
- +charged (=protonated): there is NH$_3^+$ instead of N, patch=GLYP
- neutral (`pdb -n`) : there is NH$_2$ instead of N, patch=GLYPN
- counterion (`pdb -n-1`) : there is NH$_3^+$Cl$^-$ instead of N, patch=GLYPCL
  (The reason for a special treatment of GLY is because of CH2E instead of CH1E as C$_\alpha$)

`pdb` since V1.3i selects the appropriate patch automatically according to the value of option `-n`. However, if the patch is specified by `-h`patch, no check is made and the requested patch is used.

WARNING: names of hydrogens vary in different versions of pdb files and they need not be recognized. In PROPN the hydrogen is called HNC while in PROP the two hydrogens are called NH1 and NH2. This is not logical, but corresponds to the pdb files I have seen. Edit the files if you meet another names. Unrecognized hydrogens may be usually safely ignored—they are calculated in the blend stage.

19.2.3 List of residues

The most common residues follow. The corresponding file name is in lowercase and has the residue extension appended. One-letter code for aminoacids is in parentheses.

`nter ACE` acetylated N-terminus, CH$_3$-CO-

`aminoacid ALA` alanine (A)

`cter AMI` amidated C-terminus, -CO-NH$_2$, also called `CT2`

`aminoacid ARG` arginine (R)

`aminoacid ASN` asparagine (N)

aminoacid `ASP` aspartic acid (D), anion

aminoacid `ASPH` aspartic acid, protonated

cter `CT1` methylated C-terminus -CO-O-CH3

cter `CT2` amidated C-terminus, -CO-NH$_2$, also called `AMI`

cter `CT3` n-methylamide C-terminus -CO-NH-CH$_3$, also called `CMAM`

cter `CTER` C-terminal patch for aminoacids, anion -COO$^-$

cter `CTERH` C-terminal patch for aminoacids, protonated -COOH

aminoacid `CYS` cysteine (C)

aminoacid `CYSS` cysteine (to bind to another CYS)

aminoacid `GLN` glutamine (Q)

aminoacid `GLU` glutamic acid (E), anion

aminoacid `GLUH` glutamic acid, protonated

aminoacid `GLY` glycine (G)

nter `GLYP` terminal patch for GLY instead of NTER

molecule `HEM` heme BUGS: in some force fields, planarity is accomplished by different force fields terms than original. It may affect vibrational frequencies and similar. Binding of the heme with the protein can accomplished by files `.sel` and `.rep` files which are only partly available. Since there are 6 bonds of Fe, `blend` must be compiled with `#define` `MAXVAL 4` in `blend/blendpar.h`

aminoacid `HIS` histidine (H), cation (protonated)

aminoacid `HISN` histidine (neutral with proton on ND1)

aminoacid `HISNE` histidine (neutral isomer with proton on NE2)

water `HOH` TIP3P water model

aminoacid `ILE` isoleucine (I)

aminoacid `LEU` leucine (L)

aminoacid `LYS` lysine (K), positively charged

aminoacid `LYSN` lysine (K), neutral (not in all force fields)

aminoacid `MET` methionine (K)

nter `NTER` N-terminal patch, -NH$_3^+$, for all aminoacids but `PRO` and `GLY`

nter `NTERN` neutral N-terminal patch for all aminoacids but `PRO` and `GLY`

aminoacid `PHE` phenylalanine (F)

aminoacid `PRO` proline (P)

nter `PROP` N-terminal patch for `PRO` instead of `NTER`. `prop.che` is the whole residue

aminoacid `SER` serine (S)

aminoacid `THR` threonine (T)

aminoacid `TRP` tryptophan (W) **BUG:** In some force fields, planarity is accomplished by different force fields terms than original. It may affect vibrational frequencies and similar.

aminoacid `TYR` tyrosine (Y)

aminoacid `VAL` valine (V)

molecule `ZN` Zn++

19.3   Bugs and caveats

1. In spite of its name, MACSIMUS is now being developed towards simulations of fluids, ionic systems, systems with polarizability and not biochemistry applications. MACSIMUS' basic support of protein modeling and simulation will not be extended in future.

2. Only aminoacids are supported and a few other compounds.

3. There are several variants of the PDB format and therefore there is no guarantee that `pdb` will always work. See comments in the program (pdb.c) for details.

4. Information in PDB commands SSBOND and CONECT is duplicated. `pdb` writes a notice 'reconnected' if the same bond is connected again The same holds for option `-c`.

5. Cannot use different N- and C-terminal patches for different chains.

6. Some residues (esp. in alternate charge states) may be missing for some force fields.

Chapter 20

Data analysis

WARNING: This is obsolete, run the respective utilities without any argument for up-to-date help.

20.1 `showcp`: Show and analyze convergence profiles

`cook` (and in special cases also `blend`) records time-development of selected quantities (see the manual for cook, file extension `.cpi`). `showcp` enables viewing and analyzing these 'convergence profiles'. `showcp` recognizes the endian so that analyzing the cp-files on a different computer is transparent.

IMPORTANT NOTE: Always the *time development of quantities* is shown and never a *cumulative average* (sometimes also called 'convergence profile') from start of simulation because this quantity hides important information like sudden jumps, oscillations, etc! (To show a cumulative average, use `runsum`).

Usage:

```
showcp [OPTIONS] {SIMNAME | SIMNAME.cp | SIMNAME.cpz} [OPTIONS] [NAME ..]
```

Files:

`SIMNAME.cp` convergence profile file

`SIMNAME.cpz` packed convergence profile file (see Sect. 20.9)

`SIMNAME` try `SIMNAME.cp`, if it does not exist then `SIMNAME.cpz`

Options:

Select columns. The default (if no such option is given) is all columns selected.

 `-#` Select column number # (integer from 1). This option can repeat and can be combined with `NAME`.

`NAME` Select column of given name.

`-o#[,#...]` To be used with option `-p`: merge given columns to One plot. Several `-o`'s can be repeated. The default (if no option `-o` is given) is `-o2,6,7`, i.e., T, Tin, and Ttr are merged in one plot. If at least one `-o` is given, this default is overridden, further `-o`'s define variables for another plot. The first column must be also selected (by `-#` or `NAME`), the remaining are added. Columns in the cp-file (cpa-file) not listed are plotted separately. Example: to plot `512.cp`, blocked by 10, with columns 4 and 5 merged, 2, 6, 7 merged, and all other columns separate, use:

```
showcp -b10 -p 512.cp -o4,5 -o2,6,7
```

Range/lag options:

`-f#` `(1)` Analyze or show from record #

`-t#` `(0)` Analyze or show to record # (incl.). The default is `-t0` (= the last record)

`-b#` `(0)` Block (sub-average) size for showing and additional analysis. The default is `-b0`, which will adjust the block size to a power of 10 (for plotting) or according to screen width.

`-h#` Set the timestep (`noint*h` from `cook` run), applies to `-x,-c1,-c2`.

`-l#` `(29)` Set lag for statistics (up to which autocorrelation analysis is performed). `-l0` = no statistics

`-n#` `(12)` Number of consecutive blockings by 2 (not for `-c1,2`). Blocked data enable better error estimate.

Actions:

`-a` Write ASCII file `SIMNAME.cpa` in format compatible with `gnuplot` or `plot` (see Sect. 18.2)

`-a-1` As above and add column `Etot-Etot0`

`-` Output is `stdout` instead of a cpa-file and any other information but the resulting ASCII convergence profile is prepended by `#`. To be used with option `-a` to pipe the results to other programs. Example:

```
showcp - -a test.cp | plot -:0:4 :5
```

will show a merged graph of columns 4 and 5.

`-c#` `(0)` Sum of:

    `1`   write files SIMNAME.NAME.tcf with time correlation functions

    `2`   write files SIMNAME.NAME.cov with covariances

    `4`   detailed autocorrelation analysis (not just summary)

    `8`   autocorrelation analysis of data blocked by option -b

16 autocorrelation analysis with removed linear T-dependence

Option `-l` must be set, too. The default is `-c0` which means that only a brief summary of errors is printed.

`-g` print pseudoGraphs [default if columns selected and no other action]

`-m` print Merged y-x pseudograph (1 line = 1 block, variables are shown by letters)

`-p` Plot blocked selected columns with not-constant data (forces -a)

`-x` write blocked x-y files SIMNAME.NAME.xy

`-e` statistical analysis and regression of the first two selected columns

Options:

`-k#` pseudograph info line (blow the graph) contains:

1   column info, min, max, range, blocking
2   column info, first, last, difference, blocking
3   both 1+2

`-d# (1)` Set delay # s between plots, to be used with option `-p`. Increase when `showcp` is run on a distant computer and the connection is slow

`-r` Reverse endian (obsolete)

`-s#` Scroll (buffer #kB, `-s=-s30`) (obsolete)

`-u` Character set for pseudograph (option -g) is

`-u` Braille UTF-8
`-u-` ASCII

The default is guessed from LOCALE.

Example:

```
showcp mysimul -p Etot Tkin -5
```

Will show three graphs, total energy (column 1), temperatures (white=Tkin=total, yellow=translational, cyan=rotational and intramolecular), and column 5=pressure or density. The data will be averaged in appropriately selected blocks

*Source: `util/showcp.c`

20.2   cp2cp: Manipulate convergence profile files

Usage:

```
cp2cp [OPTIONS] INPUT.cp OUTPUT.cp [OPTIONS]
```

Files:

INPUT.cp input convergence profile file

OUTPUT.cp output convergence profile file

Options:

-b# input data are blocked (sub-averages are made) by # lines of data

-s# stride by #: every #-th value of input is taken, the first one considered is the #-th

-s# -b## strided data are blocked

-rKEY Rearrange columns. KEY=string of {1,2,..,9,A,..,0}, where 1=1st column, A=10th column, 0=empty column (filled by 0)

-hNAMENAME.. headers of empty columns (added by using key 0 in -k), every 4 chars correspond to one 0

-f# read the input file from record # (records are numbered from 0)

-n# max. # of records read (must appear after -f)

-t# read the input to record # (record # is not included; equivalent to the sum of -f -n)

BUGS: does not accept old 5-column format, opposite endian, packed (.cpz) files

Examples:

Omit first 1000 records, delete columns 6+7 (of 12), and block by 10:

```
cp2cp in.cp out.cp -f1000 -r1234589ABC -b10
```

Replace column 4 by 0 (header ZERO) and swap columns 6 and 7:

```
cp2cp in.cp out.cp -r1230576 -hZERO
```

Split a.cp (100000 records long) into chunks by 10000 records

```
tab 10000 100000 10000 "cp2cp a.cp b0.cp -f0 -n10000" | sh
```

Source: util/cp2cp.c

20.3   `rdfg`: Analyze and show radial distribution functions

`rdfg` prints and/or plots the radial distribution functions from binary data stored in the rdf-file. The running coordination numbers are provided, too (as column 4 of the g-files).

Usage:

```
rdfg SIMNAME[.rdf] [KEY [FMT]]
```

`KEY` is a string of:

- u   Unix version only: Create separate files `SIMNAME.SITE1.SITE2.g` for each site–site pair in the rdf-file

- d   DOS style: Create separate files `SITE1SITE2.g` for each site–site pair in the rdf-file

- none of d u Create one merged file `SIMNAME.g` of all site–site RDFs

- p   Start plot of the of generated g-file(s). See Sect. 18.2.

- r   Reverse endian on input. To be used for rdf-files obtained on a computer with the opposite sex (endian)

`FMT` is the C-format for printing g(r), default="%8.4f' (four dec. digits).

Example (generate files with DOS-like names, accuracy=5 dec.digits, plot them):

```
rdfg mysim dp %9.5f
```

*Source: `util/rdfg.c`

20.4   `smoothg`: Smooth histogram-based RDF

Smoothing of g(r) [obtained by histogram]. Call by:

```
smoothg WINDOW ORDER DR DATA[:COLR:COLG] FROMR[:TOR[:DR]] [FORMAT]
```

`WINDOW`   `RECT[+TRIANG]` rectangular width+triangular half-width
       `-WIDTH` Gaussian half-width: $\exp[-(x/WIDTH)^2/2]$

`ORDER` order of the polynomial (1=linear)

`DR` grid of R in DATAFILE (must match file!)

`DATA` file of r,g data

`COLR:COLG` columns of r,g (default=1,2), COLR=0 means 0,1,..

`FROMR:TOR:DR` output range and grid of R

`FORMAT` output format for double x,y (default="%.12g %.12g")

*Source: `util/smoothg.c`

20.5   `staprt`: **Print a sta-file**

Prints the contents of a sta-file generated e.g. by `cook`. Usage:

```
staprt FILE.sta [mode]
```

where `MODE` is

+   More decimal digits of the output

-    Print merged time correlation to stdout

`anything else` Print time correlation files to files `FILE.NAME.tcf`, where `NAME` is derived
    from the variable name (problematic characters are edited)

*Source: `util/staprt.c`

20.6   `sfourier`: **Structure factor from RDF**

Using the fast Fourier transform, RDF (from a g-file) is converted to a structure factor (SF).
Cf. structure factors obtained directly from configurations (see cook, option `-f`).

To get a partial SF from site–site RDF:

```
sfourier GRID CUTOFF NS < FILE.X.X.g > FILE.gsf
```

where `FILE.X.X.g` is one of g-files obtained by `rdfg FILE u`

The total SF for a mixture from a set of RDFs

```
sfourier GRID CUTOFF NS SITE1:b1 SITE2:b2 ... < FILE.g > FILE.gsf
```

where `FILE.g` is obtained by `rdfg FILE` (or `cat FILE.*.*.g`) Note that `sfourier` reads also
information printed in the header of the g-files.

`GRID` Number of grid points/Å, must match the grid of the RDF files

`CUTOFF` Should be `rdf.cutoff` (see cook input data). If CUTOFF<`rdf.cutoff`, then then the
    data are truncated; if CUTOFF>`rdf.cutoff` then the data are padded by $g(r) = 1$.

`NS`   total number of sites (atoms, not types of sites)

`SITE1, SITE2` name of site of given number. Sites may be listed in any order

`b#`   scattering length of site #

*Source: `util/sfourier.c`

20.7   `coordn`: Coordination number

Calculates the coordination number from a (r g(r)) file. Normally not needed because `rdfg` prints the coordination number as well. Call by:

```
coordn COLUMN_OF_G DR [RHO G0 CONST] < INFILE > OUTFILE
```

Calculates CONST + RHO * integral_0^R [g(r)-G0] dV. Default RHO=1, CONST=0. The integral is over dV = 4 PI r^2 dr and is replaced by the sum over dV = 4 PI/3*[(r+DR/2)^3-(r-DR/2)^3

Source: `util/coordn.c`

20.8   `hbonds`: H-bonds for liquid water

Hydrogen bonds are generated for water based on intermolecular O-H distance. The default version is for TIP4P, this can be changed in the source file.

Call by:

```
hbonds PLBFILE [O-H distance]
```

where PLBFILE is the input playback file and the second parameter is the threshold O-H distance to define an H-bond; the default is 2.44. A series of files hb####.mol, hb####.gol, hb####.plb is generated where #### are consecutive frames.

Source: `util/hbonds.c`

20.9   `cppak`: Loss (de)compression of convergence profile files

This compressor samples the min–max interval of variables using integers of given number of bits and stores the differences. Typical compression rates are to 25–50%, depending on the number of bits and possible columns of zeros only.

Call by:

```
cppak FILE.EXT [NBITS]
```

where the action is determined by the extension .EXT:

`.cp` compresses FILE.cp to FILE.cpz

`.cpz` decompresses FILE.plz to FILE.cp

NBIT is the number of bits to store the min–max interval; default=12 (accuracy 3-4 dec. digits).

Packed cp-files are transparently recorded by `cook` (see variable `CPnbit`) and shown or analyzed by `showcp` (see Sect. 20.1).

Source: `util/cppak`

20.10   `autocorr`: Statistical analysis using autocorrelation function

Prints autocorrelation and error analysis of time-dependent data. Call by:

```
autocorr {FILE|-} [OPTIONS]
```

Options:

- `-l#` The lag (# of autocorrelation coefficients) for statistics [default=29]

- `-b#` The lag for calculations with data blocked by 2, 4, 8, ... items [default=2]

- `-n#` Number of blocked calculations: the maximum block size is 2^# [default=8]

- `-h` Higher precision (more digits) on output [default=off]

- `-m-1` Input data are angles in radians (mod 2PI) and they are reconstructed to be continuous. To be used for the time correlation function of, e.g., dihedral angles.

- `-m#` As above, modulo # (must be positive integer). E.g., for angles in degrees use -m360 (as for output of `ramachan`)

- `-c#` Column of data [default=1]

- `-t` Write time correlation functions (column 1 = 0,1,2,...; column 2 = autocorrelation coefficients) in separate files `NAME#.tcf` Number # is used only if given by option `-c#`. [default=off]

- `-` Instead of `FILE` means stdin

The output contains first the name, number of data and their range. In the second line there is the Mean and Variance. Then, the autocorrelation coefficients and other statistics are printed:

```
c[t] = Cov(c[0],c[t])/Var(c)
tau = SUM_t'=1^t c[t']  (partial sum)
StDev^2 = <(1+2tau) Var(c)>/n
```

where `Cov(X,Y)=<(X-<X>).(Y-<Y>)>` is the covariance and `Var(X)=Cov(X,X)` is the variance; both are estimated from the data. tau is the correlation length and StDev is estimated standard deviation of the mean (arithmetic average).

