1 Correlate

This function uses `scipy.signal.correlate` to cross correlate two discrete functions $a(t)$ and $b(t)$. The function computes $\langle a(0)b(t)\rangle$ directly via sums or using a Fast Fourier Transform algorithm, depending on which is faster (see `scipy.signal.convolve`). In addition to the original `scipy` functionality, the function tailors the correlation function so that only non-negative time values ($\langle a(0)b(t)\rangle$ for $t \geq 0$) are returned. It may only be sensible to calculate such a function if $a(t)$ and $b(t)$ (and the corresponding array elements `a[t]` and `b[t]`) reference the same point in time and the arrays `a` and `b` are of equal length. The autocorrelation function $\langle a(0)a(t)\rangle$ is computed if `b=None` (default).

1.1 Function

`alexandria.correlations.correlate(a, b=None)`

Parameters:

a:	one-dimensional ndarray or list Discrete values of the function $a(t)$.
b:	one-dimensional ndarray or list, optional Discrete values of the function $b(t)$. If `None` the autocorrelation function $\langle a(0)a(t)\rangle$ is computed. Default is `None`.

Output:
An ndarray containing the correlation function $\langle a(0)b(t)\rangle$ for $t \geq 0$ is returned.

2 Lifetimes

To analyze the lifetime of a hydrogen bond or any other impermanent interaction we can define a bonding operator $h(t)$ which is unity if the criteria for the interaction are fulfilled and zero otherwise:[1,2]

$$h(t) = \begin{cases} 1, \text{if criteria are fulfilled} \\ 0, \text{otherwise} \end{cases} . \tag{2.1}$$

The fluctuations of $h(t)$ can be described by the autocorrelation function $C(t)$

$$C(t) = \frac{\langle h(0)h(t)\rangle - \langle h\rangle^2}{\langle h\rangle}, \tag{2.2}$$

which describes the probability of the bond being intact at the time t if the bond was intact at $t = 0$. The so-called intermittent lifetime of the interaction can be estimated from $C(t)$.[3–6]

The reactive flux approach[4,6–9] is another approach to estimate the lifetime of such interactions. To follow that approach we require the function $k_{\text{in}}(t)$

$$k_{\text{in}}(t) = -\frac{\left\langle \dot{h}(0)[1 - h(t)]H(t)\right\rangle}{\langle h\rangle}, \tag{2.3}$$

where $\dot{h}$ denotes the time-derivative of $h(t)$. $H(t)$ is a vicinity operator closely related to $h(t)$. If the donor and acceptor of the interaction are "near" each other $H(t)$ equals unity otherwise it equals zero.

The purpose of `calc_lifetime` is to calculate the correlation functions $\langle h(0)h(t)\rangle$ from equation 2.2 and $-\left\langle \dot{h}(0)[1 - h(t)]H(t)\right\rangle$ from equation 2.3. Normalizing the correlation functions will be up to the user, since several approaches are viable.[10] To obtain both correlations we first need to determine $h(t)$ and $H(t)$ for every donor-acceptor pair. After that, we are able to compute both correlation functions and average them over all donor-acceptor pairs.

2.1 Function

```
alexandria.lifetime.calc_lifetime(universe, timestep, xgrp, hgrp,
    cutoff_hy, cutoff_xy, angle_cutoff, ygrp=None, nproc=1,
    check_memory=True)
```

Parameters:

