Software News and Updates

*n*Moldyn: A Program Package for a Neutron Scattering Oriented Analysis of Molecular Dynamics Simulations

T. RÓG,[1,2] K. MURZYN,[1,2] K. HINSEN,[2] G. R. KNELLER[2]
[1]*Departament of Biophysics, Institute of Molecular Biology, Jagiellonian University, ul. Gronostajowa 7, 30-387 Kraków, Poland*
[2]*Centre de Biophysique Moléculaire (UPR 4301 CNRS), Rue Charles Sadron, 45071 Orléans Cedex 2, France*

Received 30 July 2002; Accepted 1 November 2002

Abstract: We present a new implementation of the program *n*Moldyn, which has been developed for the computation and decomposition of neutron scattering intensities from Molecular Dynamics trajectories (Comp. Phys. Commun 1995, 91, 191–214). The new implementation extends the functionality of the original version, provides a much more convenient user interface (both graphical/interactive and batch), and can be used as a tool set for implementing new analysis modules. This was made possible by the use of a high-level language, Python, and of modern object-oriented programming techniques. The quantities that can be calculated by *n*Moldyn are the mean-square displacement, the velocity autocorrelation function as well as its Fourier transform (the density of states) and its memory function, the angular velocity autocorrelation function and its Fourier transform, the reorientational correlation function, and several functions specific to neutron scattering: the coherent and incoherent intermediate scattering functions with their Fourier transforms, the memory function of the coherent scattering function, and the elastic incoherent structure factor. The possibility to compute memory function is a new and powerful feature that allows to relate simulation results to theoretical studies.

 J Comput Chem 24: 657–667, 2003

Key words: Molecular Dynamics; neutron scattering; memory function

Introduction

Although Molecular Dynamics (MD) simulation techniques are by now widely used in physics, chemistry, and biology, their possibilities are often not fully exploited. One reason is the lack of easy-to-use analysis tools. Although the simulation methods and algorithms themselves are widely known and discussed, the often equally complex analysis techniques are usually neglected. This is especially true for biomolecular simulations, where the complexity of the systems being studied forces most computational scientists to use standard program packages, which contain only very limited functionality for analyzing MD trajectories. Moreover, analysis techniques are necessarily problem-specific, and thus useful to a much smaller user community than the general MD simulation methods.

In this article, we present an analysis library and interactive analysis tool that was developed mainly for use in connection with neutron scattering experiments, although many of the quantities are also used in other contexts. The combination of thermal neutron scattering experiments and Molecular Dynamics simulations is a powerful tool to study the structure and dynamics of complex molecular systems. Thermal neutron scattering gives information on time *and* space correlations of atomic positions.[1,2] The relevant time scale ranges from subpicoseconds to about 10 nanoseconds, and space correlations are detected on length scales between 1 Å and about 10 nm. This time and space domain is also covered by classical molecular dynamics simulations, and both techniques can be considered as complementary. The neutron–target interaction can be described by Fermi's pseudopotential, which has zero range on the length scales under consideration and is centered on the atomic nuclei of the targets. In both experiment and simulation the atoms are thus considered as point-like particles. If recoil effects are not dominant[3] and if the scattering system under consideration is well described by classical mechanics, the differential scattering cross-section can be expressed in terms of classical time correlation functions of the spatially Fourier transformed particle density.

Correspondence to: G. R. Kneller; e-mail: kneller@cnrs-orleans.fr

The latter can be obtained easily from Molecular Dynamics simulations, enabling a direct comparison between simulated and measured neutron scattering intensities. The experimental data can be used to judge the quality of the MD force field, which is the central input for the simulations, and, conversely, the simulations allow a detailed analysis of the dynamical and structural behavior of the system under consideration. This aspect is particularly important for complex systems, such as biologic macromolecules, because model building on the basis of simulated data, where all details are accessible, is in general much more powerful than model building on the basis of neutron spectra.[4–8]

The program *n*Moldyn is designed for the efficient calculation and decomposition of neutron scattering intensities from MD simulations. The calculation of various space and time correlation functions permits a detailed analysis of the structure and dynamics of the system under consideration. *n*Moldyn permits the calculation of incoherent and coherent dynamic structure factors, elastic incoherent structure factors (EISFs), mean-square displacements, and translational velocity autocorrelation functions. In addition, rigid-body trajectories of subunits of the system can be extracted from molecular dynamics trajectory files. These subunits can be arbitrarily defined, and their size can range from a few atoms to a whole domain in a macromolecule. From the rigid body trajectories, angular correlation functions and reorientational correlation functions can be obtained.

The original *n*Moldyn[9] was developed several years ago as a modular package of Fortran 77 programs using a special-purpose data file format for efficient data access. Modern software engineering techniques permitted significant improvements in flexibility, user-friendliness, and portability. The second-generation *n*Moldyn presented in this article, which is a complete reimplementation, offers an interactive graphical user interface for standard calculations, highly flexible script-based processing for nonstandard applications, and a machine-independent compact binary file format. These improvements were made possible by the use of

1. the object-oriented high-level language Python,[10]
2. the fast array package Numerical Python,[11]
3. the efficient FFT implementation FFTW,[12]
4. the portable binary file format netCDF and the corresponding library,[13]
5. the scientific computing library Scientific Python,[14]
6. the molecular simulation library MMTK.[15]

All of these packages are developed and distributed following the Open Source principles,[16] which means that anyone can use and improve them without being hindered by licensing restrictions. As a result, they are more stable and better tested than packages that are owned by a small group of developers, and therefore, they are well suited as a basis for developing special-purpose programs such as *n*Moldyn.

The use of a high-level interpreted language may be surprising at first sight, because the calculation of most of the quantities implemented in *n*Moldyn is rather time consuming. A closer inspection reveals that all the time-critical parts of the algorithms use efficient implementations in C or Fortran, which are provided by other packages (Numerical Python, FFTW, and MMTK) together with a Python interface. *n*Moldyn itself contains only Python code. This demonstrates the usefulness of a mixed-language development approach, in which ultimately only small parts of the total code must be written in efficient but relatively cumbersome low-level languages.

Quantities and Algorithms

In this section we give an overview of the quantities calculated by *n*Moldyn and how they can be obtained efficiently. For the theoretical background of the quantities specific to neutron-scattering, the reader is referred to refs. 1, 2, and 17.

