# LiquidLib

A comprehensive toolbox for analyzing classical and ab initio molecular dynamics simulations of liquids and liquid-like matter with applications to neutron scattering experiments

## 1 Introduction

*LiquidLib* is a open source package to compute the following quantities:

- Pair Distribution Function
- Structure Factor
- Mean Squared Displacement
- Non-Gaussian Parameter
- Four Point Correlation
- Velocity Auto Correlation
- Self van Hove Correlation
- Coherent van Hove Correlation
- Self Intermediate Scattering Function
- Coherent Intermediate Scattering Function
- Bond Order Parameter

## 2 Acknowledgments

*LiquidLib* is an open source code and is published under the GNU Lesser General Public License. The core contributors are Zhikun Cai, Abhishek Jaiswal, Nathan Walter, and Yang Zhang.

If you distribute a modified code or release any of the source code that includes *LiquidLib* source files then it must also be open sourced. Anyone is free to use modify or extend. This has been a joint effort by Z Lab at University of Illinois at Champaign-Urbana. More information can be found at `http://zhang.npre.illinois.edu/`. Contact information for any of the contributors can also be found at the former address. Any issues, bugs, or other related material can be sent to us via our contact information listed on the group webpage or on Github directly.

Please cite the following reference:

- N. P. Walter, A. Jaiswal, Z. Cai, and Y. Zhang, LiquidLib: A comprehensive toolbox for analyzing classical and ab initio molecular dynamics simulations of liquids and liquid-like matter with applications to neutron scattering experiments, Comput. Phys. Commun., vol. 228, pp. 209218, 2018. DOI

## 3 Installation

To make a quick and basic installation of *LiquidLib*, begin by opening a terminal window.

```
$ mv liquidlib.tar.gz /installation/directory/
$ cd /installation/directory/
$ tar -zxvf liquidlib.tar.gz
$ cd liquidlib
$ make
```

This will create a directory /liquidlib/bin/ with all of the executables contained within it.

To make a GROMACS compatible build, open Makefile and in the optional components section change "USE_XDRLIB" to "yes". This will add the appropriate flags for *LiquidLib* to be built to read .xtc and .trr trajectory files. Follow the instructions in Makefile for linking the xdrfile headers to make a build that can read trajectories that are dumped as .trr or .xtc binary files, such as GROMACS does, or .xtc for LAMMPS. In order to read these file types, LiquidLib relies on the open source library libxdrfile. The source code can be downloaded from the GROMACS website or the following link: `ftp://ftp.gromacs.org/pub/contrib/xdrfile-1.1.1.tar.gz` Newer versions of the xdfile library should be compatible with LiquidLib, however, we recommend 1.1.1 as this is the version used for testing purposes.

Once libxdrfile.a and xdrfile.h, xdrfile_trr.h, and xdrfile_xtc.h have been installed, open the *LiquidLib* Makefile, and edit the lines contained in the comment section for the GROMACS build. For compiling, the libxdrfile.a and xdrfile.h, xdrfile_trr.h, and xdrfile_xtc.h headers can be placed inside of the *LiquidLib* include directory, or you can specify the location inside of the Makefile. Recompile LiquidLib and the binaries should be compatible with GROMACS.

To build parallelized OpenMP support, open Makefile in a text editor and in the optional components section change "USE_OMP" to "yes. This will add the appropriate flags for *LiquidLib* to be built with OpenMP support included. This works for gcc and g++. If you use a different c complier, please also make sure the appropriate complier flags are added to CFLAGS. Recompile the program thereafter.

In order for the bond order parameter calculation to be performed, *LiquidLib* requires either Boost or GSL to be installed. Once one of these has been installed, in the makefile switch either USE_BOOST or USE_GSL to "yes," if both are set to yes, then GSL will be used with priority.

All user options for the Makefile are listed at the beginning of the Makefile.

## 4 Description of Quantities

This section briefly describes the theoretical background of neutron scattering lengths, the structural quantities, and the dynamical quantities that can be computed by LiquidLib. For more complete descriptions of the quantities, please refer to [1, 2]. The quantities can be weighted by the appropriate scattering lengths (coherent or incoherent) if the user wants to compare to experimentally measurable quantities.

List of common variables relevant to all quantities:

- $d$ : the dimension of the system; can be any positive integer
- $\mathbf{r}$ : the $d$-dimensional position vector of a particle
- $r$ : the scalar distance between two particles in space, or the distance of the same particle between two different times
- $\rho$ : the total number density of the system
- $t$: the time
- $N$ : the number of particles in the system
- $b_{coh}$ : the coherent neutron scattering length
- $b_{inc}$ : the incoherent neutron scattering length
- $\langle\cdot\rangle$: the ensemble average, represents averaging over multiple trajectory frames and time origins
- $L$ : the size of the simulation box
- $\mathbf{k}$ : the $d$-dimensional reciprocal space vector or wave vector transfer; can only take values of $\frac{2\pi}{L}(n_1, n_2, ..., n_d)$, where $n_i$ is a non-negative integer
- $V$ : the volume of the system
- $T$ : the temperature of the system
- $\delta(\cdot)$ : the Dirac delta function

### 4.1 Neutron Scattering Lengths

The interaction between a neutron (in a wide energy range) and a nucleus can be quantified by the scattering length $b$, which depends on the species of the nucleus and the spin state of the nucleus. Here, the magnetic scattering is ignored. When the interference of scattered neutrons from multiple nuclei is considered, the total double differential cross section can be readily separated into two parts: the coherent scattering and incoherent scattering. The coherent scattering length $b_{coh}$ describes the average of the spin-dependent scattering lengths. The incoherent scattering length $b_{inc}$ describes their standard deviation [3]. Therefore, LiquidLib computations can weigh the relevant quantities by the appropriate neutron scattering lengths in order to directly compare LiquidLib results to neutron scattering experiments. Unweighted calculations can be performed in LiquidLib by omitting the scattering lengths or setting the scattering lengths to unity. For more information on the theory of neutron scattering and scattering lengths, please refer to [1, 2, 3].

