gmq

THE USER MANUAL

Version 6

by

David Brown

LEPMI (GUIDE), Université Savoie Mont Blanc, Campus Scientifique,

73376 Le Bourget du Lac, France.

1. Introduction

This document provides a guide to the use of the molecular dynamics (MD) program *gmq* and its implementation for parallel computers, *ddgmq*. It is not a comprehensive description of the technique of molecular dynamics, standard texts being available for that purpose [1-3]. The two programs are, to a very large extent, equivalent in functionality and operation. This manual also includes details of the ancillary analysis and manipulation programs available. As the programs are under constant revision it is essential that the appropriate version of the manual is used with the corresponding program revision.

The program *gmq* was principally designed for MD simulation of dense materials in three dimensions (3D) with periodic boundary conditions. The shape and size of the primary MD box is defined by the 3×3 matrix **h** made up from the three basis (column) vectors {**a**, **b**, **c**} which allows for non-orthorhombic cells. These vectors can be treated as fixed (constant volume) or as dynamic variables to allow changes in the size and shape of the primary MD box. The changes in box size and shape can be in response to differences between the internally measured pressure tensor and the externally required pressure tensor or simply to a desired user-specified change of the basis vectors.

Given the boundary conditions and various other parameters the program integrates the equations of motion of a set of particles (point masses) subject to a rudimentary force field and in the possible presence of a number of rigid constraints. The particles ("atoms") can be linked in an arbitrary way to form entities ("molecules") of varied complexity. Connections are allowed between an atom and the periodic image of another atom thus allowing simulations of "infinite" chains, e.g. in polymer crystals, or connected lattices. The program is best suited for flexible and semi-flexible models of molecules. It treats completely rigid molecules as a collection of atoms held at fixed relative distances apart by a number of rigid constraints. For technical reasons this precludes certain geometries.

2. Force Field

2.1. "Non-bonded" potentials

In *gmq* molecules are modelled as a number of atoms connected by a network of bonds. All atoms in one molecule interact with all atoms in another through, what is referred to as, a non-bonded pair potential. That is to say, any two atoms on different molecules will experience a force between them independent of all the other atoms present. In reality, this is not truly the case but expedience dictates that this is an approximation that is almost universally made in classical MD. The main purpose of the non-bonded potential is usually to exclude two atoms from the same region of space (repulsive interaction). Attractive terms in the potential add the 'glue' that, in the absence of containing walls, hold liquid and solid systems together and influence the preferred packing structure of the molecules to a greater or lesser extent depending on their exact nature.

The most commonly utilised non-bonded pair potentials in *gmq* are the Lennard-Jones (LJ) 12-6 potential:

$$\Phi_{\mathrm{LJ}}\left(\left|\vec{\boldsymbol{r}}_{ij}\right|\right)=4\varepsilon\left(\left(\frac{\sigma}{\left|\vec{\boldsymbol{r}}_{ij}\right|}\right)^{12}-\left(\frac{\sigma}{\left|\vec{\boldsymbol{r}}_{ij}\right|}\right)^{6}\right), \tag{2.1.1}$$

the Weeks-Chandler-Andersen (WCA) potential:

$$\Phi_{\mathrm{WCA}}\left(\left|\vec{\boldsymbol{r}}_{ij}\right|\right)=4\varepsilon\left(\left(\frac{\sigma}{\left|\vec{\boldsymbol{r}}_{ij}\right|}\right)^{12}-\left(\frac{\sigma}{\left|\vec{\boldsymbol{r}}_{ij}\right|}\right)^{6}\right)+\varepsilon \qquad \text{for } \left|\vec{\boldsymbol{r}}_{ij}\right|\le 2^{\frac{1}{6}}\sigma \tag{2.1.2a}$$

$$\Phi_{\mathrm{WCA}}\left(\left|\vec{\boldsymbol{r}}_{ij}\right|\right)=0 \qquad \text{for } \left|\vec{\boldsymbol{r}}_{ij}\right|\le 2^{\frac{1}{6}}\sigma \tag{2.1.2b}$$

the Buckingham exp-6 potential:

$$\Phi_{\mathrm{Buck}}\left(\left|\vec{\boldsymbol{r}}_{ij}\right|\right)=A\exp\left(\frac{-\left|\vec{\boldsymbol{r}}_{ij}\right|}{B}\right)-\frac{C}{\left|\vec{\boldsymbol{r}}_{ij}\right|^{6}} \tag{2.1.3}$$

and the Coulombic potential

$$\Phi_{\mathrm{C}}\left(\left|\vec{\boldsymbol{r}}_{ij}\right|\right)=\frac{q_i q_j}{4\pi\varepsilon_0\left|\vec{\boldsymbol{r}}_{ij}\right|} \tag{2.1.4}$$

In the above equations the pair of interacting atoms, $\{i,j\}$, have a separation vector denoted by $\vec{\boldsymbol{r}}_{ij}\left(=\vec{\boldsymbol{r}}_i-\vec{\boldsymbol{r}}_j\right)$. In Eq. 2.1.1, and the other LJ forms, ε is the well-depth of the potential and σ is the distance at which the potential is zero. The WCA potential given in Eq. 2.1.2 is simply the LJ 12-6 potential truncated at the minimum and raised by the well depth to give a short range purely repulsive potential that decays smoothly to zero. In Eq. 2.1.4, q_i and q_j are the partial charges on the two atoms and ε_o is the permittivity of the vacuum.

In addition to the above, customised versions of *gmq* and *ddgmq* can be generated which use the generalised LJ N-M form of potential,

$$\Phi_{\mathrm{LJ}}\left(\left|\vec{\boldsymbol{r}}_{ij}\right|\right)=\left(\frac{1}{(M/N)^{(M/(N-M))}-(M/N)^{(N/(N-M))}}\right)\varepsilon\left(\left(\frac{\sigma}{\left|\vec{\boldsymbol{r}}_{ij}\right|}\right)^{N}-\left(\frac{\sigma}{\left|\vec{\boldsymbol{r}}_{ij}\right|}\right)^{M}\right) \tag{2.1.5}$$

with N and M positive integers and $N > M > 1$, e.g. the 9-6 form is a common alternative. There exists equally the equivalent generalised *N-M* WCA form of these potentials.

An inherent problem with the form of the Buckingham potential, Eq. 2.1.3, is that the initial exponential term tends to the finite value A at zero separation, whereas the attractive part tends to minus infinity. An illustration of this is given in Fig. 2.1 for the case of the Si···O interaction in the Tsuneyuki [4] model of silica. The potential passes by a maximum at a separation of ~1 Å. Generally this is not usually a problem in molecular dynamics so long as initial distances between particles are greater than the

maximum in the potential. As can be seen in Fig. 2.1, energies are very high in the region of the maximum with respect to k_BT, at least for ambient temperatures. At high temperatures, however, it can happen that particles get too close in MD and access the infinite negative hole in the Buckingham potential. This causes algorithmic breakdown as the magnitude of the forces become extremely high. Problems can also exist for initial configurations, where considerable overlaps can be present due the generation procedure, or in Monte Carlo algorithms. For this reason a "no hole" option has been developed for the Buckingham potential. The normal Buckingham potential is used up to the point (r_{max}) where the force ($F(r)=-d\Phi(r)/dr$) passes through a maximum (F_{max}) but for all values of $r < r_{max}$, the "no hole" force is given by

$$F(r) = F_{max}(r_{max}/r)^{13} \tag{2.1.6}$$

and thus the "no hole" potential becomes

$$\Phi(r) = \Phi(r_{max}) + F_{max}(r (r_{max}/r)^{13} - r_{max})/12 \tag{2.1.7}$$

The force and the potential in this case are shown as dotted lines in Fig. 2.1.

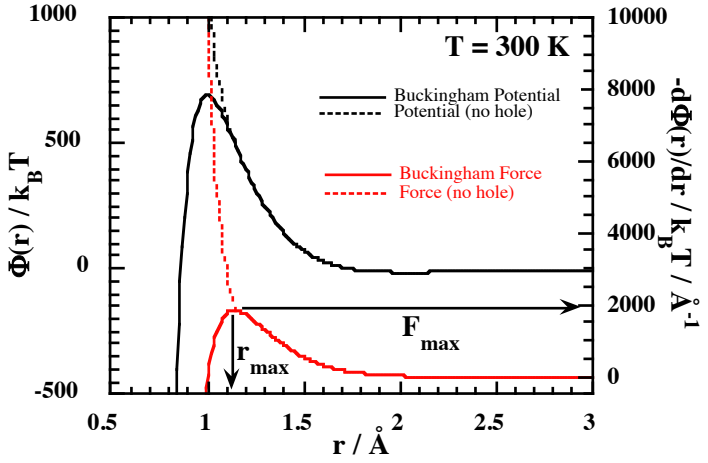


Fig. 2.1 *The Buckingham potential and force for the Si···O interaction in the Tsuneyuki [4] model of silica compared to those for the "no hole" option.*

In what follows, the LJ, WCA and Buckingham potentials will be referred to as Van der Waals (VdW) interactions. In practice only one type of VdW potential can be used at any one time. All can be used in conjunction with the Coulomb potential.

In principle, both the VdW and Coulombic potentials imply that particles interact whatever their distance apart. In practice, the functional form of the VdW potentials means that the interaction energy reaches relatively small values within short distances and it is customary to ignore the forces between atoms further apart than some cut-off distance, r_c. For this reason the VdW potential is often referred to as being 'short-range'. In *gmq* standard estimations are made of the contribution to the energy and the pressure of the VdW interactions beyond the cutoff [1]. These long-range corrections are made on the

basis that beyond r_c the radial distribution function, i.e. the ratio of the mean number of particles between r and r+Δr centred around a subject particle to that expected from a random distribution, is equal to one.

On the other hand the Coulombic potential diminishes slowly with range and is often referred to as a 'long-range' potential; although it has a very significant influence at short range too. From several studies already conducted [5-8], it is very clear that approximate techniques that either simply truncate the 1/r potential or force it to zero using a switching function lead to drastic unphysical effects that only disappear when the non-negligible long range part of the Coulomb potential is handled in a proper manner. In *gmq* the Ewald summation method [9, 10] is used and this will be described in more detail later.

Now, atoms within the same molecule may also be interacted in the same way as described above but this depends on if they are deemed to be 'non-bonded' or 'bonded' interactions. In *gmq* there are two possible definitions of non-bonded pairs within the same molecule: atoms separated by more than three bonds ("1···5" interactions and above) or atoms separated by more than two bonds ("1···4" interactions and above). The choice depends on the parametrization of the torsional energy and this will be explained later.

2.2. Bonds

In *gmq* atom pairs forming a chemical bond can be kept at a fixed distance apart by a rigid constraint

$$\left|\vec{r}_{ij}\right|^2 - b_0^2 = 0 \tag{2.2.1}$$

or, alternatively, quite close to it by a harmonic spring potential

$$\Phi_b\left(\left|\vec{r}_{ij}\right|\right) = \frac{1}{2}k_b\left(\left|\vec{r}_{ij}\right| - b_0\right)^2 \tag{2.2.2}$$

where for a bonded pair of atoms, *i* and *j*, k_b is the force constant and b_o the equilibrium bond length. To maintain a low amplitude motion though, the force constant has to be quite high and this in turn implies high frequency motions. High frequency vibrations lead to a short time step in the integration of the equations of motion of the atoms and can lead to difficulties with equipartition of energy if the frequency is so high as to be largely uncoupled to other motions in the system. For these reasons, it is recommended that bonds are constrained rigidly.

2.3. Bending potentials

To maintain bond angles close to their equilibrium values, the following bending potentials is generally used in *gmq*

$$\Phi(\theta) = \frac{1}{2}\mathrm{k}_\theta(\cos\theta - \cos\theta_0)^2 \tag{2.3.1}$$

where k_θ is a constant with units of energy determining the flexibility of the angle and θ_0 is the equilibrium bond angle. The angle θ is a function of the positions of the three contiguous atoms, i.e. {*i*, *j*, *k*} with *i* bonded to *j* and *j* bonded to *k* and *i*≠*k*, in the molecule

$$\cos\theta = \frac{\left(\vec{r}_{ij} \cdot \vec{r}_{kj}\right)}{\left|\vec{r}_{ij}\right|\left|\vec{r}_{kj}\right|} \tag{2.3.2}$$

Equation 2.3.1 has computational advantages over the popular harmonic form,

$$\Phi(\theta) = \frac{1}{2} k_{\theta}^{'} (\theta - \theta_0)^2, \tag{2.3.3}$$

in that no expensive inverse cosine function calls have to be made. However, although the potential is well behaved around θ=π, where the form of Eq. 2.3.3 introduces a singularity in the force due to a $(\sin\theta)^{-1}$ term appearing, there is a problem with the restoring force produced. For a value of $\theta_0 = \pi$ the curvature, $\frac{d^2\phi}{d\theta^2}$, at the minimum is zero. This leads to a "flattening" of the potential around the minimum as demonstrated in Fig. 2.2.

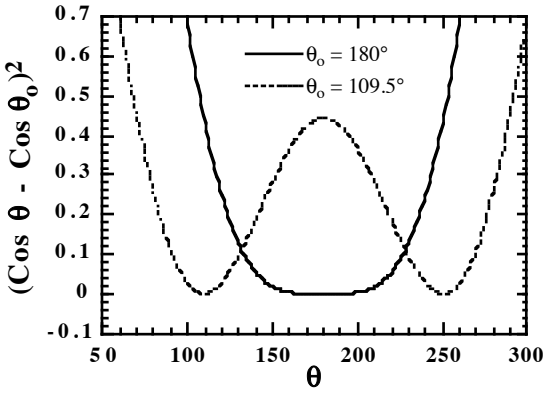


Fig. 2.2 The form of Eq. 2.3.1 for values of θ_0= 109.5 and 180 degrees.

To partly overcome this problem an alternative potential is offered for bends having a minima close to π [11]. The general form taken is the follows:

$$\Phi(\theta) = \frac{1}{2} k_{\theta} \left(1 - \cos\left(\theta - \theta_0\right)\right) \tag{2.3.4}$$

For the case where $\theta_0 = \pi$ this simplifies to

$$\Phi(\theta) = k_{\theta} \left(1 + \cos\theta\right) \tag{2.3.5}$$

which is free of singularities in the force. However, for all other values of θ_0 there is a problem in that the potential in Eq. 2.3.4 also has a singularity in the force at $\theta = \pi$. This is handled in an *ad hoc* way in the code simply by setting the force to zero should ever $\sin\theta = 0$. As this is not very satisfactory using this form of potential with anything other than $\theta_0 = \pi$ is discouraged.

In practice, to specify that the second form of bending potential is to be used for a given type of angle, a negative value should be input for k_θ in the `PARAM` file. The negative sign simply serves as a

switch and the absolute value of k_θ will be used internally (see the section concerning the format of the `PARAM` file).

2.4. Rigid bends

2.4.1 *Linear rigid bends*

Near-linear triatomic molecules, such as carbon dioxide, can pose problems for classical MD simulations. If we represent such a molecule as three mass points then it should have 9 degrees of freedom. If we remove the bond stretching modes then this reduces the number to 7; 3 translations, 3 rotations, and an angle bending mode. Two of the rotations are around axes more or less perpendicular to the long axis of the molecule and are well-defined. The third rotation though has an axis parallel to the long axis of the molecule and consequently a very low, and variable, moment of inertia. This leads to a poor coupling of this mode with the others and great difficulties are encountered achieving equipartition using such models. Such near-linear triatomic molecules are thus better modelled as rigid rods and this can be done in *gmq* using a combination of bond constraints and a special vector constraint.

Consider three collinear atoms *i*, *j* and *k*. A true vector collinear constraint forces the position of the atom *j* to be expressed in terms of the vector from *i* to *k* in the following form

$$\vec{\boldsymbol{r}}_j = \vec{\boldsymbol{r}}_i + \gamma(\vec{\boldsymbol{r}}_k - \boldsymbol{r}_i) = \vec{\boldsymbol{r}}_i + \gamma\vec{\boldsymbol{r}}_{ki} \tag{2.4.1}$$

where γ is an arbitrary scalar. This corresponds to the removal of two degrees of freedom as the value of γ can vary with time, i.e. *j* retains one degree of freedom as it can slide along the vector from *i* to *k*. The vector constraint thus takes the following form

$$\vec{\boldsymbol{\sigma}} = \vec{\boldsymbol{r}}_i - \vec{\boldsymbol{r}}_j + \gamma\vec{\boldsymbol{r}}_{ki} = \vec{\boldsymbol{r}}_{ij} + \gamma\vec{\boldsymbol{r}}_{ki} = \vec{\boldsymbol{0}} \tag{2.4.2}$$

The value of γ can be calculated from the fact that the constraint forces act perpendicular to the $\vec{\boldsymbol{r}}_{ki}$ vector.

A criterion for convergence can be written for each cartesian direction, e.g.

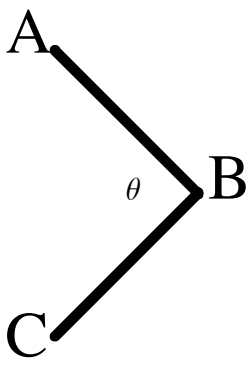
$$\frac{\left|x_{ij} + \gamma x_{ki}\right|}{\left|\vec{\boldsymbol{r}}_{ki}\right|} < \text{TOLER} \tag{2.4.3}$$

where TOLER is the user-defined required relative tolerance.

In practice, the absence of a bending force constant and an equilibrium angle of 180° in the `PARAM` file signals the presence of a linear rigid bend (See Section 5.2).

2.4.2 *Non-linear rigid bends*

Consider three non-collinear atoms A, B and C



In certain circumstances it is useful to keep the ABC angle fixed at the required value of θ. Assuming that the A-B and B-C bond lengths are constrained to fixed values using a regular bond constraint, there are 9 - 2 =7 degrees of freedom as two are removed by the rigid bonds. If we freeze the angle too then we remove one more degree of freedom and we are left with three translational degrees of freedom and three rotational degrees of freedom. The cosine of the ABC angle can be written as

$$\cos\theta = \frac{\vec{\boldsymbol{r}}_{AB}}{d_{AB}} \bullet \frac{\vec{\boldsymbol{r}}_{CB}}{d_{CB}} \tag{2.4.4}$$

The corresponding constraint relationship is then [12]

$$\sigma = \cos\theta - \frac{\vec{\boldsymbol{r}}_{AB}}{d_{AB}} \bullet \frac{\vec{\boldsymbol{r}}_{CB}}{d_{CB}} = 0 \tag{2.4.5}$$

and a convergence criterion for a rigid angle constraint can be written as

$$\left| \cos\theta - \frac{\vec{\boldsymbol{r}}_{AB}}{d_{AB}} \bullet \frac{\vec{\boldsymbol{r}}_{CB}}{d_{CB}} \right| < \text{TOLER} \tag{2.4.6}$$

In general it is not recommended to use rigid constraints on the non-linear bends in chain molecules due to the bias that this introduces into the torsional degrees of freedom [13]. However, they can be used in certain cases where hydrogens are present and their degrees of freedom cannot be removed using special constraints (see Section 2.7) and thus the usual problems of short time steps and equipartition of kinetic energy can be avoided. A good example is the rigid 3-site SPCE model of water where tests have shown that replacing the fictive H-H bond by a rigid H-O-H angle leads to a reduction by a factor of 3 in the number of SHAKE iterations with concomitant economies in computing time.

In practice, the absence of a bending force constant and an equilibrium angle ≠ 180° in the `PARAM` file signals the presence of a non-linear rigid bend (See Section 5.2).

2.5. Torsion potential

In general, the bonding and bending terms are deemed ‘bonded’ interactions and therefore the atoms involved do not interact through the non-bonded potential. However, this simple distinction does

not stretch to the torsional potential restricting rotations around bonds. In this case, there are two possible approaches available in *gmq*. Both of these use, either in part or wholly, a parametrization in terms of the torsion angle, τ. Given four contiguous atoms {i, j, k, l} in a molecule, the torsion (or dihedral) angle between the planes of the two groups {i, j, k} and {j, k, l} can be defined as

$$\cos(\tau) = -\frac{(\vec{r}_{ij} \times \vec{r}_{jk}) \bullet (\vec{r}_{jk} \times \vec{r}_{kl})}{\left|\vec{r}_{ij} \times \vec{r}_{jk}\right| \left|\vec{r}_{jk} \times \vec{r}_{kl}\right|} \tag{2.5.1}$$

In the convention used in *gmq* the dihedral angle varies from -180° to +180° with τ=0° corresponding to the *trans* conformation, i.e. where all four atoms lie in the same plane and atoms i and l are at their furthest distance apart. The parametrization used takes the form of a polynomial in cos τ

$$\Phi(\tau) = \sum_{m=0}^{n} C_m cos^m \tau \tag{2.5.2}$$

where the C_m are the coefficients of a polynomial of order n. One option is to have part of the torsional energy described by Eq. 2.5.2 whilst non-bonded interactions between the atoms at the ends of the torsion, i and l, makes up the remainder, i.e. atoms separated by more than two bonds are deemed non-bonded. The alternative is to choose the coefficients in Eq. 2.5.2 to represent the entire torsional energy, in which case i and l are deemed not to interact through the non-bonded potential, i.e. atoms separated by more than three bonds are deemed non-bonded.

2.6. Out-of-Plane potential

To restrict the motion of a central tri-valent atom, i, out of the plane of the three atoms, {j,k,l}, it is directly bonded to, an out-of-plane potential is available. The form of the potential is as following

$$\Phi_{oop}(s) = \frac{1}{2} k_{oop} s^2 \tag{2.6.1}$$

where s is the perpendicular distance of atom i from the plane of {j,k,l}

$$s = \vec{r}_{ji} \bullet \frac{\left(\vec{r}_{jk} \times \vec{r}_{jl}\right)}{\left|\vec{r}_{jk} \times \vec{r}_{jl}\right|} \tag{2.6.2}$$

This form of out-of-plane potential is preferred to the improper dihedral form as the latter is ill-defined; there being three possible definitions of the improper dihedral angle and hence three different energies.

2.7. Special constraints for Hydrogen atoms

In addition to the simple bond constraints mentioned in Section 2.2 and the rigid linear bend constraint in Section 2.4, there exists in *gmq* the option of freezing out all the high frequency modes associated with hydrogen atoms in CH_3 and CH_2 groups using the methods described by Hammonds and Ryckaert [12]. For CH_3 groups extra rigid bonds are added to remove all but the rotation around the terminal bond. For an X-CH_2-Y group, holding the mid-point of the H-H vector on the bisector of the X-C-Y bending angle and maintaining the H-H vector perpendicular to the vector joining the X and Y

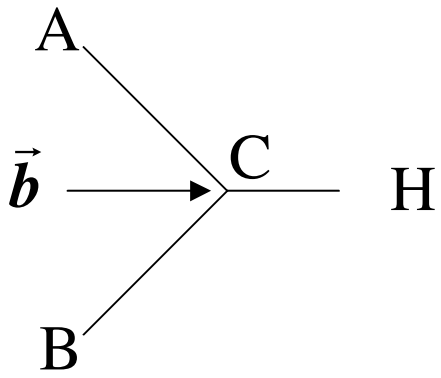
atoms removes all the degrees of freedom of the hydrogens. Such a constraint involves the positions of five atoms. Although there are different methods available to incorporate a solution of the constraints problem into the equations of motion, *gmq* uses probably the most common one based on the iterative procedure 'SHAKE' [14]. Each constraint is satisfied in turn and the process repeated until all the constraints are at least satisfied to a user specified tolerance. In *gmq* the user specifies a relative tolerance, variable TOLER, which applies to all constraints. For example, in the case of rigid bonds it means that after the application of SHAKE all bond lengths satisfy the inequality

$$\frac{\left|\left|\vec{r}_{ij}\right| - b_0\right|}{b_0} < \text{TOLER} \tag{2.7.1}$$

How the relative tolerance criterion is applied to the special CH_2-group constraints is discussed in [12].

The specific atom types involved in special CH_3 and CH_2 constraints are specified in the `DATA.NEW` file, as explained in Section .5.3.

As well as special CH_2 and CH_3 constraints, *gmq* can also apply a coplanar vector bisector constraint, designed to remove the degrees of freedom of hydrogen atoms in aromatic CH groups. Consider the four coplanar atoms A, B, C and H.



We wish to keep the light H atom in the plane of ABC and on the bisector of the ACB angle. We assume that the C-H bond length is constrained to a fixed value using a regular bond constraint. The two remaining degrees of freedom of the H atom should then removed by the coplanar bisector vector constraint. The vector that bisects the ACB angle can be written as

$$\vec{\beta} = \frac{\vec{r}_{CA}}{d_{CA}} + \frac{\vec{r}_{CB}}{d_{CB}} \Rightarrow d_{CA}\vec{\beta} = \vec{r}_{CA} + \frac{d_{CA}}{d_{CB}}\vec{r}_{CB} = \vec{r}_{CA} + \alpha\vec{r}_{CB} = \vec{b} \tag{2.7.2}$$

This allows for different A-C and B-C bond lengths. The position of H can then be written in terms of C and the bisector vector as

$$\vec{r}_H = \vec{r}_C + \gamma\vec{b} \tag{2.7.3}$$

where γ is a scalar which varies with time. It can be determined from

$$\gamma = \frac{-\vec{r}_{CH} \bullet \vec{b}}{\vec{b} \bullet \vec{b}} \tag{2.7.4}$$

The corresponding set of three constraint relationships is then

$$\vec{\sigma} = \vec{r}_{CH} + \gamma \vec{b} = \vec{0} \tag{2.7.5}$$

This corresponds to the removal of two degrees of freedom as the value of γ can vary with time, i.e. H retains one degree of freedom as it could slide along the bisector vector, if the C-H bond was not maintained rigid.

From the constraint relation, a criterion for convergence can be written for each cartesian direction, e.g.

$$\frac{\left|\tilde{x}_{CH} + \gamma\left(\tilde{x}_{CA} + \alpha \tilde{x}_{CB}\right)\right|}{\left|\tilde{r}_{CH}\right|} < \text{TOLER} \tag{2.7.6}$$

where the overstrike "~" indicates the value at the current iteration of SHAKE.

As these groups are identified by a central tri-valent atom they are treated as a special form of out-of-plane group. A group to be subject to this special constraint is thus indicated in the `PARAM` file in the list of out-of-planes, as explained in Section 5.2.

2.8. Diatomic molecules with zero mass interaction sites

Some models of diatomic molecules require that extra coulombic interaction sites be defined along the bond vector. A typical case is a homonuclear diatomic molecule with a partial charges of q on the atoms and -2q on the centre-of-mass. Such a 3-interaction site model is one simple example of how to incorporate a quadrupole moment into the interaction potential. The dynamics of such systems can be solved within the framework of special constraints, as explained by Ciccotti *et al.* [15]. In practice the forces on the secondary (zero mass) sites are transferred to the primary atoms having a non-zero mass. The equations of motion are then solved for the primary atoms and then the new positions of the secondary sites are generated from the new bond vector. For example, assume that the diatomic molecule has primary atoms (with non-zero mass) at the vector positions $\vec{R}_1$ and $\vec{R}_2$. A secondary interaction site, with zero mass, can be arbitrarily defined from one of the atoms and the bond vector

$$\begin{aligned} &\vec{r}_\alpha = \vec{R}_1 + C_{\alpha 2}\vec{R}_{21} = \vec{R}_1 + C_{\alpha 2}\vec{R}_2 - C_{\alpha 2}\vec{R}_1 = \left(1 - C_{\alpha 2}\right)\vec{R}_1 + C_{\alpha 2}\vec{R}_2 = C_{\alpha 1}\vec{R}_1 + C_{\alpha 2}\vec{R}_2 \\ &\Rightarrow \vec{r}_\alpha = \sum_{i=1}^{2} C_{\alpha i}\vec{R}_i \end{aligned} \tag{2.8.1}$$

where $C_{\alpha i}$ are scalar constants. According to Ciccotti *et al.* [15], the force on the secondary site is partitioned between two primary atoms such that $\vec{F}_i' = \vec{F}_i + C_{\alpha i}\vec{f}_\alpha$. This ensures that the sum of the forces

$$\sum_{i=1}^{2} \vec{F}_i + \vec{f}_\alpha = \sum_{i=1}^{2} \vec{F}_i' \tag{2.8.2}$$

and the sum of the torques

$$\sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \vec{\boldsymbol{F}}_i + \vec{\boldsymbol{r}}_\alpha \times \vec{\boldsymbol{f}}_\alpha = \sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \vec{\boldsymbol{F}}'_i \quad , \tag{2.8.3}$$

and hence the dynamics, are preserved. This latter equality follows from the definition of $\vec{\boldsymbol{F}}'_i$:

$$\sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \vec{\boldsymbol{F}}'_i = \sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \left(\vec{\boldsymbol{F}}_i + C_{\alpha i} \vec{\boldsymbol{f}}_\alpha \right) = \sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \vec{\boldsymbol{F}}_i + \sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times C_{\alpha i} \vec{\boldsymbol{f}}_\alpha = \sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \vec{\boldsymbol{F}}_i + \left(\sum_{i=1}^{2} C_{\alpha i} \vec{\boldsymbol{R}}_i \right) \times \vec{\boldsymbol{f}}_\alpha$$

$$= \sum_{i=1}^{2} \vec{\boldsymbol{R}}_i \times \vec{\boldsymbol{F}}_i + \vec{\boldsymbol{r}}_\alpha \times \vec{\boldsymbol{f}}_\alpha$$

The method is not limited to just one extra interaction site as an arbitrary number of extra sites can be defined along the bond vector. For example, a five site model of a diatomic molecule gives extra liberty in the definition of the quadrupolar moment [16]. Such models can be used in *gmq* and the diatomic molecules concerned are identified as being those that contain atom types of zero mass, as indicated in the `PARAM` file. This is explained in Section 5.2.

Although the general technique can be applied to any kind of molecule [15], the current version of *gmq* is limited to diatomics.

3. Computational aspects of non-bonded force calculation.

3.1. Ewald summation

As mentioned in Section 2.1, *gmq* employs the Ewald method to handle the Coulombic potential for which simple truncation is not a justifiable option due to the slow convergence of the sum of terms only decaying as 1/r. In the Ewald method the total Coulombic energy for a molecular system in 3D periodic boundary conditions, within the "tinfoil" approximation [1], can be written as [10]

$$U_c = \frac{1}{2V\varepsilon_0}\sum_{\mathbf{k}\neq\mathbf{0}}^{\infty} A_k \left|Q_\Sigma\right|^2 + \frac{1}{4\pi\varepsilon_0}\sum_{j=1}^{N}\sum_{m>j}^{N}\frac{q_i q_m}{\left|\vec{\boldsymbol{r}}_{jm}\right|} erfc\left(\alpha\left|\vec{\boldsymbol{r}}_{jm}\right|\right) - \frac{\alpha}{4\pi^{3/2}\varepsilon_0}\sum_{j=1}^{N} q_j^2 - \frac{1}{4\pi\varepsilon_0}\sum_{j=1}^{N}\sum_{m>j}^{m\ \text{"bonded" to}\ j}\frac{q_j q_m}{\left|\vec{\boldsymbol{r}}_{jm}\right|} \tag{3.1.1}$$

In Eq. 3.1.1, *erfc* is the complementary error function, α the Ewald separation parameter, $\vec{\boldsymbol{r}}_{jm}$ the minimum image separation vector of the two atoms, the term A_k is given by

$$A_k = \frac{\exp(-\vec{\boldsymbol{k}}^2/4\alpha^2)}{\vec{\boldsymbol{k}}^2} \tag{3.1.2}$$

and

$$Q_\Sigma = \sum_{j=1}^{N} q_j \exp\left(-i\vec{\boldsymbol{k}}\bullet\vec{\boldsymbol{r}}_j\right) \tag{3.1.3}$$

It should also be noted that

$$\left|Q_\Sigma\right|^2 = Q_\Sigma Q_\Sigma^* = \left(\sum_{j=1}^{N} q_j \exp\left(-i\vec{\boldsymbol{k}}\bullet\vec{\boldsymbol{r}}_j\right)\right)\left(\sum_{j=1}^{N} q_j \exp\left(i\vec{\boldsymbol{k}}\bullet\vec{\boldsymbol{r}}_j\right)\right) \tag{3.1.4}$$

i.e. Q_Σ^* is the complex conjugate of Q_Σ. The reciprocal space vectors $\boldsymbol{k}$ are defined with respect to the matrix of basis vectors **h** as

$$\vec{\boldsymbol{k}} = 2\pi\left(\mathbf{h}^{-1}\right)^t\begin{pmatrix} l \\ m \\ n \end{pmatrix}, \tag{3.1.5}$$

with l , m , n integers. In practice, the number of reciprocal space vectors included in the summation is limited by a judicious choice of α (see later), and by the use of an upper bound, K_{max}, i.e. $\left(l^2+m^2+n^2\right) \le K_{\max}^2$. Implicit also in the above equations is that the value of α is such that only nearest images need to be taken in the real space calculation. In *gmq* the real space part of the calculation is truncated at a user-specified range of R_{max}. In the equations for the Ewald summation given above it is assumed that the sum of the charges of the N particles in the system is zero. In *gmq* a fatal error is issued and the program exits if this is not the case.

As it is written, the double sum in the second term of equation (3.1.1) includes all possible pairs in the system. Together with the reciprocal space calculation (first term in equation (3.1.1)), it implies that all pairs have interacted through the full Coulomb potential. In *gmq*, atoms belonging to the same molecule are allowed to interact via the non-bonded part of the potential only if they are separated by more than two, or optionally more than three bonds, therefore these interactions have to be subtracted from the total Coulombic contribution. They are removed by the fourth term in equation (3.1.1). In *gmq* a slightly different, but entirely equivalent, expression is actually used to calculate the Coulombic energy

$$U_c = \frac{1}{2V\varepsilon_0}\sum_{\vec{k}\neq\vec{0}}^{\infty} A_k \left|Q_\Sigma\right|^2 + \frac{1}{4\pi\varepsilon_o}\sum_{j=1}^{N} \sum_{\substack{m<j \\ m\ 'not\ bonded'\ to\ j}}^{N} \frac{q_j q_m}{|\vec{r}_{jm}|} erfc\left(\alpha\left|\vec{r}_{jm}\right|\right) \\ - \frac{\alpha}{4\pi^{3/2}\varepsilon_0}\sum_{j=1}^{N} q_j^2 - \frac{1}{4\pi\varepsilon_o}\sum_{j=1}^{N} \sum_{m>j}^{m\ 'bonded'\ to\ j} \frac{q_j q_m}{|\vec{r}_{jm}|} erf\left(\alpha\left|\vec{r}_{jm}\right|\right) \tag{3.1.6}$$

where *erf* is the error function. By avoiding the "bonded" pairs in the second term of Eq. 3.1.6 we return the usual all-pair loop used for non-bonded interactions. In doing this we must now modify the correction term (fourth term) accordingly to reflect that it is now only correcting for that part of the Coulombic energy calculated in reciprocal space.

3.2. Pair-searching techniques: Link-cells and neighbour tables

In general the non-bonded part of the force calculation accounts for much of the CPU time expended in an MD simulation. Hence it is usually the factor that most limits the size of system that can be simulated and the time for which the trajectory can be reasonably followed. Disregarding for the moment those pair interactions deemed to be "bonded", there are nominally N(N-1)/2 pair separations to be calculated for a set of N atoms. In practice the number of pair separations that have to be calculated at each step is reduced in the case of the VdW potential by the application of the truncation at r_c. For the Coulombic potential the use of the Ewald summation means that we can also restrict the explicit calculation of pair separations to a range of R_{max}. In principle then, for each atom we only need to calculate the distance between it and those atoms within a radius R_c where $R_c=\max\{r_c,R_{max}\}$. Of course, in any system where the atoms are diffusing there is no *a priori* way of knowing what the indices of these atoms are so we have to resort to some computational procedures.

In *gmq* two common techniques are combined. A link-cell routine [1] is used at the first step, and periodically thereafter, to build a standard Verlet neighbour table [1]. In the link-cell technique the MD box is subdivided into smaller sub-volumes, link-cells, of dimensions equal to or greater than $R_c+\Delta R_c$ where ΔR_c is termed the 'shell width'. In an order N operation, atoms are then sorted according to their coordinates into these link-cells in such a way that atoms within the same link-cell can be found easily using a linked list. The search for neighbours of an atom can then be restricted to just those atoms in the same link-cell and those in nearest-neighbour link-cells. As each pair separation need only be calculated once, this means searching only 13 of the 26 near-neighbour link-cells (in 3D).

To calculate the non-bonded pair interactions even more efficiently a standard Verlet neighbour table technique is also used in *gmq*. At the first step, during the link-cell search, all the neighbours of each subject particle within a radius of $R_c+\Delta R_c$ are stored sequentially in a large one dimensional array. For a number of subsequent steps this table is used, thus pre-eliminating many of the possible interactions. The table is automatically deemed redundant once any particle has diffused a distance greater than or equal to half the shell width: there is no need for the user to set the frequency of neighbour table rebuilds. At this point a new table is formed, again using the link-cell search, and the process then continues in the same way as before. In using the neighbour table approach there is a trade-off between forming a long list, i.e. a large shell width, which is updated less frequently, but where the intervening steps take longer to process, and using a small shell width with more frequent updates of a consequently shorter list. To optimize the shell width short test runs can be carried out with different values of ΔR_c. An example of the typical results obtained is given in Fig. 1 of [17].

4. Integration Algorithm

The integration algorithm used within *gmq* is based on that given previously [17]. The main difference between the algorithm described before [17] and that used in *gmq* is in the extension of the pressure tensor calculation to account for systems in which Ewald summations are used [18]. In *gmq* the pressure tensor is measured in what is termed the "atomic" frame of reference, i.e. momentum is localized at atomic positions, rather than to the alternative "molecular" pressure tensor, momentum localized at the molecular centres-of-mass. The calculation of the atomic pressure tensor in *gmq* is described in detail in [18].

In brief then, the equations of motion are numerically integrated in *gmq* using the leapfrog form of the Verlet algorithm [1] in combination with an iterative routine (SHAKE) to solve the constraints [12]. The algorithm incorporates loose-coupling techniques [17, 19] so that simulations can be carried not only at constant energy (NVE) but also with control over the temperature and pressure. The following options are currently available:-

4.1. Temperature control ("NVT")

The loose-coupling method of Berendsen *et al* [19] is used to control the temperature of the system. In effect, the internally measured temperature, $T(t)$, is coupled to an external heat bath at a temperature $T_{req}(t)$,

$$\dot{T}(t) = \frac{-1}{\tau_T}\Big(T(t) - T_{req}(t)\Big) \qquad (4.1.1)$$

with the user-specified relaxation time, τ_T, determining the rate of heat flow. The temperature of the external heat bath does not have to be constant; the user has control over the initial value, $T_{req(0)}$, and its rate of change, $\dot{T}_{req}(t)$. Thus cycles of heating and cooling at fixed rates are relatively straightforward to perform.

4.2. Temperature and pressure tensor control ("N**P**T")

In addition to the temperature control of the previous section, all six independent elements of the atomic pressure tensor are controlled; NB the atomic pressure tensor is symmetric. In this case the size and shape of the MD box change according to the following equation

$$\dot{\boldsymbol{h}}(t) = \frac{\boldsymbol{P}(t) - \boldsymbol{P}_{\text{req}}(t)}{\tau_P \mu} \tag{4.2.1}$$

where μ is a pre-defined constant and τ_P is a user-specified relaxation time determining the rate at which the $\boldsymbol{h}$ matrix responds to the imbalance between the internally measured pressure tensor, $\boldsymbol{P}$, and that required, $\boldsymbol{P}_{req}$. As with the temperature, both the initial value, $\boldsymbol{P}_{req}(0)$, and its rate of change, $\dot{\boldsymbol{P}}_{req}$, are user-definable.

It should be noted that there is a problem with the formulation given in Eq. 4.2.1 in that τ_P is not a universal constant. For example, in systems of nominally the same material but of a different size a re-tuning of τ_P is required, particularly if mechanical tests are being made. Effectively similar differences in the required pressure lead to proportionately different changes in the shape and size of the MD box.

4.3. Temperature and scalar pressure control ("NpT")

In this case the equation for the rate of change of the **h** matrix becomes

$$\dot{\mathbf{h}}(t) = \frac{p(t) - p_{req}(t)}{\tau_p \mu}\mathbf{1} \tag{4.3.1}$$

where p is the scalar pressure, i.e. $p = 1/3\ \text{Tr}\{\boldsymbol{P}\} = (P_{xx}+P_{yy}+P_{zz})/3$, and **1** is the unit tensor. This is a special case designed for isotropic liquid systems that are, probably, initially set up with a cubic MD box and there is no reason to suggest that they should become anisotropic. If liquid systems are simulated using full flexibility of the **h** matrix there is a strong likelihood that fluctuations in the pressure tensor will drive the system away from its cubic origins. Ultimately the box may become so distorted that the perpendicular distance between any two faces may become less than two times the desired non-bonded potential truncation distance.

A better alternative to the above would be to have a method that allows the box to fluctuate in response to differences between the desired and measured scalar pressure but maintains the shape of the original cell. This would be possible by defining $\mathbf{h}(t)=\mathbf{g}L(t)$ where **g** is a constant matrix and L varies according to

$$\dot{L}(t) = \frac{p(t) - p_{req}(t)}{\tau_p \mu} \tag{4.3.2}$$

This would be more a general scheme encompassing the cubic-box-remaining-cubic option already implemented. It is possible this method will be implemented in a future release.

4.4. Notes on the use of loose-coupling techniques

It has been argued that loose-coupling techniques, even when the desired temperature and pressure are fixed, do not correspond to a well-defined thermodynamic ensemble [20]. Although this has been contested [21] in practical terms it is of very little importance except when trying to calculate 2nd order properties from fluctuations, e.g. specific heat. All first order properties such as energies, pressure, temperature, structure, etc. are unaffected [20]. In addition, loose-coupling techniques are relatively simple to implement and are robust. The first of these points is particularly pertinent when it comes to systems in which the pressure tensor itself is difficult to calculate [18], e.g. systems containing "infinite" chains with constraints present and Ewald summations in use. The alternative extended Hamiltonian methods [22] are difficult to apply to such systems. With regard to robustness, the loose-coupling techniques are naturally over-damped and produce smoother, less oscillatory responses to sudden changes in pressure or temperature. This is particularly important in non-equilibrium simulations of the response of systems to applied changes in the pressure tensor [23, 24].

For the cautious, who are doing equilibrium studies, it is recommended that the N**P**T or NpT algorithms be used just to drive the system to some state point of interest. After equilibration production runs can be performed in the NVE ensemble or using the NVT option with very loose-coupling just to keep the system close to the average temperature required.

4.5. Choice of loose-coupling constants τ_P and τ_T

In choosing values for these two coupling constants there are two competing factors that have to be taken into account. Firstly, to minimize the disruption to the system it is desirable to reduce the coupling of the system with the external pressure and temperature "baths" as much as possible, and this can be achieved by using larger values of τ_P and τ_T. However, τ_P and τ_T must not be so large as to have no effect on the time scale of the simulation. No unique universal values of these parameters can be given as they will depend on many factors, e.g. the system being studied, the applied conditions, the rate of change of those conditions, etc. In the past, some procedures were introduced that can be used as a guide to the choice of τ_P and τ_T [17]. It was shown that in NVT simulations a value of $\tau_T = 100\Delta t$, where Δt is the user-specified time step in the integration algorithm, leads to temperature fluctuations very close to those in the NVE case, i.e. $\tau_T = \infty$. In addition, even values of $\tau_T = \Delta t$ can be used without any breakdown in program stability as in this case the algorithm corresponds to the simple *ad-hoc* velocity re-scaling method long used in MD as a means to adjust temperature. The choice of τ_P is more critical in this respect as too large changes in the basis vectors can lead to algorithmic breakdown. For this reason the maximum change in elements of $\boldsymbol{h}$ is limited in *gmq*. If this limit is exceeded a warning is issued at the step at which it first occurred and at the end of the run a message printed as to the number of times it happened and the user is advised to increase τ_P before the next run.

4.6. "Energy Minimization"

Two forms of rudimentary Energy Minimization (EM) can be performed in *gmq* either at constant pressure or at constant volume. The EM algorithm is simply the MD algorithm with the velocities zeroed

at each step, i.e. all motions are inertialess and thus derive from the forces acting. Displacements are limited to an arbitrarily assigned maximum value to avoid algorithmic breakdown. This method is not particularly efficient at finding true energy minima but is useful to remove high energy contacts in initial configs.

5. Preparing to run an MD simulation with *gmq* (v10.2 - 19/07/2019)

Before using *gmq* a minimum of three input files have to be prepared; this increases to four for *ddgmq* but this will be discussed later in a separate section. The file `CONFIG.NEW` contains information associated with the configuration, coordinates and velocities of the atoms, connectivity table. The file `PARAM` contains information regarding the parameters used in the various potentials and the masses and types of atoms. The file `DATA.NEW` is used to control the length of the simulation, the output produced and various other aspects. The exact format of these three obligatory input files is given below along with that of a number of other files, such as `SHAPE` and `CHARGE`, which in certain circumstances are also required. It should be noted that the `PARAM` and `DATA.NEW` files depend on whether the LJ (either 12-6 or generalised N-M) or Buckingham forms of the potential are to be used. In what follows below the LJ form is being used. The changes required to use the Buckingham form will be described in a separate section later.

5.1. The `CONFIG.NEW` input file

Record 1: Arbitrary title.

Record 2: Comment (ignored).

Record 3: Number of the job, Time into the run in ps, Required temperature in Kelvin (T_{req}).

Record 4: Comment .

Record 5: Required values of P_{xx}, P_{yy} and P_{zz} in bars (1 bar = 10^5 Pa).

Record 6: Comment .

Record 7: Required values of P_{xy}, P_{xz} and P_{yz} in bars.

Record 8: Comment.

Record 9: First row of **h**/2 matrix, $h_{11}/2$, $h_{12}/2$ and $h_{13}/2$, in metres then corresponding logical variables.

Record 10: Second row of **h**/2 matrix, $h_{21}/2$, $h_{22}/2$ and $h_{23}/2$, in metres then corresponding logical variables.

Record 11: Third row of **h**/2 matrix, $h_{31}/2$, $h_{32}/2$ and $h_{33}/2$, in metres then corresponding logical variables.

Records 12-16: Comments.

Record 17: Total number of molecules in the system (NUM).

Record 18: Total number of atoms in molecule 1 (NAT(1)).

Record 19: Global atom index; Atom Type; No. of bonds (connected neighbours); x, y and z coordinate in metres.

Record 20: x, y and z velocities in metres/second; Global indices of bonded atoms.

The rest of atoms in molecule 1 then follow with two records for each atom as above. After the last atom is the record for the number of atoms in molecule 2 and so on. Part of a typical `CONFIG.NEW` file follows:-

Example start to a CONFIG.NEW file

```
**** Snake protein echistatin *****
**** No._of_job       Time_into_run/ps      Reqd._Temp/K
         2                0.2000            300.000000
**** Reqd.   Pxx            Pyy             Pzz    (in bars)
           0.000           0.000           0.000
**** Reqd.   Pxy            Pxz             Pyz    (in bars)
           0.000           0.000           0.000
**** H Matrix of HALF basis vectors; Variable? (T or F)
  8.82105701E-09  0.  0.    T F F
  0.  4.91449439E-09  0.    F T F
  0.  0.  4.62553493E-09    F F T
*********************************************************************
********** Record for each atom is in the form:-            **********
********** Index Atom_Type No._of_bonds X Y Z (coords.in m) **********
********** Vx Vy Vz (in m/s) Indices_of_bonded_atoms        **********
*********************************************************************
            1  Total_No._of_molecules
          713  Atoms_in_Molecule_No.            1
1  1 4  0.78844581819655E-09 -0.12263680976542E-08 -0.13810634207111E-08
-402.0900071142408  -780.4795505577129  918.3066412566724    5  2  3  4
```

Notes on the format of CONFIG.NEW:-

(1) Everything is read-in in free format so spacing is not so important. Lines should not be longer than 80 characters though.

(2) The comment records shown are those produced when the file `CONFIG.NEW1` is written after a successful run of *gmq*.

(3) Record 3: The number of the job is kept just for information and does not affect the run in any way. The simulation time is used in the production of history files so does affect the output. Subsequent runs use the time into the run when appending history files. Normally the job number and time into run will be zeroed manually if the clock is to be reset to zero. The required temperature is kept in the `CONFIG.NEW` file as it can change linearly with time at a user-specified rate (see Section 4 and `DATA.NEW`) and so is associated with the configuration.

(4) As with the required temperature, the required pressure must be associated with the configuration as it can also vary with time.

