

Gibbs Ensemble Monte Carlo Simulations for Fluid Phase Equilibria

Authors

Michael Haring, BSc.

Niklas Mayr, BSc.

Sebastian Thaler, BSc.

Thomas Steiner, BSc.

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Thomas Wallek
Graz, January 2024

Executive Summary

The following list states the main concepts implemented in the present software package:

- **Project Structure**

As proposed by Allen and Tildesley, the simulation consists of three phases, namely the warm-up, equilibration and production phase.[1]

- **Gibbs Ensemble Monte Carlo Algorithm**

The basis of the present software package is the Gibbs ensemble Monte Carlo algorithm in the NVT and NPT ensemble.[33, 28, 31]

- **OPLS Force Field**

The OPLS Force Field is used for the calculation of the intermolecular interactions. [20]

- **Tail Corrections for Energy and Pressure**

Analytical tail corrections are used to account for the truncation of the intermolecular interactions of the Lennard-Jones potential.[19]

- **Various Combining Rules**

The calculation of cross interactions can be done using the geometric mean or the Lorentz-Berthelot combining rules [23]. Other combining rules can be easily implemented.

- **Pressure Calculation**

The pressure is calculated based on the ideal and virial contribution.[26]

- **Widom Ghost Particle Method**

The Widom ghost particle method is used to determine the chemical potential.[51]

- **Electrostatic Long-Range Correction**

The electrostatic potential at long distances can be calculated with either the Ewald or Wolf summation.[8, 52]

- **Improved Insertion Algorithm for Binary Mixtures**

An improved insertion algorithm to increase the acceptance rate of molecule transfers based on identity changes is implemented for binary mixtures.[29]

- **Constraint-Specific Intramolecular Trial Moves**

Additional trial moves to allow for intramolecular bond, angle and dihedral angle flexibility.

- **Continuous Fractional Component (CFC) Method**

The continuous fractional component method is implemented as suggested by Ali Pour-saeidesfahani in the working group of Thijs J. H. Vlugt.[36]

- **Auto-calibration of Transfers**

If activated, the number of attempted transfers per cycle can be auto-calibrated to yield a required number of successful trial moves per cycle.[5]

- **Species Choice Probability for Insertions**

The probability to choose a species for an attempted insertion can be tuned or auto-calibrated to yield an equal number of successful transfers per species.[31]

- **Combined Analysis of Simulations**

Results of multiple continuous simulations and extensions can be summarized and displayed in a single file

Contents

Executive Summary	ii
1. Introduction	1
2. Quick Start	3
2.1. Obtaining the Simulation Algorithm	3
2.2. Executing Simulations	4
2.3. Execution from the Command Line	5
2.4. Remote Execution	5
2.5. Example Simulations	6
2.6. Literature	7
3. Theoretical Background	8
3.1. The Principle of Monte Carlo	8
3.2. Metropolis Algorithm $M(RT)^2$	9
3.3. Simulation of Phase Equilibria	10
3.4. Lennard-Jones Potential	11
3.5. Gibbs Ensemble	13
4. Methods	14
4.1. Energy Calculation	14
4.1.1. OPLS Force Field	14
4.1.2. Interaction Parameters	16
4.1.3. Periodic Boundary Condition	17
4.1.4. Cut-off Radius and Tail Correction	19
4.1.5. Overlap Radius	22

4.2.	Determination of Pressure and Chemical Potential	22
4.2.1.	Pressure Calculation	23
4.2.2.	Calculation of the Chemical Potential	23
4.3.	Improved Insertion Algorithm for Binary Mixtures	24
4.4.	Gibbs Ensemble Continuous Fractional Component	25
4.4.1.	Scaled LJ Potential	26
4.4.2.	Wang-Landau Algorithm	28
4.4.3.	CFC Trial Moves	29
4.4.4.	Calculation of the Chemical Potential	30
4.5.	Adjustment of Parameters	31
4.5.1.	Trial Move Limits	32
4.5.2.	Number of attempted Transfers per Cycle	33
4.5.3.	Species Choice Probability for Insertions	35
4.6.	Electrostatic Long-Range Correction	38
4.6.1.	Ewald Summation	38
4.6.2.	Wolf Summation	41
5.	Implementation	43
5.1.	Units used during the Simulation	43
5.2.	01 - Setup.wls	43
5.2.1.	Molecules and their Parameters	44
5.2.2.	System Properties	45
5.2.3.	Number of Cycles and Trial Moves	50
5.2.4.	Trial Move Limits	51
5.2.5.	Initial Molecule Positions and Orientations	52
5.2.6.	Overview of Initial Conditions	54
5.2.7.	Custom Parameters	57
5.3.	02a - Execution.wls	57
5.3.1.	Initialization Phase	57
5.3.2.	Trial Moves	65
5.3.3.	Execution Phase	75
5.3.4.	02b - ExecutionHanlder.wls	77
5.3.5.	02c - ExecutionUnitTests.wls	78
5.3.6.	02d - ExecutionDebugging.wls	78

5.4.	03a - Analysis.wls	79
5.4.1.	Calculations of Thermodynamic Properties	79
5.4.2.	Plots and Charts	84
5.4.3.	Ensemble Averages to obtain Bulk Properties	85
5.4.4.	Export of Simulation Data	86
5.4.5.	Analysis of Multiple Continuous Simulations	86
5.4.6.	03b - Combined Analysis.wls	87
5.4.7.	Custom Evaluations	88
5.5.	04 - Extension.wls	89
5.6.	Packages	90
5.6.1.	Analysis	91
5.6.2.	ChemicalThermo	91
5.6.3.	ForceFieldOPLS	92
5.6.4.	HelperFunctions	93
5.6.5.	MolecularSampling	94
5.6.6.	Packages	98
5.6.7.	Property	98
5.7.	Conventions & Workflow	99
5.7.1.	Program Structure	99
5.7.2.	Code Style	100
5.7.3.	Documentation	102
6.	Parameterization	105
6.1.	Pure Lennard-Jones Fluids ¹	105
6.1.1.	Guidelines	105
6.1.2.	Equation of State for pure LJ Fluids	107
6.1.3.	Results	108
6.2.	Binary Lennard-Jones Fluids	110
6.2.1.	Guidelines	110
6.2.2.	Equation of State for binary Mixtures of LJ Fluids	113
6.2.3.	Results	114

¹Andreas Schwarz contributed to this section.

6.3. Rigid Molecules ²	116
6.3.1. Guidelines	116
6.3.2. Results	117
6.4. Continuous Fractional Component (CFC) Method	119
6.4.1. Guidelines	119
6.4.2. Results	123
6.4.3. Conclusion	125
6.5. Charged Components	127
6.5.1. Initial Parameters for Ewald and Wolf Summation	127
6.5.2. Guidelines	129
6.5.3. Results	130
6.6. Intramolecular Flexible Components	132
6.6.1. Custom Parameter	132
6.6.2. Simulation Results	133
Appendices	I
I. Step by Step Instructions for the first Simulations ³	II
I.I. First Example - Pure LJ fluid at moderate temperature	II
I.II. Second Example - Pure LJ fluid at high temperature	XIII
I.III. Third Example - Carbon Dioxide	XVI
I.IV. Usage of the "00-Estimation Notebook"	XXII
Future Extensions and Adaptations	XXVI
II. Changelog and Credits	XXVII
III. Reference Tables	XXX
III.I. List of Figures	XXX
III.II. List of Tables	XXXII
References	XXXII

²Theresa Plesch contributed to this section.

³Theresa Plesch contributed to this section.

1. Introduction

The present report details the implementation of the Gibbs ensemble Monte Carlo (GEMC) technique to simulate phase equilibria. The package is developed in the Wolfram Language with Mathematica.[53] The software can be used to simulate pure components and binary mixtures of rigid partially charged molecules in the NVT and NPT ensemble. The OPLS force field is used for calculating intermolecular interactions, and different combining rules can be used for unlike pair interaction parameters. Both the Ewald and the Wolf summation can be used for calculation of the electrostatic long-range correction. Simulations can be conducted using conventional as well as more advanced methods for insertions like the identity swap method for binary mixtures and the continuous fractional component (CFC) method. It is possible to auto-calibrate simulation parameters such as trial move limits, number of attempted transfers and species selection pseudo-probability for transfers to ensure sufficient successful trial moves. A simulation is conducted in three phases: (i) the warm-up, (ii) the equilibration, and (iii) the production phase. A schematic of the GEMC method is displayed in Figure 1.1.

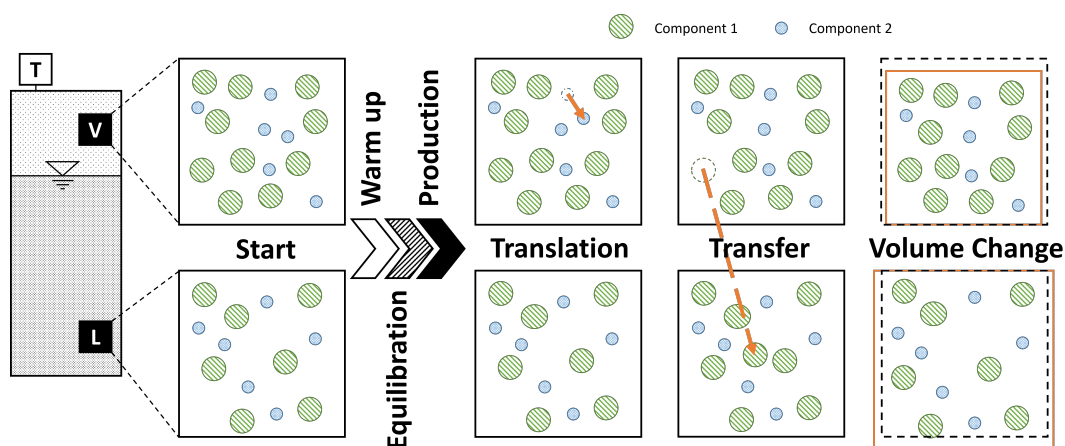


Figure 1.1.: Schematic of the Gibbs ensemble Monte Carlo simulation method [30]

This document is structured as follows: In chapter 2, concise introductions for newcomers are provided. In chapter 3, the theoretical background of the GEMC method is briefly sketched. In chapter 4, the implemented methods are described and in chapter 5 their implementation is laid out. Finally, in chapter 6, guidelines and exemplary simulation results are given. Some step-by-step introductions for the first simulations are provided in the appendix in section I.

Good Luck!

2. Quick Start

The following chapter explains how the simulation code can be acquired and how to get started with the execution of simulations. Exemplary simulations and some literature suggestions for newcomers to Monte Carlo simulations are provided. It is assumed that Mathematica [53] (version 13.1 or later) is installed. Furthermore, we recommend to use GIT for synchronizing simulation setups and results between workstations.

2.1. Obtaining the Simulation Algorithm

The software package is hosted at [the GIT-Lab server](#) of TU Graz. A step-by-step manual on how the software package is obtained can be found there once access has been granted. A public version is also available for download at [notebookarchive.org](#).^[43]

The GIT application can be downloaded from <https://git-scm.com/> and installed using the recommended default settings. It can be installed on Linux by executing:

```
sudo apt-get install git
```

Even though GIT might appear a bit complicated/time-consuming at first sight, it is quite useful once one has become accustomed to it. The main advantage compared to conventional synchronization tools (e.g., Google Drive) is, that one can add a brief description to particular changes. Also, one has full control over the whole process and can thereby be sure that the versions used on different computers are consistent. Further code changes can be tracked and compared, whereby changes can be easily merged.

2.2. Executing Simulations

The principle of the workflow is illustrated in Figure 2.1.

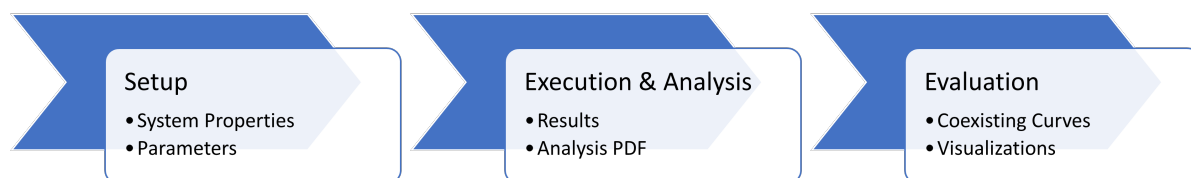


Figure 2.1.: Illustration of the workflow

First, all required simulations are set up using the '*01 - Setup.wls*' notebook to create a folder containing the '*simulationDefinitions.m*' and a possible '*parameter.m*' file. The folder is conveniently named after the system used, the executed number of cycles as well as further simulation details.

Second, simulations can be executed using the '*02a - Execution.wls*' or '*02b - ExecutionHandler.wls*' notebook creating a '*results.zip*' file in the same folder. In case multiple simulations are set up, it is recommended to store them in one common parent folder, so that they can conveniently be executed consecutively using the '*02b - ExecutionHandler.wls*' notebook without having to specify the paths of the individual simulations.

Third, results obtained from the simulation can be analyzed using the '*03a - Analysis.wls*' notebook. The results of the analysis are summarized in a PDF document and saved as an association in an additional file (*analysisResults.m*) which can be further evaluated using the '*03b - CombinedAnalysis.wls*' notebook. With the combined analysis, phase diagrams, comparison plots or tables of properties of interest can be created and exported conveniently. In case the execution handler is used, the analysis can also be executed automatically and the results are summarized in a CSV File (*combinedResults_YYYY-MM-DDTHH-MM-SS.csv*), which can be imported into a spreadsheet software.

For more details about each notebook, see chapter 5 and the notebooks themselves.

2.3. Execution from the Command Line

In case no Mathematica front end is available, the notebook '*02a - Execution.wls*' can also be executed from the command line using 'WolframScript'. First, ensure that the installation folder of 'WolframScript' is added to the 'PATH' of your operating system. Afterward, enter the following line into the command console, specifying the path to the '*02a - Execution.wls*' notebook and the flag '-p' or '--path' followed by the path towards the simulation folder to be executed.

```
wolframscript <path-to-02a-Execution.wls> [--options]
```

The following options are implemented:

```
[-p | --path] <path-to-simulation-folder>
```

Specify the path to the simulation folder containing the 'simulationDefinitions.m'-file.
This option needs to be specified.

```
[-v | --verbose]
```

Prints additional real-time visualization of the simulation progress.
This option can be omitted to reduce effort.

2.4. Remote Execution

Once the simulations are set up, one can execute them locally or on a remote workstation. For remote execution, GIT can be used to conveniently synchronize the locally configured simulations. This is done by executing the following commands on the local computer while operating on the export-branch:

```
git add -A  
git commit -m "<description>"  
git push
```


Now that the simulations are synchronized with the GIT server, we have to switch to the remote computer. The remote computer can either be accessed via console or by using the "Remote Desktop" connection. Once logged in at the remote computer the export directory is updated by executing the following command in the console.

```
git pull
```

After the simulations are finished, the results are retrieved and analyzed locally. To upload the results to the GIT server, the changed files are added and uploaded as described above. Afterwards, the results can be downloaded to the local computer to be further analyzed.

It is recommended to minimize the work on the remote computer since there is a significant time delay for mouse and keyboard input due to the VPN, the virtual machine, the internet connection, etc. This workflow is illustrated in Figure 2.2.

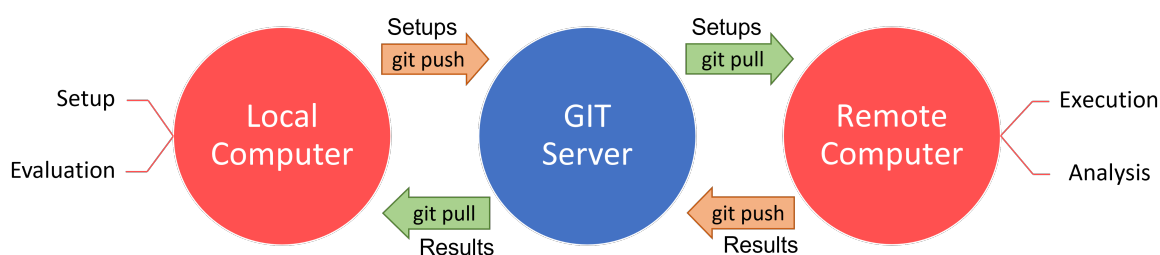


Figure 2.2.: Illustration of the workflow with remote execution.

2.5. Example Simulations

Exemplary simulations with results are provided in the folder 'exampleSimulations'. A brief description of each example is given below:

- **Example 1: argon**

Argon is simulated as a LJ fluid at a moderate temperature in the NVT ensemble.

- **Example 2: carbon dioxide**

Carbon dioxide is simulated as a partially charged rigid molecule in the NVT ensemble. The electrostatic long-range correction is calculated with Ewald summation.

- **Example 3: ethane and methane**

A binary mixture of ethane and methane is simulated in the NPT ensemble. Ethane is modeled as a rigid molecule with two centers and methane is simulated as an LJ fluid. Insertions are conducted using the identity swap method of Panagiotopoulos. (see section 4.3)

- **Example 4: argon with CFC**

Argon is simulated with a setup similar to example 1, but insertions are conducted using the continuous fractional component (CFC) method. (see section 4.4)

- **Example 5: methanol with OPLS-AA**

Methanol is simulated using the OPLS-AA force field model. This simulation includes trial moves for intramolecular constraints.

2.6. Literature

The literature is stored in the TUGraz-Cloud as well as [the GIT-Lab server](#) and maintained by Thomas Wallek. In the literature folder, relevant publications are collected. It might be overwhelming at first sight which is why selected publications are assembled in the folder '_Introduction'. These are briefly described below:

- **Documentation**

The most recent version of the present report which documents the software package and provides some background.

- **Panagiotopoulos 1987 [28]**

One of the first papers laying out the concept of Gibbs ensemble (GE) simulations of phase equilibria. It is a good start and introduction to the GEMC technique.

- **Metropolis 1953 [25]**

A rather theoretical paper on the principle of the Metropolis algorithm itself. Not a must-read, yet (probably) interesting from a conceptual point of view.

- **Chen 1998 [4]**

In this paper different force fields for alkanes or simple organic liquids are described.

- **Fernandes 2015 [42]**

This paper gives a quick overview of the concepts in Gibbs ensemble (GE) simulations and describes the implementation of an algorithm in a Java based program.

3. Theoretical Background

In this chapter, basic principals behind Monte Carlo simulations and the Gibbs ensemble are explained. It should provide a basic understanding before going deeper into the methods used in this software package. The principles of Monte Carlo simulation and the rejection technique using the Metropolis algorithm are explained. Furthermore, the criteria for the simulation of phase equilibria and the intermolecular potential are presented before the unique simulation setup of the Gibbs ensemble is explained.

3.1. The Principle of Monte Carlo

Monte Carlo (MC) simulations approximate results on the basis of sampling random numbers. A commonly used, illustrative example is the determination of π : One can either use mathematical, analytical derivations to derive its exact value or draw a circle with a circumscribed square. The ratio of the area of the circle to the area of the square is $\frac{\pi}{4}$. Therefore, one could, for example, count the number of raindrops that hit the area of the circle and the total number of raindrops. Assuming that the positions of the raindrops are uniformly distributed, the ratio of the number of the raindrops converges against the true value of $\frac{\pi}{4}$ for sufficiently large numbers.[2]

Certainly there is no reasonable purpose for using a MC simulation for the determination of π . Yet, if one considers that there is no general analytical solution for the "three-body problem" [11], it is obvious that an analytical solution for molecular systems in the orders of 10^2 molecules is hardly possible. Therefore, MC simulations are used to generate molecular configurations that simulate phase equilibria. By averaging a sufficiently large number of states, bulk properties can be determined on the basis of statistical thermodynamics.

3.2. Metropolis Algorithm $M(RT)^2$

The goal of a Monte Carlo simulation in statistical thermodynamics is to create molecular configurations that are distributed around the equilibrium. For this reason, the configurations are created following a Markov chain which will yield the desired distribution if it is long enough. Methods to create such chains include rejection techniques.

Rejection techniques are Monte Carlo (MC) methods of particular interest due to their very general formulation, whereby a wide variety of problems can be described. Its basic principle is the sampling of a trial value. This trial is proposed and subject to one or several tests which might involve the sampling of one or more other random variables.[21]

Based on these tests, the proposed change is accepted or rejected. If the trial value is rejected, the test is repeated. Thus, the normalization of the probability distribution need not be explicitly known for the sampling process. A significant disadvantage, however, is that rejection techniques have a low efficiency in case the acceptance rate of the proposed changes is low.

A rejection technique of particular interest is the Metropolis algorithm. It is motivated by the behavior of stochastic systems, like ensembles in statistical mechanics, to converge asymptotically against their equilibrium: Two neighboring states near the equilibrium have to be equally likely to move from one to another. Thus, the probability of a successful transition from X to Y $K(Y|X)$ times the probability of finding such a state $f(X)$ has to equal the probability of moving from Y to X $K(X|Y)$ times the probability of finding such a state $f(Y)$. This relationship is formulated in the detailed balance stated in Equation 3.2.1.

$$K(X|Y)f(Y) = K(Y|X)f(X) \quad (3.2.1)$$

The probability of observing a particular state $f(X)$ is determined iteratively in a random walk by accepting or rejecting the proposed trial moves: $f(X) = \lim_{n \rightarrow \infty} \Phi_n(X)$. For a successful application of the $M(RT)^2$ method an average acceptance probability of approximately 50 % is recommended. The reason is that the performance of the algorithm decreases the more proposed trial moves are rejected. Thus, possibly poor performance of the algorithm might be a significant drawback of the method. Another drawback is that there might be a positive

correlation between the successive states which results in a higher variance of the calculated ensemble averages.[21]

3.3. Simulation of Phase Equilibria

The fundamental Gibbs criteria of phase equilibrium in a pure component system are stated in Equation 3.3.1. Simultaneously, systems in equilibrium show maximum entropy, which corresponds to a minimal internal energy, as stated in Equation 3.3.2.

$$T_V \stackrel{!}{=} T_L \Leftrightarrow p_V \stackrel{!}{=} p_L \Leftrightarrow \mu_V \stackrel{!}{=} \mu_L \quad (3.3.1)$$

$$\{S \stackrel{!}{=} \max\}_{U,V,N=\text{const.}} \Leftrightarrow \{U \stackrel{!}{=} \min\}_{S,V,N=\text{const.}} \quad (3.3.2)$$

This fundamental concept is the basis for the calculation of phase equilibria with the Metropolis method. After a trial move is proposed, it is accepted or rejected based on the change of internal energy as stated in Equation 3.3.4. In Equation 3.3.3, the change of the internal energy from the old configuration (o) to the new configuration (n) is calculated. The internal energy of all interacting molecules in the system is calculated using an adequate potential function $\nu(r_{ij})$. While different potentials like the square-well potential for example have been proposed in literature, the commonly used Lennard-Jones (LJ) potential will be discussed in more detail in section 3.4.[1]

$$\Delta\nu_{o \rightarrow n} = \sum_{j=1}^N \nu(r_{j,n}) - \sum_{j=1}^N \nu(r_{j,o}) \quad (3.3.3)$$

$$\frac{p_n}{p_o} = \min[1, \exp(-\beta \Delta\nu_{o \rightarrow n})] \quad , \text{ with } \beta = \frac{1}{k_B \cdot T} \quad (3.3.4)$$

The proposed move is always accepted if the calculated energy difference is smaller than zero. If the potential energy of the proposed state is, however, greater than the previous state, the acceptance probability is compared against a random number ξ_i which is uniformly distributed in the range $[0, 1]$ as shown in Figure 3.1. In case ξ_i is greater than the calculated acceptance probability the move is rejected.