### 4.2 Pair Distribution Function: $g(r)$

A widely used quantity to describe the microscopic structure of a system is the Pair Distribution Function (PDF) or Radial Distribution Function (RDF) $g(r)$. $g(r)$ measures the probability of finding a particle a distance $r$ from the reference particle. From this quantity, the average distance of the nearest neighbors, second nearest neighbors, the coordination number (number of nearest neighbors), etc. can be directly determined. For crystalline materials, $g(r)$ shows sharp peaks corresponding to the periodic arrangements of particles. For isotropic systems such as liquids, $g(r)$ only depends on the modulus of relative atomic distance $r$ and can be computed as follows:

$$g(r) = \frac{1}{4\pi r^2 dr}\frac{1}{\rho N \langle b_{coh}\rangle^2}\left\langle \sum_{l\neq l'} b_l b_{l'} \delta(r - |\mathbf{r}_l - \mathbf{r}_{l'}|)\right\rangle \tag{1}$$

LiquidLib can compute both the partial $g(r)$ between two individual types of atoms (e.g. oxygen to oxygen in water), and the total $g(r)$ of a selected group of atoms. The non-weighted $g(r)$ can be readily obtained by setting the neutron coherent scattering lengths $b_l$ and $b_{l'}$ to unity. The total $g(r)$ can also be obtained by the summation of the partial $g(r)$ weighted by the respective atomic number densities and the respective coherent neutron scattering lengths, if applicable.

### 4.3 Structure Factor: $S(k)$

The static structure factor, $S(k)$, is another widely used quantity to describe the microscopic structure of a system. Unlike the pair distribution function $g(r)$, which is a measure of the structure in real space, the structure factor $S(k)$ measures the structure of the system in the reciprocal space. $S(k)$ can be directly measured by elastic scattering (diffraction) experiments such as with neutrons, X-rays, and electrons. Once again, for isotropic systems such as liquids, $S(k)$ only depends on the modulus of the wave vector transfer $k$. $S(k)$ can be computed directly from its definition:

$$S(k) = \frac{1}{N\langle b_{coh}\rangle^2}\left\langle \sum_{l,l'=1}^{N} b_l b_{l'} \exp\left\{-i\mathbf{k}\cdot[\mathbf{r}_l - \mathbf{r}_{l'}]\right\}\right\rangle \tag{2}$$

Here, $S(k)$ is weighted by the proper coherent neutron scattering lengths $b_l$ and $b_{l'}$, thus it can be directly compared with a neutron diffraction experiment.

A computationally efficient representation of $S(k)$ is shown in Eq. 3 by expanding the exponential in Eq. 2:

$$S(k) = \frac{1}{N\langle b_{coh}\rangle^2}\left\langle \left[\sum_{l=1}^{N} b_l \cos(\mathbf{k}\cdot\mathbf{r}_l)\right]^2 + \left[\sum_{l=1}^{N} b_l \sin(\mathbf{k}\cdot\mathbf{r}_l)\right]^2\right\rangle \tag{3}$$

The computational complexity is reduced to $\mathcal{O}(N)$ by using Eq. (3) in performing the computation of $S(k)$. The sampled directions are determined by the implementation of periodic boundary conditions of the simulation. Furthermore, in LiquidLib, the structure factor $S(k)$ can be computed for individual atom types (e.g. oxygen in water) or for any selected atom types in any selected groups (e.g. all water molecules). This is especially useful to identify the origin of features in the experimentally measured structure factor.

An alternative way to compute $S(k)$ is through the Fourier transform of $g(r)$. To be more precise,

$$S(k) = 1 + 4\pi\rho \int_0^{\infty} (g(r) - 1)\frac{\sin(kr)}{kr} r^2 dr \tag{4}$$

Therefore, $S(k)$ and $g(r)$ can be obtained from one another by Fourier transform. Under certain conditions, the Fast Fourier Transform (FFT) algorithm can be applied.

## 4.4 Bond Order Parameter: $q_l(i,t)$, $\bar{q}_l(i,t)$, $Q_l(t)$

The bond orientational order parameter is a quantity that measures, as its name indicates, the local bond orientational order in a system [4, 5]. The bond orientational parameter cannot be directly compared to scattering experiments. However, it is very useful for determining the microscopic structural order and phases in a system. For example, the bond orientational order parameter can be used to determine if a system is BCC or FCC or amorphous. Further, this quantity measures local structural order for each atom, whereas, the structure factor and the pair distribution function measure the structure of the complete system.

We use $q_l$ for the bond orientational order parameter of individual atoms in the system, and $Q_l$ for the system-wide bond orientational order parameter. $\bar{q}_l$ is equivalent to $q_l$ averaged over an atom's nearest neighbors. Thus, $\bar{q}_l$ has the benefit of reducing the overlapping $q_l$ values between liquids and crystals [5]. Hence, $Q_l$ is equivalent to $q_l$ averaged over all the atoms in the system [4]. The bond orientational order parameter of individual atoms $q_l$ is defined as

$$q_l(i,t) = \sqrt{\frac{4\pi}{2l+1}\sum_{m=-l}^{l} |q_{lm}(i,t)^2|} \tag{5}$$

where

$$q_{lm}(i,t) = \frac{1}{N(i)}\sum_{j=1}^{N(i)} Y_{lm}(\hat{\mathbf{r}}_{ij}(t)) \tag{6}$$

$m$ ranges from $-l$ to $l$; $l$ is the order of the bond orientational order parameter, which has a typical value 4 and 6, and should not be confused with the dummy summation index $l$ used previously; $i$ is the index of an atom; $N(i)$ is the number of nearest neighbors around atom $i$; $Y_{lm}$ is the spherical harmonic function of order $l$ and $m$; and $\hat{\mathbf{r}}_{ij}$ is the unit vector from atom $i$ to atom $j$, which can be converted to spherical coordinates.