(5) ***NB***. Largely for historical reasons, the matrix **h**/2 is read-in from the `CONFIG.NEW` file. Note also that basis vectors are given by columns of **h** so the vector **a**, say, is given by (h_{11},h_{21},h_{31}). Each element of **h** has an associated logical variable (T or F) which controls whether the corresponding element of **h** is variable (T) or fixed (F) in the case where controlled pressure (N**P**T) dynamics is used, i.e. IMOV=1 (see `DATA.NEW` file).

(6) The global atom index is only for guidance when looking at the `CONFIG.NEW` file. It is not actually used in the program to assign an atom index rather it is the order in which atoms appear in the `CONFIG.NEW` file which determines their global index.

(7) Atom types must correspond to one of those listed in the `PARAM` file (see later). There is no universal list of atom types in *gmq* so one has to be careful to keep `CONFIG.NEW`s with the corresponding `PARAM` file.

(8) The indices of atoms to which an atom is immediately bonded, given in the same record as the velocities, are global indices.

5.2. The `PARAM` input file (LJ or WCA VdW interactions)

Record 1: Arbitrary title.

Record 2: The number of different atom types in `CONFIG.NEW` (NTYPE).

Record 3: Comment.

Record 4: Atom label, mass in g/mole and charge in e (charge on the proton) for atom type 1.

Record 5: Atom label, mass in g/mole and charge in e for atom type 2.

....and so on down to....

Record 3+NTYPE: Atom label, mass (in g/mole) and charge (in e) for atom type NTYPE.

Record 4+NTYPE: No of different types of bonds (NBOND).

Record 5+NTYPE: Comment.

Record 5+NTYPE+1: Atom types, b_o (in Ångstroms) and k_b (in kg s^{-2}) for bond type 1.

....and so on down to....

Record 5+NTYPE+NBOND: Atom types {i,j}, b_o and k_b for bond type NBOND.

Record 6+NTYPE+NBOND: No. of different types of bends (NBEND).

Record 7+NTYPE+NBOND: Comment.

Record 7+NTYPE+NBOND+1: Atom types {i,j,k}, θ_0 (in deg.) and k_θ (in kJ/mole) for bend type 1.

....and so on down to....

Record 7+NTYPE+NBOND+NBEND: Atom types {i,j,k}, θ_0 and k_θ for bend type NBEND.

Record 8+NTYPE+NBOND+NBEND: No. of different torsion types (NTOR).

Record 9+NTYPE+NBOND+NBEND: Comment.

Record 9+NTYPE+NBOND+NBEND+1: Atom types {i,j,k,l} in torsion type 1.

Record 9+NTYPE+NBOND+NBEND+2: Coeffs. C_m ($0 \le m \le 6$) (in J/mole) in torsion type 1.

Record 9+NTYPE+NBOND+NBEND+3: Atom types {i,j,k,l} in torsion type 2.

Record 9+NTYPE+NBOND+NBEND+4: Coeffs. C_m ($0 \le m \le 6$) (in J/mole) in torsion type 2.

....and so on down to....

Record 9+NTYPE+NBOND+NBEND+2*NTOR-1: Atom types {i,j,k,l} in torsion type NTOR.

Record 9+NTYPE+NBOND+NBEND+2*NTOR: Coeffs. C_m ($0 \le m \le 6$) in torsion type NTOR.

Record 10+NTYPE+NBOND+NBEND+2*NTOR: No. of non-bonded interaction types (NNB).

Record 11+NTYPE+NBOND+NBEND+2*NTOR: Comment.

Record 11+NTYPE+NBOND+NBEND+2*NTOR+1: Atom types {i,j}, σ (in Å), ε/k_B (in K) in non-bonded interaction type 1.

....and so on down to....

Record 11+NTYPE+NBOND+NBEND+2*NTOR+NNB: Atom types {i,j}, σ, ε/k_B in non-bonded interaction type NNBTYP.

Record 12+NTYPE+NBOND+NBEND+2*NTOR+NNB: No. of out-of-plane types.

Record 13+NTYPE+NBOND+NBEND+2*NTOR+NNB: Comment.

Record 13+NTYPE+NBOND+NBEND+2*NTOR+NNB+1: Central atom type, i, followed by three other atom types {j,k,l}, force constant in kg s^{-2} for OOP type 1.

....and so on for all OOP types.

Three examples of `PARAM` files follow:-

Example `PARAM` file 1 (LJ or WCA VdW interactions)

```
**** Benzene ****
           2 # No._of_different_atom_types
 #   Label     Mass/(g/mole)     Charge/e
     cp           12.011150      -0.1D0
     h            1.007970        0.1D0
           2 # No._of_different_types_of_bonds
 # Type_1 Type_2   Bond_Length/A   F_con/kg/s/s
     1    1         1.34000         666.9792135406
     1    2         1.08000         504.9816347078
          2 # No._of_different_types_of_bond_angles
 # Types_1_2_3         Theta_0     k_theta/(kJ/mole)
          1 1 1         120.000     1004.1600000000
          1 1 2         120.000     412.8213333333
            1 # No._of_different_types_of_torsions
 # Atom_Types_1--2--3--4  & on next line Coeffs. in J/mole
  -1 1 1 -1
  100416.000       0.000 -100416.000       0.000       0.000       0.000
           2 # No._of_types_of_non-bonded_interactions
# Type_1 Type_2        Sigma/A         Eps/k/K
    1     1         3.6170487995     74.47605475
    2     2         2.4499714540     19.12223106
           1 # No._of_types_of_out-of-planes
# Central_atom_type   Outer_atom_types    Koop/kg/s/s
          1             1  1  2              660.0
```

Example PARAM file 2 (LJ or WCA VdW interactions)

```
**** PVC ****
           6 No._of_different_atom_types
 **** Label     Mass/(g/mole)     Charge/e
   Chlorine       30.93          -0.1987
   C_methy        12.01          -0.1792
   H_methy         1.01           0.0990
   C_methi         8.71           0.0313
   H-methi         8.81           0.1486
   Methyl         15.04           0.0094
          5  No._of_different_types_of_bonds
 ****  Type_1  Type_2     Bond_Length/A
         1       4          1.83
         2       3          1.09
         2       4          1.53
         4       5          1.09
         4       6          1.53
          10  No._of_different_types_of_bond_angles
 ****  Type_1  Type_2  Type_3  Theta_0    k_theta/(kJ/mole)
         2       4       2     108.2      823.49488
         6       4       2     108.2      823.49488
         4       2       4     112.0      830.77504
         5       4       2     106.3      517.22608
         4       2       3     106.3      0.0
         6       4       5     106.3      517.22608
         3       2       3     109.0      0.0
         1       4       2     106.3      626.63768
         1       4       6     106.3      626.63768
         1       4       5     103.0      554.42184
           3  No._of_different_types_of_torsions
 ****  Atom_Types_1--2--3--4  & on next line Coeffs. in J/mole
                  4  2  4  2  are types involved in torsion   1
   3974.8  7740.4  0.0 -11715.2 0.0  0.0  0.0
                  2  4  2  4  are types involved in torsion   2
   3974.8  7740.4  0.0 -11715.2 0.0  0.0  0.0
                  6  4  2  4  are types involved in torsion   3
   3974.8  7740.4  0.0 -11715.2 0.0  0.0  0.0
     21  No._of_types_of_non-bonded_interactions
**** Type_1  Type_2     Sigma/A    Eps/k/K
      1    1          3.5386        80.9172
      1    2          3.5154        46.8916
      1    3          3.2004        18.7132
      1    4          3.5154        46.8916
      1    5          3.2004        18.7132
      1    6          3.6292        76.0132
      2    2          3.4921        27.1737
      2    3          3.1771        10.8443
      2    4          3.4921        27.1737
      2    5          3.1771        10.8443
      2    6          3.6059        44.0497
      3    3          2.8621         4.3277
      3    4          3.1771        10.8443
      3    5          2.8621         4.3277
      3    6          3.2909        17.5791
      4    4          3.4921        27.1737
      4    5          3.1771        10.8443
      4    6          3.6059        44.0497
      5    5          2.8621         4.3277
      5    6          3.2909        17.5791
      6    6          3.7197        71.4064
```

Example PARAM file 3 (LJ or WCA VdW interactions)

```
*N2 EVH version of VSH model with zero mass COM & posns. of charges ******
        5 No._of_different_atom_types
******  Label         Mass/(g/mol)         Charge/e  ******
        N1            14.0067400            0.0          #   1    N
        N2            14.0067400            0.0          #   2    N
        COM_N          0.0                 10.87         #   3    COM_N
        QA_N           0.0                 -5.435        #   4    QA_N
        QB_N           0.0                 -5.435        #   5    QB_N
       10 No._of_different_types_of_bonds
******  Type_1    Type_2           Bond_Length/A   ******
           1         2              1.0464      -1.000  #   1   N1   N2
           1         3              0.5232      -1.000  #   2   N1   COM_N
           2         3              0.5232      -1.000  #   3   N2   COM_N
           1         4              0.35714     -1.000  #   4   N1   QA_N
           2         5              0.35714     -1.000  #   5   N2   QB_N
           1         5              0.68926     -1.000  #   6   N1   QB_N
           2         4              0.68926     -1.000  #   7   N2   QA_N
           3         4              0.16606     -1.000  #   8   COM_N   QA_N
           3         5              0.16606     -1.000  #   9   COM_N   QB_N
           4         5              0.33212     -1.000  #  10   QA_N   QB_N
       0 No._of_different_types_of_bond_angles
       0 No._of_different_types_of_torsions
       5 No._of_types_of_non-bonded_interactions
******  Types_1   Types_2       Sigma/A      Eps/k (K)  ******
           1         1           3.3211     34.897     #   1   N1        N1
           2         2           3.3211     34.897     #   2   N2        N2
           3         3           3.3211        0.0     #   3   COM_N   COM_N
           4         4           3.3211        0.0     #   4   QA_N    QA_N
           5         5           3.3211        0.0     #   5   QB_N    QB_N
```

Notes on the format of PARAM:-

(1) Everything is read-in in free format so spacing is not so important.

(2) The atom types should be the ones required by the corresponding `CONFIG.NEW` file.

(3) The `PARAM` file assumes all atoms of the same type have the same charge. This has been found to be somewhat restrictive and so can be overridden (see description of file `CHARGE` and logical flag LOGCHG in file `DATA.NEW`).

(4) For the bonds types, if k_b is omitted or has a negative value then bonds of this type are assumed to be rigidly constrained to the equilibrium distance, b_o.

(5) If bonding parameters are omitted for a pair of bonded atoms the user is warned and the run terminates.

(6) If bending parameters are omitted for angles that exist in a molecule then the end atoms ("1···3" interaction) are deemed to interact via the non-bonded potential. If this is not wanted then the appropriate bend type should be included in the `PARAM` file with k_θ=0.

(7) To access the second form of bending potential (Eq. 2.3.4), appropriate for angles with minima close to 180 degrees, input the k_θ as a **negative** value. The negative sign serves simply as a switch to recognise that the potential to be used is of the form specified at Eq. 2.3.4. The absolute value of the k_θ input will be used internally in the program. Check the `OUTPUT.NEW` file to see if the desired form has been selected.

(8) If the k_θ force constant is omitted and the equilibrium angle bond angle is set to 180° then this signals that the molecule is a linear rigid triatomic, e.g. CO_2. If the k_θ force constant is omitted and the equilibrium angle bond angle ≠ 180° then this signals that the bend is rigid.

(9) If atom types *i* and *l* in the torsion {*i,j,k,l*} are set to -1 then the torsion type is assumed to be of the BIOSYM-like wild card type. This means that the coefficients given in the next line give the *total* energy of rotation around the type of bond {*,*j*,*k*,*}. This means individual torsions have energies that depend on the number of atoms bonded to the atoms of type *j* and *k*.

(10) If torsion potential parameters are omitted for torsions that exist in a molecule then the end atoms ("1···4" interaction) are deemed to interact via the non-bonded potential even if these are switched-off by the variable LJ14 in the file `DATA.NEW`. If this is not wanted then the appropriate torsion type should be included in the `PARAM` file with all C_m=0.

(11) The coefficients of the polynomial, C_m, used for the torsional potential can be entered for $0 \le m \le 6$, i.e. a maximum of 7 numbers on the line. Each line is first parsed to see the number of separate words on the line so each line can have a different order of polynomial; all higher orders being set to zero. It is important that the line doesn't contain any comments or extraneous information.

(12) For the non-bonded interactions the parameters σ_{ij} and ε_{ij} for unlike atom type pairs ($i \neq j$) can be assumed to be given by certain combining rules. Currently there are two options: Lorentz-Berthelot $\{\sigma_{ij}=(\sigma_{ii}+\sigma_{jj})/2,\ \varepsilon_{ij}=(\varepsilon_{ii}\varepsilon_{jj})^{1/2}\}$ and BIOSYM $\{\sigma_{ij}=(\sigma_{ii}\sigma_{jj})^{1/2},\ \varepsilon_{ij}=(\varepsilon_{ii}\varepsilon_{jj})^{1/2}\}$. The combining rule can be specified by the user (see variable IMIX in the file `DATA.NEW`).

(13) For bonds, bends, torsions and non-bonded parameters symmetry is assumed, i.e. torsion type {*i,j,k,l*} is the same as torsion type {*l,k,j,i*}. Equivalent or duplicate entries in `PARAM` cause *gmq* to stop. Valid entries but for which there are no such types in the `CONFIG.NEW` file are not flagged.

(14) For compatibility with older `PARAM` files, the omission of the number of out-of-plane types is flagged as a warning but the program continues.

(15) Groups of 4 atoms subject to the special coplanar vector bisector "CH" constraint are entered as an out-of-plane group with a missing value for the k_{oop} parameter, i.e. just the four atom types constituting the group are defined. The first atom type has to be that of the central "C" atom and the ***second*** atom type has to be that of the "H" atom to be constrained. The other two atom types are then the third and fourth entries on the line. A check is made of the mass and the connectivity of the "H" atom. If its mass is greater than that of a hydrogen atom a warning is issued. If it is bonded to more than one other atom a fatal error occurs.

(16) Atom types of zero mass are only allowed in rigid diatomic molecules, as explained in Section 2.8. Each zero mass site must be bonded to two atoms with non-zero mass. These two non-zero mass atoms must appear first in the connectivity table of the zero mass site. To avoid unwanted non-bonded interactions all other bonds to other zero mass sites in the same diatomic molecule must be defined (See Example 3 `PARAM` file above).

5.3. The `DATA.NEW` input file (LJ or WCA VdW interactions)

Record 1: Arbitrary title.

Record 2: IMOV, an integer determining the type of simulation to be carried out. Current options include the four types discussed in Section 4 with 0=NVE, 1=N**P**T, 2=NpT and 3=NVT. In addition, IMOV=4 allows one to drive the size and shape of the primary MD box from that contained in `CONFIG.NEW` to a user-specified one defined in the file `SHAPE`, IMOV=5 is a rudimentary "Energy Minimization" (EM) option at constant pressure and IMOV=6 is EM at constant volume option. With IMOV=1individual elements of the **h** matrix can be set to be variable or fixed (see description of `CONFIG.NEW` file).

Record 3: NOFSTP, the number of time steps to be carried out.

Record 4: DT, the time step to be used in the algorithm in femtoseconds (10^{-15} s).

Record 5: The rates of change of the on-diagonal components of the required pressure tensor in bar/ps in the order xx, yy, zz.

Record 6: The rates of change of the off-diagonal components of the required pressure tensor in bar/ps in the order xy, xz, yz.

Record 7: The rate of change of the required temperature, $\dot{T}_{req}$, in K/ps.

Record 8: TAUT, the temperature relaxation time, τ_T, in ps.

Record 9: TAUP, the pressure relaxation time, τ_P, in ps.

Record 10: WHENST, the time between storing data in history files (`STO`, `ST`, etc.) in ps.

Record 11: RIPSTO, the time between storing configurations in ps.

Record 12: IPOT, an integer flag determining whether the LJ or the Weeks-Chandler-Andersen potential is used (1= LJ potn., 2= WCA potn.).

Record 13: CUTVWI, the potential truncation range for the VdW potential in Ångstroms. This variable is only used if IPOT=1.

Record 14: DELCUT, the shell-width, Δr_c, used in the formation of the neighbour table in Å.

Record 15: CUTQI, the real space potential truncation range for the Ewald summation in Å.

Record 16: ALPHA, the factor α appearing in the Ewald summation in $Å^{-1}$.

Record 17: KMAX, the upper bound K_{max} used to limit the range of k-vectors in the reciprocal space part of the Ewald summation.

Record 18: TOLER, The maximum relative tolerance to which all constraints must be satisfied in SHAKE.

Record 19: ICFLAG, an integer flag controlling whether configurations are to written to the `CONF1` and `CONFIG.NEW1` files (1=Yes, 0=No).

Record 20: NTCH2, an integer defining the number of groups to be constrained using the Ryckaert-Hammonds CH_2 type constraints.

Record 21: If NTCH2≥1, ICTCH2 and IHTCH2, are the atom types of CH_2 Carbons and Hydrogens (one pair for each of the NTCH2 lines following).

Rec. 21+NTCH2: NTCH3, an integer defining the number of groups to be constrained using the Ryckaert-Hammonds CH_3 type constraints.

Rec. 22+NTCH2: If NTCH3≥1, ICTCH3 and IHTCH3, are the atom types of CH_3 Carbons and Hydrogens (one pair for each of the NTCH3 lines following).

Rec. 22+NTCH2+NTCH3: LJ14, an integer flag determining whether "1···4" non-bonded interactions are to be included (1=Yes, 0=No).

Rec. 23+NTCH2+NTCH3: IPB, the periodicity of the system (3=3D). This is a dummy variable for the moment.

Rec. 24+NTCH2+NTCH3: LOGCHG, a logical flag determining whether the individual charges for those atoms listed in the file `CHARGE` are to be used (.TRUE.) in preference to those for atom types in the `PARAM` file. If .FALSE. is specified only those in `PARAM` are used.

Rec. 25+NTCH2+NTCH3: IMIX, the mixing rule to be used for the parameters of the LJ potential for missing unlike pairs (0=none, 1=Lorentz-Berthelot, 2=BIOSYM).

An example `DATA.NEW` file follows:

Example `DATA.NEW` file.

```
 **** Converted from BIOSYM *****
0       IMOV   (0=NVE,1=NPT,2=NpT,3=NVT,4=Change H matrix,5=EM-P,6=EM-V)
10      NOFSTP (Number_of_time_steps)
1.0     DT     (Time_step in fs)
0.0 0.0 0.0 dPxx/dt dPyy/dt dPzz/dt (in bar/ps)
0.0 0.0 0.0 dPxy/dt dPxz/dt dPyz/dt (in bar/ps)
0.0     dT/dt  (Rate_of_change_of_Temp. in K/ps)
1.0     TAUT   (Temperature_relaxation_time in ps)
5.0     TAUP   (Pressure_relaxation_time in ps)
0.001   WHENST (Time_between_storing_data in ps)
5.0     RIPSTO (TIME_BETWEEN_STORING_CONFIGS in ps!!)
1       IPOT   (1= LJ potn.,2= WCA potn.)
15.0    CUTVWI (VdW_Potn._Cutoff in Angstroms. If IPOT.NE.2)
1.5     DELCUT (Shell_Width_for_neighbour_table in Angstroms)
15.0    CUTQI  (Cutoff_in_real_space_for_Ewald_potn. in Angstroms)
0.3     ALPHA  (alpha_factor_in_Ewald_sum in Angstroms**(-1))
8       KMAX   (largest_k_vector_to_use_in_Ewald_sum)
1D-6    TOLER  (Relative_Tolerance_for_constraints)
1       ICFLAG (Write_Configs_Every_RIPSTO_ps_and_at_end?    1=Yes, 0=No)
0       NTCH2  (No. of different types of Ryckaert-Hammonds constrained CH2)
0       NTCH3  (No. of different types of Ryckaert-Hammonds constrained CH3)
1       LJ14   (Use non-bonded potn. on 1...4? 1=Yes, 0=No)
3       IPB    (Periodicity: 3=3D)
T       LOGCHG (Use charges in CHARGE not those in PARAM? T or F)
2       IMIX   (Mixing Rule for LJ potn: 0=none,1=Lorentz-Berthelot,2=BIOSYM)
```

Notes on the format of `DATA.NEW`:-

(1) Everything is read-in in free format so spacing is not so important.

(2) The value of IMOV chosen affects the way certain other variables are treated. For example, if IMOV=0 then TAUT and TAUP are not used but must still be included in the `DATA.NEW` file.

(3) The value of WHENST is converted within *gmq* to an integer number of time steps (ISTO=WHENST/DT) between outputting information to various history files. The minimum value is thus every time step. **If this conversion, however, implies that the number of times data is written to the history files is less than two then data is written every time step instead** (i.e. ISTO=1).

(4) **NB**. **Be very careful setting the value of RIPSTO**. If you have even a modest number of atoms your filestore will grow very rapidly if you output the configuration many times. The amount of information written every RIPSTO ps is approximately 3N*4 bytes. Think carefully how often you need the configuration and check this against filestore limitations. Note also that the writing takes place with respect to the time into run as stored in the `CONFIG.NEW` file. Writes are made at 0 ps, RIPSTO ps, 2*RIPSTO ps etc. irrespective of the length of the current run. Individual jobs do not thus have to be an integer multiple of RIPSTO to ensure the same interval between writes.

(5) If IPOT=2 is selected each type of WCA potential has its own cutoff set to the minimum in the corresponding LJ potential. The value of CUTVWI is ignored.

(6) If there are no charges present in the system then CUTQI and KMAX must both be set to zero otherwise a warning is issued and *gmq* terminates.

(7) The optimisation of the three parameters, CUTQI, ALPHA and KMAX, for the Ewald summation will be dealt with in a separate section.

(8) Setting ALPHA to zero and KMAX to zero returns the Coulombic potential to a straight sum in real space. In a dense periodic system this isn't recommended but might be useful for mimicking "vacuum" boundary conditions, i.e. defining an MD box far larger than the extent of the physical system.

(9) A value for the relative tolerance of 10^{-6} is usually adequate but some systems require 10^{-7}. Increasing the value reduces the number of SHAKE iterations that have to be performed and saves CPU time but will ultimately lead to less stable numerical integration and energy drift under NVE conditions. Some discussion of this point is given in [25] and will be covered further later. Tests of stability under NVE conditions should always be made using different values of the relative tolerance on all new systems.

(10) The ICFLAG=0 option can save CPU time and filestore if a number of repeat runs are being carried out from the same initial conditions for the purpose of optimising time steps or tolerance, etc.

(11) As of v8.6 of *gmq* and v6.4 of *ddgmq*, the special CH_2 constraints can be applied to more than one type of "Carbon" and "Hydrogen". One line per CH_2 group is required to specify the atom types ICTCH2 and IHTCH2, respectively. Similar comments apply to the CH_3 constraints.

(12) The parameter IPB is redundant but must be present and set to 3.

5.4. The CHARGE input file

To allow more flexibility in the specification of atomic charges, *gmq* allows the user to override the charges assigned by atom types in the PARAM file by reading those contained in a file called CHARGE. The override is signalled by setting the variable LOGCHG=.TRUE. in the DATA.NEW file. The format of the CHARGE file is very straightforward with one record for each atom that is to have the charge overriden. Each line contains just the index of the atom and the required charge in e. An example follows:-

Example CHARGE file

```
      1   -0.5000
      2    0.3600
      3    0.3600
      4    0.3600
```

Notes on the format of CHARGE:-

(1) Everything is read-in in free format so spacing is not so important.

(2) The first integer is the index of the atom and is used to reset the charge on the corresponding atom. The index of the atom corresponds to the order in which atoms are read from the CONFIG.NEW file.

(3) If LOGCHG=.TRUE. in DATA.NEW but no file called CHARGE exists then an error message is printed and a template CHARGE.TEMP file written based on the charges found in the PARAM file.

5.5. The SHAPE input file

To allow the user to drive the size and shape of the system smoothly from that given in CONFIG.NEW to a different one, *gmq* provides the option to read in a new **h**/2 matrix from the file SHAPE. Of course, the **h**/2 matrix in CONFIG.NEW can be manually edited, and for very small changes this is quite possible, but often an instantaneous redefinition of the MD box can cause extreme forces to be generated and algorithmic breakdown. Therefore, if a change of size and shape is required it is best done by setting IMOV=4 in DATA.NEW and defining the new **h**/2 matrix in a file called SHAPE. This takes the same form as the three lines in CONFIG.NEW. Currently the user has no control over the rate at which the change is made. This is set internally to the maximum rate at which the elements of the h matrix can vary.

Example SHAPE file

```
 1.4E-09  0.  0.
 0.  1.5E-09  0.
 0.  0.  1.6E-09
```

Notes on the format of SHAPE:-

(1) Everything is read-in in free format so spacing is not so important.

(2) If IMOV=4 in `DATA.NEW` and the file `SHAPE` does not exist then *gmq* writes an error message and terminates.

5.6. The `DMOM` input file

To allow the user to obtain the correlation function for the total system dipole moment vector, $\vec{M}$, the possibility of storing $\vec{M}$ at each step is offered. The user simply has to create an empty file called `DMOM` before running *gmq*. The existence of the file `DMOM` is the signal to output the data to the file `DMOM1`. **NB.** The significance of the total system dipole moment vector is not always clear in a periodic system. $\vec{M}$ is nominally defined as

$$\vec{M} = \sum_{i=1}^{N} q_i \vec{r}_i \tag{5.6.1}$$

where q_i is the charge on an atom and $\vec{r}_i$ is its position vector. For charge-neutral molecules the vector $\vec{M}$ is independent of the origin of the coordinates. Thus for a periodic system only containing these sorts of molecules the vector is uniquely defined. For charged molecules and point ions there is an *origin dependence* which means the sum in Eq. 5.6.1 is ill-determined. At the moment the code omits point ions from the sum but charged molecules are counted. *It is strongly recommended that you only use this facility for systems of charge-neutral molecules*. Note also that, for the moment, the frequency of writing the data is set to each time step. The file `DMOM1` thus grows in size very rapidly. Be careful when using this option and keep an eye on the filestore usage. It should also be noted that the dipole moment vector can also be obtained by post-processing the `config.bin` file and, in either case, the list of $\vec{M}(t)$ can be analysed to create the total dipole moment correlation function (see the descriptions of the *gmq_dmom* and *dipcor* in the chapter on ancillary programs).

5.7. The `DMTA` input file

The `DMTA` input file can be used to impose oscillatory external pressure fields in a manner similar to dynamical mechanical thermal analysis. A required element of the external pressure tensor varies according to the following functional form

$$P_{\alpha\beta_{req}}(t) = P_{\alpha\beta_o} + \Delta P_{\alpha\beta} \, (\sin(2\pi\omega(t-t_o)) \tag{5.8.1}$$

where $P_{\alpha\beta_o}$ is the initial value (as read from `CONFIG.NEW`), $\Delta P_{\alpha\beta}$ is the amplitude of the oscillation, ω is the frequency of the oscillation, and t_o is the time origin for the imposition of the oscillation minimum value. An example of a `DMTA` file is given below in the case of an oscillatory shear experiment beginning at a point 100 ps into the run, i.e. the first 100 ps serve as a baseline, with an amplitude of 200 bars in the xy component.

Example `DMTA` file

```
100.0                    TZERO       (Time origin for oscillation in ps)
0.001                    FREQ        (Frequency of oscillation in ps**(-1))
  0.0   0.0   0.0        AMPREQ(I,I) (Amplitude of Pxx, Pyy, Pzz in bars)
200.0   0.0   0.0        AMPREQ(I,J) (Amplitude of Pxy, Pxz, Pyz in bars)
```

Notes on the format of DMTA:-

(1) Everything is read-in in free format. Only the first number of the lines for t_o (TZERO) and ω (FREQ) are read-in, similarly just the three first numbers are read-in for the $\Delta P_{\alpha\beta}$ (AMPREQ) required amplitude lines. In all cases the text afterwards is ignored.

(2) The units used are ps for t_o, ps^{-1} for , and bars for $\Delta P_{\alpha\beta}$.

(3) The required amplitude matrix is symmetrized at the start so that $\Delta P_{\alpha\beta} = \Delta P_{\beta\alpha}$; the atomic pressure tensor being symmetric.

(4) A check is made at the start of the simulation to make sure that t_o is less than the time into the simulation. If not an error is issued and the program stops.

(5) The existence of a `DMTA` file is only checked for if IMOV=1, i.e. N**P**T simulation.

(6) The values of DPREQ read from `DATA.NEW` are ignored in a DMTA run.

6. Beginning an MD simulation

With the three obligatory files (`CONFIG.NEW`, `DATA.NEW` and `PARAM`) prepared plus, if required, any of the optional files (`CHARGE`, `SHAPE`, `DMOM` etc.) it is now possible to attempt to run *gmq*. Access to *gmq* and all the utility and analysis codes etc. is usually accomplished by adding the following lines to the end of your `.cshrc` file.

```
setenv BHOME ~db_home
source $BHOME/cshrc.gmq
```

where `db_home` is the username where the *gmq* and associated files are kept (***ask DB***). It is prudent to use the `tcsh` as your default shell or, alternatively, `csh`. If you change your `.cshrc` file you it is best to logout and log back in again to make sure the commands in `cshrc.gmq` are executed. You can then check that the paths have been set properly by doing, for example, a

```
which gmq
```

This should return the absolute path name of the *gmq* program.

Assuming you have access to all the executable files, as a first step it is highly recommended that you execute the program *setup* which initialises the files `STO, ST, HS, PT, INDEX` and `config.count` which are used by *gmq*. These files will ultimately contain a history of previous runs but at the moment are in their initial state. The job can then now be executed just by typing "gmq". In the first instance it is as well to try one step first before proceeding to longer runs. The progress of the job can be followed by looking at the file called `STATUS`. This is updated at intervals of ISTO steps and simply reports the number of steps completed out of the total required.

6.1. Output from the run

If a job is successful the following files should be produced. These are presented in turn below.

6.1.1. `OUTPUT.NEW`

This is the log file. It should be examined carefully for warnings and error messages and checked against the input data. It prints all the information read from `DATA.NEW` and `PARAM` and certain information read from the `CONFIG.NEW` file. Certain other information evaluated from the input data sets is also printed at this stage such as long-range corrections to the energy and pressure. Instantaneous temperatures are printed both before and after the total linear momentum of the system has been zeroed. At the end of the run it prints information concerning the CPU time and the neighbour table and link-cells. It then gives a table with the following headings:-

```
STEP  TOTALE ANGP TORPE VdWLRC EWALD DENSITY TEMP PRESS TTEMP BONDP OOPPE
```

Most of these are self-explanatory, STEP is the time step into the run; TOTALE is the total energy; ANGP is the total angle bending energy; TORPE is the total torsional energy; VdWLRC is the total Van der Waals energy including any long range energy correction (energy and pressure corrections are based on the assumption that the radial distribution function, g(r), equals 1 beyond r_c); EWALD is the total Coulombic energy; DENSITY is the density in kg m^{-3}; TEMP is the temperature in Kelvin as measured from all degrees of freedom in the system; PRESS is the isotropic (scalar) pressure in bar, which is also long range corrected ; TTEMP is the temperature as measured from all the molecular centres of mass degrees of freedom (this calculation is an option in *ddgmq*); BONDP is the total bonding energy and OOPPE is the total out-of-plane energy. ***NB.*** **all energies are reported in Joules per mole *of atoms***. Unless the data is printed at every step, all quantities are sub-averaged over the number of steps corresponding to the interval specified by WHENST in `DATA.NEW`. Only a maximum of 100 entries are printed in the file `OUTPUT.NEW` but the full table is stored in file `STO1` (see next section). After the table appears the means of the above properties and the root mean-square deviations of the (possibly sub-averaged) data from these averages are given.

Afterwards various other averages are printed concerning the Ewald summation, pressure tensor, intramolecular contribution to the pressure tensor etc. In particular, the values of the direct and indirect calculations of the k-space contributions to the scalar pressure are given [18]. The comparison of these two values is an important indicator of the extent to which the Ewald sum is converged. The procedure for optimising the Ewald summation will be discussed in detail later.

Some further information is then output including instantaneous temperatures from the final run before and after the linear momentum is zeroed. In the case of simulations which allow the box size and shape to change it is as well to check the information given concerning the cut-offs, in particular to see if they have not been automatically reduced. This can happen if the box undergoes a contraction.

6.1.2. `CONFIG.NEW1`

This is the restart configuration file, i.e. it will be the `CONFIG.NEW` of any continuation run. The format is thus just as described for `CONFIG.NEW` except that some helpful comments are added.

6.1.3. `STO1`

This is a history file in text format containing in the first line the number of records to follow. There then follows a table of data with columns containing:-

```
TIME  TOTALE ANGP TORPE VdWLRC EWALD DENSITY TEMP PRESS TTEMP BONDP OOPPE
```

This is in the same order as in the `OUTPUT.NEW` file except that now the number of time steps has been converted to a time in ps into the simulation: the origin for this time is controlled by the variable read-in from the `CONFIG.NEW` file. The output frequency is set by the variable ISTO as determined from WHENST, DT and NOFSTP in the way described in the section on `DATA.NEW`. If any data exists in `STO` before the run then `STO1` contains the same information at the start and then data from the current run is appended. The first record will now contain the combined number of records.

6.1.4. `ST1`

This a history file in text format created in the same manner as `STO1` except that the information contained largely concerns the pressure tensor. After the first counter record, the following data, sub-averaged over ISTO steps, is given in columns:-

```
TIME    PXX    PXY    PXZ    BCR    PYY    PYZ    QDOT    PYY_req    PZZ
```

where TIME is again the time into the simulation and PXX, PXY, PXZ, PYY, PYZ, PZZ are the six independent components of the atomic pressure tensor given in Pa (1 Pa = 1 Nm^{-2} = 1 kg m^{-1} s^{-2} = 10^{-5} bar). For the three remaining columns BCR stands for "barrier crossing rate" in ps^{-1}, QDOT is the rate of input of kinetic energy in J mol^{-1} ps^{-1} and PYY_req is the value of the externally required yy component of the pressure tensor in Pa. Some further explanation of BCR and QDOT follows.

In *gmq* torsions are arbitrarily designated to be in a certain state based on the actual value of the angle. From -180° to -60° is considered as a *gauche minus* (G^-) state, from -60° to +60° is *trans* (T) and from +60° to +180° is *gauche plus* (G^+). N.B. *These states are attributed irrespective of the actual torsional potential so for non-threefold symmetry torsions they have little meaning*. This could be rectified by actual designating states according to maxima in the torsional potential which is perfectly feasible but has not been implemented as yet. In spite of this drawback, states are allocated at a frequency corresponding to the time given in the `ST1` file. At these times a count is also made of the number of torsion angles that have changed state, i.e. crossed a "barrier". The number of such transitions is then divided by the total number of torsions in the system and the time interval between counts (ISTO*DT). Angles which re-cross the barrier back to their former state in the intervening period are thus not counted. The maximum barrier crossing rate is thus attained using ISTO=1. The subtleties of this measurement and other relevant points are discussed in detail in [26-28].

The QDOT term output to the file `ST1` is an approximation to the average value of dQ/dt, the rate of heat energy input to the system by the thermal bath, over the ISTO steps composing each sub-average. This average is calculated simply from the cumulative amount of kinetic energy added to the system by the loose-coupling thermostat in the preceding ISTO steps and dividing by the time interval (ISTO*DT) in ps. Thus the total kinetic energy input to the system can be obtained by summing the QDOT and

multiplying by the total time interval. In this convention, a heating of the system gives rise to a positive value of QDOT.

6.1.5. HS1

This a history file in text format containing the elements of the **h**/2 matrix. After the first counter record, the following *instantaneous* data at intervals of ISTO*DT is given in columns:-

TIME $h_{11}/2$ $h_{12}/2$ $h_{13}/2$ $h_{21}/2$ $h_{22}/2$ $h_{23}/2$ $h_{31}/2$ $h_{32}/2$ $h_{33}/2$

where TIME is again the time into the simulation in ps and the elements of the **h**/2 matrix are given in Ångstroms.

6.1.6. PT1

This a history file in text format containing the instantaneous averages of the percentages of torsion angles in the *gauche minus* (G^-) and *trans* (T) states (see discussion of variable BCR output in the file ST1) for each torsion type defined within the PARAM file. The first record of PT1 contains the counter for the number of following records then a heading for each of the 2*NTTYPE columns, e.g.

```
  20        CCCC%G- CCCC%T  MCCC%G- MCCC%T
```

where the first four characters of each heading are formed from the first letters of the four atom types involved in the torsion. Each heading is then suffixed by %G- or %T accordingly. The actual conformational data is then given at ISTO*DT intervals with the time into the run being the first column.

6.1.7. INDEX1

This a history file in text format containing, in a compacted form, the instantaneous states of all torsion angles (G^-, T or G^+) (see discussion of variable BCR output in the file ST1) at time intervals of ISTO*DT. As there are, by definition, only three states for an angle, a variable $I_T(i)$ is set equal to 1, 2 or 3 corresponding to the state G^-, T or G^+, respectively, for each torsion angle in the system. The first 19 indices are then used to produce an integer using base three arithmetic,

$$\mathrm{INDX} = \sum_{i=1}^{19} 3^{i-1}\left(I_T(i)-1\right), \tag{6.1.1}$$

19 being the largest power to which 3 can be raised and still be less than 2^{31}-1, i.e. the largest 4 byte (32 bit) integer allowed. Subsequent blocks of 19 indices are then converted in the same way and then the whole sequence of INDX are written out in blocks of ten free format integers preceded by a line containing just the time into the run at which the indices were calculated. An example `INDEX1` file follows:-

```
 1500.0000
   581130733   624177535   195836336   582725084   581132920   581662174
   561998857  1097695759   451991218   581307880
   581130733   581209384   581111050   581130838   753315673   581130733
          40
 1500.0010
   581130733   624177535   195836336   582725084   581132920   581662174
   561998857  1097695759   451991218   581307880
   581130733   581209384   581111050   581130838   753315673   581130733
          40
```

It contains 17 integers for each time the data is written. This last number (40 in both cases) is smaller than the others as it is constructed from the final group of indices, which doesn't necessarily come to 19. A value of 40 can only be given by $1*3^0+1*3^1+1*3^2+1*3^3$ indicating the final sequence of angles in this example is T-T-T-T; the same basis conversion can be used to reconstitute all the states of the angles if required. So in this example, 16*19+4 = 308 torsion angle indices have been written at each time interval. The data contained in the `INDEX1` file can be analysed to produce the relaxation functions describing the rate at which interconversion takes place between the different conformational states [26-28] (see description of the analysis program *gmq_rfunc*). Of course, similar remarks apply as with the barrier crossing rate as regards the relevance if the barriers in the torsional potential don't match the definitions given for the different states.

6.1.8. `CONF1`

This is a sequential binary file used to create, in a further stage, the history file for the configuration. It is created so long as the flag ICFLAG is set equal to one in `DATA.NEW`. The intervals between writes of the configuration are 0 ps, RIPSTO ps, 2*RIPSTO ps etc. where the time is relative to the variable TIMER, read-in from the `CONFIG.NEW` file, and RIPSTO is read from the `DATA.NEW` file. Each record in the `CONF1` is actually written in the following manner:-

```
WRITE (26) OT,OD,OTEMP,OH,OPREQ,OX,OY,OZ
```

where OT is the time in ps, OD is the current density in kg m^{-3}, OTEMP is the temperature in Kelvin, OH is 3×3 **h**/2 matrix in metres, OPREQ is the 3×3 required pressure tensor in Pa and OX, OY and OZ are arrays containing the positions of the atoms in metres. All variables are 4 byte reals so as to cut down on the file store used. **NB** *This file is not intended for the user to work with*. In a separate program the data in `CONF1` is converted into a direct access binary format and appended to any previous configurational history of the system. This is discussed further in the next section with regard to continuing a simulation.

6.1.9. `STOSCR`, `STSCR`, `HSSCR` and `PTSCR`

These are the binary scratch files containing the data for the current run that is written during the course of the run by *gmq* before being eventually read back in and used to create `STO1`, `ST1`, `HS1` and

`PT1` at the end of the job. They are not intended for direct user examination although they can be useful sometimes in the event of a breakdown. A utility program provided for this purpose, called *readtmp*, will be described later.

6.2. Examining the output

Before proceeding with the simulation it is as well to check that nothing untoward has happened during the course of the current run. As mentioned above, the `OUTPUT.NEW` file should be carefully scanned for warnings and errors and any discrepancies between the input data and that read-in and printed out by *gmq*. It is often useful to plot the behaviour of the variables stored in the history files, particularly those in `STO1` and `ST1`. For machines supporting *gnuplot* there are scripts available which invoke it and plot the data in succession to give a good idea of how the simulation is proceeding; these will be described in the section on utilities. Otherwise the history files are in a format which can be easily read-in by other graphing programs. Later some practical guidelines will be discussed as to how to go about doing a simulation and how to assess if it is proceeding satisfactorily or not. This latter topic is still somewhat of a grey area in MD and is more than a little subjective in nature, what is satisfactory for one person may not be for another. Nevertheless, for the time being it is assumed that the user can make a decision from the output generated as to whether it is worthwhile continuing or not.

6.3. When it all goes horribly wrong

With the best will in the World, it is very difficult to produce a completely 100% bug free code. Although many fatal error conditions are already flagged before they can cause a disorderly breakdown of the code, it is very likely that there are others waiting to frustrate the uninitiated user. That said many of the common conditions which lead to fatal errors are already known.

6.3.1. Array bounds

The programs *gmq* and *ddgmq* are written using Fortran 77 and hence there are no means to dynamically allocate array bounds. If these prove to be a problem then, until the conversion to Fortran 90 is made, the user can contact DB to have them altered. In many cases the limits are either written to the `OUTPUT.NEW` file or printed out when they are transgressed.

6.3.2. Catastrophic algorithmic breakdown

A catastrophic breakdown of the algorithm or “blow-up” can take place for many reasons and is characterized often by high temperatures and pressures accompanied by wild fluctuations and the eventual crash of the program completely. Almost always such breakdowns are caused by too large displacements of atoms either due to too high forces, too high velocities or too long a time step. In cases where rigid constraints are used the problem shows up as an orderly fatal error exit from the SHAKE routine with a non-convergence message. SHAKE is the “pressure valve” in any dynamics code employing rigid constraints. Almost always the problem does not lie in the SHAKE routine *per se*. Normally the non-convergence results from too large displacements of atoms for the reasons stated above. These large displacements can very easily propel atoms to “free-flight” positions from where the forces of constraint cannot bring them back to positions where the constraints can be satisfied anew. There are many reasons

why the forces might be too high but two common ones are high energy initial configurations and errors in the `PARAM` file or connectivity given in `CONFIG.NEW`. If the initial configuration is at a high energy, choosing the energy minimization at constant volume, EM-V, option (IMOV=6) for a short run will quickly remove the excess energy from the system.

If the problem persists then a logical procedure at this stage is to start from the beginning and introduce the potential in stages to see where the problem lies. Start by checking the connectivity given in `CONFIG.NEW`. Switch off all interactions except the bonding potential or rigid constraints. Any bond lengths far from equilibrium will give a high bonding energy or lead to non-convergence of SHAKE. If rigid bonds are used and SHAKE cannot converge then the harmonic bonding potential can be used to bring the bond lengths closer to the equilibrium value before switching back on the constraints. This is best done again using, at least initially, EM-V conditions. This should be first tried with all the other potentials switched on otherwise it may cause further problems by introducing overlaps between non-bonded atoms. To speed up this procedure it is always recommended that a short real space cutoff be used as problems are almost always caused by large forces being generated in the short range part of the non-bonded potential. One way to do this neatly is to use the WCA potential (IPOT=2) and just the real space part of the Ewald summation. Setting ALPHA equal to a fairly high value, say 0.7, gives a very short range smoothly decaying potential which can be truncated around 5 Å. In certain cases the Ewald summation can be completely omitted but be warned, in certain potentials there are no explicit VdW non-bonded terms, e.g. for hydrogen-hydrogen and hydrogen-oxygen non-bonded interactions in simple water models. In these cases it is essential to maintain the real space part of the Ewald sum.

If, on the other hand, the bonds are reasonably close to their equilibrium positions then one can proceed to introducing the bending potential and then the torsion potential. Bends far from equilibrium can cause problems and sometimes even torsions but the latter are usually of relatively too low an energy to cause a blow-up. If there are high energy terms then once again the energy minimization procedure can be used.

A more likely result of a problem with the connectivity will be felt when the non-bonded potential is introduced. Two atoms which are linked together do not interact through the non-bonded potential, however, if the connectivity of the two atoms has been omitted from the `CONFIG.NEW` file and the atoms are close together then a blow-up is the very likely outcome of any simulation begun with the non-bonding potential switched on. Of course, this may not be immediately be distinguishable from a severe overlap between two atoms truly interacting through the non-bonded potential. This can sometimes be distinguished by selectively removing some of the non-bonded interactions. In certain cases, however, this is the inevitable result of the method used to generate the initial configuration and the EM-V option should be used to remove energy from input configurations with severe overlaps.

7. Continuing an MD simulation

7.1. General considerations

Assuming everything goes well with the initial run, it is very likely the user will wish to continue simulations as seamlessly as possible. In principle, it is possible to do an entire simulation in one go but often there are limitations in terms of the amount of CPU time that can be used per job or to the stability of the computer. It is generally wise to break the simulation into smaller pieces and do a number of continuation runs. This way the simulation can be checked intermittently to view its progress and terminated if things go wrong and, on an unaccounted machine, one doesn't run the risk of running into a system crash and having to sort out the mess of a disorderly end to the job.

With *gmq* the philosophy is that all the input files remain unaltered after a run. This way a repeat run of the same *gmq* executable binary should produce the same results, except for CPU timings printed in `OUTPUT.NEW`, so long as no change has been made to the input data. There are no randomizing factors in *gmq* so the trajectory and hence the output must be identical. *The only exceptions to this rule have so far been shown to be caused by hardware faults*. Of course, different versions of *gmq* may well produce slightly different output but all new versions are checked against some standard jobs and only accepted if the differences can be attributed to different round-off and truncation errors or are reasonable considering the change in the code. All users are encouraged to have there own standard short tests so as to be able to verify that for their systems new versions of *gmq* are producing results consistent with previous ones. Any discrepancies should be immediately reported to DB.

7.2. Text file manipulations

To continue a job, whilst preserving the entire history, the *SWAP* shell script should be executed in the directory containing the job. This *SWAP* shell script can be always be examined on-line to see the file manipulations that have to be done before continuing a run.

7.3. The configurational history file `config.bin`

To build up a history of the configuration as a function of time it is necessary to append the data stored in the sequential binary file `CONF1`, see section 6.1.8, onto any previous history file. To make this procedure more efficient and to provide a format with which specific configurations can be readily accessed this data is stored on a *direct access* binary file called "`config.bin`". The conversion of `CONF1` and the appending of `config.bin` are performed by the program *polyconb*; the shell script *SWAP* invokes *polyconb* automatically. The current number of the records in the `config.bin` file is kept on the first line of the text file "`config.count`". Initially this is set to zero when created by the program *setup*. After the first run of *polyconb* the second line of `config.count` contains the number of atoms in each record.

To open a direct access binary file it is necessary to know the length of each record to be written. Unfortunately some machines use *bytes* for counting others use *words*. For this reason it is necessary for *polyconb* to know the type of machine on which it is running. At the moment this is accomplished by

setting the environment variable DBVAR before ever running *polyconb*. Normally this is set when the `cshrc.gmq` script file is sourced by your `.cshrc` file. The setting can be checked using the Unix command "setenv".

Environment variables can be accessed within Fortran programs using the GETENV function and so the type of machine can determined internally and the switch set for bytes or words in the definition of the record length. Within *polyconb* then the binary direct access file is opened in the manner shown below where IYES=1 implies words are used as the unit:

```
IF (IYES.EQ.1) THEN
   IRECL = (1+1+1+1+9+9+3*NUMA)
ELSE
   IRECL = (1+1+1+1+9+9+3*NUMA)*4
ENDIF
OPEN (2,FILE='config.bin',FORM='unformatted',ACCESS='direct',
&     RECL=IRECL)
```

On the machines using bytes then a factor of four appears in the record length (IRECL) as the data to be written are 4-byte words. The length of the record is determined by the data written:

```
WRITE (2,REC=NREC) NREC,TIMER,DENSIT,TEMP,H,PREQ,X,Y,Z
```

Here NREC the integer record number, TIMER is the time into the run in ps at which the data was written, DENSIT is the density of the system in g cm^{-3}, TEMP is the temperature in Kelvin, H is the **h**/2 3×3 matrix in Ångstroms, PREQ is the 3×3 required pressure tensor in bar and, finally, X, Y and Z are arrays of length NUMA containing the coordinates of the atoms in Ångstroms.

Note well that some unit conversions have taken place on going from `CONF1` to `config.bin`.

Provided this conversion/append takes place error free and the other exchanges have been made, the continuation run can now be carried out. The user is free to change any of the variables in the `DATA.NEW` and `PARAM` input files. However, if the time between configuration writes is changed then some of the analysis programs will not be able use all of the data to perform averaging over time origins, say. Changes in the `CONFIG.NEW` file regarding the number of atoms should not be made within the context of a continuation run as the `config.bin` file does not allow for a record length varying with the number of the record.