The following notation is used throughout this section: N is the number of (selected) atoms in the system, or in the subsystem for which the analysis is performed. The atoms are labelled by Greek indices α, β, ... Their positions are denoted by $\mathbf{R}_\alpha$, and their velocities by $\mathbf{v}_\alpha$. The coherent and incoherent scattering lengths are defined by

$$b_{\alpha,\mathrm{coh}} = \overline{b_\alpha}, \tag{1}$$

$$b_{\alpha,\mathrm{inc}} = \sqrt{\overline{b_\alpha^2} - \overline{b_\alpha}^2}, \tag{2}$$

respectively, where the symbol $\overline{\ldots}$ denotes an average over isotopes and relative spin orientations of neutron and nucleus. In quantities that are averages over all atoms, w_α denotes a set of weights. *n*Moldyn implements different weighting schemes:

1. Equal weighting: $w_\alpha = 1$
2. Mass weighting: $w_\alpha \propto m_\alpha$ and $\sum_{\alpha=1}^{N} w_\alpha = N$.
3. Incoherent neutron scattering: $w_\alpha \propto b_{\alpha,\mathrm{inc}}^2$ and $\sum_{\alpha=1}^{N} w_\alpha = N$.
4. Coherent neutron scattering: $w_\alpha \propto b_{\alpha,\mathrm{coh}}$ and $\sum_{\alpha=1}^{N} w_\alpha^2 = N$.

Static Correlation Functions

Static Structure Factor and EISF

There are two important thermodynamical averages that are related to the intermediate scattering functions described later.

The first average is the *static structure factor,* defined as

$$S(\mathbf{q}) = \frac{1}{N} \sum_{\alpha,\beta} b_{\alpha,\mathrm{coh}} b_{\beta,\mathrm{coh}} \langle \exp[i\mathbf{q} \cdot (\mathbf{R}_\beta - \mathbf{R}_\alpha)] \rangle. \tag{3}$$

It can be related the *coherent intermediate scattering function,*

$$S(\mathbf{q}) = \mathcal{F}_{\mathrm{coh}}(\mathbf{q}, 0), \tag{4}$$

and to the static pair correlation function.[18]

The second average is the *elastic incoherent structure factor* (EISF), which is defined as the long-time limit of the *incoherent intermediate scattering function* (see next section),

$$EISF(\mathbf{q}) = \lim_{t \to \infty} \mathcal{F}_{\mathrm{inc}}(\mathbf{q}, t), \tag{5}$$

and may be computed as

$$EISF(\mathbf{q}) = \frac{1}{N}\sum_{\alpha} b_{\alpha,\mathrm{inc}}^2 \langle |\exp[i\mathbf{q}\cdot\mathbf{R}_\alpha]|^2\rangle, \tag{6}$$

The EISF describes the amplitude of the elastic line in the neutron scattering spectrum, and therefore, the amplitude of atomic fluctuations in the system.

The static structure factor can be calculated exactly as defined in eq. (4); therefore, no special routines are provided in *n*Moldyn. This is not the case for the EISF, which in practice, cannot be obtained from eq. (5) due to bad statistics in correlation functions for large time arguments. It is better to calculate it from relation (6).

Time Correlation Functions

In the following we describe briefly the time correlation functions that are accessible by neutron scattering. Time correlation functions describing the rotational motion of molecules or parts thereof are described later.

Velocity Autocorrelation Function

The velocity autocorrelation function (VACF) is a standard quantity in physical chemistry and statistical physics. It is defined as

$$C_{vv}(t) = \frac{1}{N}\sum_{\alpha=1}^{N} w_\alpha \frac{1}{3}\langle \mathbf{v}_\alpha(0)\cdot\mathbf{v}_\alpha(t)\rangle. \tag{7}$$

In some cases, for example, for nonisotropic systems, it is useful to define VACFs along a specific axis,

$$C_{vv}(t;\,\mathbf{n}) = \frac{1}{N}\sum_{\alpha=1}^{N} w_\alpha \langle v_\alpha(0;\,\mathbf{n}) v_\alpha(t;\,\mathbf{n})\rangle, \tag{8}$$

where $v_\alpha(t;\,\mathbf{n})$ is given by

$$v_\alpha(t;\,\mathbf{n}) = \mathbf{n}\cdot\mathbf{v}_\alpha(t). \tag{9}$$

The vector $\mathbf{n}$ is a unit vector defining a space-fixed axis.

Mean-Square Displacement

The Mean-square displacement (MSD) is defined by

$$\Delta^2(t) = \frac{1}{N}\sum_{\alpha=1}^{N} w_\alpha \langle \mathbf{d}_\alpha^2(t)\rangle \tag{10}$$

with

$$\mathbf{d}_\alpha(t) = \mathbf{R}_\alpha(t) - \mathbf{R}_\alpha(0). \tag{11}$$

It is not a time correlation function, but can be written as the autocorrelation function of $\mathbf{R}_\alpha(t)$ plus two time-dependent functions that are simple to calculate. As for the VACF, it is useful to define a mean-square displacement with respect to a given axis:

$$\Delta_\alpha^2(t;\,\mathbf{n}) = \frac{1}{N}\sum_{\alpha=1}^{N} w_\alpha \langle d_\alpha^2(t;\,\mathbf{n})\rangle, \tag{12}$$

with

$$d_\alpha(t;\,\mathbf{n}) \doteq \mathbf{n}\cdot\mathbf{d}_\alpha(t). \tag{13}$$

If the atoms under consideration perform *un*confined motions, the long-time limit of $\Delta^2(t)$ is related to the diffusion constant, D:

$$\lim_{t\to\infty} \Delta^2(t) = 6Dt. \tag{14}$$

The diffusion constant can also be defined in terms of the VACF:

$$D = \int_0^\infty dt C_{vv}(t). \tag{15}$$

Intermediate Scattering Functions

The *coherent intermediate scattering function,* $\mathcal{F}_{\mathrm{coh}}(\mathbf{q},\,t)$, is defined as

$$\mathcal{F}_{\mathrm{coh}}(\mathbf{q},\,t) = \frac{1}{N}\sum_{\alpha,\beta} b_{\alpha,\mathrm{coh}} b_{\beta,\mathrm{coh}} \langle \exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(0)]\exp[i\mathbf{q}\cdot\mathbf{R}_\beta(t)]\rangle. \tag{16}$$

It gives information about collective motions in condensed matter. Single particle correlations are described by the *incoherent intermediate scattering function,* $\mathcal{F}_{\mathrm{inc}}(\mathbf{q},\,t)$, which contains only "self"-terms,

$$\mathcal{F}_{\mathrm{inc}}(\mathbf{q},\,t) = \frac{1}{N}\sum_{\alpha} b_{\alpha,\mathrm{inc}}^2 \langle \exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(0)]\exp[i\mathbf{q}\cdot\mathbf{R}_\alpha(t)]\rangle, \tag{17}$$

and thus gives only information about single-atom motions.