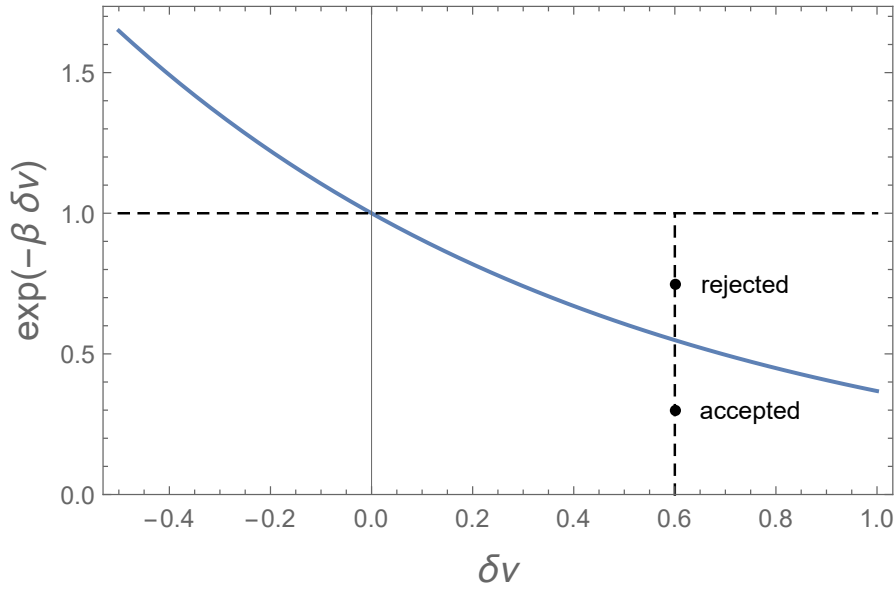


Figure 3.1.: "Accepting uphill moves in the MC simulation" [1]

3.4. Lennard-Jones Potential

The commonly used Lennard-Jones (LJ) potential is described in Equation 3.4.1. It consists of an attractive and a repulsive part. These two parts are modeled as the ratio of the size parameter σ and the actual distance r_{ij} between two LJ molecules, raised to the power of 12 and 6, respectively. The potential is scaled with the energy parameter ϵ . The parameters can be determined either by fitting experimental data, e.g. virial coefficients, or using "ab-initio" methods such as quantum mechanics.[1]

$$\nu_{\text{LJ}}(\epsilon_{ij}, \sigma_{ij}, r_{ij}) = 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] \quad (3.4.1)$$

The Lennard-Jones potential can easily be separated as proposed by Weeks et al. 1971 [50] by splitting the potential at the minimum as displayed in Figure 3.2. Thereby, the potential function can be represented by a repulsive (ν_{RJ}) and an attractive potential (ν_{AJ}).[1]

In literature, simulation results are often normalized with the LJ parameters as stated in Equation 3.4.2. These normalized values, denoted with an asterisk, will also be referred to as the '*LJ reduced*' values. They are used to compare the results of LJ fluids. The indices

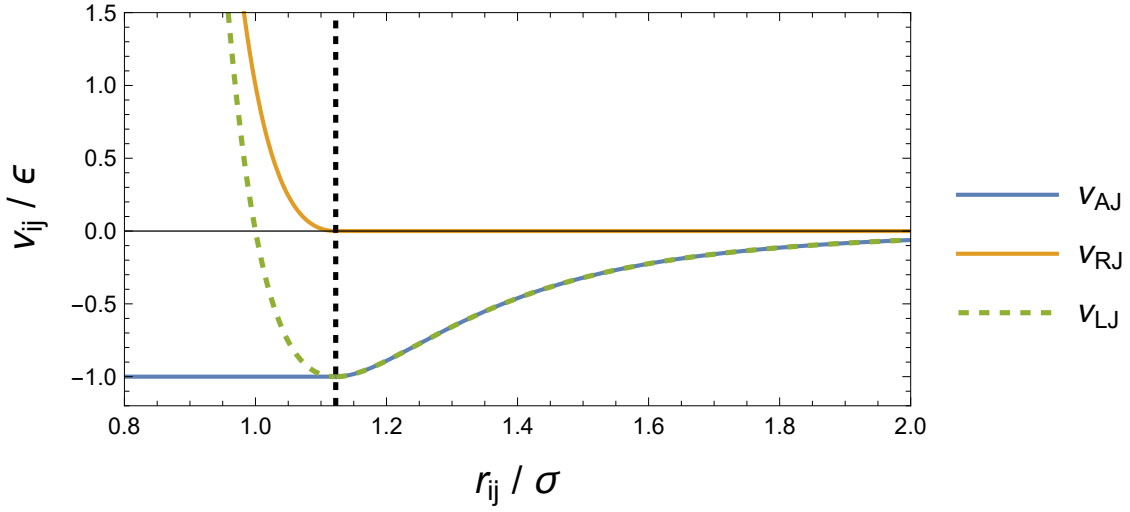


Figure 3.2.: "The separation of the Lennard-Jones potential ν_{LJ} into attractive and repulsive components, ν_{AJ} and ν_{RJ} , respectively. The vertical dashed line shows the position of r_{\min} ." [1]

indicate the interacting species. For mixtures, it is assumed that the component with the larger σ value is the first species in the system. The standard reduced values are calculated with respect to the critical point of the pure component.

$$\begin{aligned}
 T^* &= k_B T / \epsilon_{11} &\Leftrightarrow & T_{\text{red}} = T / T_C \\
 P^* &= P \cdot \sigma_{11}^3 / \epsilon_{11} &\Leftrightarrow & P_{\text{red}} = P / P_C \\
 \rho^* &= \rho \cdot \sigma_m^3 &\Leftrightarrow & \rho_{\text{red}} = \rho / \rho_C \\
 U^* &= U / \epsilon_m &\Leftrightarrow & U_{\text{red}} = U / U_C
 \end{aligned} \tag{3.4.2}$$

The reduced density and internal energy are calculated using mixture parameters according to the van der Waals one-fluid approximation. The mixture parameters can be calculated using Equation 3.4.3 and Equation 3.4.4 which are derived in the work of Harismiadis 1991.[16]

$$\sigma_m^3 = \sum_i \sum_j x_i x_j \sigma_{ij}^3 \tag{3.4.3}$$

$$\epsilon_m = \frac{1}{\sigma_m^3} \sum_i \sum_j x_i x_j \epsilon_{ij} \sigma_{ij}^3 \tag{3.4.4}$$

3.5. Gibbs Ensemble

In 1987 the Gibbs ensemble (GE) was proposed by Panagiotopoulos for the calculation of phase equilibria. It is based on a Monte Carlo (MC) simulation of a two-region system whereby vapor-liquid equilibria (VLE) can be calculated. While each region itself is equilibrated by spatial displacements, the pressures of the two boxes are equalized by volume changes and the chemical potential is equilibrated by random transfers of molecules between the two boxes. These three so-called 'trial moves' are illustrated in Figure 3.3.[28]

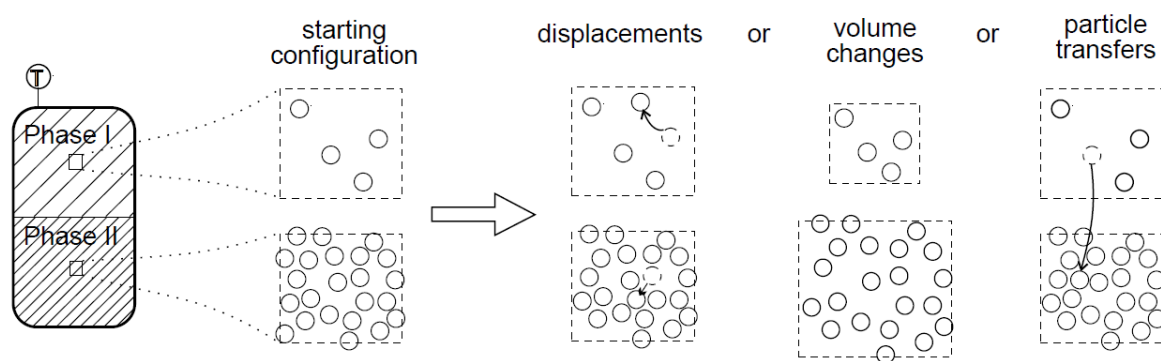


Figure 3.3.: Basic idea behind the Gibbs ensemble Monte Carlo method [30]

The main advantages of the Gibbs method are that the coexisting properties of a system with an arbitrary number of components can be determined directly from one computer simulation and that relatively small systems consisting of a few hundred molecules yield accurate results. The reason for this is that periodic boundary conditions are applied for each phase separately. The ensemble can also be easily extended for constant pressure simulations in which the equilibria of mixtures can be simulated conveniently.[28, 31]

However, the kinetic energy contribution cannot be taken into account directly since no information on the velocity of the molecules is obtained.[37] Probably the most significant drawback of the Gibbs method is that the acceptance rate of molecule transfers is low for insertions in dense phases, especially if components with a large difference in molecular sizes are simulated.[29, 30]

4. Methods

In the following chapter the methods implemented in the present software package are laid out. The focus of this chapter lies on a precise description of the methodological concepts. More details are given in chapter 5 where all relevant aspects of the implementation are discussed.

4.1. Energy Calculation

To calculate various properties and to check if the trial moves should be accepted or rejected the interactions between all the molecules and their atoms have to be calculated. How these particles interact with each other and the way those interactions result in a specific total energy is defined by a force field. In this code, the Optimized Potentials for Liquid Simulations (OPLS) force field is used.[20]

4.1.1. OPLS Force Field

The total energy of a system is split up into four parts as shown in Equation 4.1.1.

$$E_{\text{total}} = E_{\text{nonbonded}} + E_{\text{bond}} + E_{\text{angle}} + E_{\text{torsion}} \quad (4.1.1)$$

The first term, $E_{\text{nonbonded}}$ describes the interactions between non-bonded atoms. It can be separated into an intermolecular and an intramolecular part as shown in Equation 4.1.2. The intermolecular part of $E_{\text{nonbonded}}$ is calculated with Equation 4.1.3.

$$E_{\text{nonbonded}} = E_{\text{nonbonded,inter}} + E_{\text{nonbonded,intra}} \quad (4.1.2)$$

$$E_{\text{nonbonded,inter}} = \sum_a^N \sum_{b>a}^N \sum_i^a \sum_j^b \delta(r_{ij}) \cdot U_{ij}(r_{ij}) \quad (4.1.3)$$

$$\delta(r_{ij}) = \begin{cases} 1 & \text{for } r \leq r_c \\ 0 & \text{for } r > r_c \end{cases} \quad (4.1.4)$$

$$U_{ij}(r_{ij}) = \underbrace{4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right]}_{\text{Lennard-Jones potential}} + \underbrace{\frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}}}_{\text{Coulomb's law}} \quad (4.1.5)$$

The four sums in Equation 4.1.3 represent the sum of all intermolecular interaction pairs where N is the total number of molecules, a and b are specific molecules and i and j are specific atoms of those molecules. The notation $b > a$ is used to prevent double counting of interaction pairs. The term $U_{ij}(r_{ij})$ is the non-bonded energy between atom i and j and is a combination of the Lennard-Jones potential and Coulomb's law, as shown in Equation 4.1.5. In Equation 4.1.5, ϵ_{ij} and σ_{ij} are parameters of the Lennard-Jones potential, q_i and q_j are the charges for Coulomb's law and r_{ij} is the distance between the two atoms i and j . The constant ϵ_0 is the vacuum permittivity and the constant e is the elementary charge. The Dirac delta function $\delta(r_{ij})$, described in Equation 4.1.4, indicates that only interactions lying within the cut-off radius r_c are considered directly. The cut-off potential is considered with a tail-correction (see section 4.1.4).

The intramolecular part of $E_{\text{nonbonded}}$ is calculated with Equation 4.1.6, where M represents atom pairs separated by three or more bonds. The notation $j > i$ is used to prevent double counting of interaction pairs. The scaling factor f_{ij} is equal to 0.5 if the atom pair is separated by exactly three bonds and 1 otherwise.

$$E_{\text{nonbonded,intra}} = \sum_a^N \sum_i^a \sum_{j>i}^a f_{ij} \cdot U_{ij}(r_{ij}) \quad (i, j) \in M \quad (4.1.6)$$

The second term, E_{bond} , calculated with Equation 4.1.7, describes the energy resulting from

stretching of intramolecular bonds which are defined by the distance r between two atoms.

$$E_{\text{bond}} = \sum_{\text{bonds}} K_r (r - r_{eq})^2 \quad (4.1.7)$$

The third term, E_{angle} , calculated with Equation 4.1.8, describes the energy resulting from bending of angles of the intramolecular bonds which are defined by the angle θ between two bonds.

$$E_{\text{angle}} = \sum_{\text{angles}} K_{\theta} (\theta - \theta_{eq})^2 \quad (4.1.8)$$

Both E_{bond} and E_{angle} include a factor K_r or K_{θ} and an equilibrium constant r_{eq} or θ_{eq} , respectively, which have certain values depending on the position in the molecule. The equilibrium constant defines the state at which the bond stretching or angle bending results in the lowest intramolecular energy (which is 0).

The last term, E_{torsion} , calculated with Equation 4.1.9, describes the energy resulting from the rotation of intramolecular bonds. The torsion is expressed with a dihedral angle ϕ defined by four atoms. The variables V_1 , V_2 , V_3 , f_1 , f_2 and f_3 are parameters specific to a dihedral of a molecule.

$$E_{\text{torsion}} = \sum_i \left(\frac{V_1^i}{2} [1 + \cos(\phi_i + f_1)] + \frac{V_2^i}{2} [1 - \cos(2\phi_i + f_2)] + \frac{V_3^i}{2} [1 + \cos(3\phi_i + f_3)] \right) \quad (4.1.9)$$

4.1.2. Interaction Parameters

The interaction parameters for the Lennard-Jones potential between two atoms i and j are calculated with combining rules that use the OPLS parameters of those specific atoms. The

default combining rules used with the OPLS force field [20] are Eqs. 4.1.10 and 4.1.11.

$$\sigma_{ij} = \sqrt{\sigma_i \sigma_j} \quad (4.1.10)$$

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \quad (4.1.11)$$

Other commonly used combining rules are the Lorentz-Berthelot combining rules listed in Equation 4.1.12 and 4.1.13.

$$\sigma_{ij} = \frac{\sigma_i \sigma_j}{2} \quad (4.1.12)$$

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \quad (4.1.13)$$

A modified version of the Lorentz-Berthelot combining rules introduces the parameters ξ_{ij} and ζ_{ij} for deviations of the size and energy cross parameters.[44] The resulting combining rules are listed in Eqs. 4.1.14 and 4.1.15, which can be implemented if needed.

$$\sigma_{ij} = \zeta_{ij} \frac{\sigma_i \sigma_j}{2} \quad (4.1.14)$$

$$\epsilon_{ij} = \xi_{ij} \sqrt{\epsilon_i \epsilon_j} \quad (4.1.15)$$

4.1.3. Periodic Boundary Condition

Because molecular structures on a macroscopic scale cannot be simulated due to computational speed restrictions and wall or surface interactions are difficult to model, periodic boundaries are used to approximate bulk fluids. This means that copies of one simulation box, in which all trial moves are performed, are arranged in an infinitely large lattice. If an element passes through one side of the simulation box, it enters through the opposite side of the box, which is illustrated in Figure 4.1.

The periodic boundary condition (PBC) is used in the calculation of distances between atoms and in trial moves that change the position of the atoms because new positions can lie outside of the simulation box. The basic equation used to ensure that the distances between two

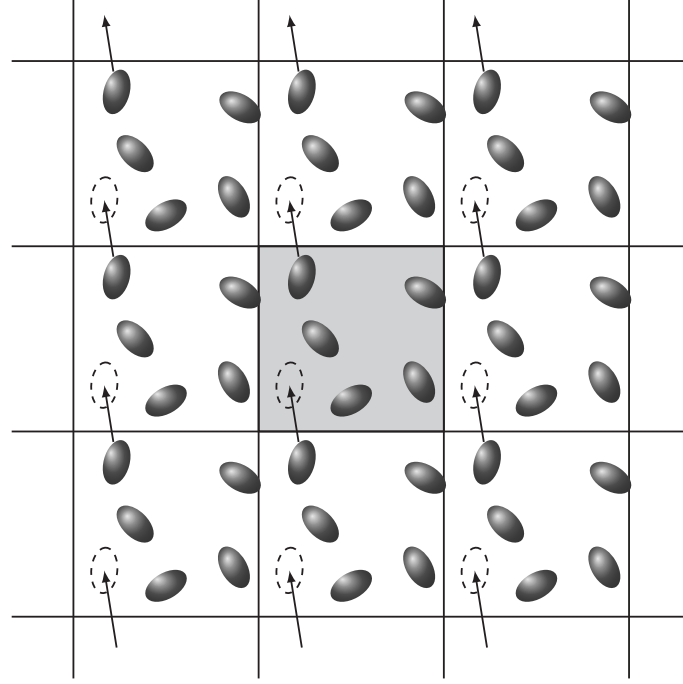


Figure 4.1.: Periodic Boundary Condition [1]

atoms satisfy the PBC is displayed in Equation 4.1.16.

$$\vec{r}_{ij} = \begin{bmatrix} x_i - x_j \\ y_i - y_j \\ z_i - z_j \end{bmatrix} \quad r_{ij} = \left\| \vec{r}_{ij} - L_{\text{box}} \cdot \text{round} \left[\frac{\vec{r}_{ij}}{L_{\text{box}}} \right] \right\| \quad (4.1.16)$$

First, \vec{r}_{ij} is calculated, which is the vector between the two atoms i and j in the simulation box. Then, parts of the vector which are larger than half of the box length (L_{box}) are reduced by the box length before normalizing the vector to get the distance. In this way, the atoms can interact with virtual copies of the simulation box. If, for example, an atom has its position on the right side of the simulation box and another atom has its position on the left side of the box, they would lie right next to each other according to the PBC.

4.1.4. Cut-off Radius and Tail Correction

The use of an infinitely large lattice would require the calculation of an infinite number of interactions between all the atoms in the system. To overcome this problem, the calculations of the interaction energies between two atoms are only evaluated for interaction distances below a certain value. This value is called the cut-off radius, r_c , which defines the distance at which Equation 4.1.5 is truncated.

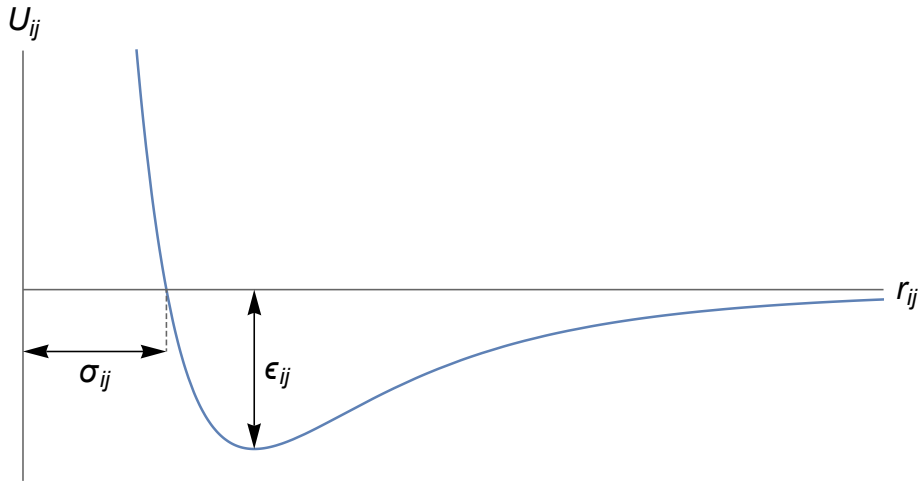


Figure 4.2.: Lennard-Jones potential with the resulting energy U_{ij} between two atoms with the distance r_{ij} (σ_{ij} and ϵ_{ij} are the Lennard-Jones parameters)

In the example of the Lennard-Jones potential shown in Figure 4.2, it is clear that with this method the calculation of the total amount of energy lacks a small amount which would be contributed by interactions from wider distances. To correct this lack of energy, a so-called tail correction is used and added to the total energy which was calculated with the cut-off. This is shown in Equation 4.1.17, where E_{Cut} is the energy with cut-off and E_{Tail} is the tail correction. The tail correction needs to be taken into account for all trial moves when performing the Metropolis checks.

$$E = E_{\text{Cut}} + E_{\text{Tail}} \quad (4.1.17)$$

It is important to note that the accuracy of the analytical tail correction depends on the cut-off radius, as displayed in Figure 4.3. At the same time, a large cut-off radius, especially for systems consisting of a large number of molecules, increases the calculation effort significantly.

The maximum cut-off radius is half the box length to avoid problems with the periodic boundary condition. The cut-off radius should at least satisfy the rule $r_c/\sigma_{ij} > 2$ [19].

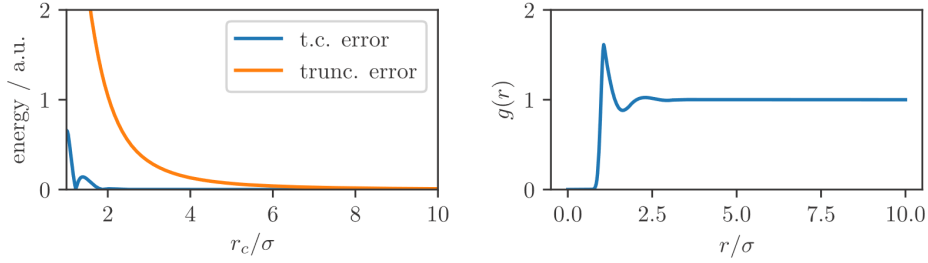


Figure 4.3.: "Error for truncated potential (trunc. error) and truncation with tail correction (t.c. error) for a Lennard-Jones fluid." [19]

The principle behind the tail correction is to estimate the number of atoms at each distance from certain atoms and to sum up all these interactions. The base of the tail correction is Equation 4.1.18, describing the amount of atoms distributed between the two radial distances r_1 and r_2 from a certain atom, where ρ is the density of that atom type in the system. The function $g(r)$ is called *radial distribution function* and describes the density of atoms at certain distances away from a reference atom.

$$N(r_1, r_2) = 4\pi\rho \int_{r_1}^{r_2} r^2 g(r) dr \quad (4.1.18)$$

The next step is to include the calculation of the interaction energy into Equation 4.1.18 and to integrate between the cut-off radius and infinity. The resulting Equation 4.1.19 is the tail correction for one atom interacting with every other atom in the system.

$$U_{\text{Tail}} = 4\pi\rho \int_{r_c}^{\infty} u(r) r^2 g(r) dr \quad (4.1.19)$$

To correct the interaction energy of every atom in the simulation box, first, the equation is multiplied by the total number of atoms, N . Second, to avoid counting every interaction twice, the equation is also divided by 2. This results in Equation 4.1.20.

$$U_{\text{Tail}} = 2\pi N\rho \int_{r_c}^{\infty} u(r) r^2 g(r) dr \quad (4.1.20)$$

In the case of tail corrections, the radial distribution function is estimated to be $g(r) = 1$

at every radius larger than the cut-off radius. If Equation 4.1.20 is then integrated with $g(r) = 1$ and $u(r)$ as the Lennard-Jones potential, the results is Equation 4.1.21.[9]

$$U_{\text{Tail}} = \frac{8}{3}\pi N\rho\epsilon\sigma^3 \left[\frac{1}{3} \left(\frac{\sigma}{r_c} \right)^9 - \left(\frac{\sigma}{r_c} \right)^3 \right] \quad (4.1.21)$$

Because the pressure is also calculated from the interaction energies, the same procedure must also be used to calculate a tail correction for the pressure. The integral form of this is shown in Equation 4.1.22 and can be derived from Equation 4.1.18 with the same steps described for the energy tail correction. Note that the factor $1/3$ of the virial term is included in this derivation, which leads to the factor $2/3$ in Equation 4.1.22.

$$P_{\text{Tail}} = \frac{2}{3}\pi N\rho \int_{r_c}^{\infty} -\frac{\partial u(r)}{\partial r} r^3 g(r) dr \quad (4.1.22)$$

If again the estimation of $g(r) = 1$ is used and the equation is integrated with the Lennard-Jones potential, the result is Equation 4.1.23.[9] Subsequent integration results in the equation for the calculation of pressure, Equation 4.1.24.

$$P_{\text{Tail}} = \frac{16}{3}\pi N\rho\epsilon\sigma^3 \left[\frac{2}{3} \left(\frac{\sigma}{r_c} \right)^9 - \left(\frac{\sigma}{r_c} \right)^3 \right] \quad (4.1.23)$$

$$P = \frac{Nk_{\text{B}}T}{V} + \frac{1}{V} \frac{1}{3} \sum_a^N \sum_{b>a}^N \sum_i^a \sum_j^b \left[-\frac{\partial U_{ij}(r_{ij})}{\partial r_{ij}} \frac{\langle r_{ab}, r_{ij} \rangle}{|r_{ij}|} \right] + \frac{P_{\text{Tail}}}{V} \quad (4.1.24)$$

In the case of mixtures and molecules with different atoms, the tail corrections have to be adjusted to include all different atom types. Equation 4.1.18 describes the number of interacting atoms of one type. If the final form of the tail correction is used (Equation 4.1.21), N denotes the reference atom and ρ the interacting atoms in the bulk system. Taking the example of the binary mixture of Argon and Krypton, there has to be a correction for the pairs Argon-Argon, Argon-Krypton, Krypton-Argon and Krypton-Krypton, where the first atom type of those pairs is the reference atom and the second one represents the interacting

atoms in the bulk phase.

$$U_{\text{Tail}} = \frac{8}{3}\pi \sum_{x \in M} \sum_{y \in M} N_x \rho_y \epsilon_{xy} \sigma_{xy}^3 \left[\frac{1}{3} \left(\frac{\sigma_{xy}}{r_c} \right)^9 - \left(\frac{\sigma_{xy}}{r_c} \right)^3 \right] \quad (4.1.25)$$

$$P_{\text{Tail}} = \frac{16}{3}\pi \sum_{x \in M} \sum_{y \in M} N_x \rho_y \epsilon_{xy} \sigma_{xy}^3 \left[\frac{2}{3} \left(\frac{\sigma_{xy}}{r_c} \right)^9 - \left(\frac{\sigma_{xy}}{r_c} \right)^3 \right] \quad (4.1.26)$$

The sums in Eqs. 4.1.25 and 4.1.26 account for all atom pairs. In this case, M is the set of all available atom types, x is the type of the reference atom and y is the type of the interacting atoms.