With the above procedures the user has a certain amount of flexibility in terms of which data to keep and which to erase. If just tests are being made on the stability of the algorithm then it is often useful to run the same simulation from the same configuration using a different time step. In this case it is useful just to build up a history of the behaviour of the energies by just moving `STO1` onto `STO` at the end of each trial, changing DT in `DATA.NEW` and then re-running. Often a production simulation will be preceded by a certain amount of equilibration. This in itself may be interesting but may want to be discarded. This can be achieved simply by running *setup* to zero the history files, erasing `config.bin` and then, optionally, re-setting the time counter to zero in `CONFIG.NEW`.

8. Optimising the time step

This is a difficult issue to discuss in any absolute terms. All that can be given are some tips on how to assess the stability of the algorithm with a certain time step and some rough guidelines concerning

the final choice. The choice of time step is obviously an important point as too small a time step will increase the computing costs significantly whereas too long a time step may cause algorithmic breakdown or lead to systematic errors in the results.

One procedure that can be used is to first "equilibrate" the system to be simulated at the chosen temperature and density and then to perform a series of short NVE simulations starting from the same configuration but using different values of the time step, Δt. This sounds fairly straightforward but some care has to be taken. For example, truncation of the LJ potential leads to discontinuities in the energy which mask the underlying degree of stability in the integration algorithm. It is thus advisable to perform tests in the absence of any such uncertainties in the first instance. Very often the limit of stability of the algorithm is governed by the high frequency intramolecular modes in the system so a reasonable starting points for these tests is the "equilibrated" system in the absence of all the non-bonded potentials. In Figure 8.1 some typical results are shown for short 1 ps runs from the same initial configuration of a model water system which was first fully relaxed with all potentials switched on. The `PARAM` file for the fully interacting system follows:

```
***** BIOSYM water molecules ****
            2  No._of_different_atom_types
 **** Label     Mass/(g/mole)     Charge/e
    Hydrogen       1.000797D0       0.41D0
    Oxygen        15.9994D0        -0.82D0
            1  No._of_different_types_of_bonds
 ****  Type_1  Type_2     Bond_Length/A   Kb/kg/s**(-2)
         1       2          0.96D0        751.2320278
            1  No._of_different_types_of_bond_angles
 ****  Type_1  Type_2  Type_3  Theta_0     K/(KJ/mole)
         1       2       1     104.5D0     446.3838702
            0  No._of_different_types_of_torsions
            1  No._of_types_of_non-bonded_interactions
 ****  Type_1  Type_2      Sigma/A        Eps/k/K
         2       2         3.165520088    78.20812888
```

The parameters were taken from a model of water used in the BIOSYM package. It is similar in style to other three-site models such as SPC and TIP3P. The system equilibrated contained 263 molecules in a periodic cube of side length 20 Å. For the tests shown in Fig. 8.1 all non-bonded potentials were switched-off, i.e. no LJ interactions and no Coulombic interactions. The system thus evolves simply under the influence of the bonding and bending potential. The time variation for the total energy of the NVE simulations of duration 1 ps are shown in Fig. 8.1.

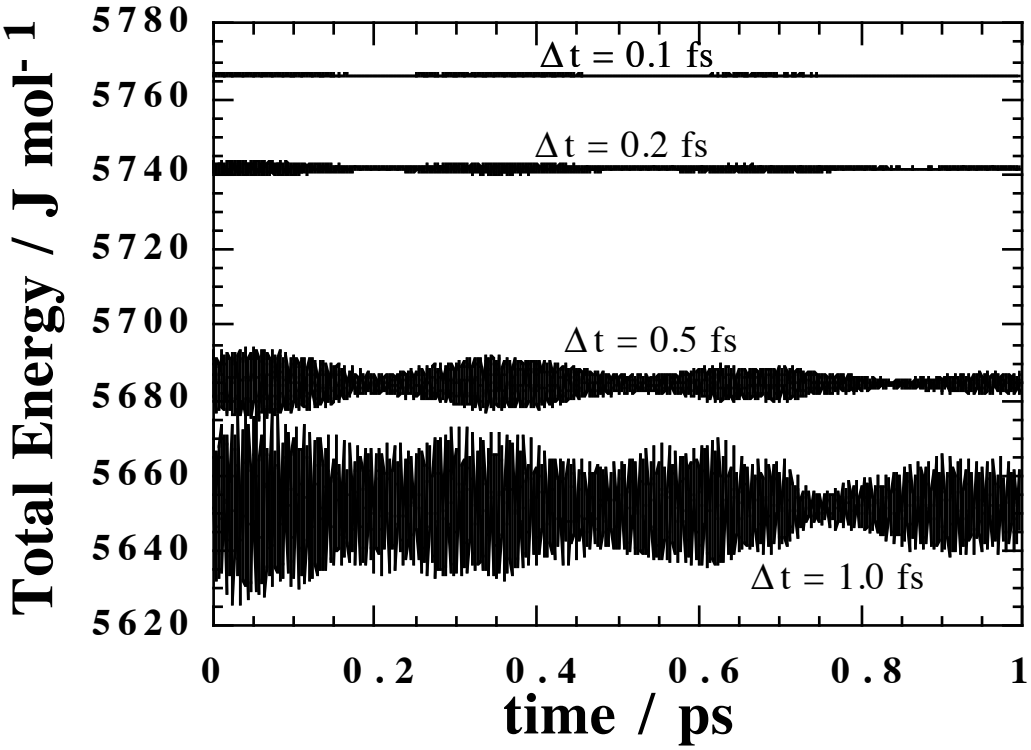


Figure 8.1 *The variation of the total energy as a function of time for the four NVE simulations carried out from the same initially equilibrated configuration with four different time steps.*

The first thing to note about Fig. 8.1 is that the total energy fluctuates about a different mean for each value of Δt. This is to be expected. It stems from the calculation of the kinetic energy at the first step; the potential energy has to be the same whatever the time step, as it is calculated from the coordinates as read-in. The on-step kinetic energy in the leapfrog algorithm, however, is only known once the forces have been calculated and the trajectories integrated forward one step. Thus the actual value of the time step enters into the calculation of the first kinetic energy. In fact it is simple to show that the initial kinetic energy, $K_e(t=0,\Delta t)$, varies as

$$K_e(t=0,\Delta t) = A + B\,\Delta t + C\,\Delta t^2 \qquad (8.1)$$

with A, B and C constants for a particular input configuration. Figure 8.2 shows a plot of $K_e(t=0,\Delta t)-K_e(t=0,1)$ for the same initial conditions as in the tests described above.

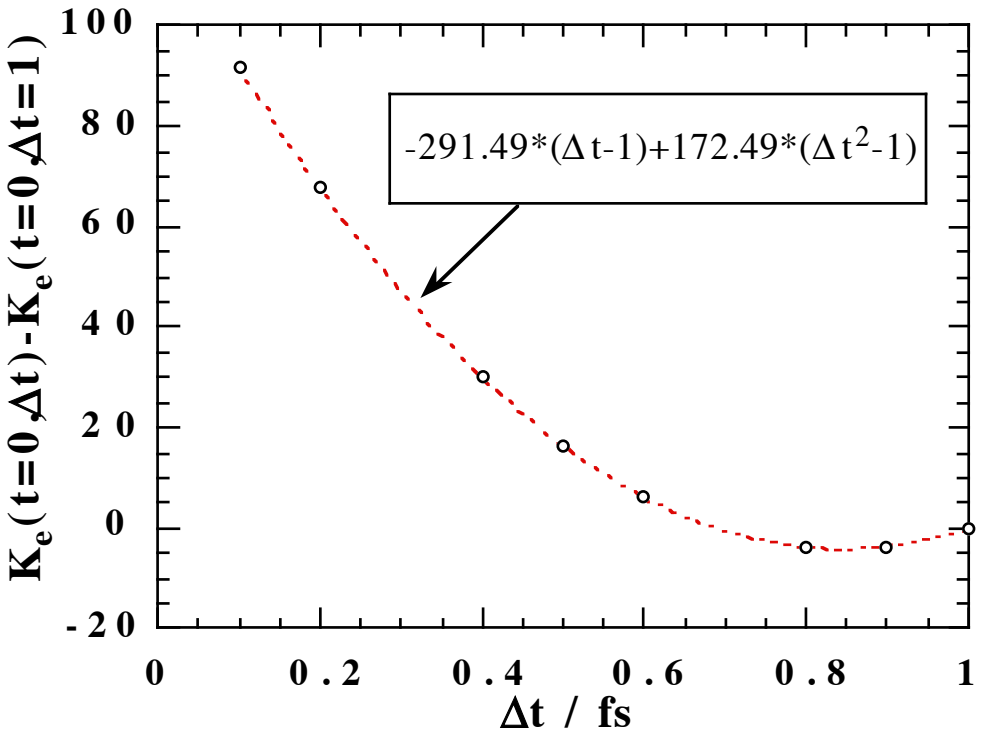


Figure 8.2 *The difference between the initial kinetic energy obtained for various time steps compared to that obtained for a Δt=1 fs from NVE simulations carried out from the same initially equilibrated configuration.*

The second thing to note in Fig. 8.1 is that the fluctuations in total energy increase with time step size. This too is expected and again it is simple to show that for the leapfrog algorithm these fluctuations increase in proportion to Δt^2. In Figure 8.3 the root mean square fluctuations in the total energy for the tests shown in Fig. 8.1 are plotted as a percentage of the corresponding root mean square fluctuations in the total potential energy; this kind of normalisation gives some standard measure of the level of energy conservation. Also plotted in Fig. 8.3 are the equivalent set of results for a system in which all the bonds have been rigidly constrained, i.e. the highest frequency motions have been removed. In both cases the slope of the log-log plot is close to two showing the expected dependence of the fluctuations on Δt^2.

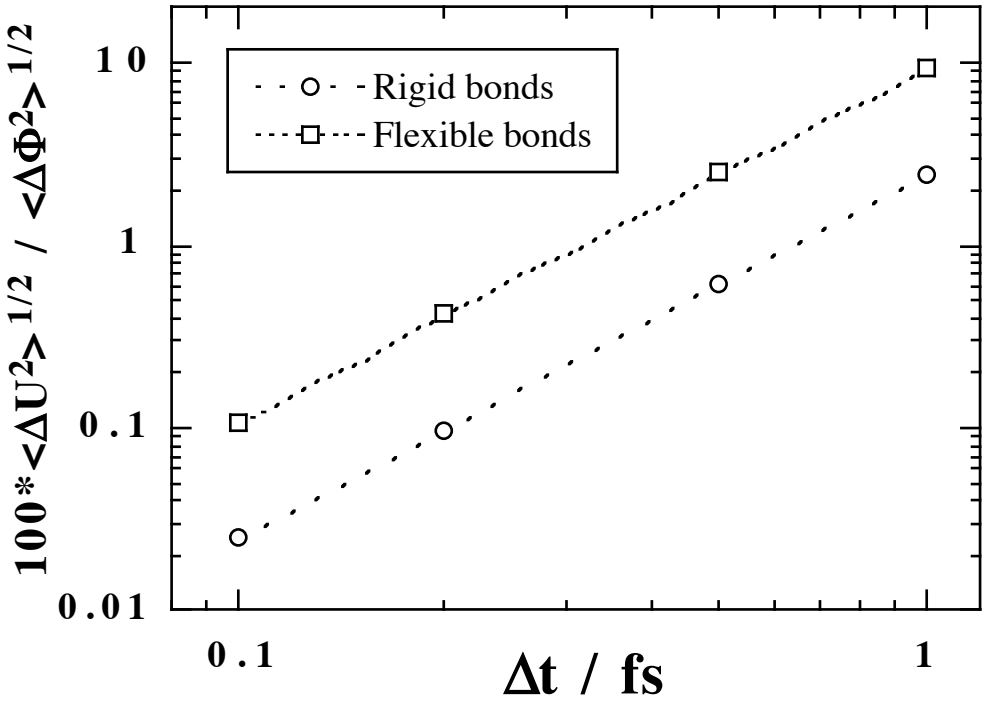


Figure 8.3 *The fluctuations in the total energy plotted as a percentage of those in the total potential energy for NVE simulations carried out from the same initially equilibrated configuration of a water model with four different time steps. The results from the original flexible bond model are compared to those where the bonds are held fixed by rigid constraints at the equilibrium value.*

Fig. 8.3 shows clearly the beneficial effect removing the high-frequency low-amplitude bond stretching motions has on energy conservation. In this particular example, a level of energy conservation can be maintained with a time step twice as large for the rigid bond case. Other systems will differ so it is worth checking on this for each individual case.

The question now arises as to which time step is appropriate? One answer to this question is the largest time step which will not cause systematic errors in the properties we are interested in. That is to say, if we could achieve converged results for a set of properties using different time steps then, within errors, below a certain threshold time step we should get the same answers. One thing that is not possible is to have the system follow the same trajectory through phase space using different time steps. It is well known that MD trajectories diverge exponentially away from each other in this respect following no matter how small an initial perturbation. Certainly changing the time step is one form of obvious perturbation, as are altering one or more of the initial coordinates or velocities even by the smallest amounts. Other more subtle perturbations are running the same MD code on different types of computer or even running the same code on the same machine having compiled it at a different optimisation level.

One other relevant source of perturbation is running *gmq* and *ddgmq* on the same initial configuration. The parallel code accumulates the forces in a different order and does the constraints in a different order and this is more than sufficient to cause the trajectories to diverge.

The issue of trajectory divergence has been around a while now in MD and the standard texts give more information and references. On the whole, it should not matter as even though trajectories diverge from each other they should all sample the same phase space. Even the most cynically minded would grant that MD can be seen as, at worst, just another way of doing Monte Carlo sampling. Certainly it raises questions regarding the calculation of time correlation functions out to times beyond which the trajectory has long since departed from the "true" one. This latter point may be less important as at long times correlation functions tend to be overwhelmed by the statistics of small samples in any case. A more likely "problem" is the possibility of systems becoming trapped in different regions of phase space. An example might be a system close to a phase boundary. In one simulation the boundary might by chance be crossed and the system crystallizes, say. In another simulation, starting from the same initial configuration but perturbed in some way, it might well avoid the crystallisation process and remain liquid for the limited amount of time that can be simulated. Similar examples exist in the case of protein simulations [29].

So returning to the question of what is an appropriate time step; the ideal solution, of running several production simulations using different time steps, is a costly affair for all but the simplest of systems. In the majority of cases this procedure won't be possible, therefore, some approximate guidelines have to be used. One common guideline based on experience is to use a time step which limits the fluctuations in the total energy with respect to those in the total potential energy to "a few percent". Thus, from Fig. 8.3 one might reject a time step of 1 fs in the case of flexible bonds, 0.5 fs probably being more reasonable, but accept it in the case of rigid bonds. Another criterion would be the drift in total energy. For the examples shown in Fig. 8.1 the level of drift is difficult to perceive. If it were visible on such a short time scale then it would be a cause for concern.

Having checked the conservation of the energy in the case where the non-bonded potential is not operating it is worth then repeating the tests with the non-bonded potential switched-on. To avoid the discontinuities in the truncated LJ potential it is advisable to use the WCA potential for such tests. These tests should first be conducted in the absence of the Coulomb potential. The last stage will be to introduce the Coulombic potential but that first requires some discussion of the tuning of the Ewald summation parameters, which follows in the next section.

9. Choosing parameters for the Ewald sum

In *gmq* there are three parameters (ALPHA, KMAX and CUTQ) read from the `DATA.NEW` file which control the Ewald summation. These correspond to the variables α, K_{max} and R_{max} discussed in Section 3.1. In principle, the Ewald summation should converge for a range of different values of the α parameter controlling the relative amounts of the sum calculated in real and reciprocal space. However, practical considerations place some restrictions on the range of α for which convergence can be attained. As explained before, in *gmq* the maximum real space cutoff that can be used is half the minimum

perpendicular distance between any two sides of the periodic box. This effectively provides the lower limit for the value of α for which the Ewald sum converges. Although in theory there is no upper limit for α, it is controlled in practice by the setting a maximum for the largest value of KMAX that can be used in *gmq*. Typically this is ~20 but can be changed on demand.

9.1. Procedure for initial choice of α, K_{max} and R_{max}.

To obtain a reasonable value for α the procedure of Smith [30] can be used. Again starting from an "equilibrated" configuration of the system of interest a number of 1 step simulations are carried out using different values of α but with R_{max} and K_{max} fixed; the shellscript *scan_Alpha* exists for automating this procedure (See Section 15.9). Providing these real-space and reciprocal-space cut-offs are large enough there should be a range of α values for which consistent results are obtained for the Coulombic energy, U_c. To illustrate this approach a series of tests have been carried out on a model system containing N=500 ions. Each ion has a charge of ±1e and the system is electrically neutral. To prevent overlap, particularly of oppositely charged species, the short range Weeks-Chandler-Andersen (WCA) purely repulsive potential is used. The actual values of the potential parameters were ε/k_B=327 K and σ=5.881 Å. Configurations were equilibrated at T^* ($=Tk_B/\varepsilon$)~3 and ρ^* ($=N\sigma^3/V$) =0.85. The integration time step was $\Delta t^*=\Delta t(\varepsilon/m\sigma^2)^{1/2}$=0.001 and the value of the shell width used was Δr_c=0.17σ.

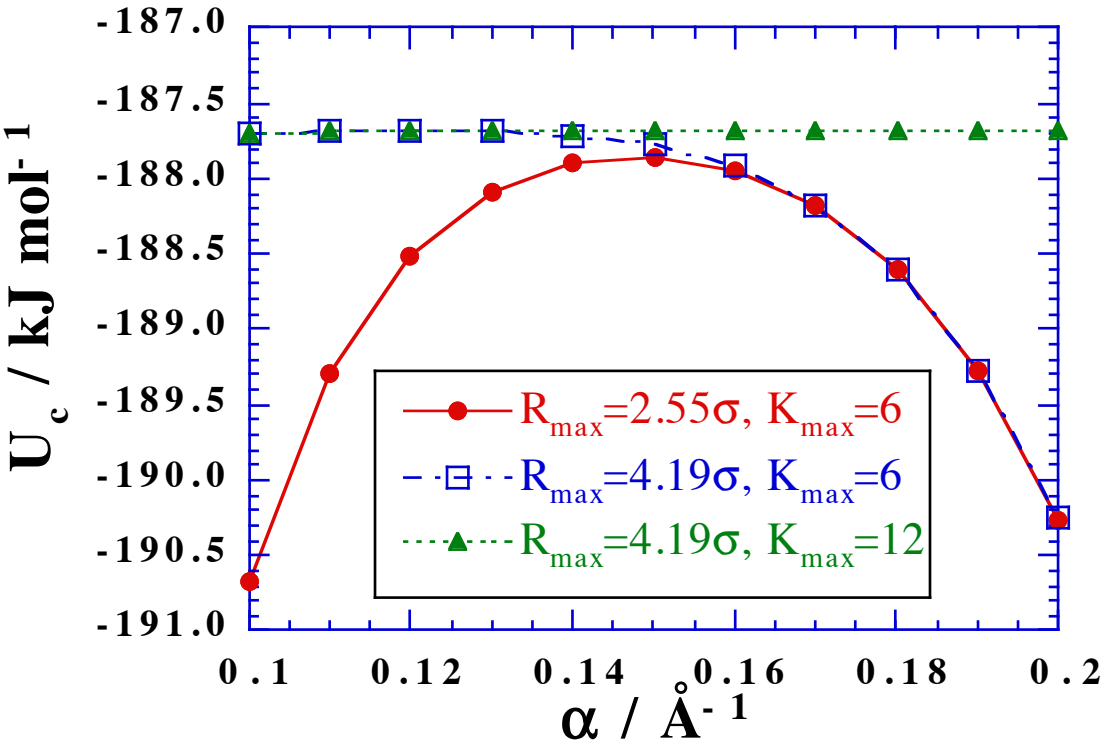


Figure 9.1.1 The dependence of the Coulombic energy, as evaluated using the Ewald method, as a function of the α parameter for a system of N=500 ions. Values used for the real- and reciprocal-space cut-offs are given in the legend

Fig. 9.1.1 shows some typical results for just one configuration of a molten salt containing N=500 ions. The effect of varying α is clear to see. As α increases more of the sum has to be done in reciprocal space and vice-versa. A value around α=0.15 Å^{-1} seems to allow relatively small values of R_{max} and K_{max} to be used whilst maintaining good precision.

Note well that each different system will require its own separate tuning of the Ewald parameters. The parameters optimized for one system are not guaranteed to work in a different one. Furthermore even if the same system is simulated at the same density with a larger number of atoms it still may not be correct to use the same values. This is because the convergence of the reciprocal space sum depends on the actual magnitude of the largest ***k***-vector, which is related to K_{max} via Eq. 3.1.5. For example, if a cubic system of side length S was being used then Eq. 3.1.5 reduces to

$$\vec{\boldsymbol{k}} = \frac{2\pi}{S}\begin{pmatrix} l \\ m \\ n \end{pmatrix} \tag{9.1.1}$$

Now if we were to double the side length of the cubic system, i.e. increase the system size by a factor of eight, then for a fixed value of α, K_{max} should nominally be doubled to have the same level of convergence in the reciprocal space sum. Of course, doubling K_{max} seriously affects the efficiency of the Ewald method as the number of ***k***-vectors increases with $K_{\max}^3$. For this reason it is often advocated that if system size is increased all the Ewald parameters be readjusted to minimize the CPU time consumed. A typical analysis is given in [31] and leads to the conclusion that the Ewald method scales as N$^{1.5}$ for scalar programs. For the parallel version of the Ewald sum in *ddgmq* other considerations enter into the optimisation and these will discussed later.

9.2. Run time monitoring of convergence

In *gmq* a check on the convergence of the Ewald sum is provided in the form of a comparison between two calculations of the Fourier space contribution to the hydrostatic pressure , p_c^F, [18]. p_c^F can either be calculated directly from the expression for the contribution of the Fourier space part of the Ewald summation to the pressure tensor ,

$$p_{c\ \text{direct}}^{F} = \frac{1}{3}\,\text{Tr}\,\{\mathbf{P}_c^F\}, \tag{9.2.1}$$

or indirectly from the Coulombic energy and the real space contribution of the Ewald sum to the pressure tensor,

$$p_{c\ \text{indirect}}^{F} = \frac{1}{3V}U_c - \frac{1}{3}\,\text{Tr}\,\{\mathbf{P}_c^{\text{R}}\}. \tag{9.2.2}$$

where U_c is simply the total Coulombic energy. The latter result can be explained by the following analysis. The Coulombic contribution to the hydrostatic pressure is defined as

$$p_c = \frac{P_{c_{xx}} + P_{c_{yy}} + P_{c_{zz}}}{3} = \frac{1}{3V}\sum_{i=1}^{N}\sum_{\substack{j>i \\ j \text{ not 'bonded' to } i}}^{\infty} \vec{r}_{ij} \bullet \vec{f}_{ij}^{\,c} \tag{9.2.3}$$

where, in the sum over *j*, only those atoms with which *i* interacts through the Coulomb potential are included. The *1/r* nature of the Coulombic energy means that the number of *j* to be taken in a 3D periodic system is effectively infinite. However, the Coulombic force on *i* due to *j* is given by

$$\vec{f}_{ij}^{\,c} = \frac{d\Phi_c\left(\left|\vec{r}_{ij}\right|\right)}{d\vec{r}_i} = \frac{q_i q_j}{4\pi\varepsilon_0\left|\vec{r}_{ij}\right|^3}\vec{r}_{ij} \tag{9.2.4}$$

so substituting for $\vec{f}_{ij}^{\,c}$ in Eq. (9.2.3) returns the familiar result,

$$\begin{aligned} p_c &= \frac{1}{3V}\sum_{i=1}^{N}\sum_{\substack{j>i \\ j \text{ not 'bonded' to } i}}^{\infty} \vec{r}_{ij} \bullet \vec{f}_{ij}^{\,c} = \frac{1}{3V}\sum_{i=1}^{N}\sum_{\substack{j>i \\ j \text{ not 'bonded' to } i}}^{\infty} \frac{q_i q_j}{4\pi\varepsilon_0\left|\vec{r}_{ij}\right|^3}\vec{r}_{ij} \bullet \vec{r}_{ij} \\ &= \frac{1}{3V}\sum_{i=1}^{N}\sum_{\substack{j>i \\ j \text{ not 'bonded' to } i}}^{\infty} \frac{q_i q_j}{4\pi\varepsilon_0\left|\vec{r}_{ij}\right|} = \frac{1}{3V}U_c \end{aligned} \tag{9.2.5}$$

Of course, U_C is calculated via the Ewald summation method, so a comparison of p_c^F derived from Eqns. 9.2.1 and 9.2.2 is a good check on the convergence.

The averages of these two values for p_c^F are printed in the `OUTPUT.NEW` file. Differences between them should certainly be no greater than the typical statistical errors expected in the pressure. No universally applicable figures can be given for the difference as this will be system dependent. However, tests of the Coulombic energy convergence for a single configuration can be used as a guide to the level of convergence expected in the p_c^F. For the crystalline polyethylene oxide example given in [18], a 5 figure convergence in the Coulombic energy translated to a less than 1 bar difference. In subsequent simulations the discrepancy in the p_c^F should be carefully monitored, especially during equilibration and particularly if the **h** matrix is changing. A significant increase in the difference indicates a re-tuning of the Ewald parameters is required.

10. The parallel MD code *ddgmq* (v7.9 - 19/07/2019)

The *ddgmq* code is a parallel version of *gmq* developed within the domain decomposition framework. To give the user an overview of this approach a brief review of some of the points the user has some measure of control over is given below. The differences in the two codes and the operation of *ddgmq* are then specified.

10.1. Domain decomposition

The domain decomposition method used in *ddgmq* is based on methods developed in the past for atomic systems [32] and linear chain systems [25] but which have been recently adapted so as to be able

to simulate more complex molecular geometries [33]. The interested reader is directed to these articles for detailed information.

The decomposition of space used in *ddgmq* is the same as described previously [25] in that initially atoms are assigned to $N_{p_x} \times N_{p_y} \times N_{p_z} = N_p$ processors on the basis of their x, y and z coordinates, i.e. 3-D decomposition. The exact shape of the MD box does not have to be orthorhombic and assignments are carried out using the set of scaled coordinates $\mathbf{s}_i = \mathbf{h}^{-1}\mathbf{r}_i$ where the matrix $\mathbf{h}=\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ is defined from the basis vectors of the MD box. The scaled space is cubic and facilitates easy assignment of particles to processors. The scaled space also facilitates the assignment of atoms to the $N_{c_x} \times N_{c_y} \times N_{c_z} = N_c$ link-cells within each domain. At this point each processor element (PE) is only aware of particles in its own domain. As atoms are assigned to processors it is inevitable that some molecules will be distributed among two or more of them.

Once the assignments to domains and link-cells have been made, the performance of a search for all interacting pairs of atoms can be carried out. This requires the communication of those coordinates required by other domains to complete their "working" environment. Details of the modification of the conventional link-cell search pattern which minimise the amount of communication at this stage have been given previously [32]. The build up of the subject domain's working environment of link-cells requires three communication stages in the three perpendicular directions. The convention we have used previously is that the neighbouring PE in the -y (“southward”) direction is called South, that in the +x (“eastward”) direction is called East and that in the +z “upward” direction is called Above.

Following this initial communication stage, each processor has access to enough of the coordinate information to perform its share of the pair search. This initial pair search not only allows the interaction of non-bonded pairs but also the formation of neighbour tables of non-bonded pairs. These neighbour tables are used in subsequent steps until deemed redundant. As in the scalar case, redundancy occurs automatically when any site has diffused a distance greater than or equal to half the shell width since the last rebuild. Note also that only at this stage is it necessary for sites to get reassigned to different PEs should they have diffused out of their previous home PEs, so as to ensure sites appear in their correct link-cells for the next rebuild. Subsequent steps are handled by the neighbour tables and so it is irrelevant if a site wanders out of the region that its home PE is nominally responsible for.

The inclusion of arbitrary multi-particle potentials and constraints into the above framework is covered in detail in [33]. More communication is required at the list-building steps but subsequent steps require no more communication than that required for the pair search. The constraints are handled in an iterative manner in the same way as in *gmq* but in this case communication is required at each SHAKE iteration.

10.2. User considerations

Whether to use *ddgmq* or *gmq* depends, apart from the availability of a parallel computer, presently very much on the size of the system in relation to the largest potential truncation radius. In its current form *ddgmq* requires that each domain contains at least 2×2×2 link-cells. This is because communications are only carried out with nearest neighbour domains. *ddgmq* can be run on one

processor, so long as there are 3×3×3 link-cells in the system, and so the output can normally be checked against that produced by *gmq* and this is always recommended. If more than one processor is to be used then currently the system has to be large enough in that dimension chosen. The user can control the decomposition by specifying the number of processors $\{N_{p_x},N_{p_y},N_{p_z}\}$ along each dimension. Some thought has to go into the choice of values for the non-bonded cut-offs. In turn this affects the values chosen for the Ewald summation parameters. In some circumstances it can be advantageous to trade a shorter real space cutoff in the Ewald sum for a larger k-space cutoff in order to increase the possible number of domains.

10.3. Differences between *gmq* and *ddgmq*

Although it is the intention to have the scalar and parallel versions of the *gmq* code as compatible as possible there are at present a few differences which are listed below:-

(1) The calculation of the temperature from the centre-of-mass translational degrees of freedom is an option in *ddgmq* (see specification of `DATA.DD` file in next section). This is so as the calculation involves operations which can cause some degradation of parallel performance.

(2) The output file `INDEX1` is not written in *ddgmq* (an `INDEX` file can be instead generated from the `config.bin` file using *gmq_index*).

(3) The barrier crossing rate is not calculated in *ddgmq*.

10.4. The `DATA.DD` input file

In addition to the input files required by *gmq*, *ddgmq* requires one other file called `DATA.DD` which controls those aspects of the parallel version accessible to the user. The format of the `DATA.DD` file is:-

Record 1: NPX, the number of processors in the x-direction.
Record 2: NPY, the number of processors in the y-direction.
Record 3: NPZ, the number of processors in the z-direction.
Record 4: LOGCOM, a logical variable controlling whether the COM temperature is calculated or not.

An example `DATA.DD` file follows.

Example `DATA.DD` file.

```
2       NPX (No. of PEs in X direction)
1       NPY (No. of PEs in Y direction)
1       NPZ (No. of PEs in Z direction)
T       LOGCOM (Calculate COM temperature? T or F)
```

10.5. The `EXTRA` file

For certain large systems it is sometimes useful to be able to store the coordinates of certain atoms at frequent intervals. Storing all the coordinates at a frequent intervals can quickly lead to a filling up of the file store. To enable the storing of coordinates of certain user-specified atoms at certain intervals a file called `EXTRA` can be created in the directory where *ddgmq* is to be executed. The format of the `EXTRA`

file is straightforward. On the first line is the desired interval between writing the coordinates in picoseconds. There then follow, at one entry per line, the global indices of the atoms that are required to be written out. The `EXTRA` file is read until the end-of-file record is found. The number of atoms read-in, NEXSTO, is written to the text file called `EXTRA.COORDS1`, along with the associated density, time into the simulation, temperature, **h**/2 matrix and coordinates. The exact format is the following:-

```
NEXSTO, time, density, temperature
h11/2, h12/2, h13/2
h21/2, h22/2, h23/2
h31/2, h32/2, h33/2
i, xi(t),  yi(t),  zi(t)
etc. for all other atoms in the list.
```

with time given in ps, density in g cm^{-3}, temperature in Kelvin and **h**/2 matrix components and coordinates in Ångstroms. For continuation runs the *SWAP* script appends the data in `EXTRA.COORDS1` to the end of the file `EXTRA.COORDS` to produce a history file for the entire simulation.

10.6. Running an MD simulation using *ddgmq*

So long as all the input files required by *gmq* are in place, then to run *ddgmq* just requires the extra file `DATA.DD` just described. As the program uses the MPI message passing library to do the communications it is necessary to invoke the execution of *ddgmq* using the correct command. As each type of machine has its own way of doing this it is necessary to contact DB first.

11. Using the Buckingham form of Van der Waals potential - *gmq_buck*

The choice of which Van der Waals potential is used is governed in the first instance by the name of the executable. To utilise the Buckingham VdW potential the user should invoke the program using *gmq_buck*, or *ddgmq_buck* for the parallel version. The VdW potential parameters in the file `PARAM` have to be specified accordingly and the meaning of the variables IPOT and IMIX in the `DATA.NEW` file also changes.

11.1. Specification of `PARAM` file

Apart from the specification of the VdW terms, everything else remains as described in Section 5.2. The format for the VdW terms when IPOT=1, i.e. Buckingham potential, is:

```
     3  No._of_types_of_non-bonded_interactions
**** Type_1  Type_2     A_Buck/eV         B_Buck/A     C_Buck/eV*A**6
       1       1       2528.07956         0.24849       8.33234
       1       2       1861.69156         0.27550      15.29926
       2       2       1370.95981         0.30251      28.09145
```

with the A, B and C parameters of the Buckingham potential, Eq. 2.1.3, being given in eV, Ångstroms and eV Å^6, respectively. The choice of electron volts as the unit of energy (1eV=1.60217733×10^{-19} Joule) at this point is simply to maintain compatibility with older versions of the code. As with the LJ

potential, there is also a second option available for the Buckingham potential. If IPOT is set to 2 in the `DATA.NEW` file then only the exponential part of the Buckingham potential is retained, all C values are set to zero. In addition, the A parameter can be scaled by a factor called ASCALE (see next section below) if that particular interaction is set for scaling. The application of scaling is controlled by an extra logical variable read from the `PARAM` file, e.g. in the case where IPOT=2 the `PARAM` file will now look like this:

```
     3  No._of_types_of_non-bonded_interactions
**** Type_1  Type_2     A_Buck/eV        B_Buck/A    C_Buck/eV*A**6  SCALE?
       1        1       2528.07956        0.24849      8.33234       .FALSE.
       1        2       1861.69156        0.27550     15.29926       .TRUE.
       2        2       1370.95981        0.30251     28.09145       .FALSE.
```

In the case above the self interactions (1-1 & 2-2) will not be scaled but the cross term (1-2) will, i.e. $A_{12} \rightarrow A_{12} \times$ASCALE. The IPOT=2 option allows for a smooth introduction of VdW interactions in systems which have large initial overlap energies. Note that for interactions of the Buckingham type which have C=0 the code automatically resets the cutoff for each type of interaction to $r_c=\log_e(E_{cut}/A_{ij})*B_{ij}$ where $E_{cut}=0.1k_B$, is the cutoff point for the energy, i.e. interactions with an energy less than this are considered negligible. When this happens a warning is issued.

11.2. The `SCALE` file

When IPOT=2 is selected in the `DATA.NEW` file an attempt is made to read from a file called `SCALE` the values of ASCALE and dASCALE/dt, i.e. the rate of change of ASCALE with time, per ps. If IPOT=2 and `SCALE` does not exist then a warning is issued, a template `SCALE` file is created and the program exits. The `SCALE` file should look like this (only the numbers are read-in):

```
  0.00000     ASCALE  (Scale factor for Buck. A parm.)
  0.10000     DASCDT  (Rate of change of ASCALE in /ps)
```

The above example implies the initial value of ASCALE is zero and it will increase to a maximum value of 1 at a rate of 0.1 ps^{-1}. At the end of the run the file `SCALE1` is written with the final value of ASCALE and the original value of DASCDT. Before continuing with a simulation `SCALE1` must be moved onto `SCALE`; this happens automatically if the *SWAP* script is invoked.

11.3. Specification of `DATA.NEW` file

In the case where Buckingham VdW interactions are used the parameters IPOT and IMIX in the `DATA.NEW` file mean different things. IPOT=1 implies the use of the full Buckingham potential whereas IPOT=2 implies the use of just the repulsive exponential term with the additional option of scaling down the interactions, as explained in the two preceding sections.

As with the LJ case, the value of IMIX controls the way in which missing interactions are set. If IMIX=0 then no mixing rules are applied. This means that no VdW interactions will be taken between atom types having no interactions specified. *NB*. It does not mean that there will be no Coulomb

interactions between these atom types. If IMIX=1 the following rules are used $\{A_{ij}=(A_{ii}A_{jj})^{1/2}$, $B_{ij}=(B_{ii}+B_{jj})/2$, $C_{ij}=(C_{ii}C_{jj})^{1/2}\}$ and if IMIX=2 the following $\{A_{ij}=(A_{ii}A_{jj})^{1/2}$, $B_{ij}=(B_{ii}B_{jj})^{1/2}$, $C_{ij}=(C_{ii}C_{jj})^{1/2}\}$.

11.4. Other differences between *gmq* and *gmq_buck*

Apart from the differences already pointed out, it should be noted that the output written to column 9 of the file `ST1` by *gmq_buck* is different to *gmq* and is a function of the input value of IPOT. If IPOT is not equal to 2 (*See Section 11.3*), instead of `PYY_req` being written at column 9 (*See Section 6.1.4*), the required pressure in the Z direction, `PZZ_req`, is written. If IPOT=2 then the value of the variable ASCALE (See Section 11.2) is written.

11.5. Using the Buckingham potential in the parallel code

To use the Buckingham potential in the parallel code one simply uses *ddgmq_buck* in place of *ddgmq*. All the comments above on the differences between using LJ and Buckingham potentials apply as do the ones in Section 10.3 regarding the differences between the scalar and parallel codes.

12. Amorphous polymer sample generation using *gmqpmc* (v4.2 - 19/07/2019)

To facilitate the generation of relaxed samples of amorphous polymers, versions of *gmq* are available which perform an efficient sampling of the configurations of polymer chains in the absence of all intermolecular interactions and just a user-determined amount of the non-bonded interactions along the chains. This is represents a generalised implementation of a Flory type approach to the sampling of configurations of polymer chains in the amorphous state and further details can be found elsewhere [34]. Flory has argued that there is complete screening of long range interactions in pure melts composed of linear homo-polymers [35]. The random coils adopted by the molecules are not expanded, as is often the case in a good solvent at high dilution, nor contracted, as they may be in a poor solvent. The dimensions of the chains in the melt can then be determined by considering *isolated* chains with only a certain number of specific near-neighbour interactions. This is the basis of Flory's analysis of chain structure and is a fundamental premise underpinning the rotational isomeric state (RIS) theory approximation [35]. With regard to the generation of amorphous polymers we take the approach that if the configurations of polymer chains in the melt are predictable then we have a sound basis on which to construct models of amorphous polymers. A recent review lists a number of examples of the successful use of this approach [36].

In the generalised code that is now available, called *gmqpmc* and the like, only a certain number of torsions are subjected to occasional PMC attempted moves. The program decides internally which torsions are significant regarding the global configurations of the chain. The program also identifies the primary (main chain) torsions and attempted pivots are made 50% of the time on such torsions. This is to try to avoid wasting pivots on torsions which don't directly lead to relaxation of the main chain. The rest of the modes of the chains are sampled using MD for the moment. Effectively an MD simulation is

carried out on the system in which occasional PMC moves are made. All intermolecular interactions are switched off during the MD and the range of interactions along the chain are decided by the parameter NBICUT. Nominally NBICUT defines the maximum number of bonds allowed between two atoms which can interact through the non-bonded potentials. In practice this definition is slightly more complicated. In fact to define the interacting neighbours a search of the connectivity first takes place for each atom. Atoms directly bonded to the subject atom are labelled as "1" neighbours unless these atoms are only bonded to the subject atom, i.e. pendant atom, in which case it is associated to the subject atom and given an index "0". All the atoms bonded to the "1" neighbours are then searched and those which are not the subject atom are labelled as "2" neighbours, again with the exception that pendant atoms inherit the label of their unique directly bonded neighbour. This continues out until all the "NBICUT" atoms are found. This effectively leads to full interactions between "groups" separated by NBICUT bonds.

It has to be stated that some care has to be exercised in the choice of NBICUT. The one sure way to find the optimum value for NBICUT for a particular polymer is to first generate well equilibrated results for a range of relatively short chain lengths for the corresponding bulk melts. The corresponding theoretical results can then be generated for a range of NBICUT using *gmqpmc* and the best fit selected on the basis of the agreement between the mean square end-to-end distances, mean square radii of gyration and percentages of certain conformers. However, the MD simulation of a number of bulk melts of different chain lengths is very time consuming.

12.1. Specification of the `PMC` file

By default NBICUT is set to 4 but this can be changed by creating a file called `PMC`. The `PMC` file can simply contain one line containing NBICUT. A second line can be added which contains the seed (ISEED) for the random number generator . A third line can be used to define the number of time steps between attempted pivots (ITFREQ). Finally, a fourth line can be used to exclude certain atom types from the list of pivotable torsions. If any atom in a particular torsion is on the excluded list then the torsion is not considered pivotable.

Example `PMC` file.

```
6       NBICUT
99      ISEED
100     ITFREQ
2 6     # atom types to exclude from list of torsions
```

Notes on the format of `PMC`-

(1) Everything is read-in in free format. Only the first integer is read on lines 1-3. On line 4 a # character signifies the end of the list of atom types to exclude and the start of a comment.

(2) If `PMC` does not exist then the program will use the default value NBICUT=4.

(3) If `PMC` exists but the file is empty or the first line is unreadable an error is issued and the program exits.

(4) Specifying ISEED is only necessary for performing repeat runs with the same sequence of random numbers. Normally this should not be necessary and the program will generate internally a seed. A negative value of the integer ISEED sets the random number sequence to a non-repeatable one.

(5) The number of steps between attempting pivots is set by default to 100/NUM, where NUM is the number of molecules in the system. ITFREQ must be ≥ 1.

12.2. User considerations when running *gmqpmc*

12.2.1. TAUT and IMOV

Using *gmqpmc* is very similar to *gmq*. All the input files are the same with the possibility of the above mentioned `PMC` file. When running *gmqpmc* it has to be realised that, even if your initial configuration contains many chains, each molecule is effectively isolated in space. After each successful pivot all atomic velocities in the molecule concerned are randomized. Although this should maintain about the right temperature it is advisable to execute *gmqpmc* under controlled temperature conditions in any case. TAUT can be adjusted using short test runs.

At this stage the "volume" is not really important so setting IMOV=3 is recommended. However, care has to be taken if NBICUT is large and the box is relatively small as the program still imposes periodic boundary conditions. It is better to set a box size larger than the range of any possible intramolecular non-bonded interaction before running *gmqpmc*. The desired volume can always be re-imposed at the introduction of excluded volume stage.

12.2.2. KMAX and ALPHA

One other consideration is the treatment of electrostatic interactions. These should be kept as straight Coulombic terms in *gmqpmc*. To do this set the values for ALPHA and KMAX to zero. This effectively switches off the Ewald summation leaving the charge-charge interactions as direct 1/r terms.

12.3. The FIXANG option *gmqpmc_fixang* (v4.2 - 19/07/2019)

The program *gmqpmc_fixang* is a variant of *gmqpmc* which allows one or more torsion angles to be restrained to specific values during the course of a run. The restraint is imposed using a potential of the following harmonic form

$$\Phi(\tau) = \frac{1}{2} k_\tau (\tau - \tau_0)^2 \qquad (12.3.1)$$

Where τ is the torsion angle, τ_0 is the target torsion angle required, and k_τ is the associated force constant. A value of k_τ=300 kJ mol^{-1} rad^{-2} is sufficient to maintain the torsion angle close to the desired value.

Defining which angles are to be restrained is done using the `FIXANG` data file. This file just contains, on each line, the indices of the four atoms involved in the torsion and the τ_0 value required (in degrees). An example is shown below.

Example `FIXANG` *file:-*

```
      3     4     5     6     10.000    Indices &  Angle required / deg.
      1     2     3     4      0.000    Indices &  Angle required / deg.
      2     3     4     5      0.000    Indices &  Angle required / deg.
      4     5     6     7      0.000    Indices &  Angle required / deg.
      5     6     7     8      0.000    Indices &  Angle required / deg.
      6     7     8     9      0.000    Indices &  Angle required / deg.
      7     8     9    10      0.000    Indices &  Angle required / deg.
      8     9    10    11      0.000    Indices &  Angle required / deg.
```

It should be noted that all torsions that are restrained are no longer subject to the torsion potential defined in the `PARAM` file. Neither are all torsions which share the same middle bond.

A particular utility of the *gmqpmc_fixang* code is the ability of it to obtain the total energy of rotation around a particular torsion. This can be useful when attempting to map the results of *ab initio* calculations onto a classical force field. To do this it is generally necessary to rotate a molecule around a particular torsion whilst minimizing the energy of all the other degrees of freedom. This can be achieved indirectly by setting up a sequence of minimized configurations at different values of the desired torsion angle in a `config.bin` file and then using the *gmq_en* program to extract the energies using the full the force field. In this case the molecule should be set up initially with all other torsions in the wells required. For this *gmqpmc_fixang* can be used to drive the system to a specific sequence of torsion angles. To avoid the PMC algorithm from rotating torsions the variable ITFREQ can be set to a value larger than the number of steps in the run. In general for this kind of calculation NBICUT should also be set to a value larger than the maximum number of intervening bonds between any two atoms. This will ensure all atoms interact with each other, provided that the distance cutoffs for the non-bonded potentials should also be made larger than the maximum possible distance between any two atoms. The minimization runs can then be performed at different values of τ_0. The final minimized configuration at each value of τ_0 should then be added to a `config.bin` file for later analysis. This process can be automated using the following shell script.

Example shellscript for turning an angle of a molecule through 180°:-

```
#!/bin/csh
set resdir=spin3456
# Get angle to be spun
set atom1=3
set atom2=4
set atom3=5
set atom4=6
echo $atom1 $atom2 $atom3 $atom4
# Set counters
set dangle=10
set angle=0
set endang=180
while ( $angle <= $endang )
echo $angle
rm -f FIXANG.n
echo $atom1 $atom2 $atom3 $atom4 $angle > FIXANG.n
# Script assumes that a FIXANG file already exists and that the first line is the
# angle to be rotated and all other fixed angles follow!
tail -n +2 FIXANG >> FIXANG.n
mv -f FIXANG.n FIXANG
# Execute code
gmqpmc_fixang
SWAP
gmq2conf
mv -f CONF1 $resdir
cd $resdir
# Script assumes a CONFIG.NEW file already exists in $resdir
ls
polyconb
viewbin
cd ..
@ angle = ($angle + $dangle)
end
```

In the above shellscript it should be noted that the angle to be rotated should be the first in the list in the `FIXANG` file. Unless other angles are not in stable wells there is probably no point restraining them at fixed values.

13. Cavity generation using *gmq_cav*

In the modelling of composite materials, for instance, it is some times necessary to introduce an interface into a polymer matrix. For inclusions of roughly spherical shape custom versions of *gmq* are available, e.g. *gmq_cav*, which operate in the same way as all other versions of with the exception that an input file called `CAVITY` is required. The data in `CAVITY` defines the positions required of the cavities, their dimensions and the rate at which they are to be created. Once the cavities are created in the matrix the inclusions can be inserted without too much disruption of the system. For matrices with short relaxation times it is possible to create the initial structure with the inclusion in place. However, for a polymer matrix we have first the difficulty of creating the relaxed amorphous structure. Introducing the inclusions from the start would either bias the polymer configurations, if we took into account polymer-inclusion interactions, or lead to a large disruption of the inclusion if polymer-inclusion interactions were disregarded at this stage, i.e. if we were to superimpose the two systems and then let the energy minimizer do the job of separating the matrix from the inclusion. So it is preferable to first create and relax the polymer matrix then create cavities of the appropriate size before inserting the desired inclusions.

To create cavities a soft repulsive potential is used of the following form

$$\Phi_{cav}(\vec{r}_i) = \frac{1}{2}\left(\left|\vec{r}_i - \vec{R}_0\right| - r_{cav}\right)^2 \qquad \text{for } \left|\vec{r}_i - \vec{R}_0\right| < r_{cav} \tag{13.1}$$

where k_{cav} is a force constant (currently set to 50 kg s^{-2}), $\vec{r}_i$ is the current position vector of atom i, $\vec{R}_0$ is the position vector of the origin of the centre of the cavity and r_{cav} is the current radius of the cavity. By restricting the interaction to just particles which fall within the cavity we have one side of a harmonic potential which gives a smoothly decaying soft repulsive wall. In addition, the forces that originate from this potential are limited by an upper bound which corresponds to a maximum displacement in one time step of 0.1 Å.

The values of r_{cav} and $\vec{R}_0$ are read, for each cavity to be introduced, from the file `CAVITY` as well as the value of $\dot{r}_{cav}(=\frac{dr_{cav}}{dt})$ and the maximum and minimum values r_{cav} can take. This allows the possibility to introduce cavities into a system, increase their size up to a pre-defined value, at a pre-defined rate, and then hold them at that value. Should it be necessary, the cavities can also be reduced in size by using a negative value for $\dot{r}_{cav}$ and setting the desired minimum value.