`universe`:	MD.Analysis.Universe Universe containing the trajectory.
`timestep`:	int or float Timestep between configurations in the `universe`. The unit is freely selectable and will influence the units of the output.
`xgrp`:	AtomGroup from MDAnalysis AtomGroup containing all atoms X involved in the interaction X–H $\cdots$ Y. Has to be the same size as `hgrp`.
`hgrp`:	AtomGroup from MDAnalysis AtomGroup containing all atoms H involved in the interaction X–H $\cdots$ Y. Has to be the same size as `xgrp`.
`ygrp`:	AtomGroup from MDAnalysis or None, optional MDAnalysis AtomGroup containing all atoms Y involved in the interaction X–H $\cdots$ Y. If `None` is given, it is assumed that Y=X (interaction X–H $\cdots$ X) and `xgrp` is taken as acceptor group. The default is `None`.
`cutoff_hy`	int or float Criterion for the H$\cdots$Y distance in Å to define $h(t)$. The criterion is fulfilled if the distance between a HY-pair is smaller than the value specified.
`cutoff_xy`	int or float Criterion for the X$\cdots$Y distance in Å to define $H(t)$. The criterion is fulfilled if the distance between a XY-pair is smaller than the value specified.
`angle_cutoff`	int or float Criterion for the angle $\alpha\angle$XHY in in radian to define $h(t)$. The cutoff is set so that if $\alpha >$ angle_cutoff the criterion is fulfilled.

`nproc`	int, optional Number of processors available to parallelize the execution of the script. The default is 1.
`check_memory`	bool, optional Perform an approximate check if the amout of memory is sufficient. The default is `True`.

Output:

For every donor i in `xgrp` a file `ct_i.dat` will be created. The file contains the results in three columns. The first column contains the timestep t in the same unit given in the option `timestep`. The second column contains $\langle h(0)h(t)\rangle$ for that donor. The third column contains $-\left\langle \dot{h}(0)[1-h(t)]H(t)\right\rangle$ for that donor in inverse units of `timestep`. As long as the amount of acceptors (`ygrp`) is constant the data of multiple donors i can be averaged by computing the arithmetic mean of the desired quantity.

3 gofr

The radial distribution function $g_{\mathrm{AB}}(r)$ describes the density of particle B in a spherical shell of width dr at distance r around particle A in relation to the average number density of B $\langle \rho_{\mathrm{B}} \rangle$ in the system

$$g_{\mathrm{AB}}(r) = \frac{\langle \rho_{\mathrm{B}}(r) \rangle}{\langle \rho_{\mathrm{B}} \rangle} = \frac{1}{\langle \rho_{\mathrm{B}} \rangle \cdot N_{\mathrm{A}}} \left\langle \sum_{i \in \mathrm{A}}^{N_{\mathrm{A}}} \sum_{j \in \mathrm{B}}^{N_{\mathrm{B}}} \frac{\delta(r_{ij} - r)}{4\pi r^2} \right\rangle . \tag{3.1}$$

Here, N_{A} and N_{B} references the number of particles A and B in the system, respectively. It should be noted that $g_{\mathrm{AB}}(r) = g_{\mathrm{BA}}(r)$.

From $g_{\mathrm{AB}}(r)$ and $\langle \rho_{\mathrm{B}} \rangle$ the average cumulative number of neighbors $N_{\mathrm{B}}(R)$ of paricles B in a sphere of radius R around a particle A is obtainable via

$$N_{\mathrm{B}}(R) = \rho_{\mathrm{B}} \cdot 4\pi \int_0^R g_{\mathrm{AB}}(r)\, r^2 \mathrm{d}r \tag{3.2}$$

Similarily, $N_{\mathrm{A}}(R)$ can be computed using $\langle \rho_{\mathrm{A}} \rangle$.

Three modes (`mode`) are implemeted at the moment: `"site-site"`, `"cms-cms"`, and `"site-cms"`. The mode `"site-site"` computes the average $g_{\mathrm{AB}}(r)$ between all atoms A in `agrp` and all atoms B in `bgrp`. For example, if `agrp` contains atoms A0 and A1 while `bgrp` contains atoms B0 and B1, the pairs A0B0, A0B1, A1B0, and A1B1 will contribute to $g_{\mathrm{AB}}(r)$.