The mean-square displacement can be related to the incoherent intermediate scattering function via the cumulant expansion[19,20]

$$\mathcal{F}_{\mathrm{inc}}(\mathbf{q},\,t) = \frac{1}{N}\sum_{\alpha} b_{\alpha,\mathrm{inc}}^2 \exp[-q^2\rho_{\alpha,1}(t) + q^4\rho_{\alpha,2}(t) \mp \cdots]. \tag{18}$$

The cumulants $\rho_{\alpha,k}(t)$ are defined as

$$\rho_{\alpha,1}(t) = \frac{1}{2!}\langle d_\alpha^2(t;\,\mathbf{n}_q)\rangle \tag{19}$$

$$\rho_{\alpha,2}(t) = \frac{1}{4!}\left[\langle d_\alpha^4(t;\,\mathbf{n}_q)\rangle - 3\langle d_\alpha^2(t;\,\mathbf{n}_q)\rangle^2\right]$$

$$\vdots\,, \tag{20}$$

where $d_\alpha(t;\,\mathbf{n}_q)$ is defined according to (13) with $\mathbf{n}_q$ being the unit vector in the direction of $\mathbf{q}$. In the Gaussian approximation, the above expansion is truncated after the q^2-term. For certain model systems like the ideal gas, the harmonic oscillator, and a particle undergoing Einstein diffusion, this is exact. For these systems the incoherent intermediate scattering function is completely determined by the mean-square displacement.

Frequency Spectra

The main quantity of interest in most neutron scattering experiments with thermal neutrons is the *dynamic structure factor,* $S(\mathbf{q},\,\omega)$, which describes the number of neutrons scattered with momentum transfer $\hbar\mathbf{q}$ and energy transfer $\hbar\omega$. It is usually split into a coherent and an incoherent part,

$$S(\mathbf{q},\,\omega) = S_{\mathrm{coh}}(\mathbf{q},\,\omega) + S_{\mathrm{inc}}(\mathbf{q},\,\omega), \tag{21}$$

where $S_{\mathrm{coh}}(\mathbf{q},\,\omega)$ is the time-domain Fourier transform of the coherent intermediate scattering function $\mathcal{F}_{\mathrm{coh}}(\mathbf{q},\,t)$ and $S_{\mathrm{inc}}(\mathbf{q},\,\omega)$ is the time-domain Fourier transform of the incoherent intermediate scattering function $\mathcal{F}_{\mathrm{inc}}(\mathbf{q},\,t)$,

$$S_{\mathrm{coh/inc}}(\mathbf{q},\,\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} dt\,\exp[-i\omega t]\mathcal{F}_{\mathrm{coh/inc}}(\mathbf{q},\,t). \tag{22}$$

Because the incoherent scattering cross-section of hydrogen atoms dominates all other neutron scattering cross-sections by about a factor of 20, one measures essentially the self-correlations of hydrogen atoms when studying samples like biologic macromolecules and many polymers that contain a large amount of hydrogen atoms.

Another quantity whose time-domain Fourier transform is of interest is the velocity autocorrelation function (VACF). Its Fourier transform is the *density of states* (DOS), $G(\omega)$.

$$G(\omega) = \int_0^{\infty} dt\,\cos[\omega t]C_{vv}(t). \tag{23}$$

According to (15), the value of the DOS at $\omega = 0$ gives the diffusion constant:

$$D = \int_0^{\infty} dt C_{vv}(t) = G(0). \tag{24}$$

If the VACF is calculated neutron weighted, then the corresponding neutron-weighted density of states is related to the incoherent dynamic structure factor by

$$\lim_{q\to 0}\frac{\omega^2}{q^2}\,S_{\mathrm{inc}}(\mathbf{q},\,\omega) = \frac{G(\omega)}{\pi}. \tag{25}$$

Frequency Filtering

It is often of interest to restrict attention to motions in a specific frequency interval, both for quantitative analysis and for visualization by animated displays. This is particularly useful to study low-frequency motions without being distracted by the high-frequency "noise." *n*Moldyn can apply a frequency filter to the atomic trajectories, either of the whole system or of a user-defined subset. The result is stored in a new trajectory file that contains only motions in the chosen interval.

Frequency filtering uses a straightforward algorithm: take the discrete Fourier transform of each particle coordinate, set the Fourier coefficients outside the filtering interval to zero, and transform back to the time domain. This corresponds to applying a rectangular window in the frequency domain.

Memory Functions

Memory functions have been used for a long time in theoretical statistical physics to describe the time dependence of autocorrelation functions, especially the velocity autocorrelation function, whose memory function can be seen as the generalization of the friction constant.[18,20] Its use in the context of Molecular Dynamics simulations has been hindered by the lack of a suitable numerical algorithm for their calculation. Such an algorithm has been published recently,[21] and is now implemented in *n*Moldyn. The reader is referred to this article for more details.

Definition

In the context of VACF memory functions one usually considers the normalized VACF, which is defined as

$$\psi(t) := \frac{\langle v(t)v(0)\rangle}{\langle v^2(0)\rangle}. \tag{26}$$

Here, $v(t)$ is the x-, y-, or z-component of the velocity of a "tagged" atom. Assuming that all selected atoms are physically equivalent, $\psi(t)$ can be identified with $C_{vv}(t)/C_{vv}(0)$, where $C_{vv}(t)$ has been defined in (7). The memory function $\xi(t)$ of $\psi(t)$ is defined by the relation

$$\frac{d}{dt}\,\psi(t) = -\int_0^t d\tau\xi(t-\tau)\psi(\tau). \tag{27}$$

Equation (27) is called the *memory function equation* (*ME*).

Autoregressive (AR) Process

To compute $\xi(t)$ from an MD velocity trajectory, the latter is considered a discrete "signal," $v(n) \equiv v(n\Delta t)$, which is modeled by an *autoregressive stochastic process* of order P,[22,23]

$$v(t) = \sum_{n=1}^{P} a_n^{(P)} v(t - n\Delta t) + \varepsilon_P(t). \tag{28}$$

Here, $\varepsilon_P(t)$ is *white noise* with zero mean and amplitude σ_P. The coefficients $\{a_n^{(P)}\}$ are fitted to the MD data using Burg's algorithm,[24,25] and σ_P is given by

$$\sigma_P^2 = 1 - \sum_{n=1}^{P} a_n^{(p)} \psi(n\Delta t). \tag{29}$$

In all following calculations *n*Moldyn works with a set of coefficients $\{a_n\}$ that has been averaged over all selected atoms and the three Cartesian coordinates.