The nearest neighbor averaged bond orientational order parameter $\bar{q}_l$ is defined as

$$\bar{q}_l(i,t) = \sqrt{\frac{4\pi}{2l+1}\sum_{m=-l}^{l} |\bar{q}_{lm}(i,t)^2|} \tag{7}$$

where

$$\bar{q}_{lm}(i,t) = \frac{1}{N(i)}\sum_{j=0}^{N(i)} q_{lm}(j,t) \tag{8}$$

The system-wide bond orientational order parameter $Q_l$ is defined as

$$Q_l(t) = \sqrt{\frac{4\pi}{2l+1}\sum_{m=-l}^{l} |Q_{lm}(t)^2|} \tag{9}$$

where

$$Q_{lm}(t) = \frac{\sum_{i=1}^{N} N(i) q_{lm}(i,t)}{\sum_{i=1}^{N} N(i)} \tag{10}$$

Lastly,

$$\bar{Q}_l(t) = Q_l(t) \tag{11}$$

because the averaging is taken over the entire system.

## 4.5 Mean Squared Displacement: $\langle r^2(t)\rangle$

A commonly used dynamical quantity is the Mean Squared Displacement (MSD), which measures on average how far a particle moves in the system as a function of time:

$$\langle r^2(t)\rangle = \frac{1}{N}\left\langle \sum_{l=1}^{N} |\mathbf{r}_l(t) - \mathbf{r}_l(0)|^2 \right\rangle \tag{12}$$

It is related to the second moment of the self van Hove correlation function, which will be discussed later in the paper. The implementation of MSD involves averaging of displacements of all particles of interest at two time points. In LiquidLib, the MSD, like many other quantities computable by LiquidLib, can be computed for any selected atom types in any selected groups.

Typically for liquids, particles will show ballistic motions at very short times, and eventually undergo random walk diffusive motion. The self-diffusion coefficient $D$ can be extracted from the slope of the long time diffusive tail of the MSD using the Einstein equation:

$$\lim_{t\to\infty} \langle r^2(t)\rangle = 2dDt \tag{13}$$

## 4.6 Non-Gaussian Parameter: $\alpha_2(t)$

While the MSD is dependent on the second order of the particle locations, the non-Gaussian parameter, $\alpha_2(t)$, is dependent on the forth order of the particle locations. It is defined as follows:

$$\alpha_2(t) = \frac{3\left\langle \sum_{l=1}^{N} [\mathbf{r}_l(t) - \mathbf{r}_l(0)]^4 \right\rangle}{5\left\langle \sum_{l=1}^{N} [\mathbf{r}_l(t) - \mathbf{r}_l(0)]^2 \right\rangle^2} - 1 \tag{14}$$

The non-Gaussian parameter measures the deviation from the Gaussian behavior of the motion of particles. For ballistic motion and Brownian diffusion, the distribution of particle displacements, $P(\Delta\mathbf{r}(t)) = |\mathbf{r}_l(t) - \mathbf{r}_l(0)|$, is Gaussian for a given time, t. Thus, $\alpha_2(t)$ is zero for Brownian diffusion and ballistic motions. However, at intermediate times, the distribution of particle displacements become non-Gaussian as clusters of mobile and immobile particles develops in the system. At these times, the non-Gaussian parameter, $\alpha_2(t)$, is greater than zero. Thus, $\alpha_2(t)$ is used to measure the dynamic heterogeneity of a system [6]. In LiquidLib, $\alpha_2(t)$ can be computed for any selected atom types in any selected groups.

## 4.7 Four Point Correlation: $\chi_4(t)$

Continuing from the non-Gaussian parameter, the four point density correlation function $\chi_4(t)$ is another higher order quantity commonly used to quantify the dynamic heterogeneity [7]. Direct quantification of a length scale characterizing correlated dynamics involves correlations of more than two points. $chi_4(t)$ measures the dynamic heterogeneity by computing the susceptibility of a system to fourth-order density fluctuations or density-density correlations. Equivalently, $\chi_4(t)$ is defined as:

$$\chi_4(t) = \frac{V}{T}\int G_4(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, t)d\mathbf{r}_1 d\mathbf{r}_2 d\mathbf{r}_3 d\mathbf{r}_4 \tag{15}$$

where, $G_4(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, t)$ is a four-point, time-dependent, density-density correlation function. For more information on the physical meaning of $chi_4$, we refer readers to [7] and [8].

We adopt the "coarse graining" approach [7] by using an overlap function, $w(|r_1 - r_2)$, to measure the overlap between configurations at two times. The overlap function is defined as unity for $|r_1 - r_2| \le a$ and 0 otherwise. The distance parameter $a$ is chosen to be larger than the square root of the plateau value of $\langle r^2(t)\rangle$ to filter out vibrational contributions. Consequently, the four point correlation function $\chi_4(t)$ is defined as:

$$\chi_4(t) = \frac{V}{T}[\langle Q_s^2(t)\rangle - \langle Q_s(t)\rangle^2] \tag{16}$$

where

$$Q_s(t) = \frac{1}{N}\sum_{l=1}^{N} w(|\mathbf{r}_l(t) - \mathbf{r}_l(0)|) \tag{17}$$

Contribution to $\chi_4(t)$ arises from both localized particles characterizing self correlations and non-localized particles characterizing distinct correlations. However, the distinct correlations have been shown to be much smaller and thus $\chi_4(t)$ is essentially dominated by the self correlations. Therefore, only the self correlations have been assessed in Eq. 16 and 17.