13.1. The `CAVITY` input file

An example of a `CAVITY` file, containing two cavity definitions, is given below.

Example `CAVITY` file

```
-7.E-10        X-coord of cavity (in metres)      1
-7.E-10        Y-coord of cavity (in metres)      1
-7.E-10        Z-coord of cavity (in metres)      1
 1.E-09        Current cavity radius (in metres)  1
 1.E-09        Max. radius required (in metres)   1
 0.            Min. radius required (in metres)   1
 1.E-10        Rate of change of radius (in m/ps) 1
 7.E-10        X-coord of cavity (in metres)      2
 7.E-10        Y-coord of cavity (in metres)      2
 7.E-10        Z-coord of cavity (in metres)      2
 1.E-09        Current cavity radius (in metres)  2
 1.E-09        Max. radius required (in metres)   2
 0.            Min. radius required (in metres)   2
 1.E-10        Rate of change of radius (in m/ps) 2
```

Notes on the format of `CAVITY`:-

(1) Everything is read-in in free format so spacing is not important. Only the first number of each line is read-in; the text afterwards is just a comment.

(2) The units used are metres for coordinates and radii and metres/ps for the rate of change of the radius.

(3) If the `CAVITY` file is absent and a cavity-creating version of *gmq* is executed a warning is issued, a template `CAVITY` file is written to `CAVITY1` and the program exits. `CAVITY1` can be edited and renamed `CAVITY` and execution attempted again.

(4) If the `CAVITY` file is incomplete a warning is issued and the programs uses default values.

(5) At the end of a successful run the file `CAVITY1` is produced. This contains the same or, possibly, updated value of the cavity radius.

(6) The calculation of the pressure in the system is somewhat complicated by the introduction of a cavity. It is, thus, not advised to try to run cavity introduction under any form of constant pressure conditions. It is better to either leave the volume fixed until after the cavity has been grown, the inclusion inserted and the system relaxed. Alternatively, one can increase the volume of the box beforehand, using the `SHAPE` file and IMOV=4, by an amount corresponding to the size of cavity to be inserted and then insert the cavity.

(7) As introducing the cavity does work on the system make sure the temperature control is reasonably tight, i.e. TAUT ~ 10 Δt, to avoid heating the system too much.

14. Membrane simulations using *gmq_wall* (v10.2) and *ddgmq_wall* (v7.9)

In the case of polymer membrane modelling there is a problem of how to create the flat interfaces. The polymer sample generation code, *gmqpmc*, is designed specifically for bulk periodic systems and is not easily adapted to the case of flat interfaces. For this reason custom versions of *gmq* and *ddgmq* exist, called *gmq_wall* and *ddgmq_wall*, which place smooth flat repulsive walls at positions $-x_{left}$ and $+x_{right}$ either side of the origin perpendicular to the *x* axis. All atoms in the simulation interact with the wall if their *x* positions are either greater than x_{right} or less than $-x_{left}$:

$$\begin{aligned} \Phi_{wall}(x_i) &= \frac{1}{2}k_{wall}\left(x_i + x_{left}\right)^2 \qquad \text{if } x_i < -x_{left} \\ \Phi_{wall}(x_i) &= \frac{1}{2}k_{wall}\left(x_i - x_{right}\right)^2 \qquad \text{if } x_i > x_{right} \end{aligned} \tag{14.1}$$

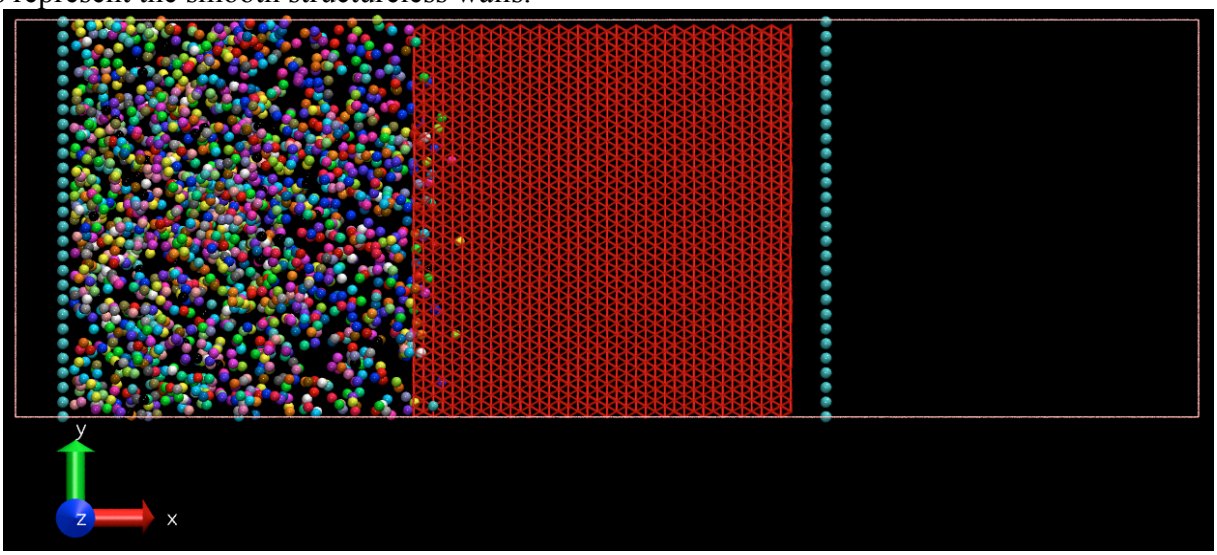
where k_{wall} is a force constant (currently set to 50 kg s^{-2}). By restricting the interaction in this way we have one side of a harmonic potential which gives a smoothly decaying soft repulsive wall. In addition, the forces that originate from this potential are limited by an upper bound which corresponds to a maximum displacement in one time step of 0.1 Å.

The position of the walls can be controlled by the user in two different modes. If MODE=0 is used, then $x_{right} = x_{left} = x_{wall}$ and the user has control over x_{wall}, the speed, $\dot{x}_{wall}$, at which x_{wall} changes, and the maximum and minimum values x_{wall} can take. This allows the possibility to introduce walls into a system, decrease their distance from the origin down to a pre-defined minimum value, at a pre-defined rate, and then hold them at that value. Thus, for instance, the walls can be used to compress a system to form a membrane model. Should it be necessary, the wall-origin distance can also be increased by using a positive value for $\dot{x}_{wall}$ and setting the desired maximum value.

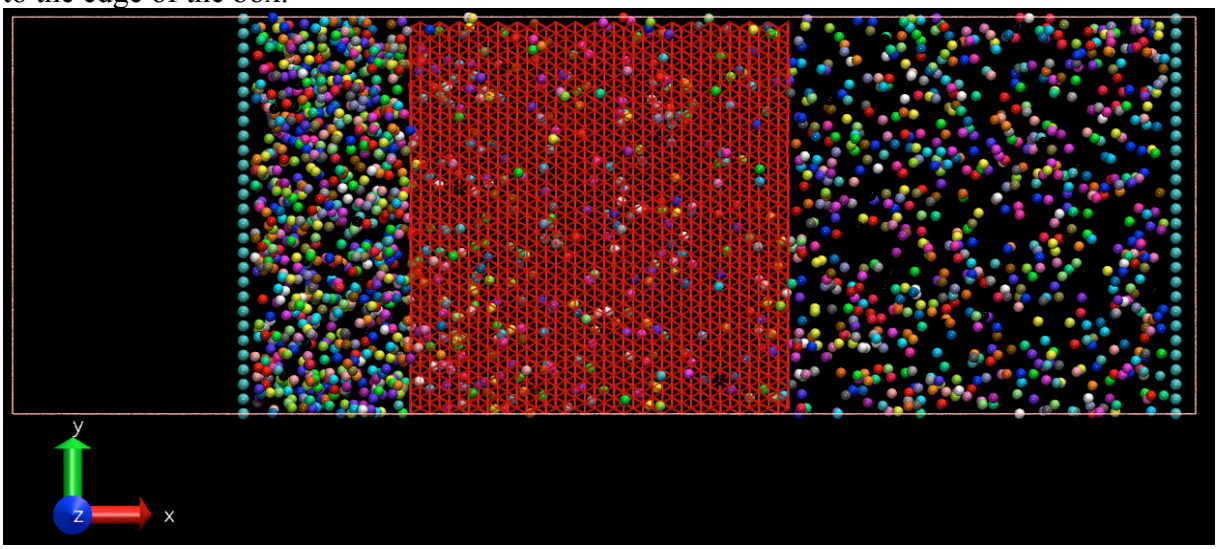
If MODE=1 is used the distances of the walls from the origin can move in order to control the pressure on the walls. The user can either specify one required pressure, in which case the left and right walls are the same distance from the origin and move in response to the difference between the average measured pressure on both walls and a user-defined required (normal) pressure. This can be useful for modelling sorption experiments where a gas, or other permeant, is initially to either side of the membrane and gradually permeates into the middle of the membrane. Holding the system at constant (gas) reservoir pressures in the *x* direction mimicks more closely the usual experimental set-up. This is also the most common boundary condition used in the solution of diffusion equations so also facilitates comparisons with theory.

Alternatively, separate pressures can be defined for the left and right walls in which case the left and right walls move ***independently***. This latter case only makes sense if the membrane is fixed in space; the pressures on the two walls will always tend to equalize if the contents of the box are free to move. To prevent the membrane from translating in the x, y, and z directions a rigid constraint is applied to its centre-of-mass (COM) to fix it at its initial position, i.e. as read-in. This contraint only removes three degrees of freedom, those of the COM, thus it does not prevent the atoms of the membrane from moving

relative to one another so does not have any direct influence on its swelling characterisitics, for example. The COM constraint allows the system to maintain separate pressures of the gas in the left and right hand reservoirs. This can be used to mimick permeation experiments where typically the gas is present initially on the feed side of the membrane at some chosen pressure and the pressure on the permeate side is maintained at some lower pressure. Of course, depending on the permeability of the membrane, the feed side reservoir will be inexorably depleted and the permeate side will fill up and thus this limits the duration of the simulation and the period at which the system can be assumed to be in a steady state. The two snapshots below illustrate this in the case where just the left hand reservoir is initially filled with gas and the left wall is maintained at a pressure of 30 bar whereas the right hand reservoir is initially empty and is then subject to a required pressure of 10 bar on the right hand wall. The upper image shows the simulation in the initial stages as gas in the feed (left) side reservoir starts to permeate into the membrane; the red mesh in the centre of the box represents the membrane and the vertical lines of cyan-coloured atoms represent the smooth structureless walls.



The second image below shows the same simulation some time later when the right hand wall is now close to the edge of the box.



In the case where pressure coupling is used (MODE=1), a loose-coupling algorithm is used to control the pressure(s). Attention thus has to be paid to the value of TAUP chosen, particularly in the case where different pressures are required on the left and right walls, as a value of TAUP that is too high will lead to a very sluggish response of the walls and differences between the measured and required wall pressures. *For example, in the tests shown above a TAUP of just 5 fs was used*. The use of the rigid constraint to maintain the COM of the membrane allows also the calculation of a net force on the membrane and thus a net pressure on the membrane. This net pressure should be the difference between the pressures on the left and right walls.

In practice *gmq_wall*/*ddgmq_wall* operate in the same way as all other versions of *gmq*/*ddgmq* except that two additional input files are required: `WALL` and `nano.types`. The data in `WALL` depends on the mode of operation: controlled wall position (MODE=0) or controlled wall pressure (MODE=1). The data in `nano.types` indicates the types of atoms that are considered to constitute the membrane. This file may be empty but must exist. An empty file implies that the rigid COM constraint is not applied, for example in the case where a membrane is being constructed by compression between the two walls.

When *gmq_wall*/*ddgmq_wall* is run successfully an extra output file called `WALL.STO1` is created. This file contains sub-averaged data concerning the walls. The four columns contain the following information:- time/ps, x_{left}/Å, x_{right}/Å, pressure on the ***left-hand*** wall in bar, pressure on the ***right-hand*** wall in bar, energy of interaction with the ***left-hand*** wall in J/mole of atoms, energy of interaction with the ***right-hand*** wall in J/mole of atoms, and the net pressure on the membrane in bar. ***N.B.*** `WALL.STO` files generated by versions of *gmq_wall/ddgmq_wall* prior to 10.0 and 7.7, respectively, *are not compatible* with subsequent ones.

14.1. Example `WALL` input files

An example of a `WALL` file in the case when pressure coupling is switched off:-

Example `WALL` file: MODE=0

```
  0             # MODE: Pressure coupling OFF
  50.0E-10      # Origin-Wall distance (in metres)
  90.0E-10      # Max. distance  required (in metres)
  50.0E-10      # Min. distance required (in metres)
 -1.0E-10       # Rate of change of distance (in m/ps)
```

An example of a `WALL` file in the case when pressure coupling is switched on:-

Example `WALL` file: MODE=1

```
  1             # Pressure coupling ON
  180.0E-10     # Origin-LEFT Wall distance (in metres)
   30.0000      # Required LEFT wall pressure (in bars)
   70.0E-10     # Origin-RIGHT Wall distance (in metres)
   10.0000      # Required RIGHT wall pressure (in bars)
```

Notes on the format of `WALL`:-

(1) Everything is read-in in free format so spacing is not important. Only the first number of each line is read-in; the text afterwards is just a comment.

(2) The units used are metres for distance and metres/ps for the rate of change of the distance.

(3) If the `WALL` file is absent and a *gmq_wall* is executed a warning is issued, template `WALL` files are written to `WALL_V` (MODE=0) and `WALL_P` (MODE=1) and the program exits. Either of these template files can be edited and renamed `WALL` and execution attempted again.

(4) If the `WALL` file is incomplete a warning is issued and the programs uses default values.

(5) At the end of a successful run the file `WALL1` is produced. This contains the same or, possibly, updated value of the origin wall distance.

(6) The standard calculation of the pressure tensor in the system is somewhat complicated by the introduction of a wall. It is, thus, not advised to try to run *gmq_wall* under any of the usual "constant pressure" conditions (IMOV=1 or 2). It is better to fix the the nominal volume (IMOV=3) of the MD box and then use wall pressure constant pressure loose-coupling mode to control the normal pressure in the *x* direction.

(7) As compressing a system does work on it make sure the temperature control is reasonably tight, i.e. TAUT ~ 10 Δt, to avoid heating the system too much.

14.2. Example `nano.types` input file

An example of a `nano.types` file:-

Example `nano.types` ***file***

```
  1    # Type Oxygen   is in membrane
  2    # Type Carbon   is in membrane
  8    # Type Carbon   is in membrane
  9    # Type Oxygen   is in membrane
 10    # Type Nitrogen is in membrane
```

Notes on the format of **`nano.types`:-**

(1) Everything is read-in in free format so spacing is not important. Only the first number of each line is read-in; the text afterwards is just a comment. The number read-in should correspond to an atom type as defined in the corresponding `PARAM` file. All atoms of this type are considered to be part of the membrane that will have its COM fixed.

15. Ancillary programs

In addition to the main dynamics programs, there are an assortment of related programs which provide different functions such as file conversions and compressions, graphics, analysis and error estimation. A number of the more useful ones are described here. Others exist and before embarking on writing software to analyse any of the output from *gmq* it would be as well to contact DB before hand.

15.1. Converting from BIOSYM input files using *bio2gmq*

There is a limited capability of converting the Biosym/Discover input files *.car and *.mdf to those required for *gmq*. Coordinates and connectivity are taken from the *.car and *.mdf files and a file of the "cvff.frc" or "cff91.frc" type is used to obtain the parameters for the potentials. The harmonic bonds, harmonic bends, torsions and out-of-planes of the intramolecular potential are converted; cross terms not being present as yet in the potential used in *gmq*. As the bends are harmonic in the cosine of the angle in *gmq*, instead of harmonic in the angle itself, a conversion is made of the force constant. This is based on equating the curvatures, $d^2\Phi(\theta)/d\theta^2$, at the minimum of the two potentials and in most cases leads to a good correspondence between them. Exact comparisons of the bending energies for the same configurations read-in to *gmq* and Biosym/Discover will give different energies though. Similarly, the out-of-plane potential used in *gmq* (see Section 2.6) is different to the ill-defined improper torsional form used in cvff or the Wilson form used in cff. Again conversions are made on matching the curvature at the minimum of the potentials. As with the bending energies, exact comparisons of the out-of-plane energies between Biosym/Discover and *gmq* are not possible.

To convert to *gmq* input files simply run *bio2gmq* in the directory which contains the appropriate `xxx.mdf` and `xxx.car` Biosym format files. The operation thereafter is reasonably self-explanatory and*bio2gmq* should produce a set of *gmq* input files as well as some other scratch files which can be erased afterwards if there are no problems or examined in the case of error.

15.2. Converting from *gmq* to Biosym format using *bin2arc*

The `config.bin` configurational history file produced by *gmq* can be converted to the Biosym "arc" format using the program *bin2arc*. For this to work it is essential a matching template "car" file exists, e.g. the one from which the files were originally converted to *gmq* format. The programs *cmpbin* and *redbin* (see next section) can be used first if the amount of data to be converted into arc format needs reducing beforehand.

15.3. Creating configurations from *Alchemy* style files using *alc2gmq*

The program *alc2gmq* can be used for creating the input file `CONFIG.NEW` for *gmq* from a set of fragment files. The format of the fragment files is the *.MOL type created by the program *Alchemy* but as this is a relatively simple text format it is not necessary that the files originate from there as such. The fragments to use and the order in which they are added together to make the larger molecule is user-defined. This is particularly useful in the case of polymers.

15.3.1. The fragment files

A sample fragment file follows:-

```
  13 ATOMS,     13 BONDS,      0 CHARGES,
   1 4       -0.2351   0.9235   0.0829     0.0000     0.0000
   2 4        0.7635   0.6597   1.0190     0.0000     0.0000
   3 3        0.7837  -0.5612   1.6914     0.0000     0.0000
   4 4       -0.1946  -1.5183   1.4276     0.0000     0.0000
   5 4       -1.1934  -1.2548   0.4917     0.0000     0.0000
   6 3       -1.2137  -0.0338  -0.1807     0.0000     0.0000
   7 1        1.7583  -0.8183   2.6046     0.0000     0.0000
   8 6       -1.9541  -1.9993   0.2868     0.0000     0.0000
   9 6       -0.1788  -2.4679   1.9505     0.0000     0.0000
  10 6        1.5243   1.4040   1.2242     0.0000     0.0000
  11 6       -0.2508   1.8729  -0.4401     0.0000     0.0000
  12 -2        1.7561  -1.7538   2.8202     0.0000     0.0000
  13 -1       -1.9903   0.1713  -0.9085     0.0000     0.0000
   1      1      2  AROMATIC
   2      2      3  AROMATIC
   3      3      4  AROMATIC
   4      4      5  AROMATIC
   5      5      6  AROMATIC
   6      6      1  AROMATIC
   7      3      7  SINGLE
   8      5      8  SINGLE
   9      4      9  SINGLE
  10      2     10  SINGLE
  11      1     11  SINGLE
  12      7     12  SINGLE
  13      6     13  SINGLE
```

The first line contains the number of atoms, bonds and charges. The number of atoms and bonds are used to read the rest of the file, the number of charges is read but ignored. There then follows a line for each atom. The first integer is just the index of the atom in the fragment. The second integer is the atom type. This should correspond to the type defined in the `PARAM` file kept in the same directory as the fragment files. Atom types "-1" and "-2" indicate the phantom "start" and "end" atoms. These atoms are bonded to the atoms which form links between the fragments. The phantom atoms are eliminated when the fragments are bonded together. Fragments will normally have just one "start" and one "end" atom unless they are initial fragments, which just have an "end" atom or, a terminal fragments, which just have a "start" atom.

The following three columns contain the atom coordinates in Ångstroms. The column after can be used to set the partial charges of atoms in the fragment. By default charges are defined by the `PARAM` file. The *alc2gmq* program gives the user the choice of setting charges in the fragment files and then writing a `CHARGE` file for use with *gmq* (See Section 5.4). The final column is ignored.

After the atoms the connectivity table is read. The first integer is just the index of the bond, the two integers that follow are the indices of the atoms involved in the bond.

15.3.2. The `STRUCTURE` file

In addition to the fragment files and the `PARAM` file it is necessary to have a file called `STRUCTURE` before running *alc2gmq*. This file specifies where the fragment files are kept, which

fragments are to be used, the order in which they are to be combined to form molecules etc. An example of a `STRUCTURE` file follows:-

```
4                                       Number of different fragments
/usr/lmpc3/brown/ALCHEMY/PEEK           Directory where fragments are found
HINIT.MOL                               Name of initiator
PEEKETH.MOL                             Name of monomer 1
PEEKKET.MOL                             Name of monomer 2
HTERM.MOL                               Name of terminator
1000.0                                  Density required in kg/cu. m
2                                       Number of Molecules to create
8                                       Number of fragments in a Molecule
1                                       1st Monomer in sequence
2                                       2nd Monomer in sequence
2                                       Etc.
3
2
2
3
4
```

The comments to the right are self-explanatory. In the example four types of fragments are used to make the molecules. The sequence in which the fragments are to be put together is given at the end. It is highly recommended that the sequence input starts with an "initiator" fragment and finishes with a "terminator", if not atoms with unsatisfied valence will result. For the moment there is no other way to specify the sequence than by inserting it into the `STRUCTURE` file. Long and/or random sequences can always be created separately, e.g. using a short Fortran program, and then inserted into the `STRUCTURE` file.

15.4. Creating configurations from *Hyperchem* PDB files using *hyp2gmq* (v1.9 03/09/2020)

In a similar way as to *alc2gmq*, the program *hyp2gmq* can be used for creating the input file `CONFIG.NEW` for *gmq* from a set of fragment files and a `STRUCTURE` file. The format of the fragment files is based on the PDB format (*.ent) created by the program *Hyperchem.* Again this is a relatively simple text format so it is not necessary that the files originate from there as such. Changes have in any case have to be made by hand to convert the element symbol (character) atom types of the PDB format to the number format atom types of the corresponding `PARAM` file. Again the fragments to use and the order in which they are added together to make the larger molecule is user-defined.

15.4.1. The fragment files

A sample fragment file follows:-

```
HETATM    1  -1     1       -14.62    3.98    4.87
HETATM    2  1      1       -14.03    3.10    5.97
HETATM    3  1      1       -12.50    3.14    5.88
HETATM    4  -2     1       -12.02    4.55    6.25
CONECT    1  2
CONECT    2  1  3
CONECT    3  2  4
CONECT    4  3
END
```

The file must contain only "HETATM" records, which must come first, and "CONECT" records and must be terminated with an "END". In the HETAM records, the first integer is just the index of the atom in the fragment. The second integer is the atom type. It is important to note that the types must correspond to the types defined in the `PARAM` file *which is kept in the same directory as the fragment files*. A `PARAM` file which is in the directory where *hyp2gmq* is executed is ignored, unless you happen to execute *hyp2gmq* in the directory where you keep your fragment files. As for *alc2gmq*, atom types "-1" and "-2" indicate the phantom "start" and "end" atoms. These atoms are bonded to the atoms which form links between the fragments. The phantom atoms are eliminated when the fragments are bonded together. Fragments will normally have just one "start" and one "end" atom. Initiator fragments will just have one or more "end" atoms and terminal fragments just have a "start" atom. As of v1.7, multifunctional initiator fragments with more than one "end" atom can have a different sequence of fragments attached to each "end" atom. Fragments without any start or end atoms are considered as entire molecules.

The third integer is ignored but then the following three columns contain the atom coordinates in Ångstroms. The column after can be used to set the partial charges of atoms in the fragment but it is not necessary to add this column if the charge is zero. By default charges are defined by the `PARAM` file. As *alc2gmq*, *hyp2gmq* gives the user the choice of setting charges in the fragment files and then writing a `CHARGE` file for use with *gmq* (See Section 5.4).

After the atoms the connectivity table is read. The first integer is the index of the subject atom, and the integers that follow are the indices of the atoms bonded to the subject atom.

The format of the `STRUCTURE` file is similar to that for *alc2gmq*. An example is given below for the case of the creation of two eight-armed POSS molecules. Each branch effectively contains two fragments, the initiator POSS core (Frag 1) and the terminator fragment (Frag 2 or Frag 3). The sequence for each molecule starts with the eight-armed initiator fragment and then there follows the eight terminator fragments (chosen at random from the two possible cases in the example below).

```
3                                          Number of different fragments
/home/dbrown/devel_hyp2gmq/try8branches
T8basic.ent                                Frag 1
mol1chain.ent                              Frag 2
mol2chain.ent                              Frag 3
5                                          Density required in kg/cu. m
2                                          Number of Molecules to create
2                                          Number of fragments per branch
1                                          Initiator with 8 arms
2                                          Branch 1
3                                          Branch 2
3                                          Branch 3
3                                          Branch 4
2                                          Branch 5
2                                          Branch 6
2                                          Branch 7
3                                          Branch 8
1                                          Initiator with 8 arms
3                                          Branch 1
2                                          Branch 2
2                                          Branch 3
3                                          Branch 4
2                                          Branch 5
3                                          Branch 6
2                                          Branch 7
3                                          Branch 8
```

The molecules that are created are randomly reorientated and the user has the choice of randomly displacing the centre-of-mass of each molecule within the box or placing the centres-of-mass on grid points. This latter option is useful for preventing the overlap of relatively small molecules. The coordinates are output to a `CONFIG.NEW` file and the charges, if read from the fragment files, can be used to create a `CHARGE` file.

15.5. Utilities for converting from TRIPOS to *gmq* `PARAM` format

The following programs operate on text files containing parameters in the TRIPOS force field [37] format in order to convert them to the units used as input to *gmq* in the `PARAM` file. In all cases a `PARAM` file needs to exist in the directory where the conversions are to be made. This `PARAM` file does not need to be complete but must respect the standard `PARAM` format upto the end of the list of atom types.

bond2gmq

This program reads from a file called "bonds" a list of bonds in the units of TRIPOS. Each line is of the following format:-

I J $d^0_{i,j}$ $k_{i,j}$

I and J are the *gmq* atom types (numbers) involved in the bond and these must correspond to the values given in the `PARAM` file. $d^0_{i,j}$ is the equilibrium bond distance and $k_{i,j}$ is the force constant for the harmonic potential. *Both of these parameters must be given in the units used in the TRIPOS force field* (Table II, pages 985-6 of [37]). The program converts the input data to *gmq* units and outputs to the screen the corresponding lines that then have to be manually pasted into the `PARAM` file.

bend2gmq

This program reads from a file called "bends" a list of harmonic angle bending parameters in the units of TRIPOS. Each line is of the following format:-

I J K *Theta* *k*

I, J and K are the three atom types (as defined in the `PARAM` file) involved in the bend with J being the central atom. *Theta* is the equilibrium bond angle and *k* the associated force constant. Again both these should be given in TRIPOS units (Table III, pages 987-990 of [37]). Output in `PARAM` format is written to the screen.

tor2gmq

This program reads from a file called "tor" a list of torsion parameters in the units of TRIPOS. Each line is of the following format:-

I J K L *k* *s*

I, J, K and L are the four atom types (as defined in the `PARAM` file) involved in the torsion with J and K being the atoms forming the central bond. *k* is the associated force constant (in TRIPOS units) and *s* is the integer giving the phase of the cosine potential. These values are given in Table V, pages 991-2, of [37]. *tor2gmq* can only handle values of *s* of -3, -2, -1, 1 and 3 for the moment. Output in `PARAM` format is written to the screen.

oop2gmq

This program reads from a file called "oop" a list of out-of-plane parameters in the units of TRIPOS. Each line is of the following format:-

I J K L *k*

I, J, K and L are the four atom types (as defined in the `PARAM` file) involved in the out-of-plane *with I being the central (sp^2) atom*. *k* is the associated force constant in TRIPOS units (Table IV, page 990, of [37]). Output in `PARAM` format is written to the screen.

vdw2gmq

This program reads from a file called "vdw" a list of LJ 12-6 potential parameters in the units of TRIPOS. Each line is of the following format:-

I *r* *k*

I is the atom type (as defined in the `PARAM` file) involved. *r* is the radius parameter and *k* is the energy parameter in TRIPOS units (Table VI, page 993, of [37]). Output in `PARAM` format is written to the screen.

15.6. Other file conversion/manipulation programs

The following programs operate on the output produced by *gmq*:-

bin2txt

Converts the `config.bin` file into the text formatted file `config.txt`. This is one sure way of preserving this data when moving between different platforms which are binary incompatible. The reverse process is accomplished by the *txt2bin* code (see below). Most machines are now binary compatible. A test can be made by transferring a `config.bin` file created on one machine to another and then running *viewbin* to see if is compatible. If it is not then the output of *viewbin* will be scrambled. If many conversions have to be made the shellscript *mbin2txt* (see below) can be used. An alternative approach is to use the *gmq_swapEndian* utility (see below).

mbin2txt

Is a shellscript that invokes *bin2txt* for all `config.bin` files found in a target directory and all sub-directories of the target (usage: *mbin2txt my_dir*). It can be run in both an interactive and unsafe mode. Interactive mode is highly recommended in cases where `config.bin` files have been stored from binary incompatible machines. A part of each `config.bin` file is printed on the screen to see if it is compatible with the machine on which the command is being run. The user then has the choice of whether to continue or not. The user also has the choice of erasing the `config.bin` and `config.count` files once the `config.txt` has been created and also compressing the `config.txt` file (recommended if you do not want to more than double your filestore!). In the unsafe mode *mbin2txt* will plough on regardless of whether the `config.bin` is compatible or not with the system, in which case the data is corrupted beyond repair should *bin2txt* not crash. It also erases automatically the `config.bin` and `config.count` files and also compresses the `config.txt` file.

txt2bin

Converts the `config.txt` file created by *bin2txt* back to the binary formatted `config.bin`. Note well that *txt2bin* will believe what is written in any existing `config.count` file found in the same directory. If you convert a `config.bin` file to `config.txt` for transferring to another machine ***you should not transfer the*** `config.count` ***file too***. A `config.count` file will be recreated automatically by *txt2bin*. However, if you do really want to append the data in a `config.txt` file to an existing `config.bin` file then you can use *txt2bin* directly as it does attempt to read an existing `config.count` file and use the data therein to append the `config.bin` file. (See also *gmq_swapEndian*).

gmq_swapEndian*

This is a utility to swap the endianess (little-endian to big-endian or vice versa) of the binary file `config.bin`. The default endianess of different machines can be different and this can lead to problems of binary incompatibility, if the `config.bin` file is transferred from one machine to another. *gmq_swapEndian* can be used to swap the endianess directly of the binary file and thus avoid the laborious passage via the `config.txt` file when transferring data to different machines. The usage is:-

```
gmq_swapEndian <input binary file> <swapped output binary file>
```


appbin

Appends two different `config.bin` files. It assumes that an existing `config.bin` file exists and is to be appended by another file called `XXX.bin`. The corresponding `config.count` file should also exist as `XXX.count`.

cmpbin

Reduces a `config.bin` file in size by writing records between certain intervals and at a certain frequency. Output is written to `confign.bin` and `confign.count`. If a large number of sub-directories have to be treated then the command *mcmpbin* can be used (see below).

mcmpbin

Is a shellscript that invokes *cmpbin* for all `config.bin` files found in a target directory and all sub-directories of the target (usage: *mcmpbin my_dir*). It can only be run in an interactive mode for the moment.

mcmpbin_txt

Is a shellscript that invokes *cmpbin* for all `config.txt` and `config.txt.Z` files found in a target directory and all sub-directories of the target (usage: *mcmpbin_txt my_dir*). It can only be run in an interactive mode for the moment.

redbin

Produces a `config.bin` format file containing only the first N_{torit} atoms. N_{torit} can either be read from the file `REDBIN.NTORIT` or is prompted for when *redbin* is invoked. The output files written are called `confign.bin` and `confign.count`. This can be useful to reduce the amount of data say in the case where you have a lot of solvent molecules and you are only interested in viewing/analysing the solute. The program *cmpbin* can be used to pre-eliminate any unwanted configurations.

cmpdat

Reduces the data files of type `STO`, `ST`, `HS` or `PT` by sub-averaging the data over a user-specified interval. The output is written onto the file `DATOUT`.

combdat (v1.7 - 21/03/2018)

This program averages together a number of data files of the same type. This can be useful, for example, if a number of repeat runs have been performed in different directories of the same system subject to different start conditions. It recognizes `STO`, `ST`, `HS`, `PT` , `HTAR` and `PEDIST.WIDOM` files but, in principle, any set of text files containing the same number of columns can be used. The files

should be the same length but, if not, then the output file, `COMBDAT.xxx`, will contain as many valid data lines as the shortest file. The user will be asked for the name of a file containing the list of files to be averaged together. The list should contain one file name per line with the file's locations either given relative to the directory where the list is contained or as an absolute path.

Note that an attempt is made to give the file a sensible title line, the one just before the actual data, but this should be checked carefully against the original data.

fcc

This program sets up an fcc crystal of atoms and writes the configuration file in `CONFIG.NEW` format. It was originally designed for use with systems interacting through a Lennard-Jones 12-6 potential. The input data file is called `xdata` and an example follows for LJ Argon with explanatory text:-

`8`	`NC`	An integer that determines the number of atoms in the cell as 4*NC**3
`119.8`	`EPSILN`	LJ ε/k_B parameter in units of K
`3.405E-10`	`SIGMA`	LJ σ parameter in metres
`39.95`	`AMASS`	Atomic mass of atom in g/mole
`1176.3037`	`DENSIT`	Required density in kg/m^3
`239.6`	`TREQ`	Required temperature in K

Note that only the first number of each line is read.

gmq_addsol (v1.9 23/11/2017)

A program designed principally to solvate systems with any kind of "solvent". *gmq_addsol* assumes the coordinate, parameter , and (optional) charge data of the “solute” are in the files `CONFIG.SOLUTE`, `PARAM.SOLUTE` and `CHARGE.SOLUTE`. In the files `CONFIG.SOLVENT`, `PARAM.SOLVENT` and (optional) `CHARGE.SOLVENT` it expects the corresponding data for the solvent. The idea is that the box of solvent is superimposed on the solute system and all those solvent molecules within some critical user-specified distance of a solute atom are eliminated. For this approach to function the two configuration files must have the same dimensions, otherwise the program stops. The program then checks to see if the solute configuration already contains solvent molecules. This is necessary for setting the correct atom types of the solvent molecules to be added. It then proceeds by eliminating all solvent molecules which are too close to solvent atoms according to some critical distance specified by the user. The combined set of coordinate data is written to a file called `CONFIG.ADDSOL` and a combined charge file will be written to `CHARGE.ADDSOL` in the case where the corresponding solute and/or solvent charge files were present. A combined parameter file is also created called `PARAM.ADDSOL`. This should be checked very carefully!
N.B. The use of this program is deprecated since the development of the load option of *nano_tpi*.

gmq_chgtyp (v1.2 26/03/2018)

This program can be used to change the types of one or more user-specified types of atoms from a `CONFIG.NEW` file. The user also has the possibility to restrict the changes to certain molecules. The new

configuration is written to CONFIG.NEW_CHGTYP and the new charge file to CHARGE_CHGTYP. Note that the code only changes the atom types. If the charges are defined in the PARAM file by the type of the atom then the charges will change. *However, if the charges are defined in the CHARGE file then they will not change*.

gmq_delat (v1.4 23/11/2017)

This program can be used to remove one or more user-specified types of atoms from a CONFIG.NEW file. The new configuration is written to CONFIG.NEW_DELAT and the new charge file to CHARGE_DELAT.

gmq_delmol (v1.5 16/01/2019)

This program can be used to remove one or more molecules of a user-specified type from a CONFIG.NEW file. All molecules to be deleted must have the same number of atoms. The program detects molecules having the same number of atoms but having a different sequence of atom types. In this case the user must choose the molecule type to be used. The user can choose from a sequential or random mode of deletion or can supply a list of molecules to delete. The list has to be called LIST.DELMOL. A template file will be generated if this file is not found. The format of the file is given below:

```
# List_No.  Mol._ID  Delete(T/F?)  Scaled_in-box_COM_Coords
      1        17           F        0.334804   -0.954103    0.403941
      2        18           F        0.174308   -0.394178   -0.954092
```

The user has the option of selecting the molecules to be deleted, by changing "F" in the third column to "T" and of changing the scaled in-box centre-of-mass position of the molecule. The user can also choose to change the box size and shape using a SHAPE file.
NB. Molecules that are not in the primary box are moved back into it by default.

The new configuration is written to CONFIG.NEW_DELMOL and note that the timer is zeroed. The new charge file is written to CHARGE_DELMOL.

gmq_extract (v1.4 16/06/2020)

Creates a file called CONFIG.EXTRACT in CONFIG.NEW format from user-defined record of a config.bin file. *NB.* the velocities of atoms are taken from those in the CONFIG.NEW present in the directory so re-equilibration may well be required. The user has the choice of writing to the extracted file one of the following:- the required temperature in the CONFIG.NEW file, the temperature associated with the record chosen from the config.bin file or inputting a required temperature.

gmq_index

Creates the INDEX file (See Section 6.1.7) from coordinates stored in the config.bin file. This can be useful for simulations carried out using *ddgmq* where an INDEX file is not created.

gmq_phantom (v1.7 23/11/2017)

This code can insert phantom atoms into the middle of cyclic groups. This is particularly useful when chains sampled using the *gmqpmc* code are to have their intermolecular interactions switched on, i.e. introduction of excluded volume stage of the construction of a sample of an amorphous polymer. The `PARAM` file needs to contain the parameters for the phantom atoms to be introduced. Different phantom atoms can be defined for 5-atom cycles, 6-atom cycles etc. The new configuration file is called `CONFIG.NEW_PHANTOM` and the new `CHARGE` file, should an old one exist, is called `CHARGE_PHANTOM`. The program *gmq_delat* can be used afterwards to remove the phantom atoms. **N.B.** the numbering of atoms is changed using *gmq_phantom*.

gmq_pseudo

This program allows a group of atoms to be collapsed into one virtual site, or pseudo-atom, in order that the structure, i.e. the g(r), can be presented in a simplified form. Originally this was developed for use with chains containing cyclic groups. By representing each ring as the mass unweighted average position of the atoms constituting the ring, the rdfs for these virtual sites could be compared and further information extracted.

To define the groups of atoms a file called `TEMPLATE` must first be created. This file applies to the first molecule in the `CONFIG.NEW` file and all other molecules having the same number of atoms as the first molecule. The `TEMPLATE` file must contain, on the first line, the number of atoms in the first pseudo-atom (NATCYC) and the type of this pseudo-atom (ITCYC). There then follows, for each of the NATCYC atoms in the first pseudo-atom, the index of the atom in the corresponding `CONFIG.NEW` file with one entry per line. All subsequent pseudo-atoms are defined in the same way in `TEMPLATE` with one line for NATCYC and ITCYC and then one line for each atom index in the pseudo-atom.

The program the uses the template to produce the corresponding configuration file, `CONFIG.PSEUDO`, and stored configuration files, `config.bin_pseudo` and `config.count_pseudo`. These can then be used with *gmq_rdf*, for example, in order to calculate the rdfs for the pseudo-atoms. *NB.* a corresponding `PARAM` file has to be created for the pseudo-atoms beforehand.

gmq_putback (v1.7 - 23/11/2017)

Reads in the current `CONFIG.NEW` and translates the centres of mass of all molecules back into the primary MD box. The output is written to `CONFIG.PUTBACK`.

gmq_offset (v1.1 - 23/11/2017)

Reads in the current `CONFIG.NEW` and translates all atoms by a user-defined offset vector. The output is written to `CONFIG.OFFSET`.

gmq_reflect (v1.1 - 23/11/2017)

Reads in the current `CONFIG.NEW` and scales all atoms by a user-defined scale vector. The output is written to `CONFIG.REFLECT`. For example, applying the scale vector (1,1,-1) will reflect all atoms about the z=0 plane.

gmq_redring (v1.5 - 18/09/2013)

This program can be used on systems containing cyclic groups to automatically search for rings and replace them by a single site. As of v1.4, rings are identified not only by the number of atoms in the ring but also the sequence of atom types of atoms in the ring. This can be useful for reducing the number of atoms to display graphically, for example. It has a similar function to that of *gmq_pseudo*. Output data files are `CONFIG.REDRING` and `config.bin_redring` and `config.count_redring`.

gmq_scaleup (v1.6 - 23/11/2017)

Enables the size of a system to be scaled-up by an integer multiple of the current size of the `CONFIG.NEW` file. The output configuration is written to `CONFIG.SCALEUP`. If a `CHARGE` file exists then a scaled-up version is written to `CHARGE.SCALEUP`. The user has a choice of keeping the velocities of all copies of an original atom the same or assigning new random velocities. The former option should always be tried first to ensure the scaled-up system gives the same results as the original one. Slight differences in temperature are to be expected. The temperature calculation depends on the number of degrees of freedom in the system which does not scale exactly with system size as three degrees of freedom are removed due to the conservation of linear momentum. *NB* as of version 1.1 systems containing atoms linked to periodic images of other atoms, "infinite" chains, are now scaled up correctly.

gmq_sortatom (v1.4 - 23/11/2017)

If, despite all the recommendations, a `CONFIG.NEW` file has been constructed with a sequence of atoms which is not in the *gmq*-required molecule-by-molecule format then *gmq_sortatom* can be used as a last resort to reorder the atoms so that it is. The reordered configuration is written to `CONFIG.NEW_SORT`. and the reordered `CHARGE` file, should it exist, to `CHARGE_SORT`. *NB.* This program should only be used if *gmq_sortmol* fails.

gmq_sortmol (v1.3 - 23/11/2017)

Checks to see if each “molecule” in a `CONFIG.NEW` file is a single connected entity. If not then the system is broken up into its constituent connected entities and the configuration produced is written to `CONFIG.NEW_SORT`. This can only be done, without changing atom indices, provided that the atoms already appear in a molecule-by-molecule order in the existing `CONFIG.NEW` file. If this is not the case the program fails. At this point it is better to reconstruct the configuration anew, if possible, respecting the order required by *gmq*. Alternatively *gmq_sortatom* can be used but this involves a reordering of the atoms.

gmq_tset (v1.5 - 23/11/2017)

Uses the `CONFIG.NEW` and `PARAM` data files to produce a new set of random velocities at the target temperature (TREQ) specified in `CONFIG.NEW`. The new configuration is written to the file called `CONFIG.NEW_VEL`.

gmq2ansys

Converts a `CONFIG.NEW` file or a configuration from `config.bin` into *ANSYS* format. This is the first stage in the creation of an actual physical model of a system in plastic using a rapid prototyping machine. When first invoked the program creates a file called `ansys.types` which should then be edited before re-running *gmq2ansys*. The `ansys.types` file just contains a scaling factor from Angstroms to mm and then the radii of each atom type. The user can choose to include only certain types and clip the image to reduce the number of atoms in the output file (see *gmq2c3d*). The resulting input file for *ANSYS* is called `CONFIG.ANSYS`. (Contact DB if you want further instructions of how to actually convert the *ANSYS* file into a physical model).

gmq2c3d

Creates a file called `CONFIG.C3D` from either `CONFIG.NEW` or `config.bin` which can be read by the *Chem3D* molecular modelling program. When first invoked the program creates the file `c3d.types` which should be edited before re-running *gmq2c3d* to give the correct element names and *Chem3D* element types for each atom type in the `PARAM` file. The user can choose to include only certain types of atoms in the output file. The user can also choose to output folded (in-box) or unfolded coordinates. If the folded representation is selected, the user can clip the image in the x, y, and z planes in order to focus on a certain region and to reduce the number of atoms in the output file. The corners of the box can also be output to the file as pseudo-Vanadium atoms. If `config.bin` is selected for conversion the user can specify the range of configurations to be converted. The resulting file will thus contain a number of images which can be animated using *Chem3D*.

gmq2cav

This is a rather specific program which can be used to create a `CAVITY` file containing multiple entries. This is useful when a system of nanoparticles is to be introduced into a polymer matrix. So long as each nanoparticle is a complete molecule, as far as its `CONFIG.NEW` file is concerned, then the program calculates the centre of mass of each nanoparticle and creates an entry for it in a `CAVITY` format file. The required input files are thus a `CONFIG.NEW` containing one or more nanoparticles, but nothing else, a `PARAM` file, and a `CAVITY` file correctly formatted with just one cavity defined. The output file is called `CAVITY1`.

gmq2com (v1.2)

This program uses an existing set of `CONFIG.NEW`, `PARAM`, `config.bin` and `config.count` files for a molecular system in order to create a reduced set which just contains the positions of the centres-of-mass of each molecule. This can be useful for visualization purposes or for

obtaining structural or dynamic information for the COMs. The output files are called `CONFIG.NEW_COM`, `config.bin_com` and `config.count_com.`

gmq2conf

This program uses the `CONFIG.NEW` and `PARAM` input files to produce the `CONF1` file directly without having to run *gmq*. This can be especially useful if there is a problem running *gmq* and a connectivity problem is suspected; `CONF1` can be converted to `config.bin` using *polyconb* and then the configuration can be analysed or displayed using other codes.

gmq2dbws (v1.1 - 18/09/2013)

This program creates a template input file for use with the local version of *DBWS* (see http://www.physics.gatech.edu/downloads/young/DBWS.html for instructions on how to use *DBWS*). *DBWS* can be used to create pseudo-X-ray diffractograms from atomic coordinates. The template input file is called `DBWS.IN_new`. By default, it sets the range of 2*Theta values to be used to be quite narrow (50.0 to 50.1 in 0.02 steps) which is sufficient for a short test to estimate the total running time. *DBWS* does not produce any on-screen output. The output file `DBWS.PROFILE` contains the pseudo-diffractogram to be plotted.

gmq2discus (v1.2 - 18/09/2013)

Creates a file called `DISCUS.STRU` in the input format for the *DISCUS* program [38]. Amongst other things *DISCUS* can be used to obtain pseudo-X-ray diffractograms.

When first invoked *gmq2discus* creates the file `discus.types` which should be edited before re-running *gmq2discus* to make sure the correct element names for each atom type in the `PARAM` file are given.The user is then given a choice as to whether to include all atoms or include or exclude certain types. The user can then choose coordinates from either `CONFIG.NEW` or `config.bin` . If `config.bin` is chosen and a range of configurations, then the positions written to the `DISCUS.STRU` file are the average ones in this interval.

To use *DISCUS* a command file has to be made which reads the data file `DISCUS.STRU`. A typical example for calculating a powder diffractogram is given below. *DISCUS* is then invoked by simply giving the command:- "`discus command.file`" where "command.file" is the name of the file containing the *DISCUS* commands. See the discus user manual for detailed explanations of the various parameters.

Example DISCUS command file:-

```
################################################################
# Powder diffraction
################################################################
read
stru DISCUS.STRU
# Enter sublevel powder
powder
#
set dh,0.5
set dk,0.5
```

```
set dl,0.5
#
# The resulting intensities must be binned to a grid in 2-THETA.
# The grid size and the minimum and mximum 2-THETA values are
# specified next:
#
set dtth,1.0
set tthmin, 1.0
set tthmax,90.0
#
# Here are the remaining settings. First we set the mode to
# Fourier (which is required, unless you work with stacking
# faults - see online help):
#
set four,four
#
# Next we set temperature factors to be ignored and select
# xray scattering at a wavelength of 1.54056 Angstroms (CUA1)
#
set temp,ignore
xray
set wvle,CUA1
#
######################################################################
# Add extra parameters recommended by Reinhard Neder 13/12/2007
#
# Anomalous dispersion is switched off (or set to disp,anom)
# (This is the default, thus it does not affect your calculation)
#
set disp,off
#
# Powder diffraction is calculated by complete integration
# instead of the Debye formula
# (This is the default, thus it does not affect your calculation)
#
set calc,complete
#
# Lorentz and polarisation effects are calculated.
# Bragg-Brentano powder diffraction geometry is set. Optionally you can
# specify the monochromator angle <theta_mono>.
#
set lpcor, bragg, 26.6
#
# Switch off the the instrumental resolution function
#
set delt,0.0
#
# The calculation is performed on an axis with equal steps in 2Theta
#
set axis,tth
# (This is the default, thus it does not affect your calculation)
#
######################################################################
######################################################################
#
# We are ready to go, but first we have a look at the current
# setting using th command
#
show
#
# Now we start the calculation ..
#
run
exit
#
# Finally we need to save the result to a file. This is again
# done using the output segement of DISCUS. Note that the
# format needs to be set to powder.
#
output
```

```
format powder
outf test.powder
run
exit
#
exit
```

gmq2nano (v3.3 - 23/11/2017)

This is a very specific code that has been developed to create hybrid models of either silica nanoparticles or silica surfaces from configurations of amorphous bulk silica. Amorphous bulk models can be generated using the simple ionic models of SiO_2, for example those of Tsuneyuki [4] or BKS [39]. The hybrid model, originally developed to simulate silica nanoparticles [40, 41], contained up to six types of atoms:- core silicon and oxygen atoms (similar to the bulk models), surface silicons, non-bridging silanol oxygens, bridging oxygens, and silanol hydrogens. The surface silicons, silanol oxygens and hydrogens, and non-bridging oxygens were linked together by a network of bonds whereas the core is just held together by the original bulk model parameters for the VdW and Coulombic terms. As well as silanol groups, the current version also allows the user to place Si-$(CH_3)_2$ groups (between two non-bridging oxygens), Si-$(CH_3)_2$OH groups and CH_2-CH_3 groups. In this case various other types of atoms have to be defined:- surface dimethyl silicons attached to two CH_3 groups, bridging oxygens attached to dimethyl silicons, methyl carbons, methyl hydrogens, etc. (See example `PARAM` file below).