*-lines contain the same information calculated from blocks of averages (sub-averages) by 2, 4, 8... data. Example (realistic):

```
> autocorr logfile.anc -c3 -n12
logfile.anc.3         No=320688  range=[-1.40727e-09,2.12488e-09]=3.53215e-09
  Mean = 1.30584217e-10    Var = 5.16207293e-20
  _t__c[t]__1+2tau__StDev_  _t__c[t]__1+2tau__StDev_  _t__c[t]__1+2tau__StDev_
    0 1.0000  1.000 4.0e-13   1 0.9960  2.992 6.9e-13   2 0.9844  4.961 8.9e-13
    3 0.9667  6.894 1.1e-12   4 0.9450  8.784 1.2e-12   5 0.9214 10.627 1.3e-12
    6 0.8978 12.423 1.4e-12   7 0.8754 14.173 1.5e-12   8 0.8552 15.884 1.6e-12
```

```
    9 0.8374 17.559 1.7e-12   10 0.8220 19.202 1.8e-12   11 0.8084 20.819 1.8e-12
   12 0.7958 22.411 1.9e-12   13 0.7834 23.978 2.0e-12   14 0.7702 25.518 2.0e-12
   15 0.7556 27.029 2.1e-12   16 0.7389 28.507 2.1e-12   17 0.7201 29.947 2.2e-12
   18 0.6994 31.346 2.2e-12   19 0.6771 32.700 2.3e-12   20 0.6539 34.008 2.3e-12
   21 0.6304 35.269 2.4e-12   22 0.6074 36.484 2.4e-12   23 0.5853 37.654 2.5e-12
   24 0.5645 38.783 2.5e-12   25 0.5452 39.874 2.5e-12   26 0.5274 40.928 2.6e-12
   27 0.5110 41.950 2.6e-12   28 0.4957 42.942 2.6e-12   29 0.4812 43.904 2.7e-12
 *  0 1.0000  1.000 5.7e-13    1 0.9848  2.970 9.8e-13    2 0.9465  4.863 1.2e-12
 *  0 1.0000  1.000 8.0e-13    1 0.9516  2.903 1.4e-12    2 0.8663  4.636 1.7e-12
 *  0 1.0000  1.000 1.1e-12    1 0.8936  2.787 1.9e-12    2 0.7559  4.299 2.3e-12
 *  0 1.0000  1.000 1.5e-12    1 0.7907  2.581 2.5e-12    2 0.4804  3.542 2.9e-12
 *  0 1.0000  1.000 2.1e-12    1 0.5645  2.129 3.0e-12    2 0.1880  2.505 3.2e-12
 *  0 1.0000  1.000 2.6e-12    1 0.3299  1.660 3.3e-12    2 0.0953  1.850 3.5e-12
 *  0 1.0000  1.000 2.9e-12    1 0.2409  1.482 3.6e-12    2 0.0838  1.649 3.8e-12
 *  0 1.0000  1.000 3.3e-12    1 0.1694  1.339 3.8e-12    2 0.0917  1.522 4.1e-12
 *  0 1.0000  1.000 3.6e-12    1 0.1880  1.376 4.2e-12    2 0.1144  1.605 4.5e-12
 *  0 1.0000  1.000 3.9e-12    1 0.2105  1.421 4.7e-12    2 0.0248  1.471 4.8e-12
 *  0 1.0000  1.000 4.2e-12    1 0.1515  1.303 4.8e-12    2-0.0060  1.291 4.7e-12
 *  0 1.0000  1.000 4.4e-12    1 0.0376  1.075 4.5e-12    2 0.0331  1.141 4.7e-12
 # 1.30584217191e-10 4.76e-12 9.53e-12 320688 (av std 2*std no)
```

Analysis: These are highly correlated data. The average is 1.3058e-10. The autocorrelation coefficient with lag of 29 timesteps is 0.4812 so that the error estimate of 2.7e-12 is severely underestimated. In lines beginning with * are error estimated with blocking 2,4,8,... The higher blocking, the less influence of correlations, but fewer data so that less reliable estimate. The best results are from the t=1 column, i.e., in addition to blocking the first autocorrelation coefficient is taken into account. Watch the $c[1]$ column and find the line with $c[1]$ small enough, let us say, $c[1] < 0.2$, or $c[2] < 0.1$. It gives stdev=4.8e-12. The last #-line contains the maximum which is usually reasonable error estimate obtained automatically. Therefore the result is 1.306e-10 +- 4.8e-12 (or perhaps more optimistic 1.306e-10 +- 3.6e-12; it is difficult to say which value is more reliable. This is general problem in simulations with slowly decaying correlations.), i.e., with probability about 68% the value is within 1.306e-10 +- 4.8e-12, with probability 95% within twice this margin.

If there are enough data and the error estimates are reliable, then

Note: the autocorrelation analysis is embedded also in `cook` and `showcp`

Bug: some non-alphanumeric characters are removed from FILE.

*Source: `util/autocorr.c`

20.11   `spectrum`: Spectrum (Fourier transform)

Calculates a spectrum (Fourier transform) of data. Call by:

```
spectrum [-]NDATA [KDATA [DT]] < FILE
```

FILE should contain NDATA numbers (SPACE, tab, or EOL separated) and KDATA is the number of output frequencies (if not given, KDATA=NDATA/2. If time resolution DT is given, the output has two columns (1st column = frequency, 2nd column = squared amplitude) and peaks are also calculated. If DT is omitted, the output has only one column (squared amplitude). With

-NDATA, the input data will be multiplied by bell-like window $\sin^2(\pi * i/NDATA)$. The peaks are then 'more peaky' (delta-function like), but the resolution actually somehow worsens.

May be used for determining vibrational frequencies from MD data. For instance, let `sim.cp` be a 1000-records-long convergence profile obtained by running `cookfree` at low temperature for one molecule with `h=0.001` and `noint=1`. Then the following 'pipe':

```
showcp - -a sim | \
  mergetab -:2 | \
  spectrum 1000 2000 0.002 | \
  plot -
```

will plot a graph with peaks corresponding to vibrational frequencies. These should match the results of `blend -N sim`. Note that parameter `DT` has been set to TWICE the sampling rate `h*noint` because the kinetic temperature is a quadratic quantity and gives doubled frequencies.

Note: the algorithm uses FFT optimized for factors 2 and 3. The calculation is slowest for `NDATA`=prime number.

*Source: `util/spectrum.c`

Chapter 21

Working with playback files

WARNING: This may be incomplete, use `plbinfo +u` to get a rich list of plb-utilities (not all them in MACSIMUS).

The calculated configurations or trajectories are usually stored in files with extensions `.plb` (whole configuration) or `.p00` (one molecule). In some cases the can be analyzed again by `blend` or `cook` that have generated them. Here are utilities to manipulate with the plb files.

21.1   Format of plb-files

The plb-file is a binary file consisting of 4-byte floats:

header 2 floats The header consists of 2 float (4 byte) numbers:
> **The first float** = the number of sites (often denoted `ns`)
> NB: in the float format; the maximum `ns=16777216`
>
> **The second float** =
>
> > 0e0 Free (3D Cartesian, vacuum) boundary conditions
> >
> > L   with $L > 0$: periodic b.c., the box is a cube and L is constant
> >
> > -3e0 variable box size, all atom positions are positive
> >
> > -6e0 NOT FULLY IMPLEMENTED, see `cook -n4`: half box size subtracted from all coordinates, otherwise the same as -3e0

frame 1 One frame is:
> > L[] 12 bytes For variable box size format only: the actual box size
> >
> > r[0][] 12 bytes Position of the first atom (site) in the configuration
> > > ...
> >
> > r[ns-1][] 12 bytes Position of the last atom (site) in the configuration
> > > ...

frame 2 In the same format
> ...

last frame In the same format

21.2 `plbinfo`: Get information on plb-files

Get information on binary playback files. Examples:

```
plbinfo FILE FILE... # print info
plbinfo +u           # print list of plb utilities
plbinfo +t           # print technical info on plb files
plbinfo              # print help
```

Returns # of errors encountered (0 on success)

Source: `util/plbinfo.c`

21.3 `plbconv`: Converts old and new plb formats

The format of plb files changed from a version supporting only fixed box size to a version with variable 3D box. `plbconv` converts these two formats.

Call without arguments to get help.

Source: `util/plbconv.c`

21.4 `plb2plb`: Extract selected sites

Call without arguments to get help.

Source: `util/plb2plb.c`

21.5 `plbsites`: ASCII export of selected sites

Call without arguments to get help.

Source: `util/plbsites.c`

21.6 `plb2asc`: Conversion of plb-files to ASCII

Call without arguments to get help.

Source: `util/plb2asc.c`

21.7 `asc2plb`: plb-files from ASCII

Call without arguments to get help.

Source: `util/asc2plb.c`

21.8 `plbframe`: Extract/interpolate a frame

Call by:

```
plbframe {INPUT.plb|-} {OUTPUT.plb|-} [FRAME]
```

Source: `util/plbframe.c`

21.9 `plbcut`: extracts parts of a playback file

Call without arguments to get help.

Source: `util/plbcut.c`

21.10 `plbbox`: Change box size of a plb-file

Call without arguments to get help.

Source: `util/plbbox.c`

21.11 `densprof`: Selected density profile angular correlations.

Call without arguments to get help.

Source: `util/densprof.c`

21.12 `plb2cryst`: Sort sites to files according to crystal-like structure.

Call without arguments to get help.

Source: `util/plb2cryst.c`

21.13 `plb2nbr`: Sort sites to files according to the number of neighbors.

Call without arguments to get help.

Source: `util/plb2nbr.c`

21.14   `plbmerge`: Merge several plb files

Merge several plb files into another plb file, frames synchronously. Call without arguments to get help.

Source: `util/plbmerge.c`

21.15   `atomdist`: Atom-atom distances

Prints atom-atom distances from playback files Call by:

```
atomdist FILE INDEX1 INDEX2 [NS]
```

`FILE` The playback file (with extension `.plb, p00, ...`) or a 3db-file

`INDEX1 INDEX2` Atom indices (see the mol-file)

`NS` Number of sites (iff `FILE` a 3db file)

Source: `util/atomdist.c`

21.16   `plbmsd`: Mean square displacement of atoms

Calculates the (mean) square displacement of selected sites. Call by:

```
plbmsd [-]FILE[:FROM] L SITE [SITE ...]
```

where

`FILE.p00 or FILE.plb` input playback file

`FROM` start for linear regression, default=2=first datum (1=zero)

`FILE.msd` output file

`-FILE` reverse endian of input playback file

`L`   box size (L=0: take from FILE.EXT, $L < 0$: force FREE boundary conditions)

`SITE` site # to process (can enter more sites)

*Source: `util/plbmsd.c`

21.17 `density`: Calculate local density

Call by:

```
density FILE.plb[:FROMFRAME] X Y Z R
```

Calculates # of atoms in a sphere of center (X,Y,Z) and radius R. Reads the whole file starting with frame `FROMFRAME` (default=1).

Source: `util/density.c`

21.18 `mergeplb`: Merge several plb files into one

Merge several plb files into another plb file, frames synchronously. Stops when any plb-file reaches EOF.

```
mergeplb PLB-FILE [PLB-FILE ...] > MERGED-PLB-FILE
```

The box size L is set to the maximum L Hint: use molcfg to prepare mol (and gol) files for showing.

Source: `util/mergeplb.c`

21.19 `plbpak`: Loss (de)compression of playback files

This compressor rounds the coordinates to the nearest grid point and stores differences in coordinates. Typical compression rates are to 25–30% for resolution 10/Å and 30–40% for resolution 100/Å. Best lossless compressors (bzip2, rar) give about 90%.

Examples:

```
plbpak sim.plb 64 # compress, resolution 64/AA
plbpak sim.plz    # decompress
```

*Source: `util/plbpak.c`

21.20 `plb2diff`: Diffusion and conductivity

This utility helps calculate the diffusion, conductivity (both partial and bulk) from plb files. `plb2diff` calls cook* several times for different blocks of configurations and makes averages. Run `plb2diff` without argument to get help.

*Source: `util/plb2diff.c`

21.21   shownear: re-color atoms according to their distance

```
shownear FILE.plb[:FRAME] FILE.gol \
         FROM TO [-]DIST COLORNEAR [COLORMARK] > OUTFILE.gol
```

Creates a gol-file that, when used with show, will mark atoms closer than —DIST— to atoms [FROM..TO) by COLORNEAR and optionally atoms [FROM..TO) by COLORMARK. -DIST means that atoms [FROM..TO) are not included in tests. Configuration (frame) FRAME is analyzed from file FILE.plb. TO<=0 means TO=number of sites.

Source: util/shownear.c

Chapter 22

Molecule visualization

WARNING: This is obsolete, run the respective utilities without any argument for up-to-date help.

22.1 `molcfg`: Create configuration mol- and gol-files

This is a supporting utility to be used with `show` to show a configuration recorded by `cook*`. It generates a configuration mol-file `SIMNAME.mol` and optionally a gol-file `SIMNAME.gol` from mol-files `FILE1.mol`, `FILE2.mol` ... and optionally gol-files `FILE1.gol`, `FILE2.gol` ...

Usage 0:

```
blend -o SYSNAME MOL1 MOL2
...
cook* SYSNAME SIMNAME
```

In the above "standard" way to call cook with a force field generated by blend, `molcfg` is called transparently from `cook*`. Problems may occur with "optimized water models" recognized by cook, which may change the name. Then see below

Usage 1:

```
molcfg FILE1 FILE2 ... SIMNAME
```

Repeats `FILE1.mol` and `FILE1.gol` (if exists) `N[0]` times, `FILE1.mol` `N[1]` times, etc., where `N[0]`, `N[1]`, etc., are read from `SIMNAME.def` used in the simulation. Note: only variable `init`, `no`, `noint` are accepted as temporary variables in `SIMNAME.def` if you wish to use formulas to calculate `N[]`: `init=20 N[0]=init`^{3} is correct, `eps=20 N[0]=eps`^{3} is incorrect.

Example 1:

```
molcfg Li Al Cl I salt
show salt
```

Usage 2:

```
molcfg -COUNT1[PREFIX1:SUFFIX1:FROM1] FILE1 \
   [-COUNT2][PREFIX2:SUFFIX2:FROM2] FILE2 ... SIMNAME
```

Repeats `FILE1.mol` COUNT1 times, etc.; `SIMNAME.def` is ignored. `PREFIX` and `SUFFIX` are added to atom ID and may contain format (as %d) to hold file number, starting from `FROM` (default=1)

Example 2:

```
molcfg -1 cyto -10w hoh -3:p%d proton -1 etoh config
show config -Yp1 -Yp2 -Yp3
```

*Source: `util/molcfg.c`

22.2 `show` V2.2j: Viewing playback (trajectory) files

`show` shows playback files (trajectories), `.plb`. In addition, description of molecules (mol-file and optional gol-file) is needed; details are explained in the manual of `blend`. See also `molcfg` (see Sect. 22.1) and other utilities working with the playback files (see Sect. 21).

Run it without parameters to get list of options. Letter 's' in shell variable GUI indicates start with menu.

Usage:

```
show [OPTIONS] MOLNAME[.mol|.gol|.plb] [ PLBNAME.EXT ]
```

22.3 `ray`: The raytracer

A Reasonably Intelligent Raytracer by Mark VandeWettering, modified by J. Kolafa. The recommended raytracer for MACSIMUS. I found it more suitable for rendering molecules than PovRay (unless you want to see wooden atoms in fog..) because (1) it is simple, (2) can better handle the ambient light, and (3) uses the normal right-handed coordinate system. Nevertheless, if you prefer PovRay, you will find info on how to use it in `show/show.c`.

The scene file for this raytracer has extension `.nff` (for Neutral File Format) and can be generated by `show`. The output file is PPM (see Sect. 22.2).

To get help on options, run

```
ray -h
```

Simple example (render `sim-0000.nff` dumped by `show`):

```
ray -n sim-0000
```

Another example (as above, 1/2 size, no antilaliasing (faster), watch progress, view the final picture by 'xv')

```
ray -n sim-0000 -S.5 -j1 -t -vxv
```

Usage details:

```
ray { -Option Argument | -OptionArgument } ...
```

`-h` Help

`-i FILE` Input scene file (Neutral File Format), recommended extension `.nff`

`-o FILE` Output file (P6 Portable Pixel Map), recommended extension `.ppm`

`-n FILE` Input=`FILE.nff`, output=`FILE.ppm`

`-t` Show progress indicator in %

`-u` Cheap and fast antialiasing by 2x2 blur (formerly -f)

`-v VIEW` Show picture using viewer `VIEW`. For linux, ImageMagick viewer `display` is recommended.

`-j #` Antialiasing and jittering. The default is `-j-9`

> `#=0` No antialiasing nor jittering (fastest)
>
> `#>0, #!=N^2` Random jittering with # samples/pixel
>
> `#>0, #=N^2` Antialiasing by N*N supersamples/pixel in a square
>
> `#<0, #=-N^2` As above but supersample only if contrast > `c`.
>
> `#<0, #=-N^2 -d#` With option `-d#`: if error > `c`, enlarge supersample N →2N+1 (N even) or N →3N (N odd)

`-d #` Diffuse light. # is the light size for Gaussian jittering (for each light source). Use `-j` with a large argument (at least 100). Will create soft shadows.

`-c #` Threshold for smart supersampling (`-j-#`), default=0.02

`-x #` x size in pixels, default=command 'resolution' in NFF file

`-y #` y size in pixels, default=command 'resolution' in NFF file

`-r #` The same as `-x# -y#`

`-s #` Scale view angle (zoom) by # [default=1]. Use # < 1 if you do not see the whole molecule, # > 1 if you wish to look closer

`-S #` Scale x and y size by # [default=1]. Will create smaller image but does not change proportions.

`-a #` Pixel aspect ratio (y/x) [default=1]

`-f #` Fog from z-coordinate (no fog in front of this z) [0]

`-F #` Fog thickness for exponential attenuation to 1/e (to background color), 0=off [0] Note: the fog algorithm is simple and works OK only with uniform background, best light cyan or blue

-b FILE Use given background image (must be P6 PPM file), instead of color (command b in the NFF file)

-X # Scale background image # times horizontally [1]

-Y # Scale background image # times vertically [1]

-U # Move background image up by #*height [0]

-R # Move background image right by #*width [0]

-B # How the background image is treated [default=+2]

   #>0: tile background image

   #<0: chessboard mirror tile (make nonperiodic images continuous)

   +1 -1 Use the color of command b in the NFF file front background (this is not directly visible, but may reflect in the rendered spheres)

   +2 -2 Calculate background color for front as background image average [default]

   -l # Light scaling factor [1] (effective brightness adjusted to -I -N). Use # > 1 if the image is too dark and vice versa.

-I # Ambient isotropic light [0.1]. This is the light shining on the scene from all sides isotropically and reflecting isotropically.

-N # Ambient light proportional to cos angle(normal,ray) [0.2]. This is the light shining on the scene from all sides isotropically but reflecting more to the normal. Both -I -N improve the appearance of spheres.

NFF: Detailed explanation is in ray/NFF.desc. Note that ray uses the normal right-handed coordinate system.

Comment

v   1st command to start scene description

from X Y Z The viewpoint (eye)

at X Y Z Look at point (center of the 'screen' or 'paper')

up X Y Z Direction up vector on the 'screen' or 'paper'

angle ANGLE Viewing angle of the width of the

'screen' or 'paper', in degrees

hither 1 ?

resolution 533 400 Size of the image in pixels

l X Y Z Position of a point light source. There may be several light sources. All have the same intensity

b R G B Background, in RGB (Red Green Blue in interval [0,1]).

f R G B diffuse specular Phong transmittance index Set the material. `diffuse` is the amount in [0,1] of the light reflected diffusely, `specular` is the amount in [0,1] of the light reflected in the same angle (this makes the 'mirror' effect), Phong is the power determining the size and brightness of the highlights. For molecules, the `transmittance` and the `index` of refraction will never be used.

s X Y Z RADIUS A sphere

c X1 Y1 Z1 RADIUS1 X2 Y2 Z2 RADIUS2 A cone of the axis defined by (X1,Y1,Z1)-(X2,Y2,Z2). If both radii are the same, this is a cylinder.

p # Polygon of # vertices. # lines of **X Y Z** should follow.

pp # Polygonal patch primitive (one-sided). # lines of **X Y Z Xnormal Ynormal Znormal** should follow. Not used by `show`

*Source: `ray/main.c`

22.4   `ppm2ps`: PPM, PBM, PGM to PostScript

Both `ray` and `show` generate pictures in the PPM format. Since I was not satisfied with the quality of printing these files using available utilities in 90s, I wrote my own. All six versions (P1–P6) are supported.