## 4.8 Velocity Auto Correlation: $C_{vv}(t)$

The mean squared displacement, non-Gaussian parameter, and the four point correlation function are measures of the dynamics based on the positions of atoms, but the Velocity Auto Correlation Function (VACF) is a measure of the dynamics based on the velocities of atoms. The velocity auto correlation function is a self correlation between the velocities of the same particle at two times [1, 2]:

$$C_{vv}(t) = \frac{1}{\langle b_{inc}^2\rangle}\langle b_i\mathbf{v}_i(0)\cdot b_i\mathbf{v}_i(t)\rangle \tag{18}$$

It can be used to determine the self-diffusion coefficient via the Green-Kubo relation:

$$D = \frac{1}{d}\int_0^{\infty} C_{vv}(t)dt \tag{19}$$

and the phonon Density Of States (DOS) via the Fourier transform:

$$g(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} C_{vv}(t)e^{-i\omega t}dt \tag{20}$$

## 4.9 Self van Hove Correlation: $G_s(r,t)$

Single particle's motion can be studied using the self-part of the van Hove correlation function. In the liquid state, the system is isotropic and hence this quantity depends only on the magnitude of displacements, but not their directions. The physical meaning of $G_s(r,t)$ is the volumetric probability density of finding a particle $i$ in the vicinity of some distance $r$ at some other time $t$ given that it was at the origin at time $t = 0$ [1, 2].

$$G_s(r,t) = \frac{1}{4\pi r^2 dr}\frac{1}{N\langle b_{inc}^2\rangle}\left\langle\sum_{l=1}^{N} b_l^2\delta(\mathbf{r} + \mathbf{r}_l(0) - \mathbf{r}_l(t))\right\rangle \tag{21}$$

## 4.10 Coherent van Hove Correlation: $G(r,t)$

The collective van Hove correlation function is a measure of the density-density correlation at two spatial points as a function of time. In the liquid state, the system is isotropic and hence this quantity depends only on the magnitude of the displacements, but not their directions. The physical meaning of this function is the probability density of finding a particle $l'$ in the vicinity of $\mathbf{r}$ at a given time $t$ knowing that another particle $l$ is in the vicinity of the origin at time $t = 0$. In the long time limit, the system loses memory of its initial configuration and this the correlation function becomes independent of the distance $\mathbf{r}$ [1, 2].

$$G(r,t) = \frac{1}{4\pi r^2 drN\langle b_{coh}\rangle^2}\left\langle\sum_{l=1}^{N}\sum_{l'=1}^{N} b_l b_{l'}\delta(\mathbf{r} + \mathbf{r}_l(0) - \mathbf{r}_{l'}(t))\right\rangle \tag{22}$$

### 4.11 Self Intermediate Scattering Function: $F_s(k,t)$

Self Intermediate Scattering Functions (SISF) is defined as the spatial Fourier transform of $G_s(r,t)$ to the reciprocal space. SISF characterizes the density fluctuations of the same particle in the system at time $t = 0$ and at another subsequent time $t$ [1, 2].

$$F_s(k,t) = \frac{1}{N\langle b_{inc}^2 \rangle} \left\langle \sum_{l=1}^{N} b_l^2 \exp\left\{-i\mathbf{k} \cdot [\mathbf{r}_l(t) - \mathbf{r}_l(0)]\right\} \right\rangle \tag{23}$$

The SISF and the following CISF can be measured directly by photon correlation spectroscopy or neutron spin echo technique, or indirectly by inelastic neutron scattering experiments. Oftentimes, either the SISF or CISF may dominate the scattering process depending on the relative contributions from coherent and incoherent scattering in the system. The incoherent dynamic structure factor, $S_s(k,E)$, which is directly measured by incoherent inelastic neutron scattering experiments, and the $F_s(k,t)$ are Fourier pairs:

$$S_s(k,E) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_s(k,t) e^{-iEt} dt \tag{24}$$

### 4.12 Coherent Intermediate Scattering Function: $F(k,t)$

Coherent Intermediate Scattering Functions (CISF) is defined as the spatial Fourier transform of $G(r,t)$ to the reciprocal space. CISF characterizes the density correlation between one particle at time $t = 0$ and another particle at subsequent time $t$ [1, 2].

$$F(k,t) = \frac{1}{N\langle b_{coh} \rangle^2} \left\langle \sum_{l=1}^{N} \sum_{l'=1}^{N} b_l b_{l'} \exp\left\{-i\mathbf{k} \cdot [\mathbf{r}_l(t) - \mathbf{r}_{l'}(0)]\right\} \right\rangle \tag{25}$$

The coherent dynamic structure factor, $S(k,E)$, which is directly measured by coherent inelastic neutron scattering experiments, and the $F(k,t)$ are Fourier pairs:

$$S(k,E) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(k,t) e^{-iEt} dt \tag{26}$$

## 5 Execution

*LiquidLib* provides several binaries, where each binary is designed to calculate one quantity. Each quantity requires an input file providing all essential and optional parameters. If no file is provided, *LiquidLib* searches the current directory for a default named input file. Similarly, if no output path is provided, *LiquidLib* uses a default path. The following list summarizes the default input (.in) and output (.txt) file names.

- **Pair Distribution Function:**
  g_r.in
  g_r.txt
- **Structure Factor:**
  S_k.in
  S_k.txt
- **Mean Squared Displacement:**
  r2_t.in
  r2_t.txt
- **Non-Gaussian Parameter:**
  alpha2_t.in
  alpha2_t.txt
- **Four Point Correlation:**
  chi4_t.in
  chi4_t.txt
- **Velocity Auto Correlation:**
  vacf_t.in
  vacf_t.txt
- **Self van Hove Correlation:**
  Gs_rt.in
  Gs_rt.txt
- **Coherent van Hove Correlation:**
  G_rt.in
  G_rt.txt

- **Self Intermediate Scattering Function:**
  Fs_kt.in
  Fs_kt.txt
- **Coherent Intermediate Scattering Function:**
  F_kt.in
  F_kt.txt
- **Bond Order Parameter:**
  BOP.in
  BOP.txt

While these are the default files that will be used by *LiquidLib*, the file names used are ultimately up to the user. The user may use any input and output name they please. To use a non-default input name, the file used must be specified during command line operation, and to use a non-default output name, the file used must be specified in the input file under the option outputfilename. The command line operation to compute a quantity (for example MeanSquaredDisplacement) is as follows:

```
$ cd /execution/directory/
$ /installation/directory/bin/computeMeanSquaredDisplacement −i user_msd_file.in
```

where "-i user_msd_file.in" is not needed if the input file is the default *r2_t.in*. If execution is successful, the program will print "Successfully computed Mean Squared Displacement". If execution is not successful, various WARNINGS and ERRORS will be printed to screen. The user should refer to the Common Errors section for solutions to the problem. For the options that may be used in the input file please refer to the Input Parameters section below.