4.1.5. Overlap Radius

A common time-saving method is to reject attempted trial moves in case the proposed new position of the molecule is too close to another molecule. This is done by checking the distance between two interacting atoms against the chosen overlap radius r_{Ov} . In case the distance between the two atoms is smaller than the overlap radius, the energy calculation is immediately aborted and the proposed trial move is rejected. The underlying idea is that such positions are not plausible due to strong repulsive forces. Simultaneously, such positions are very unlikely to be accepted, as can be seen in Figure 4.2.[1]

In the present implementation, the overlap check is performed during all trail moves in which the position of a single molecule is changed. This includes translations, rotations and insertions. The overlap rejection is not performed during volume changes, as a test revealed that overlaps rarely occur in rejected trial moves. Therefore, it is not reasonable to make an early rejection in volume changes due to overlap.

4.2. Determination of Pressure and Chemical Potential

In the following section, the calculations of the box pressure and the chemical potential are described.

4.2.1. Pressure Calculation

The pressure of a bulk fluid in molecular simulations can be calculated using the *virial theorem*. According to Equation 4.2.1 the calculation includes an ideal (gas) contribution and a residual contribution due to intermolecular interactions.[26] Like in the energy calculation, the four sums account for all intermolecular pair interactions. The negative partial derivative with respect to the distance between two atoms represents the force between those two atoms, where $U_{ij}(r_{ij})$ is the intermolecular energy and r_{ab} describes the distance between the center of two molecules a and b. With the Lennard-Jones potential and Coulomb's law, the force is calculated with Equation 4.2.2.

$$P = \frac{Nk_B T}{V} + \frac{1}{V} \frac{1}{3} \sum_a^N \sum_{b>a}^N \sum_i^a \sum_j^b \left[-\frac{\partial U_{ij}(r_{ij})}{\partial r_{ij}} \cdot r_{ij} \cdot \frac{\langle r_{ab}, r_{ij} \rangle}{|r_{ij}|^2} \right] \quad (4.2.1)$$

$$-\frac{\partial U_{ij}(r_{ij})}{\partial r_{ij}} = \underbrace{48\epsilon_{ij} \frac{\sigma_{ij}^{12}}{r_{ij}^{13}} - 24\epsilon_{ij} \frac{\sigma_{ij}^6}{r_{ij}^7}}_{\text{Force from Lennard-Jones interactions}} + \underbrace{\frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}^2}}_{\text{Force from electrostatic interactions}} \quad (4.2.2)$$

The additional term involving r_{ab} accounts for the correction of poly-atomic molecule interactions and equals 1 if only LJ fluids are in the system. It is important to note that, at the end of the simulation, only the ensemble average of the pressure can be used as the result. The ensemble average is the average over all cycles after the system has reached equilibrium and only fluctuates around certain values. More information about this step is described in section 5.4.3. Because of this, not only the interaction energies of every cycle are saved for later analysis, but also the virial terms of every cycle that consist of the contribution from intermolecular interactions without the division by the volume, as stated in Equation 4.2.1.

4.2.2. Calculation of the Chemical Potential

The chemical potential is determined with the Widom test particle insertion method. First, a random position is sampled in one of the two boxes. Then, the energy change due to the insertion of a molecule at the proposed position is calculated, given that the sampled position is not rejected because of an overlap. Based on this energy, ΔE , the total chemical potential

μ_i can be calculated with Equation 4.2.3. The formula for the total chemical potential can be split into an ideal, also referred to as the ideal mixture or ideal gas, contribution and an excess contribution. The ideal contribution, μ_i^{id} , describes the chemical potential of an ideal gas with the same system properties and is calculated using the current number density as well as the thermal de Broglie wavelength λ_i . The excess contribution, μ_i^{E} , is calculated with the energy change due to the inserted molecule. The term inside the angled brackets $\langle \dots \rangle$ needs to be averaged over multiple sampled states and will be referred to as the partial contribution.[51, 9]

$$\mu_i = \mu_i^{\text{id}} + \mu_i^{\text{E}} = -k_{\text{B}}T \ln \left[\underbrace{\left\langle \frac{V}{\lambda_i^3 \cdot (N_i + 1)} \right\rangle}_{\text{ideal}} \underbrace{\exp \left[-\frac{\Delta E}{RT} \right]}_{\text{excess}} \right] \quad (4.2.3)$$

4.3. Improved Insertion Algorithm for Binary Mixtures

The acceptance rate of conventional GE transfers is low for dense phases and for simulating molecules with large size differences. This is indicated by the orange boundary in Figure 4.4a. To overcome this problem, Panagiotopoulos proposed a modification for binary mixtures: Instead of inserting the larger molecule at a randomly sampled position, it is exchanged with a molecule of the smaller species, as shown in Figure 4.4b.[29]

The acceptance rule for a conventional GE transfer is given in Equation 4.3.1, where i designates the index of the transferred species, $\Delta\nu_{\text{o} \rightarrow \text{n}}^{I,II}$ is the energy change in the respective box, N_i is the number of molecules of the species in the respective box and V the total volume of the respective box. In Equation 4.3.2, the acceptance rule for the exchange step is given. By comparing it to Equation 4.3.1 it can be seen that it comprises two transfer steps that take place simultaneously.[29]

$$\frac{p_{\text{n}}}{p_{\text{o}}} = \min \left[1, \frac{N_i^{II} + 1}{N_i^I} \frac{V^I}{V^{II}} \exp \left[-\beta \left(\Delta\nu_{\text{o} \rightarrow \text{n}}^I + \Delta\nu_{\text{o} \rightarrow \text{n}}^{II} \right) \right] \right] \quad (4.3.1)$$

$$\frac{p_{\text{n}}}{p_{\text{o}}} = \min \left[1, \left(\frac{N_1^I}{N_1^I + 1} \frac{V^{II}}{V^I} \right) \left(\frac{N_2^{II} + 1}{N_2^I} \frac{V^I}{V^{II}} \right) \exp \left[-\beta \left(\Delta\nu_{\text{o} \rightarrow \text{n}}^I + \Delta\nu_{\text{o} \rightarrow \text{n}}^{II} \right) \right] \right] \quad (4.3.2)$$

Another important aspect of the modified transfer step is the equilibration of the chemical potential. In the conventional GE method, where each species is transferred individually,

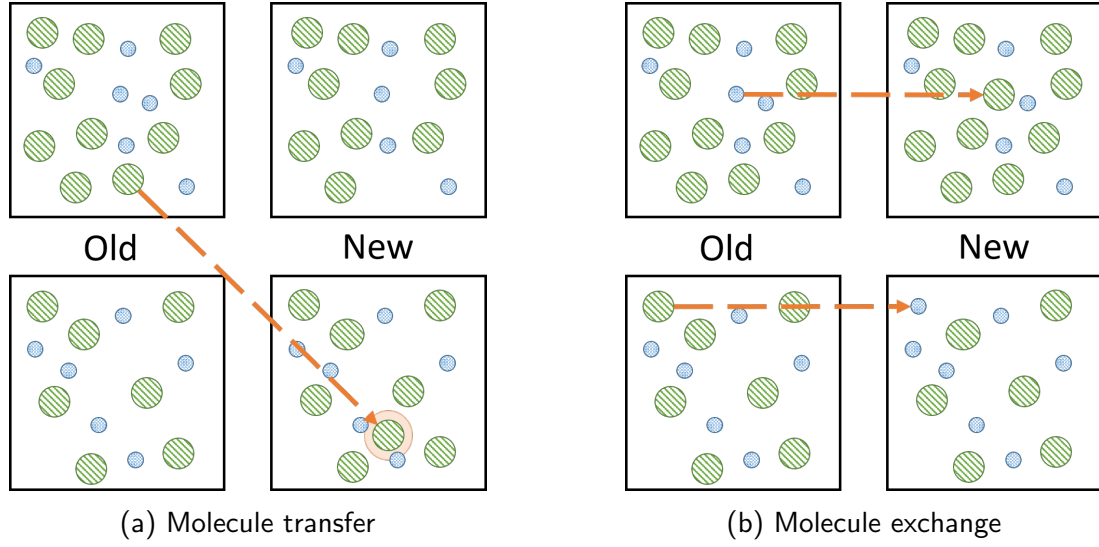


Figure 4.4.: Schematic illustration of the exchange step for highly asymmetric mixtures.[29]
The blue and green spheres represent different species.

the chemical potential can equilibrate individually as well. Since this method for insertions includes both components for transfers of the bigger species, the equilibration of the chemical potential is slightly different, according to Eqs. 4.3.3 and 4.3.4.[29]

$$\mu_2^I = \mu_2^{II} \quad (4.3.3)$$

$$\mu_2^I - \mu_1^I = \mu_2^{II} - \mu_1^{II} \quad (4.3.4)$$

Since the smaller species is still transferred directly, its equilibration is not affected by the modified swap algorithm. By contrast, the direct equilibration of the larger species is based on the equilibration of the difference in the chemical potential of the two species.[29]

4.4. Gibbs Ensemble Continuous Fractional Component

The purpose of the Gibbs ensemble continuous fractional component (GECFC) method is an increase of the efficiency of molecule transfers between the two boxes. This is especially important at low temperatures, dense phases and large differences in the molecular sizes, since the acceptance probability is low due to overlaps. This issue is overcome by replacing the conventional molecule transfer, where a full molecule is moved directly from one box to

a randomly sampled position in the other box, with a gradual insertion and deletion.

The GECFC method proposed by Ali Poursaeidesfahani in the working group of Thijs J. H. Vlugt is similar to the original method by Wei Shi in the working group of Edward Maginn.[41] However, only one fractional is used per component instead of using two coupled ones. Thus, the fractional can only be present in one of the two boxes, and an additional third GECFC trial move is introduced. This trial move transfers the fractional from one box to the other one. Another significant difference is that the type of the GECFC-trial move to be executed is randomly chosen and independent of the current value of λ .

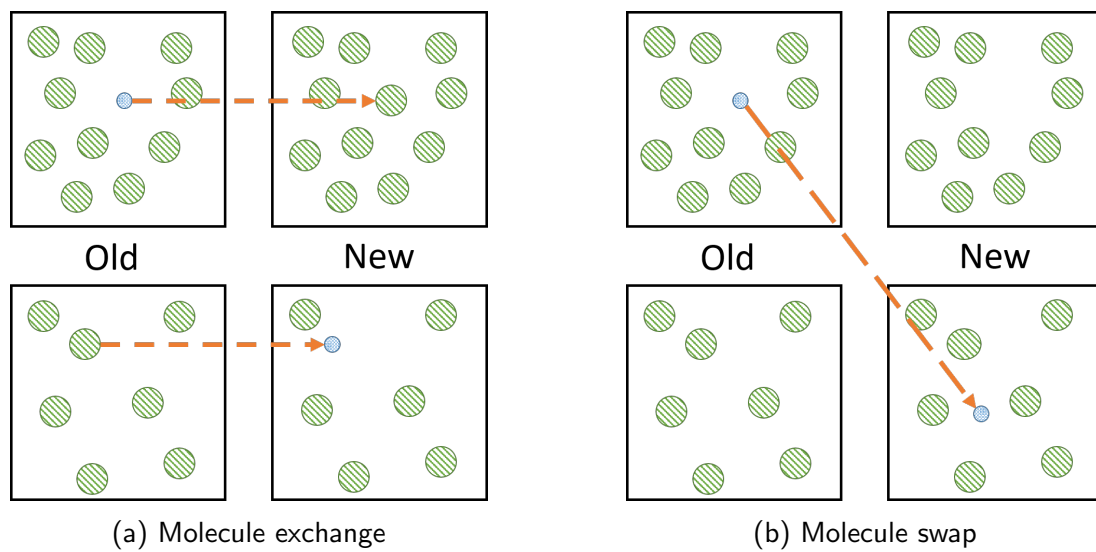


Figure 4.5.: Schematic illustration of the molecule exchange and molecule swap trial moves of the GECFC method.[36] The green spheres represent full molecules and the blue sphere represents the fractional.

4.4.1. Scaled LJ Potential

To facilitate gradual insertions and deletions, a new state variable λ is introduced. It controls the interactions of the so-called 'fractional' molecule, which is added to the system before the simulation is started. In Eqs. 4.4.2 and 4.4.3, the scaled potential function and the virial term are stated. Although other choices are possible, a commonly used value for ξ is 0.5. Internal

energy contributions like bond stretching, angle bending or torsion are not scaled.[41, 36]

$$A(r_{ij}, \lambda) \equiv \xi(1 - \lambda)^2 + \left(\frac{r_{ij}}{\sigma_{ij}}\right)^6 \quad (4.4.1)$$

$$\nu(r_{ij}, \lambda) = 4\lambda\epsilon_{ij} \cdot \left(\frac{1}{A^2 - A}\right) \quad (4.4.2)$$

$$-\frac{\partial \nu(r_{ij}, \lambda)}{\partial r_{ij}} = 4\lambda\epsilon_{ij} \cdot \left(\frac{12}{A^3} \frac{r_{ij}^5}{\sigma_{ij}^6} - \frac{6}{A^2} \frac{r_{ij}^5}{\sigma_{ij}^6}\right) \quad (4.4.3)$$

The two main features of this mathematical form are the following: First, the potential is equal to the unaltered LJ potential for $\lambda = 1$; thus, the potential continuously converges against this boundary value for increasing values of λ . Second, the adjusted potential function yields finite values for very close interactions even at distances close to zero at low values of λ , such as $\lambda = 0.1$. Both features can be seen in Figure 4.6.

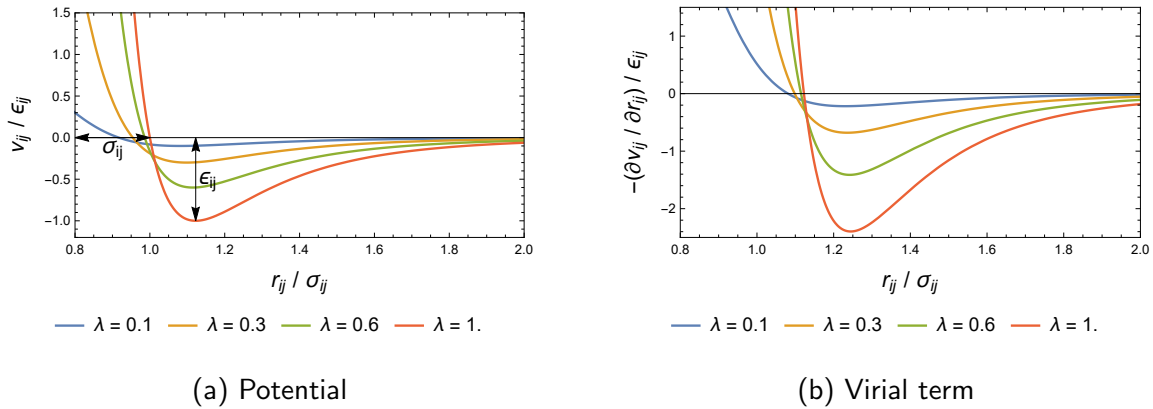


Figure 4.6.: Scaled Lennard-Jones potential function for different values of λ

In case mixtures are simulated, multiple fractionals can be in one box and thus interact. The coupling factor λ used in Equation 4.4.2 is then calculated with Equation 4.4.4.[47]

$$\lambda = \lambda_1 \cdot \lambda_2 \quad (4.4.4)$$

For the scaling of the electrostatic interactions, different mathematical descriptions have been proposed in literature. One method involves the decoupling of the LJ and Coulombic interactions using separate scaling factors.[38] The method used by Ali Poursaeidesfahani and

Wei Shi is implemented in the present algorithm and is stated in Equation 4.4.5.[41, 36]

$$\nu_{\text{el}}(r_{ij}, \lambda) = \lambda^5 \frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}} \quad (4.4.5)$$

4.4.2. Wang-Landau Algorithm

Both the method proposed by Ali Poursaeidesfahani and the one by Wei Shi utilize the Wang-Landau approach to iteratively determine the biasing function which is used to flatten the λ space. Specifically, the difference of the biasing function $\eta_{o \rightarrow n}$ of the old state (subscript o) and new state (subscript n) is calculated. The extended acceptance rule is stated in Equation 4.4.6.

$$\frac{p_n}{p_o} = \min[1, \exp(-\beta \Delta \nu_{o \rightarrow n} + \Delta \eta_{o \rightarrow n})] \quad , \text{ with } \Delta \eta_{o \rightarrow n} = \eta_o - \eta_n \quad (4.4.6)$$

The biasing function $\eta(\lambda)$ is initialized as zero at the start of the simulation and determined iteratively during the equilibration phase. For this purpose, a biasing increment v is used. Further, the λ space is divided into ten equidistant bins, e.g., the first bin ranges from $0 < \lambda < 0.1$. To guarantee a smooth distribution, an interpolation is used between these ten mesh points.[40]

Every time a new λ state is visited, the respective bin and thus $\eta(\lambda)$ is incremented by $\ln(v)$. For example, if a trial move resulting in a λ value of 0.15 is accepted, the biasing value of the second bin is incremented by $\ln(v)$ and the interpolation is updated. Since the biasing is taken into account in the acceptance rule, trial moves resulting in λ bins that have been visited often are less likely to be accepted. In consequence, the λ distribution is flattened out.[40]

After 10^4 configurations have been sampled, the biasing function is evaluated. If each bin contains at least 1 % of all lambda states that have been visited in the respective evaluation period, the biasing increment v is decreased according to Equation 4.4.7.

$$v_{\text{new}} = \sqrt{v_{\text{old}}} \quad (4.4.7)$$

Thereby, convergence against the final values of the biasing function is guaranteed, since Equation 4.4.7 is decreasing strictly monotonously for any $v \in \mathbb{R}^+$ and $\ln(v)$ is initialized to unity.[36, 40, 49]

After the simulation is finished, the biasing must be corrected with Equation 4.4.8.[36, 49]

$$\langle X \rangle_{\text{Boltzmann}} = \frac{\langle X \exp[-\eta(\lambda)] \rangle_{\text{modified}}}{\langle \exp[-\eta(\lambda)] \rangle_{\text{modified}}} \quad (4.4.8)$$

4.4.3. CFC Trial Moves

The GECFC method proposed by Ali Poursaeidesfahani in the working group of Thijs J. H. Vlugt replaces the conventional insertion trial move with the ' λ Change', the 'CFC Exchange' and the 'CFC Swap'. If an insertion trial move is attempted using the CFC insertion method, one of the three trial moves is chosen at random with a fixed probability.

λ Change

This trial move attempts to change the coupling value λ of one species. This changes the interactions of the fractional. The acceptance rule is stated in Equation 4.4.9. In case the biasing is not taken into account, $\Delta\eta_{o \rightarrow n}$ is zero.

$$\frac{p_n}{p_o} = \min [1, \exp(-\beta\Delta\nu_{o \rightarrow n} + \Delta\eta_{o \rightarrow n})] \quad (4.4.9)$$

CFC Exchange

When performing the GECFC exchange trial move, displayed in Figure 4.5a, the fractional is exchanged with a randomly chosen full molecule of the same species in the opposite box. While the coupling value λ and the position of the molecules remain unchanged, the number of full molecules in the boxes changes. The acceptance rule is stated in Equation 4.4.10.

$$\frac{p_n}{p_o} = \min \left[1, \frac{N_{o,i}}{N_{n,i}} \cdot \exp(-\beta\Delta\nu_{o \rightarrow n} + \Delta\eta_{o \rightarrow n}) \right] \quad (4.4.10)$$

Here, the energy change due to the trial move is multiplied by the ratio of the number of full molecules of the exchanged species i in the old and new box. In case the biasing is not taken into account, $\Delta\eta$ is zero. This trial move is likely to be accepted for values of λ close to unity where the fractional interactions are almost equal to a full molecule.[36]

CFC Swap

The GECFC swap trial move is displayed in Figure 4.5b. While the coupling factor λ and the number of full molecules stay constant, the fractional is transferred from one box to the other. The acceptance rule is stated in Equation 4.4.11. The energy change due to the trial move is multiplied by the ratio of the total volume of the old and new box. In case the biasing is not taken into account, $\Delta\eta$ is zero. This trial move is likely to be accepted for small values of λ where the interactions of the fractional are weak.[36]

$$\frac{p_n}{p_o} = \min \left[1, \frac{V_o}{V_n} \cdot \exp(-\beta \Delta\nu_{o \rightarrow n} + \Delta\eta_{o \rightarrow n}) \right] \quad (4.4.11)$$

4.4.4. Calculation of the Chemical Potential

Another important feature of the CFC method is that the chemical potential can be calculated from the λ state distribution, whereby the use of test molecules can be omitted. This is particularly interesting for dense phases and highly size-asymmetric mixtures, as the acceptance rate of inserted test molecules is low due to overlaps. While the full derivation is given in literature [36], the practical implementation is presented in the following section. Without further proof, the chemical potential of a component in a box can be calculated as the ratio of the probability that $\lambda \rightarrow 0$ and $\lambda \rightarrow 1$, as displayed in Equation 4.4.12.

$$\mu^E = -k_B T \cdot \ln \left[\frac{p_{\lambda \rightarrow 1}}{p_{\lambda \rightarrow 0}} \right] \quad (4.4.12)$$

During the simulation, all visited λ states are collected separately for each box in 100 bins. To calculate the probability of one particular state, p_i , the states of the respective bin n_i need to be divided by the total number of visited states and divided by the bin width $\Delta\lambda$ as stated in Equation 4.4.13. Since the λ states are collected in 100 bins the bin width is

0.01.

$$p_\lambda = \frac{n_\lambda}{\Delta\lambda \sum n_\lambda} \quad (4.4.13)$$

If Equation 4.4.13 is reinserted in Equation 4.4.12, the bin width and the total number of visited states cancel out as shown in Equation 4.4.14.

$$\frac{p_{\lambda \rightarrow 1}}{p_{\lambda \rightarrow 0}} = \frac{n_{\lambda \rightarrow 1}}{\sum n_\lambda \cdot \Delta\lambda} \left[\frac{n_{\lambda \rightarrow 0}}{\sum n_\lambda \cdot \Delta\lambda} \right]^{-1} = \frac{n_{\lambda \rightarrow 1}}{n_{\lambda \rightarrow 0}} \quad (4.4.14)$$

Finally, the boundary values of $\lambda \rightarrow \{0, 1\}$ are calculated with a linear extrapolation, as recommended in literature. The reason is that the probability associated with the first bin actually corresponds to a λ value of 0.05, while the probability associated with the last bin actually corresponds to a λ value of 0.95 and the λ distribution can be steep.[36]

In case the biasing is applied, the calculation of the probabilities has to be adjusted using Equation 4.4.8, as stated in Eqs. 4.4.15, 4.4.16 and 4.4.17.

$$p_\lambda = \frac{n_\lambda \cdot \exp[-\eta(\lambda)] \cdot \langle \exp[-\eta(\lambda)] \rangle^{-1}}{\Delta\lambda \sum n_\lambda \cdot \exp[-\eta(\lambda)] \langle \exp[-\eta(\lambda)] \rangle^{-1}} \quad (4.4.15)$$

$$\Leftrightarrow p_\lambda = \frac{n_\lambda \cdot \exp[-\eta(\lambda)]}{\Delta\lambda \sum n_\lambda} \quad (4.4.16)$$

$$\stackrel{4.4.14}{\Rightarrow} \frac{p_{\lambda \rightarrow 1}}{p_{\lambda \rightarrow 0}} = \frac{n_{\lambda \rightarrow 1} \cdot \exp[-\eta(\lambda \rightarrow 1)]}{n_{\lambda \rightarrow 0} \cdot \exp[-\eta(\lambda \rightarrow 0)]} \quad (4.4.17)$$

The total chemical potential is then calculated analogously, based on an ideal and excess contribution as stated in Equation 4.2.3.

4.5. Adjustment of Parameters

For a successful application of the GEMC algorithm it is necessary to use reasonable parameters like the trial move limits, the number of attempted transfers per cycle or the species swap probability. One approach is to conduct pre-simulations and determine these values iteratively. However, this process can be time-consuming and tedious since it is necessary to observe the simulated system for a sufficiently long computation time. Thus, it has been proposed in literature to calibrate these parameters during the equilibration phase.[1]

4.5.1. Trial Move Limits

The adjustments of trial move limits are made automatically to achieve an acceptance ratio of approximately 50%. This is done by examining the acceptance ratio of the last cycle and multiplying the trial move limit by a factor to increase or decrease the limit, which should lead to an improved acceptance ratio in the next cycle.[1] The acceptance ratio is defined by the ratio of the accepted trial moves, N_{Acc} , over the total amount of trial moves, $N_{\text{Acc}} + N_{\text{Rej}}$, as stated in Equation 4.5.1.

$$R_{\text{Acc}} = \frac{N_{\text{Acc}}}{N_{\text{Acc}} + N_{\text{Rej}}} \quad (4.5.1)$$

There are two occasions to change the acceptance ratio: If the ratio was too low in the last cycle, decrease the limits, which leads to smaller molecular adjustments which would lead to more moves being accepted. If the ratio was too high in the last cycle, increase the limits to make more drastic molecular adjustments which would lead to more rejections. The formula for calculating the new trial move limit is stated in Equation 4.5.2.

$$x_{\text{limit,new}} = \begin{cases} 1.05 \cdot x_{\text{limit,last}} & \text{if } R_{\text{Acc,last}} > 50\% \\ 0.95 \cdot x_{\text{limit,last}} & \text{if } R_{\text{Acc,last}} < 50\% \end{cases} \quad (4.5.2)$$

According to Allen and Tildesley, the trial move limits shall be evaluated if at least two moves of the respective trial move type have been attempted, since the acceptance rate needs to be calculated. This, however, can be problematic if the number of trial moves attempted per MC cycle is low. In particular, volume changes are affected by this problem, since usually only 1 to 5 trial moves are executed per MC cycle.