Call by:

```
ppm2ps [OPTIONS] [INFILE [OUTFILE]] [OPTIONS]
```

Source: `ray/ppm2ps.c`

22.5   `ppminfo`: Get information on ppm, pbm, pgm-files

Call by (verbose info):

```
ppminfo FILE
```

Call by (brief info):

```
ppminfo FILE FILE [FILE ...]
```

Source: `ray/ppminfo.c`

22.6   `stereo`: Stereogram

This program calculates stereograms normally with the eye cross point behind the paper, i.e., your eyes should watch an imaginary point lying behind the paper in approximately the same distance as the eye-paper distance (this is for the recommended default of the x-pattern size one

half of the eye's distance). Different people may prefer different distances or even stereograms with the cross point above the paper.

Two input files are needed:

1. The ZB-file with a z-buffer data. It is a PGM (raw Portable GrayMap, P5). It can be generated by `show`.

2. Optional pattern file. Its x-size should correspond to about 5 cm. If not given, a random pattern is generated.

Run without options to get more info.

*Source: `show/stereo.c`, upgraded 2016

Chapter 23

Miscellaneous utilities

WARNING: This is obsolete, run the respective utilities without any argument for up-to-date help.

23.1 `pdb2pdb`: Rearrange pdb-files

Rearranges a pdb-file so that the order of the backbone atoms is N-CA-C-O[-sidechain]. This is 'standard', though most programs (incl. `pdb`) do not care. Call by:

```
  pdb2pdb [MAXLINES] < INPUTPDB > OUTPUTPDB
```

Source: `blend/pdb2pdb.c`

23.2 `ramachan`: Ramachandran plot from blend and playback files

Call by:

```
  ramachan [OPTIONS] SYSNAME [SIMNAME.plb|SIMNAME.p00] [OPTIONS]
```

Options:

  `-a` Angles in interval [0,360) [default=[-180,180]]

  `-r` Write results to separate files `SIMNAME.r#`, where #=frame number. [default=all frames concatenated to one file `SYSNAME.ram`]

  `-s` Write summary phi,psi file (1line=1frame; for columns see SYSNAME.mar) This file can be used, e.g, by `autocorr` to obtain time correlation functions. Example:

```
      autocorr -t -c2 -m360 test.sum
```

  where see `test.sum` which dihedral is column 2 (of `-c2`) In addition, this file can be used as SIMNAME.ddf for the DIHHIST=-1 version of cook as selection of phi,psi angles.

-f# From frame (first frame to process) [default=1]

-t# To frame (last frame processed) [default=-1=until eof]

-b# By frame (stride) [default=1=every frame]

-p# The calculated Ramachandran plot is plotted (using plot, see Sect. 18.2)).

-n# Number of molecules. It should be the same as N[0] specified in the simulation def-file. [default=1] **BUG:** Only simulations of identical molecules are supported (N[1]=N[2]=..=0 in the simulation def-file). WARNING: if no option -n# is specified, only the 1st molecule from the plb-file is processed. Similarly if -n# is less than the actual number of molecules (N[0]).

-PPARSET Parameter set (prepends environment variable BLENDPATH) The default parameter set is that defined in file SYSNAME.ble.

Files:

SYSNAME.ble Input ble-file

SIMNAME.plb,SIMNAME.p00 Input playback file

SYSNAME.mar Output dihedral angle info extracted from SYSNAME.ble

SYSNAME.ram Output Ramachandran plot(s) omega,phi,psi,omega: The first table from SYSNAME.ble with site info, tables calculated from playback files follow (if not option -r).

SYSNAME.sum Output Ramachandran summary of phi,psi by columns, 1 line= 1 frame of plb-file

SIMNAME.r1,... Tables calculated from playback files (if -r)

${BLENDPATH}/PARSET.par Parameter file (if given by option -p). Only table 'backbone' is used from this file.

Undefined angles/sites are denoted 999/-1

Source: blend/ramachan.c

23.3   makepept: Makes a peptide in che-format

Makes a peptide of given residues in a che-format. The environment is the same as for pdb and blend.

Print one letter aminoacid codes:

```
makepept ANYPARM
```

Make a peptide:

```
makepept RSDDIR {rsd | RRR} [rsd | RRR ...]
```

RSDDIR Subdirectory of BLENDPATH with residues in che-format.

rsd Residue or terminus name in lowercase

RRR Chain of one-letter aminoacid codes (uppercase)

Example (make peptide Acetyl-ALA-PRO-THR-HIS(neutral)-Methyl)

```
makepept charmm22 ace APT hisn ct1 > pept.che
```

BUG: For termini, the output file has to be edited (ct1 above)

Recommended blend commands:

```
blend -e40 pept.che      # no pept.mol,pept.3db
blend -e40 -r2 pept.che # to try again (overwrite pept.mol,pept.3db)
```

Source: blend/makepept.c

23.4   blefilt: Blend-file filter.

Extracts some information 'hidden' in ble-files. Call by:

```
blefilt TABLE [COL [COL2 ...]] [TABLE2 ... ] < INPUT.ble > OUTPUT
```

where TABLE is one of { sites bonds angles dihedrals impropers aromatics } and optional list of columns follows. The default column is the column with the value of bond length or angle, or x y z for sites. The output is in the order of the blend-file. Examples:

```
blefilt angles dihedrals < cyto.ble > cyto.int
blefilt sites < cyto.ble > cyto.xyz
blefilt bonds 2 4 7 < cyto.ble > cyto.int
```

Source: blend/blefilt.c

23.5   bonds: Make (show-able) mol-file from coordinates

Call by:

```
bonds { FILE.3dt | FILE.atm | FILE.pdb } [DIST]
```

FILE.3dt Data are in 3 columns x,y,z. DIST is the bond threshold [default=1.5]

FILE.atm Data are in 4 columns, ATOM,x,y,z, where ATOMS is atom symbol. The bond threshold = r0*DIST, where r0 is maximum bond length for given atom-atom bond. Only atoms H C O N S P are explicitly considered, other atoms use certain general values that may be OK in many cases. [default DIST=1.2, values less than 1 do not have too much sense]

FILE.pdb As above, PDB format.

FILE.mol output mol-file

FILE.plb output plb-file

The output mol and plb files can be used by `show`, but are not suitable for blend.

Hint: combination of `bonds FILE.pdb; show FILE` can serve as a simple PDB viewer.

Source: `blend/bonds.c`

23.6 `cutprt`: Shorten a prt-file

Shortens prt-files (generated by `cook`) leaving the first header and the final statistics. Call by:

```
cutprt [-]FILE.prt [[-]FILE.prt ...]
```

FILE.prt rewrites file `FILE.prt` by the shortened one, the old one is renamed to `FILE.prt`

-FILE.prt output to stdout

Source: `util/cutprt.c`

23.7 `lattice`: Make a cubic lattice

Call by

```
lattice N L LATTICE
```

N   Number of vertices

L   Box size

LATTICE=1 Simple cubic lattice

LATTICE=2 Body centered lattice=1

LATTICE=3 Face centered lattice

Source: `c/lattice.c`

23.8 `ice`: Make ices and clathrates

Example

```
ice Ih 5y 3z 3x -sHOMH -T0
show Ih533HOMH.plb
```

Source: `c/ice.c`

23.9   `naclcryst`: Make rotated NaCl crystals

Example

```
naclcryst 3 4 5 -x1.5 -m600
show nacl-0000.plb
```

Source: `c/naclcryst.c`

Part IV

Appendixes

Chapter 24

Ewald summation

24.1 Point charges

To simplify notation, let us first define function

$$\mathrm{e}(y) = \frac{\mathrm{erfc}(y)}{y} = \frac{1}{y}\frac{2}{\sqrt{\pi}}\int_y^\infty \exp(-t^2)\mathrm{d}t. \tag{24.1}$$

The basic formula for the Ewald **electrostatic energy** in periodic boundary conditions surrounded by a dielectric continuum (dielectric constant ϵ_r') in infinity in SI units is

$$4\pi\epsilon_0 U = \sum_{j<l,\,r_{jl}<r_c} q_j q_l\,\alpha\mathrm{e}(\alpha r_{jl}) \tag{24.2}$$

$$+ \sum_{\vec{k}\neq\vec{0},\,\kappa<\kappa_c} \frac{\exp[-(\pi/\alpha)^2\kappa^2]}{2\pi V\kappa^2}\,Q(\vec{\kappa})^2 \tag{24.3}$$

$$+ \frac{2\pi}{2\epsilon_r'+1}\frac{M^2}{V} \tag{24.4}$$

$$- \frac{\alpha}{\sqrt{\pi}}\sum_j q_j^2 \tag{24.5}$$

$$\equiv 4\pi\epsilon_0(U_r + U_k + U_M + U_s) \tag{24.6}$$

where

$$Q(\vec{\kappa}) = \sum_j q_j \exp[2\pi i\vec{\kappa}\cdot\vec{r}_j], \tag{24.7}$$

$$\vec{\kappa} = \vec{k}/\vec{L} \equiv (k_x/L_x, k_y/L_y, k_z/L_z),\ \kappa = |\vec{\kappa}|, \tag{24.8}$$

$$\vec{M} = \sum_j q_j\vec{r}_j,\ M = |\vec{M}|, \tag{24.9}$$

and κ_c is the k-space cutoff. The first sum, (24.2) (the r-space or pair sum), assumes that the cutoff r_c is less than half the minimum box size; if not, a sum over periodic images should be added. α is the separation parameter. $\vec{M}$ is the dipole moment of the simulation cell. It is well-defined only for configurations of neutral molecules which are never split by a periodic cell boundary.

The implementation of $e(y)$ in MACSIMUS is based on the following two functions,

$$\text{eru}(x) = \alpha e(\alpha\sqrt{x}), \tag{24.10}$$

$$\text{erd}(x) = -\alpha^3 \frac{e'(\alpha\sqrt{x})}{\alpha\sqrt{x}}, \tag{24.11}$$

$$\text{erud}(x) = \text{eru}(x), \text{ byerd} = \text{erd}(x) \tag{24.12}$$

where $e'(x) = de(x)/dx$. Note that the term appearing in the r-space sum is $\alpha e(\alpha r_{lj}) = \text{eru}(r_{lj}^2)$, where r_{lj}^2 is directly available. These functions are implemented by splines, See Sect. 11.6.

The **forces** are

$$4\pi\epsilon_0 \vec{f}_j = -4\pi\epsilon_0 \frac{\partial U}{\partial \vec{r}_j}$$

$$= \sum_{l,\, r_{lj}<r_c} q_j q_l \, \alpha^3 \frac{e'(\alpha r_{lj})}{\alpha r_{lj}} \vec{r}_{lj} \qquad \text{NB: } \alpha^3 \frac{e'(\alpha r_{lj})}{\alpha r_{lj}} = \text{erd}(r_{lj}^2)$$

$$+ \quad q_j \sum_{\vec{k}\neq \vec{0},\, \kappa<\kappa_c} \frac{\exp[-(\pi/\alpha)^2\kappa^2]}{V\kappa^2} 2\vec{\kappa}\, \text{Im}\left\{ Q(k)^* \, \exp[2\pi i\vec{\kappa}\cdot\vec{r}_j]\right\}$$

$$+ \quad \frac{4\pi}{2\epsilon'_r + 1} q_j \frac{\vec{M}}{V}.$$

The **electrostatic potential** is

$$4\pi\epsilon_0 \Phi(r_j) = 4\pi\epsilon_0 \frac{\partial U}{\partial q_j} \tag{24.13}$$

$$= \sum_{l,\, r_{lj}<r_c} q_l \alpha e(\alpha r_{lj}) \tag{24.14}$$

$$+ \sum_{k\neq 0,\, \kappa<\kappa_c} \frac{\exp[-(\pi/\alpha)^2\kappa^2]}{\pi V\kappa^2} \text{Re}\left\{ Q(\kappa) \exp[2\pi i\vec{\kappa}\cdot\vec{r}_j]\right\} \tag{24.15}$$

$$+ \frac{4\pi}{2\epsilon'_r + 1} \frac{\vec{r}_j \cdot \vec{M}}{V} \tag{24.16}$$

$$- \frac{2\alpha}{\sqrt{\pi}} q_j. \tag{24.17}$$

Note that contributions of $\vec{k}$ and $-\vec{k}$ are the same in the k-space sums, which is (along with treating null components of $\vec{k}$ separately) used in the code.

For ready-to-code formulas including the pressure tensor, see [33]. Differences from our notation are:

- $h = 2\pi\vec{\kappa}$ (vector)

- the r-space parameter is called κ (here: α, while κ means k-space)

- dielectric response ϵ'_r

24.2   Gaussian charges

Gaussian charge i of width σ_i at $\vec{r}_i$ has the charge density

$$\rho_i(\vec{r}) = \frac{q_i}{(2\pi\sigma_i^2)^{3/2}} \exp\left[-\frac{(\vec{r} - \vec{r}_i)^2}{2\sigma_i^2}\right]. \tag{24.18}$$

The interaction energy of a pair i, j of these charges r apart is

$$4\pi\epsilon_0 u = \frac{q_i q_j}{r}\mathrm{erf}(\alpha r), \quad \alpha = 1/\sqrt{2\sigma_i^2 + 2\sigma_j^2} \tag{24.19}$$

Replacing point charges by Gaussian charge distribution[38] leads to more complicated formulas, and from the implementation point of view to a necessity to have a separate spline for r-space contributions for every charge-charge pair. In addition, hyperbolic splines cannot be used and have to be replaced by (slightly slower) cubic splines (#define SPLINE 3).

Let us again split the energy (and other quantities) into four parts

$$U = U_r + U_k + U_M + U_s. \tag{24.20}$$

24.2.1 r-space part

The r-space **energy** is

$$4\pi\epsilon_0 U_r = \sum_{j<l,\, r_{jl}<r_c} \frac{q_j q_l}{r_{jl}} \left[\underline{\mathrm{erf}(\alpha_{jl}^{\mathrm{g}} r_{jl})} - \mathrm{erf}(\alpha_j^{\mathrm{s}} r_{jl})/2 - \mathrm{erf}(\alpha_l^{\mathrm{s}} r_{jl})/2\right], \tag{24.21}$$

where

$$\alpha_{ij}^{\mathrm{g}} = 1/\sqrt{2\sigma_i^2 + 2\sigma_j^2}, \quad \alpha_i^{\mathrm{s}} = 1/\sqrt{2\sigma_i^2 + 1/\alpha^2}. \tag{24.22}$$

The corresponding **forces** are

$$4\pi\epsilon_0 \vec{f}_j = -4\pi\epsilon_0 \frac{\partial U}{\partial \vec{r}_j} = \sum_{l,\, r_{jl}<r_c} \frac{q_j q_l}{r_{jl}^2} \left\{ \frac{1}{r_{jl}} \left[\underline{\mathrm{erf}(\alpha_{jl}^{\mathrm{g}} r_{jl})} - \mathrm{erf}(\alpha_j^{\mathrm{s}} r_{jl})/2 - \mathrm{erf}(\alpha_l^{\mathrm{s}} r_{jl})/2\right] \right.$$

$$\left. + \frac{1}{\sqrt{\pi}} \left[\underline{2\alpha_{jl}^{\mathrm{g}} \mathrm{e}^{-\alpha_{jl}^{\mathrm{g}\,2} r_{jl}^2}} - \alpha_j^{\mathrm{s}} \mathrm{e}^{-\alpha_l^{\mathrm{s}\,2} r_{jl}^2} - \alpha_l^{\mathrm{s}} \mathrm{e}^{-\alpha_j^{\mathrm{s}\,2} r_{jl}^2}\right]\right\}.$$

Since the underlined terms above correspond to the interactins of the original Gaussian charges, by omitting them we get the Ewald corrections for 1–2 and 1–3 interactions; the extension to 1–4 forces is straightforward.

The r-space contribution to the **electrostatic potential** is

$$4\pi\epsilon_0 \Phi_r(r_j) = 4\pi\epsilon_0 \frac{\partial U_r}{\partial q_j} \tag{24.23}$$

$$= \sum_{l,\, r_{lj}<r_c} \frac{q_l}{r_{jl}} \left[\mathrm{erf}(\alpha_{jl}^{\mathrm{g}} r_{jl}) - \mathrm{erf}(\alpha_j^{\mathrm{s}} r_{jl})/2 - \mathrm{erf}(\alpha_l^{\mathrm{s}} r_{jl})/2\right]. \tag{24.24}$$

24.2.2 k-space part

To calculate the k-space contributions, we define

$$
\begin{aligned}
Q(\vec{\kappa}) &= \sum_j q_j \exp[2\pi i \vec{\kappa} \cdot \vec{r}_j], \\
T(\vec{\kappa}) &= \sum_j q_j \exp[2\pi i \vec{\kappa} \cdot \vec{r}_j] \exp[-2\pi^2 \sigma_j^2 \kappa^2], \\
S(\vec{\kappa}) &= \sum_j q_j \exp[2\pi i \vec{\kappa} \cdot \vec{r}_j] \exp[-2\pi^2 \sigma_j^2 \kappa^2] \sigma_j^2.
\end{aligned}
$$

The k-space **energy** is

$$
4\pi\epsilon_0 U_k = \frac{1}{2\pi V} \sum_{k\neq 0,\, \kappa < \kappa_c} p(\kappa^2)\Big[\mathrm{Re}Q(\vec{\kappa})\mathrm{Re}T(\vec{\kappa}) + \mathrm{Im}Q(\vec{\kappa})\mathrm{Im}T(\vec{\kappa})\Big], \tag{24.25}
$$

where

$$
p(x) = \frac{\exp(-\pi^2 x/\alpha^2)}{x}. \tag{24.26}
$$

The k-space **forces** are

$$
4\pi\epsilon_0 \vec{f}_j = \frac{q_j}{V} \sum_{k\neq 0,\, \kappa < \kappa_c} \vec{\kappa} p(\kappa^2) \Big[s_j \mathrm{Re}T(\vec{\kappa}) - c_j \mathrm{Im}T(\vec{\kappa}) + \exp(-2\pi^2 \kappa^2 \sigma_j^2)(s_j \mathrm{Re}Q(\vec{\kappa}) - c_j \mathrm{Im}Q(\vec{\kappa})) \Big],
$$

$$
\tag{24.27}
$$

where real numbers c_j, s_j are defined by

$$
c_j + i s_j = \exp(2\pi i \kappa \cdot \vec{r}_j). \tag{24.28}
$$

The k-space **virial** components (to calculate the pressure tensor) are

$$
\begin{aligned}
4\pi\epsilon_0 W_{ab} = \frac{1}{2\pi V}\Big[&-4\pi\epsilon_0 U_k \delta_{ab} \\
&+ 2\sum \frac{\vec{k}_a}{\vec{L}_a}\frac{\vec{k}_b}{\vec{L}_b} p(\kappa^2) \left(\frac{1}{\kappa^2} + \frac{\pi^2}{\alpha^2}\right) p(\kappa^2) \Big\{ [\mathrm{Re}Q(\vec{\kappa})\mathrm{Re}T(\vec{\kappa}) + \mathrm{Im}Q(\vec{\kappa})\mathrm{Im}T(\vec{\kappa})] \\
&+ (2\pi^2)[\mathrm{Re}Q(\vec{\kappa})\mathrm{Re}S(\vec{\kappa}) + \mathrm{Im}Q(\vec{\kappa})\mathrm{Im}S(\vec{\kappa})] \Big\} \Big].
\end{aligned}
$$

The k-space part of the **potential** at position $\vec{r}_j$ is

$$
\begin{aligned}
4\pi\epsilon_0 \Phi_k(r_j) &= 4\pi\epsilon_0 \frac{\partial U_k}{\partial q_j} \\
&= \frac{1}{2\pi V} \sum_{k\neq 0,\, \kappa < \kappa_c} p(\kappa^2) \\
&\quad \times \Big[c_j \mathrm{Re}T(\vec{\kappa}) + s_j \mathrm{Im}T(\vec{\kappa}) + \exp(-2\pi^2 \kappa^2 \sigma_j^2)(c_j \mathrm{Re}Q(\vec{\kappa}) + s_j \mathrm{Im}Q(\vec{\kappa})) \Big].
\end{aligned}
$$

24.2.3   Dipolar terms

are the same as for point charges. E.g., for $\Phi_M(r_j)$, see (24.16).