## 6 Input Parameters

This section of the manual describes the potential options that can be used in the input files. The first subsection Trajectory Parameters shows the options that are universal to all of the codes, and the following sections show options that are special to that quantity. Only options listed as *mandatory* are required for operation of any given quantity. All other inputs are optional since a default value is provided, however, its is recommended that the user set those values to their needs.

### 6.1 Trajectory Parameters

#
: Marks a comment in the input file. Entire lines or ends of lines may be comments.

=
: Can be used after a parameter name for aesthetic purposes. A trailing space must be used after a = in order for the values to be correctly determined from the input file.

**is_run_mode_verbose**
- [default: false, optional] false, no, or 0 informs the program to run in a concise mode.
- true, yes, or a non-zero number informs the program to run in a verbose mode and keep flushing the progresses of reading trajectory and computation. One can also turn on the verbose mode by adding a flag "-v" behind the execution command. The value in input file will overwrite this "-v" option, if they disagree.

**is_wrapped**
- [default: false, optional] false, no, or 0 informs the program that the trajectory is unwrapped.
- true, yes, or a non-zero number informs the program that the trajectory is wrapped in a periodic box and thus needs to be unwrapped in order to produce correct values for dynamic quantities. If set to true, LiquidLib will unwrap the coordinates before computation.

**start_frame**
- [default: 0, optional] This value determines the first frame number to read from the trajectory. If this option is set, *LiquidLib* will skip the frames prior to this number. This is useful when using an unequilibrated trajectory and the user wishes to skip the unequilibrated portion of the trajectory.

**end_frame**

- [default: 0, optional] This value determines when the last frame that will be read from the trajectory file. This parameter is optional and will be overwritten by other parameters such as start_frame, frame_interval, and number_of_frames_to_average during the computation. User must provide either end_frame or number_of_frames_to_average (preferred) in their scripts.

**dimension**

- [default: 3, optional] This value determines the dimension of the trajectory. LAMMPS can provide 2-d and 3-d trajectories. GROMACS and VASP can provide 3-d trajectories. However, *LiquidLib* is capable of calculating the quantities in any finite dimension when supplied with a such a trajectory, such as the trajectories created by our groups' any dimensional md package.

**trajectory_delta_time**

- [default: 1, optional] This value determines the $\Delta t$ between frames in the trajectory. For GROMACS, this value can be determined from the trajectory file. However, for other programs, such as LAMMPS, VASP, or other trajectory sources, this parameter is not determined automatically from the trajectory. Thus, for time dependent quantities, such as mean squared displacement or intermediate structure factor, the trajectory $\Delta t$ is required for proper output time_scale provided by user. If this parameter is not provided, the time step will be set to 1 (a.u.), or equivalently 1 step.

**output_file_name**

- [default: see here for defaults, optional] This value sets the name and location for the output data to be written to. Only condition on the location of outputfile is the directory must exist prior to execution as *LiquidLib* will not create the directory.

**trajectory_file_name**

- [**mandatory**] This option sets the name and location of the trajectory file. As of now, the only file types that are recognized for reading the trajectory are .trr, .xtc, .xyz, .dump, .atom, and XDATCAR. dump and atom extensions are for LAMMPS trajectories and XDATCAR is for VASP. .trr and .xtc can only be read if *LiquidLib* is built in GROMACS compatible mode.

**trajectory_file_type**

- [optional] This option can be set to "abc" in order to signal to the program that a ABC trajectory is being read in. ABC trajectories are produced by a module created in GROMACS by our research group. ABC is also only usable with bond order parameter, since there is no time in ABC trajectories.

**trajectory_data_type**

- [default: coordinate, optional] This sets the type of information that will be read from the trajectory file. For .trr files, this will change how the file is read. This needs to be set to velocity for VACF.

**gro_file_name**

- [**Conditionally Required in GROMACS mode**] This sets the name and location of the .gro file for GROMACS trajectories. This is needed to determine the atom types and groups in the system. If the trajectory is monoatomic or one wishes to compute a quantity for the entire system, this does not need to be provided. However, if the user wants to compute a quantity for a specific atom type or group from a GROMACS trajectory, this is required.

**atom_types**

- [default: empty, mandatory] This sets the type of atom that will be focused on for the computation. For example, if user wishes to compute the mean squared displacement of only hydrogen this can be set to H. The atom_type value should be compatible with the types used in trajectory file. Can be a letter for GROMACS, VASP, LAMMPS trajectories, or a number for LAMMPS trajectories. We now support multiple atom types, for example a user may request both Hydrogen and Oxygen by specifying HW OW

**scattering_lengths**

- [default: 1, optional] This sets the scattering lengths for the atom types request from atom_types.

The order of the scattering lengths must be provided in the same order as the atom types were in order for the scattering lengths to be paired correctly to their corresponding atom type. This parameter is used for weighting quantities based on the scattering lengths so that the quantity is more readily comparable to scattering experiments.

**atom_group**
- [default: "system", optional] Only useful for trajectories, where groups of atoms exists. This parameter sets if the computation is performed on particular type(s) of atom from the whole system [default], or a sub group such as the solvent [solvent], or protein [protein] or, or non-solvent [non-solvent].

**number_of_frames_to_average**
- [default: 1, optional] This parameter sets the number of frames that will be used to average with. For example, if set to N frames, this quantity will use N unique frames to compute the value of the static quantities, i.e. will use N frames to calculate the radial distribution function. Or, for dynamic quantities, N frames will be used to average the value at every time point, i.e. N frames will be used to average the mean squared displacement at each time point $t$.