VACF within the AR Model

Within the AR-model the z-transform of the VACF has the form

$$\Psi^{(AR)}(z) = \frac{1}{a_P^{(P)}} \frac{-z^P \sigma_P^2}{\Pi_{k=1}^{P}(z - z_k)\, \Pi_{l=1}^{P}(z - z_l^{-1})}. \tag{30}$$

Here, the $\{z_k\}$ are the zeros of

$$p(z) = z^P - \sum_{k=1}^{P} a_k^{(P)} z^{P-k}. \tag{31}$$

We recall that the z-transform of an arbitrary discrete function $f(n)$ is given by $F(z) = \Sigma_{n=-\infty}^{+\infty} f(n) z^{-n}$, and the inverse transform by $f(n) = (1/2\pi i) \oint_C dz z^{n-1} F(z)$. Applying the inverse z-transform to (30) yields

$$\psi^{(AR)}(n) = \sum_{j=1}^{P} \beta_j z_j^{|n|}, \tag{32}$$

where the coefficients β_j are given by

$$\beta_j = \frac{1}{a_P} \frac{-z_j^{P-1} \sigma_P^2}{\Pi_{k=1,k\neq j}^{P}(z_j - z_k)\, \Pi_{l=1}^{P}(z_j - z_l^{-1})}. \tag{33}$$

Note that $\psi^{(AR)}(n)$ has a multiexponential form, and that the stability criterion

$$|z_j| < 1, j = 1, \ldots, P, \tag{34}$$

must be fulfilled. This is guaranteed by the Burg-algorithm.[24,25]

Density of States within the AR Model

Evaluating $\Psi^{(AR)}(z)$ as given by (30) at $z = \exp(i\omega\Delta t)$ yields the density of states within the AR model:

$$G^{(AR)}(\omega) = \frac{\Delta t}{2} \frac{k_B T}{M} \Psi^{(AR)}(\exp[i\omega\Delta t]). \tag{35}$$

Here, M is the mass of the tagged atom, k_B is the Boltzmann constant, and T the temperature. Note that the VACF and the density of states within the AR model are entirely determined by the coefficients $a_n^{(P)}$.

Discrete Memory Function within the AR Model

The discrete VACF $\psi(n)$ obeys the discretized memory function eq. (27),

$$\frac{\psi(n+1) - \psi(n)}{\Delta t} = -\sum_{k=0}^{n-1} \Delta t \xi(n-k)\psi(k), \tag{36}$$

which can be subjected to a *one-sided* z-transform to yield

$$\Xi_>(z) = \frac{1}{\Delta t^2}\left(\frac{z}{\Psi_>(z)} + 1 - z\right), \tag{37}$$

using that $\psi(0) = 1$. The one-sided z-transform of an arbitrary discrete function $f(n)$ is defined as $F_>(z) = \Sigma_{n=0}^{\infty} f(n) z^{-n}$. As shown in ref. 21, the AR model allows to express $\Psi_>(z)$ as

$$\Psi_>^{(AR)}(z) = \sum_{n=0}^{\infty} \psi^{(AR)}(n) z^{-n} = \sum_{j=1}^{P} \beta_j \frac{z}{z - z_j}, \tag{38}$$

where the coefficients β_j are given by eq. (33), and the roots z_j must fulfill the stability criterion (34).

Inserting (38) into (37) yields $\Xi_>^{(AR)}(z)$, the z-transform of the discrete memory function within the AR model,

$$\Xi_>^{(AR)}(z) = \frac{1}{\Delta t^2}\left(\frac{z}{\Psi_>^{(AR)}(z)} + 1 - z\right). \tag{39}$$

In *n*Moldyn the time series $\xi^{(AR)}(n)$ is obtained from $\Xi_>^{(AR)}(z)$ by polynomial division. Writing (39) in the form $\Xi_>(z) = c_0 + c_1 z^{-1} + c_2 z^{-2} + \ldots$ one can identify $c_0 \equiv \xi(0)$, $c_1 \equiv \xi(1)$, ...

Friction Coefficient within the AR Model

The friction coefficient is defined as the integral over the memory function. In the discrete case we write

$$\xi_0 := \sum_{n=0}^{\infty} \Delta t \xi(n) = \Delta t \Xi_>(1). \tag{40}$$

Using (39), we thus obtain within the AR model

$$\xi_0^{(AR)} = \frac{1}{\Delta t} \frac{1}{\sum_{j=1}^{P} \beta_j \frac{1}{1 - z_j}}. \quad (41)$$

This shows that ξ_0 can be obtained from the zeros z_j of the characteristic polynomial $p(z)$, defined in (31).

Coherent Scattering Memory Function

Another memory function that can be calculated by *n*Moldyn is the memory function related to the coherent intermediate scattering function. It is defined through the corresponding memory function equation

$$\partial_t \mathcal{F}(\mathbf{q}, t) = -\int_0^t d\tau \xi(\mathbf{q}, t - \tau)\mathcal{F}(\mathbf{q}, \tau). \quad (42)$$

The memory function $\xi(\mathbf{q}, t)$, which depends on q as well as on time, permits the analysis of memory effects on different length scales. From a numerical point of view its calculation is completely analogous to the case of the velocity memory function, the discrete time signal being here

$$\sum_{\alpha=1}^{N} b_{\alpha,\mathrm{coh}} \exp[-i\mathbf{q} \cdot \mathbf{R}_\alpha(t)]. \quad (43)$$

For each q-vector one obtains a set of P complex coefficients $\{a_n\}$ for the AR model.

Rigid-Body Motions

In molecular systems it is often interesting to look at different motion types. In particular, the overall translation and rotation of whole molecules or parts of large molecules (e.g., side chains in a protein), as opposed to their internal motions, are of interest in many situations. To study this global motion, particular groups of atoms are approximated by a rigid body with six degrees of freedom—three translational and three rotational; the relative motions of the atoms within these groups are discarded. Such a separation of global and internal motions can be useful to explore their relative contributions to the scattering function or the density of states. Moreover, some quantities, which are described in this section, are defined only for rigid objects.

The first step in any rigid-body analysis is the construction of a rigid-body trajectory from the original trajectory. This rigid-body trajectory is constructed such that its atomic trajectories stay as close as possible to the original atomic coordinates, subject to the rigid-body constraint. "As close as possible" is to be understood in the least-squares sense: the root-mean-square deviation of the rigid-body trajectory from the original trajectory is smaller than for any other possible rigid-body trajectory.

The rigid-body trajectory is constructed as follows: at each time step, find the orthogonal transformation (translation plus rotation) that produces the optimal superposition (in the least-squares sense) with a reference structure. This reference structure can simply be the initial configuration in the trajectory, or an externally defined "standard" configuration. The inverse of this optimal orthogonal transformation is then applied to the reference structure, and the resulting atomic positions replace the original ones in the trajectory. A detailed description and an efficient algorithm for finding the optimal orthogonal transformation can be found in ref. 27.

For calculating the quantities discussed in this section, the rigid-body motions must be described in a different way, namely by the center-of-mass trajectory and some suitable set of variables that describes the orientation of a rigid body. The most convenient set for numerical computations consists of the four components of a normalized quaternion, which describes the rotation relative to some arbitrary reference structure.[28] From such a quaternion trajectory, other quantities related to rotation, such as angular velocities, can be derived easily.