Figure 24.1: Verification of the pressure calculation for the system of 200 Li$^+$, 200 Na$^+$, 400 Cl$^-$, and 800 H$_2$O (Kiss–Baranyai model). P_{ref} denotes the standard pressure calculated from the pressure tensor, P_{dV} denotes the virtual volume change (molecule-based) value.

24.2.4 Self-energy

The self-energy is

$$U_s = -\frac{1}{\sqrt{\pi}} \sum_i \alpha_i^{\text{s}} q_i^2. \tag{24.29}$$

The "self-forces" are zero. The self-energy contribution to the potential is

$$\Phi_s(r_j) = -\frac{2}{\sqrt{\pi}} \alpha_j^{\text{s}} q_j. \tag{24.30}$$

24.2.5 Pressure test

A robust test of MACSIMUS Gaussian charges and pressure code is shown in Fig. 24.1. Note that the dV→ 0 limit can be (for the sake of this test) considered as exact value. It is seen that the most accurate trajectory (with 2 iterations of the SCF + ASPC($k = 2$), $\omega = 1$) gives pressure with precision about 40 kPa. The standard integration is 350 kPa off (opposite sign), leading to a small density error of $< 0.2\%$. Half-timestep-based value is of the same speed as the double-iterated value, but less accurate with respect to pressure (80 kPa, about 1/4 of the standard timestep result.)

The calculations used `el.grid=64` and the default Ewald accuracy. Changes to `el.grid=32` or `el.grid=128` or higher Ewald accuracy did not change the results for given trajectory lengths.

Figure 24.2: Verification of the Yeh–Berkowitz correction. Left: 600 SPC/E waters + 1 NaCl in the slab geometry calculated for several box sizes L_z and several z-positions of the slab in the box (lines of the same color). Right: 600 SPC/E water molecules + one ion Na^+, with the additional Ballenegger–Arnold–Cerdà correction. Note the different scales of both figures.

24.3 Yeh–Berkowitz correction and charged systems

The Yeh–Berkowitz correction removes the influence of the slab dipole moment in the periodic boundary conditions, mimicking thys a 2D-periodic system. For the slab perpendicular to the z-coordinate, it is [9]

$$
\begin{aligned}
F_{z,i} &= -\frac{4\pi q_i}{V} M_z \quad \text{CGS} \\
&= -\frac{q_i}{V \epsilon_0} M_z \quad \text{SI}
\end{aligned}
$$

MACSIMUS allows to select the coordinate(s) for which this correction is applied by variable `el.corr`.

If the system is charged, either charges are redistributed (`el.bg=2`) or a background is added (`el.bg=1`). In the latter case, the above correction is modified [10] as follows

$$
\begin{aligned}
F_{z,i} &= -\frac{4\pi q_i}{V}(M_z - Q_{\text{tot}} z_i) \quad \text{CGS} \\
&= -\frac{q_i}{V \epsilon_0}(M_z - Q_{\text{tot}} z_i) \quad \text{SI}
\end{aligned}
$$

where $Q_{\text{tot}} = $ `charges.sum` = sum of charges = $-$ background charge.

Chapter 25

MD of Polarizable Force Fields

Jiří Kolafa, 1999

25.1 Notation

Vector from i to j: $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$, $r_{ij} = |\vec{r}_{ij}|$

Gradient: $\vec{\nabla}_i = \partial/\partial\vec{r}_i$, $\vec{\nabla}_i r_{ij} = -\vec{r}_{ij}/r_{ij}$, $\vec{\nabla}_j r_{ij} = \vec{r}_{ij}/r_{ij}$

Direct (tensor) product: $\overset{\leftrightarrow}{A} = \vec{a}\vec{b}$

Dot (scalar) product of two vectors: $A = \vec{a} \cdot \vec{b}$

Dot (matrix) product of two tensors: $\overset{\leftrightarrow}{C} = \overset{\leftrightarrow}{A} \cdot \overset{\leftrightarrow}{B}$

Inverse tensor: $\overset{\leftrightarrow}{A} \cdot \overset{\leftrightarrow}{A}^{-1} = \overset{\leftrightarrow}{A}^{-1} \cdot \overset{\leftrightarrow}{A} = 1$

25.2 Polarizability

Induced dipole on particle i is generally a function of field $\vec{E}_i$ at position $\vec{r}_i$:

$$\vec{\mu}_i = \vec{\mu}_i(\vec{E}_i) \tag{25.1}$$

where we assume that $\vec{\mu}_i = 0$ for $\vec{E}_i = 0$. As we shall see later, field $\vec{E}_i$ is rather an "effective" field which may contain certain terms behaving as electrostatic field. We shall consider two cases, the *linear polarizability*

$$\vec{\mu}_i(\vec{E}_i) = \overset{\leftrightarrow}{\alpha}_i \cdot \vec{E}_i \tag{25.2}$$

where $\overset{\leftrightarrow}{\alpha}_i$ is the polarizability tensor, and a simple model of *hyperpolarizability* exhibiting saturation given implicitly by

$$\vec{\mu}_i(\vec{E}_i) = \overset{\leftrightarrow}{\alpha}_i \cdot \frac{\vec{E}_i}{1 + \vec{E}_i \cdot \vec{\mu}_i/E_{\text{sat}}} \tag{25.3}$$

which can be solved for $\vec{E}_i$:

$$\vec{E}_i = \frac{\overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i}{1 - \vec{\mu}_i \cdot \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i/E_{\text{sat}}} \tag{25.4}$$

The "saturation energy" E_{sat} has a physical meaning of the energy of significant deviation from the linear law. For the denominators in the above formulas it holds

$$(1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\text{sat}})(1 - \vec{\mu}_i \cdot \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i / E_{\text{sat}}) = 1 \tag{25.5}$$

Finally, for $\overset{\leftrightarrow}{\alpha}_i = \alpha_i$ (scalar):

$$\vec{\mu}_i(\vec{E}_i) = \frac{\alpha_i \vec{E}_i}{\frac{1}{2} + \sqrt{\frac{1}{4} + \alpha_i E_i^2 / E_{\text{sat}}}} \tag{25.6}$$

25.3 Pair operators of electrostatic interaction

$$T_{ij} = \frac{1}{r_{ij}} \tag{25.7}$$

$$\vec{T}_{ij} = \vec{\nabla}_i T_{ij} = \frac{\vec{r}_{ij}}{r_{ij}^3} = -\vec{\nabla}_j T_{ij} = -\vec{T}_{ji} \tag{25.8}$$

$$\overset{\leftrightarrow}{T}_{ij} = \vec{\nabla}_i \vec{\nabla}_j T_{ij} = \vec{\nabla}_j \vec{T}_{ij} = -\vec{\nabla}_i \vec{T}_{ij} = \frac{1}{r_{ij}^3} - \frac{3\vec{r}_{ij}\vec{r}_{ij}}{r_{ij}^5} \tag{25.9}$$

$$\overset{\leftrightarrow}{\vec{T}}_{ij} = \vec{\nabla}_i \overset{\leftrightarrow}{T}_{ij} = -\vec{\nabla}_j \overset{\leftrightarrow}{T}_{ij} = 3\frac{\overset{\leftrightarrow}{1}\vec{r}_{ij}}{r_{ij}^5} + 3\frac{(\vec{r}_{ij})}{r_{ij}^5} + 3\frac{\vec{r}_{ij}\overset{\leftrightarrow}{1}}{r_{ij}^5} - 15\frac{\vec{r}_{ij}\vec{r}_{ij}\vec{r}_{ij}}{r_{ij}^7} \tag{25.10}$$

where $\overset{\leftrightarrow}{1}$ is a unit isotropic tensor, $\overset{\leftrightarrow}{1}_{\alpha\beta} = \delta_{\alpha\beta}$, and $(\vec{r})_{\alpha\beta\gamma} = \delta_{\alpha\gamma}\vec{r}_\beta$ so that $\vec{a} \cdot (\vec{r}) \cdot \vec{b} = (\vec{a} \cdot \vec{b})\vec{r}$.

25.4 Electrostatic energies

Charge-charge: $U_{\text{CC}}(i,j) = q_i T_{ij} q_j$

Dipole-charge: $U_{\text{DC}}(i,j) = \vec{\mu}_i \cdot \vec{T}_{ij} q_j$

Dipole-dipole: $U_{\text{DD}}(i,j) = \vec{\mu}_i \cdot \overset{\leftrightarrow}{T}_{ij} \cdot \vec{\mu}_j$

Polarization self-energy of dipole $\vec{\mu}_i$ is

$$U_{\text{self}}(\vec{\mu}_i) = \int \mathrm{d}\vec{\mu}_i \cdot \vec{E}_i(\vec{\mu}_i) \tag{25.11}$$

where the integration is along any path from zero induced dipole to $\vec{\mu}_i$ and $\vec{E}_i$ is the total effective electrostatic field acting on the dipole (what "effective" means becomes clear later). For the linear and saturated models, respectively:

$$U_{\text{lin}}(\vec{\mu}_i) = \frac{1}{2}\vec{\mu}_i \cdot \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i = \frac{1}{2}\vec{E}_i \cdot \vec{\mu}_i \tag{25.12}$$

$$U_{\text{sat}}(\vec{\mu}_i) = -\frac{E_{\text{sat}}}{2} \ln\left(1 - \frac{\vec{\mu}_i \cdot \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i}{E_{\text{sat}}}\right) = \frac{E_{\text{sat}}}{2} \ln\left(1 + \frac{\vec{E}_i \cdot \vec{\mu}_i}{E_{\text{sat}}}\right) \tag{25.13}$$

Note that $\lim_{E_{\text{sat}} \to \infty} U_{\text{sat}}(\vec{\mu}_i) = U_{\text{lin}}(\vec{\mu}_i)$.

Repulsive antipolarization (shell-core model, deformable dipole model—how shall we call it?) is based on an idea that close contact of two atoms pushes the electron shell around atom (normally anion) out of center giving rise to a dipole in the opposite direction that is the polarized dipole:

$$U_{\mathrm{rep}} = -\kappa_i \vec{\mu}_i \cdot \vec{f}_{ij} \tag{25.14}$$

where i is a polarizable atom, j any other atom, and

$$\vec{f}_{ij} = -\vec{\nabla}_i u_{ij}(r_{ij}) = \frac{\vec{r}_{ij}}{r_{ij}} u'_{ij}(r_{ij}) = -\vec{f}_{ji} \tag{25.15}$$

is the repulsive pair force acting on atom i due to atom j and u_{ij} is the repulsive pair potential. This potential does not include electrostatic interaction[1]. In the typical case i is an anion, j a cation, and then κ_i is positive.

25.5 Electrostatic field

Field at $\vec{r}_i$ caused by charge q_j at $\vec{r}_j$: $\vec{E}_i = -q_j \vec{T}_{ij}$

Field at $\vec{r}_i$ caused by dipole $\vec{\mu}_j$ at $\vec{r}_j$: $\vec{E}_i = -\vec{\mu}_j \cdot \overleftrightarrow{T}_{ij}$

Fictitious field caused by repulsive antipolarization: $\vec{E}_i = \kappa_i \vec{f}_{ij}$
This is not real electrostatic field but as regards its interaction with dipoles, (25.14) and (25.1), can be treated in the same way.

25.6 Total energy

The total electrostatic energy of a set of n polarizable atoms is

$$U = \sum_{i,j,i<j} q_i T_{ij} q_j + \sum_{i,j,i\neq j} \vec{\mu}_i \cdot \vec{T}_{ij} q_j + \sum_{i,j,i<j} \vec{\mu}_i \cdot \overleftrightarrow{T}_{ij} \cdot \vec{\mu}_j - \sum_{i,j,i\neq j} \kappa_i \vec{\mu}_i \cdot \vec{f}_{ij} + \sum_i U_{\mathrm{self}}(\vec{\mu}_i) \tag{25.16}$$

This $U(\{\vec{\mu}_i\})$ as a functional of $\{\vec{\mu}_i\}$ reaches a minimum for $\{\vec{\mu}_i\}$ satisfying (25.1) for all i where

$$\vec{E}_i \equiv \frac{\mathrm{d}U(\vec{\mu}_i)}{\mathrm{d}\vec{\mu}_i} = \sum_{j,j\neq i} \left(-q_j \vec{T}_{ij} - \vec{\mu}_j \cdot \overleftrightarrow{T}_{ij} + \kappa_i \vec{f}_{ij} \right) \tag{25.17}$$

is the total (real electrostatic and fictitious) field at $\vec{r}_i$.

For $\kappa_i = 0$ (term (25.14) is missing), we can get a somehow simpler expression not containing the dipole-dipole energy by inserting (25.1) and (25.17) into (25.16):

$$U = \sum_{i,j,i<j} q_i T_{ij} q_j + \frac{1}{2} \sum_{i,j,i\neq j} \vec{\mu}_i \cdot \vec{T}_{ij} q_j \tag{25.18}$$

$$+ \begin{cases} 0 & \text{for linear polarizability} \\ \sum_i \frac{1}{2} \left[E_{\mathrm{sat}} \ln(1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\mathrm{sat}}) - \vec{E}_i \cdot \vec{\mu}_i \right] & \text{for saturated polarizability} \end{cases}$$

which is especially simple if the polarizability is linear.

---

[1]There is a question whether it should generally include dispersion forces; the Tosi et al. potential does not include cation-anion dispersion forces at all.

25.7   Forces

Let us take a minus gradient of (25.16). Since $\vec{\mu}_j$ depend on $\vec{r}_i$, several terms containing $\vec{\nabla}_i \vec{\mu}_j$ appear, however, they all cancel out as a consequence of (25.1) with (25.17)[2]. The resulting force ready for implementing is

$$\vec{f}_i = -\vec{\nabla}_i U = \sum_{j,j\neq i} \left(-q_i \vec{T}_{ij} q_j + \vec{\mu}_i \cdot \overset{\leftrightarrow}{T}_{ij} q_j - q_i \overset{\leftrightarrow}{T}_{ij} \cdot \vec{\mu}_j - \vec{\mu}_i \cdot \overset{\leftrightarrow\!\!\leftrightarrow}{T}_{ij} \cdot \vec{\mu}_j - \kappa_i \vec{\mu}_i \cdot \vec{\nabla}_i \vec{f}_{ij} \right)$$
$$- \begin{cases} \vec{\mu}_i \cdot \vec{\nabla}_i \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i & \text{for linear polarizability} \\ \vec{\mu}_i \cdot \vec{\nabla}_i \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i (1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\text{sat}}) & \text{for saturated polarizability} \end{cases}$$

25.7.1   Gradient of the repulsive antipolarization

The force caused by the repulsive antipolarization term is

$$\vec{f}_{\text{rep},i} = -\kappa_i \vec{\mu}_i \cdot \vec{\nabla}_i \vec{f}_{ij} = \kappa_i \left[\frac{u'_{ij}}{r_{ij}} \vec{\mu}_i + \frac{\vec{\mu}_i \cdot \vec{r}_{ij}}{r_{ij}} \left(\frac{u'_{ij}}{r_{ij}} \right)' \vec{r}_{ij} \right], \tag{25.19}$$

to be completed by $\vec{f}_{\text{rep},j} = -\vec{f}_{\text{rep},i}$.

25.7.2   Gradient of the polarizability tensor

For isotropic and constant (=not depending on configuration) polarizability, $\overset{\leftrightarrow}{\alpha}_i = \overset{\leftrightarrow}{1}\alpha_i$, $\vec{\nabla}_j \alpha_i = 0$, the last term in (25.19) is zero. Let us consider the simplest case of axial polarizability of atom i where the polarizability in the direction of a chemical bond (towards certain atom with position $\vec{r}_t$) is α_{zz} and in the perpendicular directions it is $\alpha_{xx} = \alpha_{yy}$. The polarization tensor is

$$\overset{\leftrightarrow}{\alpha}_i = \alpha_{xx} \overset{\leftrightarrow}{1} + (\alpha_{zz} - \alpha_{xx}) \frac{\vec{r}_{it} \vec{r}_{it}}{r^2} \tag{25.20}$$

The inverse tensor is

$$\overset{\leftrightarrow}{\alpha}_i^{-1} = \alpha_{xx}^{-1} \overset{\leftrightarrow}{1} - \frac{\alpha_{zz} - \alpha_{xx}}{\alpha_{zz}\alpha_{xx}} \frac{\vec{r}_{it} \vec{r}_{it}}{r^2} \tag{25.21}$$

by taking the gradient we obtain the corresponding 'axial polarization' force term

$$\vec{f}_{\text{ax},i} = -\vec{\mu}_i \cdot \vec{\nabla}_i \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i = \frac{\alpha_{zz} - \alpha_{xx}}{\alpha_{zz}\alpha_{xx}} \left[\frac{\vec{\mu}_i \cdot \vec{r}_{it}}{r_{it}^2} \vec{\mu}_i - \frac{(\vec{\mu}_i \cdot \vec{r}_{it})^2}{r_{it}^4} \vec{r}_{it} \right] \tag{25.22}$$

and finally $\vec{f}_{\text{ax},t} = -\vec{f}_{\text{ax},i}$.

25.7.3   Fluctuating charge

Currently, MACSIMUS implements the fluctuating charge model of water with four charges. The method comes from the FQ models [39].

---

[2]That is why it is more convenient to take a gradient of (25.16) and not of the simpler form (25.18) which would in fact lead to more complicated calculations to eliminate terms containing $\vec{\nabla}_j \vec{\mu}_i$

Let us consider a neutral molecule with four interaction sites numbered $i \in \{0, 1, 2, 3\}$ which do not lie in a plane. Each site bears a permanent charge q_{i0}, $i \in \{0, 1, 2, 3\}$, $\sum_i q_{i0} = 0$. In electric field with potential Φ_i at site i the charges change into $q_i = q_{i0} + \delta q_i$. We write the electrostatic energy as

$$U = U_{\mathrm{self}} + \sum_i (q_{i0} + \delta q_i)\Phi_i, \qquad (25.23)$$

where the self energy is [39]

$$U_{\mathrm{self}} = \sum_i \sum_j \frac{A_{ij}}{2} \delta q_i \delta q_j. \qquad (25.24)$$

At fixed field Φ_i the configuration of charges minimizes energy U under constraint $\sum_i \delta q_i = 0$. By adding term $\lambda \sum_i \delta q_i$ to (25.23) (λ is the Lagrange multiplier) and taking derivatives over δq_i, we arrive at four equations

$$\sum_j A_{ij} \delta q_j + \Phi_i + \lambda = 0. \qquad (25.25)$$

Given A_{ij} and Φ_i, these linear equations (along with $\sum_i \delta q_i = 0$) yield the unknown charges.