**output_precision**
- [default: 15, optional] This parameter sets the scientific precision of the output file.

**boxlength**
- [optional] This parameter takes three values x, y, z to indicate the box lengths of the simulation. This property can be inferred from many trajectory types, such as .xtc or .trr, but for many other types, such as .xyz or XDATCAR, this value should be set to get exact computation.

## 6.2 Pair Distribution Function: $g(r)$

User can input two atom types using the **atom_types1** and **atom_types2** parameters listed below. If the scattering lengths are provided then, they are appropriately weighted. If no value is provided then it is set to 1.

**atom_types1**
- [default: empty, mandatory] This parameter sets the first atom types being studied. By default, the program requires atom types to be provided in order for computation to proceed. But this parameter can be set to the atom identifier used in the trajectory or .gro file, i.e. HW for hydrogen in water, or Li for lithium, or 1 if the atom type is numeric in the trajectory. Can also be set to multiple atom types.

**atom_types2**
- [default: empty, mandatory] The parameter sets the second types being studied. If this parameter is not set, the program requires atom types to be provided in order for computation to proceed.

**atom_group1**
- [default: "system", optional] This parameter can be set if user wants only the atomtype to be from a sub-group solvent[solvent]. The options available are system, solvent, and non-solvent

**atom_group2**
- [default: equal to atomgroup, optional] This parameter can be set if user wants only the atomtype2 to be from a sub-group solvent[solvent]. The options available are system, solvent, and non-solvent

**scattering_lengths1**
- [default: 1, optional] This sets the scattering lengths for the atom types request from atom_types1. The order of the scattering lengths must be provided in the same order as the atom_types1 were in order for the scattering lengths to be paired correctly to their corresponding atom type.

**scattering_lengths2**
- [default: 1, optional] This sets the scattering lengths for the atom types request from atom_types2. The order of the scattering lengths must be provided in the same order as the atom_types2 were in order for the scattering lengths to be paired correctly to their corresponding atom type.

**number_of_bins**

- [default: 200, optional] This value sets the number of spatial ($|r|$) bins used between 0 and maxcutofflength.

**max_cutoff_length**

- [default: half of the smallest box side] This value sets the max $|r|$ used for g(r). By default this value will be set to the smallest box length in any dimension. If a max value greater than half the smallest box length is provided, the value will be reset to the half the smallest box length as spatial correlations for values of $|r|$ greater than half of box length cannot be accurately determined and thus should not be used.

**boxlength**

- [default: read from file] This parameter is for file types that do not contain box information. For file types such as .trr, this will be read from the file.

## 6.3 Structure Factor: $S(k)$

User can input selected atom types using only the **atom_types** parameter. Scattering lengths can be provided in the same order of atom types in order to compute the total structure factor comparable to experiments. Structure Factor is now computed with fixed k vectors based on multiples of the box.

**k_start_index**

- [default: 0, optional] This value determines the first multiple of $\delta k$ used, where $\delta k = 2*\pi/$**boxlength**.

**number_of_bins**

- [default: 50, optional] This sets the number of multiples of $\delta k$ where structure factor will be computed starting from k_start_index and ending at k_start_index + number_of_bins.

## 6.4 Mean Squared Displacement: $\langle r^2(t) \rangle$

The binary to compute Mean Squared Displacement is /bin/computeMeanSquaredDisplacement. User can input selected atom types using only the **atom_types** parameter. The options that can be included in the input file in addition to those listed in Trajectory Parameters are as follows:

**number_of_time_points**

- [default: 0, optional] This is the number of points that will be calculated for the mean squared displacement. If this value is not included it will be derived from the number of frames provided and the number of frames to average. This sets the final frame that will be studied. For linear scale, we will compute the quantity at every frame_interval until number_of_time_points*frame_interval. For log scale, we will compute the quantity at every frame_interval^t where t are integers from 0 to number_of_time_points. This means the final frame used is frame_interval^number_of_time_points. This can result in the output having less than number_of_time_points because there will be repeated points.

**time_scale_type**

- [default: linear, optional] This can be set to "linear" or "log". This sets the time scale of the output. Mean squared displacement can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**

- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater than or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the mean squared displacement to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the mean squared displacement. For example, if this value is set to 1.5, then the mean squared displacement will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**is_partial_msd_output**

- [default: false, optional] This parameter specifies whether partial mean squared displacements along each dimension are also output or not. For example, for a 3D system, setting this parameter to "true" or "yes" will write, besides $\langle r^2(t)\rangle$, the three components $\langle x^2(t)\rangle$, $\langle y^2(t)\rangle$ and $\langle z^2(t)\rangle$ to the output file.

## 6.5 Non-Gaussian Parameter: $\alpha_2(t)$

User can input selected atom types using only the **atom_types** parameter.

**number_of_time_points**

- [default: 0, optional] This is the number of points that will be calculated for the non-Gaussian parameter. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale_type**

- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Non-Gaussian parameter can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**

- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the non-Gaussian parameter to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the non-Gaussian parameter. For example, if this value is set to 1.5, then the non-Gaussian parameter will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

## 6.6 Four Point Correlation: $\chi_4(t)$

User can input selected atom types using only the **atom_types** parameter.

**number_of_time_points**

- [default: 0, optional] This is the number of points that will be calculated for the four point correlation function. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale_type**

- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output, four point correlation function can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**

- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the four point correlation function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the four point correlation function. For example, if this value is set to 1.5, then the four point correlation function will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**overlap_length**

- [**mandatory**] This value sets the overlap length. It is usually set to the square root of the plateau of the mean squared displacement so as to filter out vibrations. For more information on this parameter and how to set the value, please refer to N. Lacevic, F. W. Starr, T. B. Schroder, S. C. Glotzer, "Spatially heterogeneous dynamics investigated via a time-dependent four-point density correlation function," J. Chem. Phys. 119, 7372 (2003).