The mode `"cms-cms"` can be used to compute center-of-mass radial distribution functions. It will calculate the center-of-mass of atoms belonging to the same molecule in `agrp` and `bgrp` and procede to calculate the radial distribution function of these centers-of-mass. For example, given an `agrp` containing four atoms belonging to two different molecules (A0 and A1 belonging to molecule M0, A2 and A3 belonging to M1) and the same for `bgrp` (B0 and B1 belonging to M2, B2 and B3 belonging to M3) it will first calculate the centers-of-mass $\mathrm{cms}_{\mathrm{M0}}$(A0,A1), $\mathrm{cms}_{\mathrm{M1}}$(A2,A3), $\mathrm{cms}_{\mathrm{M2}}$(B0,B1), and $\mathrm{cms}_{\mathrm{M3}}$(B2,B3). The radial distribution function will then contain contributions from the pairs $\mathrm{cms}_{\mathrm{M0}}\mathrm{cms}_{\mathrm{M2}}$, $\mathrm{cms}_{\mathrm{M0}}\mathrm{cms}_{\mathrm{M3}}$, $\mathrm{cms}_{\mathrm{M1}}\mathrm{cms}_{\mathrm{M2}}$, and $\mathrm{cms}_{\mathrm{M1}}\mathrm{cms}_{\mathrm{M3}}$.

The mode `"site-cms"` is a mix of both of the functions described above, where `agrp` is taken atom-wise as in `gofr` and for `bgrp` the center-of-mass of atoms belonging to the same molecule is calculated as in `gofr_cms`.

3.1 Functions

```
alexandria.gofr.Gofr(universe, agrp, bgrp, rmax, rmin=0, bins=100,
    mode="site-site", outfilename="gofr.dat")
```

Parameters:

`universe`:	MD.Analysis.Universe Universe containing the trajectory.
`agrp`:	AtomGroup from MDAnalysis AtomGroup containing all atoms A.
`bgrp`:	AtomGroup from MDAnalysis AtomGroup containing all atoms B.
`rmax`:	int or float The upper boundary of the A···B distance used for the $g(r)$ in units of Å.
`rmin`:	int or float, optional The lower boundary of the A···B distance used for the $g(r)$ in units of Å. The default is 0.
`bins`:	int or sequence of scalars or str, optional Specifies the number of points between `rmin` (inclueded) and `rmax` (excluded). Will be used directly by numpy.histogram. From the numpy documentation: "If bins is an int, it defines the number of equal-width bins in the given range. If bins is a sequence, it defines a monotonically increasing array of bin edges, including the rightmost edge, allowing for non-uniform bin widths. If bins is a string, it defines the method used to calculate the optimal bin width, as defined by histogram_bin_edges." The default is 100.
`mode`:	str, optional

Sets the mode for calculating different radial distribution functions: "site-site", "cms-cms", "site-cms". If mode is set to "site-site", the average radial distribution function of all sites in agrp to all sites in bgrp will be computed. The mode "cms-cms" will first compute the center-of-mass of sites belonging to the same molecule in agrp and bgrp, respectively, and then determin the radial distribution function between those centers of mass. The mode "site-cms" is a mix between the two, where every site in agrp is taken individualy but for bgrp the center-of-mass of sites belonging to the same molecule is computed firs. The default is "site-site".

`outfilename`: str, optional
The name of the output file. The default is "gofr.dat".

Output:
The program creates a file named `outfilename` with the distance r in Å (first column), the radial distribution function $g_{\mathrm{AB}}(r)$ (second column), the cumulative number of neighbors A in a sphere of radius r around particle B $N_{\mathrm{A}}(r)$ (third column), and the cumulative number of neighbors B in a sphere of radius r around particle A $N_{\mathrm{B}}(r)$ (fourth column).

Class Methods:

`rdat`: Distance r (center of bins).

`edges`: Edges of the bins.

`hist`: Radial distribution function $g_{\mathrm{AB}}(r)$.