Thus, the proposed algorithm has been slightly adjusted as follows: Instead of evaluating if at least two trial moves of the respective type have been executed every MC cycle, it is checked if at least a certain number of trial moves has been executed. In case the number of moves collected so far is below a certain threshold, the counters of the accepted and rejected moves are incremented and checked again after the next cycle. In a study conducted by Andreas Schwarz [39] it was found that ten is a reasonable number for the threshold. It guarantees a sufficiently fast calibration of the trial move limit as well as a stable adjustment, since the block size is sufficiently large.

The behavior of the algorithm is shown in Figure 4.7a, where argon was simulated as a LJ fluid at different temperatures. The start value for the maximum volume change was chosen as 1 % of the total system volume. Figure 4.7b shows the ratios between the final trial move limit to the total box volume at different LJ-reduced temperatures.

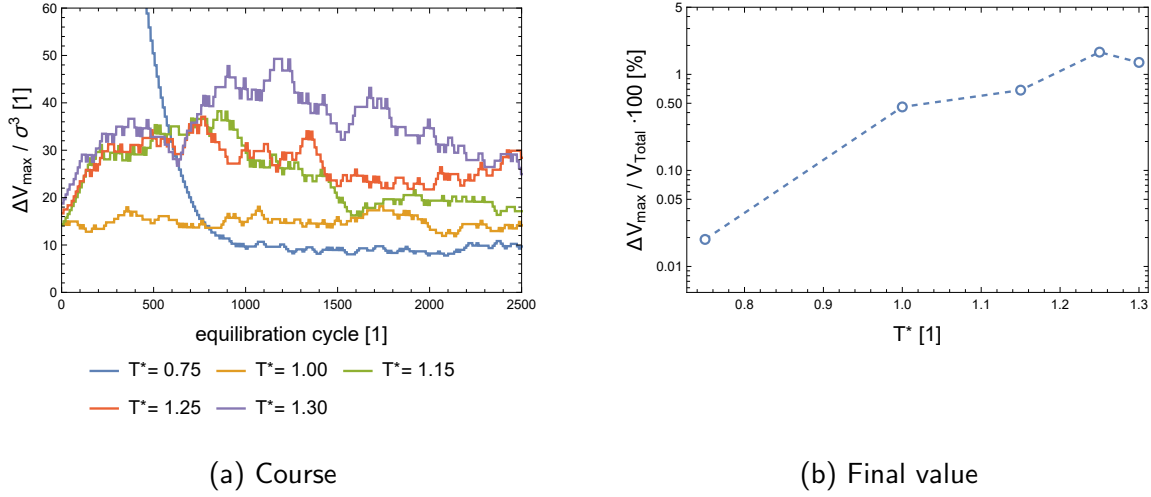


Figure 4.7.: Simulation results of argon simulated as a LJ fluid at different LJ-reduced temperatures using the new algorithm for adjusting trial move limits. (a) Course of the calibrated maximum volume change; (b) Comparison of the final value with the total volume. In the equilibrated state, about 100 molecules are in the vapor and 400 molecules in the liquid phase.

4.5.2. Number of attempted Transfers per Cycle

As discussed in section 3.5, the acceptance rate of the particle transfers is low for dense phases or asymmetric mixtures due to overlaps. Thus, a drastically increased number of attempted transfers is recommended in literature.[28] For example, while only 150 transfers are attempted at $T^* = 1.00$, $2 \cdot 10^3$ transfers are attempted at $T^* = 0.75$.

Therefore, different proposals for choosing an appropriate number of transfers from literature were assessed in a study by Andreas Schwarz.[39] An approach that yielded consistent results in good agreement with literature and the equation of state for LJ fluids by Thol (Thol EoS) [46] is to calibrate the attempted number of transfers in the equilibration phase to yield one successful transfer per MC cycle. This is done by increasing or decreasing the number of

transfers by 5 % if the number of successful particle transfers is too low or high, respectively. At the same time, the number of displacements (translations and rotations) is kept constant and set to the number of molecules in the system. An alternative would be to execute a constant number of trial moves each MC cycle and adjust the likelihood of the respective trial moves. However, the number of displacements would also have to be adjusted in this approach. Since at low temperatures a very high number of attempted transfers is required to yield one successful swap per cycle, the number of displacements would be drastically reduced, whereby the internal equilibration might not be sufficient.

The course of the calibration is shown in Figure 4.8. Despite minor fluctuations, the proposed algorithm converges within the used equilibration phase of 2,000 cycles. It should be noted that the goal is not to yield exactly one successful swap per cycle but to derive an easily applicable parameterization rule with which the phase space can be sampled efficiently. The recommendations found in literature, Figure 4.8b, and the results of the proposed algorithm are in good agreement.

Note that the number and composition of trial moves are kept constant during the production phase. Also, it is necessary to ensure a sufficiently long equilibration phase.

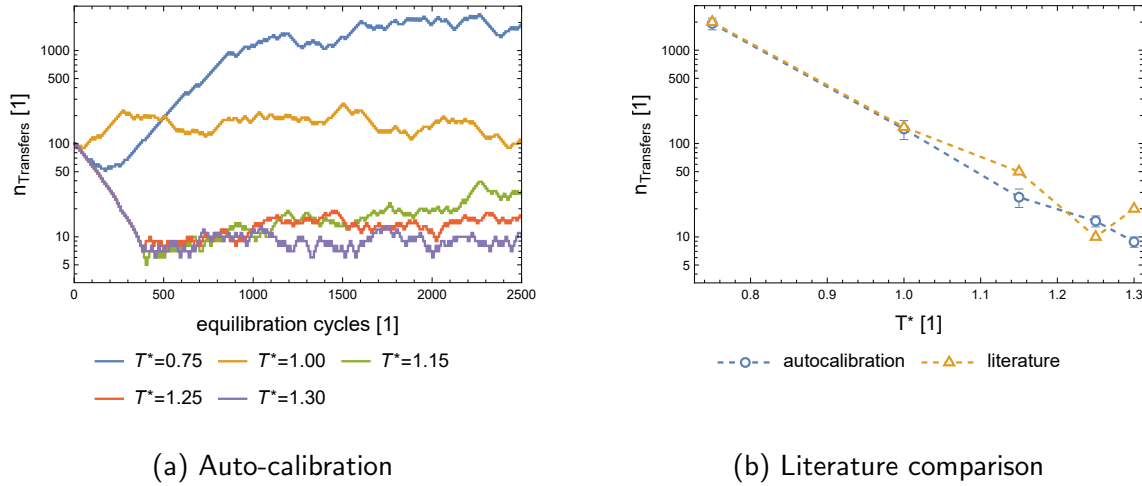


Figure 4.8.: (a) Course of the calibrated number of attempted transfers in the equilibration phase. (b) Comparison to values stated in literature.[28] The data points and error bars indicate the mean and single standard deviation of the last 500 cycles.

4.5.3. Species Choice Probability for Insertions

The *Swap Move Probability* (SMP) describes the probability with which a species is selected for an attempted insertion. By default, the species are selected from the system with equal probability. If a mixture of highly asymmetric components is simulated, the acceptance rate of insertions will vary significantly between the species. This may lead to a large disparity in successful insertions per component, resulting in a possibly insufficient equilibration of the chemical potential. Therefore, in the work of Harismiadis, it is recommended to change the SMP to a fixed value to achieve an equal number of successful insertions per species.[16]

In the current implementation, it is possible to either set the SMP to a fixed value or adjust it during the equilibration phase. If the SMP is adjusted, three different algorithms for auto-calibration are available.

The first algorithm is essentially equivalent to the procedure for the adjustment of the trial move limits (cf. section 4.5.1). If one of the components is accepted too often, then the probability of attempting a transfer of that component is decreased by 5 % of its current value. The formula for the adjustment is stated in Eqs. 4.5.3, 4.5.4 and 4.5.5.

$$\Delta p = p_{i,\text{old}} \cdot 0.05 \quad (4.5.3)$$

$$p_{i,\text{new}} = p_{i,\text{old}} - \Delta p \quad (4.5.4)$$

$$p_{j,\text{new}} = p_{j,\text{old}} + \Delta p, \quad (4.5.5)$$

where component i is accepted too often, while component j is accepted less often.

The second algorithm is based on the currently observed deviation of the acceptance rate from an even distribution and adjusts the SMP accordingly. The formula for the adjustment is stated in Eqs. 4.5.6, 4.5.7 and 4.5.8.

$$\Delta p = \zeta \cdot \underbrace{\left(0.5 - \frac{n_{\text{TR,Acc},i}}{n_{\text{TR,Acc}}}\right)}_{\text{I}} \cdot \underbrace{\min[p_{i,\text{o}}, p_{j,\text{o}}]}_{\text{II}} \quad (4.5.6)$$

$$p_{i,\text{new}} = p_{i,\text{old}} + \Delta p \quad (4.5.7)$$

$$p_{j,\text{new}} = p_{j,\text{old}} - \Delta p, \quad (4.5.8)$$

where $n_{\text{TR},\text{Acc},i}$ is the recorded number of accepted transfers of component i , $n_{\text{TR},\text{Acc}}$ is the total number of accepted transfers and ζ is set to 0.99. Note that, ideally, half of the accepted transfers are accepted for each component, whereby $n_{\text{TR},\text{Acc},i} \cdot n_{\text{TR},\text{Acc}}^{-1}$ approaches 0.5 and, consequently, Δp becomes zero. Term I accounts for the currently observed deviation, and Term II provides the smaller SMP of the two components as a scaling factor for the adjustment. It is necessary to use the smaller of the two SMPs to prevent an adjustment where Δp is greater than one of the probabilities, which would result in a negative parameter. Furthermore, since the SMPs can become considerably small, especially if very asymmetric mixtures are simulated, the change of the smaller SMP would be drastic in case the larger SMP is used as a reference factor.

The third algorithm is based on a separate adjustment of the number of attempted transfers for each component: Instead of adjusting the total number of transfers and the SMP separately, the number of attempted transfers per component is adjusted. Thereby, the adjustment of the probability and the total number of transfers do not interfere with each other. The formula for the adjustment is stated in Equation 4.5.10

$$\zeta = \frac{n_{\text{TR},\text{acc},i}}{n_{\text{TR},\text{acc}}} \quad (4.5.9)$$

$$n_{\text{TR},i,\text{new}} = \begin{cases} \text{floor}[n_{\text{TR},i,\text{old}} \cdot 0.95] & \text{if } \zeta > 0.5 \\ \text{ceil}[n_{\text{TR},i,\text{old}} \cdot 1.05] & \text{if } \zeta < 0.5 \end{cases}, \quad (4.5.10)$$

where $n_{\text{TR},i}$ is the number of attempted transfers per cycle of component i and j .

The method used in a simulation is specified in the custom parameters. A description of how the parameters need to be set is given in the '*01 - Setup.wls*' notebook.

To showcase the difference between the three algorithms for auto-calibrating the SMP, a strongly asymmetric binary mixture of LJ fluids ($\sigma_{22}/\sigma_{11} = 0.5$, $\epsilon_{22}/\epsilon_{11} = 0.5$) has been simulated at $T^* = 1.0$ and $P^* = 0.1$. The results of these simulations are displayed in Figure 4.9, where the course of 'equal' SMPs, a 'fixed' adjustment using the first algorithm, a 'deviation' based adjustment using the second algorithm, and a 'separate' adjustment using the third algorithm are compared. The total number of transfers was adjusted using the method described in section 4.5.2. In Figure 4.9, the course of the simulation parameters is plotted against the CPU time used, since the simulation effort varied with different amounts of attempted transfers per cycle.

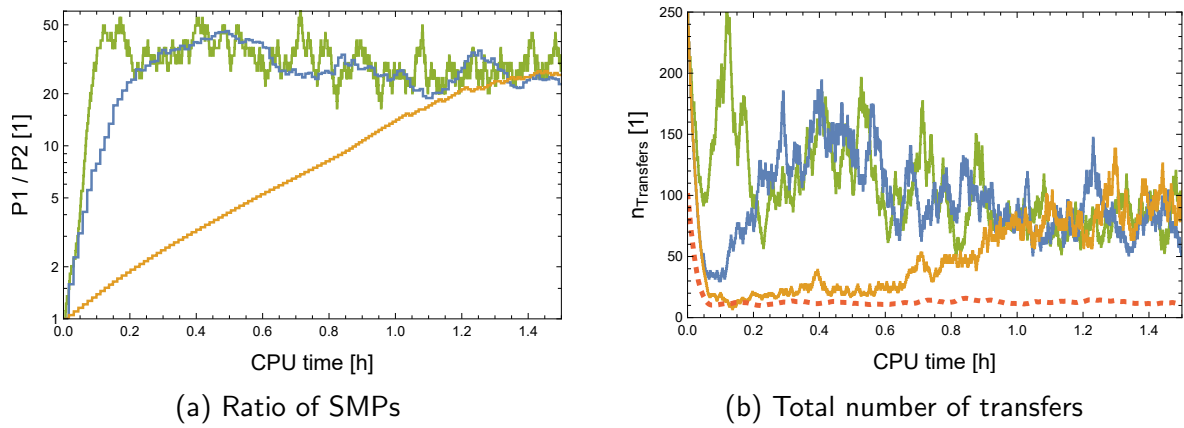


Figure 4.9.: Results of simulations of a strongly asymmetric binary mixture of LJ fluids at $T^* = 1.0$ and $P^* = 0.1$. $\sigma_{22}/\sigma_{11} = 0.5$; $\epsilon_{22}/\epsilon_{11} = 0.5$; The LJ Parameters of the first component are set to unity. (a) Course of the ratio of the adjusted swap move probabilities (SMPs). (b) Total number of attempted transfers. Algorithms: - - - 'equal', — 'fixed', — 'deviation', — 'separate'

4.6. Electrostatic Long-Range Correction

When simulating molecules with partially charged sites, the correct value of the Coulomb potential energy is important for the phase equilibrium. The site-site interactions within the maximum cut-off radius are insufficient, even in systems containing 500 molecules. The potential cannot simply be cut and estimated with a long-range correction like the Lennard-Jones potential. This is because the Coulomb potential decreases slowly with distance (r^{-1}) and the mathematical series $\frac{1}{n}$ is known to diverge. Therefore, the total Coulomb potential is conditionally convergent. The Ewald summation splits the potential and calculates the long-range correction in reciprocal space, while the Wolf summation shifts the potential closer and calculates everything within the cut-off radius.

4.6.1. Ewald Summation

The Ewald summation is a method to calculate the electrostatic interactions at long distances. The problem with the slowly decreasing potential is tackled by splitting the potential into two different terms with the dampening function ϕ (cf. Equation 4.6.1). This results in a quickly decreasing short-range term and a modified long-range term that can be evaluated using Fourier transformation. While the splitting function can be any quickly decreasing function, it is commonly defined as the Gauss error function. κ is a parameter for the width of the Gaussian distribution. For a more detailed description of the implementation, reference is made to literature.[13]

$$U^C = \frac{q_i q_j \cdot \phi(\kappa, r_{ij})}{r_{ij}} + \frac{q_i q_j \cdot (1 - \phi(\kappa, r_{ij}))}{r_{ij}} \quad (4.6.1)$$

Energy Potential Terms

The total Coulomb potential energy, U_{total}^C , in a simulation box of partially charged, non-ionic, poly-atomic molecules can be described in five terms, as displayed in Eqs. 4.6.2 and 4.6.3.

$$U_{\text{total}}^C = U_r + U_k - U_{k0} - U_{\text{bond}} - U_s \quad (4.6.2)$$

$$U_r = \frac{1}{2} \sum_n' \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{b=1}^N \sum_{j=1}^{N_b} \frac{q_{ia}q_{jb}}{|r_{iajb} + nL|} \cdot \text{erfc}(\kappa \cdot |r_{iajb} + nL|) \quad (4.6.3a)$$

$$U_k = \frac{2\pi}{V} \sum_{k \neq 0} \frac{1}{k^2} e^{\frac{-k^2}{4\kappa^2}} \left(\sum_{a=1}^N \sum_{i=1}^{N_a} q_{ia} e^{-ik \cdot r_{ia}} \right)^2 \quad (4.6.3b)$$

$$U_{k0} = \frac{\kappa}{\sqrt{\pi}} \sum_{a=1}^N \sum_{i=1}^{N_a} q_{ia}^2 \quad (4.6.3c)$$

$$U_{\text{bond}} = \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{j>i}^{N_a} \frac{q_{ia}q_{ja}}{r_{iaja}} \cdot \text{erf}(\kappa \cdot r_{iaja}) \quad (4.6.3d)$$

$$U_s = \frac{2\pi}{(1 + 2 \cdot \epsilon_s)V} \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{b>a}^N \sum_{j=1}^{N_b} q_{ia}q_{jb} \cdot r_{iajb}^2 \quad (4.6.3e)$$

The **real space term** (U_r) is the second part of the split potential above. It calculates the site-site interactions between the atoms in the simulation box to all other atoms including images outside the simulation box due to periodic boundary conditions. The width parameter, κ , however, is usually chosen so that only interactions within half the box length need to be considered, and the summation over n is only here for completion.[1]

The **reciprocal space term** (or k -space term) (U_k) is the first part of the split potential above. The function includes a sum over the reciprocal wave vectors $k = n_k \cdot 2\pi/L$, where n_k is a vector consisting of 3 integer values (e.g.: $\{1,1,1\}$) representing a neighboring periodic image of the unit cell. The summation includes all variations of n_k with a maximum magnitude of the specified 'range'. More information on how the function was derived using the potential sum method and Fourier transformation can be found in literature.[1]

The **reciprocal space self-interaction correction** (U_{k0}) corrects the k -space calculation by removing the interaction of all atoms in the unit cell with themselves. This term is most often already integrated into the implemented k -space function.[1]

The **reciprocal space bond correction** (U_{bond}) calculates the interaction energies between bonded atoms in the unit cell. These interactions are omitted from the total energy and, instead, considered in a different form (cf. section 4.1.1). The reciprocal space term U_k includes these interactions already, which is why this value needs to be subtracted.[6]

The **surface correction** (or dipole correction) (U_s) calculates the interaction of a macro-sphere consisting of multiple unit cells with its surrounding with relative permittivity ϵ_s . This way, the boundary conditions for the energy calculation can be set to be between vacuum condition ($\epsilon_s = 1$) and tin foil condition ($\epsilon_s = \infty$).[18]

In Equation 4.6.3, n represents a vector with 3 integers values (e.g.: $\{1,0,0\}$). The term r_{iajb} describes the distance between site i of molecule a and site j of molecule b . The prime in the summation of the real space term notes that interactions where $a = b$ are omitted if $n = \{0,0,0\}$. For a detailed explanation of the derivation of each term, reference is made to literature.

Virial Terms

The virial term is used to estimate the pressure in the simulation box. The formula to calculate the virial term W for the Ewald summation was derived by applying the volume scaling relation, Equation 4.6.4, to the relation between the virial term and the derivative of the internal energy over a change of the total volume, Equation 4.6.5.

$$r_{iajb} = V^{1/3} \cdot \rho_{ab} + d_{iajb} \quad (4.6.4)$$

$$W = \frac{\partial U}{\partial V} \cdot V \quad (4.6.5)$$

Here, ρ_{ab} is the relative position of a molecule center to the simulation box length ($\rho_{ab} = r_{ab}/L$), and d_{ia} is the distance from a molecule center to the individual site ($d_{ia} = r_{ia} - r_a$) and $d_{iajb} = d_{jb} - d_{ia}$. [13]

The total virial term from the Ewald summation can be calculated with four terms summarized in Equation 4.6.6. The summation over n was omitted since the real space term is only

calculated within the unit cell ($n = \{0, 0, 0\}$).

$$\begin{aligned}
 3 \cdot W_{\text{total}}^{\text{C}} = & \frac{1}{2} \sum_{\substack{a,b=1 \\ i,j=1}}^N \frac{q_{ia}q_{jb}}{r_{iajb}^2} \cdot \left(\text{erfc}(\kappa \cdot r_{iajb}) + r_{iajb} \cdot \frac{2\kappa}{\sqrt{\pi}} \cdot e^{-\kappa^2 r_{iajb}^2} \right) \cdot r_{iajb} \frac{\langle r_{ab}, r_{iajb} \rangle}{|r_{iajb}|^2} \\
 & + \frac{2\pi}{V} \sum_{k \neq 0} \frac{1}{k^2} \left(1 - \frac{k^2}{2\kappa^2} \right) \cdot e^{\frac{-k^2}{4\kappa^2}} \cdot \left(\sum_{a=1}^N \sum_{i=1}^{N_a} q_{ia} e^{-ik \cdot r_{ia}} \right)^2 \\
 & + \frac{4\pi}{V} \sum_{k \neq 0} \frac{1}{k^2} e^{\frac{-k^2}{4\kappa^2}} \cdot \text{Im} \left[\sum_{\substack{a=1 \\ i=1}}^N q_{ia} e^{-ik \cdot r_{ia}} \cdot \sum_{\substack{a=1 \\ i=1}}^N q_{ia} \langle k, d_{ia} \rangle e^{-ik \cdot r_{ia}} \right] \\
 & - 3 \cdot U_s + \frac{4\pi}{(1 + 2 \cdot \epsilon_s)V} \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{b>a}^N \sum_{j=1}^{N_b} q_{ia}q_{jb} \cdot \langle r_{iajb}, r_{ab} \rangle
 \end{aligned} \tag{4.6.6}$$

4.6.2. Wolf Summation

The electrostatic long-range correction can also be calculated with a method suggested by Wolf. Here, the problem of the slowly decreasing electrostatic potential is tackled with a shifted potential function. The correction terms in addition to the shifted potential are derived with charge neutrality and the Madelung potential in mind.[52]

The total Coulomb Energy ($U_{\text{total}}^{\text{C}}$) in a simulation box of partially charged, non-ionic, polyatomic molecules can be described in four terms as displayed in Equation 4.6.7:

$$U_{\text{total}}^{\text{C}} = U_r - U_{k0} - U_{\text{bond}} - U_s, \tag{4.6.7}$$

where

$$U_r = \frac{1}{2} \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{b=1}^N \sum_{j=1}^{N_b} q_{ia}q_{jb} \cdot \left(\frac{\text{erfc}(\kappa \cdot r_{iajb})}{r_{iajb}} - \frac{\text{erfc}(\kappa \cdot R_c)}{R_c} \right) \tag{4.6.8a}$$

$$U_{k0} = \left(\frac{\text{erfc}(\kappa \cdot R_c)}{2R_c} + \frac{\kappa}{\sqrt{\pi}} \right) \cdot \sum_{a=1}^N \sum_{i=1}^{N_a} q_{ia}^2 \tag{4.6.8b}$$

$$U_{\text{bond}} = \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{j>i}^{N_a} q_{ia}q_{ja} \cdot \left(\frac{\text{erf}(\kappa \cdot r_{iaja})}{r_{iaja}} + \frac{\text{erfc}(\kappa \cdot R_c)}{R_c} \right) \tag{4.6.8c}$$

$$U_s = \frac{2\pi}{(1 + 2 \cdot \epsilon_s)V} \sum_{a=1}^N \sum_{i=1}^{N_a} \sum_{b>a}^N \sum_{j=1}^{N_b} q_{ia} q_{jb} \cdot r_{iajb}^2 \quad (4.6.8d)$$

The notations in Equation 4.6.8 are the same as for the Ewald summation in section 4.6.1. The virial term was not explicitly derived for the Wolf summation. Therefore, simulations use the Ewald method to calculate the virial term. The Wolf summation is only recommended for well-studied systems, as simulations showed considerable instability if the cut-off radius is not large enough.

5. Implementation

The following chapter presents the structure of the code, details about each notebook, and the implementation of the methods described in chapter 4.

5.1. Units used during the Simulation

The following units are used throughout the simulation: Temperatures in [K], lengths in [\AA], angles in [rad], charges in [$1/e$] (where e is the elementary charge) and energies and virials in [kcal/mol].

Note that in almost all calculations, the Boltzmann constant k_B is replaced by the universal gas constant R because of the molar units used in the energy and virial terms. The conversion between the two constants is the following: $R = N_A \cdot k_B$ where N_A is the Avogadro constant.

5.2. 01 - Setup.wls

Before running the simulation, the environment, its molecules and how those molecules interact with each other have to be defined. All the setup that is needed takes place in the chapter *Simulation Definitions* in the notebook, which is divided into parts that fulfill a specific purpose. It is advised to go through all those sections in the given order.

GEMC - SETUP

In this workbook the simulation environment is defined.

Do not use the "Run all Code" button and go through all sections carefully since some are test-case specific!