## 6.7 Velocity Auto Correlation: $C_{vv}(t)$

**number_of_time_points**

- [default: 0, optional] This is the number of points that will be calculated for the velocity auto correlation. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale_type**

- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Velocity auto-correlation can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps). We recommend usage of linear scale for future Fourier transform needs.

**frame_interval**

- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the velocity auto-correlation function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the velocity auto-correlation function. For example, if this value is set to 1.5, then the velocity auto-correlation will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

## 6.8 Self van Hove Correlation: $G_s(r,t)$

User can input selected atom types using only the **atom_types** parameter.

**number_of_time_points**

- [default: 0, optional] This is the number of points that will be calculated for the self van Hove correlation function. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale**

- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Self van Hove correlation can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**

- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the self van Hove correlation function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the self van Hove correlation function. For example, if this value is set to 1.5, then the self van Hove correlation will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**number_of_bins**

- [default: 200, optional] This value determines the number of spatial ($|r|$) bins used between 0 and max_cutoff_length to sample the self van hove correlation.

**max_cutoff_length**

- [default: half of the smallest box side] This value sets the max $|r|$ used for g(r). By default this value will be set to the smallest box length in any dimension. If a max value greater than half the smallest box length is provided, the value will be reset to the half the smallest box length as spatial correlations for values of $|r|$ greater than half of box length cannot be accurately determined and thus should not be used.

## 6.9 Coherent van Hove Correlation: $G(r, t)$

User can input selected atom types using only the **atom_types** parameter. Scattering lengths can be provided in the same order of atom types.

**number_of_time_points**
- [default: 0, optional] This is the number of points that will be calculated for the coherent van Hove correlation function. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale**
- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Self van Hove correlation can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**
- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the self van Hove correlation function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the self van Hove correlation function. For example, if this value is set to 1.5, then the self van Hove correlation will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**number_of_bins**
- [default: 200, optional] This value determines the number of spatial ($|r|$) bins used between 0 and max_cutoff_length to sample the self van hove correlation.

**max_cutoff_length**
- [default: half of the smallest box side] This value sets the max $|r|$ used for g(r). By default this value will be set to the smallest box length in any dimension. If a max value greater than half the smallest box length is provided, the value will be reset to the half the smallest box length as spatial correlations for values of $|r|$ greater than half of box length cannot be accurately determined and thus should not be used.

## 6.10 Self Intermediate Scattering Function: $F_s(k, t)$

User can input selected atom types using only the **atom_types** parameter.

**number_of_time_points**
- [default: 0, optional] This is the number of points that will be calculated for the self intermediate scattering function. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale_type**
- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Self intermediate scattering function can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**
- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the self intermediate scattering function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the self intermediate scattering function. For example, if this value is set to 1.5, then the self intermediate scattering function will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**k_start_index**
- [default: 0, optional] This value determines the first multiple of $\delta k$ used, where $\delta k = 2*\pi/$**boxlength**.

**number_of_bins**
- [default: 50, optional] This sets the number of multiples of $\delta k$ where the self intermediate scattering function will be calculated at, starting from kstartindex and ending at kstartindex + numberofbins.

### 6.11 Coherent Intermediate Scattering Function: $F(k,t)$

User can input selected atom types using only the **atom_types** parameter. Scattering lengths can be provided in order of atom types to compare with experiments.

**number_of_time_points**
- [default: 0, optional] This is the number of points that will be calculated for the self intermediate scattering function. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale_type**
- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Self intermediate scattering function can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**
- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point to skip. For example, if the trajectory has a time step of .1 ps, and you wish for the self intermediate scattering function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the self intermediate scattering function. For example, if this value is set to 1.5, then the self intermediate scattering function will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**k_start_index**
- [default: 0, optional] This value determines the first multiple of $\delta k$ used, where $\delta k = 2*\pi/$**boxlength**.

**number_of_bins**
- [default: 50, optional] This sets the number of multiples of $\delta k$ where the self intermediate scattering function will be calculated at, starting from kstartindex and ending at kstartindex + numberofbins.

### 6.12 Bond Order Parameter: $q_l(i,t)$, $\bar{q}_l(i,t)$, $Q_l(t)$

User can input selected atom types using only the **atom_types** parameter.

**number_of_time_points**
- [default: 0, optional] This is the number of points that will be calculated for the self intermediate scattering function. If this value is not included it will be derived from the number of frames provided and the number of frames to average.

**time_scale_type**
- [default: linear, optional] This can be set to "linear" or "'log". This sets the time scale of the output. Self intermediate scattering function can either be computed in linear scale or log scale, log scale allows for a quicker calculation if the trajectory is very long (i.e. longer than 10000 time steps).

**frame_interval**
- [default: 1, optional] This value has two meanings. If the time_scale is is set to "linear", this value should be an integer greater to or equal to 1, and sets how many time steps between each data point

to skip. For example, if the trajectory has a timestep of .1 ps, and you wish for the self intermediate scattering function to be computed every .2 ps, this value should be set to 2.

- If the time_scale is set to "log", this value determines the base of the scaling for calculating the self intermediate scattering function. For example, if this value is set to 1.5, then the self intermediate scattering function will be computed at nearest rounded integer time points 0, $1.5^0(1)$, $1.5^1(2)$, $1.5^2(2)$, $1.5^3(3)$, ... , $1.5^n$.

**is_averaged**

- [default: no, optional] This is a boolean value. Available options include, no, yes, false, true, or a integer numeric. 0 is considered false, and any other numerical value is considered true. This value determines whether the bond order parameter is calculated per atom or per the system. If false, the bond order parameter is calculated for each atom, if true the bond order parameter is calculated for the system at each frame.

**add_bar**

- [default: no, optional] This is a boolean value. Available options include, no, yes, false, true, or a integer numeric. 0 is considered false, and any other numerical value is considered true. This value determines whether the bond order parameter is calculated or if the nearest neighbor averaged bond order parameter is calculated. If false, the bond order parameter is calculated. If true, the nearest neighbor averaged bond order parameter is calculated.