The polarizability tensor α can be calculated by applying homogeneous fields in three directions and calculating the response dipole moment. However, in constructing the force field the task is opposite: to determine ten independent components of symmetric matrix A_{ij} given six independent components of α. The task is overdetermined, but all solutions are equivalent because the set of shifted constants

$$A'_{ij} = A_{ij} + a_i \qquad (25.26)$$

gives the same charges and thus the same energy (the Lagrange multiplier λ need not be the same, though). The dimensionality of this space is four.

For the FQ4 model of water there are a number of symmetries. We have five independent components of matrix A

$$A = \begin{pmatrix} A_{\mathrm{HH}} & A_{\mathrm{H'H}} & A_{\mathrm{HL}} & A_{\mathrm{HL}} \\ A_{\mathrm{H'H}} & A_{\mathrm{HH}} & A_{\mathrm{HL}} & A_{\mathrm{HL}} \\ A_{\mathrm{HL}} & A_{\mathrm{HL}} & A_{\mathrm{LL}} & A_{\mathrm{L'L}} \\ A_{\mathrm{HL}} & A_{\mathrm{HL}} & A_{\mathrm{L'L}} & A_{\mathrm{LL}} \end{pmatrix} \qquad (25.27)$$

where HH stands for the diagonal (quadratic) terms, $H_i H_i$, while H'H denotes the off-diagonal terms, $H_i H_j$, $i \neq j$. There are only two independent conditions (25.26). We therefore we will impose $A_{\mathrm{H'H}} = A_{\mathrm{L'L}} = 0$. The solution is

$$\delta q_{\mathrm{H1}} = \frac{(4A_{\mathrm{HL}} - 2A_{\mathrm{HH}} - A_{\mathrm{LL}})\Phi_{\mathrm{H1}} + (A_{\mathrm{LL}} - 4A_{\mathrm{HL}})\Phi_{\mathrm{H2}} + A_{\mathrm{HH}}(\Phi_{\mathrm{L1}} + \Phi_{\mathrm{L2}})}{2A_{\mathrm{HH}}(A_{\mathrm{HH}} + A_{\mathrm{LL}} - 4A_{\mathrm{HL}})} \qquad (25.28)$$

and the remaining q_i can be obtained by H1 $\leftrightarrow$ H2 and H $\leftrightarrow$ L symmetries.

For the FQ4 model of water in the standard orientation the polarizability tensor is diagonal and thus the remaining three values, A_{HL}, $A_{\mathrm{H'H}}$, $A_{\mathrm{L'L}}$, can be determined from the diagonal components of the polarizability tensor.

Let the positions of charge sites are (with respect to Oxygen):

$$\vec{r}_{\mathrm{H1}} = (H_x, H_y, 0) \quad \vec{r}_{\mathrm{H2}} = (H_x, -H_y, 0) \qquad (25.29)$$
$$\vec{r}_{\mathrm{L1}} = (L_x, 0, L_z) \quad \vec{r}_{\mathrm{L2}} = (L_x, 0, -L_z) \qquad (25.30)$$

The the polarizability tensor is

$$\alpha = \text{diag}\left(\frac{2(H_x - L_x)^2}{A_{\text{HH}} + A_{\text{LL}} - 4A_{\text{HL}}}, \ \frac{2H_y^2}{A_{\text{HH}}}, \ \frac{2L_z^2}{A_{\text{LL}}}\right) \tag{25.31}$$

from which the unknown constants A_{HH}, A_{LL}, A_{HL} can be readily calculated.

25.7.4   Implementation

Fluctuating charges are stored in the 2nd vector (rpol). ASPC is directly applicable.

Chapter 26

Time-reversible velocity predictor

Time-reversible predictors for Verlet+SHAKE with a velocity-dependent rhs have been published [40]. For version with a barostat, see Sect. 12.3.2.

26.1 The task

Our task is to integrate numerically by the Verlet method (with optional SHAKE) the following set of equations

$$\ddot{x} = a(x, \dot{x}) \tag{26.1}$$

where x stands for the set of $\vec{r}_i$ and the Nose variable $\xi = \ln s$. The Verlet algorithm reads as

$$x(t+h) = 2x(t) - x(t-h) + h^2 a(t) \tag{26.2}$$

where $\dot{x}(t)$ (needed to calculate $a(t)$) is not known. It may be expressed by

$$\dot{x}(t) = \frac{x(t+h) - x(t-h)}{2h} + \mathcal{O}(h^2) \tag{26.3}$$

but the resulting equations have to be calculated by iterations. If SHAKE is involved, the SHAKE part must be repeated in this iteration. Another possibility is the velocity Verlet with RATTLE.

26.2 MACSIMUS solution

MACSIMUS implements a set of predictors with a good time-reversibility. The velocity predictor can be written in the form

$$\dot{x}^p(t) = \frac{1}{h} \sum_{i=0}^{k+1} A_i x(t - ih), \tag{26.4}$$

The $k + 2$ $A_i, i = 0, ..., k + 1$ can be determined from the Taylor expansion of the right-hand side

$$\sum_{i=0}^{k+1} A_i x(t - ih) = \sum_{i=0}^{\infty} X_i x^{(i)} h^i \tag{26.5}$$

It must hold ($k + 2$ equations)

$$
\begin{aligned}
X_0 &= \sum_{i=0}^{k+1} A_i = 0 \\
X_1 &= -\sum_{i=0}^{k+1} i A_i = -1 \\
X_2 &= \sum_{i=0}^{k+1} i^2 A_i = 0 \\
X_4 &= \sum_{i=0}^{k+1} i^4 A_i = 0 \\
&\vdots \\
X_{2k} &= \sum_{i=0}^{k+1} i^{2k} A_i = 0
\end{aligned}
$$

The odd terms are not nullified because they are time-reversible.

The solution is

$$
\begin{aligned}
A_0 &= \frac{2k + 1}{k + 1} \\
A_1 &= -2(2k + 1)\frac{1}{k + 2} \\
A_2 &= +2(2k + 1)\frac{k}{(k + 2)(k + 3)} \\
A_3 &= -2(2k + 1)\frac{k(k - 1)}{(k + 2)(k + 3)(k + 4)} \\
&\vdots
\end{aligned}
$$

MACSIMUS code uses expansion in the first differences

$$
\dot{x}^p(t) = \sum_{i=0}^{k} B_i \frac{x(t - ih) - x(t - (i + 1)h)}{h} \tag{26.6}
$$

with $B_0 = A_0$ and

$$
B_j = (-1)^j (2k + 1)\frac{k(k - 1)\cdots(k + 1 - j)}{(k + 1)(k + 2)\cdots(k + 1 + j)} \tag{26.7}
$$

or recursively

$$
\begin{aligned}
B_0 &= \frac{2k + 1}{k + 1} \\
B_j &= -B_{j-1} \cdot \frac{k + 1 - j}{k + 1 + j}, \quad j > 0
\end{aligned}
$$

which is directly coded in MACSIMUS. The default is $k = 2$.

Several first predictors in the Verlet-style:

$$
\dot{x}_0^p(t) = \frac{1}{h}[x(t) - x(t - h)]
$$

$$\dot{x}_1^p(t) = \frac{1}{h}\left[1.5x(t) - 2x(t-h) + 0.5x(t-2h)\right]$$

$$\dot{x}_2^p(t) = \frac{1}{h}\left[\frac{5}{3}x(t) - 2.5x(t-h) + x(t-2h) - \frac{1}{6}x(t-3h)\right]$$

$$\dot{x}_3^p(t) = \frac{1}{h}\left[1.75x(t) - 2.8x(t-h) + 1.4x(t-2h) - 0.4x(t-3h) + 0.05x(t-4h)\right]$$

The same predictors in the leap-frog style, where $\dot{v}(t-h/2) = [x(t) - x(t-h)]/h$, etc.

$$\dot{x}_0^p(t) = v(t-h/2)$$

$$\dot{x}_1^p(t) = 1.5v(t-h/2) - 0.5v(t-3h)/2$$

$$\dot{x}_2^p(t) = \frac{5}{3}v(t-h/2) - \frac{5}{6}v(t-3h/2) + \frac{1}{6}v(t-5h/2)$$

$$\dot{x}_3^p(t) = 1.75v(t-h/2) - 1.05v(t-3h/2) + 0.35v(t-5h/2) - 0.05v(t-7h/2)$$

26.3   Algorithm

One step of the combined Verlet+SHAKE method is

- calculate forces (accelerations) $a(t)$ from known $x(t)$

- predict velocities $\dot{x}$ from known $x(t) - x(t-h)$, $x(t-h) - x(t-2h)$, ..., $x(t-kh) - x(t-(k+1)h)$

- perform one step of the Verlet method (12.5) to get $r(t+h)$

- run the SHAKE algorithm; $x(t+h)$ is modified and $x(t+h) - x(t)$ recalculated

- calculate the kinetic temperature (MACSIMUS supports several formulas, see the switch VERLET; VERLET=9 uses the predicted value)

- perform one step of the Verlet method (12.5) to get $\dot{\xi}(t+h)$ and $\xi(t+h)$

- advance time, $t := t + h$, and evaluate the Hamiltonian

Chapter 27

Always Stable Predictor-Corrector (ASPC) instant

Based on papers [41, 42, 43, 44].

27.1   Task

To integrate numerically the Newton equations of motion

$$\ddot{\boldsymbol{r}}_i = \frac{1}{m_i}\boldsymbol{f}_i(\boldsymbol{r}_1,\ldots,\boldsymbol{r}_N;\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_m) \tag{27.1}$$

where $\boldsymbol{r}_i$ are positions of atoms (nuclei) and $\boldsymbol{\mu}_i$ are any variables which are given by an implicit equation of a self-consistent field type:

$$\boldsymbol{\mu}_i = \boldsymbol{M}_i(\boldsymbol{r}_1,\ldots,\boldsymbol{r}_N;\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_m), \quad i = 1,\ldots,m \tag{27.2}$$

This equation must converge for all $\boldsymbol{r}_1,\ldots,\boldsymbol{r}_N$ from the trajectory and (a linearization of) $M_{ij} = \boldsymbol{M}_i(\boldsymbol{\mu}_j)$ must be symmetric.

Example of $\boldsymbol{\mu}$: induced dipoles, $\boldsymbol{\mu}_i = \alpha_i\boldsymbol{E_i}(\boldsymbol{r}_1,\ldots,\boldsymbol{r}_N;\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_m)$, where $\boldsymbol{E}_i$ is the electric field

27.2   ASPC

The recommended version with $\mathcal{O}(h^7)$ time reversibility follows:

Predictor:

$$\mu^p(t) = 2.8\mu(t-h) - 2.8\mu(t-2h) + 1.2\mu(t-3h) - 0.2\mu(t-4h) \tag{27.3}$$

where $\mu \equiv \{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_m\}$, and similarly below.

Corrector:

$$\mu(t) = \omega M(r(t);\mu^p(t)) + (1-\omega)\mu^p(t) \tag{27.4}$$

where $\omega = \frac{4}{7}$ guarantees stability (for any converging SCF equation), but a certain $\omega > \frac{4}{7}$ may give more accurate results.

In the above equation, $M(r(t); \mu^p(t))$ is calculated at time t and current $r(t)$. Normally the forces $f(t)$ (along with E) are calculated at the same step. Then one step of Verlet gives $r(t+h)$ (which becomes $r(t)$ in the next step) and the corrector gives $\mu(t)$ (which becomes $\mu(t-h)$ in the next step, ready for the predictor).

The method may work also in certain cases of asymmetric M_{ij}. It will probably work also for complex $\boldsymbol{\mu}$.

Chapter 28

Specific heat C_V in the molecular dynamics microcanonical ensemble

In the *canonical* ensemble we have for the kinetic energy E_k and potential (configurational) energy E_p:

$$\mathrm{Var} E_x = kT^2 \frac{\partial E_x}{\partial T} \equiv kT^2 E_x' \tag{28.1}$$

where $x \in \{k, p\}$ and

$$\mathrm{Cov}(E_p, E_k) = 0 \tag{28.2}$$

$$E_k' = fk/2 \tag{28.3}$$

$$E_p' + E_k' = C_V \tag{28.4}$$

where f is the number of degrees of freedom and C_V is the heat capacity (at constant volume) of the whole system.

Thus the (unnormalized) probability distribution for given T and a state with given E_k and E_p is:

$$w(T, E_p, E_k) \sim \exp\left[-\frac{(\delta E_p - E_p'\delta T)^2}{2\,\mathrm{Var} E_p} - \frac{(\delta E_k - E_k'\delta T)^2}{2\,\mathrm{Var} E_k} \right] \tag{28.5}$$

where we linearize at certain point $(T, E_p, E_k) = (T_0, E_{p,0}, E_{k,0})$, $\delta X \equiv X - X_0$. In the MD NVE ensemble there is $E_p + E_k = \mathrm{const}$. We write $E_p = -E_k \equiv E$ and express E_x' via $\mathrm{Var} E_x$:

$$w(T, E) \sim \exp\left[-\left(\frac{1}{2k_\mathrm{B} T^2 E_p'} + \frac{1}{2k_\mathrm{B} T^2 E_k'} \right) \delta E^2 - \left(\frac{E_p'}{2k_\mathrm{B} T^2} + \frac{E_k'}{2k_\mathrm{B} T^2} \right) \delta T^2 \right] \tag{28.6}$$

Note that there is no $\delta E \delta T$ cross term! Thus, from the term at δT^2,

$$\mathrm{Var} T = \frac{kT^2}{E_p' + E_k'} = \frac{kT^2}{C_V} \tag{28.7}$$

which is known formula for the fluctuation of temperature in the microcanonical ensemble – useless for MD because we do not know T (the "kinetic temperature" derived from E_k is *not* T).

From the term at δE^2 we have

$$\mathrm{Var} E = \mathrm{Var} E_p = \mathrm{Var} E_k = \frac{k_\mathrm{B} T^2}{1/E_p' + 1/E_k'} = \frac{k_\mathrm{B} T^2}{1/(C_V - E_k') + 1/E_k'} \tag{28.8}$$

and because we know E'_k finally

$$C_V = \frac{fk}{2} \left[\left(\frac{2\,T^2}{f\,\mathrm{Var}T_k} - 1 \right)^{-1} + 1 \right] \tag{28.9}$$

where $T_k \equiv E_k/(fk/2)$ is the "kinetic temperature".

Chapter 29

Dielectric constant in SI

The permittivity ϵ is defined via the electric field intensity $\vec{E}$ and the electric displacement field $\vec{D}$. In SI it holds

$$\vec{D} = \epsilon\vec{E} = \epsilon_0\epsilon_r\vec{E} \tag{29.1}$$

where $\epsilon_0 = 8.85418782\text{e-}12$ F/m is the vacuum permittivity and ϵ_r is the relative permittivity which equals the dielectric constant defined in the CGS units. The permittivity of an isotropic medium is a scalar.

In CGS simply $\vec{D} = \epsilon_r\vec{E}$ (usually subscript $_r$ is not used, but we will keep it for clarity). Note that MACSIMUS uses internally CGS-like (Gaussian electrostatic) units, however, it transparently transforms the units mostly into SI; one notable exceptions is the dipole moment where the internal units are used because it is not clear whether to prefer Debye or Cm.

The electric displacement is

$$
\begin{aligned}
\vec{D} &= \epsilon_0\vec{E} + \vec{P} \quad \text{SI} \\
\vec{D} &= \vec{E} + 4\pi\vec{P} \quad \text{CGS}
\end{aligned}
$$

where

$$\vec{P} = \frac{\vec{M}}{V} \tag{29.2}$$

is the electric polarization which equals the volume density of dipole moment $\vec{M}$; this equation in SI and CGS is formally identical, only the units differ.

For completeness, note the electric work per unit volume in both systems:

$$
\begin{aligned}
\frac{\mathrm{d}W}{V} &= \vec{E} \cdot \mathrm{d}\vec{D} \quad \text{SI} \\
\frac{\mathrm{d}W}{V} &= \frac{\vec{E} \cdot \mathrm{d}\vec{D}}{4\pi} \quad \text{CGS}
\end{aligned}
$$

29.1 Dielectric constant from external field

In simulations we apply an external (microscopic) field $\vec{E}^{\,\text{ext}}$ (entered as `el.E[]` in input data in the SI units of V/m). The total intensity $\vec{E}$ is a sum of the the external field and the

field caused by charges in the system. It depends on the boundary conditions. In the Ewald periodic boundary conditions we have a continuum (of relative permittivity[1] $\epsilon_r' = $ `el.epsinf`) surrounding the infinite array of periodic boxes spherically in infinity; to be able to write formulas, we will consider the radius $R \gg L$. The polarization $\vec{P}$ of the sphere is equivalent to a point dipole $\frac{4}{3}\pi R^3 P$ placed in the center. It induces charges at the surface of the R-cavity and hence a field which damps the external field. It holds [45] (in SI):

$$\vec{E} = \vec{E}^{\text{ext}} - \frac{1}{\epsilon_0}\frac{\vec{P}}{2\epsilon_r' + 1} \tag{29.3}$$

or equivalently

$$\vec{E} = \frac{3}{2 + \epsilon_r'}\vec{E}^{\text{ext}} \tag{29.4}$$

Let the external field acts in the direction of z (`el.E[2]`). During simulation we determine quantity

$$\chi_z = \frac{\langle M_z \rangle}{\epsilon_0 E_z^{\text{ext}} V} \tag{29.5}$$

or in the units used by MACSIMUS

$$\chi_z = \frac{4.4266218\mathrm{e}{+09} \times (\texttt{Mz}/\mathrm{prog.unit})}{[E_z^{\text{ext}}/(\mathrm{V/m})] \times (V/\text{Å}^3)} \tag{29.6}$$

The dielectric constant is then

$$\epsilon_r = \frac{\epsilon_0 E_z + P_z}{\epsilon_0 E_z} = 1 + \frac{2\epsilon_r' + 1}{2\epsilon_r' + 1 - \chi_z}\chi_z = 1 + \frac{1}{\frac{1}{\chi_z} - \frac{1}{2\epsilon_r'+1}} \tag{29.7}$$

where we inserted E_z from (29.3). Some like the equivalent implicit formula,

$$\chi_z = \frac{(\epsilon_r - 1)(2\epsilon_r' + 1)}{2\epsilon_r' + \epsilon_r} \tag{29.8}$$

For the tin-foil boundary conditions ($\epsilon_r' = $ `el.epsinf` $= \infty$) it holds $\epsilon_r = 1 + \chi_z$.