The *angular velocity autocorrelation function* is the rotational analogue of the VACF described earlier. It is given by

$$C_{\omega\omega}(t) = \sum_i \langle \omega_i'(0)\omega_i'(t)\rangle, \quad (44)$$

where $\omega_i'(t)$ is the angular velocity of rigid body i in the body-fixed coordinate system and related to the quaternions and their derivatives by

$$\begin{pmatrix} 0 \\ \omega_x' \\ \omega_y' \\ \omega_z' \end{pmatrix} = 2 \cdot \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{pmatrix} \begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix}. \quad (45)$$

The *reorientational correlation function* is defined in terms of the conditional probability $p(\mathbf{\Omega}_1, t_1|\mathbf{\Omega}_0, t_0)$ for a molecule to have orientation $\mathbf{\Omega}_1$ at time t_1 given that it had orientation $\mathbf{\Omega}_0$ at time t_0. In an isotropic equilibrium system, it depends only on the time difference $t = t_1 - t_0$ and on the change in orientation $\mathbf{\Omega}$. It can then be written as $p(\mathbf{\Omega}, t|\mathbf{0}, 0)$. To describe the orientation dependence, it is convenient to expand the reorientational correlation function in Wigner rotation matrices,[29] which form a complete set of basis functions in Ω:[30]

$$p(\mathbf{\Omega}, t|\mathbf{0}, 0) = \sum_{jmn} \frac{2j + 1}{8\pi^2} p_{mn}^j(t) D_{mn}^{*j}(\mathbf{\Omega}). \quad (46)$$

The expansion coefficients $p_{mn}^j(t)$ can be calculated via

$$p_{mn}^j(t) = 4\pi\langle Y_m^j[\mathbf{q}(t)]Y_n^{*j}[\mathbf{q}(0)]\rangle, \quad (47)$$

where $Y_m^j[\mathbf{q}(t)]$ are the spherical harmonics expressed in terms of quaternions.[31] Equation (47) shows that the expansion coefficients $p_{mn}^j(t)$ are effectively time correlation functions of spherical harmonics. Some of these coefficients can be measured by suitable experiments, for example, $p_{00}^1(t)$ by infrared spectroscopy (dipole–dipole correlation function) and $p_{00}^2(t)$ by relaxation NMR experiments. The numerical evaluation in *n*Moldyn is based directly on eq. (47).

Efficient Calculation of Time Correlation Functions

Correlation functions of discrete time series can be calculated efficiently using the Fast Fourier Transform (FFT).[32] The so-called Fast Correlation Algorithm (FCA) allows reduction of the complexity (measured by the number of multiplications) from $\propto N_t^2$ to $\propto N_t \log_2(N_t)$. In *n*Moldyn all time correlation functions are computed using the FCA method, which will be outlined in the following.

We consider two time series

$$a(k) = a(k \cdot \Delta t), \quad b(k) = b(k \cdot \Delta t), \quad k = 0 \ldots N_t - 1, \quad (48)$$

of length $T = (N_t - 1) \cdot \Delta t$, which are to be correlated. In the following, the shorthands $a(k)$ and $b(k)$ will be used. The discrete correlation function of $a(k)$ and $b(k)$ is defined as

$$c_{ab}(m) = \begin{cases} \dfrac{1}{N_t - m} \displaystyle\sum_{k=0}^{N_t-m-1} a^*(k) b(k+m), & m = 0 \ldots N_t - 1, \\ \dfrac{1}{N_t - |m|} \displaystyle\sum_{k=|m|}^{N_t-1} a^*(k) b(k - |m|), & m = -(N_t - 1) \ldots -1. \end{cases} \quad (49)$$

The prefactors in front of the sums ensure the proper normalization of the individual channels, $m = -(N_t - 1) \ldots N_t - 1$. The asterisk denotes a complex conjugate. According to (49), $c_{ab}(m)$ has $2N_t - 1$ data points and obeys the symmetry relation

$$c_{ab}(m) = c_{ba}^*(-m). \quad (50)$$

In case that $a(k)$ and $b(k)$ are identical, the corresponding correlation function $c_{aa}(m)$ is called an *auto*correlation function. Now we define the extended, periodic time series

$$A(k) = \begin{cases} a(k) & k = 0 \ldots N_t - 1 \\ 0 & k = N_t \ldots 2N_t - 1 \end{cases}, \quad (51)$$

$$B(k) = \begin{cases} b(k) & k = 0 \ldots N_t - 1 \\ 0 & k = N_t \ldots 2N_t - 1 \end{cases}, \quad (52)$$

which have the period $2N_t$,

$$A(k) = A(k + m \cdot 2N_t), \qquad B(k) = B(k + m \cdot 2N_t), \qquad m = 0, \pm 1, \pm 2, \ldots. \quad (53)$$

The discrete, cyclic correlation of $A(k)$ and $B(k)$ is defined as

$$S_{AB}(m) = \sum_{k=0}^{2N_t-1} A^*(k) B(k+m). \quad (54)$$

It is easy to see that

$$c_{ab}(m) = \frac{1}{N_t - |m|} S_{AB}(m), \qquad -(N_t - 1) \le m \le N_t - 1. \quad (55)$$

Using the correlation theorem of discrete periodic functions,[32] $S_{AB}(m)$ can be written as

$$S_{AB}(m) = \frac{1}{2N_t} \sum_{n=0}^{2N_t-1} \exp\left[2\pi i \left(\frac{mn}{2N_t}\right)\right] \tilde{A}^*\left(\frac{n}{2N_t}\right) \tilde{B}\left(\frac{n}{2N_t}\right) \quad (56)$$

where $\tilde{A}(n/2N_t)$ and $\tilde{B}(n/2N_t)$ are the discrete Fourier transforms of $A(k)$ and $B(k)$, respectively:

$$\tilde{A}\left(\frac{n}{2N_t}\right) = \sum_{k=0}^{2N_t-1} \exp\left[-2\pi i \left(\frac{nk}{2N_t}\right)\right] A(k), \quad (57)$$

$$\tilde{B}\left(\frac{n}{2N_t}\right) = \sum_{k=0}^{2N_t-1} \exp\left[-2\pi i \left(\frac{nk}{2N_t}\right)\right] B(k). \quad (58)$$

If the Fourier transforms of the signals $A(k)$ and $B(k)$ as well as the inverse transform in (56) are computed by FFT, $S_{AB}(m)$ can be computed by $\propto N_t \log_2(N_t)$ instead of $\propto N_t^2$ multiplications. It is sometimes said that the FFT method induces spurious correlations. We emphasize that this is only the case if the time series $a(k)$ and $b(k)$ are not properly extended, as indicated in eqs. (51) and (52). The FFT method and the direct scheme (49) give, apart from round-off errors, *identical results.*