**bond_parameter_order**

- [required] This value sets the order, $l$, of the bond order parameter. A non-negative integer is required for this option. This determines if say $q_6$ or $q_4$ is calculated

**max_cutoff**

- [required] This value sets the distance to search for nearest neighbors. This should be set to the first minimum of the pair distribution function.

## 7 Common Errors

**ERROR: the trajectory file: FILENAME does not exist.**

This means that the trajectory file provided in the input file failed to be opened. Most likely this means the path was incorrectly provided, there is read protection on the file that prevents opening, or an incompatible file type was provided.

**ERROR: the gro file: FILENAME does not exist.**

This means that the .gro file provided in the input file failed to be opened. Most likely this means the path was incorrectly provided, there is read protection on the file that prevents opening, or an incompatible file type was provided..

**ERROR: .trr files are not compatible in non GROMACS compatible version of LiquidLib.**

This means that a .trr or .xtc trajectory file was provided but *LiquidLib* was not built with the necessary header files and library provided. If you attempted to build a GROMACS version and still obtained this error, this means that the paths provided in the Makefile are most likely incorrect and should be checked.

**ERROR: The trajectory file type: .EXTENSION does not match any known type**

At this time, only .xtc, .trrm ,xyz, .dump, .atom, and XDATCAR files can be provided. We use these extensions to determine how to read in the trajectory, so we require this extension type. We plan to expand on this option in future releases.

**ERROR: unrecognized atom type/group "BLANK"**

This means that the string provided for the atomtype or atomgroup option in the input file did not match any type in the trajectory or .gro files provided. At this point, only system, all, solvent, or non-solvent can be provided to the group option. Maybe check the letter case for consistency or spelling.

**ERROR: Failed to read trajectory, please check the total number of frames**
This occurs usually when the number of frames needed to be read exceeds the number of frames in the trajectory file. If this occurs check the values of endframe, startframe, and numberofframestoaverage to see that they do not add to a value in excess of the number of frames in the trajectory.

**ERROR: The amount of memory to store all of the frames exceeds memory allotment**
There is currently no fix to this problem other then to decrease the number of frames you are attempting to read in from a trajectory file and use for the computation. To help with this, the development version of LiquidLib can be used.

**ERROR: Input file location, BLANK does not exist, please check input**
The input file provided at the command line does not exist. Most likely this is an incorrect path problem.

**ERROR: Output file for "QUANTITY": FILENAME could not be opened**
This means that the output file could not be opened. This most likely occured due to inadequate write priviledge or the path to the file does not exists. Liquidlib cannot create directories if they do not exist. Hence, the output file path must exist.

**ERROR: Require more information to proceed**
This error is to inform the user that program execution cannot continue without either mandatory parameters provided. For instance, frameend or numberoftimepoints provided in the input file for a time dependent quantity. We need this to determine how many data points along the time dimension should be created. One can be derived from the other, like wise, if both are provided numberoftimepoints will take precedence.

**WARNING: no matching input type for: VARIABLE disregarding this variable**
This means one of the options in the input file does not match one of our check variables for the given quantity. Most likely this would result from a typo or a case inconsistency, remember program requires use of all lower case letters.

**WARNING: time step of simulation could not be derived from trajectory**
This merely means that the time step was not provided in the input file and could not be extracted from the trajectory, thus program will set the trajectory $\Delta t$ to 1 arbitrary step or a.u. This warning is to inform the user to be cautious with the output since the units are arbitrary.

**WARNING: number of frames required is greater then the number supplied**
This means that the number of frames needed to compute the numberoftimepoints specified is greater than that given by frameend and framestart. Thus program will change the value of endframe to allow for the necessary number of frames. Note this may still lead to needing more frames then are in the trajectory and thus an error will be thrown in this instance.

**WARNING: max cutoff is greater than half of the boxlength**
The pair distribution function cannot be computed for values of $|r|$ greater than half the box length, the statistics are not accurate past this point. Thus program will reset the maxcutoff to be half of the smallest box length of all dimensions.

## Acknowledgments

This work is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Materials Sciences and Engineering Division, under Award Number DE-SC-0014804.

## References

[1] P. Egelstaff, An Introduction to the Liquid State, Academic Press, 1967. doi:10.1016/B978-0-12-395515-9.50006-0.

[2] J.-P. Hansen, I. R. McDonald, Theory of Simple Liquids, 3rd Edition, Academic Press, Burlington, 2006. doi:10.1016/B978-012370535-8/50003-3.

[3] G. L. Squires, Introduction to the Theory of Thermal Neutron Scattering, 3rd Edition, Cambridge University Press, 2012.

[4] P. J. Steinhardt, D. R. Nelson, M. Ronchetti, Bond-orientational order in liquids and glasses, Phys. Rev. B 28 (2) (1983) 784. doi:10.1103/PhysRevB.28.784.

[5] W. Lechner, C. Dellago, Accurate determination of crystal structures based on averaged local bond order parameters, The Journal of Chemical Physics 129 (11) (2008) 114707.

[6] M. S. Shell, P. G. Debenedetti, F. H. Stillinger, Dynamic heterogeneity and non-gaussian behaviour in a model supercooled liquid, Journal of Physics: Condensed Matter 17 (49) (2005) S4035.
URL http://stacks.iop.org/0953-8984/17/i=49/a=002

[7] N. Laevi, F. W. Starr, T. B. Schrder, S. C. Glotzer, Spatially heterogeneous dynamics investigated via a time-dependent four-point density correlation function, The Journal of Chemical Physics 119 (14) (2003) 7372–7387.

[8] S. C. Glotzer, V. N. Novikov, T. B. Schrder, Time-dependent, four-point density correlation function description of dynamical heterogeneity and decoupling in supercooled liquids, The Journal of Chemical Physics 112 (2) (2000) 509–512. doi:10.1063/1.480541.