`annn`: Average number of neighbors A in a sphere of radius r around particle B $N_{\mathrm{A}}(r)$.

`bnnn`: Average number of neighbors B in a sphere of radius r around particle A $N_{\mathrm{B}}(r)$.

`avvol`: Average volume of the universe.

`na`: Number of particles A in `agrp`. If mode is "site-site" or "site-cms", `na` is the number of sites in `agrp`. If mode is "cms-cms", `na` is the number of molecules (centers-of-mass) in `agrp`.

nb: Number of particles B in `bgrp`. If mode is `"site-site"`, `na` is the number of sites in `agrp`. If mode is `"site-cms"` or `"cms-cms"`, `nb` is the number of molecules (centers-of-mass) in `bgrp`.

3.2 Examples

We start with a simulation of water, where all oxygen atoms are named `"ow"` and all hydrogen atoms `"hw"`. The resname of the water molecules is `"SOL"` for solvent. We will compute three different radial distribution functions to show reveal the differences in `"site-site"`, `"cms-cms"`, and `"site-cms"`: Firstly, we will use `"site-site"` to calculate the radial distribution function between all hydrogen and oxygen (H$\cdots$O) atoms, which could for example be used to define the hydrogen bond O–H$\cdots$O. Secondly, we will calculate the center-of-mass radial distribution function of all water molecules (cms$\cdots$cms) using `"cms-cms"`. Thirdly, we use `"site-cms"` to compute the radial distribution function of all hydrogen atoms to the centers-of-mass of all water molecules (H$\cdots$cms).

We first have to create a universe and select different AtomGroups to achieve the goals described above.

```python
from alexandria import gofr
import MDAnalysis

topology = "/Path/to/topology.tpr"
trajectory = "/Path/to/trajectory.xtc"

u = MDAnalysis.Universe(topology, trajectory)
hgrp = u.select_atoms("name hw")
ogrp = u.select_atoms("name ow")
watergrp = u.select_atoms("resname SOL")

sitesite = Gofr(universe=u, agrp=hgrp, bgrp=ogrp, rmin=1.0, rmax=6,
    bins=200, mode="site-site", outfilename="h_o.dat")
cmscms = Gofr(universe=u, agrp=watergrp, bgrp=watergrp, rmin=1.0,
    rmax=6, bins=200, mode="cms-cms", outfilename="cms_cms.dat")
sitecms = Gofr(universe=u, agrp=hgrp, bgrp=watergrp, rmin=1.0,
    rmax=6, bins=200, mode="site-cms", outfilename="h_cms.dat")
```

In Fig. 3.1 the three different $g(r)$ are plotted. Additionaly, we obtain the neighbour numbers $N_\mathrm{A}(r)$ $N_\mathrm{B}(r)$ for each pair. In case of the H$\cdots$O distribution $N_\mathrm{A}(r)$ would be the average number of hydrogen atoms in a sphere of radius r around an oxygen atom. In case of the cms$\cdots$cms distribution $N_\mathrm{A}(r) = N_\mathrm{B}(r)$ denotes the number of water molecules in a sphere of radius r around a water molecule. In case of the H$\cdots$cms distribution $N_\mathrm{A}(r)$ is the number of water molecules around in a sphere of radius r around a hydrogen atom.