Also keep in mind, that this notebook (unlike the Execution or Analysis) does NOT overwrite existing simulation definitions. Therefore make sure, that the configuration is properly exported by defining a valid path and/or deleting existing simulation definitions in advance if necessary.

Init »

Simulation Definitions »

Export »

Credits »

Figure 5.1.: Structure of the *Setup* notebook

5.2.1. Molecules and their Parameters

In the first sections all the molecule information is set up by defining the following variables:

<code>system</code>	list of molecule names in the simulation
<code>moleculeData</code>	association with the atom positions of each molecule
<code>moleculeBonds</code>	list of all bonded atom pairs in each molecule
<code>moleculeNonBondedPairs</code>	association with the atom pairs separated by three or more bonds and the scaling factors of each molecule
<code>ffLabelsNonBonded</code>	list of associations with non-bonded labels per atom
<code>ffLabelsBondStretching</code>	list of associations with bond stretching labels and parameters per bond
<code>ffLabelsAngleBending</code>	list of associations with angle bending labels and parameters per angle

<code>ffLabelsTorsion</code>	list of associations with torsion labels and parameters per axis
<code>ffParamCombiningRule</code>	name of combining rule used for unlike pair interactions

Both `moleculeData` and `moleculeBonds` can be imported automatically from a '*.mol'-file by choosing the 'automatic' procedure. The name of a component in `system` needs to be identical to the filename of the '*.mol'-file for the automatic import to work.

The variable `moleculeNonBondedPairs` is created automatically from the information stated in `moleculeBonds`. It contains an association per molecule in the system with the atom pairs separated by three or more bonds as keys. The atom pairs are associated to the appropriate scaling factors for the LJ-12, LJ-6 and electrostatic interactions in the format: `<| {atomIndex1,atomIndex2} → {fij,LJ12,fij,LJ6,fij,C}, ... |>`.

The formats of all variables are described in the notebook with an example molecule commented out. It is important to note that in the definition of the variable `ffLabelsNonBonded` the atoms have to be set up with the order of occurrence in `moleculeData`. To identify the atom order and bond positions and to select the right labels for the variables, the molecules can be visualized with the order labeled on top of each atom and the bond positions. Figure 5.2 shows such a visualization with methanol as the molecule of choice.

The parameters associated with the specified labels, set in the ff-variables, are defined in the *MolecularSampling* package. To verify the correct definition of the ff-variables, a consistency check can be executed. The necessary amount of labels is calculated based on the definition in `moleculeData` and `moleculeBonds`. A grid as shown in Figure 5.3 displays whether the variables are set properly or not.

5.2.2. System Properties

The simulation environment is set up in the next subsections, by defining the following variables:

<code>ensembleType</code>	specifies the ensemble used in the simulation (1 → NVT, 2 → NPT)
<code>ntot</code>	total number of molecules in the simulation

<code>x1V</code> , <code>x1L</code>	initial molar fraction of the first component per box
<code>T</code> , <code>Psys</code>	temperature and pressure in the simulation
<code>vV</code> , <code>vL</code>	initial molar volume per box (Vapor or Liquid)

The ensemble type defines if the simulation should be carried out with $V_{\text{total}} = \text{const.}$ or with $P = \text{const.}$ To use the ensemble with constant pressure, a mixture of molecules has to be used because for pure substances there are too few degrees of freedom to fix so many quantities. The ensemble with constant total volume can be used for pure substances and mixtures. Note that the pressure (`Psys`) is only used if the simulation is carried out with $P = \text{const.}$

The total number of molecules drastically influences the amount of time the simulation will need to execute because the computation time increases exponentially with the number of atoms in the system. There are two reasons for this time increase: The first one is the increased amount of interactions, which have to be calculated in each move. The second one is the requirement of more moves per cycle to create new states.

The initial mole fractions of components in each box should be chosen in such a way that the total mole fraction is between the dew line and the bubble line in the phase diagram. Otherwise, the system pressure will equilibrate to a higher or lower value than expected, depending on the total mole fraction.

For this purpose, it is recommended to use experimental data or a thermodynamic model to estimate the initial mole fractions. Figure 5.4 illustrates the estimation of mole fractions for the two-component system argon + krypton using a $\{P, x, y\}$ diagram for a simulation at constant pressure.

The initial molar volumes of the boxes should be estimated with an *Equation of State*, for example. If the initial guesses of those values are too low or too high, one can end up with two identical phases.

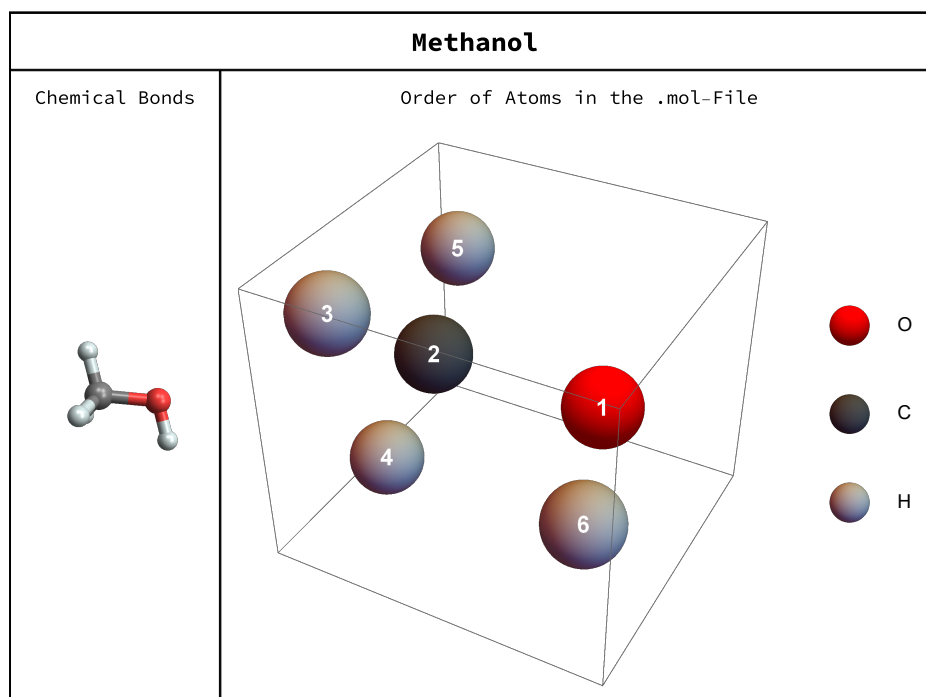


Figure 5.2.: Visualization of a molecule to identify the right order of atoms in the list

Calculation of box specific conditions

After all the quantities are set up, some calculations are carried out to create the right box sizes and the right amount of molecules in each box, by defining the following variables:

<code>nvec</code>	list of molecules in each box
<code>Vvec</code>	list of volume of each box in [\AA^3]
<code>Lvec</code>	list of box side length of each box in [\AA]
<code>nCompVec</code>	list of number of molecules per species in each box

The total number of molecules in each box (`nvec`) can be specified explicitly or calculated using one of the following procedures:

'vapourDoubleVolumeLiquid'

`nvec` is set so that with the specified molar volumes, `vV` and `vL`, the initial box side

OPLS Setup Checks	
Molecule Bonds	☺
Non-bonded Lables	☺
Bond-Streching Lables	☺
Angle Bending Labels	☺
Torsion Lables	☺

Figure 5.3.: Output from consistency check of labels for energy calculation

length of the first simulation box is twice the box side length of the second simulation box.

'equalMolecule'

`nvec` is set with equal amounts of molecules in each simulation box.

'equalVolume'

`nvec` is set so that with the specified molar volumes, `vV` and `vL`, the initial volume of each simulation box is equal.

'explicit'

`nvec` is set with the explicitly stated values for `nV` and `nL` for the first and second simulation box.

Alternatively, if an estimation for the molar volumes from an equation of state (EoS) is available, the function `NTvapourDoubleVolumeLiquid2` can be used to set `nvec`, `vV` and `vL` with the following constraints:

- the vapor box has twice the volume of the liquid box
- an equal amount of molecules is in each box at the start
- at equilibrium, 100 molecules are in the vapor phase

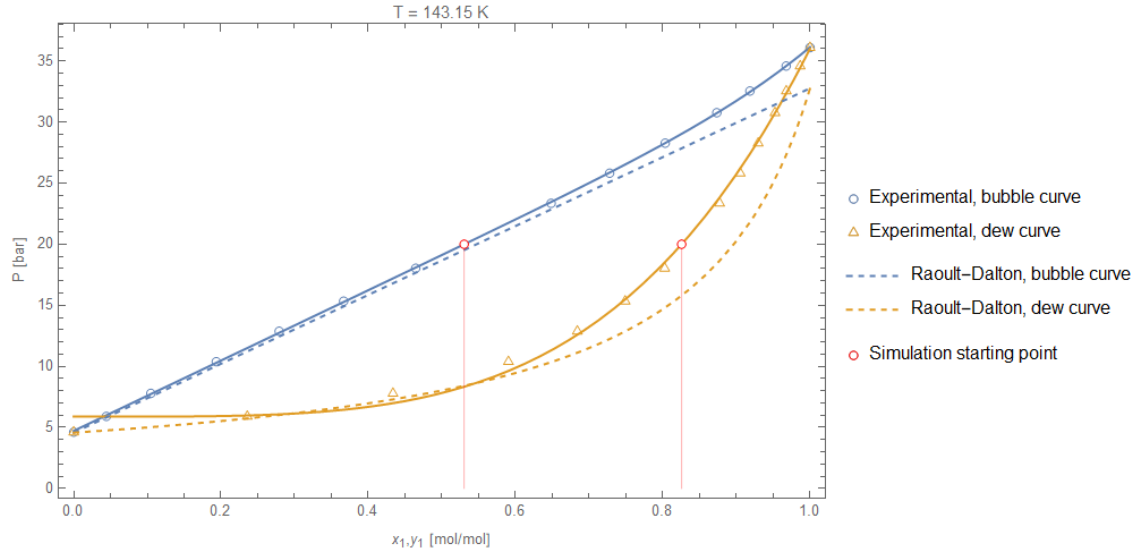


Figure 5.4.: Proposed mole fractions of $x_1 = 0.5303 \text{ mol mol}^{-1}$, $y_1 = 0.8256 \text{ mol mol}^{-1}$ for a simulation of the two-component system argon (1) + krypton (2) at 143.15 K and 20 bar.

The calculation process of the '*vapourDoubleVolumeLiquid*' procedure is as follows. First, the number of molecules in each box is specified with the specific volumes and the prerequisite that the box length of the vapor box is two times larger than the liquid box. Equation 5.2.1 is the basis, which can be rewritten to Eqs. 5.2.2 and 5.2.3, which we can then use for the calculation of the number of molecules in each box. The conditions ($v_1 > v_2$ and $v_1 < v_2$) are used to identify the vapor box. The results for N_1 and N_2 are then rounded to obtain integer values.

$$v_1 = \frac{V_1}{N_1} = \frac{L_1^3}{N_1} \quad v_2 = \frac{V_2}{N_2} = \frac{L_2^3}{N_2} \quad (5.2.1)$$

if $v_1 > v_2$:

$$L_1 = 2L_2 \quad N_2 = \text{round} \left[\frac{N_{\text{total}} \cdot v_1}{2^3 v_2 + v_1} \right] \quad N_1 = N_{\text{total}} - N_2 \quad (5.2.2)$$

if $v_1 < v_2$:

$$L_2 = 2L_1 \quad N_1 = \text{round} \left[\frac{N_{\text{total}} \cdot v_2}{2^3 v_1 + v_2} \right] \quad N_2 = N_{\text{total}} - N_1 \quad (5.2.3)$$

The box sizes are then simply calculated with the specific volumes and the number of molecules as shown in Eqs. 5.2.4 and 5.2.5.

$$V_1 = v_1 \cdot N_1 \qquad L_1 = \sqrt[3]{V_1} \qquad (5.2.4)$$

$$V_2 = v_2 \cdot N_2 \qquad L_2 = \sqrt[3]{V_2} \qquad (5.2.5)$$

The initial number of molecules of a specific component in each box is then calculated according to Equation 5.2.6, where $x_{1,i}^{\text{init}}$ and $x_{2,i}^{\text{init}}$ represent the initial mole fractions of every molecule type i in boxes 1 and 2.

$$N_{1,i} = x_{1,i}^{\text{init}} N_1 \qquad N_{2,i} = x_{2,i}^{\text{init}} N_2 \qquad (5.2.6)$$

5.2.3. Number of Cycles and Trial Moves

In this section, the number of cycles for each phase and the number of trial moves are defined with the following variables:

<code>nWarmUpCycles</code>	number of warm-up cycles
<code>nEquiCycles</code>	number of equilibration cycles
<code>nProdCycles</code>	number of production cycles
<code>nMovesPerCycles</code>	list of number of trial moves to be executed per cycle

The simulation, as proposed by Allen and Tildesley [1], consists of three different phases, namely the warm-up phase, the equilibration phase and the production phase. In the warm-up phase, only translations and rotations are executed and the cycle results are not saved. During the warm-up and equilibration phase, trial move limits are adjusted to achieve acceptance ratios of about 5%. More information about the phases can be found in section 5.3.

The order of values in `nMovesPerCycles` is important because every index corresponds to a certain trial move type. An example for this is given in Listing 5.1, where the name of the trial move type is stated in the same row.

Listing 5.1: Definition of the number of trial moves used in each cycle

```

1 (* number of trial moves per cycle *)
2 nMovesPerCycles = {
3     500, (* translations *)
4     500, (* rotations *)
5     1,   (* volume changes *)
6     250, (* insertions *)
7     125  (* ghost insertions -> widom insertion method *)
8 };

```

As a rule of thumb for the number of trial moves in each cycle, use as many trial moves as there are molecules in the system, since this ensures relatively new molecular states after each cycle.

5.2.4. Trial Move Limits

Some trial moves need a limit at which they can change the molecules. The limits are then used by the trial moves which change the molecule with a random number between 0 and the specified limit. In the translation trial move, for example, a molecule will be translated into a random direction with a distance between 0 and the specified limit.

<code>transMaxVec</code>	maximum translation distance per species in [\AA]
<code>rotaMaxVec</code>	maximum rotation angle per species in [rad]
<code>volChangeMaxVec</code>	maximum volume change per box in [\AA^3]

The trial move limits will be adjusted during the warm-up and equilibration phase to reach an acceptance ratio of approximately 50%.

The cut-off radius and the overlap radius are values that are used during the calculation of the interaction energies. The background of the cut-off and overlap radius is described in sections 4.1.4 and 4.1.5.

<code>rCutVec</code>	cut-off radius per box in [\AA]
<code>rOvlVec</code>	overlap radius per box in [\AA]

If the box side length changes during the simulation, e.g., due to a volume change trial move, then the cut-off radius is scaled to stay constant in relation to the box side length (see section 5.3.2). By default, `rCut` is set to be half of the smallest box side length for both boxes. The cut-off radius can also be set as half of each box side length, respectively, to avoid problems in case the box phases switch. The overlap radius can be defined using one of the following two options:

Manual If this option is chosen, any value can be set for the overlap radius. However, it is recommended to follow recommendations in literature to obtain plausible simulation results. It is important to choose the overlap radius appropriately according to the simulated LJ parameters, σ_i and ϵ_i .

Automatic If this option is chosen, the overlap radius is automatically calculated. The calculation is based on the recommendation of Panagiotopoulos [28] that the overlap radius should be chosen according to Equation 5.2.7.

$$\exp \left[\frac{\nu(r_{\text{Ov}})}{k_{\text{B}}T} \right] \stackrel{!}{=} -10^{-15} \quad (5.2.7)$$

Figure 5.5a shows the LJ potential energy at the recommended radius for overlapping. Due to the high interaction energy between the closest sites, trial moves are rarely accepted. The result of a short test, where the shortest distance to another site was documented for every accepted insertion, is displayed in Figure 5.5b. The test shows that insertions are never accepted if the distance to the closest particle is below the calculated value from Equation 5.2.7.

5.2.5. Initial Molecule Positions and Orientations

The last variables that have to be set are the ones that hold all the information about the positions, orientations and types of the molecules and their atoms. All those variables have the suffix 'Vec' at the end, which indicates that the first element of the nested list is for the first box and the second element for the second box. The elements of those variables are accessed with the following scheme: `variableName[[boxIndex,moleculeIndex,atomIndex]]`.

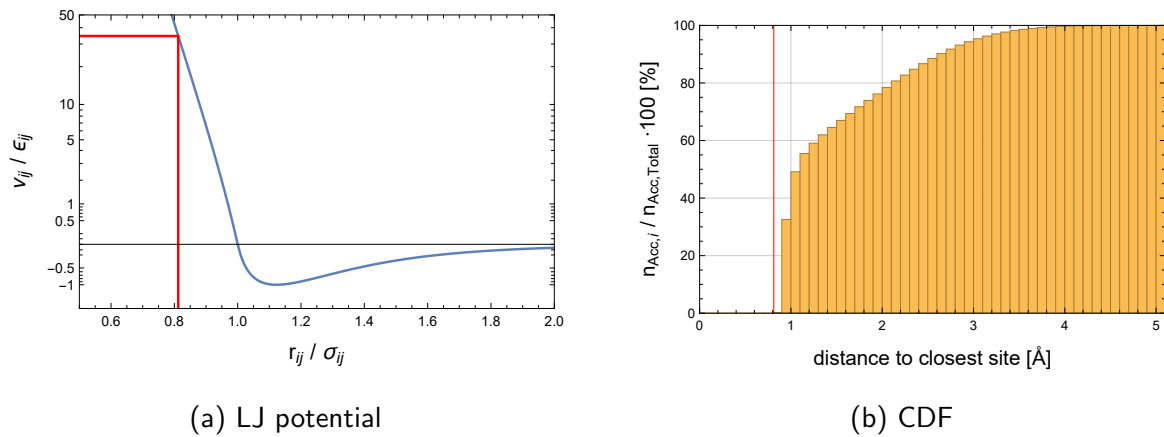


Figure 5.5.: (a) LJ potential and (b) CDF of accepted insertions over distance to closest interacting site. The red vertical indicator shows the recommended overlap radius according to Panagiotopoulos [28] at the LJ-reduced temperature $T^* = 1$.

<code>moleculeCoordVec</code>	coordinates of every molecule in each box
<code>moleculeNamesVec</code>	name of every molecule in each box
<code>moleculeAtomsCoordVec</code>	coordinates of every atom of every molecule in each box
<code>moleculeAtomsNamesVec</code>	name of every atom of every molecule in each box
<code>moleculeAtomsNonBondedLabelsVec</code>	OPLS label for every atom of every molecule in each box
<code>moleculeAtomsIndicatorsVec</code>	integer value for every atom of every molecule in each box, which defines the index in the <code>atomTypes</code> variable (this is used in compiled energy calculation functions, because compile does not work with string values)

Grid System

To distribute all molecules in the two boxes, while preventing overlaps as well as possible, a grid system is used. The used grid system is a face-centered-cubic (FCC) crystal structure,

which consists of stacked unit cells, each containing four possible positions for molecules, as illustrated in Figure 5.6. The coordinates for one unit cell with a cell length of 1 are listed in Equation 5.2.8.

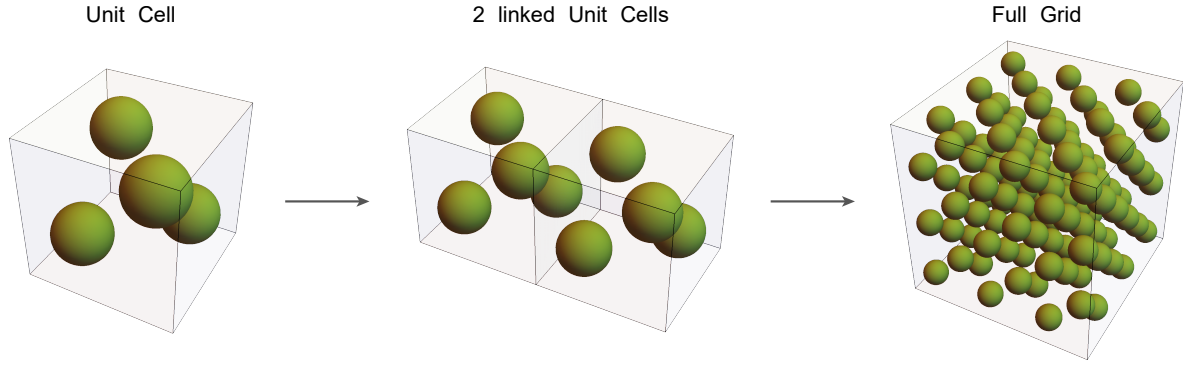


Figure 5.6.: Creation of a FCC crystal structure with unit cells linked to each other

$$\begin{aligned}
 p_1 &= \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \end{bmatrix} & p_2 &= \begin{bmatrix} 0.25 \\ 0.75 \\ 0.75 \end{bmatrix} & p_3 &= \begin{bmatrix} 0.75 \\ 0.75 \\ 0.25 \end{bmatrix} & p_4 &= \begin{bmatrix} 0.75 \\ 0.25 \\ 0.75 \end{bmatrix}
 \end{aligned} \tag{5.2.8}$$

Equation 5.2.9 is used to calculate the number of unit cells needed per dimension for one box. After that, the size of one unit cell is calculated with Equation 5.2.10. The now available positions in all the unit cells are then randomly filled to match the number of molecules that have to be placed in the box. Every molecule is inserted into the grid with a random orientation.

$$cellsPerDim = \text{ceil} \left[\sqrt[3]{\frac{N_{\text{Box}}}{4}} \right] \tag{5.2.9}$$

$$cellLength = \frac{L_{\text{Box}}}{cellsPerDim} \tag{5.2.10}$$

5.2.6. Overview of Initial Conditions

After everything is set up accordingly, an overview can be given to check if all settings are set the way they should be. For demonstration purposes only, the following figures show

the setup of a mixture of ethane and argon with constant total volume. In Figure 5.7, all relevant values of the initial conditions like the number of molecules in each box or the resulting box sizes after the automatic calculations are given. In Figure 5.8, an overview of the simulation-specific settings is displayed, including the number of cycles and the limits for the trial moves. In Figure 5.9, all the OPLS force field settings are displayed. In this overview the intermolecular non-bonded settings and the intramolecular settings for bond stretching, angle bending and torsion are presented.

ENVIRONMENT			
Ensemble Type	Gibbs Ensemble with constant total Volume		
Components	Ethane, Argon		
T [K]	200.		
P [bar]	12.		
	BOX 1	BOX 2	TOTAL
L [Å]	71.8027	35.9492	
V [Å ³]	370189.	46458.9	416647.
ρ [Molecules/Å ³]	4.64628×10^{-4}	9.21245×10^{-3}	1.44007×10^{-3}
v [dm ³ /mol]	1.29612	0.0653696	0.418185
Number of Molecules	172	428	600
Ethane	86	214	300
Argon	86	214	300
Mole Fractions			
Ethane	0.5	0.5	0.5
Argon	0.5	0.5	0.5

Figure 5.7.: Overview of the initial conditions for the boxes

SIMULATION DETAILS		
warm-up cycles	500	
equilibration cycles	1000	
production cycles	2000	
translations per cycle	150	
rotations per cycle	150	
volume changes per cycle	5	
insertions per cycle	300	
ghost insertions per cycle	150	
total number of moves per cycles	755	
	BOX 1	BOX 2
cutoff distance [\AA]	17.	17.
overlap distance [\AA]	1.	1.
max translation distance [\AA]	2.	2.
max rotation angle [rad]	0.436332	0.436332
max volume change [\AA^3]	4166.47	4166.47

Figure 5.8.: Overview of the simulation settings

OPLS-AA DEFINITIONS				
	non-bonded	bond stretching	angle bending	torsion
Ethane	C,RCH3 C,RCH3 H,RH,alkanes H,RH,alkanes H,RH,alkanes H,RH,alkanes H,RH,alkanes H,RH,alkanes	{1, 2} \rightarrow {CT-CT, 0.3}	{3, 1, 2} \rightarrow {HC-CT-CT, 20.} {4, 1, 2} \rightarrow {HC-CT-CT, 20.} {5, 1, 2} \rightarrow {HC-CT-CT, 20.} {6, 2, 1} \rightarrow {HC-CT-CT, 20.} {7, 2, 1} \rightarrow {HC-CT-CT, 20.} {8, 2, 1} \rightarrow {HC-CT-CT, 20.}	{3, 1, 2, 6} \rightarrow {alkane H-C-C-H, 60.}
Argon	Ar,Tan			

Figure 5.9.: Overview of the OPLS force field settings

5.2.7. Custom Parameters

Additional implemented features can be enabled using the custom parameters. Custom parameters are set as rules in the association variable `custParam`. Each rule is set up with a key name and an associated value in the format `"keyname" → value`. The variable `custParam` is exported into the `'parameter.m'` file for other notebooks to read. The values of the custom parameters can be accessed via `Part` or the helper function `checkGetAssocParameter`.

More details on the available custom parameters and their effect can be found in chapter 6 and the `'01 - Setup.wls'` notebook.