To use *gmq2nano* an initial configuration of amorphous silica is required; this can be generated, for example, by quenching molten systems of SiO_2. *N.B.* the initial configuration should have all coordinates within the primary cell; this can be assured by using *gmq_putback* beforehand although the *gmq2nano* will translate all atoms into the primary MD box. The `PARAM` file for the pure silica system then needs to be changed to include the extra atom types, needed by *gmq2nano*, and all the other relevant terms in the potential. N.B. that *gmq2nano* expects the original Si atoms to be Type 1 and the original O atoms to be Type 2. A typical `PARAM` file for the hybrid model is given below:-

```
* SiO2 bulk/surface model with Si-(CH3)2, Si-(CH3)2-OH & CH2-CH3 groups ****
          17 No._of_different_atom_types
**** Label     Mass/(g/mole)     Charge/e
     Si        28.086D0            2.4D0            # 1
     O        15.9994D0           -1.2D0            # 2
     Si_surf  28.086D0            1.3               # 3
     O_nb     15.9994D0           -0.533            # 4
     O_b      15.9994D0           -0.629            # 5
     HO_nb    1.007970            0.206             # 6
     Si_DM    28.086D0            1.3D0             # 7
     O_DMb    15.9994D0           -0.629D0          # 8
     C_DM     12.0111500          0.0D0             # 9
     H_DM      1.0079700          0.0D0             # 10
     O_DMnb    15.9994D0          -0.533            # 11
     HO_DMnb  1.007970            0.206             # 12
     O_ethox  15.9994D0           -0.533            # 13
    C2_ethox  12.0111500          0.0D0             # 14
    C3_ethox  12.0111500          0.0D0             # 15
    H2_ethox   1.0079700          0.0D0             # 16
    H3_ethox   1.0079700          0.0D0             # 17
          15  No._of_different_types_of_bonds
****  Type_1  Type_2     Bond_Length/A
        2       3          1.53000000   1300.0      # 1  O-Si_surf
```

```
         3       4          1.53000000   760.0       # 2  Si_surf-O_nb
         3       5          1.53000000   1000.0      # 3  Si_surf-O_b
         4       6          0.9500000                # 4  O_nb-HO_nb
         7       9          1.90000000               # 5  Si_DM-C_DM
         9      10          1.09000000               # 6  C_DM-H_DM
         3       8          1.53000000   1000.0      # 7  Si_surf-O_DMb
         7       8          1.53000000   1000.0      # 8  Si_DM-O_DMb
         7      11          1.53000000   760.0       # 9  Si_DM-O_DMnb
        11      12          0.9500000                # 10 O_DMnb-HO_DMnb
         3      13          1.53000000   760.0       # 11 Si_surf-O_ethox
        13      14          1.43                     # 12 O_ethox-C2_ethox
        14      15          1.54                     # 13 C2_ethox-C3_ethox
        14      16          1.09                     # 14 C2_ethox-H2_ethox
        15      17          1.09                     # 15 C3_ethox-H3_ethox
          12  No._of_different_types_of_bond_angles
**** Type_1  Type_2  Type_3  Theta_0     K/(KJ/mole)
    3       4       6     116.0       500.0D0   #  1 Si_s-O_nb-HO_nb
    7       9      10     109.471     0.0D0     #  2 Si_DM-C_DM-H_DM
   10       9      10     109.471     0.0D0     #  3 H_DM-C_DM-H_DM
    9       7       9     109.471   700.0D0     #  4 C_DM-Si_DM-C_DM
    7      11      12     116.0       500.0D0   #  5 Si_DM-O_DMnb-HO_DMnb
    3      13      14     116.0       500.0D0   #  6 Si_surf-O_ethox-C2_ethox
   13      14      15     110.000   727.2871727 #  7 O_ethox-C2_ethox-C3_ethox
   13      14      16     109.471     0.0       #  8 O_ethox-C2_ethox-H2_ethox
   15      14      16     109.471     0.0       #  9 C3_ethox-C2_ethox-H2_ethox
   14      15      17     109.471     0.0       # 10 C2_ethox-C3_ethox-H3_ethox
   16      14      16     109.471     0.0       # 11 H2_ethox-C2_ethox-H2_ethox
   17      15      17     109.471     0.0       # 12 H3_ethox-C3_ethox-H3_ethox
          5  No._of_different_types_of_torsions
# Atom_Types_1--2--3--4 & on next line Coeffs. in J/mole
   9  7  9  10    #   1  C_DM    Si_DM    C_DM  H_DM
    7100.000   21300.00       0.000  -28400.000   0.000   0.000   0.000
   3  13 14 15    #   2 Si_surf-O_ethox-C2_ethox-C3_ethox (COCC from PEO!)
    5181.027   5608.209    6267.126  -15416.994  -677.628 -4563.147 3566.186
   3  13 14 16    #   3 Si_surf-O_ethox-C2_ethox-H2_ethox (COCH from PEO!)
    0.0   0.0    0.0  0.0  0.0 0.0 0.0
   13 14 15 17    #   4  O_ethox-C2_ethox-C3_ethox-H3_ethox (OCCH from PEO!)
    7100.000   21300.00       0.000  -28400.000   0.000   0.000   0.000
   16 14 15 17    #   5  H2_ethox-C2_ethox-C3_ethox-H3_ethox (HCCH from PEO!)
    0.0   0.0    0.0  0.0  0.0 0.0 0.0
          50  No._of_types_of_non-bonded_interactions
 ****  Type_1  Type_2      Sigma/A        Eps/k/K
       1       1           3.0             0.0        # 1 Si-Si
       1       2           1.53         13275.88      # 2 Si-O
       1       3           3.0             0.0        # 3 Si-Si_s
       1       4           1.53         13275.88      # 4 Si-O_nb
       1       5           1.53         13275.88      # 5 Si-O_b
       2       2           2.995086994    237.8161    # 6 O-O
       2       3           1.53         13275.88      # 7 O-Si_s
       2       4           2.995086994    237.8161    # 8 O-O_nb
       2       5           2.995086994    237.8161    # 9 O-O_b
       3       3           3.0             0.0        # 10 Si_s-Si_s
       3       4           1.53         13275.88      # 11 Si_s-O_nb
       3       5           1.53         13275.88      # 12 Si_s-O_b
       4       4           2.995086994    237.8161    # 13 O_nb-O_nb
       4       5           2.995086994    237.8161    # 14 O_nb-O_b
       5       5           2.995086994    237.8161    # 15 O_b-O_b
       7       1           3.0             0.0        # 16 Si_DM-Si
       7       2           1.53         13275.88      # 17 Si_DM-O
       7       3           3.0             0.0        # 18 Si_DM-Si_s
       7       4           1.53         13275.88      # 19 Si_DM-O_nb
       7       5           1.53         13275.88      # 20 Si_DM-O_b
       7       7           3.0             0.0        # 21 Si_DM-Si_DM
       7       8           1.53         13275.88      # 22 Si_DM-O_DMb
       7      11           1.53         13275.88      # 23 Si_DM-O_DMnb
       8       1           1.53         13275.88      # 24 O_DMb-Si
       8       2           2.995086994    237.8161    # 25 O_DMb-O
```

```
 8       3           1.53          13275.88     # 26 O_DMb-Si_s
 8       4           2.995086994     237.8161   # 27 O_DMb-O_nb
 8       5           2.995086994     237.8161   # 28 O_DMb-O_b
 8       8           2.995086994     237.8161   # 29 O_DMb-O_DMb
 9       9           3.3150          52.82586   # 30 C_DM-C_DM
10      10           2.4553          22.13655   # 31 H_DM-H_DM
11       1           1.53          13275.88     # 32 O_DMnb-Si
11       2           2.995086994     237.8161   # 33 O_DMnb-O
11       3           1.53          13275.88     # 34 O_DMnb-Si_s
11       4           2.995086994     237.8161   # 35 O_DMnb-O_nb
11       5           2.995086994     237.8161   # 36 O_DMnb-O_b
11       8           2.995086994     237.8161   # 37 O_DMnb-O_DMb
11      11           2.995086994     237.8161   # 38 O_DMnb-O_DMnb
13       1           1.53          13275.88     # 39 O_ethox-Si
13       2           2.995086994     237.8161   # 40 O_ethox-O
13       3           1.53          13275.88     # 41 O_ethox-Si_s
13       4           2.995086994     237.8161   # 42 O_ethox-O_nb
13       5           2.995086994     237.8161   # 43 O_ethox-O_b
13       8           2.995086994     237.8161   # 44 O_ethox-O_DMb
13      11           2.995086994     237.8161   # 45 O_ethox-O_DMnb
13      13           2.995086994     237.8161   # 46 O_ethox-O_ethox
14      14           3.3150          52.82586   # 47 C2_ethox-C2_ethox
15      15           3.3150          52.82586   # 48 C3_ethox-C3_ethox
16      16           2.4553          22.13655   # 49 H2_ethox-H2_ethox
17      17           2.4553          22.13655   # 50 H3_ethox-H3_ethox
0 No._of_different_types_of_out_of_planes
```

When invoked *gmq2nano* gives the user a choice of constructing a nanoparticle or slab. For the slab a laboratory axis (*x*, *y*, or *z*) has to be chosen that will be perpendicular to the slab. Checks are made to be sure that, in the case of a slab, the shape of the MD box is compatible with the cut that is being made, i.e. the box axis is parallel to the chosen lab axis and the face of the MD box perpendicular to the axis is in the plane of the cut. In both cases the user can control separately which Si and O atoms are retained by defining the distance, R_n, either radial or perpendicular from the COM, beyond which atoms are discarded. After this first elimination of atoms the user has the option to write the configuration directly before any transformation to a hybrid model. This allows the user to relax the system first before going further. Allowing separate values of R_n for Si and O atoms permits the user to adjust the numbers of each type to maintain electroneutrality in the case of a relaxation step. The relaxation step can change the structure at the surfaces in particular if annealing is carried out to higher temperatures. If the complete transformation is required then the user is asked to set the distance, b_n, defining an Si-O bond (b_n=1.9 Å by default). There is then the choice of keeping an ionic core to the system or using the molecular model throughout. In the former case a thickness, r_n, has to be defined for the molecular surface layer. This parameter is set by default to 9 Å but the resulting surface needs to be examined to check that the ionic core is not partially exposed.

With the above information an iterative algorithm is then used to set up the connectivity and atom types. In the case of a slab the algorithm has the following steps:-

STEP 1:	Read the previously prepared bulk configuration of amorphous SiO_2.
STEP 2:	Discard all Si and O atoms that lie further from the plane of the origin than the nominal half-width, R_n, of the slab. NB. R_n can be set independently for Si and O atoms. If an inner core of SiO2 is too be kept then define the thickness, r_n, of the surface molecular layer. By default this is set to 9 Å.
STEP 3:	For each silicon identify, store and count all the near-neighbour oxygens, i.e. those that lie within the critical distance b_n.
STEP 4:	Eliminate all silicons in the surface layer, i.e. those further from the plane of the origin than R_n-r_n, which have less than a critical number of oxygen neighbours,N_{cox}. By default N_{cox} is set to four oxygen neighbours but the user can define this.
STEP 5:	Eliminate all oxygens in the surface layer having no silicons within the critical distance b_n.
STEP 6:	Identify all the "non-bridging" oxygens as those having just one silicon within a distance of b_n. Set the connectivity table for each atom to indicate a bond between them.
STEP 7:	Label all silicons with a "non-bridging" oxygen as a neighbour as being "surface silicons".
STEP 8:	Visit all surface silicons and label all ***non*** "non-bridging" oxygens within the critical distance b_n as being "bridging" oxygens. Update the respective connectivity tables to form bonds between the "surface silicons" and the "bridging oxygens".
STEP 9:	Loop over all "bridging oxygens" and find all ***non*** "surface silicons" within b_n that are further from the origin plane than R_n-r_n. Label these silicons as "surface silicons" and update the connectivity tables.
STEP 10:	If either the number of "non-bridging oxygens" added at STEP 8 or the number of "surface silicons" added at STEP 9 is non-zero go to STEP 8.
STEP 11:	Visit all the "bridging oxygens" and check if they are bridged between two "surface silicons". If not change the type of the atom back to a regular core oxygen BUT keep the connectivity table unchanged to indicate a bond between this oxygen and the surface silicon.

After applying the above algorithm there are a number of surface silicons and some of these (silanol Si) are bonded to non-bridging oxygens (nbOs). The proportions that are bonded to one, two or three non-bridging oxygens are given as well as an estimation of the surface coverage (in non-bridging oxygens per square nm). The user can choose at this stage to add Si-$(CH_3)_2$ groups between pairs of nbOs. The user has to define the maximum number of Si-$(CH_3)_2$ groups to add. ***N.B.*** The actual number added will depend on the number of nbOs and their arrangement on the surface with respect to each other. The user then has to define a critical distance (4Å by default) defining the maximum distance apart two nbOs can be to be considered close enough for the insertion of a Si-$(CH_3)_2$ group between them. A second parameter is also required to define the closest approach of other non-bonded "heavy" (non-Hydrogen) atoms to the dimethyl silicon and carbons, i.e. all non-Hydrogen atoms except the two nbOs to which it will be potentially attached. By default this parameter is set to 2 Å. This parameter is intended to prevent Si-$(CH_3)_2$ groups being added in regions already occupied by other atoms. This can happen as some nbOs can be created below the surface. The code then progresses by choosing an nbO at random and then searching for all nbOs within the critical distance. A silicon is then added at a point between the two nbOs in the plane defined by vector normal to the surface and using an O-Si-O bond angle equal to the tetrahedral angle. The two carbon atoms are then added respecting the bond lengths given in the `PARAM` file and tetrahedral geometry. A check is then made to see if the silicon or either of the carbons overlap

strongly with any of the other atoms in the system, the two nbOs to which it is linked excepted, as described above. If there are strong overlaps then the silicon and carbons are removed and the next possible pair of nbOs is checked. In the case of a succesful addition the hydrogens are added to the carbons and the two nbOs have their types changed into bridging oxygens attached to dimethyl silicons. The search for suitable pairs continues by choosing at random a new subject nbO and searching for its near neighbours. The search stops either when the required number of Si-$(CH_3)_2$ groups has been added or when all possble pairs have been searched. At the end of the search the total number of Si-$(CH_3)_2$ groups added is printed out and a warning is issued if this is less than that required. The estimated coverage in terms of the number of Si-$(CH_3)_2$ groups per square nm is also printed out.

In a similar manner, the following 3 steps allow the user to add Si-$(CH_3)_2$-OH and Si-$(CH_3)_2$-O-CH_2-CH_3 groups to *single* nbOs. The user is first invited to specify the number of "Si-$(CH_3)_2$-O" groups to add to single nbOs. This is thus the desired total number of Si-$(CH_3)_2$-OH and Si-$(CH_3)_2$-O-CH_2-CH_3 groups. To add the Si-$(CH_3)_2$-O groups, an nbO is chosen at random and the Si and O atoms are added using equilibrium bond lengths read from the `PARAM` file, tetrahedral bond angles and random torsions. The two carbon atoms are then added in the plane perpendicular to the O-Si-O plane using equilibrium bond lengths read from the `PARAM` file and tetrahedral bond angles. At this point a check for overlaps of the added Si, O and two carbon atoms is made with all other non-Hydrogen atoms except the subject nbO. If there is an overlap then the four atoms are removed and another attempt is made to add them to the same nbO with different random torsion angles for the Si and O atoms. If 100 unsuccessful attempts have been made to add the four heavy atoms to a specific nbO then no further attempts will be made and a new nbO is chosen from those remaining on the list. If the addition is succesful, i.e. no overlaps, then the methyl hydrogens are added using equilibrium bond lengths read from the `PARAM` file and tetrahedral bond angles. The bridging oxygen has its type reset to that of a bridging oxygen attached to a dimethyl silicon and the non-bridging oxygen to that of a non-bridging oxygen attached to a dimethyl silicon. Further Si-$(CH_3)_2$-O groups are in added in the same way. At the end of this step the total number of Si-$(CH_3)_2$-O groups added is printed out and a warning is issued if this is less than that required.

Once all the required, or as many is possible given the overlap criterion, Si-$(CH_3)_2$-O groups have been added the user can specify how many of these should be terminated with CH_2-CH_3 groups. Once this required number has been entered, a Si-$(CH_3)_2$-O group is chosen at random amongst the ones which have just been added and the two carbon atoms of the CH_2-CH_3 are first added using equilibrium bond lengths and angles read from the `PARAM` file and random torsional angles. A test for overlaps between all existing "heavy" atoms, except the subject non-bridging dimethyl Oxygen (nbdmO), is made with the two carbons. If an overlap is found then the two carbons are removed and a new attempt is made two add them to the same nbdmO. After 100 unsuccessful attempts to add the CH_2-CH_3 group to a specific nbdmO no further attempts are made and a new Si-$(CH_3)_2$-O group is selected from those remaining on the list. If the addition is succesful, i.e. no overlaps, then the hydrogens are added using equilibrium bond lengths read from the `PARAM` file and tetrahedral bond angles. The atom type of the nbdmO is changed

into an ethoxy group Oxygen. At the end of this step the total number of Si-$(CH_3)_2$-O-CH_2-CH_3 groups added is printed out and a warning is issued if this is less than that required. The estimated coverage in terms of the number of Si-$(CH_3)_2$-O-CH_2-CH_3 groups per square nm is also printed out.

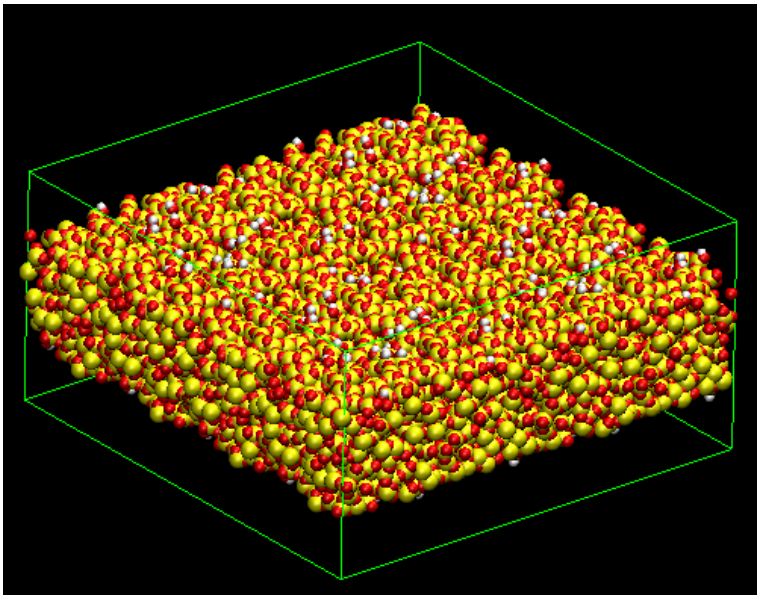
In the next step, all the remaining Si-$(CH_3)_2$-O groups, i.e. those that have not had CH_2-CH_3 groups added, are automatically hydroxylated. The hydroxyl hydrogen is added using an equilibrium bond length and bond angle read from the `PARAM` file. At the end of this automatic hydroxylation the total number of Si-$(CH_3)_2$-OH groups added is printed out and a warning is issued if this is less than that required. The estimated coverage in terms of the number of Si-$(CH_3)_2$-OH groups per square nm is also printed out.

In the following step the user has the possibility to add CH_2-CH_3 groups to some of the remaining nbOs. Again the two carbon atoms are first added using equilibrium bond lengths and angles read from the `PARAM` file and random torsional angles. A test for overlaps between all existing "heavy" atoms (except the subject nbO) is made with the two carbons. If an overlap is found then the two carbons are removed and a new attempt is made two add them to the same nbO. After 100 unsuccessful attempts to add the CH_2-CH_3 group to a specific nbO no further attempts are made and a new nbO is selected from those remaining on the list. If the addition is succesful, i.e. no overlaps, then the hydrogens are added using equilibrium bond lengths read from the `PARAM` file and tetrahedral bond angles.

In the next step the number of remaining nbOs is then printed out as well as the estimated coverage and the user then has the choice of reducing the number of non-bridging oxygens, for example if some target surface coverage of Si-O-H groups is desired. Finally the user can choose whether or not to add the silanol hydrogens to the remaining nbOs. These are added using the equilibrium O-H bond length and Si-O-H bond angle read from the `PARAM` file.

As atoms can be eliminated and added by *gmq2nano*, the code recalculates the total charge on the system at the end and prints out the adjusted charge for the surface silicons which would ensure electroneutrality. It is important to then change the corresponding entry in the `PARAM` file before trying to run *gmq* on the resulting configuration. Depending on the geometry chosen, the output configuration file is written to `CONFIG.NEW_NANOP` or `CONFIG.NEW_SLAB`. ***N.B.*** In both cases the configuration has been sorted so that the atoms appear in the molecule-by-molecule order required by all *gmq* codes.

The figure below shows an example of a model silica surface created by *gmq2nano* from a bulk silica system. The colour code is Si=Yellow, O=Red, H=White.



gmq2pdb (v3.1 - 10/03/2020)

This program can be used to convert either `CONFIG.NEW` or `config.bin` into PDB format. The output file is called `CONFIG.pdb`. PDB format is one of a number of formats that can be read by the visualization program *VMD* [42]. If the PDB file is to be used by *VMD* then a second file is required to define the connectivity; *VMD* ignores the connectivity in the PDB file and by default uses an internally generated connectivity table based on atom types and separations. This second file is called `CONFIG.psf`. When first invoked the program creates a file called `pdb.types` containing a list of the different atom types and the corresponding element symbol which is determined from the mass. This file should be checked and edited before re-running *gmq2pdb* to give the required element names for each atom type in the `PARAM` file. Particular attention should be paid if massless atoms are present in the system, e.g. models of N_2 and O_2 with charges on their centres-of-mass so as to give a quadrupole moment [43]. The user can choose to include only certain types of atoms in the output file. The user can also choose to output folded (in-box) or unfolded coordinates. If the folded representation is selected, the user can clip the image in the x, y, and z planes in order to focus on a certain region and to reduce the number of atoms in the output file. The corners of the MD box are output to separate files (called `BOX.pdb` and `BOX.psf`) as pseudo-Vanadium atoms. If `config.bin` is selected for conversion the user can specify the range of configurations to be converted. The resulting file will thus contain a number of images which can be animated using *VMD*.

The user can choose to centre the view on one particular atom or molecule. The user can also supply the name of a file containing a list of atoms or molecules to be displayed. The list must contain on the first line the number of atoms or molecules in the list and then on subsequent lines the indices of the atoms or molecules.

An additional option allows the trajectory of atoms of a user-selected type to be displayed as a (static) pseudo-chain. If the trajectory option is selected then the output files are called

`TRAJECTORY.pdb` and `TRAJECTORY.psf`. The user also has the choice of either displaying the trajectory in three different ways:-
(1) as an unfolded trajectory using the coordinates directly from the `config.bin` file
(2) as (1) except that the first point is translated back into the primary MD box
(3) as a folded trajectory with all points within the primary MD box
The third option is particularly useful to see what parts of the box are accessed, for example, by penetrant molecules in a polymer matrix.

If a `WALL.STO` file is present in the directory, i.e. as generated by simulations using gmq_wall/ddgmq_wall, the the user has the option of creating a file called `WALL.pdb` so that the positions of the left and right walls can be visulaized.

gmq2shape

Is a utility program that uses a `CONFIG.NEW` and `PARAM` file to calculate the size of a new **cubic** box which corresponds to a new user-specified density. The box size is written out to the `SHAPE` file and can be used with *gmq/ddgmq* when IMOV=4.

ikww (v1.2 - 14/01/2008)

Is a utility program that calculates the relaxation time of the stretched exponential (KWW) form for best-fit parameters supplied by the user. For example, if a correlation function has been fitted to the form:

$$C(t) = \exp\left(-\left(\frac{t}{\alpha} \right)^{\beta} \right)$$

then the corresponding relaxation time, τ, is defined as:

$$\tau = \int_0^\infty \exp\left(-\left(\frac{t}{\alpha} \right)^{\beta} \right) dt = \frac{\alpha}{\beta} \int_0^\infty s^{\frac{1}{\beta} - 1} \exp(-s)\, ds = \frac{\alpha}{\beta} \Gamma\left(\frac{1}{\beta} \right)$$

Effectively *ikww* uses standard numerical methods [44] to evaluate values of the Gamma function.

The user has to supply values for α and β as well as their error bars. The input can be done interactively or via the file called `ALPHA.BETA`, useful if multiple evaluations have to be made. The format expected for each line of the `ALPHA.BETA` file is:-
ALPHA Error_ALPHA BETA Error_BETA

polyconb

Converts the configurations stored in the `CONF1` sequential binary file during a run to a direct access binary file and appends it to the `config.bin` configurational history file (see Section 7.3 for a fuller description).

pressit (v1.7 - 10/07/2017)

This code can be used to determine the pressure for the next iteration of a single-component gas in "polymer" matrix system when using the pressure iteration technique [45, 46] to determine the sorption isotherm. When calculating the uptake of a gas in a polymer system it can be shown [45] that the equality of the chemical potentials of the gas in the gas phase and the gas in the polymer phase amounts to the following equation when the gas molecules are modelled as rigid bodies:

$$\frac{\rho_{pol}}{\rho_{gas}} = \frac{S_{pol}}{S_{gas}}$$

where ρ is the density of gas and S the gas solubility and the superfixes "gas" and "pol" indicate the gas phase and polymer phase, respectively. It requires that simulations have been performed already of the pure gas phase at a range of densities and that the density vs. pressure and solubility vs. pressure have been fitted to the following "virial" forms:-

$$\rho(p) = \frac{M*10^5}{RT}\left(p + ap^2 + bp^3\right)$$

$$S(p) = 1 + \alpha p + \beta p^2 + \gamma p^3$$

where M is the molecular mass of the gas (in kg/mole) and the factor of 10^5 takes into account that the pressure, p, is given in bar (1 bar=10^5 Pa). Both fits ensure that the correct behaviour, i.e. that of an ideal gas, is obtained at low pressures. The coefficient $M*10^5/RT$ and the best fit parameters are kept in files prefixed `gas.data_*` in the directory indicated by the GAS_HOME environment variable. Such files are specific for each gas model used and for a specific temperature. They have the following form:-

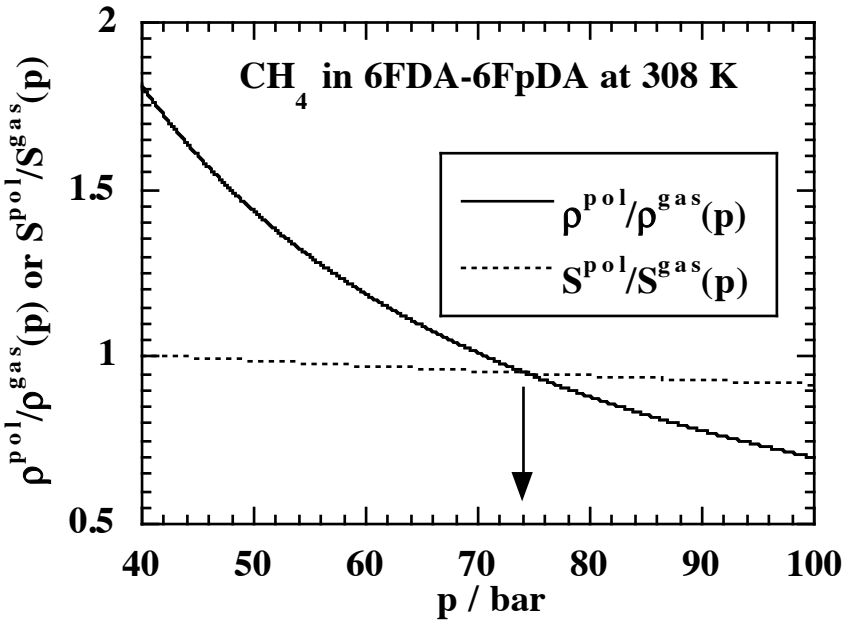
```
# Methane Yin_Rigid model at 308K
308.0D0                                  # Temperature in K
16.04246D0                               # Gas Mol. Mass in g/mole
100.0D0                                  # Max. Pressure in bar
0.0021195D0 -2.6775D-06   0.0            # Coeffs. of S(p)=1+AS*p+BS*p**2+CS*p**3
0.626449534D0 0.0013022 -4.9032E-06      # Coeffs. of rho(p)=AD*(p+BD*p**2+CD*p**3)
```

To choose the required `gas.data_*` file from those available in the directory indicated by the GAS_HOME environment variable the name of the file is given in the `gas.fname` file in the dir where *pressit* is executed. Below is an example of the contents of a `gas.fname` file:

```
gas.data_CO2_308K
```

The coeff. AD is not a fit parameter but $M*10^5/RT$. Its value is checked in the code and an error is issued if there is a significant inconsistency. The maximum pressure is that used in the pure gas systems, i.e. the maximum pressure for which the fitted curves can be assumed to be valid. The code then asks the user to input information concerning the gas+polymer system: the number of gas molecules, the average volume of the gas+polymer system (in m^3), the average solubility of the gas in the polymer and the error on this

quantity. The shell script *run_pressit* automates this process (see below). The code then finds the pressure for which the ratio of $\frac{\rho_{pol}}{\rho_{gas}(p)}$ is closest to that for $\frac{S_{pol}}{S_{gas}(p)}$ This pressure should then be used at the next iteration of the gas+polymer system. Output from the run is written to the file `pressit.out` including the values of the density ratios and solubility ratios as a function of *p*. An example curve is given below for methane in the 6FDA-6FpDA polyimide. The crossing point occurs at ~74 bar.



run_pressit

This is a shell script that automates the running of the *pressit* code (see above). It requires that the following files, or symbolic links, exist in the execution directory:- `gas.fname`, `CONFIG.NEW`, `STO.WIDOM` and `RESULTS`. The latter file should contain the results of running *datav* on the `STO.WIDOM` file, i.e. after the completion of a TPI analysis. The script attempts to obtain the number of penetrant gas molecules from the `CONFIG.NEW` file. It assumes the last molecule is a gas molecule and all the other gas molecules have the same number of atoms as this one. This is only true of single-component gas containing systems. The script then obtains the average volume, average solubility and error in the solubility from the `RESULTS` file and then inputs all of these values to the *pressit* code. If the code runs successfully the user has te choice of plotting the output in the `pressit.out` file using the *gnu_pressitx* script (see below).

gnu_pressitx

Script to plot the results for the density ratios and solubility ratios in the `pressit.out` file (see description of *pressit* above). The data are plotted using the default scale and then replotted using different scales on the y axis so as to better see the intersection.

pressest (v1.1 - 10/07/2017)

This code provides a very crude estimate of the number of molecules that should be inserted into a pure polymer matrix. It is used as a starting point for the iterative technique for obtaining gas uptake in a polymer matrix, or any sort of matrix in general. The method used is related to that described in the related *pressit* code (please read description of *pressit* first). In the case of gas molecules modelled as rigid bodies, the equality of chemical potentials gives rise to the following basic relation for the equilibrium between a gas in a gas phase and a gas in a "polymer" phase:

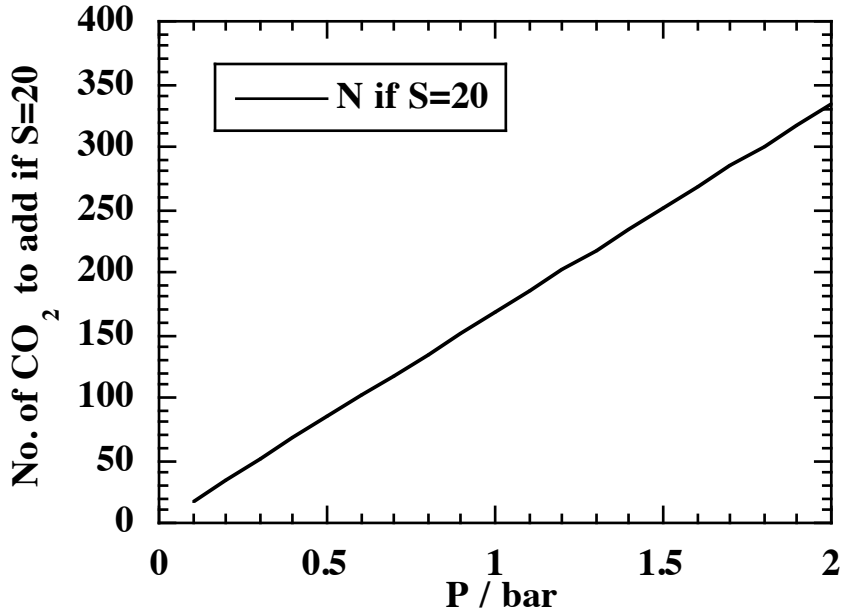
$$\frac{\rho_{pol}}{\rho_{gas}} = \frac{S_{pol}}{S_{gas}}$$

Now if we assume that at low pressures the solubility in the polymer is constant and equal to the infinite dilution value, S^0_{pol}, this is a poor approximation in general for glassy polymers, then you can obtain an (over)estimate of the number of gas molecules to insert for a certain pressure since

$$\frac{\rho_{pol}(N)}{\rho_{gas}(p)} \approx \frac{S^0_{pol}}{S_{gas}(p)} \Rightarrow \frac{Nm}{V^0_{pol}} \approx \frac{S^0_{pol}\rho_{gas}(p)}{S_{gas}(p)} \Rightarrow N \approx \frac{S^0_{pol}\rho_{gas}(p)V^0_{pol}}{S_{gas}(p)m}$$

where we approximate the gas density in the polymer phase as $\rho_{pol}(N) = \frac{Nm}{V^0_{pol}}$ with V^0_{pol} the relaxed volume of the pure matrix. As with *pressit*, it is assumed that fits of the gas phase densities and solubilities have been pre-calculated at the required temperature and are stored in a `gas.data_*` file in the directory indicated by the GAS_HOME environment variable. When executed *pressest* first reads the `gas.fname` file to obtain the name of the relevant file containing the gas phase data and then requests V^0_{pol}, S^0_{pol} and the error in S^0_{pol}. Again this can be automated using the *run_pressest* script (see below).

The code the estimates the number of gas molecules in the matrix in the range of pressures from 0 to 2 bar. The output from the run is written to the file `pressest.out`. An example curve is given below for CO_2 in a polymer.



This gives an overestimate of N=85 molecules for a pressure of 0.5 bar.

run_pressest

This is a shell script that automates the running of the *pressest* code (see above). It requires that the following files, or symbolic links, exist in the execution directory:- `gas.fname`, `STO.WIDOM` and `RESULTS`. The latter file should contain the results of running *datav* on the `STO.WIDOM` file, i.e. after the completion of a TPI analysis on the pure matrix. The script then obtains the average volume, average solubility and error in the solubility from the `RESULTS` file and then inputs all of these values to the *pressest* code. If the code runs successfully the user has the choice of plotting the output in the `pressest.out` file using the *gnu_pressestx* script (see below).

gnu_pressitx

Script to plot the results in a `pressest.out` file (see description of *pressest* above) for the estimates of the number of gas molecules sorped by a matrix at low pressures. The high and low estimates are also shown.

setup

Initialises the files `STO, ST, HS, PT, INDEX` and `config.count` (see Section 6). Its use is highly recommended in a directory before running any *gmq* code.

tidy

Script to remove interactively all output files produced by a *gmq* run. It is highlyrecommended to tidy-up a directory where the output from short trial runs exist before embarking on a series of longer repeat runs. Scripts like *SWAP* use the existence of certain output files as evidence of a successful completion.

tidyw

Script to remove interactively all output files produced by test runs of *nano_widom/nano_tpi*.

viewbin

Simply prints the header for each record in the binary direct access file `config.bin`. This is a simple way to test that the `config.bin` file is readable on a particular machine. Binary files are not necessarily compatible between different operating systems.

fug

Prints out the fugacity of different real gases (H_2, N_2, O_2, CO_2, NH_3, CH_4, He) at a user-defined temperature and over a range of user-defined pressures, $p > 0$, based on the Beattie-Bridgeman equation of state, see http://www.osti.gov/scitech/servlets/purl/4289497. The output is written to the file `FUG.OUT`.

15.7. Data analysis programs

datav (v2.3 - 16/06/2020)

This program analyses the data in a user-specified history file (`STO`, `ST` etc.) and produces a file called `RESULTS` which contains at the end a table with the following headings:

```
AVERAGE     STAND.ERROR        RMSD          STAT.INEFF.
```

with entries for each column of the input data. The range over which the data is analysed is also user-specified. The standard error is calculated using a crude blocking method [1] from the root mean square deviations (RMSD) in the (possibly sub-averaged) data and the estimated statistical inefficiency (STAT.INEFF), reported in units of the sampling interval in the original history file. The statistical inefficiency is an estimate of the correlation in the data, the larger the statistical inefficiency the longer the data is correlated therefore the fewer the number of independent data points. For data which is oscillating about a well-defined mean and spans several correlation times this method produces reasonable values of the standard errors but short simulations on still relaxing samples will, of course, lead to unreliable estimates.

For files of type `STO.WIDOM` the calculations of <V*exp(-ΔU/kT)>/<V> and <V*exp(-ΔU/kT)*U>/<V*exp(-ΔU/kT)> are made automatically and printed out along with their standard error (see description of *nano_widom/nano_tpi*).

datploav (v1.4 - 16/06/2020)

This is a program to sub-average the data in `STO`, `ST` and other files of a similar format. For files of type `STO`, `ST` and `HTAR` the resultant columns in the output file (`DATOUT`) are generally labelled with a meaningful header.

dipcor

Calculates the time auto-correlation function for the total system dipole moment vector from the data kept in the file `DMOM`. See section 5.6 for the details of the definition of the dipole moment vector and its run-time calculation and storage.

get_enth (v1.2 - 07/09/2009)

This program uses the `STO` and `CONFIG.NEW` files to calculate the enthalpy (H=U+PV) of the system. Using *get_enth* is preferred to trying to calculate the enthalpy by hand due to the numerous possibilities for confusion between the units used. The resulting titled data file is called `ENTHALPY`. The first integer is the number of lines in the file. The first column contains the time, as in `STO`, and the 8 following columns are labelled:-

```
T/K, H/J/box, H/J/mole_atoms, H/J/kg, E/J/mole_atoms, PE/J/mole_atoms, PV/J/mole_atoms,
Hex/J/mole_atoms
```

where H is the enthalpy, E is the kinetic energy, PE is the potential energy, PV is the PV part of the enthalpy, and Hex is the excess part of the enthalpy, i.e. the sum of the potential energy and PV.

gmq_clust (v1.8 - 19/03/2021)

Given two different user-specified atom types and a user-specified distance the program calculates the distribution of cluster sizes involving the two species from the configurations on the `config.bin` file. Output is written to `CLUST.OUT` in the form of a probability density of cluster size. To obtain the amount of material in clusters of a certain size then the probability density of cluster size has to be multiplied by the cluster size and then renormalized.

Since version 1.4 an extra analysis has been added of the square radius of gyration, $<S^2>$, of the clusters. The idea being that the variation of $<S^2>$ with cluster size, n, gives an indication of the form of the clusters. For example, globular clusters should have an $n^{2/3}$ dependence whereas more open chain or network-like clusters give a linear dependence of $<S^2>$ on n.

The program also writes, for the last configuration used in the analysis, a coordinate file for the clusters in a set of coordinates which should show the clusters as a single entity. This `CONFIG.CLUST` file has to be transferred to a temporary directory, renamed `CONFIG.NEW`, and then you can run *gmq2pdb* (so long as you have a copy or link to the original `PARAM` file) to produce a pdb file that can be read and displayed by *VMD*. There is a slight problem with *gmq2pdb* in that it will not write bonds that are longer than ~2 times the equilibrium values given in `PARAM` to the psf file but in any case the HBOND option of *VMD* will still work.

gmq_corete (v1.7 - 18/09/2013)

Calculates the time auto-correlation functions for the square end-to-end distance, $\left\langle \vec{\boldsymbol{R}}^2(0)\vec{\boldsymbol{R}}^2(t)\right\rangle$, and the end-to-end vector, $\left\langle \vec{\boldsymbol{R}}(0)\bullet\vec{\boldsymbol{R}}(t)\right\rangle$, of molecules. This calculation only really makes sense for linear molecules for which ends are well defined. The average is performed over all molecules for which two ends can be distinguished and over all time origins selected from the `config.bin` file. These

configurations should be equally spaced in time. The program can do an automatic search for chain ends and the user can restrict the analysis to molecules having more than a user-specified number of atoms (e.g. to exclude solvent molecules) and can exclude atoms of less than a certain user-specified atomic weight (e.g. hydrogens). The correlation functions and their normalized forms are written to the file `CORETE.OUT`.

gmq_dis (v2.1 - 22/02/2021)

Calculates the probability density of either distances between pairs of atoms, angles between three atoms or torsion angles between four atoms from configurations stored in `config.bin`. The user can choose from the list of bond, bend and torsion types in the `PARAM` file or use a customised list of atom global indices. The custom list should just contain the global indices of the atoms involved in the pair, bend angle or torsion with one line for each. The user can opt to take the default bin width in each case or enter a different one. The probability density output is written to `DIS.OUT`. The average property as a function of time is written to the file `AVG.OUT` along with the corresponding root mean square deviation from the mean.

To facilitate the construction of custom lists, the user is given the option to print a list of the bonds, bends or torsions used in the distribution to the file `DIS.LIST`. This can then be edited by hand to select certain specific lines of interest.

With *gmq_dis* the user is also given the option of outputting the list of values, used in the creation of the distributions, as a function of time into a file called `TIME.LIST`. Be careful with this option. If your distribution includes many pair distances, angles or torsions for each record you will generate a huge amount of output. This option is really designed for looking at just a few values at a time. For torsions there is the possibility of screening out redundant torsions, i.e. torsions that share the same middle bond.

From v1.3 on *gmq_dis* gives the user the choice of not using nearest image vectors to determine the separations of atoms. This can be useful in the case where custom lists are used to obtain, for example, distances between atoms in the primary chain which exceed half the box length of the MD cell.

gmq_disp (v2.0 - 20/03/2021)

Is principally used to calculate the mean square displacements of the atoms in the system as a function of time from the `config.bin` file. There are currently four modes of operation though. The user has the following options:-

(1) The average mean-squared displacement vs. time behaviour of atoms of each type, along with an average over all species, can be calculated. The output is written to the file `DISP.OUT`. *Note that columns are only output for atom types that are actually present in the system*. The user can specify whether to average over all time origins or just one time origin. For averaging over all time origins the data should be equally spaced in time. To see if there is a correlation in the displacements in x, y and z the cross-correlation function, $\Delta x(t)*\Delta y(t)+\Delta x(t)*\Delta z(t)+\Delta y(t)*\Delta z(t)$, is also calculated and written to the file `DISPCOR.OUT`.

(2) As (1) except that the *individual* mean-squared displacement vs. time curves can be output for each atom of a certain type. In this case the output is written to `DISPI.OUT`. The contents of `DISPI.OUT` can be viewed on an SGi console using the program *vdisp*.
(3) The actual x, y and z coordinates of a user-specified selection of atoms are written to the file `TRAJ.OUT` as a function of time. This allows individual particle trajectories to be followed in space.
(4) The probability density distribution of the absolute value of components of the displacement vector, $\rho(|\alpha|,t)$ for α=x, y, z, is calculated for a user-specified atom type at a user-specified time interval, t. The data is written to the file `DISPDIST.OUT` (column "Cmpt_Prob") along with that expected of a Gaussian distribution (column "Gauss_prob"):

$$\rho_{Gauss}\left(|\alpha|,t\right)=\sqrt{\frac{3}{2\pi\left\langle\Delta r^2(t)\right\rangle}}\exp\left(-\frac{3\alpha^2}{2\left\langle\Delta r^2(t)\right\rangle}\right)$$

Also calculated and written to the output file (column "R_prob") is the probability density of the magnitude of the displacement vector, $\rho(|\Delta\vec{\mathbf{r}}|,t)$; this is related to the self part of the Van Hove function. If a particle's ***vector*** displacement in a time t is $\Delta\vec{\mathbf{r}}_i(t)=\vec{\mathbf{r}}_i(t)-\vec{\mathbf{r}}_i(0)$ then the Van Hove self-correlation function can be defined as

$$G_s(\Delta\vec{\mathbf{r}},t)=\left\langle\delta\left(\Delta\vec{\mathbf{r}}-\Delta\vec{\mathbf{r}}_i(t)\right)\right\rangle$$

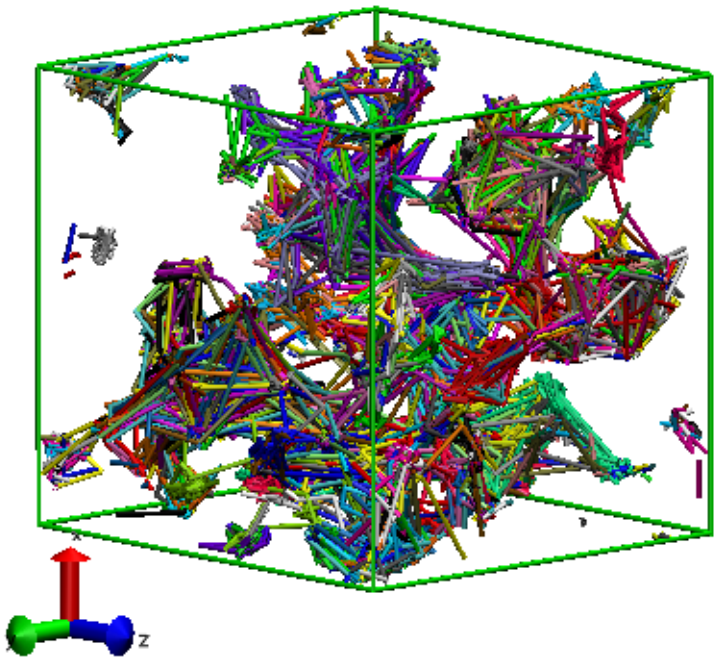
i.e. it is the probability density that a particle has a displacement vector $\Delta\vec{\mathbf{r}}$ in a time t. The function $\rho(|\Delta\vec{\mathbf{r}}|,t)$ is thus related to the Van Hove self-correlation function by

$$\rho(|\Delta\vec{\mathbf{r}}|,t)=4\pi|\Delta\vec{\mathbf{r}}|^2\,G_s(\Delta\vec{\mathbf{r}},t)$$

gmq_dispx (v2.2 - 18/05/2018)

This program provides a crude method to extend the time range over which the mean square displacements of penetrant molecules in polymer systems can be evaluated in fully periodic systems. The method is a variant of the Kinetic Monte Carlo (KMC) approach for which their applications in the field of polymer-penetrant systems has been reviewed by Theodorou [47]. This particular method requires as input a simulation of the polymer-penetrant system over a time interval where the penetrant molecules explore a significant part of the periodic system. A detailed description of the method and a number of examples of its use can be found elsewhere [48].

The condition that penetrant molecules explore a significant part of the periodic system can be checked in advance by using the trajectory option of *gmq2pdb*. As an example, the following *VMD* picture shows the trajectories taken by carbon dioxide molecules through a glassy polyimide matrix at 400 K. Although the trajectories are restricted to certain zones of the MD box by the presence of the polymer, there is still a sufficient amount of space accessible to the penetrant molecules to allow them to diffuse, i.e. they can percolate the system.