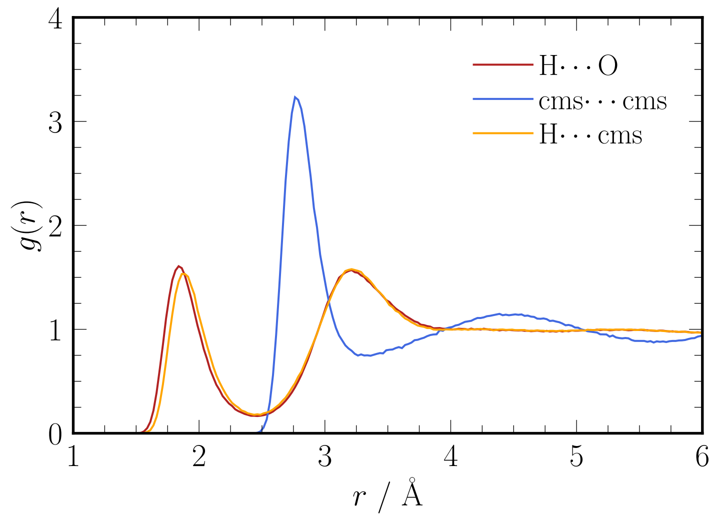

Fig. 3.1: Radial distribution functions obtained from the example above.

4 hb_analyze

Geometric criteria can be used to define a hydrogen bond. Distance criteria can often be derived from pair-correlation functions but it may be required to include angular restrictions on the interaction. Twodimensional potentials of mean force (PMFs) can be used to obtain such criteria.[11,12] The PMF is calculated using the probability density of finding a donor-acceptor pair with the respective donor-acceptor ($\mathrm{H}\cdots\mathrm{Y}$) distance r and angle α ($\alpha\angle\mathrm{XHY}$).

This density can be derived from populations from equilibrium molecular dynamics trajectories. Therefore, donor-acceptor pairs with a distance r between `rmin` and `rmax` and an angle $\cos(\alpha)$ between `cosalphamin` and `cosalphamax` will be counted in a twodimensional histogram according to the number of bins specified with the option `bins`. Each count is weighted with the respective r^{-2} to account for the growth of the spherical volume element with increasing r. The histogram is then normalized to the respective probabilty density function $P(r, \cos(\alpha))$ using the area of each bin $\mathrm{d}r \cdot \mathrm{d}\cos(\alpha)$ and the sum of all counts so that the integral over P is unity. At the end, the natural logarithm of P in each bin is calculated, due to the connection between P and the PMF via

$$F = -k_{\mathrm{B}}T\ln(P) + c, \tag{4.1}$$

with the Boltzmann constant k_{B}, the temperature T, and an unknown constant c. The output is a twodimensional grid where each bin contains $\ln(P)$ of the respective bin.

4.1 Function

```
alexandria.hb_analyze.hb_analyze(universe, xgrp, hgrp, rmax, ygrp=None,
    rmin=0, cosalphamin=-1, cosalphamax=1, bins=50,
    outfilename="hb_analyze.dat", ralphalist=False)
```

Parameters:

`universe`: MD.Analysis.Universe

Universe containing the trajectory.

`xgrp`: AtomGroup from MDAnalysis
AtomGroup containing all atoms X involved in the interaction X–H $\cdots$ Y. Has to be the same size as `hgrp`.

`hgrp`: AtomGroup from MDAnalysis
AtomGroup containing all atoms H involved in the interaction X–H $\cdots$ Y. Has to be the same size as `xgrp`.

`ygrp`: AtomGroup from MDAnalysis or None, optional
MDAnalysis AtomGroup containing all atoms Y involved in the interaction X–H $\cdots$ Y. If `None` is given, it is assumed that Y=X (interaction X–H $\cdots$ X) and `xgrp` is taken as acceptor group. The default is `None`.

`rmax`: int or float
The upper boundary of the H $\cdots$ Y distance in units of Å.

`rmin`: int or float, optional
The lower boundary of the H $\cdots$ Y distance in units of Å. The default is 0.

`cosalphamin`: int or float, optional
The lower boundary of $\cos(\alpha)$ ($\alpha \angle$XHY). The default is -1.

`cosalphamax`: int or float, optional
The upper boundary of $\cos(\alpha)$ ($\alpha \angle$XHY). The default is 1.