5.3. 02a - Execution.wls

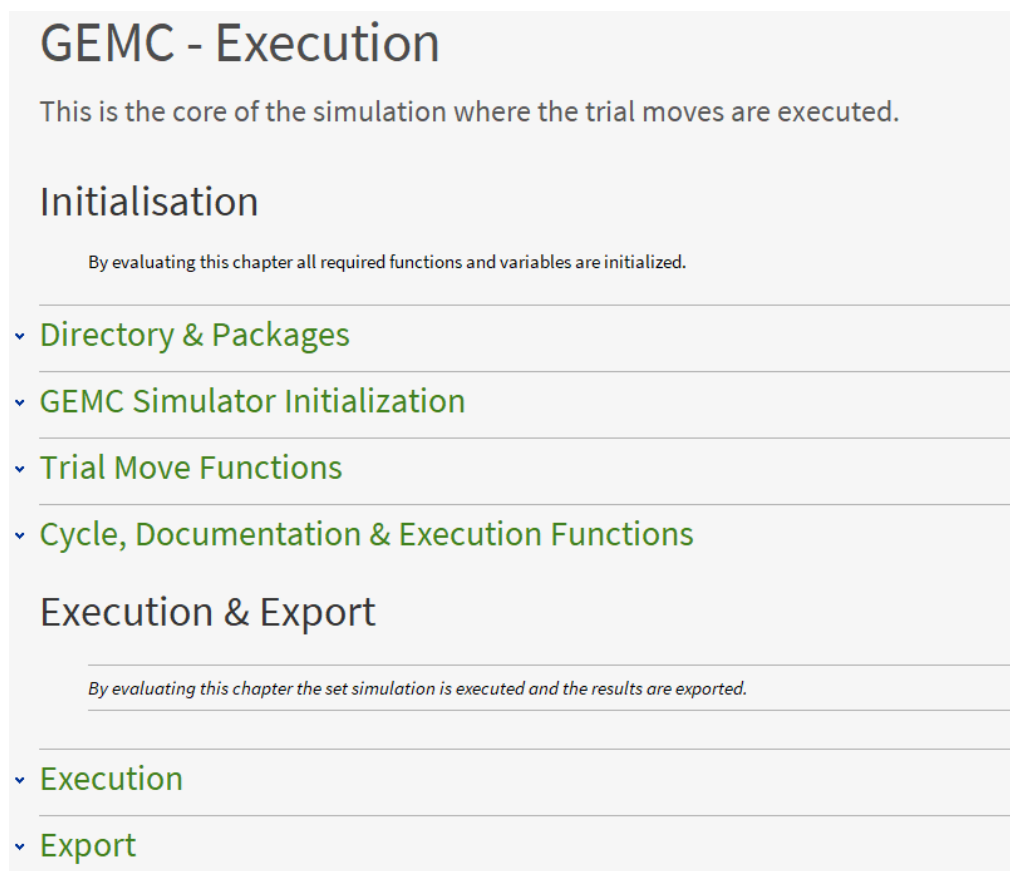
The execution notebook is built in such a way that once the setup is made, the whole notebook can be executed at once, ensuring that everything runs in the right order. First, the settings are imported, followed by the initialization of variables and functions. Then the execution of the simulation takes place. The execution finishes by exporting the results. This section provides information about the implementation of the simulation.

5.3.1. Initialization Phase

In the initialization phase, variables and functions are set up and some calculations take place.

Force Field Parameters

In the setup notebook, the OPLS labels for every molecule were defined. Now, the parameters for each of those molecules are extracted from the *MolecularSampling* package, and the parameters for all possible intermolecular interaction pairs have to be calculated using the specified combining rule. The combining rules are discussed in section 4.1.2.

Figure 5.10.: Structure of the *Execution* notebook

First, the non-bonded interaction parameters are imported and prepared. The parameters for every atom type are stored with the following units:

- σ ... distance at which the potential is zero [\AA]
- ϵ ... depth of the lowest point of the potential [kcal/mol]
- q ... electric charge in [$1/e$] where e is the elementary charge

To store the non-bonded interaction pair parameters, two variables are set up:

`ffInteractionTable` association with a string list of the two OPLS labels of the interacting atoms as the key and the interaction parameters as the value

`ffParamsNonBonded` list consisting only of the interaction parameters so that the variable can be used in functions that should be compilable (the index of the parameters in the list is also the identifier of the atom type which is specified in `moleculeAtomsIndicatorsVec`)

An example of how these variables are structured for a binary mixture of LJ fluids can be found in Figure 5.11. For the electrostatic interactions, the charges just have to be multiplied

```
ffInteractionTable = <|
  {"Ar","Ar"} → {0.0, 3.39520, 0.232086},
  {"Ar","Kr"} → {0.0, 3.50938, 0.273829},
  {"Kr","Kr"} → {0.0, 3.62740, 0.323080}
|>

ffParamsNonBonded = {
  {
    {0.0, 3.39520, 0.232086},
    {0.0, 3.50938, 0.273829}
  },
  {
    {0.0, 3.50938, 0.273829},
    {0.0, 3.62740, 0.323080}
  }
}
```

Figure 5.11.: Example of the variables `ffInteractionTable` and `ffParamsNonBonded` with parameters for the interactions between argon and krypton.

for Coulomb's law. To save some computational time during the simulation, everything of Coulomb's law except the distance will be calculated together in this step, which is subsumed in $q_{ij,\text{total}}$ according to Equation 5.3.1.

$$E_{\text{Coulomb}} = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}} = q_{ij,\text{total}} \frac{1}{r_{ij}} \quad (5.3.1)$$

In the end, the parameter list of one interaction pair consists of three values. The first value is for the electrostatic interaction, the second one is σ for the LJ interactions and the third one is ϵ , also for the LJ interactions: $\{q_{ij,\text{total}}, \sigma_{ij}, \epsilon_{ij}\}$

After all the non-bonded interaction parameters are set up, all the intramolecular energy parameters have to be extracted from the *MolecularSampling* package. The parameters for bond stretching, angle bending and torsion do not need any calculation before a variable can be created, because the parameters are specific for a certain atom constellation. The following variables are created to store all the parameters: `ffParamsBondStretching`, `ffParamsAngleBending` and `ffParamsTorsion`.

Variable Initialization

In this section, all variables used to monitor progress and store results of every cycle are initialized, as well as certain Booleans that are necessary for function declarations. With predefined variables, the performance of the program can be improved because the lists only have to be adjusted and not recreated every time a new value is added.

<code>evalTime</code>	association with total execution times of specific code parts for later analysis of performance bottlenecks
<code>timingCPUStart</code>	initial CPU time used in the current session [s]
<code>nDocuCycles</code>	number of cycles that save the results (equilibration and production phase)

The following variables store all the configurational values of the system for each cycle, indicated with the common suffix '*List*':

<code>nvecList</code>	total number of molecules in each box
<code>VvecList</code>	volume of each box [\AA^3]
<code>nCompVecList</code>	number of molecules per species in each box
<code>interEnergyList</code>	intermolecular potential energy in each box [kcal/mol]
<code>interVirialList</code>	virial term in each box [kcal/mol]
<code>intraEnergyList</code>	intramolecular potential energy in each box [kcal/mol]
<code>tailCorrEnergyList</code>	tail correction for the potential energy in each box [kcal/mol]
<code>tailCorrPressureList</code>	tail correction for the virial term in each box [kcal/mol]
<code>muPartialList</code>	sum of the energy increases due to test particle insertion for the chemical potential (see section 5.3.2)
<code>muCounterList</code>	number of test particles used for the calculation of <code>muPartialList</code> for later division to calculate an average
<code>timingCPUList</code>	CPU time spent per MC cycle [s]

The counter variables document the amount of successful or rejected executions of all trial moves. The variables used to document the amount of rejected trial moves due to overlap or if no particle is present are omitted here.

<code>counterTransAcceptDocu</code>	accepted translations in each box
<code>counterTransRejectDocu</code>	rejected translations in each box
<code>counterRotaAcceptDocu</code>	accepted rotations in each box
<code>counterRotaRejectDocu</code>	rejected rotations in each box
<code>counterVolChangeAcceptDocu</code>	accepted volume changes (boxIndex = box that experienced a volume reduction)
<code>counterVolChangeRejectDocu</code>	rejected volume changes (boxIndex = box that experienced a volume reduction)
<code>counterInsertAcceptDocu</code>	accepted insertions in each box (boxIndex = box in which the particle is transferred to)
<code>counterInsertRejectDocu</code>	rejected insertions in each box (boxIndex = box in which the particle is transferred to)

It is important to note the Wolfram Language allows for arrays to be stored as so-called '*PackedArrays*', to minimize memory usage and speed up certain compiled computations. The calculation can be conducted more quickly if arguments of compiled functions are already stored as such. Imported values from the 'simulationDefinition' file are not set as '*PackedArrays*' by default, so the developer function `Developer`ToPackedArray` is used to explicitly set them as such. This currently only involves the variables `moleculeCoordVec`, `moleculeAtomsCoordVec`, `interEnergyStateVec` and `ffParamsNonBonded`. Even though `moleculeAtomsCoordVec` and `interEnergyStateVec` are not tensor objects, individual sub-parts can still be converted into '*PackedArrays*'. The implementation of the trial moves is done with that in mind to keep these variables as '*PackedArrays*' throughout the simulation.

Initial Energy Calculations

In this step, the variable `interEnergyStateVec`, representing the intermolecular energy, and the variable `intraEnergyStateVec`, representing the intramolecular energy of every molecule of the current state of the system, are initialized. To save computational time during the simulation, each interacting energy term between two molecules is stored in those variables. During trial moves, this list can be used to extract certain energy values without recalculating the whole system and only needs to be partially edited, in case the system was changed.

The virial terms are calculated for each box at the end of a cycle and stored for analysis. In a previous implementation, the virial term was calculated, stored and updated as a list similar to the intermolecular energy. This approach, however, was disregarded for time-saving purposes and due to the different distance calculation of r_{ab} and r_{ij} in case of poly-atomic molecules.

The variable `interEnergyStateVec` is structured as a quadratic array of size equal to the number of particles per box. The interaction energy between molecule a and b is stored at the position `interEnergyStateVec[[boxIndex,a,b]]`. To prevent double counting of molecule pairs, similar to the idea mentioned in section 4.1.1, only elements where $b > a$ are used for storage while the rest are kept at 0. This leads to a structure where each row contains one less stored value than the previous one. This means that the second row, in which the interactions of the second molecule are stored, does not contain the interaction between molecule 1 and molecule 2, since this pair was already stored in the first row. The last molecule in the list does not have any values stored in its row since all interactions are already present in the last column. With this structure, `interEnergyStateVec` can be stored as a 'PackedArray' in which all interaction of one molecule are present in a single row and column for quick extraction.

In the example shown in Figure 5.12, four molecules are interacting with each other and the structure of the list containing the energy values for those interactions is displayed on the right side. The colored arrows indicate the row where the interactions are stored.

The energy values for each of the interactions between two molecules are stored as a list of three elements that represent specific parts of the interaction energy calculation (Equation 4.1.5). The sum of all energy parts is split up in such a way that every term with a



Figure 5.12.: Example of four molecules with the corresponding interactions between them (arrows) on the left side and the structure of the array containing the energy information of those interactions on the right side.

different exponent of r_{ij} has its own list item. This format is convenient to improve the performance of the volume change trial move. More information about that is found in section 5.3.2.

$$\text{energy term} = U_{ij}(r_{ij}) = \underbrace{4\epsilon_{ij} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12}}_{\text{List Item 1}} - \underbrace{4\epsilon_{ij} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6}_{\text{List Item 2}} + \underbrace{\frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}}}_{\text{List Item 3}} \quad (5.3.2)$$

$$\text{virial term} = -\frac{\partial U_{ij}(r_{ij})}{\partial r_{ij}} = \underbrace{48\epsilon_{ij} \left(\frac{\sigma_{ij}^{12}}{r_{ij}^{13}} \right)}_{\text{List Item 1}} - \underbrace{24\epsilon_{ij} \left(\frac{\sigma_{ij}^6}{r_{ij}^7} \right)}_{\text{List Item 2}} + \underbrace{\frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}^2}}_{\text{List Item 3}} \quad (5.3.3)$$

To calculate the intramolecular energies, the variable `intraEnergyStateVec` is created. `intraEnergyStateVec` is a list containing a single energy value for every molecule, to keep the amount of list items short. Since the calculation of the intramolecular energy does not involve other molecules in the system, the calculation is very fast and can be applied in trial moves with little effort.

Energy Calculation Functions

The energy calculation is one of the most time expensive steps in the simulation. Therefore, a simple declaration using `SetDelayed` is out of the question since it is bound to evaluate relatively slowly while retaining a lot of freedom for the choice of arguments and variables.

Instead, the function `Compile` is used to create compiled functions in an external library, which are linked back to the Wolfram kernel.

The initial implementation involved the use of a general function for the interaction between two molecules and multiple functions for energy calculation per trial move implemented. The goal of using individual functions per trial move was to optimize the loops performed in individual trial moves and to separate the additional effort of additional features without slowing down the conventional implementation.

A significant downside to these functions, besides the amount of redundancy, was that, since they are compiled to low-level C code, they could only handle lists and quadratic arrays. This was problematic when simulating mixtures of molecules with different amounts of atoms (like ethane and methane). This suggested the development of the '*ArrayPadding*'-feature which padded each molecule with dummy atoms until the atom coordinate lists were tensor objects.

In version 2.4, the function `CalcInteractionsTwoMolecules` has been given the additional attribute '*Listable*', which allows fast parallel looping over multiple molecule interactions, even if the atom coordinate list is not a tensor object, making some of the above functions redundant.

The energy calculation functions currently only consist of a routine calculating the interaction between two molecules with the '*Listable*' attribute. The additional effort of features like the CFC insertion or the electrostatic long-range correction is reduced by using the Booleans `cfcActive` and `eLRCActive` as constants in the compiled functions. Additional features are introduced as separate compiled functions which are inlined into the main compiled function. A summary of the current energy calculation functions is listed below:

<code>ffEnergy</code>	returns the intermolecular interaction energy between two sites
<code>ffEnergyCFC</code>	returns the intermolecular interaction energy between one or two fractional sites
<code>checkLambdaCFC</code>	returns the scaling factor λ for interactions of a fractional molecule
<code>eLRCphi</code>	returns the dampening function value $\phi(\kappa, r_{ij}, R_c)$ for the electrostatic long-range correction

<code>CalcInteractionsTwoMolecules</code>	returns the intermolecular interaction between two molecules (has the 'Listable' attribute)
<code>VolChangeCalculateNewEnergy</code>	returns the intermolecular interactions based on a change of volume

Note that the function `VolChangeCalculateNewEnergy` is declared differently depending on whether only LJ fluids are simulated or not (cf. section 5.3.2).

Adjustments of Trial Move Limits

The trial moves are adjusted as described in section 4.5.1. At the end of every cycle, it is checked if an adjustment is necessary, using the function `adjustTrialMoveLimits`.

There are two ways implemented to check if an adjustment is necessary:

- **Conventional**

The first approach is to execute an adjustment after each cycle if at least two moves have been carried out for the respective move type. This algorithm proposed by M. P. Allen and D. J. Tildesley [1] works well for moves like translations of which several hundred are executed per cycle. By default, this approach is used.

- **Custom**

The second approach executes the adjustment only if a specified number of moves has been attempted. This procedure works also well for adjusting the trial move limits of volume changes of which only several are executed per cycle. This procedure is used if the custom parameter `trialMoveLimitsAdoptionMode` is set to "numberMoves".

5.3.2. Trial Moves

In this section, all the implementation steps and acceptance rules of the trial moves are described.[31, 32, 30] Every trial move is programmed as a function which is then used in the loops of the three execution phases to execute those trial moves. Those functions manipulate the variables of the system directly and return a string describing the outcome of the trial move, which is used for the timing documentation. The basic algorithm behind

every trial move consists of a change of the molecular state followed by a recalculation of the energy in the system and accepting the new state with a probability depending on the energy difference ΔE between the old and the new state. It is important to note that for trial moves that change the density of the system (e.g., volume changes, particle insertions and Widom insertions), the tail correction (cf. section 4.1.4) has to be included in the calculation of the energy difference between the old and the new state.[31] Because the units of all energies are given in kcal/mol in the following equations, they have to be divided by the universal gas constant R instead of the Boltzmann constant k_B .

In the following equations, ΔE is the energy difference between the old and the new state in [kcal/mol], R is the universal gas constant in [kcal/(mol K)], T is the temperature in [K], N is the number of molecules, V is the volume in [\AA^3] and P is the pressure in [kcal/(mol \AA^3)]. The superscripts *I* and *II* correspond to the first and the second box respectively.

Translation

1. Select randomly one box where a molecule should be translated
2. Select randomly one molecule in this box
3. Translate that molecule in a random direction with a distance between 0 and the trial move limit (and apply the periodic boundary condition)
4. Calculate the new energies due to the translation of that molecule
5. Accept the trial move with the probability p_T calculated with Equation 5.3.4

$$p_T = \exp \left[-\frac{\Delta E}{RT} \right] \quad (5.3.4)$$

To apply a translation to a molecule, two variables have to be edited. First, the variable `moleculeCoordVec` has to be edited to translate the center of the selected molecule. Then, `moleculeAtomsCoordVec` has to be edited so that every atom in the selected molecule is translated as well. Afterwards, the periodic boundary condition has to be applied to the translated positions to prevent molecule centers outside of the box.

Two special cases require fewer calculations to take place: If no molecule is present in the randomly selected box, the trial move is immediately rejected. If only one molecule is present, no interactions with other molecules are possible and therefore no new energy calculation has to take place after the translation, which also means that the trial move is always accepted.

Rotation

1. Select randomly one box where a molecule should be rotated
2. Select randomly one molecule in this box
3. Rotate that molecule with a random rotation axis with an angle between the negative and positive value of the trial move limit
4. Calculate the new energies due to the rotation of that molecule
5. Accept the trial move with the probability p_R calculated with Equation 5.3.5

$$p_R = \exp \left[-\frac{\Delta E}{RT} \right] \quad (5.3.5)$$

To rotate a molecule, only `moleculeAtomsCoordVec`, the variable that holds the positions of the atoms, has to be edited. The center of the selected molecule which is stored in `moleculeCoordVec` stays the same during the rotation.

The two special cases in which no molecule or only one molecule is present in the box are treated the same way as the two special cases in the translation trial move.

Volume Change with $V=\text{const.}$

1. Define a random volume change ΔV between the negative and positive value of the trial move limit ($-\Delta V_{\text{max}}$ and $+\Delta V_{\text{max}}$)
2. Change the volume of box 1 with ΔV and the volume of box 2 with $-\Delta V$
3. Modify the positions of all molecules in the boxes so that they scale to the new volumes
4. Calculate new total energies of both boxes

5. Accept the trial move with the probability p_V calculated with Equation 5.3.6

$$p_V = \exp \left[-\frac{\Delta E^I + \Delta E^{II} - N^I RT \ln \left(\frac{V^I + \Delta V}{V^I} \right) - N^{II} RT \ln \left(\frac{V^{II} + \Delta V}{V^{II}} \right)}{RT} \right] \quad (5.3.6)$$

If the trial move is programmed as stated above, the energy terms of the whole system would have to be recalculated for the scaled boxes. To prevent this step, it is possible to scale the energy terms without the calculation of all interactions, if they are stored in the right format.[42] How the energy terms are stored in the right format is described in section 5.3.1. After the volume change ΔV has been defined, the change of the box length for each box can be calculated and can then be used to define the scaling factors used in Equation 5.3.7 to calculate the new energy terms. If the virial terms are stored as a list similar to the intermolecular energy it can also be recalculated using the scaling factors described in 5.3.8. The current implementation, however, does not store the virial terms as a list. In case molecules with multiple atoms are in the system, the scaled energy is not valid, and the whole system is recalculated instead.

$$\begin{aligned} \text{scaled energy term} = & \left(\frac{L_{\text{old}}}{L_{\text{new}}} \right)^{12} \left[4\epsilon_{ij} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} \right] \\ & + \left(\frac{L_{\text{old}}}{L_{\text{new}}} \right)^6 \left[-4\epsilon_{ij} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] \\ & + \left(\frac{L_{\text{old}}}{L_{\text{new}}} \right) \left[\frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}} \right] \end{aligned} \quad (5.3.7)$$

$$\begin{aligned} \text{scaled virial term} = & \left(\frac{L_{\text{old}}}{L_{\text{new}}} \right)^{12} \left[48\epsilon_{ij} \left(\frac{\sigma_{ij}^{12}}{r_{ij}^{13}} \right) \right] \\ & + \left(\frac{L_{\text{old}}}{L_{\text{new}}} \right)^6 \left[-24\epsilon_{ij} \left(\frac{\sigma_{ij}^6}{r_{ij}^7} \right) \right] \\ & + \left(\frac{L_{\text{old}}}{L_{\text{new}}} \right) \left[\frac{1}{4\pi\epsilon_0} \frac{q_i q_j e^2}{r_{ij}^2} \right] \end{aligned} \quad (5.3.8)$$

If the trial move is accepted, all variables that also depend on the size of the boxes are

scaled accordingly. This includes the cut-off radius and the trial move limit of translations. It is important to note that this adjustment of the trial move limit also occurs during the production phase of the simulation. While this contradicts the idea of building a reversible Markov chain, it was deemed more important to preserve the acceptance ratio of the trial moves during the execution of a simulation.

$$\text{rCut}_{\text{new}} = \text{rCut}_{\text{old}} \cdot \left(\frac{L_{\text{new}}}{L_{\text{old}}} \right) \quad (5.3.9)$$

$$\text{transMax}_{\text{new}} = \text{transMax}_{\text{old}} \cdot \left(\frac{L_{\text{new}}}{L_{\text{old}}} \right) \quad (5.3.10)$$

Volume Change with $P=\text{const.}$

1. Select randomly one box for the volume change
2. Define a random volume change $\text{vol}\Delta V$ between the negative and positive value of the trial move limit ($-\text{volChangeMax}$ and $+\text{volChangeMax}$)
3. Change the volume of the selected box with $\text{vol}\Delta V$
4. Modify the positions of all molecules in the selected box so that they scale to the new volume
5. Calculate the new total energy of the selected box
6. Accept the trial move with the probability p_V calculated with Equation 5.3.11 if box 1 was selected or Equation 5.3.12 if box 2 was selected

$$p_V = \exp \left[-\frac{\Delta E^I - N^I RT \ln \left(\frac{V^I + \Delta V}{V^I} \right) + P \Delta V}{RT} \right] \quad (5.3.11)$$

$$p_V = \exp \left[-\frac{\Delta E^{II} - N^{II} RT \ln \left(\frac{V^{II} + \Delta V}{V^{II}} \right) + P \Delta V}{RT} \right] \quad (5.3.12)$$

This trial move is handled in the same way as the volume change trial move with constant total volume. This means that the scaling technique is used to keep the calculation time at a minimum unless molecules with multiple atoms are in the system.

Particle Insertion

1. Select randomly in which box a molecule is removed (donor box) and inserted in the other box (insertion box)
2. Select randomly which molecule type should be transferred to the insertion box
3. Select randomly a molecule in the donor box with the molecule type defined previously
4. Remove the selected molecule from the donor box and insert it into the insertion box with a random orientation
5. Calculate the change of energies in both boxes due to removal and insertion
6. Accept the trial move with the probability p_I calculated with Equation 5.3.13 if box 1 was selected as the donor box and with Equation 5.3.14 if box 2 was selected as the donor box, where i is the selected molecule type.

$$p_I = \exp \left[- \frac{\Delta E^I + \Delta E^{II} + RT \ln \left(\frac{V^I(N_i^{II}+1)}{V^{II}N_i^I} \right)}{RT} \right] \quad (5.3.13)$$

$$p_I = \exp \left[- \frac{\Delta E^I + \Delta E^{II} + RT \ln \left(\frac{V^{II}(N_i^I+1)}{V^I N_i^{II}} \right)}{RT} \right] \quad (5.3.14)$$

After the index of the molecule which will be deleted in the donor box has been defined, all information about the inserted molecule in the insertion box is generated. This includes the coordinates and names of the molecule and its atoms and the interaction labels for each atom. The molecule position and orientation are randomly generated.

If the trial move is accepted, the new molecule has to be integrated into the existing variables and the donor molecule has to be removed from the existing variables. The removal of the donor molecule is accomplished by deleting the items with the right indices in the variables. The integration of the new molecule is made by inserting the new information at the first index in the variables. This ensures easy integration of all interaction terms without editing

the variables too much. How the indices change by inserting the new terms is visualized in Figure 5.13, where a particle is inserted in a system of three particles.