Numerical Computation of Frequency Spectra

Both the dynamic structure factor and the density of states, which have been discussed earlier, are obtained numerically by a Fourier transform of the corresponding discrete time correlation function. To obtain a good estimate for the spectrum, the latter can be smoothed by applying a window in the time domain:[22]

$$\tilde{C}\left(\frac{n}{2N_t}\right) = \Delta t \cdot \sum_{m=-(N_t-1)}^{N_t-1} \exp\left[-2\pi i \left(\frac{nm}{2N_t}\right)\right] W(m) \frac{1}{N - |m|} C(m), \quad (59)$$

Here, $C(m)$ is the respective time correlation function and $\tilde{c}(n/2N_t)$ the corresponding spectrum. The time step Δt in front of the sum yields the proper normalization of the spectrum. In *n*Moldyn a Gaussian window[33] is used:

$$W(m) = \exp\left[-\frac{1}{2}\left(\alpha \frac{|m|}{N_t - 1}\right)^2\right], m = -(N_t - 1) \ldots N_t - 1. \quad (60)$$

Its widths in the time and frequency domains are $\sigma_t = T/\alpha$ and $\sigma_v = 1/\sigma_t$, respectively. We recall that $T = (N_t - 1) \cdot \Delta t$ is the length of the simulation.

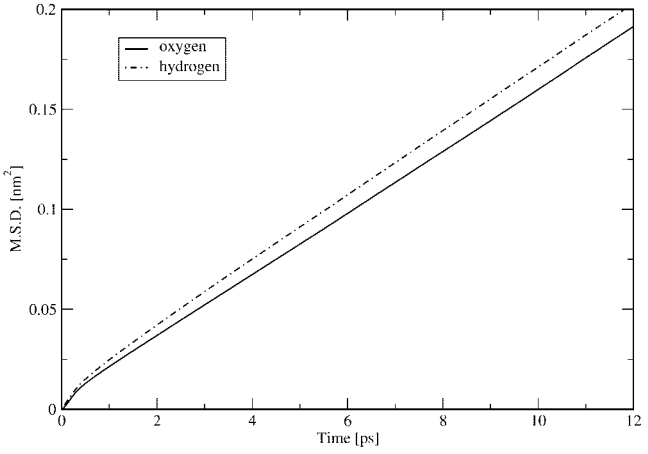

Figure 1. The mean-square displacement [MSD, eq. (10)] of oxygen and hydrogen atoms in water. The higher values for the hydrogens are due to the rotation of the molecules.

Using *n*Moldyn

Graphical User Interface

Ease of use is as important for an analysis tool as is computational efficiency, especially because many users cannot be expected to be competent programmers. *n*Moldyn provides two user interfaces to accommodate a wide range of users. The first one is a graphical user interface that can be used with little technical experience. The user first opens an input trajectory, then specifies the parameters for the calculation he wishes to perform, and finally starts the calculation itself. He can then check the status of running calculations and inspect their results.

The graphical user interface gives access to most of the functionality of *n*Moldyn. The only limitations concern the selection of the part(s) of the full system that will be used in a particular calculation. Using the graphical user interface, only the most common selections can be made: small molecules can be selected collectively (e.g., all water molecules), and macromolecules can be selected individually, allowing in addition the restriction to certain subparts (e.g., only the backbone). The same options exist for specifying which parts of the system should be deuterated. Alternatively, a detailed selection can be achieved by marking the atoms to be used in a PDB file.

With the graphical user interface, it is also possible to generate input files for the command line interface described in the next section. Such an input file provides a convenient starting point for customizations.

Command-Line Interface

In some situations a graphical user interface cannot be used for technical reasons (e.g., text-mode connections to remote machines), and in others a graphical user interface is not the most convenient solution, for example, when a large number of similar calculations are to be performed. For these situations, *n*Moldyn provides a command-line interface that reads all input information from a single specification file. For example, the specification file for a simple mean-square displacement calculation looks like the following:

```
from MMTK import *
trajectory = ["lysozyme.nc"]
output_files = {"msd": "msd.plot"}
title = "Mean Square Displacement"
time_info = (0, None, 1)
weights = "mass"
atoms = {'Protein.0': ['*']}
```

However, the command-line interface provides additional flexibility for advanced users. The specification files are in fact Python scripts, which means that competent users have the full power of Python and its libraries, specifically the Molecular Modelling Toolkit, at their disposal. The core *n*Moldyn functions are also accessible as a Python library, and can thus be used as building blocks in other programs, for example, for calculating different physical quantities.

The most common need for more flexibility is in the selection of the atoms and groups that are to be used for a given calculation. If the standard selections offered by the graphical user interface are not sufficient, the selection can usually be achieved by a few lines of Python code in the *n*Moldyn specification file. For example, the following file specifies the calculation of the mean-square displacement for only the sidechains of residues 5 to 25:

```
from MMTK import *
trajectory = ['lysozyme.nc']
output_files = {"msd": "msd.plot"}
title = "Mean Square Displacement"
time_info = (0, None, 1)
weights = "mass"

def atoms_code(trajectory):
  # retrieve all proteins
```

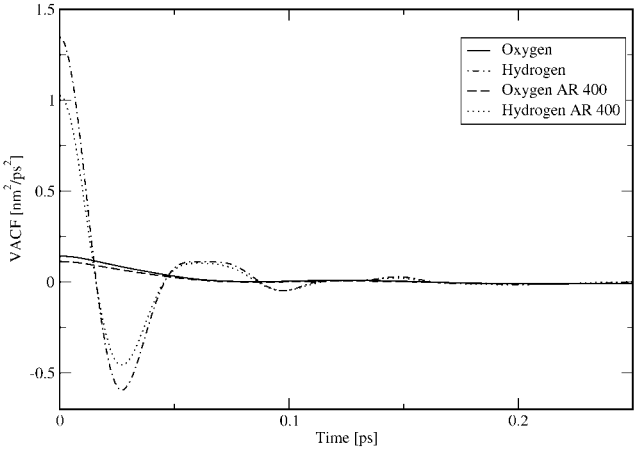

Figure 2. The velocity autocorrelation function [VACF, eq. (7)] of oxygen and hydrogen atoms in water, calculated directly and from an autoregressive model of order 400. The autoregressive model underestimates the fluctuation amplitude somewhat, but reproduces the shape of the VACF correctly.