It is inefficient to use $\epsilon_r' = 1$ for models with large ϵ_r, because then χ_z is close to 3 and we have $3 - \chi_z$ in the denominator of (29.5). On the other hand, large ϵ_r' increases correlation length and may also lead to dipole moments close to saturation and systematic errors. Anyway, stronger external field is needed to get the same response for small ϵ_r'.

Formula (29.7) is exact for both polarizable and nonpolarizable systems provided that the field is small enough. It is recommended to run several simulations with decreasing field (and increasing length to compensate for decreased accuracy).

29.2 Fluctuation formulas

For the dipole moment of the simulation cell (one configuration of charges) under influence of the external electric field in the z-direction it holds

$$\vec{M}(\vec{E}^{\text{ext}}) = \vec{M}(0) + \frac{\partial \vec{M}}{\partial \vec{E}^{\text{ext}}}\vec{E}^{\text{ext}} \tag{29.9}$$

---

[1]For the short-range (cutoff) electrostatics `el.epsinf` $= 1$ (?)

where $\partial \vec{M}/\partial \vec{E}^{\text{ext}}$ stands for the induced dipole moment per unit electric field at constant positions of atoms. (Generally, it is a tensor, however, the off-diagonal terms are small and their average in isotropic systems is zero.) By repeating the derivation of the fluctuation formula [45] we get a response caused by a small external field. We write it in the z-direction

$$\langle M_z(E_z^{\text{ext}}) \rangle = \left(\left\langle \frac{\partial M}{\partial E^{\text{ext}}} \right\rangle + \frac{\text{Var} M_z}{k_{\text{B}} T} \right) E_z^{\text{ext}} \tag{29.10}$$

In isotropic systems $\text{Var} M = 3 \, \text{Var} M_z$. We will express the direct response by a dimensionless number χ^{hf} (a sort of high-frequency susceptibility including boundary conditions[2])

$$\chi^{\text{hf}} = \frac{1}{\epsilon_0 V} \left\langle \frac{\partial M_z}{\partial E_z^{\text{ext}}} \right\rangle \quad \text{SI}$$

$$\chi^{\text{hf}} = \frac{1}{V} \left\langle \frac{\partial M_z}{\partial E_z^{\text{ext}}} \right\rangle \quad \text{CGS}$$

and the total response

$$\chi = \chi^{\text{hf}} + \frac{\text{Var} M}{3 \epsilon_0 V k_{\text{B}} T} \quad \text{SI}$$

$$\chi = \chi^{\text{hf}} + \frac{\text{Var} M}{3 V k_{\text{B}} T} \quad \text{CGS}$$

Similarly as (29.7) and (29.8) one can derive

$$\epsilon_r = 1 + \frac{1}{\frac{1}{[4\pi]\chi} - \frac{1}{2\epsilon_r' + 1}} \tag{29.11}$$

or implicitly

$$[4\pi]\chi = \frac{(\epsilon_r - 1)(2\epsilon_r' + 1)}{2\epsilon_r' + \epsilon_r}, \tag{29.12}$$

where $[4\pi] = 1$ applies for SI and $[4\pi] = 4\pi$ for CGS. Note that MACSIMUS calculates $\text{Var} M$ as $\langle M^2 \rangle$ unless an external field is applied; in the latter case $\langle M_z \rangle^2$ is subtracted.

For polarizable models it remains to determine $\langle \chi^{\text{hf}} \rangle$. MACSIMUS offers a direct numerical method based on applying an external field (`scf.E`) and iterations until the dipoles are accurate enough (`scf.epsx`, `scf.omegax`); the original and new dipole moment are subtracted and divided by the field.

By the same arguments which lead from (29.3) to (29.7) one can relate the "susceptibility" χ^{hf} to the high-frequency (instantaneous, optical) dielectric constant ϵ^{hf},

$$[4\pi]\chi^{\text{hf}} = (\epsilon_r^{\text{hf}} - 1) \frac{1 + 2\epsilon_r'}{2\epsilon_r' + \epsilon_r^{\text{hf}}} \tag{29.13}$$

It can be approximated by the the Clausius–Mossotti formula from the total polarizability (assumed to be linear), see eq. (2.6) in [46] (with $\epsilon_r^{\text{hf}} = \epsilon^\infty$)

$$\frac{\epsilon_r^{\text{hf}} - 1}{\epsilon_r^{\text{hf}} + 2} = \frac{1}{3\epsilon_0 V} \sum_i \alpha_i^{\text{SI}} = \frac{4\pi}{3V} \sum_i \alpha_i \tag{29.14}$$

In the formula above, α_i^{SI} is the SI polarizability of particle i and $\alpha_i = \alpha_i^{\text{SI}}/4\pi\epsilon_0$ is its polarizability volume (identical to "polarizability" in CGS and usually referred to as just "polarizability"; in MACSIMUS measured in Å^3). The induced dipole in SI is $\vec{\mu} = \alpha_{\text{SI}} \vec{E} = 4\pi\epsilon_0 \alpha \vec{E}$.

---

[2]Note that $\chi_{\text{SI}} = 4\pi \chi_{\text{CGS}}$

29.2.1   Notes

Saturation

To be able to calculate saturation, the molecular dipole moments should be known. This requires at least `lag.M=1` even though with the Ewald summation the total dipole moment is calculate in a different module.

Clausius–Mossotti

Eq. (29.14) is an approximation based on the assumption that a polarizable dipole is in a spherical cavity surrounded by a continuum of the optical dielectric constant ϵ_r^{hf}. An error in determination of ϵ_r^{hf} propagates to the final result more for $\epsilon_r' = 1$ and less for $\epsilon_r' = \infty$. E.g., for the DC97 water model, the error $\epsilon_r^{\mathrm{hf}} \pm 0.01$ causes the error $\epsilon_r \pm 7$ with $\epsilon_r' = 1$ while only negligible $\epsilon_r \pm 0.01$ with $\epsilon_r' = \infty$.

Eq. (29.11) for $\epsilon_r' = \infty$ (tin-foil b.c.) reduces to

$$\epsilon_r = \epsilon_r^{\mathrm{hf}} + [4\pi]\chi \tag{29.15}$$

which is identical to the formula in [47].

Bug in cook V2.6e and older

In cook V2.6e and older, formula (SI):

$$\chi^{\mathrm{hf}} = \frac{4\pi}{V} \sum_i \alpha_i \tag{29.16}$$

was incorrectly used. This formula neglects mutual interactions of induced dipoles. This formula is identical to (29.14) for $\epsilon_r' = 1$ and also for small ϵ_r^{hf} (a second-order term in $(\epsilon_r^{\mathrm{hf}} - 1) \approx \frac{4\pi}{V} \sum_i \alpha_i^2$ is neglected). E.g., for water at $25\,^{\circ}\mathrm{C}$ there is $\frac{4\pi}{V} \sum_i \alpha_i = 0.616$ and $\epsilon_r^{\mathrm{hf}} = 1.775$. The error caused by using equation (29.16) instead of (29.14) is 0.16, to be compared to the bulk water value $\epsilon_r = 78$.

CGS to SI conversion

quantity	CGS $\rightarrow$ SI
c	$c \rightarrow 1/\sqrt{\epsilon_0 \mu_0}$
$X = \vec{E}, U$	$X \rightarrow \sqrt{4\pi\epsilon_0} X$
D	$D \rightarrow \sqrt{4\pi/\epsilon_0} D$
$X = Q, I, \vec{P}, \vec{\mu}$	$X \rightarrow X/\sqrt{4\pi\epsilon_0}$
ϵ	$\epsilon \rightarrow \epsilon/\epsilon_0$
χ	$\chi \rightarrow \chi/4\pi$
R	$R \rightarrow 4\pi\epsilon_0$

Example: The CGS energy density (of linear material)

$$\frac{1}{4\pi} \vec{E} \cdot \vec{D} \tag{29.17}$$

transforms into

$$\frac{1}{4\pi}\sqrt{4\pi\epsilon_0}\vec{E}\cdot\sqrt{\frac{4\pi}{\epsilon_0}}\vec{D} = \vec{E}\cdot\vec{D} \tag{29.18}$$

29.3 Controlling saturation

The saturation of the dipole moment of the cell is for **simulations with external electric field** in the z-direction defined by

$$S = \frac{M_z}{\sum\mu}, \quad \text{averaged: } S = \frac{\langle M_z\rangle}{\sum\langle\mu\rangle} \tag{29.19}$$

where M_z is the z-component of the cell dipole moment and the sum is over molecule dipole moments. Thus, the denominator is the maximum dipole moment of the cell with all molecules aligned. (For nonpolarizable models μ are constants, for polarizable models the molecule dipole moments depend on the field. In either case the maximum saturation is $100\,\%$.) Technical note: with the Ewald summation, at least `lag.M=1` is needed.

For **zero-field simulations** (fluctuation formula route) we define the saturation as

$$S = \frac{|M|}{\sum\mu}, \quad \text{averaged: } S = \frac{\langle M^2\rangle^{1/2}}{\sum\langle\mu\rangle} = \frac{\sqrt{\text{Var}M}}{\sum\langle\mu\rangle} \tag{29.20}$$

The systematic error of the dielectric constant depends quadratically on the saturation, $\delta\epsilon_r \propto S^2$. In contrast, the statistical error is inversely proportional, $\delta\epsilon_r \propto S^{-1}$. Usually $S = 0.1$ guarantees the systematic error about $S^2 \approx 1\,\%$ which is comparable to a statistical error in moderately long runs.

It may be difficult to set the parameters (ϵ'_r in the fluctuation route and E_z in the external field route). MACSIMUS supports an automatic procedure for this setup. Variable `el.sat` is the target saturation. The controlling parameter, $\texttt{xinf} = 1/(2\epsilon'_r + 1)$[3] or E_z, is adjusted by a procedure similar to the Berendsen thermostat or barostat with estimated correlation time `tau.sat` (in ps). Recommended `tau.sat` are a few ps for setting the electrostatic field; the real `tau.sat` is only slightly longer. For the fluctuation route much longer `tau.sat` is needed, in addition, the real correlation time is shorter. The correlation time grows with system size. In both cases the simulation time should be several correlation times.

The values of parameters `el.epsinf` and `el.E` are not stored in the .cfg file, however, they are not reset between sweeps unless explicitly specified. If you wish to restart the simulation, you should take the last average from the .prt file (section "saturation autoset") and use it in the input data. Alternatively, you may use a **negative `tau.sat`** which works as for |`tau.sat`| and the following actions are performed:

- The calculated average[4] is assigned to the respective variable, `el.epsinf` or `el.E`.

- `init=1` is assigned to be used for the next sweep (you may override it, e.g., by `init=2` at next sweep start).

- `tau.sat=0` is assigned.

---

[3]This is the M^2 coupling constant, see (24.6)
[4]$1/(2\epsilon'_r + 1)$ for zero field

This way a productive run may start. In case of restart you still MUST write the used `el.epsinf` or `el.E` to the data! Examples:

- **field route:**
  ```
  el.sat=0.1 tau.sat=1 el.E[2]=2e8 no=10/(h*noint);
  init=1 tau.sat=-3 no=10/(h*noint);
  no=1000/(h*noint);;;;;
  ```
  It is useful to add `Ez` to the .cpi-file to monitor convergence; it is in the program units, 1 p.u. = 3.522592e8 V/m.

- **fluctuation route:**
  ```
  el.sat=0.1 tau.sat=1000 el.epsinf=50 no=500/(h*noint);
  init=1 tau.sat=-3000 no=500/(h*noint);
  no=1000/(h*noint);;;;;
  ```
  It is useful to add $\texttt{xinf} = 1/(2*\texttt{el.epsinf} + 1) = 1/(2\epsilon'_r + 1)$ to the `.cpi` file to monitor convergence; also available but less useful is $\texttt{einf} = \texttt{el.epsinf} = \epsilon'_r$. The obtained ϵ'_r as well as $1/(2\epsilon'_r + 1)$ may be negative. There is no problem with the method and calculated values of ϵ_r if ϵ'_r is "unphysically" negative.

In the first approximation, the dielectric constant depends on the saturation by

$$\epsilon(S) = \epsilon(0) + aS^2 \tag{29.21}$$

where the typical values of a are between $-\epsilon$ and $-\epsilon/2$. Thus, to obtain ϵ with a 1% precision, the saturation should be $S \leq 0.1$; smaller values increase the statistical error.

29.3.1   Extrapolation to zero saturation

Some efficiency is gained by extrapolation. We assume that the dependence of the dielectric constant on S^2 is linear, $\epsilon(S) = \epsilon(0) + aS^2$, where a is a constant. Our task is to determine S_1, $S_1 < S_2$, as well as the corresponding computer times, t_1 and t_2 (given the sum $t = t_1 + t_2$) so that the resulting dielectric constant is as accurate as possible. From two equations

$$\begin{aligned}
\epsilon(S_1) &= \epsilon(0) + aS_1^2 \\
\epsilon(S_2) &= \epsilon(0) + aS_2^2
\end{aligned}$$

one can readily calculate

$$\epsilon(0) = \frac{S_2^2 \epsilon(S_1) - S_1^2 \epsilon(S_2)}{S_2^2 - S_1^2} \tag{29.22}$$

The standard error $\delta\epsilon(0)$ of $\epsilon(0)$ is given by

$$\delta\epsilon(0)^2 = \left(\frac{S_2^2}{S_2^2 - S_1^2}\right)^2 \delta\epsilon(S_1)^2 + \left(\frac{S_1^2}{S_2^2 - S_1^2}\right)^2 \delta\epsilon(S_2)^2. \tag{29.23}$$

If the saturation is small, the error in determining $\langle M_z \rangle$ may be considered as independent on the saturation. Hence from (29.5) and the saturation definition it follows that the error of χ_z is inversely proportional to the saturation and if this error is small enough, the same holds

true for the error in the dielectric constant. For long enough simulations, the error is inversely proportional to the square root of the time. We may then write

$$\delta\epsilon(S) = \frac{\delta}{t^{1/2}S}. \tag{29.24}$$

By inserting this approximation into (29.23) and denoting $x = S_1^2/S_2^2$, we arrive at

$$\delta\epsilon(0)^2 = \frac{\delta^2}{S_2^2}\frac{1}{(1-x)^2}\left(\frac{1}{xt_1} + \frac{x^2}{t - t_1}\right) \tag{29.25}$$

which is to be minimized for $x \in (0,1)$ and $t_1 \in (0,t)$. The result is $x = 1/4$ (i.e., $S_1 = S_2/2$) and $t_1 = \frac{8}{9}t$.

We will therefore run two simulations, one with external field $E_{z,2}$ (chosen so that the saturation does not exceed the linear region) and time (number of steps) t_2, and the other with $E_{z,1} = E_{z,2}/2$ and time $t_1 = 8t_2$.

In order to assess the impact of nonlinearity of $\epsilon(S^2)$, let us assume that the first nonlinear term b in the expansion $\epsilon = \epsilon(0) + aS^2 + bS^4$ is at least very approximately known.

From (29.22) it follows that the systematic error of ϵ is then

$$\delta_{\mathrm{sys}}\epsilon = -S_1^2 S_2^2 b = -\frac{bS_2^4}{4}. \tag{29.26}$$

Typical values of b are usually aroud $a/2$; for $b \approx a$ and target precision $1\,\%$ we have $S_2 \leq 0.45$. or, to be even more pessimistic, $S_2 \approx 0.4$.

The statistical error of the optimized (29.25) is

$$\delta_{\mathrm{stat}}\epsilon = \frac{3\delta}{t^{1/2}S_2}. \tag{29.27}$$

This error for $S_2 = 0.4$ is $7.5/t^{1/2}$ while for the method without extrapolation ($S = 0.1$) it is $10/t^{1/2}$, i.e., the extrapolation method is almost twice as efficient (half CPU time is sufficient).

The algorithm may be:

1. Run a short simulation with `el.sat` $= S_2 = 0.4$ or less (in dependence on the system) using the autoset feature (`el.sat`) and determine the field E_z.

2. Run a productive simulation with this field E_z.

3. Run an 8-times longer simulation with field $E_z/2$.

4. Extrapolate using (29.22).

Similar optimization formula for the fluctuation method contains values of ϵ_r and χ^∞, which are unknown in advance. In addition, the saturation itself is determined with low precision which makes reliable extrapolation (with error estimation) difficult.

Notes: cook does not print dielectric constant calculations if there are free ions in the system.

Chapter 30

Fourier transform

30.1   Basic formulas

The basic formula in 1D is

$$\tilde{f}(k) = \int_{-\infty}^{\infty} f(x) \exp(-ikx) \mathrm{d}x \tag{30.1}$$

and the inverse transform is

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \tilde{f}(k) \exp(ikx) \mathrm{d}k. \tag{30.2}$$

In 3D

$$\tilde{f}(\vec{k}) = \int f(\vec{r}) \exp(-i\vec{k} \cdot \vec{r}) \mathrm{d}\vec{r}, \quad f(\vec{r}) = \frac{1}{(2\pi)^3} \int \tilde{f}(\vec{k}) \exp(i\vec{k} \cdot \vec{r}) \mathrm{d}\vec{k}. \tag{30.3}$$

If $f(\vec{r})$ is spherically symmetric, then (30.3) becomes

$$\tilde{f}(k) = \int_0^{\infty} r^2 \mathrm{d}r \int_{-1}^1 \mathrm{d}\cos\theta \int_0^{2\pi} f(r) \exp(-i\cos\theta kr). \tag{30.4}$$

After performing the integration over angles

$$k\tilde{f}(k) = 4\pi \int_0^{\infty} \sin(kr) r f(r) \mathrm{d}r \tag{30.5}$$

and back in the same way

$$r f(r) = \frac{1}{2\pi^2} \int_0^{\infty} \sin(kr) k \tilde{f}(k) \mathrm{d}k. \tag{30.6}$$

30.2   Implementation

The Fourier transforms in MACSIMUS (see `gen/fourier.c`, `gen/fft.c` and `gen/fft.h`), are implemented as generalized fast Fourier transform. The number of points n is factored and the algorithm proceeds recursively by taking the prime factors. The cases 2 and 3 are optimized. The efficiency deteriorates for large factors.

In addition to the basic complex Fourier transform, transforms of real, even and odd functions as well as the 3D Fourier transform are available.

30.3   Structure factor

To derive formula (14.24) for S_{IJ}, let us start from the definitions of g_{IJ}:

$$g_{IJ}(r) = V \langle \delta(r_{ij} - r) \rangle \times \begin{cases} 1 - 1/N_I & \text{for } I = J \\ 1 & \text{for } I \neq J \end{cases}, \tag{30.7}$$

where $i(j)$ is any atom of type $I(J)$, respectively, and δ is the Dirac delta-function. By taking the Fourier transform of (30.7) (for $I \neq J$):

$$\int (g_{IJ}(r) \exp(-i\vec{k} \cdot \vec{r}) \mathrm{d}\vec{r} = V \langle \exp(i\vec{k} \cdot \vec{r}_i) \exp(-i\vec{k} \cdot \vec{r}_j) \rangle = V \frac{\langle Q_I^*(\vec{k}) Q_J(\vec{k}) \rangle}{N_I N_J} \tag{30.8}$$

and using (30.11), one gets the partial structure factors

$$S_{IJ}(k) = 1 + N \left[\frac{\langle Q_I^*(\vec{k}) Q_J(\vec{k}) \rangle}{N_I N_J} - 1 \right] \tag{30.9}$$

and finally (14.24).