`bins`: int or array_like or [int, int] or [array, array], optional

Bins used for the 2D-histogram. Will be used directly by numpy.histogram2d. For two numbers the first will specify the bins of the H $\cdots$ Y distance (x_edges) and the second will specify the bins of $\cos(\alpha)$ (y_edges) The default is 50. Specifications:

- If int, the number of bins for the two dimensions (nx=ny=bins).
- If array_like, the bin edges for the two dimensions (x_edges=y_edges=bins).
- If [int, int], the number of bins in each dimension (nx, ny = bins).
- If [array, array], the bin edges in each dimension (x_edges, y_edges = bins).
- A combination [int, array] or [array, int], where int is the number of bins and array is the bin edges.

`outfilename`: str, optional
The name of the outputfile. The default is `hb_analyze.dat`.

`ralphalist`: bool, optional
Changes the output from the weighted probability density matrix to the list containing all the H $\cdots$ Y distances and corresponding $\cos(\alpha)$ from which the probability density is calculated. The default is `False`.

Output:
The program creates a file named `outfilename` with the weighted twodimensional histogram. The first axis represents the H $\cdots$ Y distance and the second axis represents $\cos(\alpha)$ ($\alpha\angle$XHY).

If `ralphalist=True` the file contains the distances and corresponding angles of HY-pairs as a list: in the first column the distances are written in units of Å and the second column indicates the cosine of the corresponding angle $\cos(\alpha)$, both in the respective range `rmin` to `rmax` and `cosalphamin` to `cosalphamax`.

4.2 Example and Visualization

To use `hb_analyze` we first have to create a universe, define `xgrp` and `hgrp`, the range of the histogram, and the amount of bins in each dimension. If no `ygrp` is given the program will use xgrp as acceptor group and analyze the interaction X–H $\cdots$ X instead. Here an example for a water simulation where the oxygen atoms are named "ow" and the hydrogen atoms "hw":

```
import numpy as np
from hb_analyze import hb_analyze
import MDAnalysis

topology = "/Path/to/topology.tpr"
trajectory = "/Path/to/trajectory.xtc"
xname = "ow"
hname = "hw"

u = MDAnalysis.Universe(topology, trajectory)
xgrp = u.select_atoms("name "+str(xname))
hgrp = u.select_atoms("name "+str(hname))[::2]

hb_analyze(universe=u, xgrp=xgrp, hgrp=hgrp, rmin=1.5, rmax=5,
    cosalphamin=-1, cosalphamax=1, bins=50)
```

After excecution a file hb_analyze.dat (changable by the option outfilename) can be found in the current folder. It contains the 50×50 (bins) matrix of the weighted probability function. This matrix can be plotted by matplotlibs contour and similar programs. Here an example using matplotlibs contourf:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import ticker
import matplotlib.colors as col

lowcolor = '#ffffff'
midcolor1 = '#6090f0'
midcolor2 = '#30f050'
midcolor3 = '#f0f000'
midcolor4 = '#f06000'
highcolor = '#b02000'
```