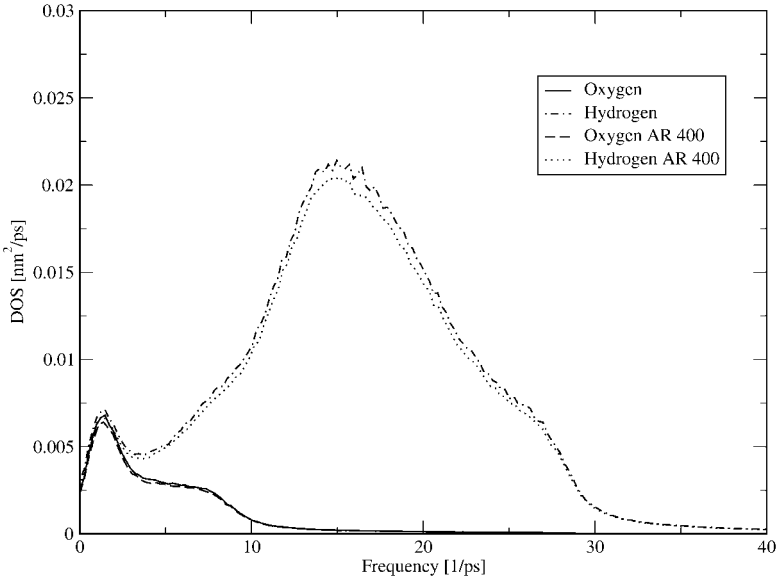

Figure 3. The density of states [DOS, eq. (23)] of oxygen and hydrogen atoms in water, calculated directly and from an autoregressive model of order 400. As in Figure 2, there is a good agreement between the two results.

```
from MMTK.Proteins import Protein
proteins = trajectory.universe.\
objectList(Protein)
# pick the lysozyme
lysozyme = proteins[0]
# pick a part of the first chain
subchain = lysozyme[0][4:25]
# select the sidechains
return subchain.sidechains()
```

File Formats

User-friendly design also extends to the data formats being used for input and output. *n*Moldyn expects trajectory files to follow the conventions of the Molecular Modelling Toolkit. This means that all trajectory data is stored in a netCDF file, which also contains the definition of the system. This format has several advantages: the files are platform-independent, compact, and self-contained, which reduces the risk of user errors. Moreover, it is possible to read trajectory data atom by atom efficiently, i.e., without reading in the whole file each time. When a trajectory is so long that it becomes inconvenient to store it in a single file, successive parts of it can be stored in several independent files, which *n*Moldyn can analyze as if it were a single long trajectory.

Because many users will have Molecular Dynamics trajectories in different formats, a format conversion is necessary. Converters for CHARMM and DLPOLY trajectories come with *n*Moldyn, and

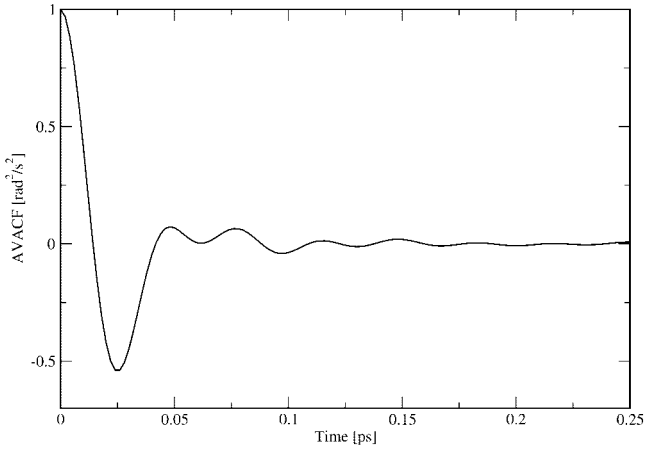

Figure 4. The angular velocity autocorrelation function [eq. (44)] of the water molecules.

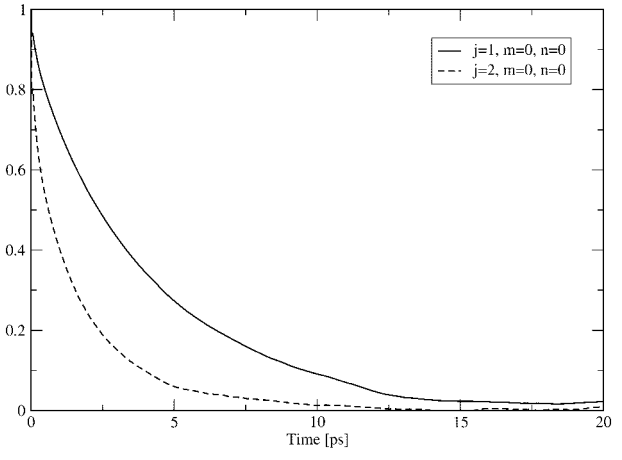

Figure 5. The expansion coefficients of the reorientational correlation function [eq. (47)] of the water molecules for $j = 1$ and $j = 2$.

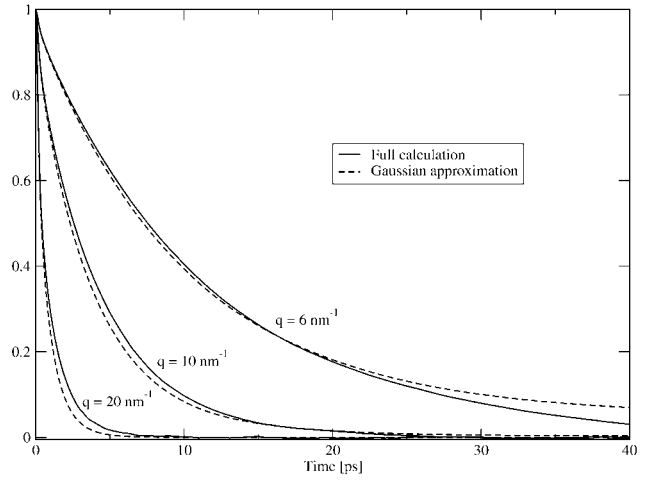

Figure 6. The incoherent intermediate scattering function $\mathcal{F}_{\mathrm{inc}}(\mathbf{q}, t)$ [eq. (17)], calculated directly and using the Gaussian approximation.

a very elaborate converter for AMBER trajectories is available separately.[34]

Some of the quantities calculated by *n*Moldyn are functions of a single variable (time, frequency, or wave number). These functions are written to a simple text file that can be read by most plotting programs. However, the scattering functions and the dynamical structure factors are functions of two variables. No standard format for such data exists, and the files can become rather large. Therefore, *n*Moldyn uses portable netCDF files for these quantities. *n*Moldyn also provides on-screen plotting facilities for these functions, with both 3D and 2D representations.

Examples

To demonstrate the usefulness of *n*Moldyn, we have calculated a selection of dynamic quantities for water. A Molecular Dynamics simulation of water using the SPCE model in the NPT ensemble at a temperature of 300 K and a pressure of 1 bar was performed using DLPOLY.[35] The rigid SPCE water molecules were simulated using a quaternion-based integrator and center-of-mass scaling for the barostat. Ewald summation was used for electrostatic interactions.

A time step of 1 fs was used to generate two trajectories, one of 4 ps with every second time step stored (for the calculation of fast quantities), and one with a total length of 100 ps in which every 10th step is stored (for the calculation of slow quantities). The two trajectories were converted to MMTK format using the converter that is part of the *n*Moldyn package. The results of the various analysis calculations are shown in Figures 1 to 8.