30.3.1   Mixtures

Formula (14.24) comes from the following 'combining rule' for the 'partial structure factors' S_{IJ}, where uppercase letters I, J index species:

$$S = \sum_I \sum_J w_{IJ} S_{IJ}, \quad w_{IJ} = \frac{N_I b_I N_J b_J}{(\sum_I N_I b_I)^2} \tag{30.10}$$

and N_I is the number of atoms of species I. Note that $\sum_I N_I b_I = \sum_j b_j$.

For the 'partial structure factors' S_{IJ} it holds

$$S_{IJ} = 1 + \frac{N}{V} \int (g_{IJ} - 1) \exp[-2\pi i \vec{k} \cdot \vec{r}] \mathrm{d}r \tag{30.11}$$

Chapter 31

Slab cutoff corrections

MACSIMUS supports two ways how to correct for cutting off the site–site (Lennard-Jones-like) interaction in the slab geometry:

1. Post-processing – cheaper, easier, faster, less accurate: The correction is estimated from the calculated Z-density profiles at the end of the simulation, see Sect. 31.2. The simulation runs with the truncated potential; consequently, the structure (mainly liquid density) of the slab is not corrected.

2. Fourier tranform slab cutoff correction – slower (by a few %), more difficult to set up, more accurate: Integrated forces and energy in the z-direction are added during the simulation; consequently, the structure of the slab is corrected as well and the liquid density is more accurate, See Sect. 31.1.

Additional corrections should be in general taken into account:

1. The stacking correction caused by van der Waals attraction of the z-periodic array of slabs, see Sect. 31.3 and variables `slab.ext.*`.

2. Dipole-dipole attraction of slabs (Yeh–Berkowitz correction), see variable `el.corr`, also see Sect. 15.9.

3. Physical finite-size effects caused by capillary waves.

31.1   Fourier transform slab cutoff correction

This method is available since V2.9b, slab geometry in `configure.sh` or `#define SLAB` in `simopt.h` is required. See also variables `slab.K` and `slab.range`.

31.1.1   Truncated site–site potential

Let $u(r)$ be the interatomic potential at atom-atom separation r. In the simulation, we use a truncated and smoothed (but not shifted) potential $u_{\mathrm{MD}}(r)$,

$$u(r) = u_{\mathrm{MD}}(r) + \Delta u(r), \tag{31.1}$$

where the correction $\Delta u(r) = 0$ for $r < C_1$ and C_1 is the inner cutoff, see Sect. 11.4.

The following formulas are implemented in `sim/xforces.c` and `sim/slabmeas.c`.

31.1.2   One atom: energy correction

For estimating the correction we will approximate the discrete distribution of identical particles by the z-density profile, $\mathcal{N}(z)$, where $z \in [0, L_z]$ and L_x, L_y, L_z are the sides of the rectangular periodic simulation box. The density profile is defined as the x, y-averaged number density so that there are $L_x L_y \int_a^b \mathcal{N}(z)\mathrm{d}z$ particles in interval $z \in [a, b]$. Since we use the periodic boundary conditions, we define $\mathcal{N}(z + nL_z) = \mathcal{N}(z)$ for integer n; i.e., we interpret the system as composed of z-periodic layers. The correction to energy of a particle at point $\vec{r}_i = (x_i, y_i, z_i)$ interacting with the layers is given by the functional

$$S(\mathcal{N}(z); z_i) = \int_0^\infty 2\pi r \mathrm{d}r \int_{-\infty}^\infty \mathrm{d}z\, \mathcal{N}(z) s(r^2 + (z - z_i)^2), \tag{31.2}$$

where $s(r^2) = \Delta u(r)$ and we chose $x_0 = y_0 = 0$ because of the x, y translational invariance. In addition, the lower bound of the first integral can be replaced by $[\max\{C_1^2 - (z - z_i)^2, 0\}]^{1/2}$ because $s(r^2) = 0$ for $r < C_1$.

Let us approximate the z-density profile by the Fourier series up to certain number of waves K. In the complex notation

$$\mathcal{N}(z) = \sum_{k=-K}^K \mathcal{N}_k \mathrm{e}^{2\pi i k z/L_z}, \tag{31.3}$$

where

$$\mathcal{N}_k = \frac{1}{L_z} \int_0^{L_z} \mathrm{d}z\, \mathrm{e}^{-2\pi i k z/L_z} \mathcal{N}(z). \tag{31.4}$$

For a discrete set of N particles (one configuration)

$$\mathcal{N}_k = \frac{1}{V} \sum_{i=1}^N \mathrm{e}^{-2\pi i k z_i/L_z}, \tag{31.5}$$

where $V = L_x L_y L_z$ is the volume.

Correction (31.2) with one density wave $\mathrm{e}^{2\pi i k z/L_z}$ instead of $\mathcal{N}(z)$ can be again approximated by the Fourier series up to K,

$$S(\mathrm{e}^{2\pi i k z/L_z}; z_i) = \sum_{m=-K}^K S_{km} \mathrm{e}^{2\pi i m z_i/L_z}, \tag{31.6}$$

where

$$
\begin{aligned}
S_{km} &= \frac{1}{L_z} \int_0^{L_z} \mathrm{d}z_i\, \mathrm{e}^{-2\pi i m z_i/L_z} S(\mathrm{e}^{2\pi i k z/L_z}; z_i) \\
&= \frac{1}{L_z} \int_0^{L_z} \mathrm{d}z_i\, \mathrm{e}^{-2\pi i m z_i/L_z} \int_0^\infty 2\pi r \mathrm{d}r \int_{-\infty}^\infty \mathrm{d}z\, \mathrm{e}^{2\pi i k z/L_z} s(r^2 + (z - z_i)^2).
\end{aligned}
$$

Let us substitute $z - z_i \to \Delta z$. The Jacobian of this transformation is unity. After rearranging

$$S_{km} = \frac{2\pi}{L_z} \int_{-\infty}^\infty \mathrm{d}\Delta z \int_{\sqrt{\max\{C_1^2 - \Delta z^2, 0\}}}^\infty r \mathrm{d}r s(r^2 + \Delta z^2) \int_0^{L_z} \mathrm{d}z_i\, \mathrm{e}^{2\pi i [k\Delta z + (k-m)z_i]/L_z}. \tag{31.7}$$

The last integral equals L_z for $k = m$ and zero otherwise; thus, only the diagonal terms S_{kk} are nonzero. By substitution $r^2 - (z - z_i)^2 \to 1/u$ (note that the London term $1/r^6$ is transformed to u^3 so that no singularity appears) we get

$$S_{kk} = \pi \int_{-\infty}^\infty \mathrm{d}\Delta z\, \mathrm{e}^{2\pi i k \Delta z/L_z} \int_0^{1/\max\{C_1^2, \Delta z^2\}} \mathrm{d}u\, \frac{s(1/u)}{u^2}. \tag{31.8}$$

It follows from the $\Delta z \to -\Delta z$ symmetry that S_{kk} are real. Thus we can integrate over positive Δz and take twice the real part of the result; in addition, non-negative wave numbers, $k \geq 0$, are enough (so they are for $\mathcal{N}_k$; we do not assume a symmetry of the z-density profile and keep the imaginary parts of $\mathcal{N}_k$, though). Therefore

$$S_{kk} = 2\pi \int_0^\infty \mathrm{d}\Delta z \, \cos(2\pi k\Delta z/L_z) \int_0^{1/\max\{C_1^2, \Delta z^2\}} \mathrm{d}u \, \frac{s(1/u)}{u^2}. \qquad (31.9)$$

The range of the second integral gets narrower as Δz increases and the correction decays fast with the distance; thus, replacing $\int_0^\infty \Delta x$ by $\int_0^{8L_z} \Delta x$ is accurate enough and still fast to evaluate. In some cases even $\int_0^{L_z/2} \Delta x$ may be sufficient, but not for determining the zz component of the pressure tensor for, e.g., vapor–liquid phase equilibria.

Coefficients S_{kk} can be tabulated in advance. The integration over u is implemented via the Gauss formula 4-th order formula, over Δz and z_i by the trapezoidal rule.

At every MD step, $\mathcal{N}_k$ are calculated first. We use the direct evaluation (the only optimization is that $\mathrm{e}^{-2\pi i m z_i/L_z}$ is calculated as powers $[\mathrm{e}^{-2\pi i z_i/L_z}]^m$), but "particle mesh" algorithms known from the k-space sums of the Ewald summations are possible.

The final expression for the energy correction in the complex notation is

$$S(z_i) = \sum_{k=-K}^{K} S_{kk} \mathcal{N}_k \mathrm{e}^{2\pi i k z_i/L_z} \qquad (31.10)$$

which with the complex numbers expanded becomes

$$S(z_i) = S_{00} \mathrm{Re}\mathcal{N}_0 + 2\sum_{k=1}^{K} S_{kk} \left[\mathrm{Re}\mathcal{N}_k \cos(2\pi k z_i/L_z) - \mathrm{Im}\mathcal{N}_k \sin(2\pi k z_i/L_z) \right]. \qquad (31.11)$$

31.1.3 One atom: force correction

The correcting force is simply minus the gradient of $S(z_i)$,

$$f(z_i) = \frac{4\pi}{L_z} \sum_{k=1}^{K} k S_{kk} \left[\mathrm{Re}\mathcal{N}_k \sin(2\pi k z_i/L_z) + \mathrm{Im}\mathcal{N}_k \cos(2\pi k z_i/L_z) \right]. \qquad (31.12)$$

This force also enters the virial of force needed to evaluate the pressure tensor.

31.1.4 Mixture of sites

The extension to mixtures is tedious, but straightforward. Let the number of site types be m. For every correction, we have to calculate m density profiles, with K k-components for each, $\mathcal{N}_k^j$, $j = 1..m$. The constants S_{kk} must be precalculated for all pairs, S_{kk}^{jl}, $j = 1..m$, $l = 1..m$. The correction for atom of type l is (in the complex notation)

$$S^l(z_i) = \sum_{j=1}^{m} \sum_{k=-K}^{K} S_{kk}^{jl} \mathcal{N}_k \mathrm{e}^{2\pi i k z_i/L_z} \qquad (31.13)$$

and analogously for forces.

31.1.5   Total energy correction

The total energy correction is one half (to avoid counting the pair interactions twice) of the sum of the corrections over all atoms,

$$\frac{1}{2} \int_0^{L_z} \mathrm{d}z \mathcal{N}(z) S(z). \tag{31.14}$$

It can be approximated in the k-space by replacing both functions by their (truncated) Fourier expansions. The result is

$$E_{\mathrm{corr}} = \frac{V}{2} \sum_{k=-K}^{K} |\mathcal{N}_k|^2 S_{kk}. \tag{31.15}$$

This result is identical to one half of the sum of (31.10) over all atoms are identical because both formulas are based on the same truncated expansion and either can be used in the computer code.

The extension to mixtures is straightforward,

$$E_{\mathrm{corr}} = \frac{V}{2} \sum_{k=-K}^{K} \sum_{j=1}^{m} \sum_{l=1}^{m} \mathcal{N}_k^j \mathcal{N}_k^{l*} S_{kk}^{jl}, \tag{31.16}$$

31.1.6   Virial of force correction

The virial of force (its zz component) cannot be calculated in the periodic boundary conditions by summing up the terms $z_i f_i$. The contribution to the pressure tensor component can be obtained by scaling the box in the z-direction:

$$P_{zz,\mathrm{corr}} = -\frac{\mathrm{d}E_{\mathrm{corr}}}{L_x L_y \mathrm{d}L_z} = \frac{1}{V} \left(E_{\mathrm{corr}} - \frac{1}{2} \sum_{k=-K}^{K} |\mathcal{N}_k|^2 S'_{kk} \right) \tag{31.17}$$

where

$$S'_{kk} = 2\pi \int_0^\infty \mathrm{d}\Delta z \sin(2\pi k \Delta z / L_z)(2\pi k \Delta z / L_z) \int_0^{1/\max\{C_1^2, \Delta z^2\}} \mathrm{d}u \, \frac{s(1/u)}{u^2}. \tag{31.18}$$

In the x, y directions, the contribution is related to the energy in the same way as for the homogenerous corrections, namely

$$P_{xx,\mathrm{corr}} = -\frac{\mathrm{d}E_{\mathrm{corr}}}{\mathrm{d}L_x L_y L_z} = \frac{1}{V} E_{\mathrm{corr}}. \tag{31.19}$$

The surface tension is (15.23)

$$\gamma = -\frac{3}{4} L_z P_{\mathrm{t}} = -\frac{1}{4} L_z [P_{xx} + P_{yy} - 2P_{zz}]. \tag{31.20}$$

Thus, the terms containing E_{corr} cancel out in P_{t}. The correction is

$$\gamma_{\mathrm{corr}} = \frac{L_z}{4V} \sum_{k=-K}^{K} |\mathcal{N}_k|^2 S'_{kk}. \tag{31.21}$$

The extension to mixtures is straightforward.

Figure 31.1: Post-corrected surface tension, calculated from one configuration of liquid argon, $N = 8000$, $\sigma_{\mathrm{LJ}} = 3.405$ Å, $\epsilon_{\mathrm{LJ}} = 119.8\,\mathrm{K}\,k_{\mathrm{B}}$, in dependence on the cutoff $C_2 = \texttt{LJcutoff}$, see (11.26).

31.2   Post-processed slab corrections

A less accurate (and a bit faster) slab cutoff correction method is bases on a simulation with uncorrected forces. Cutoff corrections are calculated after the simulation from the final z-density profiles by a 2D numerical integration over slabs. This was the only method available prior V 2.9b. Since the structure of the slab (slab width, density at slab center) is affected by the cutoff, this method captures only a part of the correction and does not correct the z-profiles. The protocol is requested by `slab.mode&1`. Note that the correction is over periodically stacked slabs in the z-direction (see the stacking correction below).

Generally, the Fourier transform slab cutoff correction method is preferred.

The homogeneous correction, `corr&3` should be zero with this method; however, since the trajectory is not affected, the correct (back-corrected) results are printed even if you forget to turn off `corr&3`.

The algorithm has been described in [48]. We write the total interatomic potential as

$$u_{ij} = u_{ij,\mathrm{MD}} + \Delta u_{ij}. \tag{31.22}$$

Let $\mathcal{N}_i(z)$ denote the z-density profile of atom (site) i. The correction in the averaged potential energy E (residual internal energy) can be approximated by integration over all (assuming uncorrelated) positions of pairs ij and then by summing over all pairs

$$\Delta E = \sum_{i<j} \frac{\int \mathcal{N}_i(z_i)\mathrm{d}\vec{r}_i \int \mathcal{N}_j(z_j)\mathrm{d}\vec{r}_j \, \Delta u_{ij}(r_{ij})}{\int \mathcal{N}_i(z_i)\mathrm{d}\vec{r}_i \int \mathcal{N}_j(z_j)\mathrm{d}\vec{r}_j}. \tag{31.23}$$

This can be partly integrated and rearranged to a 3-dimensional integral,

$$\Delta E = \sum_{i<j} \frac{\int \mathcal{N}_i(z_i)\mathrm{d}z_i \int \mathcal{N}_j(z_j)\mathrm{d}z_j I_{ij}(|z_i - z_j|)}{L_x L_y \int \mathcal{N}_i(z_i)\mathrm{d}z_i \int \mathcal{N}_j(z_j)\mathrm{d}z_j}, \tag{31.24}$$

where

$$I_{ij}(z) = \int_0^\infty 2\pi r \mathrm{d}r \Delta u_{ij}(\sqrt{r^2 + z^2}) = 2\pi \int_0^{1/\max(c_1,z)} \frac{\mathrm{d}t}{t^3} \Delta u_{ij}(1/t). \tag{31.25}$$

In practical evaluation we first calculate, for all atom type pairs ij, tables $I_{ij}(z)$ by a fourth-order Gaussian integration over $\mathrm{d}t$. The same grid given by `slab.grid` is used for both $I_{ij}(z)$ and $\mathcal{N}_i(z)$. Then the integrals over $\mathrm{d}z_i$ and $\mathrm{d}z_j$ are numerically evaluated by the trapezoidal rule with the same grid as $\mathcal{N}_i(z)$. For determining the surface tension correction, term Δu in the above formulas should be replaced by

$$\Delta u \quad \longrightarrow \quad \frac{L_z}{4V} \frac{x^2 + y^2 - 2z^2}{r} \Delta u'(r) \tag{31.26}$$

and then

$$I_{ij}(z) = \frac{L_z}{4V} 2\pi \int_0^{1/\max(c_1,z)} \frac{\mathrm{d}t}{t^2} [1/t^2 - 3z^2] \Delta u'_{ij}(1/t) \tag{31.27}$$

in (31.24) gives $\Delta\gamma$ instead of ΔE.

The performance of the algorithm is seen in fig. 31.1. The surface tension was calculated from one configuration of liquid argon (thus, the influence of the cutoff on the structure is not taken into account) in the slab geometry, $N = 8000$, box=$(72, 72, 216)$Å, vdW radius=3.822 Å ($\sigma_{\mathrm{LJ}} = 3.405$ Å), 512 histogram bins in the z-range. The correctness of the algorithm is proven. The systematic error is better than 1% for the cutoff less than about 4 Lennard-Jones sigmas; this value depends on the grid used, though.

31.3   Slab stacking corrected

The above corrections apply for an infinite array of z-stacked slabs. However, we are interested in slabs of infinite thickness of both phases. The final correction for the stacking may be estimated from the calculated z-density profiles (post-processing, see below) by comparing two results:

1. The original calculation based on the calculated z-profiles (section in the protocol named "surface tension").

2. A calculation with density profiles extended in the z-direction (both slabs are thicker). See variables `slab.ext.center`, `slab.ext.span`, `slab.ext.zero` and section of the protocol called "EXTENDED SLABS".

Numerical results indicate that the stacking correction is small; nevertheless, greater than the correction explained above.

Fine enough z-histogram grid and long LJ cutoff are neeed for the calculation.

31.3.1   Stacking error for van der Waals forces

Since the stacking error is small, it may be useful to have a simple analytic formula.

Surface tension in simulations

Let us consider N particles in a periodic box of dimensions (L_x, L_y, L_z) in the "slab geometry", i.e., with a liquid slab perpendicular to the z axis. Typically $L_x = L_y$ and the ratio L_z/L_x is about 2–3.

The surface tension is determined by formula (15.23),

$$\gamma = -\frac{3}{4} P_\mathrm{t} L_z, \tag{31.28}$$

where the tangent stress is

$$P_\mathrm{t} = \frac{P_{xx} + P_{yy} - 2P_{zz}}{3} \tag{31.29}$$

and (P_{xx}, P_{yy}, P_{zz}) are the diagonal components of the pressure (stress) tensor. They are calculated from the energy of the box by derivatives over volume, which are interpreted with respect to the desired change of the volume and shape (see also 15.7 and 15.8.4)

$$P_{xx} = \frac{Nk_\mathrm{B}T}{V} - \left(\frac{\mathrm{d}E}{\mathrm{d}V}\right)_{x\text{-scaled}} = \frac{Nk_\mathrm{B}T}{V} - \frac{1}{L_x L_y}\frac{\mathrm{d}E}{\mathrm{d}L_z}. \tag{31.30}$$

In the simulation, energy E is the averaged internal energy (or total energy if velocity-dependent constrained forces are present). The ideal gas (kinetic) parts, $Nk_\mathrm{B}T/V$, cancel out in P_t.