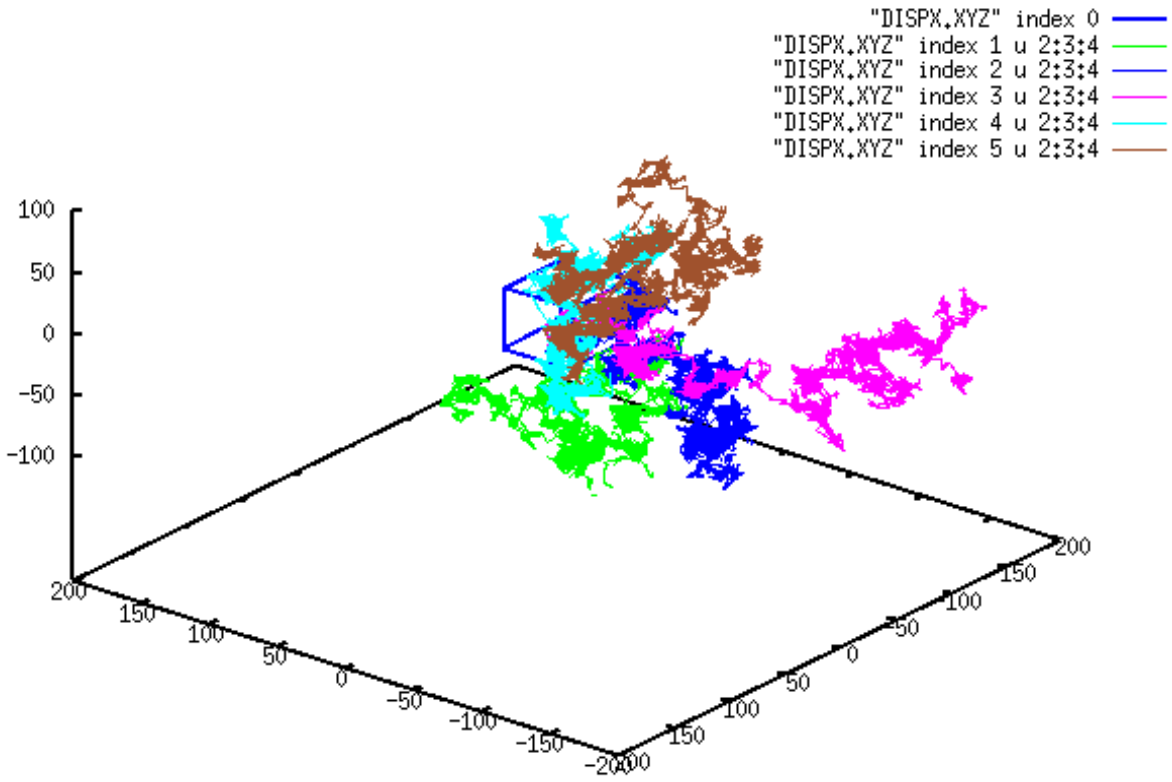
In an initial phase, the program analyses the actual trajectories of the penetrant molecules and for each stored configuration assigns them to sub-cells of the primary box based on the folded (in-box) positions. The resolution of the grid of sub-cells is set by the user-defined parameter DGRID. The cumulative number of cells visited as a function of time is written to the file `CUMUL_SUBCELL_VISIT`. A cluster analysis of the visited sub-cells is made and the results are printed to the screen and output to the file `WARN_CLUST`. A warning is issued if the visited sub-cells are not all in one cluster. This is an indication that not all random walks can percolate the system.

For each sub-cell that has been visited by a penetrant molecule, a record is kept of the identity of all sub-cells that are moved to at subsequent steps and the number of times that they are moved to. Of course, it can be that between subsequent stored configurations the penetrant remains in the same sub-cell and this is counted too. In this way a sort of probability matrix is generated for jumps to occur between two sub-cells.

In a second phase, a user-defined number of random walks are initiated from randomly selected sub-cells amongst those that have been visited. At each step of a walk, a sub-cell to jump to is selected with a probability corresponding to that found in the first phase from the list of sub-cells that can be reached from the current sub-cell. In order to avoid the random walks becoming trapped, the probability matrix is artificially symmetrized. This is equivalent to enforcing detailed balance, i.e. the flux of

particles from cell I to J is the same, on average, as those from cell J to cell I. In terms of probabilities of being in a particular cell and the rate constants for the jumps between cells this can be expressed as $p_I k_{IJ} = p_J k_{JI}$.

Jumps between cells are made using the nearest image convention so that random walks will be continuous in space. An example of just five such random walks are shown in the figure below. Each walk starts from the primary MD cell (partially obscured) and is coloured differently. The scale on the axes are in Angstroms and the trajectories were generated over 200000 ps with an interval between stored configurations of 20 ps. 20 ps is thus the time step in the KMC calculation. The user has the option of storing all the walks in a file called `DISPX.XYZ` but is warned that this can be a very large file.

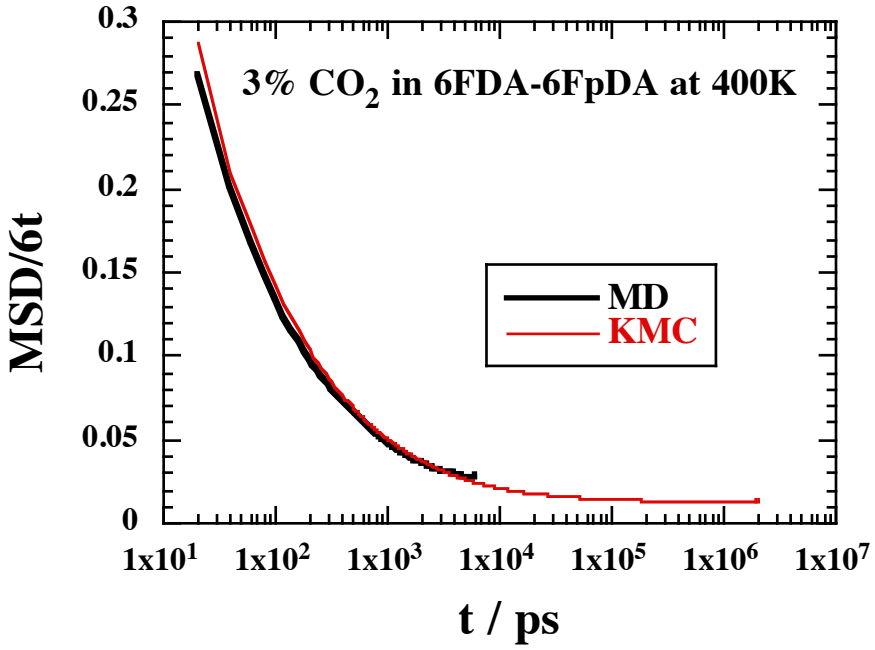


The `DISPX.XYZ` file contains two blank lines between each random walk data set so that *gnuplot* can access the different walks using the "index" keyword. The first data set (index 0) contains the lines to draw the outline of the mean MD box. The *gnuplot* commands for drawing the picture above are shown below:-

```
set xrange [-200:200]
set yrange [-200:200]
set zrange [-100:100]
splot "DISPX.XYZ" index 0 w lines lw 2 lt 3,"DISPX.XYZ" index 1 u 2:3:4 w lines,
 "DISPX.XYZ" index 2 u 2:3:4 w lines, "DISPX.XYZ" index 3 u 2:3:4 w lines, "DISP
X.XYZ" index 4 u 2:3:4 w lines, "DISPX.XYZ" index 5 u 2:3:4 w lines
pause -1
```

In general, the time interval between stored configurations should not be so large that the penetrant can have diffused more than half a box length between successive configurations nor should it be too short either as then the KMC calculation will become rather inefficient. The results are sensitive to the choice of the resolution of the grid (DGRID). This should match to some extent the natural

fluctuations a penetrant molecule will have between undergoing jumps to different sites. Too fine a resolution for the grid will lead to an underestimation of the connectivity between the different trajectories taken by the penetrants and thus lead to slower diffusion. Too coarse a grid will in turn overestimate the connectivity and lead to faster diffusion. An analysis of the connectivity of the visited cells is printed to the screen after the original MD trajectory has been read. A warning is issued if the visited cells do not form one interconnected pathway. One way to optimize the value of DGRID is to tune it to best reproduce the diffusion over the time interval of the original MD simulation. An example is given in the following graph where the mean square displacement (MSD) divided by 6t is given for CO_2 molecules in a polyimide structure.



Reasonable agreement was obtained in this particular case using a 2Å grid width. It can be seen that the KMC approach predicts a plateau value of MSD/6t at times almost two orders of magnitude longer than the original 6 ns MD simulation. Data for the MSDs of the random walks is written to the file `DISPX.OUT`.

gmq_dmom (v1.3 - 18/09/2013)

This program calculates the total dipole moment vector from the configurations stored in `config.bin`; details of the definition of the total dipole moment vector and other relevant information are given in Section 5.6. The three components of the vector are output, in units of Debye, to the file called `DMOM_from_bin`. The program also calculates and prints to the screen the root mean square and mean magnitudes of the molecular dipole moment vector. The individual averages for each molecule are written to the file `DMOM_MOL_LIST`.

gmq_en (v3.4 - 19/07/2019)

This code is used to calculate the separate contributions of the intramolecular and intermolecular interactions to the total non-bonded energy. For reasons of efficiency this is not done in the main MD codes. However, *gmq_en* allows the configurations stored in the `config.bin` file to be read back in and the corresponding intramolecular contributions to the non-bonded energies calculated. *gmq_en* produces two output files, `OUTPUT.GMQ_EN` and `STO.GMQ_EN`. These two files are equivalent to the files `OUTPUT.NEW` and `STO1` written by *gmq*. `STO.GMQ_EN` contains a table with columns labelled in the following way:-

`TIME INTERPE INTERVdW INTERC VdWLRC EWALD DENSITY TEMP INTRAB INTRANB INTRAVdW INTRAC`

The columns headed `TIME`, `VdWLRC`, `EWALD`, `DENSITY` and `TEMP` are the same as for `OUTPUT.NEW` and `STO`. `INTERPE` gives the total intermolecular potential energy, or lattice energy, and `INTERVdW` and `INTERC` are the resolutions of the intermolecular energy into the contributions due to VdW and Coulombic interactions, respectively. `INTRAB` is the total bonded contribution to the intramolecular energy. `INTRANB` is the total non-bonded contribution to the intramolecular energy and `INTRAVdW` and `INTRAC` are, respectively, its resolution into the VdW and Coulombic parts. The same columns are found in the `OUTPUT.GMQ_EN` file except that, as in *gmq*, `STEP` is given in place of TIME.

The program *gmq_en* works by calculating all the energies for a user-defined range of the stored coordinates read-in from the `config.bin` file in the same way as in *gmq* but then does a separate calculation of the intramolecular non-bonded contribution. This second calculation is made in the absence of periodic boundary conditions, i.e. using the unfolded coordinates of the molecules, and no explicit truncation is applied. Coulombic interactions are simply direct 1/r summations. For molecules of finite size there will be some very minor uncertainties in the resultant intermolecular non-bonded energies, obtained by subtracting the intramolecular contributions from the total non-bonded energies, which result from the convergence criterion applied to the Ewald sum and the truncation of the non-bonded potential. That is to say, in *gmq* non-bonded intramolecular interactions beyond the cut-off are treated in the same way as intermolecular distances beyond the cut-off, i.e. they are considered as a uniform field and their contribution calculated as a cut-off correction term. For "infinite" molecules, e.g. polymer crystals, the situation is much more complicated as we now have effectively an infinite sum of intramolecular interactions. In this case there will, for the moment, be a systematic error as only those intramolecular distances within the fundamental repeat unit will be taken. This is probably not a big problem for the VdW contribution but for the Coulombic term the approximation is not a good one. Thus, *it is recommended that the output from gmq_en be treated with caution for any system containing "infinite" molecules*. Unfortunately this renders the lattice energy of polymer crystals inaccessible within any certainty.

As *gmq_en* does calculations of the energy it is imperative that the version used corresponds to the potential employed in the simulation, e.g. if you have used a Buckingham version of *gmq* then you will need to use the corresponding Buckingham version of *gmq_en*. It should also be noted that *gmq_en* requires the corresponding `CONFIG.NEW`, `PARAM` and `DATA.NEW` files from the MD run.

The program can also be used to obtain the individual intermolecular potential energies of one or more molecules of the same type. This extra analysis is triggered by the presence of a file called `MOLEN`. This file contains, on two separate lines, the indices of the molecules at the start and end of the range required. These indices must coincide with those in the `CONFIG.NEW` file. At the moment this extra calculation is restricted to small molecules which do not have any intramolecular VdW interactions. In practice the individual intermolecular energies are calculated indirectly by calculating the total intermolecular energy for the full system and then for a system where the target molecule has its charges and VdW interactions switched off. The difference in these two energies is deemed to be the intermolecular energy of the molecule concerned; technically the suppression of one molecule suppresses also all its images so there is an issue related to the reciprocal space part of the Ewald sum. The output for the individual intermolecular energies as a function of time is written to the file `MOLEN.LIST` in units of kT. The program also calculates a probability density of these intermolecular potential energies and writes it to the file `MOLEN.DIST`. This distribution can be compared to the Boltzmann weighted probability density distribution for virtual insertions of a probe, as output by the program *nano_widom*, for insertions of the same type of molecule in the same system (see description of the output file `PEDIST.WIDOM`). This latter function is the product of the unbiased distribution of virtual insertion energies and the corresponding Boltzmann factor, i.e. $\rho_w\left(\Delta\phi\right)=\rho\left(\Delta\phi\right)\exp\left(-\frac{\Delta\phi}{kT}\right)$, and it gives a representation of the energy of likely sites of adsorption. It is directly related to the solubility since $S\approx\left\langle\exp\left(-\frac{\Delta\phi}{kT}\right)\right\rangle=\int_{-\infty}^{+\infty}\rho\left(\Delta\phi\right)\exp\left(-\frac{\Delta\phi}{kT}\right)d\Delta\phi$. To make the comparison with the distribution of individual intermolecular potential energies it has to be renormalized so that the area under the distribution is 1, rather than being the solubility, i.e. $\rho_w^{'}\left(\Delta\phi\right)=\frac{\rho_w\left(\Delta\phi\right)}{S}$.

gmq_hbond (v4.2 - 01/05/2014)

This program performs a number of static and dynamic analyses related to the formation of hydrogen bonds in a system using the stored data in `config.bin`. The user has to first select the atom types of the hydrogen bond donors (H), e.g. the hydrogen in a water molecule, and the atom types of the hydrogen bond acceptors (A), e.g. the oxygen in water. All screen output and input is also written to the file `HBOND.OUT` for reference.

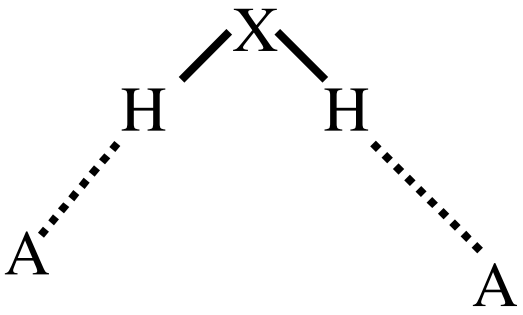
After selecting the range of configurations to analyse, the user then chooses the critical H···A distance (r_{hcut}) below which an H-bond is deemed to exist; note that, for the moment, no angle criterion is used. This distance is 2.5 Å by default but should be based on the position of the first minimum after the first peak in the corresponding *g(r)* for the atom types concerned. In fact, *gmq_hbond* calculates the radial distribution functions for all H···A pairs and the user can change the resolution of the histogram. The resulting total RDFs are written to the file `HB_RDF.OUT` and the corresponding intramolecular contributions to `HB_RDF_INTRA.OUT`

The user can also set the histogram width for calculation of the X-H···A angle distribution, X being the one atom covalently bonded to the H-bond donor. This angle distribution is collected as a histogram in δCosθ so as to give an equal weight to each bin; slices of equal δCosθ subtend surfaces of equal area on a sphere. Histograms accumulated in δθ are weighted by a term in Sinθ which forces the resulting probability density to zero at an X-H···A angle of 180°. The resulting distribution is written to the file `XHA_ANGLE.PDEN` as a function of Cosθ and also θ.

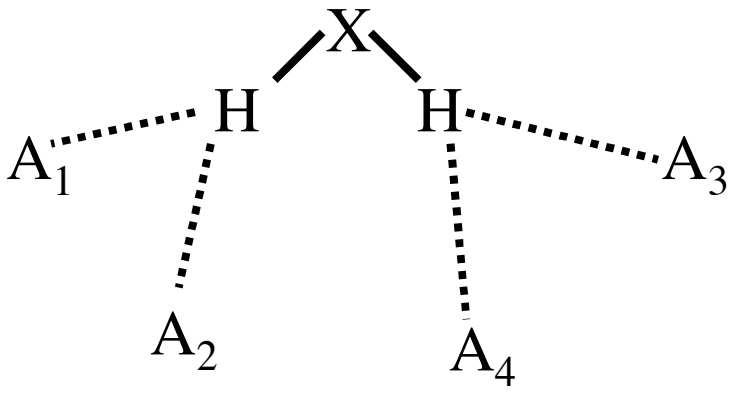
The program calculates a number of averages which are printed to the screen, e.g. the average number of H-bonds per configuration, per donor, per acceptor, etc. The probabilities of the H-bond donors having 0, 1, 2, 3, etc. H-bonds at any one particular instant are written to the file `DHBOND.DIST`; in the column headed TOTAL. A one-dimensional resolution of this distribution by acceptor types is also given in this file but as this is of limited use a full resolution of the H-bond network is also printed to the screen and the file `HBOND.OUT`. Thus the numbers of percentages of each type of possible permutation of H-bond with the different acceptor types is given.

The corresponding distribution for the acceptors is written to the file `AHBOND.DIST`. A separate distribution is also calculated for the X, bonded-to-H-donor, atoms as it is possible that the X atom is bonded to two or more H-bond donor atoms, e.g. the oxygen in water. Thus, at any particular moment in time, an X atom could be indirectly linked to a number of H-bond acceptor atoms via its H-donor atoms. The probabilities of the number of acceptors indirectly linked to X are written to the file `XHBOND.DIST`. Again a one-dimensional resolution is made by acceptor type and a full resolution is written to the screen.

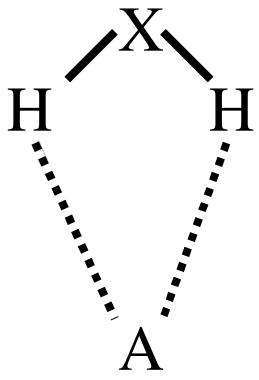
The possibility of X being bonded to two, or more, H-donor atoms leads naturally to the concept of indirect bridging between two acceptors:-



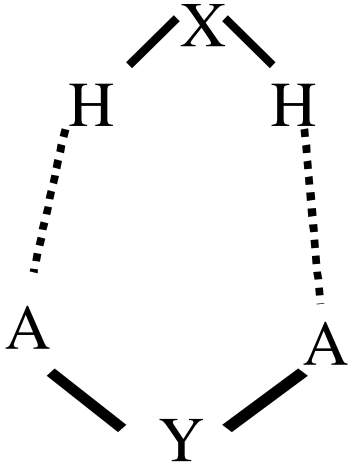
The program reports the average numbers and percentages of A···H-X-H···A bridges per configuration and per X. As H-bond donors can be associated to more than one acceptor various permutations on bridges can exist for the same X, as demonstrated in the following figure:-



In the averages all possible permutations are considered; in the case in the diagram above there are four bridges passing through the central X atom:- A_1···H-X-H···A_3, A_1···H-X-H···A_4, A_2···H-X-H···A_3, and A_2···H-X-H···A_4. A distinction is made between the average number of bridges per X atom, i.e. the total number of bridges divided by the total number of X atoms, and the average number of bridges per X atom *for those X atoms that have at least one bridge*, i.e. the multiplicity of bridges. An analysis of the bridging for each type of X atom is also made with a resolution into intermolecular and intramolecular bridges. The intramolecular bridges are further resolved into three different types. The first type are intramolecular "4-member cyclic" bridges which represent a loop back to the same acceptor, as shown in the diagram following:-



Intramolecular "6-member cyclic" bridges represent the situation shown in the following diagram where the two acceptors that are bridged share a common covalently bonded neighbour (Y):-



The third type of intramolecular bridge are all "other types" that are not either of the first two.

The contribution of all pairs of acceptor sites involved in bridges to the corresponding radial distribution functions are written to the files `HB_RDF.OUT` and `HB_RDF_INTRA.OUT`. It should be noted that these will just be partial RDF as only distances between acceptor pairs that are bridged are counted. These partial RDFs can though be compared with the results for the complete RDFs generated by *gmq_rdf*.

A list of molecules that act as donors is written to the file `DONOR_MOLEC.LIST`.

In terms of the time dependence, several analyses are currently available. The first is the cumulative number of acceptors visited by the H-donors. A count is made of all the A atoms that have formed an H-bond at some point with an acceptor since the beginning of the period studied. This number can only increase or stay constant depending on the relative mobility of the donors and acceptors in the system. The variation of these cumulative numbers, and their corresponding percentages, is output to the file `CUMUL_ACC_VISITED` for each of the different acceptor types selected by the user.

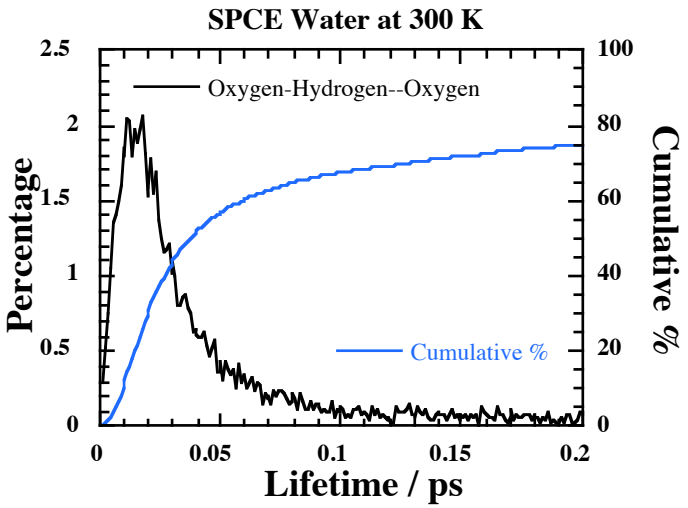
A second complementary analysis is also carried out from the point of view of each bonded-to-H-donor X atom. Again a cumulative count is made of the number of acceptors that are linked via a H-bond to an X atom. The average cumulative numbers and percentages of acceptors visited per X atom are then written to the file `CUMUL_ACC_VISITED_PERX` during the course of the analysis for the range of records chosen. A resolution by type of acceptor is made. At the end of the analysis the final cumulative numbers of acceptors visited by each X atom are used to construct a probability density. This data is written to the file `PVISIT.PDEN`.

A cumulative number of bridging sites visited in the user-selected period is also written out to the file `CUMUL_BRIDGE_VISITED`. The total number of bridges visited as well as a resolution into the different permutations of pairs of acceptors is given.

N.B. These cumulative functions are only the results from one time origin using the entire range of stored configurations chosen. There is no averaging over time origins for the moment.

An analysis is also made of the lifetime of individual H-bonds. Each time an H-bond is formed a clock is started. The clock is stopped as soon as the H-bond is broken and this lifetime is used to calculate an average lifetime and also a distribution of lifetimes for each type of X-H$^{\cdots}$A. The averages and probability density distributions are written to the output file `XHA_LIFETIME.PDEN`.

This form of lifetime analysis does have two drawbacks though. First, it does not take into account the eventual reformation of a particular H-bond. Each time the H-bond is broken the lifetime clock is stopped and starts again from zero should the H-bond reform. The averages and distributions are also rather sensitive to the time interval (`WHENST`) between the stored configurations. A `WHENST` which is too long will not be able to resolve lifetimes shorter than the storage interval and a short `WHENST` will quickly lead to excessively large `config.bin` files. Nevertheless, short ancillary runs can always be used to obtain the lifetime distribution at high resolution by storing data at each step. An example of the results that are obtained when doing this for liquid SPCE water at 300 K (10000 steps of 1 ps) are shown in the following graph:-



The distribution (in black) shows that a peak in the lifetime is observed at very short times of 10 to 20 fs and that lifetimes less than 0.2 ps make up 75% of all those recorded. The distribution has a slowly decaying tail, corresponding to longer-lived H-bonds, and consequently the average lifetime is ~0.34 ps. ***N.B.*** *It is imperative for this analysis that stored configurations are evenly spaced in time. A warning is issued if this is not the case and the results of this analysis should be discarded.*

To provide complementary information concerning the dynamics of H-bonds, an extra, optional, correlation function analysis is proposed. This is based on the original method proposed by Stillinger [49] and later used by Rapaport [50] and is similar to the relaxation function analysis of torsion angle isomerization performed by *gmq_rfunc*. For each pair, *ij*, of possible H-bond donors and acceptors a function is defined in the following manner:-

$$H\{r_{ij}(t)\} = 1 \quad \text{if } r_{ij}(t) \le r_{hcut} \quad \text{and} \quad H\{r_{ij}(t)\} = 0 \quad \text{if } r_{ij}(t) > r_{hcut}$$

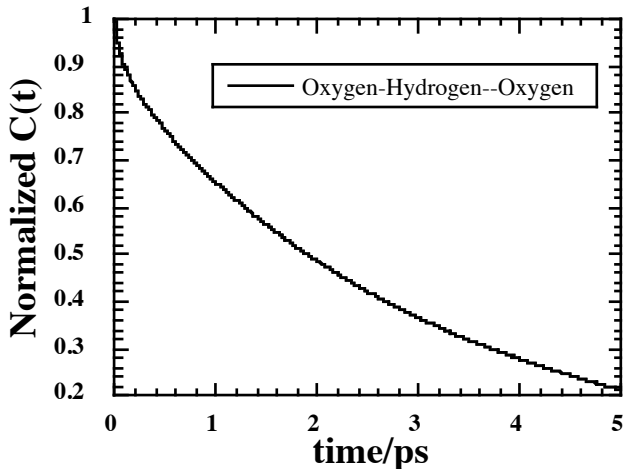
The program then computes the auto-correlation function

$$C(t) = \langle H\{r_{ij}(\tau)\}H\{r_{ij}(\tau+t)\}\rangle = \langle H\{r_{ij}(0)\}H\{r_{ij}(t)\}\rangle$$

for each type of X-H···A and the normalized forms are written to the file `XHA.RFUNC`. The normalization is straightforward for these types of functions which only take values of 1 and 0 as $C(0) = \langle H\{r_{ij}(0)\}\mathrm{H}\{r_{ij}(0)\}\rangle = \langle H\{r_{ij}(0)\}\rangle$ thus

$$\hat{C}(t) = \frac{C(t) - \left\langle H\left\{r_{ij}(0)\right\}\right\rangle^2}{\left\langle H^2\left\{r_{ij}(0)\right\}\right\rangle - \left\langle H\left\{r_{ij}(0)\right\}\right\rangle^2} = \frac{C(t) - \left\langle H\left\{r_{ij}(0)\right\}\right\rangle^2}{\left\langle H\left\{r_{ij}(0)\right\}\right\rangle - \left\langle H\left\{r_{ij}(0)\right\}\right\rangle^2} = \frac{C(t) - C^2(0)}{C(0) - C^2(0)}$$

The normalization factors C(0) and $C^2(0)$ are written to the screen and to the `XHA.RFUNC` file. This function gives the probability of an H-bond still existing between two atoms at some later time given that it did exist at the time origin. As in the intervening time it could have been broken and reformed several times it does not give the same information as the lifetime analysis. An example for the same water system as shown for the lifetime analysis is given in the following graph:-



The much slower decay in the correlation demonstrates that in water there is a high probability that an H-bond that breaks between two specific atoms will reform again.

If the user invokes the option to calculate the relaxation functions, two other parameters are demanded by the code. First is the maximum time interval, t_{max}, out to which the function will be evaluated. This is limited at the upper end by the time between the first and last of the chosen stored configurations. The second parameter to choose is the time interval between separate time origins. All the configurations that have been selected for analysis can act as a time origin, τ. However, if the storage interval is very short then there will be a correlation between time origins if they are too close. More time origins and longer maximum time intervals increase the CPU time necessary to calculate the relaxation. The user is advised to first test the CPU time required by restricting the analysis to just a small part of the stored data and for asking for a small t_{max} and an interval between time origins of the order of a few ps.

Note that, unlike the lifetime analysis, the correlation function approach can tolerate `config.bin` files where records from a certain time period are missing. Effectively the time written out with each record is used to calculate the time difference.

N.B. *although a number of H-bond donor types can be defined by the user the static distributions will not, for the moment, make distinctions beween them nor any eventual differences in the type of the unique X atom bonded to the H-bond donor, i.e. only acceptor types are used to resolve the distributions. For the lifetime analysis and the relaxation function analysis a distinction based on the types of X, H & A is made.*

gmq_rdf (v2.1 - 20/09/2018)

Calculates the radial distribution functions, g(r), from the `config.bin` file. The user has three options:-

(1) Select all possible RDFs. *NB.* there will be one g(r) for each possible combination of atom types in the output files. Output is written to the files `RDF.OUT` , `RDF_INTER.OUT` and `RDF_INTRA.OUT`. The

first file contains the total g(r), the second just the contribution of intermolecular interactions and the last the contribution of intramolecular interactions.
(2) Read/Create RDFs required file (`rdf.types`). If this option is selected and the file called `rdf.types` does not exist then a default one is written. This file can be edited in order to control which atom-type pairs will have their g(r)s calculated. The program then needs to be run again. The output is written as for the first option.
(3) Calculate total indiscriminate RDF. This option treats all atoms as if they were the same type in order to determine a g(r) averaged over all atoms.

The program also writes out the corresponding $n_{ij}(r)$:

$$n_{ij}(r) = \frac{n_j}{V} \int_0^\infty 4\pi r^2 g_{ij}(r)\ dr$$

i.e. the number of atoms of type *j* in a sphere of radius r around an atom of type *i*. Note that, unlike $g_{ij}(r)$, $n_{ij}(r)$ is not necessarily the same as $n_{ji}(r)$, unless $n_i = n_j$. For this reason two files are generally written out `NOFRIJ.OUT` and `NOFRJI.OUT`. There is also a resolution of the intramolecular contributions to $n_{ji}(r)$ and these are written to `NOFRIJ_INTRA.OUT` and `NOFRJI_INTRA.OUT`

gmq_rfunc (v2.2 - 11/03/2015)

In *gmq* torsions are arbitrarily designated to be in a certain state based on the actual value of the angle. From -180° to -60° is considered as a *gauche minus* (G^-) state, from -60° to +60° is *trans* (T) and from +60° to +180° is *gauche plus* (G^+). These states are attributed irrespective of the actual torsional potential so for non-threefold symmetry torsions they have little meaning. The *gmq_rfunc* program calculates the relaxation functions [26] (auto- and cross-correlation functions) for the state of a torsion angle. It uses as input the compressed list of torsional states, as a function of time, kept in the file `INDEX`. If the `INDEX` file does not exist, as is the case if *ddgmq* has been used to do the MD, then it can be created from the `config.bin` file using *gmq_index*.

The relaxation function methodology of characterizing rates of conformational isomerization has been demonstrated in the case of n-butane [26]. Previously it had been thought that the correlation function relating the rate constant for the dynamic equilibrium between the *gauche* and *trans* states was a collective property [51]. However, a ‘thought experiment’ led directly to the realisation that it is in fact the auto-correlation function for the state which is required. In this experiment we identify a sub-system of molecules which at some particular time are all in the same state *e.g. trans*. Instantaneously the distribution in this sub-system corresponds to a non-equilibrium state; it will be driven subsequently towards an equilibrium distribution according to spontaneous fluctuations in the system. A relaxation function can then be obtained by averaging the data obtained from many such sub-systems. This function contains all the information required to define the relaxation of the system to equilibrium. It must be emphasised that this involves no external disturbance of the system and all the analysis can be carried out by post-processing the data from the equilibrium simulation.

Taking a bulk n-butane system as an example consider a system of N molecules. Let $H_T(\alpha_i(t))$ be the characteristic function of the *trans* state where $\alpha_i(t)$ is the dihedral angle of molecule *i* at time *t* and

$$H_T(\alpha_i(t)) = 1 \quad \text{if} \quad -60^\circ < \alpha_i(t) < 60^\circ$$

$$\text{otherwise} \quad H_T(\alpha_i(t)) = 0$$

The number of molecules in the *trans* state at a particular time t_0 can then be written as

$$N_T(t_0) = \sum_{i=1}^{N} H_T\left(\alpha_i\left(t_0\right)\right)$$

At a later time t_0+t the number that are in the *trans* state of those that were originally *trans* is simply

$$N_T(t_0, t_0+t) = \sum_{i=1}^{N} H_T\left(\alpha_i\left(t_0\right)\right) H_T\left(\alpha_i\left(t_0+t\right)\right)$$

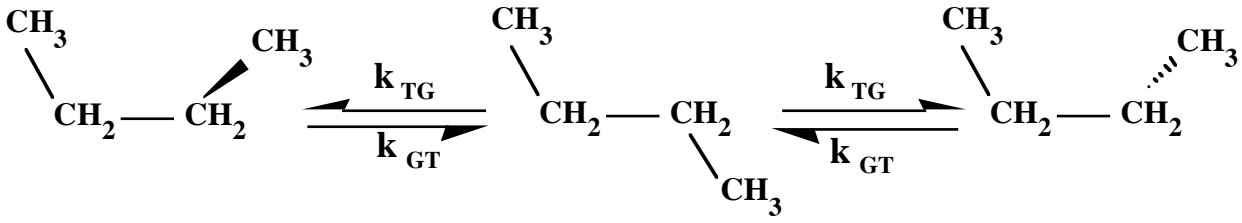
as only those terms where both *H* functions are equal to one contribute to the sum. The relaxation function, $R_{TT}(t)$, in this case can be defined as

$$R_{TT}(t) = \frac{N_T(t_0, t_0+t)}{N} = \left\langle H_T\left(\alpha_i\left(0\right)\right) H_T\left(\alpha_i\left(t\right)\right)\right\rangle$$

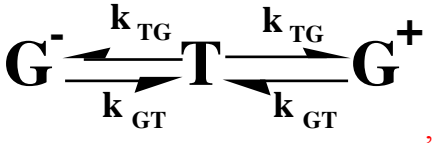
where the angle bracket denotes an average over time origins, t_0. Clearly $R_{TT}(0)=\langle X_T(t)\rangle=\langle X_T\rangle$, *i.e.* the mean fraction of molecules in the *trans* state, and at long times $R_{TT}(t)$ will tend to $\langle X_T\rangle^2$.

$R_{TT}(t)$ has exactly the time dependence required to describe the way in which on average conformational equilibrium is established in a sub-system. By selecting that subset of particles which are *trans* at any particular point in time we perform an analysis similar to the non-equilibrium experiment of Edberg *et al* [52] *but without disturbing the system at all*. The comparison with their approach can be obtained simply by dividing $R_{TT}(t)$ by $\langle X_T\rangle$ which is tantamount to starting with a configuration of all *trans* molecules.

A demonstration of the power of this technique was given for the case of n-butane [26] where it was found that the long accepted two-rate-constant mechanism,



or more succinctly



,

was inadequate. In particular it did not explain the observed behaviour of the following relaxation functions

$$R_{G^-G^+}(t) = \frac{\left\langle \sum_{i=1}^{N} H_{G^-}(\alpha_i(t_0) H_{G^+}(\alpha_i(t_0 + t) \right\rangle}{N}$$

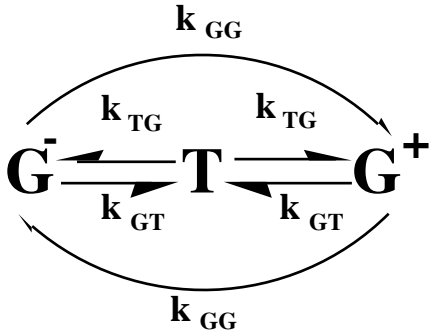
and

$$R_{G^+G^-}(t) = \frac{\left\langle \sum_{i=1}^{N} H_{G^+}(\alpha_i(t_0) H_{G^-}(\alpha_i(t_0 + t) \right\rangle}{N}$$

which by symmetry are equivalent so can be defined as

$$R_{G^\pm G^\mp}(t) = \frac{R_{G^-G^+}(t) + R_{G^+G^-}(t)}{2}$$

These functions determine the way in which the population of the opposite, initially unoccupied, *gauche* well changes with time given that the *trans* well is also initially unoccupied. Examination of the results obtained revealed immediately that the rate of change of $R_{G^\pm G^\mp}(t)$ was not consistent with the kinetic equations of the original mechanism which predicted zero slope at t=0. It could thus be concluded that the traditional *trans*⇔*gauche* interconversion mechanism was not sufficient to explain the observed data. Only by including direct *gauche-gauche* interconversions through the *trans* well could the data be explained, *i.e.* a mechanism of the form



In this mechanism direct interconversions are postulated between the two *gauche* states and so a third rate constant k_{GG} has been introduced. The third process involves direct G→G transitions via the *trans* well. They do not involve crossings of the very high $G^-⇔G^+$ barrier. These are close to adiabatic or resonant transitions in which there is not sufficient energy dissipated for the molecule to settle in the *trans* configuration.

The relaxation function approach is also applicable to molecules with multiple torsions. In this respect it is a very powerful tool to assess whether conformational equilibrium has been attained for *all* torsion angles of the same type in the entire system. The fact that, when in equilibrium, $R_{TT}(t)$ varies from

an initial value of $<X_T>$ to a value at long times of $<X_T>^2$ means that the limiting value of $R_{TT}(t)$ is predictable from its initial value! A relaxation function that does not do this is a sure sign that the system is not completely in equilibrium.

Once *gmq_rfunc* is invoked the user can specify which torsion angles to include in the calculation based either on the types given in the `PARAM` file or from a custom list; groups of four global atom indices per line in file of arbitrary name. A list of all possible torsions, and whether they are included in the averaging, and a list of included torsions can be written to the files `TOR.LIST` and `INC.LIST`. To choose specific torsions the user has the possibility of defining a mask file. The mask file contains a list of atom indices, one per line. If the two atoms forming the central bond of a torsion are in the mask file then the torsion is used in the averages. The file `INC.LIST` can be useful as a basis for making a mask file.

The user then chooses the time range of the data to use and the maximum time difference out to which the relaxation functions will be calculated as well as the required time between time origins. The relaxation functions are written to the file `RFUNC.OUT`. All 9 possible relaxation functions are written out in 9 columns after the time column:-

`RTG-  RTT  RTG+  RG-G-  RG-T  RG-G+  RG+G-  RG+T  RG+G+`

The percentages of conformers are written as a function of time for the configuration analysed to the file `PERCT`.

gmq_rgete (v1.8 - 18/09/2013)

Calculates the square end-to-end distances and square radii of gyration as a function of time of user selected molecules using the data on `config.bin`. Molecules can be selected by atom type if just the ends have the same atom type. For linear chains an automatic search can be made and the search can be limited to molecules with greater than a user-specified number of atoms; this is useful in the case where low molecular weight solvent molecules are present. The user can also choose to ignore atoms with a molar mass less than a certain threshold; this is useful for excluding light atoms like hydrogens.

This calculation only makes sense for linear chains of the same length. Output for the average over all molecules included is written to `RG_ETE`. Optionally individual values for each molecule can be written to the file `RG_ETEX`. In this latter case the user is also given the option to calculate the distribution of square end-to-end distances and square radii of gyration. This distribution, if calculated, is written to the file `PROB_RG_ETE`.

gmq_prs

This program uses the `RG_ETEX` file created by *gmq_rgete* to work out the distributions of square end-to-end distances and square radii of gyration. The distributions are written to the file `prob_sr`.

gmq_ringcos (v1.1 18/09/2013)

gmq_ringcos does an automatic search of a `CONFIG.NEW` file for rings containing a user-defined maximum number of atoms in a ring. It automatically assigns a Ring Type based on the sequence of atoms in the rings that are found. A user-defined range of configs. in the config.bin file is then analysed by first calculating the vector normal to the best-fit plane through the atoms in each ring. These normal vectors are then used to produce the second order Legendre polynomial function

$$P_2\left(Cos\theta\right)=\frac{3}{2}\left\langle Cos^2\theta\right\rangle-\frac{1}{2}$$

where θ is the angle between the vectors normal to the planes of rings *i* and *j*. $<P_2(Cos\theta)>$ = 1 for a parallel orientation, -0.5 for a perpendicular orientation, and zero for a random orientation. The function is calculated as a function of R_{ij}, the distance between the centroids of the rings, and is output to the files `RINGCOS.INTRA`, `RINGCOS.INTER` and `RINGCOS.TOTAL`. A resolution is made in these files into the different Ring Type combinations, 1-1, 1-2, 1-3, 2-2 etc. but the next to last column gives the indiscriminate average. The last column gives the average number of ring pairs considered per config.

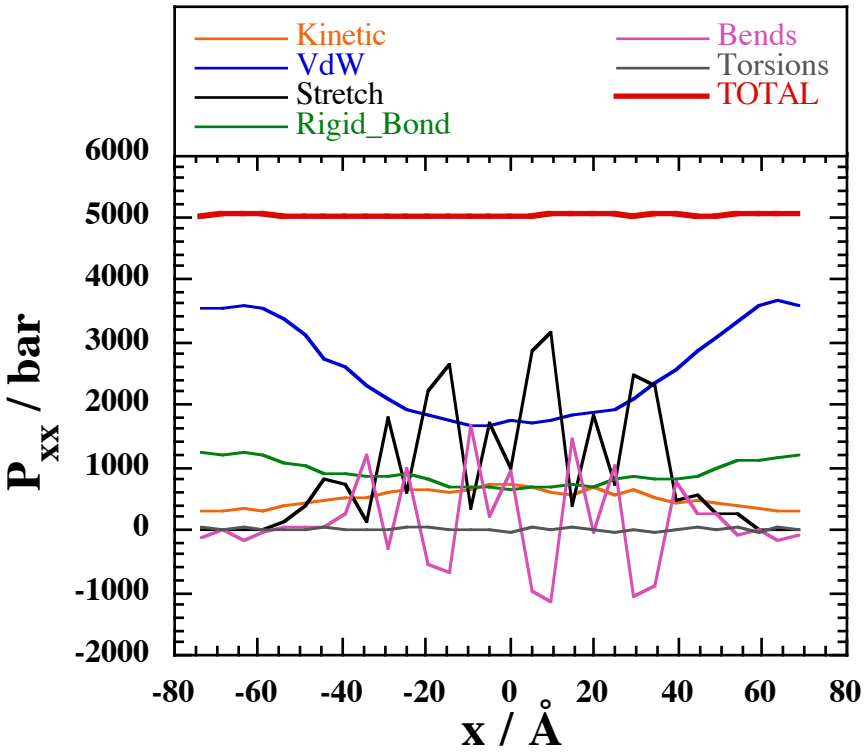
In the file `RINGCOS.CHAIN`, $<P_2(Cos\theta)>$ is given this time as a function of the difference in Ring Index along the chain, i.e. it is a resolution of the intramolecular function in terms of separation of rings down the chain rather than in terms of the distance between centroids. A further file called `RINGCOS.ANGLE` is produced which contains the average angle between the normal vectors for rings in the same chain; again as a function of the difference between ring indices. To avoid any irrelevant distinction between orientations, this angle is calculated from the absolute value of the dot product between the normal vectors, $\theta=arccos(|Cos\theta|)$ so is limited to the range from 0 to 90 degrees. It should be noted that this latter function will tend to a value of 1 radian, i.e. about 57.3 degrees, for a random orientation.

gmq_player

gmq_player is not a post-processing analysis program but is a custom version of *gmq* which calculates the different contributions to the pressure tensor at virtual interfaces in the system using the mechanical definition of the pressure as a rate of momentum flux [18]. The virtual interfaces are internally constructed in the directions perpendicular to the laboratory x, y and z axes. The numbers and distances between the planes in each laboratory direction is governed by the largest real space cutoff and the shape and size of the MD box. For the moment no long range corrections to the interfacial pressure are attempted so the use of *gmq_player* is limited to systems where the non-bonded interaction potentials are finite in range, i.e. IPOT=2. In addition, *gmq_player* can only be used for fixed volume simulations. The interfacial pressure tensor is resolved into the following contributions:- Kinetic, VdW, Stretch (flexible bonds), Rigid Bonds, Bends, Torsions, Out-of-Planes, and TOTAL. For α=constant planes (α=x,

y, or z) the average contributions to the αx, αy, and αz elements of the pressure tensor are printed out and sub-averaged data is written to the nine files `PLAY_`αβ with (α and β=x, y, or z).

The figure below shows the average results for the xx component of the pressure tensor at different interfaces through a model nanocomposite system where a single 60 Å radius nanoparticle is embedded in a polymer matrix at a volume fraction of filler of ~27% [53].



The model nanoparticle in this case being just a framework of atoms held together by flexible bonds and bends, i.e. no non-bonded interactions, there is a clear reduction in the contribution of the VdW term towards the centre of the system with a compensating rise in the bonded contributions. Despite the inhomogeneity of the system the average total value of P_{xx} at each interface is constant, as is expected from mechanical equilibrium.

gmq_super (v1.5 24/09/2018)

A program which uses stored configurations to determine the mean least-square superimposed difference between the configuration of a molecule at one time with that at some reference point in time. To do this the coordinates of the molecule at the reference point are superimposed on the coordinates of the same molecule at a different point in time making use of just rigid body rotations and translations. The superimposition algorithm attempts to minimise the least squares difference between the two sets of coordinates. The principle output is this least-squares difference as a function of time and it is written to `SUPER_DIST`. As well as the average for all atoms included a break down by atom type is given.

The user has the option of choosing whether one or more molecules should be used. In the case of more than one molecule being included the average is written to the output file. It is for the user to decide whether this average is meaningful in the case where different types of molecule are chosen.

The user also has total control over which atoms are included in the actual superimposition. Many of the possible choices are intended for protein-type molecules but there is the possibility of reading in a custom list of atoms to include. The least-squares difference output is that for those atoms actually included in the superimposition.

Other options allow for the writing of a complete list of individual least-squares differences to a file called `SUPERX.` This file can be used within the program in order to calculate the distribution of least-square differences. This distribution is then written to a file called `SUPER_PROB`.

gmq_torseq (v1.1 28/05/2007)

Calculates the probability distribution of consecutive sequences of *trans* angles. The user can select the types of torsions to include. The output is written to `TORSEQ.OUT`.

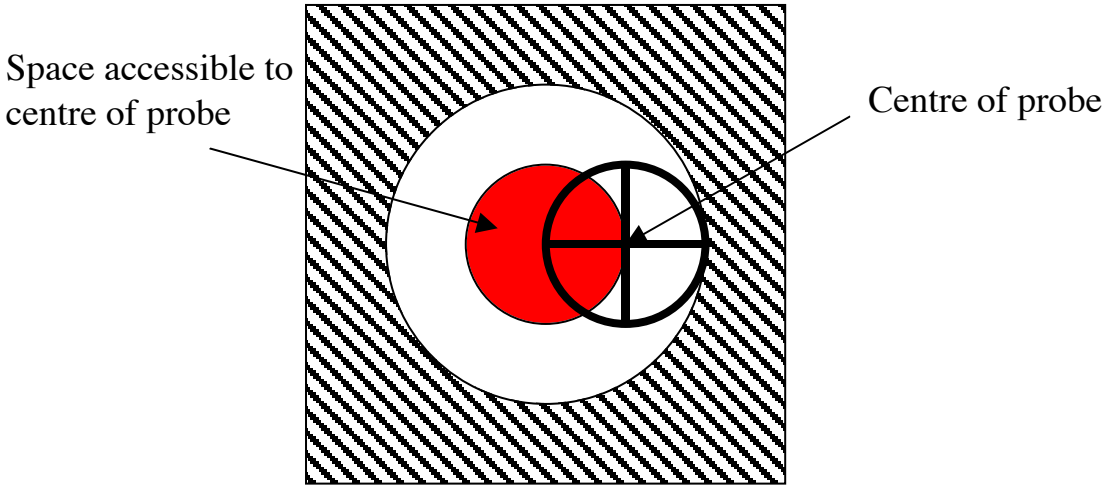
torcor

This program calculates correlation functions for the vectors associated with torsion angles. It is primarily intended for use with the output generated by *gmq_dis* for a list of torsion angles as a function of time. For a given torsion angle, α_i, the associated vector is defined as (*Cos* α_i, *Sin* α_i). The use of this vector instead of the torsion angle itself avoids the problem of discontinuities around ±180°. The user has the choice of looking at self- or cross-correlations, e.g. between neighbouring torsions.

gmq_void (v2.4 - 07/06/2016)

This program calculates the void space in a system from stored configurations using a user-specified probe radius and atom sizes. By default the probe radius, R_p, is set to 2 Å. The first time the program is invoked it creates a file called `inc.types`. This file contains a list of the different atom types and can be used to include or exclude atom types from the void analysis and to set the radius of the atom type, R_i. This `inc.types` file should thus be edited and the logical variable changed from the default false, "F", to true, "T", for each atom type to be included and the radius changed from the default value of 1.0 Å to that required. *gmq_void* can then be run a second time. For each configuration selected the program attempts to insert the probe at random positions in the system. If no atom in the system falls within the combined distance of the probe radius plus the corresponding atom type radius, R_p+R_i, then the insertion is considered accepted. The maximum number of trials per configuration is set to 1000 $Å^{-3}$ and the preferred number of accepted trials is set to 1 $Å^{-3}$. If either of these thresholds is reached the trial insertions stop for the configuration in question. If the number of trials exceeds the maximum then a warning is issued to the effect that the probe radius is probably too large. The void volume is defined as the total volume multiplied by the ratio of the number of accepted insertions to the total number of trial insertions. The data for each configuration selected is written to the file `VOID.OUT`.