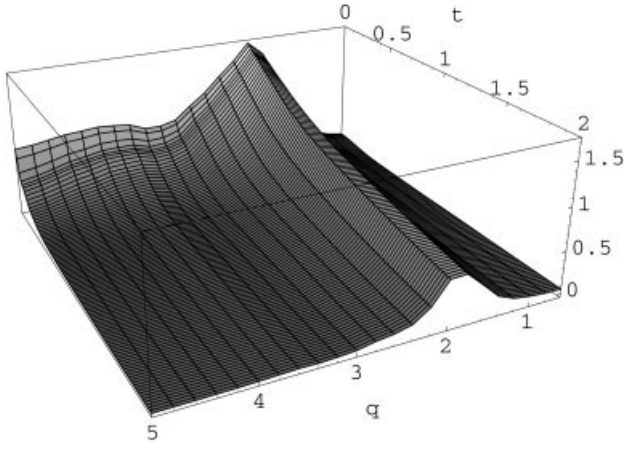

Figure 7. The coherent intermediate scattering function [eq. (16)] $\mathcal{F}_{\mathrm{coh}}(\mathbf{q}, t)$.

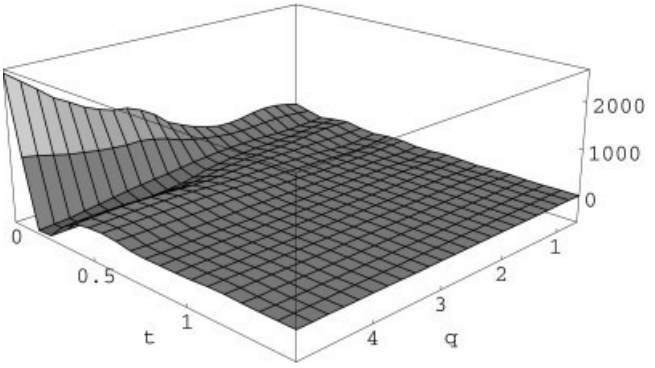

Figure 8. The memory function $\xi(\mathbf{q}, t)$ [eq. (42)] of the coherent intermediate scattering function.

Conclusion

The development of *n*Moldyn shows that it is possible to write flexible and user-friendly analysis programs for simulation data by profiting from modern software engineering methods and from high-quality Open Source libraries. We hope that the availability of *n*Moldyn will encourage researchers to analyze their Molecular Dynamics trajectories in more detail than they did in the past.

Future development plans for *n*Moldyn include parallelization and further optimization, as well as the implementation of additional analysis techniques.

References

1. Lovesey, S. Theory of Neutron Scattering from Condensed Matter; Clarendon Press: Oxford, 1984; vol. 1.
2. Bée, M. Quasielastic Neutron Scattering: Principles and Applications in Solid State Chemistry, Biology and Materials Science; Adam Hilger: Bristol, 1988.
3. Kneller, G. Mol Phys 1994, 83, 63.
4. Kneller, G.; Smith, J.; Cusack, S.; Doster, W. J Chem Phys 1992, 97, 8864.
5. Dianoux, A.; Kneller, G.; Sauvajol, J.; Smith, J. J Chem Phys 1993, 99, 5586.
6. Kneller, G.; Smith, J. J Mol Biol 1994, 242, 181.

7. Dianoux, A.; Kneller, G.; Sauvajol, J.; Smith, J. J Chem Phys 1994, 101, 634.
8. Morélon, N.-D.; Kneller, G.; Ferrand, M.; Grand, A.; Smith, J.; Bée, M. J Chem Phys 1998, 109, 2883.
9. Kneller, G.; Keiner, V.; Kneller, M.; Schiller, M. Comp Phys Commun 1995, 91, 191. Full description in report ILL95KN02T, Institut Laue–Langevin, 156 X, F-38042 Grenoble Cedex, France.
10. van Rossum, G., et al. The Python web site. http://www.python.org/.
11. Ascher, D.; Dubois, P.; Hinsen, K.; Huguin, J.; Oliphant, T. Numerical Python. Technical Report UCRL-MA-128569, Lawrence Livermoore National Laboratory, http://numpy.sourceforge.net.
12. The FFTW library. Technical report. http://fftw.org.net.
13. Rew, R.; Davis, G.; Emmerson, S.; Davies, H. Netcdf User's Guide for c, an Interface for Data Access, version 3. http://www.unidata.ucar.edu/packages/netcdf/guidec, (1997).
14. Hinsen, K. Scientific Python. http://dirac.cnrs-orleans.fr.
15. Hinsen, K. J Comp Chem 2000, 21, 79.
16. OpenSource web site. http://www.opensource.org, (1997).
17. Higgins, J.; Benoît, H. Polymers and Neutron scattering; Oxford University Press: New York, 1994.
18. McQuarrie, D. Statistical Mechanics; Harper's Chemistry Series; Harper Collins Publishers: New York, 1976.
19. Rahman, A.; Singwi, K.; Sjolander, A. Phys Rev 1961, 126, 986.
20. Boon, J.; Yip, S. Molecular Hydrodynamics; McGraw Hill: New York, 1980.
21. Kneller, G. R.; Hinsen, K. J Chem Phys 2001, 115, 11097.
22. Papoulis, A. Probability, Random Variables, and Stochastic Processes, McGraw Hill: New York, 1991; 3rd ed.
23. Makhoul, J. Proc IEEE 1975, 63, 561.
24. Burg, J. Maximum Entropy Spectral Analysis. PhD thesis, Stanford University, Stanford, CA, 1975.
25. Makhoul, J. IEEE Transactions on Acoustics, Speech, and Signal Processing. ASSP-25, 1977, 423.
26. GNU Multiple Precision Arithmetic Library http://www.swox.com/gmp/.
27. Kneller, G. Mol Sim 1991, 7, 113.
28. Altmann, B. Rotations, Quaternions, and Double Groups; Clarendon Press: Oxford, 1986.
29. Lynden–Bell, R.; Stone, A. Mol Sim 1989, 3, 271.
30. Edmonds, A. Angular Momentum in Quantum Mechanics; Princeton University Press: Princeton, NJ, 1957.
31. Kneller, G. J Chim Phys 1991, 88, 2709.
32. Brigham, E. The Fast Fourier Transform; Prentice Hall: Englewood Cliffs, NJ, 1974.
33. Press, W.; Teukolsky, S.; Vetterling, W.; Flannery, B. Numerical Recipes in C; Cambridge University Press: Cambridge, MA, 1992; 2nd ed.
34. Murzyn, K. Amber trajectory converter for MMTK/netCDF format. http://www.mol.uj.edu.pl/murzyn/convert/convert.html.
35. Forester, T.; Smith, W. The DLPOLY User Manual, version 2.0. CCLRC, Daresbury Laboratory: Daresbury, Warrington, UK, 1955.