Stacking error

Surface tension should be calculated from one slab of liquid; however, in the slab geometry there is an infinite sandwich of slabs stacked in the z-direction. These slabs attract each other by the van der Waals forces causing the "stacking error". Here we will estimate this error assuming vacuum instead of the gas phase (zero vapor pressure) and homogeneous liquid slab.

Let us consider N identical particles in the box interacting at separations of the slabs only by the van der Waals pair potential

$$u(r) = -Cr^{-6}. \tag{31.31}$$

One particle interacts with an xy-periodic infinite plane containing a smooth distribution of particles of surface density one particle per area $L_x L_y$ by potential

$$u_2(z) = -\frac{C}{L_x L_y}\int_0^\infty \frac{2\pi r\,\mathrm{d}r}{(\sqrt{z^2 + r^2})^6} = -\frac{C}{L_x L_y}\frac{\pi}{2z^4}, \tag{31.32}$$

where z is the distance of the particle from the plane.

Thin slabs

As a reference, let us consider first thin slabs stacked L_z apart. The energy of the mutual interaction of these slabs is (per one box)

$$E_0 = -N^2 \sum_{n=1}^\infty u_2(nL_z) = -\frac{N^2 C}{L_x L_y}\frac{\pi^5}{180 L_z^4}. \tag{31.33}$$

The sum is over positive subscripts only to avoid including the pair interactions twice.

The corresponding pressure contribution in the z-direction is obtained by the virtual scaling of the box,

$$P_{z0} = -\frac{dE_0}{dV} = -\frac{1}{L_x L_y}\frac{dE_0}{dL_z} = \frac{\pi^5 C N^2}{45 L_z^5 L_x^2 L_y^2}, \tag{31.34}$$

where the ideal gas (kinetic) part is omitted. Components in the x and y directions are obtained in the same way. From (31.28) we get the correction (o be added)

$$\Delta\gamma_0 = -\frac{\pi^5 C}{120}\frac{N^2}{L_z^4 L_x^2 L_y^2} = -\frac{A\pi^3}{120 L_z^2}, \tag{31.35}$$

where A is the Hamaker constant of the simulation box assumed to be homogeneous (all $-C/r^6$ terms smeared out).

Thick slabs

We assume that the thickness of a liquid slab is s and that it contains N uniformly and smoothly distributed particles. The interaction energy between slabs (excluding any interaction within a single slab) is

$$E = \sum_{i=1}^{\infty}\int_{iL_z}^{iL_z+s}dz_2\int_0^s dz_1\frac{N}{s}u_2(z_2-z_1) = E_0 f(s/L_z), \tag{31.36}$$

where function f is

$$f(x) = \frac{15\pi^2 x^2\cot^2(\pi x) + 10\pi^2 x^2 - 15}{\pi^4 x^4}. \tag{31.37}$$

Note that $\lim_{x\to 0} f(x) = 1$ and $f(0.5) = 1.589$.

The corresponding surface tension is

$$\Delta\gamma = \Delta\gamma_0 f(s/L_z) \tag{31.38}$$

with the same function f because the derivatives in (31.30) are performed at $x = s/L_z$ constant (the whole configuration is scaled). The above correction is negative (the attraction of slabs decreases the apparent surface tenson); $-\gamma$ (positive number) should be added to the simulation-based value to get the corrected result.

The correction to the vapor pressure is then

$$\Delta P_{zz} = \frac{2\Delta\gamma}{L_z}. \tag{31.39}$$

Algorithm test

As a fool-proof test of the algorithm, we compared the above formulas with the corections calculated using the post-processing calculation from extended z-profiles using a Lennard-Jones crystal[1]. We used the Lennard-Jones argon ($\sigma_{LJ} = 3.405$ Å, $\epsilon_{LJ} = 119.8$ K k_B) and a slab of the fcc crystal with 4 atoms in a unit cell of side 5.2488179 Å. The box was $(10a, 10a, n_z a)$ with the slab $20a$ thick. The cutoff was $C_2 = 20$ Å (note that the slabs are farther away than C_2). The results are:

---

[1] Of course, the surface tension of a crystal cannot be calculated from the tangent stress; however, the corrections can.

n_z	$x = s/Lz$	$-E/\mathrm{J\,mol^{-1}}$		$-\gamma/\mu\mathrm{N\,m^{-1}}$	
		eq. (31.36)	crystal	eq. (31.38)	crystal
43	0.465	139.7071	138.518	12.631	12.522
68	0.294	17.4346	17.247	1.5763	1.5584
136	0.147	0.9776	0.965	0.0884	0.0871

Chapter 32

Correcting the angular momentum

The linear momentum is exactly conserved (if the symmetry allows) with any finite-difference integrator, it is thus subject to rounding errors only. In contrast, the angular momentum is conserved (in the free boundary conditions) subject to numerical errors proportional to some power of the timestep (h^2 with Verlet). In further text we assime that the configuration is centered to the center of mass ($\vec{r}_{\text{center of mass}} = 0$).

If a body or a configuration rotates with angular velocity $\vec{\omega}$, then the velocity of atom i is

$$\vec{v}_i = \vec{\omega} \times \vec{r}_i. \tag{32.1}$$

The angular momentum is

$$\vec{M} = \sum_i \vec{r}_i \times m_i \vec{v}_i. \tag{32.2}$$

By inserting for $\vec{v}_i$ from (32.1) we get

$$\vec{M} = \sum_i \vec{r}_i \times m_i(\vec{\omega} \times \vec{r}_i) = \sum_i m_i \left[r_i^2 \vec{\omega} - r_i(r_i \cdot \vec{\omega}) \right] = \overset{\leftrightarrow}{I} \cdot \omega, \tag{32.3}$$

where $\overset{\leftrightarrow}{I}$ is the inertia tensor (see also (2.2))

$$\overset{\leftrightarrow}{I} = \sum_i m_i (\vec{r}_i^2 \overset{\leftrightarrow}{\delta} - \vec{r}_i \vec{r}_i), \tag{32.4}$$

$\overset{\leftrightarrow}{\delta}$ denotes the unit tensor and $\vec{r}_i \vec{r}_i$ is the tensor (outer) product of both vectors.

To set the angular momentum to zero, $\vec{M}$ is calculated first from (32.2), then the inertia tensor $\overset{\leftrightarrow}{I}$ from (32.4). The velocities are corrected by

$$\vec{v}_i := \vec{v}_i - \vec{\omega} \times \vec{r}_i, \quad \text{where } \vec{\omega} = \overset{\leftrightarrow}{I}^{-1} \cdot \vec{M}. \tag{32.5}$$

For the implementation, see variable `drift`.

Note: The kinetic energy of rotation is

$$E_{\text{kin rot}} = \frac{1}{2} \vec{\omega} \cdot \overset{\leftrightarrow}{I} \cdot \vec{\omega} = \frac{1}{2} \vec{\omega} \cdot \vec{M} = \frac{1}{2} \vec{M} \cdot \overset{\leftrightarrow}{I}^{-1} \cdot \vec{M}$$

Chapter 33

Electronic continuum correction (ECC)

See [49] and `notes.pdf`, see also scripts `U.sh` and `P.sh`.

Applies if compiled with `#define ECC`. See quantities `el.ecc` and `el.epsf`.

The ECC potential energy and the corresponding pressure are divided into three parts:

$$U = U_1 + U_2 + U_3, \qquad P = P_1 + P_2 + P_3, \tag{33.1}$$

U_1, P_1 includes all non-electrostatic forces (Lennard-Jones) and for pressure also the kinetic contributions and with constraint dynamics (SHAKE) also the virial of constraint forces.

U_2, P_2 comes from $\partial U_2/\partial V$, where U_2 are all pair interactions (including Ewald k-space terms) of scaled charges including the dependence of `epsf` on volume.

P_3 is the Born (for charge) or Lorentz (for dipole) solvation energy (in radius R calculated from system density; again, $\partial R/\partial V$ is taken into account).

Since cook version V3.4a, the following quantities are printed:

`ECC Epot` The ECC potential energy $U = U_1 + U_2 + U_3$

`ECC Uscaled` The conventional energy $U_1 + U_2$.

`ECC PLJ` P_1
This includes the kinetic + Lennard-Jones (or similar) terms. The kinetic terms include the ideal gas contribution, $Nk_{\mathrm{B}}T/V$, where N is the number of real sites (not dependants). For models with constraints, the pressure caused by the virial of constraint forces, (15.12), is included here, too.

`ECC P2corr` $P_2 - \overline{P}_2$, where $\overline{P}_2 = U_2/3V$.
This is the correction to the formula based on sites (therefore, the virial of constraint forces should be included).

`ECC P2full` P_2 calculated by the virial theorem.
It means that $P_2 = U_2/3V$ based on ion-ion scaling r^{-1} and scaling $P_2 = U_2/V$ based on dipole–dipole r^{-3} scaling is included (cannot combine dipoles and ions). For atomic ions and dipoles, this equals the true P_2. For finite-size ions and dipoles with fixed bonds, this term becomes inappropriate and the corresponding ECC approximation inconsistent, see `ECC P`.

ECC P3full P_3

ECC P3corr $P_3 - \overline{P}_3$, where $\overline{P}_3 = U_3/3V$

ECC P3ref $\overline{P}_3 = U_3/3V$

ECC Pvir Pressure literally from the virial theorem, but without dependence of the terms on volume: $P_{\mathrm{vir}} = P_1 + \overline{P}_2 + \overline{P}_3$.[1]

ECC P The ECC pressure $P = P_1 + P_2 + P_3$ with all corrections, calculated as `Pvir + P2corr + P3corr`. For point ions, this is the same as `PLJ + P2full + P3full`.

ECC Pscaled The conventional pressure with scaled charges, but no `epsf`-dependence taken into account. The same as `Pvir - P3ref`.

Since cook V3.4a, the ECC pressure is used internally for pressure. However, variable box (as in NPT) is not fully supported because `el.epsf` is not recalculated when volume changes. (This is likely easily doable, but I do not need it now).

In addition, other quantitites subject to the electronic continuum correction are not corrected. Therefore, electric conductivity and dielectric constant have to be multiplied by `el.epsf`. This applies to versions V3.4a or older.

---

[1]Prior V3.4a, this was called (misleadingly) `P` (the usual MACSIMUS pressure).

References

[1] M. L. Bjerking: Grafisk brugerinterface til et Molekyle-dynamik (MD) program, *MSc. thesis*, Odense University (1993).

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery:Numerical Recipes in FORTRAN, Cambridge University Press (1992), Chap. 10.6.

[3] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus: CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, *J. Comput. Chem.* **4**, 187, (1983).

[4] Jorgensen et al.: *J. Chem. Phys.* **79**, 926 (1983); Jorgensen et al.: *Molec. Phys.* **56**, 1381 (1985).

[5] A.D. MacKerell et al.: All-atom empirical potential for molecular modeling and dynamics studies of proteins, *J. Phys. Chem B* **102**, 3586–3616 (1998).

[6] Briggs J.M, Matsui T., Jorgensen W.L.: Monte Carlo Simulations of Liquid Alkyl Ethers with the OPLS Potential Functions, *J. Comput. Chem.* **11**, 958–971 (1990).

[7] *QUANTA Parameter Handbook*, Polygen Corporation (1990).

[8] G. J. Martyna, D. J. Tobias, M. L. Klein: Constant pressure molecular dynamics algorithms, *J. Chem. Phys.* **101**, 4177–4189 (1994).

[9] C. Yeh, M.L. Berkowitz, *J. Chem. Phys.* **111**, 3155 (1999).

[10] V. Ballenegger, A. Arnold, J. J. Cerdà, *J. Chem. Phys.* **131**, 094107 (2009).

[11] J. Kolafa, J.W. Perram: Cutoff errors in the Ewald summation formulae for point charge systems, Mol. Simulat. **9** 351-368 (1992).

[12] K. D. Hammonds, D. M. Heyes: Optimization of the Ewald method for calculating Coulomb interactions in molecular simulations, *J. Chem. Phys.* **157**, 074108 (2022).

[13] J. Janek, J. Kolafa: Author = Novel Gear-like predictor-corrector integration methods for molecular dynamics, *Mol. Phys.* **118**, e1674937 (2020).

[14] M. Tuckerman, Y. Liu, G. Ciccotti, G. Martyna: Non-Hamiltonian molecular dynamics: Generalizing Hamiltonian phase space principles to non-Hamiltonian systems, *J. Chem. Phys.* **115**, 1678 (2001).

[15] H. C. Andersen: Molecular dynamics simulations at constant pressure and/or temperature, *J. Chem. Phys.* **72**, 2384 (1980).

[16] N. Grønbech-Jensen, O. Farago: A simple and effective Verlet-type algorithm for simulating Langevin dynamics (2013 preprint).

[17] G. Bussi, D. Donadio, M. Parrinello: Canonical sampling through velocity rescaling, *J. Chem. Phys.* **126**, 014101 (2007).

[18] S. W. de Leeuw, J. W. Perram, H. G. Petersen: Hamilton equations for constrained dynamic-systems, *J. Stat. Phys.* **61**, 1203–1222 (1990).

[19] S. Nosé: *Mol. Phys.* **52**, 255 (1984).

[20] W.G. Hoover: *Phys. Rev. A* **31**, 1695 (1985).

[21] G. J. Martyna, M. E. Tuckerman, D. J. Tobias, M. L. Klein: Explicit reversible integrators for extended systems dynamics, *Mol. Phys.* **87**, 1117–1157 (1995).

[22] W. G. Hoover: *Phys. Rev. A* **31**, 1695 (1985).

[23] B. Dünweg, K. Kremer: *J. Chem. Phys.* **99**, 6093 (1994).

[24] In-Chul Yeh, G. Hummer: System-Size Dependence of Diffusion Coefficients and Viscosities from Molecular Dynamics Simulations with Periodic Boundary Conditions, *J. Phys. Chem. B* **108**, 15873-15879 (2004).

[25] J. Malohlava (University of Ostrava) and J. Kolafa (2010), unpublished results.

[26] Daivis P.J., Evans D. J.: Comparison of constant pressure and constant volume nonequilibrium simulations of sheared model decane, *J. Chem. Phys.* **100**, 541 (1993).

[27] M. P. Allen: Back to Basics, in *Computer Simulation in Chemical Physics*, ed. M.P. Allen and D.J. Tildesley, Proceedings of the NATO Advanced Study Institute on New Perspectives in Computer Simulation in Chemical Physics, Alghero, Sardinia, Italy, September 14-24 (1992).

[28] Bedrov D., Smith G. D.: Temperature-dependent shear viscosity coefficient of octahydro-1,3,5,7-tetranitro-1,3,5,7-tetrazocine (HMX): A molecular dynamics simulation study, *J. Chem. Phys.* **112**, 7203 (2000).

[29] S. G. Moustafa, A. J. Schultz, D. A. Kofke: Harmonically Assisted Methods for Computing the Free Energy of Classical Crystals by Molecular Simulation: A Comparative Study, *J. Chem. Theory Comput.* **13**, 825 (2016).

[30] J. Kolafa, F. Moučka, I. Nezbeda: Handling Electrostatic Interactions in Molecular Simulations: A Systematic Study *Collect. Czech. Chem. Commun.* **73**, 481–506 (2008).

[31] Z. Akdeniz, G. Pastore, M.P. Tosi: *Phys. Chem. Liq.* **32**, 191 (1996); Z. Akdeniz, G. Pastore, M.P. Tosi: *Nuovo Cimento*, **20**, 595 (1998).

[32] J. Applequist, J. R. Carl, K.-K. Fung: *J. Am. Chem. Soc.* **94**, 2952 (1972).

[33] A. Aguado, P.A. Madden: *J. Chem. Phys.* **119**, 7471 (2003).

[34] G.J. Gloor, G. Jackson, F.J. Blas, E. de Miguel: Test-area simulation method for the direct determination of the interfacial tension of systems with continuous or discontinuous potentials, *J. Chem. Phys.* **123**, 134703 (2005).

[35] R. L. Davidchack, B. B. Laird: Direct calculation of the crystal–melt interfacial free energies for continuous potentials: Application to the Lennard-Jones system, *J. Chem. Phys.* **118**, 7651 (2003).

[36] F. Cleri, V. Rosato: Tight-binding potentials for transitional metals and alloys, *Phys. Rev. B* **48**, 48–33 (1883).

[37] M. P. Allen and D. J. Tildesley, *The Computer Simulation of Liquids*, 1987 (Clarendon).

[38] Baranyai A., Kiss P. T.: A transferable classical potential for the water molecule, *J. Chem. Phys.* **133**, 144109 (2010).

[39] S. W. Rick, S. J. Stuart, B. J. Berne: *J. Chem. Phys.* **101**, 6141 (1994).

[40] J. Kolafa, M. Lísal: Time-Reversible Velocity Predictors for Verlet Integration with Velocity-Dependent Right-Hand Side, *J. Chem. Theory Comput.* **7**, 3596–3607 (2011).

[41] J. Kolafa: Numerical integration of equations of motion with a self-consistent field given by an implicit equation, *Mol. Simulat.* **18**, 193-212 (1996)

[42] J. Genzer, J. Kolafa: Molecular dynamics of potential models with polarizability: comparison of methods, *J. Mol. Liq.* **109** 63–72 (2004)

[43] J. Kolafa: Time-Reversible Always Stable Predictor-Corrector Method for Molecular Dynamics of Polarizable Molecules, *J. Comput. Chem.* **25** 335–342 (2004)

[44] J. Kolafa: Gear formalism of the always stable predictor-corrector method for molecular dynamics of polarizable molecules, *J. Chem. Phys.* **122** 164105 (2005)

[45] S. W. de Leeuw, J. W. Perram, E.R. Smith: Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. *Proc. R. Soc. Lond. A* **373**, 27–56 (1980).

[46] A. D. Buckingham: A Theory of the Dielectric Polarization of Polar Substances, *Proc. R. Soc. London A* 238, 235–244 (1956).

[47] G. Lamoureux, A.D. MacKerell, Jr., B. Roux: *J. Chem. Phys.* **119**, 5185 (2003).

[48] J. Picalek, B. Minofar, J. Kolafa, P. Jungwirth: *Phys. Chem. Chem. Phys.* **10**, 5765–5775 (2008).

[49] J. Kolafa: Pressure in Molecular Simulations with Scaled Charges. 1. Ionic Systems, *J. Phys. Chem. B*, 124, 7279–7390 (2020).

[50] J. Janek, J. Kolafa: Novel barostat implementation for molecular dynamics: *J. Chem. Phys.* submitted.

[51] C.J. Fennell., J. Gezelter: Is the Ewald summation still necessary? Pairwise alternatives to the accepted standard for long-range electrostatics. *J. Chem. Phys.* **124** 234104 (2006).

Index