N.B. this definition of "void space" is *just the space accessible to the* ***centre*** *of the probe*. There is no attempt to calculate the space "occupied" by the probe. The schematic diagram that follows illustrates, in 2D, the notion of probe centre accessible volume. The probe, circle with cross-hair, is in a circular cavity of radius two times that of the probe. The probe centre accessible space is shown in red. It is, in this case, just a circle of radius half that of the cavity.



It should be noted that the void space, as defined here, is certainly a function of the probe radius so care has to be taken as to its interpretation. A more thorough study of the dependence of the void volume on the probe radius is advisable as this will give a curve characteristic of the structure. These curves can then be compared for different systems.

Optionally the user can decide to perform a more detailed analysis of the distribution of void space. This is done by a standard cluster analysis of the accepted probe insertions. If two accepted probe insertions fall within a cluster cutoff distance, 0.5 Å by default, then they are considered to be connected and thus in the same void. By checking all possible distances between accepted probes a connectivity diagram is established and it is this which is subject to a cluster analysis. All interconnected insertions are considered to be in the same void. The results of this cluster analysis are the number of clusters found of a given size and the proportion of the probe accessible volume that this corresponds to. This data is written to the file `VOID_CLUST.OUT`. The same data is also converted to a distribution of probe accessible space and written to the file `VOID_DIST.OUT`. This is in the form of a histogram where the volume of probe accessible space that is to be found in voids of 0-10 $Å^3$, 10-20 $Å^3$, etc. is given. ***NB.*** memory restrictions mean that there is an upper limit to the number of accepted insertions per configuration that can be stored for doing the cluster analysis. This is currently set to 50000.

The user can also chose to display the successful insertions. In this case a large file called `VOID.XYZ` is created containing the coordinates of the successful insertions for each configuration selected. This data can then be displayed using *gnuplot* by invoking the shellscript *anim_void*. As for the cluster analysis, memory limitations restrict the number of accepted insertions per config. to 50000.

hdatplav

Converts the 3×3 **h**/2 matrix, made up from the three basis (column) vectors {***a****/2*, ***b****/2*, ***c****/2*}, data in a file of `HS` format to the form of basis vector magnitudes and angles between the basis vectors. The data is written to the file `HDAT`. The first column contains the time in ps. There then follow the three basis vector magnitudes $|\boldsymbol{a}|$, $|\boldsymbol{b}|$ and $|\boldsymbol{c}|$ in Ångstroms, the angles between the basis vectors , $\alpha=cos^{-1}\{(\boldsymbol{b}\bullet\boldsymbol{c})/|\boldsymbol{b}||\boldsymbol{c}|)\}$, $\beta=cos^{-1}\{(\boldsymbol{c}\bullet\boldsymbol{a})/|\boldsymbol{c}||\boldsymbol{a}|)\}$, and $\gamma=cos^{-1}\{(\boldsymbol{a}\bullet\boldsymbol{b})/|\boldsymbol{a}||\boldsymbol{b}|)\}$ in degrees, and finally the volume in $Å^3$.

The contents of the resulting `HDAT` file can also be viewed by executing the *gnu_hx* (X-windows) shell scripts which invoke *gnuplot.*

hdatfrombin

Generates a `HDAT` file from the data stored in `config.bin`. This can be useful if information about the MD box is required that is synchronized with the `config.bin` file.

htarplav (v1.2 - 16/06/2020)

This program is used to extract from the `HS` file the extension/contraction following a uniaxial tension non-equilibrium simulation. The user is first asked in which direction the tension was applied; the convention used is that 1, 2 or 3 indicate along the X, Y or Z laboratory axes, respectively. For example, if the yy component of the applied pressure tensor was changed in order to deform the system then the correct response is 2. The user is then asked to specify the times at which the baseline part of the run starts and finishes. Here the program assumes that the user has carried out first an equilibrium simulation in order to establish a base line for the average shape of the MD box prior to the non-equilibrium experiment. For example, if the first 100ps of the simulation were conducted under equilibrium conditions, before starting to change the applied pressure tensor, and the `HS` file contains one entry per ps then the correct responses here are 0.0 and 100.0. The program then processes the `HS` file to produce a file called `HTAR` containing a number of columns. The first column is the time in ps there then follow the effective "length", "area" and "width" (area$^{1/2}$) of the sample, the computed strains in these three properties, the Poisson's ratio and then the percentage strain in the "length", i.e. in the direction of elongation, and the percentage contraction in the "width". As of v1.2, *htarplav* also reads the `ST` file and uses it to add the columns of applied tension and measured tension directly to the `HTAR` file.

It is recommended that the `HTAR` file be subsequently processed using *datploav* as it produces an output file with meaningful headers for the columns.

nano_cos (v1.7 - 16/10/2013)

This program performs a spatial resolution of the 1st and 2nd order Legendre polynomials for the orientation of bond angle vectors, i.e. the vector $\boldsymbol{r}_{ik}$ in the triplet $\{i,j,k\}$ constituting a bond angle. The user can specify which types of angles to include in the analysis. The 1st and 2nd order Legendre polynomials are defined as:

$$P_1\left(Cos\theta\right) = \left\langle Cos\theta \right\rangle$$

and

$$P_2\left(Cos\theta\right) = \frac{3}{2}\left\langle Cos^2\theta \right\rangle - \frac{1}{2}$$

where θ is the angle between $\boldsymbol{r}_{ik}$ and some reference vector. The reference vector depends on the type of spatial resolution required:- (1) Spherical shells or (2) Slabs. Spherical geometry is intended for use with systems which contain a single nanoparticle. The orientation is then obtained relative to a radial vector

from the COM of the nanoparticle to the central (j) atom of the bond angle considered and the spatial resolution is as a function of distance from the COM. Slab geometry is intended for use with membrane systems where there is a flat surface present. In this case the reference vector is chosen in the direction perpendicular to the flat surface, either the *x*, *y* or *z* axis, and the spatial resolution is the distance from the plane containing the COM of the membrane. The output is written to the file `COS.OUT`. Apart from $P_1(Cos\theta)$ and $P_2(Cos\theta)$ and estimation of the errors, the last column gives the average number of angles found at each distance. In the case of slab geometry a symmetrised output is written to the file `COS_SYM.OUT`.

Normally the $P_1(Cos\theta)$ function should average to zero, as $\{i, j, k\}$ is equivalent to $\{k, j, i\}$, but it is useful to calculate it as it gives an idea of the quality of the statistics. The data for $P_2(Cos\theta)$ should be treated with caution in the range where $P_1(Cos\theta)$ is not close to zero.

nano_dens (v2.4 - 21/11/2018)

This program performs a spatial resolution of the mass density. The resolution can either be:- (1) Spherical shells or (2) Slabs. Spherical geometry is intended for use with systems which contain a single nanoparticle. The distance is then obtained relative to the COM of the nanoparticle. The slab geometry is intended for use with membrane systems where there is a flat surface present. In this case the reference vector is chosen in the direction perpendicular to the flat surface, either the *x*, *y* or *z* axis, and the spatial resolution is the distance from the plane containing the COM of the membrane. The user has the choice of using periodic boundary conditions to calculate the distances from the atoms to the COM of the nanoparticle or slab. This should be chosen in the case of a nanoparticle but is not recommended for slab geometry, unless a bulk system is being analysed. The user is asked to define the atom types that make up the nanoparticle or membrane. This information is stored in the file `nano.types`. The user can choose whether to evaluate the density distribution for all atom types or just selected ones. In the latter case the file `dens.types` is used. This file can be created automatically by running *nano_dens* a first time, then editing the resulting default `dens.types` file before re-running the program. The resulting density histograms are written, in units of kg m^{-3}, to the file `SDENS.OUT`. A symmetrized version of this file is written to `SDENS_SYM.OUT`. Corresponding files for the mass, `SMASS.OUT` and `SMASS_SYM.OUT`, and for the number of atoms, `SNUM.OUT` and `SNUM_SYM.OUT`, are also written out.

nano_dens_plus (v2.0 - 16/05/2018)

This program is similar to *nano_dens* but is designed specifically to analyse the uptake of "gas" molecules by a "membrane". The spatial resolution of "gas" atoms is in slabs perpendicular to a user-specified axis. The program *labels* atoms of a user-specified type to be either to the "left", "right", or in the "middle" of the membrane at a time origin, t=0. Atoms in the membrane are defined as in *nano_dens* and the width of the membrane can be determined automatically or specified to be of a fixed width by the user. At a user-specified time interval later, the average numbers of "gas" atoms in slabs, of a user-defined width, are printed out to the file `SDENS_PLUS.OUT`. Separate columns are given for those atoms that were labelled "Left", "Middle" and "Right" at the time origin. The final column gives the

"Left" & "Right" symmetrized result. The results are averaged over all possible time origins that allow the chosen time interval.

By running *nano_dens_plus* with different time intervals a description of how the gas molecules move into, or out of, the membrane can be built up, even if the system is in equilibrium, i.e. it is, in this case, a virtual permeation experiment in the absence of a net flux of material or pressure gradient.

The file `SDENS.OUT` is also created and it gives the mass densities averaged over *just those configurations used as time origins* (N.B. the `SDENS.OUT` file obtained from running *nano_dens_plus* does not necessarily give the same result than that obtained using *nano_dens* for this reason).

In addition, the file `LMR.TIME` gives just the number of gas molecules to the left, right and in the middle of the membrane for each configuration for *just those configurations used as time origins*; use a time interval of zero to obtain a full `LMR.TIME` file with the numbers for each configuration. The `LMR.TIME` file can be useful for monitoring the global approach to equilibrium. Also written to `LMR.TIME` are the box lengths (in Å) and angles.

A template script "run_ndp" is available in `$BOME/BIN` for running "nano_dens_plus" for a series of time intervals. As the replies to "nano_dens_plus" have to be tailored for each system, it is better to copy "run_ndp" and edit it to your needs. The syntax of "run_ndp" is :

run_ndp Start_Conf End_Conf 1st_t_interval Last_t_interval Dt_interval

E.g. run_ndp 1001 2020 10 500 10

will run nano_dens_plus for all time intervals from 10 to 500 ps in intervals of 10 ps using configuration Nos. 1001 to 2020 in the config.bin file. Each `SDENS_PLUS.OUT` output file is individually labelled with the time. E.g. `SDENS_PLUS_340_ps`.
The script also creates a file called "my_list" with a list of all the `SDENS_PLUS.OUT` files created in chronological order.

The profiles in the different `SDENS_PLUS.OUT` files can be animated by copying the template script "anim_ndp" from `$BHOME/BIN`, editing it to your needs, and running it.

The different `SDENS_PLUS.OUT` files can also be analysed to determine, as a function of time, the sum of all atoms to the left of the membrane, in the membrane, and to the right of the membrane in the "L+R_symmetrised" column using the analysis code *sdens_sum*. It uses the list of files "my_list" as input. The position of the right hand edge of the membrane is read from the respective `SDENS_PLUS.OUT` files. The output is written to the file `SUM_SDENS`. It contains the columns:-

```
time_diff/ps  SUM_Reservoir  SUM_Sorption  SUM_Permeation  Right_Edge/Angst.
```

where "SUM_Reservoir" is the number of atoms to the left of the membrane, "SUM_Sorption" is the number in the membrane, and "SUM_Permeation" is the number to the right of the membrane.

sdens_sum

See description of *nano_dens_plus*.

nano_denst (v1.5 - 12/04/2019)

This program is very similar to *nano_dens* but instead of averaging the density, in slabs or shells, over a period of time it writes out the mass density, in slabs or shells, of the selected atom types to the file `SDENST.OUT` for each configuration in the range chosen. For reference, the temperature and total density of the configuration are given in columns 2 and 3, respectively.

nano_disp (v2.0 - 13/02/2018)

This program performs a spatial resolution of the mean square displacements of selected atom types at a user-specified time interval. It is intended to give an idea of the mobility of certain species at different places in the system. As with similar codes a spatial resolution into slabs or shells is possible. In the case of nano_disp, this resolution is based on the position **at the time origin**. *It is important to understand that there is no attempt to restrict the averaging to those atoms that stay in the same bin during the specified time interval!* Atoms that migrate out of the bin of the time origin will still have their contribution added to the distribution corresponding to the time origin. The user has a choice of calculating displacements either with respect to the origin of coordinates ***OR*** with respect to the centre-of-mass of the slab or nanoparticle. This latter option is important in cases where the slab or nanoparticle COM drifts in the course of the simulation. The COM position of the slab or nanoparticle is output to the file `COM.OUT`. The histogram of spatially resolved MSDs is written to the file `BIN.OUT` and a symmetrized version is written to `BIN_SYM.OUT`. For reference the MSDs for all atoms of each type is written to the file `DISP.OUT`.

nano_hfunc (v1.6 - 26/02/2018)

This program calculates relaxation functions of "gas" atoms assigned to three different states according to their positions in a simulation of a planar membrane or planar surface. It works in a way completely analogous to *gmq_rfunc* except the states used are defined by the positions of the atoms to be analysed. As with *gmq_rfunc*, the relaxation functions provide useful information concerning the rate at which an equilibrium is established in the system.Two cases are distinguished in *nano_hfunc*:-
(1) permeation of a membrane by a gas and
(2) adsorption of a gas on the surfaces of an impermeable slab of material.
After first entering the direction perpendicular to the membrane/surface, the user has to choose one of these two options depending on the type of simulation that has been performed. The user then inputs the atom types, or verifies the atom types read from the file `nano.types,` that constitute the membrane or the surface, in the same way as with other *nano_* codes. The atom types in the gas molecules are then defined through the creation and editing of the `hfunc.types` file; an initial run can be used to create the default form of this file which can be edited before re-running. The user then chooses the range of

stored configurations to analyse and the maximum time out to which the relaxation functions will be calculated.

In the case of a permeation analysis the user chooses between an automatic calculation of the membrane half-width or setting a fixed half-width. This is used to define which gas atoms are to the left (L) of the membrane, those that are to the right (R) and those that are in the membrane (M). For example, if we assume that the z direction is perpendicular to the membrane/surface and *d* is the half-width of the membrane then

$$H_L\{z_i(t)\}=1 \quad \text{if} \quad z_{ic}(t)<-d$$
$$\text{otherwise} \quad H_L\{z_i(t)\}=0$$

where $z_{ic}(t)$ is the minimum image distance in the z direction of the atom i from the centre-of-mass (COM) of the membrane/surface. Two other H-functions are defined in a similar way to characterize the M and R states.

$$H_M\{z_i(t)\}=1 \quad \text{if} \quad -d \le z_{ic}(t) \le d \quad \text{otherwise} \quad H_M\{z_i(t)\}=0$$
$$H_R\{z_i(t)\}=1 \quad \text{if} \quad z_{ic}(t)>d \quad \text{otherwise} \quad H_R\{z_i(t)\}=0$$

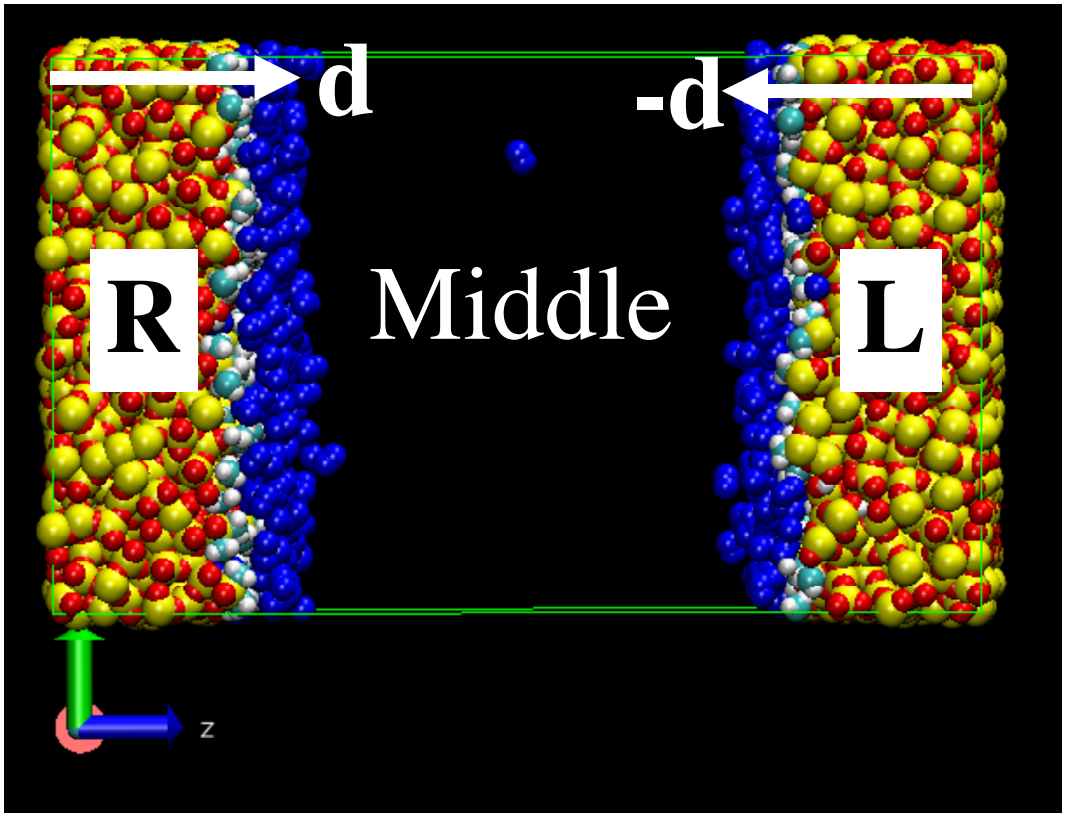
In the case of an adsorption analysis on a left and right surface, the user has to input the half-width distance which corresponds to the outer limit of the adsorbed gas layer. Atoms adsorbed on the left surface (L) are then characterized by the following function

$$H_L\{z_i(t)\}=1 \quad \text{if} \quad -d \le z_{ic}(t)<0$$
$$\text{otherwise} \quad H_L\{z_i(t)\}=0$$

Two other H-functions are defined in a similar way to characterize those gas atoms in the "Middle" (M), i.e. in the gas phase between the two surfaces, and those adsorbed on the right surface (R)

$$H_M\{z_i(t)\}=1 \quad \text{if} \quad z_{ic}(t)<-d \ \text{ or } \ z_{ic}(t)>d \quad \text{otherwise} \quad H_M\{z_i(t)\}=0$$
$$H_R\{z_i(t)\}=1 \quad \text{if} \quad 0 \le z_{ic}(t) \le d \quad \text{otherwise} \quad H_R\{z_i(t)\}=0$$

The following figure shows a representation of a surface simulation in which the COM of the surface has been offset to the edge of the periodic MD box. The surface to the right of the COM thus appears to the left in the picture and vice-versa for the left surface. The gas molecules (shown in blue) in the gas phase in this representation are thus in the "Middle".



In both cases the program then calculates the 9 possible permutations of self and cross-relaxation functions, defined as

$$R_{\alpha\beta}(t) = \left\langle H_\alpha\left\{z_i(0)\right\} H_\beta\left\{z_i(t)\right\}\right\rangle \quad \text{with} \quad \alpha = L, M, R \ \text{ and } \beta = L, M, R$$

As for the analysis of conformational relaxation, these functions characterize the dynamics of the approach to equilibrium. For example, in the case of a gas adsorption on a surface analysis, the relaxation function $R_{MM}(t)$ characterizes the approach to equilibrium of atoms in the gas phase. The zero time value of $R_{MM}(0) = \left\langle H_M\left\{z_i(0)\right\} H_M\left\{z_i(0)\right\}\right\rangle = \left\langle H_M^2\left\{z_i(0)\right\}\right\rangle = \left\langle H_M\left\{z_i(0)\right\}\right\rangle = \left\langle X_M\right\rangle$, i.e. the mean fraction of atoms in the gas phase. This follows as the *H* functions only take values of 1 and 0. At long times this relaxation function thus tends towards $\left\langle H_M\left\{z_i(0)\right\} H_M\left\{z_i(\infty)\right\}\right\rangle = \left\langle H_M\left\{z_i\right\}\right\rangle^2 = \left\langle X_M\right\rangle^2 = R_{MM}^2(0)$. In general then the self relaxation functions decay from $R_{\alpha\alpha}(0)$ to $R_{\alpha\alpha}^2(0)$. The cross-relaxation functions have initial values of zero, as only one of the state functions can be unity at any one time, and must rise to limiting values of $R_{\alpha\alpha}(0)R_{\beta\beta}(0)$.

Assuming that the mechanism of adsorption and desorption can be described by the following coupled equilibria

$$L \underset{k_a}{\overset{k_d}{\rightleftarrows}} M \underset{k_d}{\overset{k_a}{\rightleftarrows}} R$$

and that these processes are first order the chemical kinetic equations can be written down for a system with a fixed total number of gas molecules as

$$\frac{dX_M(t)}{dt} = -2k_a X_M(t) + k_d\left(X_L(t) + X_R(t)\right)$$

$$\frac{dX_L(t)}{dt} = k_a X_M(t) - k_d X_L(t)$$

$$\frac{dX_R(t)}{dt} = k_a X_M(t) - k_d X_R(t)$$

where k_a and k_d are the rate constants for adsorption and desorption and $X_\alpha(t)$ represents the fraction of molecules in the α=*L,M,R* phase such that

$$X_M(t) + X_L(t) + X_R(t) = 1$$

This latter constraint equation can be used to simplify the differential equation for the fraction of gas atoms in the middle

$$\frac{dX_M(t)}{dt} = -2k_a X_M(t) + k_d\left(1 - X_M(t)\right) = k_d - (2k_a + k_d)X_M(t)$$

the resolution of which gives the following solution

$$X_M(t) = X_M(0)\exp(-kt) + \frac{k_d}{k}\left(1 - \exp(-kt)\right)$$

where $k = 2k_a + k_d$. At long times the fraction in the middle tends to the equilibrium value $\langle X_M \rangle = \frac{k_d}{k}$.

The solution for the fraction of gas atoms on the left surface, say, can also be calculated

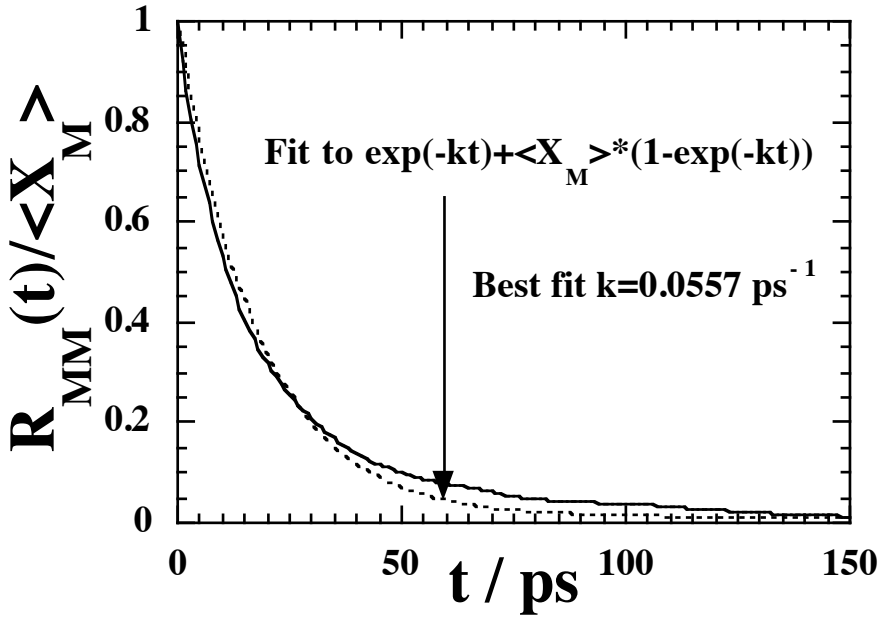
$$X_L(t) = \frac{1}{2}\left(\frac{k_d}{k} - X_M(0)\right)\left(\exp(-kt) - \exp(-k_d t)\right) + X_L(0)\frac{k_d}{k}\exp(-k_d t) + \frac{k_a}{k}\left(1 - \exp(-k_d t)\right)$$

The solution for the fraction of gas molecules adsorbed on the opposite face then follows from the constraint relation.

The predictions of the first order mechanism can be fitted to the appropriate relaxation functions. For example, in the plot below results generated for a model of nitrogen on a methyl functionalized silica surface at 77 K have been used. The plot shows the function $R_{MM}(t)/\langle X_M \rangle$ against time. This corresponds to a situation where all molecules are considered in the gas phase at *t*=0. The relaxation to equilibrium is relatively rapid and can be fitted to the form

$$X_M(t) = \exp(-kt) + \langle X_M \rangle\left(1 - \exp(-kt)\right)$$

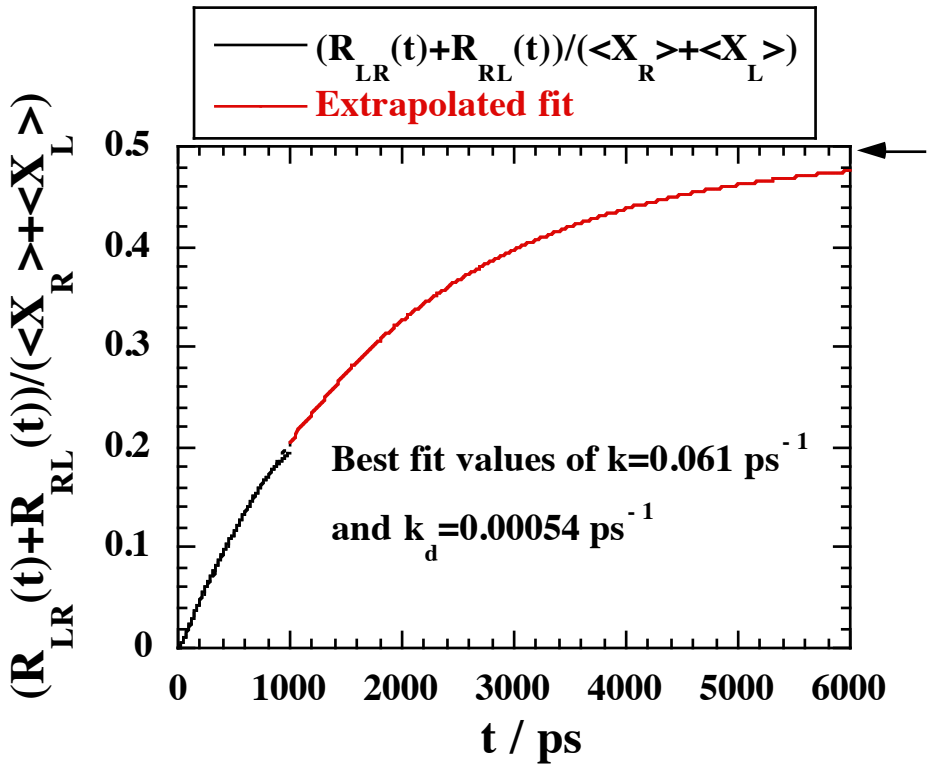
which is appropriate for the initial conditions.



The best fit k obtained of 0.0577 ps^{-1} and the mean fraction of molecules in the gas phase of 0.0111 give an estimation of k_d of 0.0577*0.0111=0.00064 ps^{-1} and thus a k_a of 0.0285 ps^{-1}. As this relaxation is dominated by the fast rate of adsorption it is less adapted for obtaining estimates of the slower desorption reaction. An alternative experiment is to consider the situation where we initially consider just the molecules adsorbed on one surface and then measure the fraction of these molecules which are at some later time on the opposite surface. This information is contained in the equivalent $R_{LR}(t)$ and $R_{RL}(t)$ functions. For example, considering all molecules to be on the right surface at t=0 the appropriate solution with which to compare the $R_{RL}(t)$ function is

$$X_L(t) = \frac{1}{2}\langle X_M \rangle \left(\exp(-kt) - \exp(-k_d t)\right) + \langle X_L \rangle \left(1 - \exp(-k_d t)\right)$$

Of course, a completely analogous solution can be used for the case where all molecules are on the left surface at t=0. In practice a symmetrized relaxation function can be used to improve statistics. The symmetrized function $\frac{\left(R_{LR}(t) + R_{RL}(t)\right)/2}{\left(\langle X_R \rangle + \langle X_L \rangle\right)/2} = \left(R_{LR}(t) + R_{RL}(t)\right) / \left(\langle X_R \rangle + \langle X_L \rangle\right)$ corresponds to the solution given above and this has been plotted in the figure below



The best fit values to the data for symmetrized function up to 1000 ps have been used to extrapolate the curve to longer times (the arrow indicates the limiting long time value of $\left(\left\langle X_R \right\rangle + \left\langle X_L \right\rangle\right)/2$. This gives an estimation of the characteristic time associated with exchange of gas molecules between the two surfaces. The best fit values for the rate constants are reasonably consistent with those obtained from the analysis of the much faster relaxation of the $R_{MM}(t)$ function. The differences give an indication of the errors involved in the procedure.

If requested by the user, the program also outputs a file called `CROSSING.LIST` which list gas molecules having crossed from the left to right or from the right to the left. The same information is also written out to two separate files called `LtoR_CROSS.LIST` and `RtoL_CROSS.LIST` in a format that can be read by graphing programs.

nano_rfunc (v1.4 - 18/09/2013)

This program performs the same analysis as *gmq_rfunc* but with spatial resolution into slabs or shells. As *nano_rfunc* needs to know the positions of the atoms as well as the states of the torsional angles it uses the `config.bin` file rather than the `INDEX` file. The recommended assignment of torsion angles to shells or slabs is that based uniquely on the position of the centre of mass of the four atoms involved in the torsion at the time origin, t_0. Although this means that at some later time an angle may belong to a different shell it does avoid the bias that would result from selecting only angles that remain in a particular shell. The user, however, is given a choice as to whether to include just those angles that remain in the same shell or slab but is warned of the consequences. The results for each slab or shell are written to separate output files and the normalizing factors, i.e. the numbers of angles contributing to the averages at each time difference, are also written to a file.

nano_rgete (v1.6 - 21/11/2018)

This program performs the same analysis as *gmq_rgete* but with spatial resolution into slabs or shells. In addition to the mean square radius of gyration and mean square end-to-end distance, the program also prints out the contributions to these quantities in the direction perpendicular to the "interface". In the isotropic case the contributions perpendicular to the interface should be 1/3 of the total. In general chains will be affected by the presence of an interface so isotropy is no longer guaranteed. The contributions to $\langle R^2 \rangle$ or $\langle S^2 \rangle$ parallel to the interface can be obtained by subtraction of the perpendicular contributions from the relevant totals. The resulting histograms are all written to the file `NANO.RG_ETE`.

nano_trans (v1.8 - 18/09/2013)

This program calculates the mean percentages of *trans* conformers for user-selected torsion types with spatial resolution into slabs or shells. As ever, a torsion angle is, by definition, *trans* if it is between ±60°. For non-threefold potentials this has little meaning. The average histogram for the user-selected time period is written to the file `TRANS.OUT`. The histograms for each configuration are written as a function of time to the file `TRANS.TIME`.

nano_void (v1.3 - 18/09/2013)

This program performs a spatially resolved analysis of the void space in a system in a similar way than *gmq_void*. The resolution can either be:- (1) Spherical shells or (2) Slabs. Spherical geometry is intended for use with systems which contain a single nanoparticle. The distance is then obtained relative to the COM of the nanoparticle. The slab geometry is intended for use with membrane systems where there is a flat surface present. In this case the reference vector is chosen in the direction perpendicular to the flat surface, either the x, y or z axis, and the spatial resolution is the distance from the plane containing the COM of the membrane. The user is asked to define the atom types that make up the nanoparticle or membrane. This information is stored in the file `nano.types`. The spatially unresolved data for each configuration selected is written to the file `NANO_VOID.OUT`. The spatially resolved data is written to the file `NANO_VOID_DIST.OUT`. In the case of slab geometry a symmetrized data file is written to `NANO_VOID_DIST_SYM.OUT`.

nano_widom (v7.1 - 06/08/2020) & nano_tpi (v7.1 - 06/11/2020)

These two programs were principally designed to analyse configurations stored in `config.bin` using the *test particle insertion* (TPI) technique in order to estimate the solubility of, for example, a penetrant gas molecule into a polymer membrane. *nano_widom* is the scalar version of the code and *nano_tpi* is the parallel version. In addition to doing TPI analysis both codes can also use the TPI method to progressively load a `CONFIG.NEW` file with a user-specified number of penetrant molecules. This loading option is a preferred alternative to the method that is implemented in *gmq_addsol* as it guarantees that penetrants are placed at favourable low-energy sites and thus avoids the need for an energy minimization step. In both cases it is preferable to use *nano_tpi* on the largest number of processors

available as this will improve the statistics as TPI calculations are generally difficult to converge. Furthermore both codes contain an implementation [46] of a Grand Canonical Monte Carlo technique (GCMC) [1, 3] which can be used to predict the equilibrium loading, for example, of a system when it is put into contact with an arbitrary *n*-component gas mixture.

When *nano_widom* or *nano_tpi* are invoked the user has a choice of the following options:-

```
Which type of run do you want to do?
 0=Analyse stored configs. in config.bin
 1=Load CONFIG.NEW with test particles
 2=Use GCMC method on stored configs. in config.bin
```

Whatever the choice of mode the replies to the interactive questions are written to the `INPUT.WIDOM` file. This file can be useful for creating a script to be used in batch mode though *care should be taken to use the same code in both cases* as there can be slight differences in the sequence of questions/replies between the scalar and parallel versions.

Option 0: Test Particle Insertion analysis

The theory behind the TPI technique is covered in detail in standard texts [2, 3]. The method involves performing repeated random test insertions of the test molecule into a configuration of an existing system. The molecule is never actually accepted into the system and does not interfere with the dynamics, and in practice the tests are made on previously stored configurations. For each random insertion the potential energy of interaction ($\Delta\Phi$) of the test molecule with atoms in the existing system is calculated and the **excess** chemical potential is approximated from the following equation in the NVT ensemble

$$\mu_{ex} \approx -kT \ln\left\langle \exp\left(\frac{-\Delta\Phi}{kT} \right)\right\rangle = -kT \ln\left\langle p_i \right\rangle$$

where p_i is effectively the probability of insertion. The excess chemical potential is related to the total chemical potential via

$$\mu_{ex} = \mu - \mu_{ig}$$

where μ_{ig} is the ideal gas contribution. According to standard texts [2, 3], this is given, in the case of a monoatomic test particle, by

$$\mu_{ig} = -kT \ln\left(\frac{V}{N\Lambda^3} \right)$$

where Λ is the momentum partition function, a.k.a. the thermal de Broglie wavelength, defined as

$$\Lambda = \left(\frac{h^2}{2\pi mkT} \right)^{\frac{1}{2}}$$

and h is Planck's constant ($6.626*10^{-34}$ Js). For polyatomic molecules Ben-Naim [54] gives the more general expression

$$\mu_{ig} = -kT \ln\left(\frac{Vq}{N\Lambda^3} \right)$$

where q is the internal partition function, which includes, notably, the rotational and vibrational partition functions of a single molecule.

Furthermore the solubility, S, and the probability of insertion, p_i, are also linked

$$S = \exp\left(\frac{-\mu_{ex}}{kT} \right) \approx \left\langle \exp\left(\frac{-\Delta\Phi}{kT} \right) \right\rangle \approx p_i$$

Thus Widom's technique can be used to estimate the solubility of one substance in another.

To run *nano_widom* or *nano_tpi* in TPI mode you will need the usual input files for *gmq* plus a `config.bin/config.count` combination. The test particle is read from a second `CONFIG.NEW`-style file called `CONFIG.WIDOM`. For TPI analysis it must contain just one molecule in a "sensible" conformation, e.g. if it's a diatomic molecule the bond length should be about the equilibrium value. *N.B.* No sampling of the internal degrees are performed so the results are not strictly valid other than for rigid probe molecules. The `PARAM` file should contain the parameters for the particle to be inserted so the atom types defined in `CONFIG.WIDOM` should be coherent with the `PARAM` file. It is recommended that the Widom probe have atom types different to those in the system being tested but this is not obligatory.

The TPI probe molecule can contain charges on the atoms but the probe molecule *must* be charge neutral, i.e. no ions. Charges must be assigned via the `PARAM` file.

In the case of a charge neutral probe molecule there is a subtle point to consider concerning the calculation of the energy difference. In principle the energy difference we require is just the interaction of the test particle with the real particles already present. If the Ewald summation is used to obtain the Coulombic part of this energy difference using $\Delta\Phi_c=\Phi_c(N+1)-\Phi_c(N)$ there is a problem in that intrinsically the Ewald sum contains contributions from probe molecule interactions with images of itself and interactions of real particles with images of the probe molecule. The interaction energy of the probe with images of itself are removed by performing the Ewald sum on the probe molecule alone in the MD box. However, this does not remove interactions of real particles with images of the probe so the energy difference does not strictly correspond to that of the interaction of one probe with an infinite array of real particles. As the probe molecules are charge neutral and likely to be small this term is probably not of any importance in practice.

Once invoked in TPI mode and the range of stored configurations to analyse has been determined, *nano_widom/nano_tpi* will propose three modes of operation:-

```
Which type of Test Particle Insertion do you want?
0=Normal random TPI, i.e. Widom
1=Test mode for setting up excluded volume map
2=TPI using an excluded volume map
WARNING! It is imperative that you use the test mode
before using the excluded volume map!
Input required TPI mode:- 0, 1 or 2
```

Mode 0 performs a standard test where insertions are made completely at random into the MD box. The user can define the number of trial insertions to be made on a per $Å^3$ basis; to aid the user in the choice of the density of trial insertions, the volume of the MD box in the `CONFIG.NEW` file is printed out. Mode 0 also allows the user to resolve, or not, the insertion probability spatially in either slabs (e.g. membranes) or spherical shells (e.g. around nanoparticles). The resulting average insertion probabilities are written as a function of the slab/shell to the file `PIDIST.OUT`.

The conventional method is somewhat inefficient as a large proportion of the random insertions will result in large overlap energies with the existing atoms thus correspondingly small values of the insertion probability. For this reason a second mode (Mode 2) is available which uses a variant of the Excluded Volume Map Sampling (EVMS) approach [55-57]. In the EVMS approach implemented here, the MD box is first divided up into a number of subcells on the basis of a user-defined subcell width, called DGRID. As the MD box is not necessarily cubic and the number of subcells in each direction has to be an integer the subcells are constructed internally on the basis of DGRID. In a second step each atom in the system is visited and all subcells that have their centres within a distance of WCUT-DIAGMX/2 are mapped out, where WCUT is a user-defined distance parameter, which reflects the "excluded" volume of an atom, and DIAGMX is the largest diagonal of a subcell. Using this criterion ensures that all points within the subcell must be within WCUT of the atom being considered. Particle insertions are then made by randomly choosing one of the "mapped-in" subcells and then randomly placing a probe molecule within the chosen subcell. In this way a large amount of the space can be pre-eliminated whilst preserving the advantages of a Monte Carlo sampling of space. The actual amount of space mapped out depends on WCUT and the DGRID parameter. A finer grid will pre-eliminate more space at the cost of a larger overhead in terms of memory and CPU time required to set up the map for each configuration. A WCUT which is too large will map out all subcells and in this case the program prints a fatal error and stops.

As the EVMS method performs a biased sampling of the space the results have to be corrected for the bias. This is done by simply assuming the mapped-out cells contribute zero to the average of $\exp(-\Delta\Phi/kT)$ in which case the true average is approximated by multiplying by the ratio of the number of mapped-in cells to the total number of cells.

The difficulty with this EVMS method is knowing *a priori* what an appropriate value of WCUT should be for a particular system. For this reason Mode 1 ***must*** be used before invoking Mode 2. In Mode 1 the map created using the user-defined value of WCUT is first put to the test. Instead of trial insertions being made into the "mapped-in" subcells, trial insertions are made into the "mapped-out" cells, i.e. those nominally occupied by the particles present in the system. If the value of WCUT is appropriate then insertions into mapped-out cells should give large positive (unfavourable) energy changes, in this case *nano_widom/nano_tpi* exits after testing the first configuration chosen with a message confirming a

successful completion. However, should a favourable insertion be made ($\Delta\Phi \leq 0$) then the program exits immediately indicating that the test has failed and to try with a smaller value of WCUT. Mode 1 can thus be used to optimize WCUT; it should be as large as possible, so as to pre-eliminate as much space as possible, but not be too large as to allow favourable insertions in the mapped-out subcells.

It is preferable to spend some time optimizing WCUT and DGRID for the system in question before embarking on what can be quite expensive calculations. At best an EVMS method can improve efficiency by a factor of the total volume divided by the mapped-in volume, e.g. a factor of 20 if 95% of the space is considered occupied. In practice the overheads associated with pre-eliminating 95% of the space will probably not be negligible. Claims to factors of improvement of EVMS over conventional Widom greater than 20 [56] should be treated with some skepticism.

In the case of EVMS-TPI, once the mode (1 or 2) has been selected the user can set the value of DGRID and then choose the number of attempted trial insertions per $Å^3$. The value of WCUT has then to be set. In the case of EVMS-TPI, no spatial resolution is currently implemented. This might be possible but would require some thought as how to have a coherence between the grid used for the EVMS and the histograms for the spatial resolution.

The main results from *nano_widom/nano_tpi* are printed in the `OUTPUT.WIDOM` file and in the file `STO.WIDOM`. Fourteen columns are given:-

1. TIME	the time as read from `config.bin` for this config.
2. UNWEIPE	the unweighted average energy of insertion $<\Delta\Phi>$
3. PELRC	the potential energy long range correction
4. V*EXP(-U/kT)*U	the product of the volume, insertion energy and the Boltzmann factor, $<V*\Delta\Phi*\exp(-\Delta\Phi/kT)>$
5. Exp(-U/kT)	the Boltzmann factor of the insertion energy, $<\exp(-\Delta\Phi/kT)>$
6. VOLUME	the box volume in m^3.
7. DENSITY	the density in kg/m^3.
8. TEMP	the temperature read from `config.bin`
9. V*EXP(-U/kT)	the product of the volume and the Boltzmann factor, $<V*\exp(-\Delta\Phi/kT)>$
10. EXCPOT	the molar excess chemical potential $-RT*\log<\exp(-\Delta\Phi/kT)>$
11. H_BAR-1	Henry's solubility coefficient in cm^3 (STP of "gas") per cm^3 (of material) per bar, i.e. $<\exp(-\Delta\Phi/kT)>*273.15/(T*1.01325)$
12. H_CMHG-1	Henry's solubility coefficient in cm^3 (STP of "gas") per cm^3 (of material) per cm Hg, i.e. $<\exp(-\Delta\Phi/kT)>*273.15/(T*76.0)$
13. UNBIAS	the unbiaising factor, i.e. the fraction of mapped-in sub-cells in the EVMS method.
14. UNBIAS*V	the unbiaising factor multiplied by the volume in m^3.

All energies are multiplied by Avogadro's number so are in "J/mole of inserted molecules", i.e. NOT as in *gmq*.

Note that the corresponding average that should be calculated for "NPT" ensembles is,

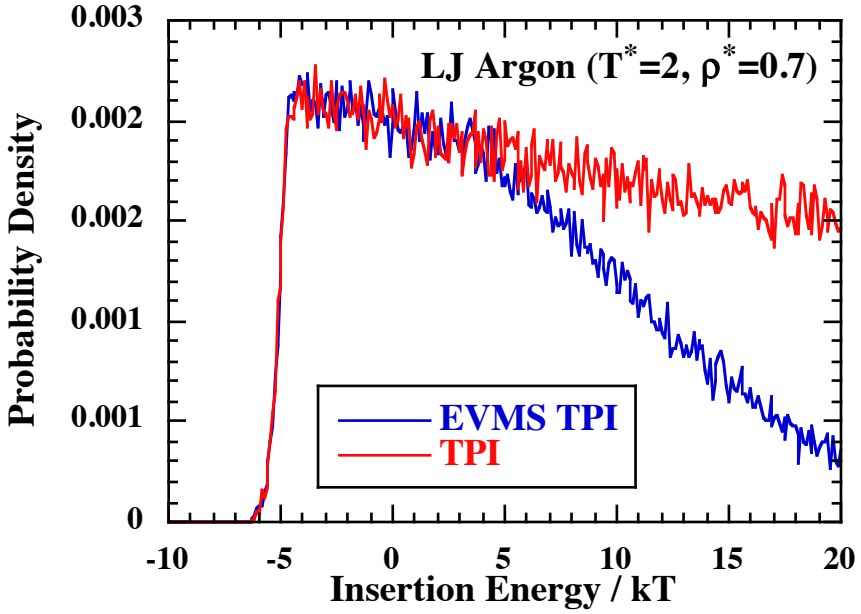
$$\langle p_i \rangle = \frac{\left\langle V \exp\left(\frac{-\Delta\Phi}{kT} \right) \right\rangle}{\langle V \rangle}$$

where V is the volume of the system, i.e. the overall average of column 9 divided by the overall average of column 6. You would then have to take the natural log of this and multiply by $-RT$ to compare with column 10. Unless volume fluctuations are important, there should not be much difference.

Note also that, in the case of the NVT and NPT ensembles, the average insertion energy can be obtained from the quotient of the average values of Columns 4 & 9

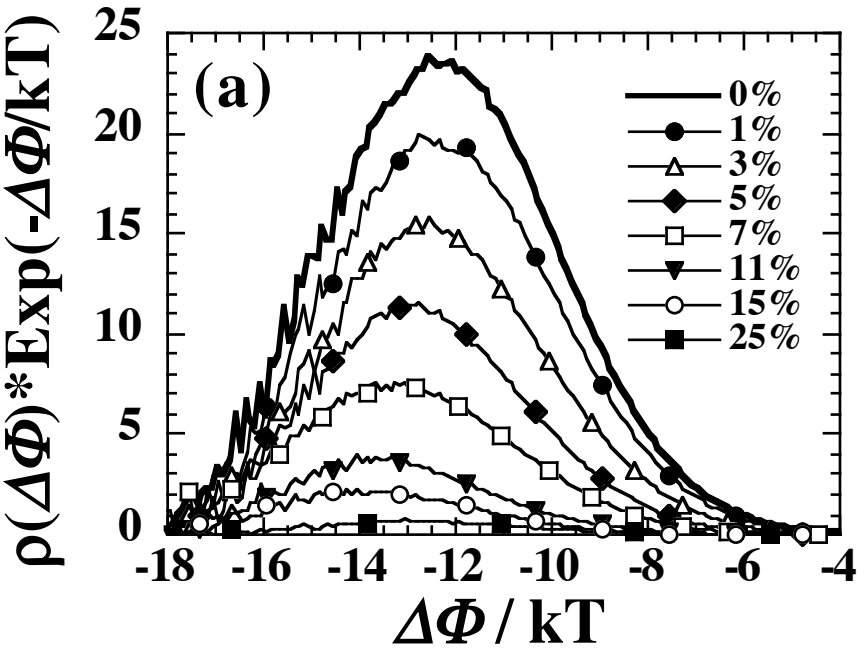
$$\langle \Delta\Phi \rangle = \frac{\left\langle V \cdot \Delta\Phi \cdot \exp\left(\frac{-\Delta\Phi}{kT} \right) \right\rangle}{\left\langle V \exp\left(\frac{-\Delta\Phi}{kT} \right) \right\rangle}$$

The code also produces several other output files. The file `PEDIST.WIDOM` contains the distribution of insertion energies. As insertion energies can be extremely high there is an upper bound to the distribution, set currently to 25 kT. Note that all insertion energies greater than this upper limit are accumulated in to the last bin. This point should be masked from any plot. As the EVMS method (Mode 2) biases the distribution of energies by pre-eliminating the higher energy ones, the normalized probability densities can differ substantially between Mode 0 and Mode 2. This bias can be partly removed by taking into account the bias in the EVMS method. This should lead to a good agreement between Mode 0 and Mode 2 distributions in the energy range where exp(-ΔΦ/kT) is significant. In the `PEDIST.WIDOM` file both the usual probability density and the unbiased distributions are printed out if IGEOM=0. In the case of Mode 0 these are the same. In the case of Mode 2 the "unbiased" distribution is no longer normalized to one but to the unbiaising factor. A comparison between the distributions obtained using Mode 0 (TPI) and Mode 2 (EVMS TPI) is shown in the following graph obtained for a system of 2048 particles of LJ Argon:



It is clear to see that EVMS TPI eliminates insertions in regions where the energy is high.