```
12 cmapown = col.LinearSegmentedColormap.from_list('own', [lowcolor,midcolor1,midcolor2,midcolor3,midcolor4,highcolor])
13 cmapown.set_over('#9e1c00')
14
15 rmin=1.5
16 rmax=5
17 cosalphamin=-1
18 cosalphamax=1
19
20 fig, ax = plt.subplots()
21 histo_matrix = np.loadtxt("hb_analyze.dat")
22 levels = ticker.MaxNLocator(nbins=60).tick_values(-3, 2)
23 cax = ax.contourf(histo_matrix, extent=(cosalphamin, cosalphamax, rmin, rmax), levels=levels, extend='both', cmap=cmapown)
24 plt.xlabel('$\\cos(\\alpha)$')
25 plt.ylabel('$r$ / \\AA')
26 plt.axis([cosalphamin, cosalphamax, rmin, rmax])
27 cbar = fig.colorbar(cax, ticks=[-3, -2, -1, 0, 1, 2])
28 cbar.ax.set_ylabel('$\\log[W(\\cos(\\alpha), r)]$')
29 plt.tight_layout()
30 plt.savefig("histo.pdf")
31 plt.clf()
```

After importing the necessary modules, we first define or own colormap `cmapown` (line 6 to 13). Standard colormaps can be found here. The output of `hb_analyze` only contains the weighted probabilty densities for each bin and not their position, so we have to tell the program in line 15–18 in which range the histogram is plotted (option `extent` of `contourf` line 23 and x- and y-axis limits line 26).

The actual plotting happens onwards from line 20. Using numpys `loadtxt` we load the histogram matrix into the array `histo_matrix` (line 21). In line 22 we define the amount of bins (`nbis=60`) and the range `(-3, 2)` of the coloraxis. The array can directly be processed by `contourf` where we also input the range, levels, and colormap. The option `extend='both'` enables the colors beyond the levels definded before (arrows above and below the coloraxis). The lines 24 to 28 are defining the axis-ticks and -labels. After that the plot is already finished and can be saved or shown directly. An example plot for a small watersimulation is shown in Fig. 4.1.

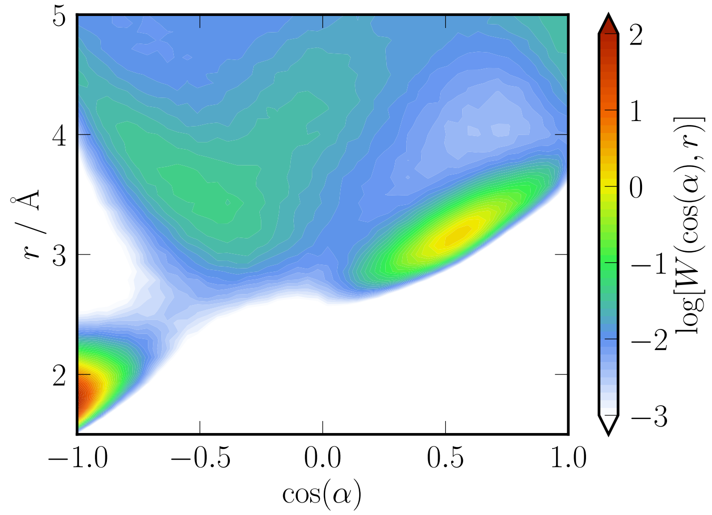

Fig. 4.1: Example plot of a twodimensional histogram computed with `hb_analyze`.

Bibliography

[1] D. Chandler, *J. Chem. Phys.* **1978**, *68*, 2959.

[2] C. H. Bennett in *Algorithms for Chemical Computations*, Chapter 4, p. 63.

[3] S. Gehrke, M. von Domaros, R. Clark, O. Hollóczki, M. Brehm, T. Welton, A. Luzar, B. Kirchner, *Faraday Discuss.* **2018**, *206*, 219.

[4] S. Gehrke, B. Kirchner, *J. Chem. Eng. Data* **2019**, *65*, 1146.

[5] J. Neumann, D. Paschek, A. Strate, R. Ludwig, *J. Phys. Chem. B* **2021**, *125*, 281.

[6] J. Neumann, R. Ludwig, D. Paschek, *J. Phys. Chem. B* **2021**, *125*, 5132.

[7] A. Luzar, D. Chandler, *Phys. Rev. Lett.* **1996**, *76*, 928.

[8] A. Luzar, D. Chandler, *Nature* **1996**, *379*, 55.

[9] A. Luzar, *J. Chem. Phys.* **2000**, *113*, 10663.

[10] J. Busch, J. Neumann, D. Paschek, *J. Chem. Phys.* **2021**, *154*, 214501.

[11] R. Kumar, J. R. Schmidt, J. L. Skinner, *J. Chem. Phys.* **2007**, *126*, 204107.

[12] D. Trzesniak, A.-P. E. Kunz, W. F. van Gunsteren, *ChemPhysChem* **2007**, *8*, 162.