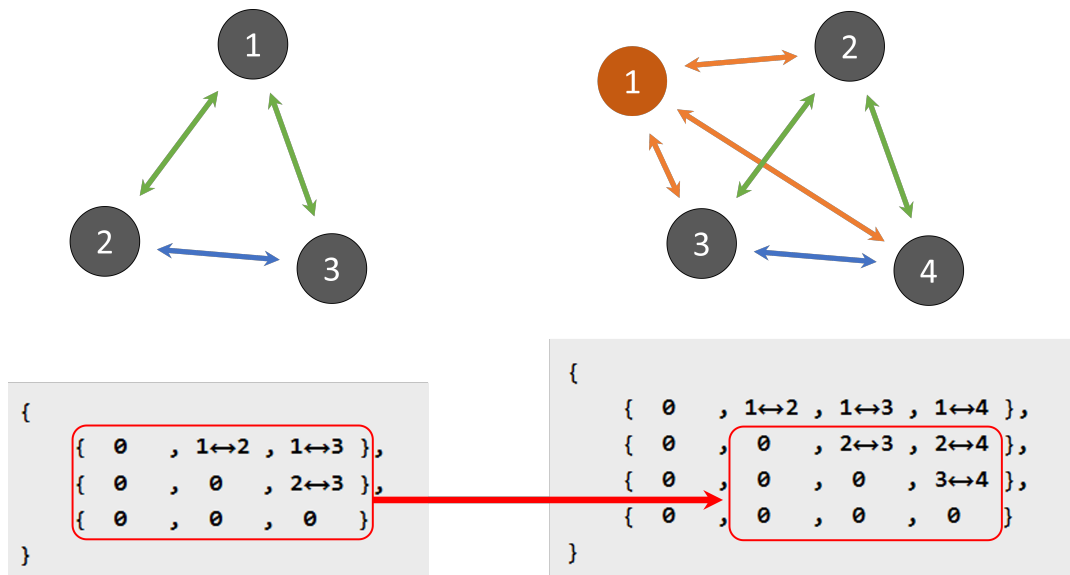


Figure 5.13.: Changes of the indices in the energy variables by an insertion of a new particle. The system on the left represents a box before the insertion and the system on the right represents a box after the insertion. The first row of the interaction list on the right shows the inserted interactions and the other rows hold the shifted indices of the old system.

Widom Insertion

Technically, the *Widom Insertion* [51] is not a trial move but follows a similar approach, with the important difference that it does not change the molecular state of the system. It is used to estimate the chemical potential of the components in each box.

1. Select randomly a box for an insertion
2. Select randomly which molecule type should be inserted
3. Insert this molecule in a random position with a random orientation into the selected box
4. Calculate the new energies due to the insertion of that molecule

5. Save the result of Equation 4.2.3 for the estimation of the chemical potential in later analysis (see section 5.7).

The creation of the inserted molecule information is the same as in the particle insertion trial move. After that, the interaction distances to all other atoms in the system are calculated to check if overlaps are present.

The partial contribution, which is the term in the angled brackets $\langle \dots \rangle$ of Equation 4.2.3, is stored as a sum of all values calculated in the same MC cycle. The sums of partial contributions per component are accompanied by counters that are incremented if no overlaps occurred and the sum has been updated.

It was investigated if overlapping positions should be rejected to reduce the number of high energy spikes in the simulation course. This approach was based on the work by Thaler.[45] However, in the study of Michael Haring [12] it was found that better agreement with experimental and literature data is achieved if no Widom insertions are rejected.

Figure 5.14 shows the average excess chemical potential of a binary mixture of hypothetical LJ-fluids using different values for the overlap radius used in the Widom insertion method. The results are compared to literature data.[48] In Figure 5.14 it can be seen that an overlap

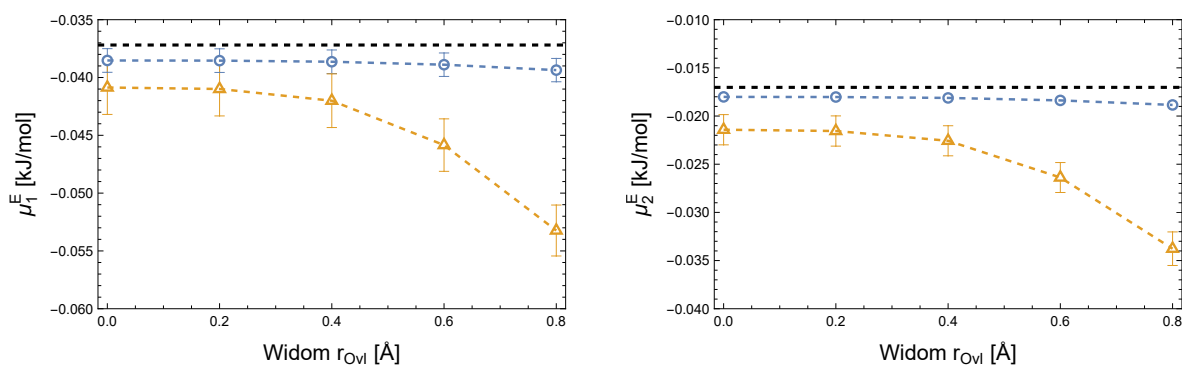


Figure 5.14.: Simulation results for the average excess chemical potential of a binary mixture of LJ-fluids. Simulations were conducted with different overlap radii for the Widom insertion method. $\sigma_{11} = \sigma_{22} = 1\text{Å}$; $\epsilon_{11}/k_B = 1.00\text{ K}$; $\epsilon_{22}/k_B = 0.50\text{ K}$; $T^* = 0.75$; $P^* = 0.065$; ○ Vapor Phase; △ Liquid Phase; ---- Literature Data [12]

radius close to $0.8 \cdot \sigma$ can cause a significant change in the chemical potential of the liquid phase. Therefore, it is recommended not to reject any Widom insertions. The occurring

energy spikes in the course of the simulation can be compensated for by using a running average.

Bond Stretching

The bond stretching trial move attempts to change the length of a bond constraint in a molecule. Each flexible bond in a molecule is specified by a list containing the indexes of the two atoms connected by the bond, a label determining the force field energy parameters and the maximum length adjustment Δl_{\max} . The first atom in the list is referred to as atom 1 and the second as atom 2 of the bond. If the bond between atoms 1 and 2 of a molecule is adjusted, then atom 1 stays fixed in global coordinates. Atom 2 and all atoms connected to it except atom 1 are translated by the same vector

$$\vec{v} = \frac{\vec{r}_2 - \vec{r}_1}{|\vec{r}_2 - \vec{r}_1|} \cdot \Delta l, \quad (5.3.15)$$

where \vec{r}_i is the position of atom i and Δl is the bond length adjustment. The process flow of the trial move is as follows:

1. choose a random simulation box
2. choose a random molecule present in the chosen box
3. choose a random flexible bond constraint of the chosen molecule
4. determine the bond length adjustment Δl from an equal distribution in the range $[-\Delta l_{\max}, \Delta l_{\max}]$
5. translate the molecule branch including atom 2 by the vector \vec{v}
6. recalculate the inter- and intramolecular interaction energies of the molecule
7. accept the trial move with the probability $p_{\text{acc},\text{bond}}$ calculated with Equation 5.3.16

$$p_{\text{acc},\text{bond}} = J_{\text{bond}} \cdot \exp \left[-\frac{\Delta E}{RT} \right], \quad \text{with} \quad J_{\text{bond}} = \frac{l_n^2}{l_o^2} \quad (5.3.16)$$

The Jacobian term J_{bond} includes the lengths l_n and l_o , which are the bond lengths of the new and old configuration, respectively.

Angle Bending

The angle bending trial move attempts to change the angle of a bond angle constraint in a molecule. Each flexible angle in a molecule is specified by a list containing the indexes of the three atoms connected by bonds that form the constraint, a label determining the force field energy parameters and the maximum angle adjustment $\Delta\phi_{\max}$. The atoms forming a bond angle constraint will be referred to in the order they occur in the list mentioned before. If the angle between atoms 1, 2 and 3 of a molecule is adjusted, then atoms 1 and 2 stay fixed in global coordinates. Atom 3 and all atoms connected to it except atom 2 are rotated around the position of atom 2.

The process flow of the trial move is as follows:

1. choose a random simulation box
2. choose a random molecule present in the chosen box
3. choose a random flexible angle constraint of the chosen molecule
4. determine the bond angle adjustment $\Delta\phi$ from an equal distribution in the range $[-\Delta\phi_{\max}, \Delta\phi_{\max}]$
5. rotate the molecule branch including atom 3 around atom 2
6. recalculate the inter- and intramolecular interaction energies of the molecule
7. accept the trial move with the probability $p_{\text{acc,angle}}$ calculated with Equation 5.3.17

$$p_{\text{acc,angle}} = J_{\text{angle}} \cdot \exp\left[-\frac{\Delta E}{RT}\right], \text{ with } J_{\text{angle}} = \frac{\sin(\phi_n)}{\sin(\phi_o)} \quad (5.3.17)$$

The Jacobian term J_{angle} includes the angles ϕ_n and ϕ_o , which are the angles of the new and old configuration, respectively.

Dihedral Torsion

The dihedral torsion trial move attempts to change the angle of a dihedral constraint in a molecule. Each flexible dihedral in a molecule is specified by a list containing the indexes of the four atoms connected by bonds that form the constraint, a label determining the force field energy parameters and the maximum angle adjustment $\Delta\theta_{\max}$. The atoms forming a dihedral constraint will be referred to in the order they occur in the list mentioned before. If the dihedral angle between atoms 1, 2, 3 and 4 of a molecule is adjusted, then atoms 1 to 3 stay fixed in global coordinates. Atom 3 and all atoms connected to it except atom 2 are rotated by the same angle around the axis formed by the bond between atoms 2 and 3.

The process flow of the trial move is as follows:

1. choose a random simulation box
2. choose a random molecule present in the chosen box
3. choose a random flexible dihedral constraint of the chosen molecule
4. determine the dihedral angle adjustment $\Delta\theta$ from an equal distribution in the range $[-\Delta\theta_{\max}, \Delta\theta_{\max}]$
5. rotate the molecule branch including atoms 3 and 4 around the dihedral axis
6. recalculate the inter- and intramolecular interaction energies of the molecule
7. accept the trial move with the probability $p_{\text{acc,dihedral}}$ calculated with Equation 5.3.18

$$p_{\text{acc,dihedral}} = \exp \left[-\frac{\Delta E}{RT} \right] \quad (5.3.18)$$

5.3.3. Execution Phase

As discussed in section 5.2.3, the simulation consists of three phases, the warm-up, equilibration and production phase that are executed in sequence. Each of those phases runs for a certain amount of cycles, each containing multiple trial moves.

Every phase has a specific purpose and is structured as a loop going through one cycle after the other until the specified number of cycles for that phase is reached. The execution

of one cycle consists basically of three steps: First, the specified number of trial moves is executed, whose order is randomly generated for every new cycle. Then, all the important results of that cycle are calculated and stored in variables for later analysis. The last step is the adjustment of the trial move limits to improve the acceptance rate. Which steps are made in each cycle is different for each phase. A short overview of which steps are executed or skipped in each phase is given in Table 5.1, where T, R, V, I, and W are the trial moves *Translation*, *Rotation*, *Volume Change*, *Insertion* and *Widom Insertion*, respectively.

Table 5.1.: Overview of tasks executed in every phase including the execution of Translations, Rotations, Volume changes, Insertions and Widom insertions

	trial moves					save important results	adjust trial move limits
	T	R	V	I	W		
Warm-up	✓	✓	×	×	×	×	✓
Equilibration	✓	✓	✓	✓	✓	✓	✓
Production	✓	✓	✓	✓	✓	✓	×

Warm-up Phase In the *warm-up* phase only translation and rotation trial moves are made to force a molecular structure and remove as many overlaps as possible. Overlaps should already be prevented by the use of the grid system in the setup of the simulation, but for larger molecules it is possible that parts of those molecules can overlap due to the random rotation in the initialization of the grid system. In this phase, the calculation and the storage of results are skipped because, with only translations and rotations, no equilibrium can be reached. The adjustments of the trial move limits are made for the translations and rotations.

Equilibration Phase The *equilibration* phase includes all three steps. All trial moves are conducted, the results are calculated and stored and the trial move limits are adjusted. This phase is used to equilibrate the system, after which it just fluctuates around certain values that should represent the macroscopic fluid.

Production Phase The *production* phase is primarily used to monitor the equilibrated system for a while to obtain a large amount of values which can then be averaged to get final

results. In this phase, every trial move is made and the results are calculated and stored. Adjustments of the trial move limits are not performed in this phase, because they should already be optimized from previous phases.

5.3.4. 02b - ExecutionHandler.wls

With the additional notebook '*02b - ExecutionHandler.wls*', sets of simulations, indicated by their folder path, can be executed and/or analyzed. The execution handler is accompanied by '*ProtocolSimulations.xlsx*' in which the obtained results of the individual simulations can be conveniently compared and summarized.

The simulations to be executed can be added in two ways:

- **Series**

If this option is selected, all simulations stored in a folder are added.

- **Simulation**

If this option is selected, only the simulation selected is added.

Recommended Usage

The usage of the GEMC package is subject to personal preferences. Yet the following two 'limiting cases' shall illustrate the idea behind the process handler.

- **Long simulations**

For long simulations it is recommended to use the '*02a - Execution.wls*' directly and run the simulations individually on separate Mathematica kernels. Then the results of these simulations are synchronized with the local machine using GIT. On the local computer, all simulations are evaluated together.

- **Short Simulations**

For short simulations, e.g. parameter tests, it is recommended to run all simulations sequentially (overnight) on a single kernel since then the analysis results are generated automatically and the effort to configure the Mathematica kernels and executions is omitted.

5.3.5. 02c - ExecutionUnitTests.wls

With this notebook, parts of the execution can be tested separately so that possible errors can be found. It is structured into three chapters:

Init In this chapter, the unit tests are initialized. It is recommended to restart the Mathematica kernel before executing any unit tests. After selecting a short exemplary simulation setup, '*MC Initialization*' and the other helper functions are executed to load the required functions.

Testing In this chapter, the required tests can be executed and their results are visualized. Currently, test functions are implemented to check if the packages are loaded, if the format of certain variables is correct and to assert the energy functions in the trial moves.

Results In this chapter, the results of all tests are summarized. More details can be found in the comments of the notebook itself.

5.3.6. 02d - ExecutionDebugging.wls

This notebook is used to develop new algorithms by checking if the intermolecular interactions are consistent after a certain trial move has been executed. The basis for this comparison is the rigorous recalculation of all intermolecular interactions. Therefore, each check consists of the following steps:

1. Initialize

In this step, the intermolecular energy state vectors are updated by recalculating all intermolecular energies. Thereby it is ensured that the system is consistent before the test is started. Afterwards, the total internal energy is stored for each box in a separate variable so that the energy change due to a trial move can be calculated after the trial move has been executed.

2. Execute

In the second step, the trial move is accepted and all proposed changes are implemented in the state vectors. Also, the calculated energy change is calculated *without* the energy contribution of the tail corrections.

3. Test

Finally, the calculated energy change by the trial move function is compared against the rigorously calculated energy change. Also, it is checked if the updated intermolecular state vectors are correctly updated. This is done by recalculating all intermolecular interactions and comparing them against the updated state vectors.

For efficient debugging, it can be useful to deactivate the Metropolis check and accept all proposed trial moves. More details can be found in the comments of the notebook itself.

5.4. 03a - Analysis.wls

In this notebook, all the analysis of the simulation results takes place. This includes the calculation of certain properties and the visualization of the evolution of those properties over the cycles.

In the *'Import/Export'* section it is possible to enable export of the plots and results from this notebook by setting `exportFlag` to `True`. If the export is enabled, the variable `exportDir` defines the location in which all the files should be exported.

In case a simulation has been extended with the *Extension* notebook (cf. section 5.5), it is possible to combine the simulation results with the base simulation. This feature is discussed in detail in section 5.4.5.

5.4.1. Calculations of Thermodynamic Properties

The following equations only represent the calculations of certain properties of a single box. In the analysis notebook, all the calculations are carried out for both boxes. The equations also include conversion factors used to obtain the properties in reasonable units.

GEMC - ANALYSIS	
This notebook analyses the results of a simulation.	
▼	Setup
▼	Setup Configuration
▼	Initial Conditions
▼	Simulation Results
▼	Ensemble Averages
▼	Comparison & Checks
▼	Test specific evaluations
▼	Export Analysis Results

Figure 5.15.: Structure of the *Analysis* notebook**Pressure**

$$P = \left[\frac{N \cdot R \cdot T}{V} + \frac{virial}{V} + \frac{P_{Tail}}{V} \right] \cdot 10^{30} \cdot 4184 \cdot \frac{1}{N_A} \cdot \frac{1}{10^5} \quad (5.4.1)$$

Table 5.2.: Description and units of symbols used in Equation 5.4.1

P	...	pressure	[bar]
N	...	number of molecules	[-]
R	...	universal gas constant	[kcal/(mol K)]
T	...	temperature	[K]
V	...	volume	[Å ³]
$virial$...	virial term	[kcal/mol]
P_{Tail}	...	pressure tail correction	[kcal/mol]

Continued on next page

Table 5.2 – continued from previous page

N_A	...	Avogadro constant	[1/mol]
-------	-----	-------------------	---------

Table 5.3.: Conversion factors used in Equation 5.4.1

10^{30}	...	$[1/\text{\AA}^3] \Rightarrow [1/\text{m}^3]$
4184	...	$[\text{kcal}] \Rightarrow [\text{J}]$
$\frac{1}{N_A}$...	$[1/\text{mol}] \Rightarrow [-]$
$\frac{1}{10^5}$...	$[\text{Pa}] \Rightarrow [\text{bar}]$

Potential Energy

The potential energy is already calculated and stored with the desired units, but the unit 1/mol is misleading, because the value of the potential energy actually corresponds to the sum of all molecules (indicated by the index 'all' in equations 5.4.2 and 5.4.3). Because of this, the value has to be divided by the number of molecules in the box.

$$U_{\text{Inter}} = \frac{U_{\text{Inter,all}}}{N} + \frac{U_{\text{Tail}}}{N} \quad (5.4.2)$$

$$U_{\text{Intra}} = \frac{U_{\text{Intra,all}}}{N} \quad (5.4.3)$$

$$U_{\text{Total}} = U_{\text{Inter}} + U_{\text{Intra}} \quad (5.4.4)$$

Table 5.4.: Description and units of symbols used in Equations 5.4.2, 5.4.3 and 5.4.4

U_{Inter}	...	intermolecular potential energy	[kcal/mol]
U_{Tail}	...	energy tail correction	[kcal/mol]
U_{Intra}	...	intramolecular potential energy	[kcal/mol]

Continued on next page

Table 5.4 – continued from previous page

U_{Total}	...	total potential energy	[kcal/mol]
N	...	number of molecules	[-]

Mole Fractions

$$x_i = \frac{N_i}{\sum_{j=1}^{N_m} N_j} \quad (5.4.5)$$

Table 5.5.: Description and units of symbols used in Equation 5.4.5

x_i	...	mole fraction of the molecule type i	[-]
N_i	...	number of molecules of the type i	[-]
N_m	...	number of molecule types	[-]
N_j	...	number of molecules of the type j	[-]

Molar Volume

$$v = \frac{V}{N} \cdot \frac{1}{10^{30}} \cdot 10^3 \cdot N_A \quad (5.4.6)$$

Table 5.6.: Description and units of symbols used in Equation 5.4.6

v	...	molar volume	[dm ³ /mol]
V	...	volume	[Å ³]
N	...	number of molecules	[-]

Continued on next page

Table 5.6 – continued from previous page

N_A	...	Avogadro constant	[1/mol]
-------	-----	-------------------	---------

Table 5.7.: Conversion factors used in Equation 5.4.6

$\frac{1}{10^{30}}$...	$[\text{\AA}^3] \Rightarrow [\text{m}^3]$
10^3	...	$[\text{m}^3] \Rightarrow [\text{dm}^3]$
N_A	...	$[-] \Rightarrow [1/\text{mol}]$

Chemical Potential

In the execution of the simulation, a partial of the chemical potential ($\mu_{i,\text{partial}}$) was already calculated and stored for later analysis. Equation 5.4.7 corresponds to the chemical potential of the molecule type i in one box [1]. The angle brackets $\langle \dots \rangle$ indicate that this part of the equation has to be averaged over the ensemble states. This is done by accumulating the partials for each cycle with all the previous cycles and dividing by the amount of successful Widom insertions (which were used to sum up the partials for each cycle) up to this point. For example, for the 15th cycle all partials of the first 15 cycles are averaged for the calculation of the chemical potential.

$$\mu_i = -k_B T \ln \left[\frac{1}{\lambda_i^3} \left\langle \underbrace{\left(\frac{V}{N_i + 1} \right) \exp \left[-\frac{\Delta E}{RT} \right]}_{\mu_{i,\text{partial}}} \right\rangle \right] \cdot N_A \cdot \frac{1}{10^3} \quad (5.4.7)$$

$$\lambda_i = \frac{h}{\sqrt{2 \cdot \pi \cdot m_i \cdot k_B \cdot T}} \cdot 10^{10} \quad (5.4.8)$$

Table 5.8.: Description and units of symbols used in Equation 5.4.7 and 5.4.8

μ_i	...	chemical potential of the molecule type i	[kJ/mol]
k_B	...	Boltzmann constant	[J/K]
T	...	temperature	[K]
λ_i	...	thermal de Broglie wavelength	[Å]
$\mu_{i,\text{partial}}$...	partial of the equation calculated in the execution phase	[Å ³]
V	...	volume	[Å ³]
N_i	...	number of molecules of the molecule type i	[-]
ΔE	...	energy difference due to the Widom insertion	[kcal/mol]
R	...	universal gas constant	[kcal/(mol K)]
N_A	...	Avogadro constant	[1/mol]
h	...	Planck constant	[J s]
m_i	...	molecule mass of the molecule type i	[kg]

Table 5.9.: Conversion factors used in Equation 5.4.7

N_A	...	[-] \Rightarrow [1/mol]
$\frac{1}{10^3}$...	[J] \Rightarrow [kJ]

Table 5.10.: Conversion factors used in Equation 5.4.8

10^{10}	...	[m] \Rightarrow [Å]
-----------	-----	-----------------------

5.4.2. Plots and Charts

In the section '*Simulation Results/Show Plots*', the ranges of the plots drawn can be specified with the following two variables:

- plotRange** This variable defines which cycles should be visualized for all the properties. Two scenarios are important. The first one is to display all the documented results with the value `{1,nDocuCycles}`. The second scenario is to display a certain range of cycles with, e.g., the value `{200,500}` to display the range from cycle 200 up to cycle 500.
- vertRange** This variable defines the range of values on the y-Axis of all plots. It is not recommended to use certain value ranges, because every plot has a different value range. The two recommended values are `Automatic`, which adjusts the range automatically to the most relevant parts, and `All`, which includes every point of the data in the plot including outlier points.

5.4.3. Ensemble Averages to obtain Bulk Properties

To obtain the final bulk results of the simulation, the results have to be averaged over a certain range. This range should only include the cycles at which the system is already in equilibrium and fluctuates around a certain value. To find out where the system has reached equilibrium, one should look at the plots in the previous section and check where the values do not change continuously and only fluctuate around certain values. If mixtures are used, the easiest method is to check the plots with the mole fractions.

To set up the range that should be used to calculate the ensemble averages, the variable `meanRange` in section '*Setup/Ensemble Averages*' has to be set. This variable will be used as a part specification of lists and needs to be defined as a `Span` of integers. The syntax is as follows: `from;;to`. By default, only the beginning of the range should be specified, and the ending of the range should be set to `-1`, which corresponds to the last documented cycle. For example, considering a range from cycle 500 up to the last cycle, the variable should be set like the following: `meanRange = 500;;-1`.

When the range is set up, the calculation of the averages is straightforward in the sense that the calculated values of the previous sections are averaged over the specified range.

5.4.4. Export of Simulation Data

At the end of the analysis, the notebook is printed as a PDF document. Thereby, it can be easily opened at any time, without having to run the analysis again.

Also the files '*analysisResults.m*' and '*analysisResults_Excel.m*' are created, in which the analysis results are exported as an association. Thereby, the results of different simulations can be combined easily and further evaluated using the '*03b - CombinedAnalysis.wls*' notebook.

5.4.5. Analysis of Multiple Continuous Simulations

After an extension of a simulation, that has been created with the *Extension* notebook (cf. section 5.5), it can be difficult to get a full view of the entire simulation development. Results of multiple simulations can be combined by setting `extendedAnalysisExecute` in section *Setup/Import & Export/Extended Analysis* to `True`. If the notebook is executed after enabling this feature, the base simulation needs to be selected first. Afterwards, a prompt window will ask if an extension simulation should be added. Select 'Yes' and navigate to the folder of the extension simulation. This can be done repeatedly until 'No' is selected in the prompt window.

When an extension is added to the analysis, the function `ImportResultAssosiation` is used to extract the results from the selected simulation as an association. Compatibility checks are performed before the association is created. The function `AddResultAssosiation` is used to properly combine the imported association with the global variables containing the already imported simulation results. When combining two simulations, the function differentiates between variables that should be added, appended or overwritten. Only variables listed in the definition of `AddResultAssosiation` will be considered for combined analysis. In case a variable is added to '*simulationResult.m*', or a custom parameter creates an additional '*.m'-file, the new variables need to be added to the appropriate list for adding, appending or overwriting.

5.4.6. 03b - Combined Analysis.wls

This notebook can be used to visualize the results of sets of simulations and compare the results against each other as well as literature references and EoS. It consists of the following sections:

Helper Functions In this section all required functions to load data, generate plots and generate tables are set up.