Also written to `PEDIST.WIDOM` is the product of the unbiased distribution and the corresponding Boltzmann factor, i.e. $\rho_w(\Delta\phi) = \rho(\Delta\phi)\exp\left(-\frac{\Delta\phi}{kT}\right)$. This Boltzmann factor weighted distribution gives a better representation of the energy of likely sites of adsorption and there is a direct connection to the solubility since $\left\langle \exp\left(-\frac{\Delta\phi}{kT}\right)\right\rangle = \int_{-\infty}^{+\infty} \rho(\Delta\phi)\exp\left(-\frac{\Delta\phi}{kT}\right) d\Delta\phi$. Examples of these weighted probability density functions are shown in the Figure below for the case of the 6FDA-6FpDA polyimide at a range of CO_2 concentrations [45]. Note that the histograms are accumulated with a bin width ($\delta\Delta\phi$) of 0.1 kT and the resulting probability densities thus have units of $(kT)^{-1}$.



The smoothness of the Boltzmann factor weighted distributions depends very much on the quality of the unweighted distributions at low energies. Poorly converged unweighted distributions will inevitably give rather ragged weighted distributions. This is a good test of the reliability of the TPI method so this weighted distribution function should always be inspected. The `PEDIST.WIDOM` file can also be used to obtain an estimate of the significant space available to the probe (see description of the *get_aspace.f* code that follows).

In the case where Mode 0 is selected and a spatial resolution is required (IGEOM=1 or 2) then the insertion energy is resolved into the shells or slabs as for the `PIDIST.WIDOM` file. In this case the distributions for each shell or slab represent just the *contribution* to the total probability density.

The file `WIDOM_RDF.OUT` contains the radial distribution functions, evaluated by taking into account the respective Exp(-ΔΦ/kT) factors of the trial insertions, for the atoms of the virtual probe molecule with the atoms in the system. A column is present in the file for all permutations of interactions between types of probe atoms and the atom types present in the system. Effectively, even if the atom types of probe atoms and system atoms are the same they are treated as if they are different. For this reason no symmetrization is performed either. A second file, `WIDOM_NOFRIJ.OUT`, contains the corresponding $n_{ij}(r)$ functions (see description of *gmq_rdf*).

$$n_{ij}(r) = \frac{n_j}{V} \int_0^\infty 4\pi r^2 g_{ij}(r)\ dr$$

Unlike *gmq_rdf*, only the $n_{ij}(r)$ are written out for all permutations where *i* is an atom type in the virtual probe molecule and *j* is an atom type in the system. In this sense the average number of system atoms in a sphere of radius *r* around a probe atom is well defined, and should be fairly invariant to system size,

whereas the corresponding $n_{ji}(r)$ functions are somewhat arbitrary due to the fact that only one probe molecule is inserted at a time, i.e. the number density of probe atoms is a function of box size. If required, the $n_{ji}(r)$ functions can always be generated from $n_{ij}(r)$ using the relation

$$n_{ji}(r) = \frac{n_i}{V} \int_0^\infty 4\pi r^2 g_{ij}(r)\ dr = \frac{n_i}{V} \frac{V}{n_j} n_{ij}(r) = \frac{n_i}{n_j} n_{ij}(r)$$

within the approximation that the rdfs would be insensitive to the number of probe molecules inserted into the system; this may or may not be a good approximation depending on the linearity of the solubility of the probe in the system.

The output file `RUNAVG` contains a running average of the exp(-ΔΦ/kT) factor for the last configuration analysed. This is intended to give an idea of the number of trials required to converge the results for one configuration.

Finally, in TPI analysis mode, the coordinates and insertion energies of the most energetically favourable inserted probes can be stored should the user so wish. Since the addition of the loading and GCMC options the storing of low energy insertions is deprecated and it no longer happens by default. To signal to the nano_widom/nano_tpi codes that low energy insertions are to be stored an empty file called `LOW` must exist in the directory of execution. Should this be the case then the coordinates of the 1000 lowest energy inserted molecules are written to the file `config.bin_low` which is in standard `config.bin` format. The corresponding `config.count_low` is also written as well as a file called `CONFIG.LOW` which is in standard `CONFIG.NEW` format except that the molecules are ranked in terms of their insertion energy; molecule 1 being the lowest energy probe inserted. The actual insertion energy, in units of *kT*, is output at the place of the *x*-velocity. The user can thus obtain a visual representation of where the most favourable insertions take place and how this evolves in time by subsequently using *gmq2pdb* and *VMD*.

Option 1: Loading `CONFIG.NEW` using TPI

In this option the CONFIG.NEW file is progressively loaded with a user-defined number of test particles. This mode can be useful to load a sample with a specific number of penetrants, for example. After specifying the number of molecules to load the user again has the choice between standard TPI and EVMS TPI etc. The code then performs TPI for the user-specified number of insertions per Å^3 and the particle with the lowest energy is actually inserted into the configuration and the process is then repeated until the desired number is attained. As the inserted particles are taken into consideration in subsequent TPI steps, the resulting configuration, written to `CONFIG.NEW_LOADED`, should be free from high energy overlaps. The results of the TPI analyses are again written to the `STO.WIDOM` file except in this case the first column gives the number of the particle being loaded, rather than the time of the analysed configuration. No rdf analysis is performed but a `PEDIST.WIDOM` is produced.

Option 2: GCMC mode

In GCMC mode a range of configurations stored in the `config.bin` file are used to obtain the average uptake of molecules in the system when put into contact with either a gas, or mixture of gases. If we assume that there is a gas phase system, of fixed composition, in equilibrium with the system in question then we know for each gas the chemical potentials are the same in both phases. For rigid penetrants it is known that this means that ratios of the solubilities in the two phases is the same as the ratio of their concentrations in the two phases [54]. The configurations read-in are static so there are no changes in positions of the original atoms although any penetrants that are already present in the configuration can be "exchanged" with the gas phase. It is also important to note that the volume of the original system is unchanged, i.e. no swelling can occur, in GCMC. At each GCMC step attempts are made to insert or delete penetrant gas molecules from the system with probabilities in accord with the condition of equality of chemical potentials in the two phases. In practice, in the method used here, there is no need for an explicit simulation of the gas phase. However, the average concentrations and solubilities in the gas phase of constant composition are required input so require pre-calculating. This approach thus mimicks the equilibrum between a membrane and a gas maintained at constant composition, e.g. by a flowing feed.

When *nano_widom/nano_tpi* is executed in GCMC mode the user first chooses a range of configurations to use. Averaging over more configurations leads to better averages for the uptake but require proportionately more computing resources so a compromise has to be made. The user then has to define the number of GCMC moves to attempt in the run. Again this choice depends on the resources available but typically several million steps are required. The GCMC simulations can be restarted though so it is possible to continue them until convergence is achieved. The user also has the choice of using standard TPI for the insertions or EVMS TPI. For dense systems it is better to use EVMS TPI but care has to be taken choosing the WCUT parameter if the gas phase system contains a mixture of gases. In this case WCUT should be the lowest value amongst those optimised for each individual gas.

To input the data for the gas phase the user has two possibilities. Either the data for a specific state point is read interactively, in which case the user enters a 3-character name for each gas, its gas phase concentration (in molecules/nm^3) and its (dimensionless) solubility. Alternatively the data for the gas phase can be read from a file called `GAS_PHASE.DATA` if the gas phase data at a fixed temperature is known as a function of pressure. This assumes that MD simulations have been performed already of the gas phase at a range of densities. An example `GAS_PHASE.DATA` file is shown below:-

```
2:1 gas phase mixture of CH4/N2
2                       # No. of components in gas phase
308.0                   # Temp. in K
1.000                   # Req. Pressure in bar
CH4                     # Label of Type 1 gas
N2                      # Label of Type 2 gas
512                     # No. of CH4 in gas phase
256                     # No. of N2 in gas phase
0.0016923 -2.9039E-6    # Coeffs. for S(p) of CH4 in mixed gas phase
0.00066902 -3.6874E-6   # Coeffs. for S(p) of N2 in mixed gas phase
0.00075491 -3.1332E-6   # Coeffs. for 1/V(p) of mixed gas phase
```

The first line is just an arbitrary title describing the contents of the file. There then follows the number of components in the gas phase, the temperature, the required pressure, a label for each gas and the numbers of each gas in the MD simulations of the gas phase. There then follows the coefficients of the fits of the solubility of each gas in the gas phase to a function of the form:

$$S(p) = 1 + \alpha p + \beta p^2 + \gamma p^3$$

the cubic term is optional and can be omitted if not required for the fit. Finally the coefficients of the fit of the reciprocal volume to the following virial form are required:

$$\frac{1}{V(p)} = \frac{10^5}{Nk_BT}\left(p + ap^2 + bp^3\right)$$

where N is the ***total*** number of molecules in the gas phase (N=512+256=768 in the example given above) and the factor of 10^5 takes into account that the pressure, p, is given in bar (1 bar=10^5 Pa). Note that both fits ensure that the correct behaviour, i.e. that of an ideal gas, is obtained at low pressures. The advantage of reading the data from the `GAS_PHASE.DATA` file is that if a range of pressures are being studied just the required pressure has to be changed. The code determines the gas phase solubilities and reciprocal volume using this pressure and functional forms and the coefficients that have been read-in.

Two adaptations give extra flexibility to the GCMC technique. It can be instructive to determine the uptake in the absence of one or more of the components. By specifying a *negative* number for a component in the gas phase the solubility of this species is artificially set to zero in the matrix phase, so any sorbed molecules are quickly removed, and no insertions from the gas phase are possible. The gas phase concentration and solubilities of the other components are unaffected so they have the same driving force for entering the matrix. This can be used to determine the maximum capacity of a set of configurations of one, or more, species in the absence of one, or more, of the others.

A second adaptation allows a gas species to be only partially present. By specifying a second number in the line defining the number of molecules in the original gas phase simulation, a reduced concentration of the species is used in the GCMC simulation, although the solubility is kept as defined in the original gas phase. The concentration of all other species are kept the same so long as the original number in the gas phase is entered correctly first on the line. This allows the uptake to be determined as the concentration of another species is progressively introduced.

The main results of the GCMC simulation are output to two files. `ANUMPEN.OUT` contains the average numbers of penetrants in the system as a function of the number of GCMC moves; the averages

being over all the configurations used. Ideally these values should reach a plateau value once equilibrium is achieved. The details for each configuration are sequentially written to the `GCMC.OUT` by *nano_widom* for all the configurations analysed whereas in *nano_tpi* this data is written out in parallel to separate files of the form `GCMC.OUT_*` where the asterisk is the number of the processor on which the configuration is analysed.

The second file, `ASOLPEN.OUT`, contains data for each configuration analysed. The first column contains the time of each selected configuration and then there follows four columns for each penetrant containing the number of trial insertions into the system, the average solubility obtained, the ratio of concentrations and ratio of solubilities. When converged these latter two columns should be the same on average.

To facilitate the restarting of a GCMC simulation the final configurations are written to files prefixed `CONFIG.GCMC_*` where the asterisk represents the time in ps at which the configuration was stored in `config.bin`. Should the time between the storing of the configurations be less than 1 ps then the batch number is used as a suffix instead. Subsequent runs read these files and restart the simulations from the number of penetrants contained in them. The `ANUMPEN.OUT` and `ASOLPEN.OUT` files are also appended in subsequent runs. The `GCMC.OUT_*` files are appended by *nano_tpi* but the `GCMC.OUT` file of *nano_widom* is overwritten.

As *nano_widom* is a scalar program and thus analyses each stored configuration sequentially performing GCMC with it can be very time consuming. Since the GCMC option is now included in the parallel version, **it is highly recommended** to use *nano_tpi*. If using the parallel code is not an option, then an alternative approach is possible using the *pbsdsh* command (if available) to run separate scalar jobs on a number of processors. This effectively means that a range of configurations can be analysed by different processors in parallel thus saving time. In practice there are overheads for using *pbsdsh* and these become severe if all the processors of a machine are used. However, it is possible to spread a *pbsdsh* job over more than one node and this reduces the overheads. **For details of this approach contact DB**.

A typical batch job script file (called *run_wjob*) follows for the case where *nano_tpi* is used:

```
#!/bin/tcsh
#
#PBS -N 6m_GCMC_CO2_60bIt27
#PBS -j oe
#PBS -m a
#PBS -l nodes=1:ppn=20
#PBS -l walltime=94:00:00
# This finds out the number of nodes we have asked for
set NPROC=`wc -l $PBS_NODEFILE | cut -f1 -d" "`
echo "Total CPU count = $NPROC"

echo ""
# Change to dir where job was submitted
cd $PBS_O_WORKDIR
echo $PWD
echo ""
echo "--------------- RUN --------------------"
mpirun -np $NPROC  --mca btl sm,self nano_tpi<<!
2          # ICMODE (0=TPI, 1=Load, 2=GCMC)
n          # Print all Batch headers? (y or n)
     181   # Start Batch No. for analysis
     200   # End Batch No. for analysis
 2000000   # NTRY : No. of MC steps to do
2          # IWMODE (0=Widom, 1=Test EVMS, 2=Use EVMS)
n          # Change subcell width? (y or n)
y          # Change excluded volume distance? (y or n)
 2.200     # WCUT : Excluded volume distance in Angst.
!
setenv  STATUS $status
echo ""
echo "------------ END ----------------------"
date

ls -al

echo ""
echo "==> Resulting STATUS = $STATUS"
echo ""
if ( ! $STATUS ) then
 RESUB_GCMC_MPI run_wjob 20000000
endif
```

The number of processes to run has to be the same as the number of configs. to be analysed. In the case above, 20 processors are assigned for the 20 configs. to be analysed. The batch job script file *run_wjob* also invokes the *RESUB_GCMC_MPI* script, should repeat runs be required. In the above example each individual job does 2 million MC trials so 10 jobs are required to do the required total of 20 million MC trials. The *plot_gcmc* script can be used to plot the data in the `ANUMPEN.OUT` and `GCMC.OUT_*` files. The *prob_gcmc* code can be used to generate a probability density of the number of penetrants in the systems from the `GCMC.OUT_*` files. Further scripts are available to automate the setting up of a GCMC analysis directory. Other scripts are available that determine the number of penetrants predicted by the GCMC method at the specific pressure and then generate a directory for the next MD step in the case of using the iterative GCMC-MD method for the construction of sorption isotherms. As these scripts are rather specific to the systems being studied the user should contact DB.

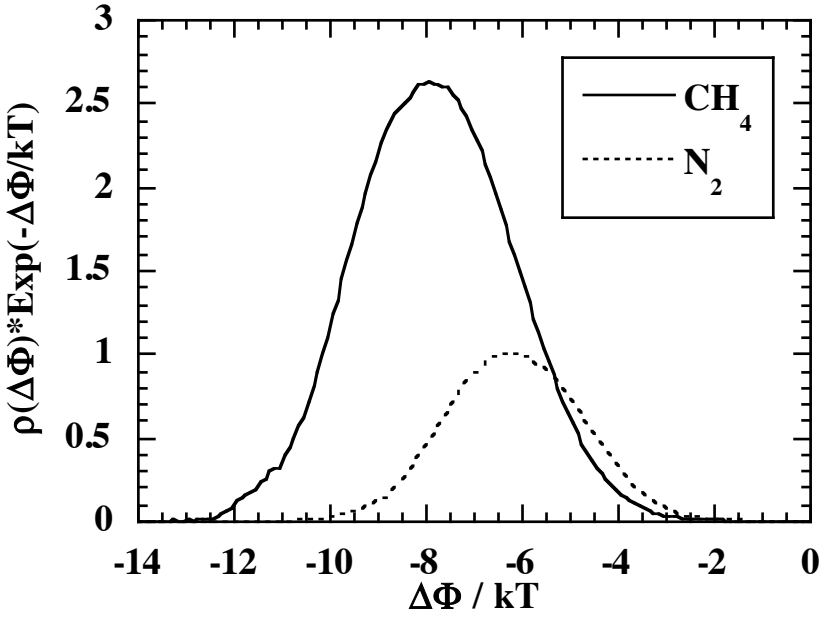
get_aspace (v1.3 - 13/12/2017)

The program *get_aspace* uses the `PEDIST.WIDOM` file produced by *nano_widom/nano_tpi* to obtain an estimate of the significant space available to a probe molecule based on energetic rather than

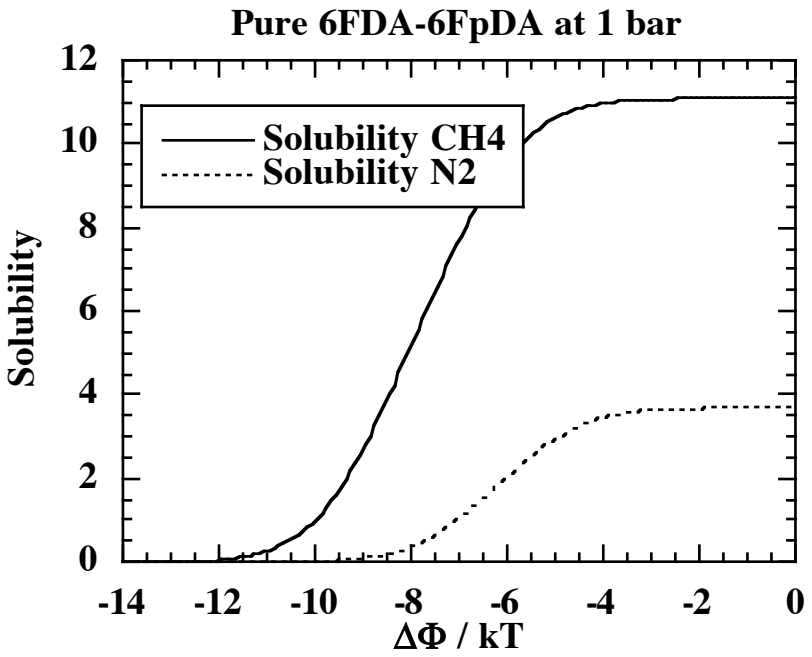
geometric criteria. As explained in the description of *nano_widom/nano_tpi*, the probability densities for the insertion energy, $\rho(\Delta\Phi)$, can be related to the solubility in the case where volume fluctuations are negligible through the following equations

$$S = \frac{\left\langle V \exp\left(\frac{-\Delta\Phi}{kT}\right)\right\rangle}{\left\langle V\right\rangle} \approx \left\langle \exp\left(-\frac{\Delta\phi}{kT}\right)\right\rangle = \int_{-\infty}^{+\infty} \rho\left(\Delta\phi\right)\exp\left(-\frac{\Delta\phi}{kT}\right) d\Delta\phi\,.$$

The Boltzmann factor weighted distribution gives a representation of the energy of likely sites of adsorption, as can be seen below for the case of nitrogen and methane in a pure sample of the 6FDA-6FpDA polyimide.



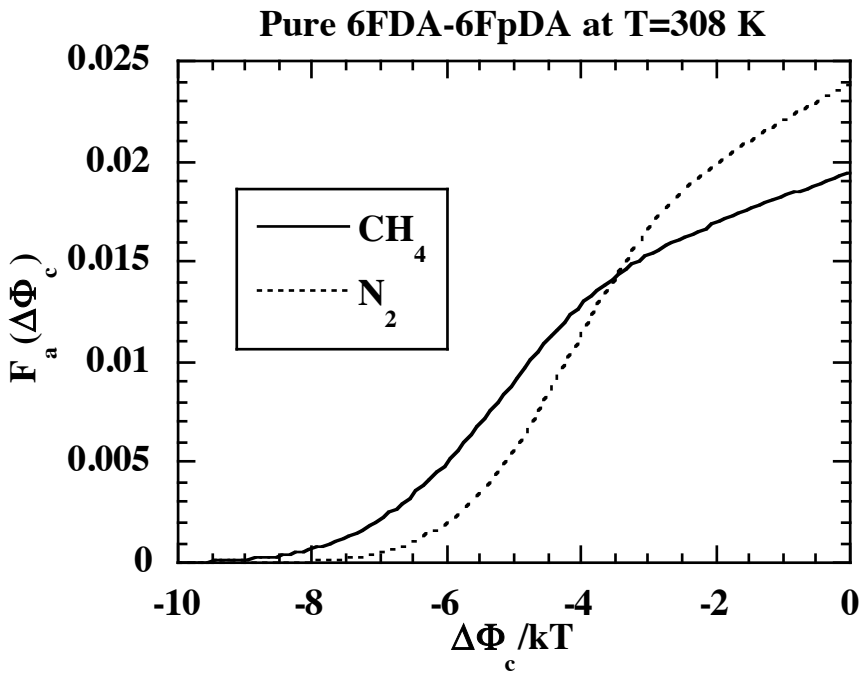
The associated integrals of these functions reach a plateau relatively quickly as the $exp(-\Delta\Phi/kT)$ weighting factor rapidly diminishes as the insertion energy rises. It is for this reason that lower insertion energies are weighted more than higher energies. In the case of N_2 and CH_4 in 6FDA-6FpDA the integrals are converged to more than 99% by $\Delta\Phi$=-2kT.



We can thus say that those insertions corresponding to energies lower than -2kT account for almost all the solubility. So if we integrate instead the probability density distribution functions up to this energy then this will give us the fraction of insertions having an insertion energy that contributes to the solubility and hence the fraction of the significant volume available to the centres of the molecules.

$$F_a(\Delta\phi_c) = \int_{-\infty}^{\Delta\phi_c} \rho\left(\Delta\phi\right)\, d\Delta\phi$$

This is effectively the fraction of random insertions into the system that would lead to energies that are significant with respect to the solubility. The corresponding integrals are shown below.



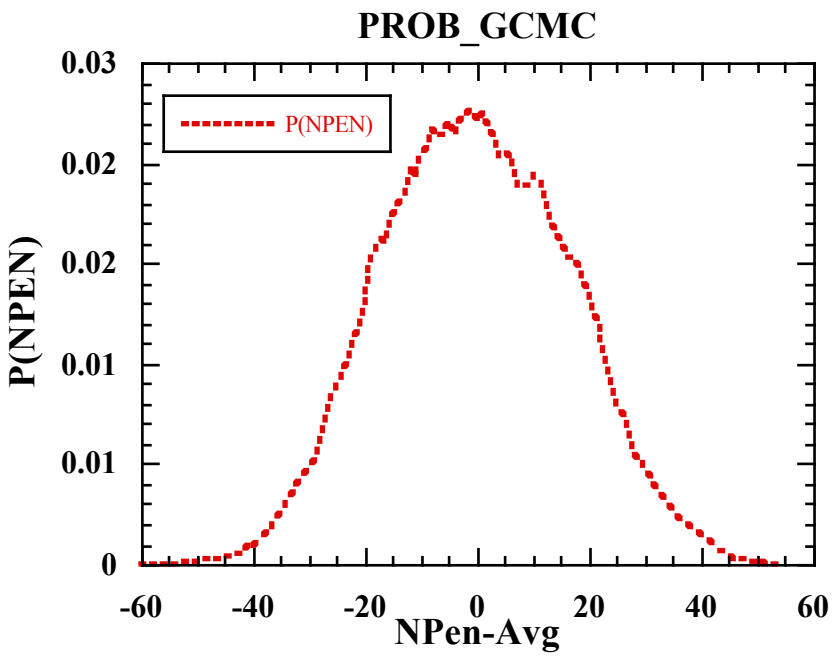
They show that the significant fractions of the available volume are typically between 1.5 and 2.5% for these probes for critical energies in the range $-2kT<\Delta\Phi_c<0$.

The *get_aspace* code uses this approach to estimate the significant space available to the probe molecule based on a user-defined convergence criterion for the solubility; by default the value of this criterion is set to 99.9%. The code then determines from this solubility convergence parameter the value of the upper limit for the integral, $\Delta\Phi_c$, that is required to obtain the convergence required for the solubility. Once $\Delta\Phi_c$ has been obtained it can then be used to determine the fraction of the significant volume available to the centre of the probe molecule. The program reports this value as a percentage.

In the case where spatial resolution has been used in the TPI analysis (IGEOM =1 or 2) the `PEDIST.WIDOM` file contains the *contribution* of each slab or shell to the total probability density. The *get_aspace* code gives the SAV for the slab/shell, or range of slab/shells, chosen as well as its contribution to the total.

prob_gcmc (v1.0 - 25/03/2021)

The program *prob_gcmc* uses the `GCMC.OUT_*` files produced by the GCMC option of *nano_tpi* to obtain a probability density distribution of the number of penetrants in the configurations that have been subject of the GCMC simulations. The resulting distribution is written to the file `PROB_GCMC`. An example distribution is shown below for the case of a reservoir of carbon dioxide at 20 bar in equilibrium with a poly-*para*-OAPS-6FDA-imide. **The current version only calculates the distribution for the first penetrant**.



stoav

Is similar to *datav* except that it only operates on the `STO` file but produces an extra column for the total potential energy of the system and calculates the fluctuations in the total energy with respect to those in the total potential energy. This is useful when doing tests of energy conservation, i.e. NVE mode.

15.8. Graphical utilities

Some basic tools are available which allow *gmq* data to be visualized using *VMD* [42] and *gnuplot*. Versions of *VMD* and *gnuplot* are freely-available for a number of different operating systems.

VMD can be used to visualize the system in a variety of different representations and in turn create snapshots or animations. At the moment, the *gmq* format data files can not be read directly by *VMD* so have to be converted first into PDB and PSF format using *gmq2pdb* (see separate instructions for this data conversion code). As *VMD* can load up several PDB/PSF file combinations, it is possible to represent different subsets of the atoms differently. Although *VMD* can run over X-windows it is likely to be very sluggish if thousands of atoms are to be visualized so it is generally recommended to transfer the PDB and PSF files to a local machine.

gnuplot is particularly adapted for drawing simple X-Y plots using X-window connections. As such it can be useful to monitor the progression of a simulation on a remote machine without having to transfer back any of the data files. A secure connection to a remote machine that allows for X-windows to be opened on the local machine can be performed using the -X option of the *slogin* command:-

```
slogin  -X  -l  remote_username  remote.machine.fr
```

N.B. though that this is subject to X-windows not being blocked by a firewall between you and the remote machine.

As an example, the command *gnu_stox*, explained below, can be used to visualize the contents of the `STO` and `ST` data files. There also exists a number of *gnuplot* scripts in the directory `$BHOME/gnu`. These scripts are a way to avoid retyping the same commands in an interactive gnuplot session. They are invoked with the syntax:-

```
gnuplot my_script
```

where `my_script` is the name of the file containing the *gnuplot* commands.

gnu_stox

This script invokes the *gnuplot* program to plot the data in either the `STO1` and `ST1` or `STO` and `ST` files, depending on whether *SWAP* has been executed or not. *gnu_stox* invokes the gnuplot script $BHOME/gnu/gmqbc_x which sets the terminal type to x11 so should work for terminals supporting X-windows graphics so long as the local machine has given the remote machine permission to open an X-window. The syntax for invoking *gnu_stox* is:-

gnu_stox {NSTART {NEND}}

where NSTART and NEND give the optional range of lines to plot. If omitted all the data is plotted.

gnu_hx

This script plots the contents of the `HDAT` file produced by *hdatplav* on terminals supporting X-windows.

gnu_tpix

This script plots the insertion probability and excess chemical potential as a function of time from the `STO.WIDOM` file produced by *nano_widom*. It also plots the Boltzmann weighted insertion energy distribution from the file `PEDIST.WIDOM`.

15.9. Other utilities

alc2gauss

This program converts files in *Alchemy* `.MOL` format to the input format for the *ab initio* program *Gaussian* [58] . (This program is largely redundant since the switch to using *Hyperchem* for the

construction of molecular fragments. The same operation can now be done using *hyp2gmq* and *gmq2gauss*. Contact DB if more details of *alc2gauss* are required.)

gmq2gauss (v1.7 - 20/05/2019)

Creates template *Gaussian* input files from `CONFIG.NEW` and `PARAM` files. The program writes two template input files called `CONFIG.com` and `CONFIG_restart.com`. The first file can be used to initiate a *Gaussian* optimization run. The header contains the "%Chk=CONFIG" keyword which forces *Gaussian* to save the checkpoint file at the end of the run with the name `CONFIG.chk`. Should *Gaussian* run out of CPU time the checkpoint file can be used to restart the optimization from the last set of stored information. The second template file contains the necessary commands to do this. In particular the two lines:-

```
%Chk=CONFIG
# B3LYP/6-31G** Opt Density=Current Pop=MK MAXDISK=13GB Geom=AllCheck Guess=Read
```

indicate that the previously created checkpoint file `CONFIG.chk` is to be used and that the optimization is to be restarted (Geom and Guess keywords).

The headers also contain the "%NProcShared=8" and "%Mem=19GB" keywords which signal the number of processors to use, this has to be consistent with the number asked for the in the job run file, and the amount of memory required. The default values correspond to those for the machine ada at IDRIS.

Since v1.7 the element abbreviations written by *gmq2gauss* are determined from the mass of the atoms given in the `PARAM` file. If an element type can not be determined from the mass then just the first two letters of the atom type names taken from the `PARAM` file are written. *Element symbols should always be checked before running Gaussian.*

If a `SETDIH.LIST` file exists (see description of *gmq_setdih*) in the directory, *gmq2gauss* will add to the Gaussian input files the syntax corresponding to the running of an optimization with one or more fixed dihedral angles.

gmq_setdih (v1.1 - 28/05/2019)

This code can read an input `CONFIG.NEW` file and set the dihedral angle of a one or more dihedrals to a user-specified value. It is intended to be used for setting up configurations of single molecules for inputting to Gaussian (see description of *gmq2gauss* above). The dihedrals that are to be set are read from the input file `SETDIH.LIST` which takes the following form:

```
1   6  12  13  180.0
```

where the four indices of the atoms involved in the dihedral are given followed by the required new dihedral angle.

If invoked in the absence of a `SETDIH.LIST` file, the code produces a template file called `SETDIH.TEMP` which lists the atom indices of all the *pivotable* dihedrals within the system, i.e. all dihedrals within closed rings are excluded from the list. Also given in `SETDIH.TEMP` are the current values of the dihedral angles for the angles listed, the atom types involved, as well as the dihedral parameter type number in the `PARAM` file. The `SETDIH.TEMP` can be renamed `SETDIH.LIST` and then edited accordingly and in particular all redundant dihedrals, i.e. those sharing a common central bond, must be removed.

hyp2gauss (v1.3 - 21/02/2019)

Creates template *Gaussian* input files from a *Hyperchem* `.ent` format file. The program writes two template input files called `prefix.com` and `prefix_restart.com`, where `prefix` is the prefix of the `.ent` file. The first file can be used to initiate a *Gaussian* optimization run. Its header contains the "%Chk=CONFIG" keyword which forces *Gaussian* to save the checkpoint file at the end of the run with the name `prefix.chk`. Should *Gaussian* run out of CPU time the checkpoint file can be used to restart the optimization from the last set of stored information. The second template file contains the necessary commands to do this. In particular the two lines:-

```
%Chk=prefix
# B3LYP/6-31G** Opt Density=Current Pop=MK MAXDISK=13GB Geom=AllCheck Guess=Read
```

indicate that the previously created checkpoint file `prefix.chk` is to be used and that the optimization is to be restarted (Geom and Guess keywords).

The headers also contain the "%NProcShared=8" and "%Mem=19GB" keywords which signal the number of processors to use, this has to be consistent with the number asked for the in the job run file, and the amount of memory required. The default values correspond to those for the machine ada at IDRIS.

gauss2charge

Converts charges output by *Gaussian* to the format of the `CHARGE` file. The default input file is called `charge.txt`. It should contain just that part of the output file from *Gaussian* that contain the ESP fit charges (1 integer, 1 character variable for the element type, and then the charge). *NB.* Valid `CONFIG.NEW` and `PARAM` files for the corresponding molecule are required to be present in the same directory.

gauss2gmq (v1.1 - 18/09/2013)

Converts the coordinates output by *Gaussian* to *gmq* `CONFIG.NEW` format. The input file should be called `gauss.conf` and should just contain the lines containing the coordinates (3 integers and then x, y, z, on each line) from the *Gaussian* output file. *NB.* Valid `CONFIG.NEW` and `PARAM` files for the

corresponding molecule are required to be present in the same directory. Output is written to `CONFIG.GAUSS`.

gauss2ent (v1.0 - 09/05/2019)

Converts the coordinates output by *Gaussian* to the *Hyperchem* `.ent` format. The input file should be called `gauss.conf` and should just contain the lines containing the coordinates (3 integers and then x, y, z, on each line) from the *Gaussian* output file. *N.B.* a valid `.ent` file for the corresponding molecule is required to be present in the same directory. Output is written to a `.ent_gopt` file.

scan_Alpha

This a shell script which facilitates the optimization of the Ewald sum (see Section 9). Starting from an "equilibrated" configuration of the system of interest a number of 1 step simulations are carried out using different values of α but with R_{max} and K_{max} fixed. The syntax of the command is:-

```
scan_Alpha alpha_min alpha_max alpha_step output_filename exec_name
```

Where `alpha_min` and `alpha_max` are the range of a values to scan with a step of `alpha_step` and `exec_name` is either the name of the MD code or shell script that will run the one step MD jobs. The data is written to the file `output_filename` in the form of a table with headers and the results for each value of α. This data can be analysed or plotted graphically to assess whether sufficient convergence is obtained with the set of input values of R_{max} and K_{max} and, if so, which is the optimum value of α.

scan_dispx

This a shell script which facilitates the optimization of the DGRID parameter in the TEKMC method (see *gmq_dispx*). The syntax of the command is:-

```
usage scan_dispx DGRID_min DGRID_max DGRID_step output_filename run_file
```

Where `DGRID_min` and `DGRID_max` are the range of a values to scan with a step of `DGRID_step` and `run_file` is the name of the shell script that will run the *gmq_dispx* analysis code. The data is written to the file `output_filename` in the form of a table of values of DGRID and the root mean square difference (RMSD) between the mean square displacement obtained from TEKMC and those obtained directly from the MD. It is important that the `DISP.OUT` file, as generated by *gmq_disp*, be present in the directory so that the comparison can be made with the `DISPX.OUT` file generated by *gmq_dispx*. The comparison is performed by the code *comp_disp* (see after).

An example script for running gmq_dispx is shown below. It has to be adapted for each particular case. Note that the DGRID line must not contain sequences of more than one space.

```
gmq_dispx<<!
32      # Atom type of diffusing molecule
n       # Print all batch headers?
200     # Start Batch No.
6000    # End Batch No.
```

```
y      # Change time difference?
10.0   # New time diff.
y      # Change the subcell width?
2.0 # DGRID: New subcell width
y      # Change the No. of walks?
10000  # New No. of random walks
y      # Change the Maxm. time for random walks?
15000  # New Maxm. time for random walks
n      # Store the random walk trajectories?
y      # Change the time resolution of random walk trajectories?
100.0  # New resolution in ps

!
```

comp_disp (v1.0 - 14/05/2018)

This is a code that compares the values for the mean square displacements in a `DISPX.OUT` file, as generated by the TEKMC method in *gmq_dispx*, with those in `DISP.OUT` as evaluated by *gmq_disp* from the the actual stored configurations of the MD simulation. The code reports the root mean square difference (RMSD) between the two as well as writing the result to the file `RMSD`.

readtmp

Should a simulation breakdown after a certain number of steps, without a neat exit message in the `OUTPUT.NEW` file, then it is sometimes useful to examine the contents of the scratch files (`STOSCR, STSCR, HSSCR` etc.) for signs of something untoward happening. The program *readtmp* can read the entries in the binary sequential scratch files and convert them to a readable format. These are printed on screen and written to the file `DATOUT`.

cordat (v1.0 - 06/05/2019)

This program can be used to correct the time column in the history files (E.g. STO, ST, HS, HDAT). This can be useful in cases where the clock has not been zeroed, for example. The user inputs the name of the file and the time offset that is required. The time-corrected file is written to a file with the same name appended by `_time_corr`.

RESUB

Is a shell script that allows for automatic job chaining when using a batch queue. Once a batch submitted *gmq* or *ddgmq* run has exited successfully, RESUB can be used to perform the *SWAP* and then resubmit the job, if required. The syntax of the command is:-

```
RESUB runjob time_in_ps
```

where `runjob` is the script file for submitting the job to the batch queue and `time_in_ps` is a number indicating the time in picoseconds up until which the simulation should run. This command should appear at the end of the file `runjob`, within some appropriate "IF" loop test for a successful finish to a *gmq* or *ddgmq* run. In order to have a mechanism to stop the execution at any intervening stage a file called `RUN`

must exist in the directory in which the job is initiated. The contents of `RUN` are irrelevant. If at any stage the run is to be stopped manually the file `RUN` can either be erased or renamed by the user. When RESUB itself detects that the simulation has finished it renames the file `RUN` as `STOP`.

SWAP

Is a shell script which automates the setting up of the *gmq* input datasets, following a successful previous run, for a continuation run (see also Section 7). *SWAP* does a number of checks to try to ensure that the job finished correctly. It checks for example that the output `STO1` file is longer than the input `STO` file, etc., and it can be that *SWAP* fails if the input files have been generated on one machine and the output files on another. In this case checks have to be made manually. If *SWAP* finishes correctly all the output files are swapped onto the corresponding input ones and copies of certain files are kept:- `CONF1` is kept on `CONF1_keep`, the original input file `CONFIG.NEW` is moved to `CONFIG.NEW_keep` before `CONFIG.NEW1` is swapped onto `CONFIG.NEW`, and the `OUTPUT.NEW` file is stored as `OUTPUT.keep`.

16. Fundamental constants

The following values are defined internally in *gmq* for certain fundamental constants:-

Avogadro's Number	6.0221367×10^{23} mol^{-1}
Boltzmann's Constant	1.380658×10^{-23} J K^{-1}
Charge on the proton (e)	1.60217733×10^{-19} C
$e^2/4\pi\varepsilon_o$	2.30691×10^{-28} J m
π	ACOS(-1.0)

all other constants are derived from these.

17. References

[1] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Clarendon Press, Oxford, 1987.

[2] J.M. Haile, Molecular Dynamics Simulation: Elementary Methods, John Wiley & Sons Ltd., New York, 1992.

[3] D. Frenkel, B. Smit, Understanding molecular simulation: from algorithms to applications, Academic Press Inc., San Diego, 1996.

[4] S. Tsuneyuki, Molecular Dynamics Simulation of Silica with a First-Principles Interatomic Potential, Molecular Engineering, 6 (1996) 157.

[5] H. Schreiber, O. Steinhauser, Molecular-Dynamics Studies Of Solvated Polypeptides - Why the Cutoff Scheme Does Not Work, Chemical Physics, 168 (1992) 75-89.

[6] D.M. York, T.A. Darden, L.G. Pedersen, The Effect Of Long-Range Electrostatic Interactions In Simulations Of Macromolecular Crystals - a Comparison Of the Ewald and Truncated List Methods, J. Chem. Phys., 99 (1993) 8345-8348.

[7] K.F. Lau, H.E. Alper, T.S. Thacher, T.R. Stouch, Effects Of Switching-Functions On the Behavior Of Liquid Water In Molecular-Dynamics Simulations, Journal Of Physical Chemistry, 98 (1994) 8785-8792.

[8] G.S. Delbuono, T.S. Cohen, P.J. Rossky, Effects Of Long-Range Interactions On the Dynamics Of Ions In Aqueous-Solution, Journal Of Molecular Liquids, 60 (1994) 221-236.

[9] P.P. Ewald, Die Berechnung optischer und elektrostatischer Gitterpotentiale, Ann. Phys., 64 (1921) 253-287.

[10] W. Smith, A Replicated Data Molecular-Dynamics Strategy For the Parallel Ewald Sum, Comput. Phys. Commun., 67 (1992) 392-406.

[11] F. Müller-Plathe, Singularity-free Treatment of Linear Bond Angles, The CCP5 Newsletter, 44 (1995) 40-41.

[12] K.D. Hammonds, J.-P. Ryckaert, On the convergence of the SHAKE algorithm, Comput. Phys. Commun., 62 (1991) 336-351.

[13] M. Fixman, Classical Statistical Mechanics of Constraints: A Theorem and Application to Polymers, Proceedings of the National Academy of Sciences of the United States of America, 71 (1974) 3050-3053.

[14] J.-P. Ryckaert, G. Ciccotti, H.J.C. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of liquid alkanes, J. comput. Phys., 23 (1977) 327-341.

[15] G. Ciccotti, M. Ferrario, J.-P. Ryckaert, Molecular dynamics of rigid systems in cartesian coordinates: A general formulation., Mol. Phys., 47 (1982) 1253-1264.

[16] C. Engin, J. Vrabec, H. Hesse, On the difference between a point multipole and an equivalent linear arrangement of point charges in force field models for vapour–liquid equilibria; partial charge based models for 59 real fluids, Molecular Physics, 109 (2011) 1975-1982.

[17] D. Brown, J.H.R. Clarke, A loose coupling constant pressure molecular dynamics algorithm for use in the modelling of polymer materials, Comput. Phys. Commun., 62 (1991) 360-369.

[18] D. Brown, S. Neyertz, A general pressure tensor calculation for molecular dynamics simulations, Mol. Phys., 84 (1995) 577-595.

[19] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola, J.R. Haak, Molecular dynamics with coupling to an external bath, J. Chem. Phys., 81 (1984) 3684-3690.

[20] D.J. Evans, B.L. Holian, The Nose–Hoover thermostat, J. Chem. Phys., 83 (1985) 4069-4074.

[21] A.P. Heiner, Predictive Aspects of Molecular Dynamics Simulations for Proteins: Application to subtilisin BPN, in, University of Groningen, Groningen, 1992.

[22] S. Nosé, M.L. Klein, Constant pressure molecular dynamics for molecular systems, Mol. Phys., 50 (1983) 1055-1076.

[23] D. Brown, J.H.R. Clarke, Molecular dynamics simulation of an amorphous polymer under tension I. Phenomenology, Macromolecules, 24 (1991) 2075-2082.

[24] J.I. McKechnie, R.N. Haward, D. Brown, J.H.R. Clarke, Effects of chain configurational properties on the stress-strain behaviour of glassy linear polymers, Macromolecules, 26 (1993) 198-202.

[25] D. Brown, J.H.R. Clarke, M. Okuda, T. Yamazaki, A Domain Decomposition Parallel-Processing Algorithm For Molecular- Dynamics Simulations Of Polymers, Comput. Phys. Commun., 83 (1994) 1-13.

[26] D. Brown, J.H.R. Clarke, A direct method of studying the reaction rates by equilibrium molecular dynamics: Application to the kinetics of isomerization in liquid n-butane, J. Chem. Phys., 92 (1990) 3062-3073.

[27] D. Brown, J.H.R. Clarke, On the determination of rate constants from equilibrium molecular dynamics simulations, J. Chem. Phys., 93 (1990) 4117-4122.

[28] D. Brown, J.H.R. Clarke, Erratum: On the determination of rate constants from equilibrium molecular dynamics simulations, J. Chem. Phys., 94 (1990) 4684.

[29] P. Auffinger, S. Louise-May, E. Westhof, Multiple Molecular Dynamics Simulations of the Anticodon Loop of tRHAAsp in Aqueous Solution with Counterions, J. Am. Chem. Soc., 117 (1995) 6720-6726.

[30] W. Smith, FORTRAN Code for the EWALD Summation Method, CCP5 Newsletter, 21 (1986) 37.

[31] D. Fincham, Optimisation of the Ewald sum, Information Quarterly for Computer Simulation of Condensed Phases, 38 (1993) 17-24.

[32] D. Brown, J.H.R. Clarke, M. Okuda, T. Yamazaki, A Domain Decomposition Parallelization Strategy For Molecular- Dynamics Simulations On Distributed Memory Machines, Comput. Phys. Commun., 74 (1993) 67-80.

[33] D. Brown, H. Minoux, B. Maigret, A domain decomposition parallel processing algorithm for molecular dynamics simulations of systems of arbitrary connectivity, Comput. Phys. Commun., 103 (1997) 170-186.

[34] S. Neyertz, D. Brown, Preparation of Bulk Melt Chain Configurations of Polycyclic Polymers, J. Chem. Phys., 115 (2001) 708-717.

[35] P.J. Flory, The Statistical Mechanics of Chain Molecules, Hanser Publishers, New York, 1988.

[36] S. Neyertz, Tutorial: Molecular Dynamics Simulations of Microstructure and Transport Phenomena in Glassy Polymers, Soft Materials, 4 (2007) 15-83.

[37] M. Clark, R.D. Cramer III, N. van Opdenbosch, Validation of the General Purpose Tripos 5.2 Force Field, J. of Computational Chemistry, 10 (1989) 982-1012.

[38] T. Proffen, R.B. Neder, DISCUS, a Program for Diffuse Scattering and Defect Structure Simulations, J. Appl. Cryst., 30 (1997) 171-175.

[39] B.W.H. van Beest, G.J. Kramer, R.A. van Santen, Force fields for silicas and aluminophosphates based on ab initio calculations, Phys. Rev. Lett., 64 (1990) 1955-1958.

[40] S. Marceau, Architecture Multiéchelle et Propriétés Mécaniques de Nanocomposites, in: Laboratoire des Matériaux Organiques à Propriétés Spécifiques (Université de Savoie), Université de Savoie, Bourget-du-Lac, 2003, pp. 193.

[41] D. Barbier, D. Brown, A.-C. Grillet, S. Neyertz, The Interface between End-Functionalized PEO Oligomers and a Silica Nanoparticle Studied by Molecular Dynamics Simulations, Macromolecules, 37 (2004) 4695-4710.

[42] W. Humphrey, A. Dalke, K. Schulten, VMD - Visual Molecular Dynamics, J. Molec. Graphics, 14.1 (1996) 33-38.

[43] J. Vrabec, J. Stoll, H. Hasse, A Set of Molecular Models for Symmetric Quadrupolar Fluids, J. Phys. Chem. B, 105 (2001) 12126-12133.

[44] C. Lanczos, A Precision Approximation of the Gamma Function, SIAM Journal on Numerical Analysis, 1 (1964) 86–96.

[45] S. Pandiyan, D. Brown, S. Neyertz, N.F.A. van der Vegt, Carbon Dioxide Solubility in Three Fluorinated Polyimides Studied by Molecular Dynamics Simulations, Macromolecules, 43 (2010) 2605-2621.

[46] S. Neyertz, D. Brown, Single- and mixed-gas sorption in large-scale molecular models of glassy bulk polymers. Competitive sorption of a binary CH4/N2 and a ternary CH4/N2/CO2 mixture in a polyimide membrane, Journal of Membrane Science, 614 (2020) 118478.

[47] D.N. Theodorou, Principles of Molecular Simulation of Gas Transport in Polymers, in: Y.P. Yampolskii, I. Pinnau, B.D. Freeman (Eds.) Materials Science of Membranes for Gas and Vapour Separation, John Wiley & Sons Ltd., Hoboken, N.J., USA, 2006, pp. 466.

[48] S. Neyertz, D. Brown, A Trajectory-Extending Kinetic Monte Carlo (TEKMC) method for estimating penetrant diffusion coefficients in molecular dynamics simulations of glassy polymers, Macromolecules, 43 (2010) 9210-9214.

[49] F.H. Stillinger, Theory and molecular models for water, Advances in Chemical Physics, 31 (1975) 1-101.

[50] D.C. Rapaport, Hydrogen bonds in water: Network organizations and lifetimes, Molecular Physics, 50 (1983) 1151-1162.

[51] D. Chandler, Statistical mechanics of isomerization dynamics in liquids and the transition state approximation, J. Chem. Phys., 68 (1978) 2959-2970.

[52] R. Edberg, D.J. Evans, G.P. Morriss, Conformational kinetics in liquid butane by nonequilibrium molecular dynamics, J. Chem. Phys., 87 (1987) 5700-5708.

[53] D. Brown, V. Marcadon, P. Mélé, N.D. Albérola, Effect of Filler Particle Size on the Properties of Model Nanocomposites, Macromolecules, 41 (2008) 1499 -1511.

[54] A. Ben-Naim, Molecular theory of solutions, Oxford University Press, New York, 2006.

[55] G.L. Deitrick, L.E. Scriven, H.T. Davis, Efficient molecular simulation of chemical potentials, J. Chem. Phys., 90 (1989) 2370-2385.

[56] G. Dömötör, R. Hentschke, Atomistically modeling the chemical potential of small molecules in dense systems, J. Phys. Chem. B, 108 (2004) 2413-2417.

[57] M. Fukuda, Solubilities of small molecules in polyethylene evaluated by a test-particle-insertion method, J. Chem. Phys., 112 (2000) 478-486.

[58] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, V.G. Zakrzewski, J.A. Montgomery Jr., R.E. Stratmann, J.C. Burant, S. Dapprich, J.M. Millam, A.D. Daniels, K.N. Kudin, M.C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G.A. Petersson, P.Y. Ayala, Q. Cui, K. Morokuma, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J. Cioslowski, J.V. Ortiz, A.G. Baboul, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, J.L. Andres, C. Gonzalez, M. Head-Gordon, E.S. Replogle, J.A. Pople, Gaussian 98 (Revision A.7), in, 1998.