Load the simulation series In this section the actual simulation results are loaded. In case a new evaluation is started, all variables can be unset by executing `ClearVariables[]`. Thereby, the *Helper Function* section does not have to be executed again. Before loading the simulation data, the way the simulations are loaded needs to be configured. In principle, there are two options:

- **Load all data**

If `loadSimulationCourse` is set to true, all simulation results are sequentially loaded. If all simulation results are loaded, all calculations usually carried out in the *Analysis* notebook are done and, thereby, all calculated quantities (pressure, chemical potential) are available. Furthermore, additional evaluations can be executed by overwriting the `returnHandle[]` method. This method will be executed after a new simulation has been completely loaded and all primary analysis calculations have been executed. It is recommended to store the custom-defined `returnHandle[]` in a separate notebook so that evaluations can be reproduced easily, if necessary.

- **Only load the results of the analysis**

If this option is used, only the results of the *Analysis* notebook stored in the result association are loaded. This option is significantly faster and therefore recommended whenever possible.

After the simulation series has been loaded, the user has to enter a name, which will be displayed in the plots. It is possible to load all simulations within a given folder or to load a single simulation. In case extensions have been conducted, more simulations are present in the directory than one might want to display in a single series. This issue can be overcome by

adding a `$` to the folder name. Thereby, the simulation is ignored by the combined analysis, and only the results of interest can be evaluated.

Finally, literature data can be loaded. It is important to ensure that the keys of the loaded simulation results and literature data are matching.

Compare the simulation series In this section, the loaded data can be visualized and compared. First, it needs to be defined in *Setup* which datasets shall be compared by defining the variable `indices`.

Afterwards, the *Simulation Progress* can be visualized if the simulation course has been loaded previously. This visualization is followed by numerous visualization options like figures, tables, EoS comparisons, etc.

Test Specific Comparison Plots In this section, some exemplary plots for test-specific comparison are defined.

5.4.7. Custom Evaluations

It is often useful to investigate a specific aspect of the algorithm in detail. Therefore, a '*Custom Evaluation*' can be conveniently used, allowing a systematic adjustment of the software package. To ensure an easy setup, a template '*_template.nb*' and an example '*SwapInvestigation.nb*' are prepared in the folder '*customEvaluations*'.

The basic principle of the '*Custom Evaluation*' is that methods can be defined which are called (i) before the MC cycles are executed, (ii) during the execution of the MC cycles, (iii) after the execution in the analysis of the conducted simulation, as illustrated in Figure 5.16. Thereby, both code can be added to the simulation package and existing functions can be altered by simply overwriting them.

Consequently, the main code does not need to be altered for an evaluation which might only be carried out a couple of times. By doing so, the main code is kept tidy and easier to maintain.

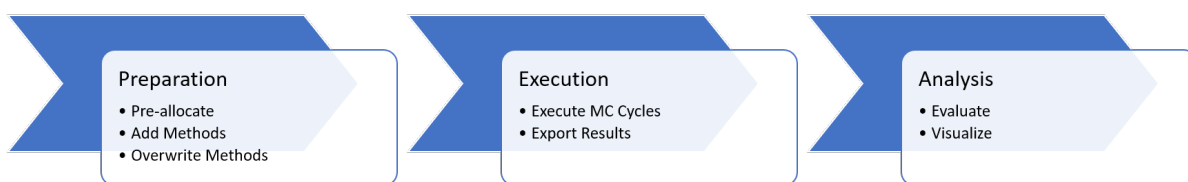
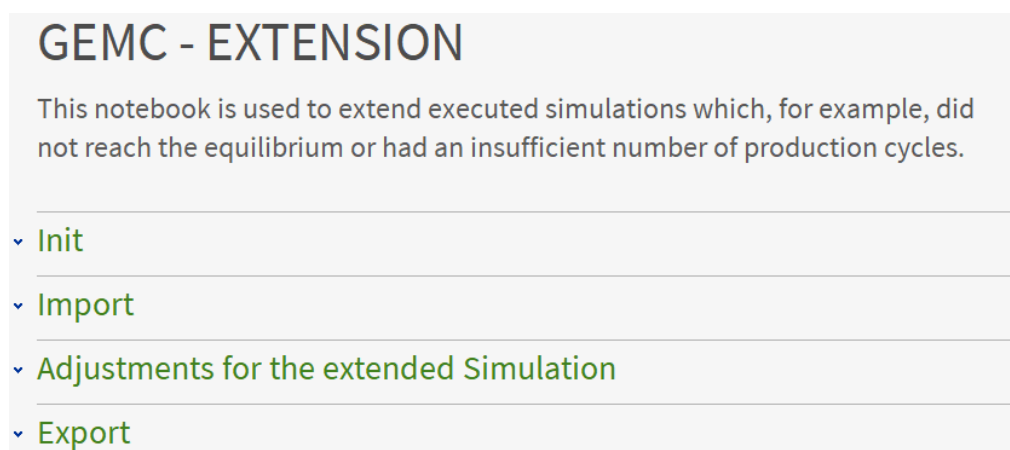


Figure 5.16.: Principle of a custom evaluation

5.5. 04 - Extension.wls

In case a simulation did not reach equilibrium or not enough cycles were executed for a meaningful averaging of the system properties, it is possible to extend finished simulations. The *Extension* notebook creates a new simulation folder with a simulation definition corresponding to the final configuration of the finished simulation. First, the settings of the old

Figure 5.17.: Structure of the *Extension* notebook

simulation are imported. Then, the results are imported which overwrites some variables to obtain the current state of the system. This includes the current positions of all molecules and atoms, the number of molecules in each box, the current box sizes, the cut-off radii and the adjusted trial move limits.

In the next section, some manual overwriting can be done. First, the number of additional cycles should be specified, and, because this is an extension of a simulation, basically no warm-up cycles should be set up although there is the possibility to include them. It is also

possible to redefine the number of trial moves and all the limits that should be used in the simulation.

After everything has been set up for the extended simulation, it can be exported in the 'Export' section. Thereby a folder is created, named like the folder of the finished simulation with the additional suffix '-EXTENDED'. The simulation settings are exported into that folder the same way as it is done in the setup notebook.

Now, this folder can be treated the same way as a new simulation made with the setup notebook. The simulation can be executed with the execution notebook and analyzed with the *Analysis* notebook. If needed, the extended simulation can also be extended again in the same way as before.

5.6. Packages

In the following section, the different packages are discussed in alphabetical order. The purpose of this section is to provide an overview of the different packages rather than describe them in detail. Table 5.11 shows which packages are loaded in the notebooks. While the two letters in the top row correspond to the first two letters of the package name, e.g. "Ch" for "ChemicalThermo" the name of the Notebook is stated in the first column.

Table 5.11.: Overview of the packages (Analysis, Chemical Thermodynamics, ForceFieldOPLS, MolecularSampling, Packages, Property - Package))

	An	Ch	Fo	He	Mo	Pa	Pr
Setup	×	✓	✓	✓	✓	✓	✓
Execution	×	✓	✓	✓	✓	✓	✓
Execution Handler	×	×	×	✓	×	×	×
Analysis	✓	✓	✓	✓	✓	✓	✓
Combined Analysis	✓	✓	✓	✓	✓	✓	✓
Extension	×	✓	✓	✓	✓	✓	✓

5.6.1. Analysis

This package contains the plot definitions and calculations. It is used in the *Analysis* and *CombinedAnalysis* notebook and consists of the following sections:

- **Initialization**

In this section, required variables are initialized.

- **Calculations**

In this section, the methods that execute the required calculations are defined. It is important to call the function `ExecuteCalculations[]` before any other evaluations.

- **Plot Definitions**

In this section, the methods that visualize the simulation course and results are defined.

- **Ensemble Averages & Comparison**

In this section, the methods that visualize the ensemble averages are defined.

- **Test specific evaluations**

In this section, the methods that visualize the test specific visualizations like the *GECFC* and *improved swap algorithm* method are defined.

5.6.2. ChemicalThermo

This package is a repository of functions frequently used in chemical thermodynamics. Of particular interest are the implemented equations of state (EoS) which are used to evaluate simulation results. The following three EoS are used for prediction:

- **LJ fluid EoS**

The EoS for LJ fluids by Thol (Thol EoS) [46] is used to evaluate the simulation results of pure LJ fluids and obtain reasonable start values.

- **Volume-translated Peng-Robinson EoS for LJ fluids**

The volume-translated Peng-Robinson EoS for LJ fluids (t-PR-LJ EoS) [15] is used to check the plausibility of simulation results of binary mixtures of LJ fluids.

- **Soave-Redlich-Kwong EoS**

The Soave-Redlich-Kwong EoS (SRK EoS) is used for predictions of molecules that are not LJ fluids. The critical point data is extracted from the *Property* package.

5.6.3. ForceFieldOPLS

This package provides the parameters for the OPLS force field. It is possible that some parameters needed for specific simulations are not yet present in this package. In that case they need to be added to the corresponding variables. It consists of two sections:

- **Molecular Data**

In this section molecular data like *Van der Waals radii* and *molar masses* are defined.

- **The OPLS-AA Force Field**

In the OPLS section of the package all the force field parameters [20] are stored. The variables and the format of the stored values are listed below.

`vdwRadius` Van der Waals radii of different atoms in [\AA] (from [3])

`molMass` molecular masses of different molecules [g/mol]

Both variables store their values as associations whose keys are strings that identify the type of an atom or molecule, where the value corresponds to its individual number. The variable `vdwRadius` is mainly used for the visualization of the atoms in graphics, and the variable `molMass` is used for the calculation of the thermal de Broglie wavelength.

`oplsNonBonded` parameters for non-bonded interactions in the format:

```
<| "label"→{q[1/e],σ[Å],ε[kcal/mol]} |>
```

`oplsBondStretching` parameters for bond stretching in the format:

```
<| "label"→{req[Å],Kr[kcal/(mol Å2)]} |>
```

`oplsAngleBending` parameters for angle bending in the format:

```
<| "label"→{θeq[deg],Kθ[kcal/(mol rad2)]} |>
```

`oplsTorsion` parameters for torsional energy functions in the format:

```
<| "label"→{V1,V2,V3,f1,f2,f2,Φ} |>
```

The parameters in `oplsTorsion` V_1 , V_2 and V_3 are stored in [kcal/mol], f_1 , f_2 , f_3 and Φ , which will be substituted with the maximum torsion angle, are stored in [deg].

5.6.4. HelperFunctions

This package contains general helper functions used throughout the project. It consists of the following sections:

- **Result Export**

This section contains functions handling the export of simulation and analysis results. In particular, the simulation results are compressed in a *ZIP* archive to save disc space, typically achieving a compression ratio of 10 : 5. Further, there are functions to export figures as a *PDF* and *PNG* file and tables as \LaTeX documents. If special characters are used for the table headers, it might be necessary to manually adjust the created '*.tex'-file.

- **Notebook Handling**

In this section, a function providing a copy of the *Analysis* notebook is defined. It is needed for the *ExecutionHandler* since the output cells of '*.wls'-files are not printed if they are executed automatically. In order not to affect the original '*.wls'-file, a copy is made. The copy is then opened and saved as a '*.nb'-file. Both files are deleted after finishing the evaluations.

- **Custom Parameters**

In this section, functions to handle the custom parameters are stored.

- **Debugging Helper**

In this section, the function `tEcho[]` to pause after each `Echo[]` call is defined. It can be used to prevent a crash of the Mathematica kernel. The default time-out is 0.5 seconds.

- **Base Directory Selection**

In this section, the function handling the setup of the base directory is set and the simulation definitions as well as the custom parameters are loaded.

- **Import & Export**

In this section, the functions to create '*simulationDefinition.m*' and '*simulationResults.m*' are defined. It is important to note that the variables are passed as strings on purpose to avoid the shadowing of variables. Further functions to export test-specific variables are defined as well.

- **Testing**

In this section the simple function `HelloWorld[]` is set up. It can be used to test if the packages are loaded correctly.

5.6.5. MolecularSampling

This package includes functions for molecule manipulation, visualization as well as basic functions for inter- and intramolecular interactions. This package is not only used for this project, so not every function it contains is used in the notebooks. The main features used include:

Mol-File Import The function `importMolFiles` is used to import the molecular structure of a molecule. This function needs a list of molecule types in the system and the directory, where all the '*.mol'-files are stored, as arguments. The '*.mol'-files that match the molecule types are imported from the directory. Note that the directory needs to be relative to the evaluating notebook directory.

If a system of molecule types is imported, the returned data is a list containing an association for every molecule type which includes every atom with its position. An example is shown in Figure 5.18, where methanol and nitrogen are imported.

Monte Carlo Functions The functions defined in this section are mainly used by this project and its notebooks. Here, just the most important functions are described, but there are more functions defined in the package.

```

{
  <|
    1 → {"O",0.7079,0.,0.},
    2 → {"C",-0.7079,0.,0.},
    3 → {"H",-1.0732,-0.769,0.6852},
    4 → {"H",-1.0731,-0.1947,-1.0113},
    5 → {"H",-1.0632,0.9786,0.3312},
    6 → {"H",0.9936,-0.8804,-0.298}
  |>,
  <|
    1 → {"N",0.9063,0.0509,0.0858},
    2 → {"N",2.3263,0.0509,0.0858}
  |>
}

```

Figure 5.18.: Data structure of the imported system of methanol and nitrogen

`visualizeBoxesMC[atomCoordinates,atomNames,boxVolumes,plotSize]`

This function is used to visualize the two boxes with all their molecules in them. The first three input parameters contain information about the coordinates and the names of all atoms and the volumes of both boxes. The last input parameter is used to set the size of the graphics and defines the option `ImageSize` in the graphics definition.

`GetMoleculeCenter[atomCoordinates]`

This function returns the geometric center of the molecule based on the positions of all atoms.

`GetNonBondedIntraPairs[moleculeBonds]`

This function is used to identify all the intramolecular non-bonded atom pairs that are separated by three or more bonds. This information is then used to calculate the intramolecular non-bonded energy (see section 4.1.1). The molecule bonds are passed to the function as a list consisting of two indices that indicate the bond between the two atoms with those indices. In the case of water, for example, where one hydrogen atom has the index 1, the oxygen atom has the index 2 and the other hydrogen atom has the index 3, the list would look like this: `{{1,2},{2,3}}`.

```
randomOrientationChange[atomCoordinates,angle]
```

The rotation of a molecule with a certain angle is accomplished with this function. The axis of rotation is created randomly, and the coordinates given as the first input parameter are rotated around the zero point which means that the atom positions in the simulation environment have to be centered before they are given to this function.

```
randomMoleculeRotationMC[atomCoordinates]
```

This function is used in insertion steps to create a random rotation of the molecule. The axis of rotation and the angle are created randomly.

```
CreateBoxCoordinates[numberOfMolecules,boxLength]
```

With this function, the box coordinates are created with the FCC grid as described in section 5.2.5.

```
metropolis[ΔE,R,T]
```

The Metropolis function returns a boolean value that defines if a trial move has been accepted (True) or rejected (False). First, it is checked if the value of ΔE is lower than 0, which means that the new molecular state has less potential energy and the trial move is immediately accepted. If ΔE is greater than 0, the new molecular state has higher potential energy and the trial move is accepted with the probability defined with Equation 5.6.1. Note that, depending on the units of ΔE , the value of R is either the universal gas constant or the Boltzmann constant.

$$p_A = \exp \left[-\frac{\Delta E}{RT} \right] \quad (5.6.1)$$

Energy and Force Calculations The following functions are used to calculate the interaction energy described in section 4.1.

```
ffEnergy[rij, qij,total, σij, εij]
```

This function returns the intermolecular interaction energy between the atoms i and j with the distance r_{ij} apart from each other. The calculation is carried out as stated in Equation 5.3.2.

```
ffVirial[rij, qij,total, σij, εij]
```

This function returns the virial of the intermolecular interaction force between the atoms i and j with the distance r_{ij} apart from each other. The calculation is carried out as stated in Equation 5.3.3.

```
ffEnergyNonBondedIntra[coord, atomIndicators, nonBondedPairs, params]
```

This function calculates the intramolecular energy of non-bonded atom pairs. The pairs are defined with `nonBondedPairs` which can be calculated using `GetNonBondedIntraPairs`. The parameters are given by the variable `ffParamsNonBonded`.

```
ffEnergyBondStretching[coord, params]
```

This function calculates the intramolecular energy of stretched bonds. The parameters are given by the variable `ffParamsBondStretching`.

```
ffEnergyAngleBending[coord, params]
```

This function calculates the intramolecular energy of bent angles. The parameters are given by the variable `ffParamsAngleBending`.

```
ffEnergyTorsion[coord, params]
```

This function calculates the intramolecular energy of rotated bonds. The parameters are given by the variable `ffParamsTorsion`.

```
ffEnergyIntra[coord, atomIndicators, nonBondedPairs, paramsList]
```

This function combines all intramolecular energies into one calculation. The parameters are given as a list of all the certain parameter variables `ffParamsNonBonded`, `ffParamsBondStretching`, `ffParamsAngleBending` and `ffParamsTorsion`.

5.6.6. Packages

This package involves the handling of packages, resetting of variables and setting system-wide options. It is necessary to ensure that the different packages are loaded in the right order to avoid shadowing of variables. This is accomplished with the function `InitGEMCpackages`.

The function `InitGEMCconfiguration` is used to reset state variables, set the seed for the random number generator and configure options of `Compile`, such as:

- **\$CompilationTarget**

By default, the Wolfram virtual machine (WVM) is used as a compilation target. A locally installed C compiler can be used as well to speed up evaluation by setting `$CompilationTarget` to "C".

- **Parallelization**

If a compiled function has the attribute 'Listable', it is possible to parallelize the evaluation. By default, `Parallelization` is set to `False`, but can be enabled in case large molecules are simulated by setting `Parallelization` to `True`. It is important to note that the CPU time is only comparable if simulations are executed with the same number of parallel threads. The process of distributing tasks across threads adds CPU time spent in the session.

5.6.7. Property

This package is a repository of frequently used physical property data. It consists of the following sections:

- **Constants**

In this section, commonly used constants like the Planck-constant or conversion factors are stored.

- **Dataset of Frequently Used Physical Properties**

In this section, a database for physical property data is set up. In particular, critical data, Lennard-Jones parameters, Antoine parameters, molar masses or the heat of evaporation are given.

The constants are used to have consistent usage without copying the values each time they have to be used. The following constants are defined in the package:

<code>kB</code>	Boltzmann constant in [J/K]
<code>NAvogadro</code>	Avogadro number in [1/mol]
<code>RJoule</code>	universal gas constant (SI-units) in [J/(mol K)]
<code>RCalor</code>	universal gas constant (caloric units) in [kcal/(mol K)]
<code>hPlanck</code>	Planck constant in [J s]
<code>uAtomic</code>	unified atomic mass unit in [kg]

5.7. Conventions & Workflow

In the following section, conventions used in the implementation are presented. The aim of these is to be consistent throughout the project and thereby simplify both development and maintenance.

5.7.1. Program Structure

To ensure a straightforward workflow, the program is structured into three folders and nine notebooks as shown in Figure 5.19. All workbooks are supported by the packages located in the folder 'pkg'. The four notebooks *Setup*, *Execution*, *Analysis* and *Extension* contain all the logic and tools needed to create, execute and analyze simulations and their results. Variations of *Execution* and *Analysis*, such as *02b - Execution Handler*, work in addition to the main file. The data created and used by these notebooks are stored in subfolders of the folder 'export'. The folder 'mol' contains '*.mol'-files, which hold the atom positions and other information about the molecules used in the simulations. In the folder 'pkg', various Mathematica packages are stored.

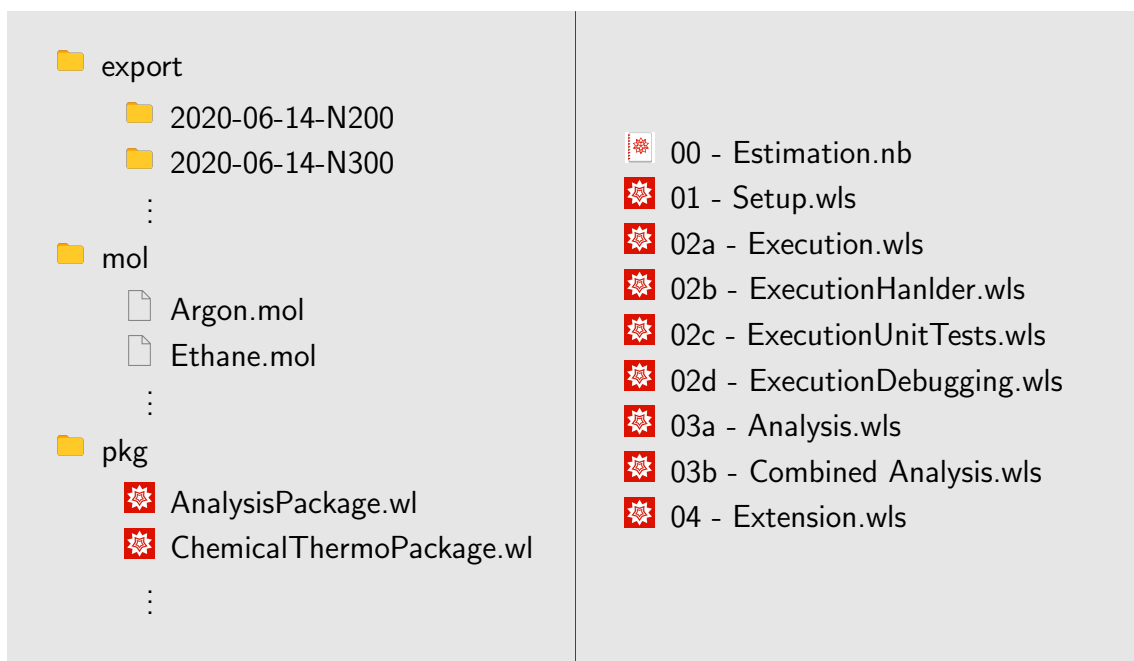


Figure 5.19.: Basic Folder Structure

5.7.2. Code Style

To keep a program user-friendly and easily readable, a certain layout for the code structure needs to be used. One of the easiest ways to implement new features is the use of `Switch` and custom parameters.

The *Setup* notebook uses many switch-functions to select one of multiple implemented procedures. To keep things simple and short it is advised to define functions for automated calculations which are executed in the switch-function if the procedure is chosen. In case a procedure needs additional input from the user it should be added at the bottom of the switch-options and if possible only consist of the necessary user input and an automated function. Indentations for switch-functions should be set as shown in Listing 5.2.

When functions with many arguments or modules with many internal variables are implemented, it is advised to group and label them according to their use or designated feature, as shown in Listing 5.3.

Additional features involving the *Execution* notebook can be easily activated through a custom parameter. An early approach of using `If`-functions to check whether a feature should

Listing 5.2: Switch statements in Mathematica

```

1 Switch["equalVolume"
2
3     ,"vapourDoubleVolumeLiquid",
4         NTvapourDoubleVolumeLiquid[];
5
6     ,"equalMolecule",
7         NTequalMolecule[];
8
9     ,"equalVolume",
10        NTequalVolume[];
11
12     ,"explicit",
13         nV = 100;      (*Molecules in the Vapor-Box*)
14         nL = ntot-nV; (*Molecules in the Liquid-Box*)
15 ];

```

Listing 5.3: Function Header in Mathematica

```

1 GetInteractionIndices = Compile[
2     {
3         (* arguments *)
4         {nBox,_Integer},
5         {refIndex,_Integer}
6     },
7
8     Module[
9         {
10            (* internal variables *)
11        },
12        ... your code ...
13        Return[0];
14    ]
15 ];

```

be executed proofed to be wearing for regular simulations. Therefore, it is advised to keep as much as possible outside of the regular code. An easier approach to minimize the additional time consumption is to use specific functions which are only declared if the custom parameter is defined such as `AdjustSpeciesProbability`.

When dealing with switch-functions or custom parameters, one can easily lose sight of how many and which features are implemented. Therefore, it is advised to use items and text cells above the code cell to describe all possible settings for the switch or custom parameter.

When implementing an If-function, it can become unclear which part is executed in case of True or False. Similar to that, in case Do-functions are implemented, it can become unclear which iterator belongs to which function. Listing 5.4 shows a way how those functions can

be designed clearly.

Listing 5.4: If- and Do-function structure in Mathematica

```

1 If[*condition*,
2   (*True*)
3   ... your code if true ..
4   ,(*ELSE*)
5   ... your code if false...
6 ];
7
8 Do[
9   ... your code ...
10  ,{*iterator*}
11 ];

```

5.7.3. Documentation

Setup A good description of the installation of \LaTeX can be found on the info page of TU Graz. In principle, it consists of the following two steps:

- **Compiler**

First a \LaTeX compiler needs to be installed. A commonly used one is MiKTeX .

- **Editor**

Afterwards a \LaTeX file editor software is needed. A commonly used one is Texmaker.

After Texmaker has successfully been installed, it is recommended to apply the settings shown in Figure 5.20. Thereby, the output files are created in a separate subdirectory and are not mixed with the source files. This mainly involves some adjustments to the build commands:

- activate Use a "build" subdirectory for output files.
- Bib(La)Tex `bibtex build/%`
- Makeindex `makeindex build/%.idx`

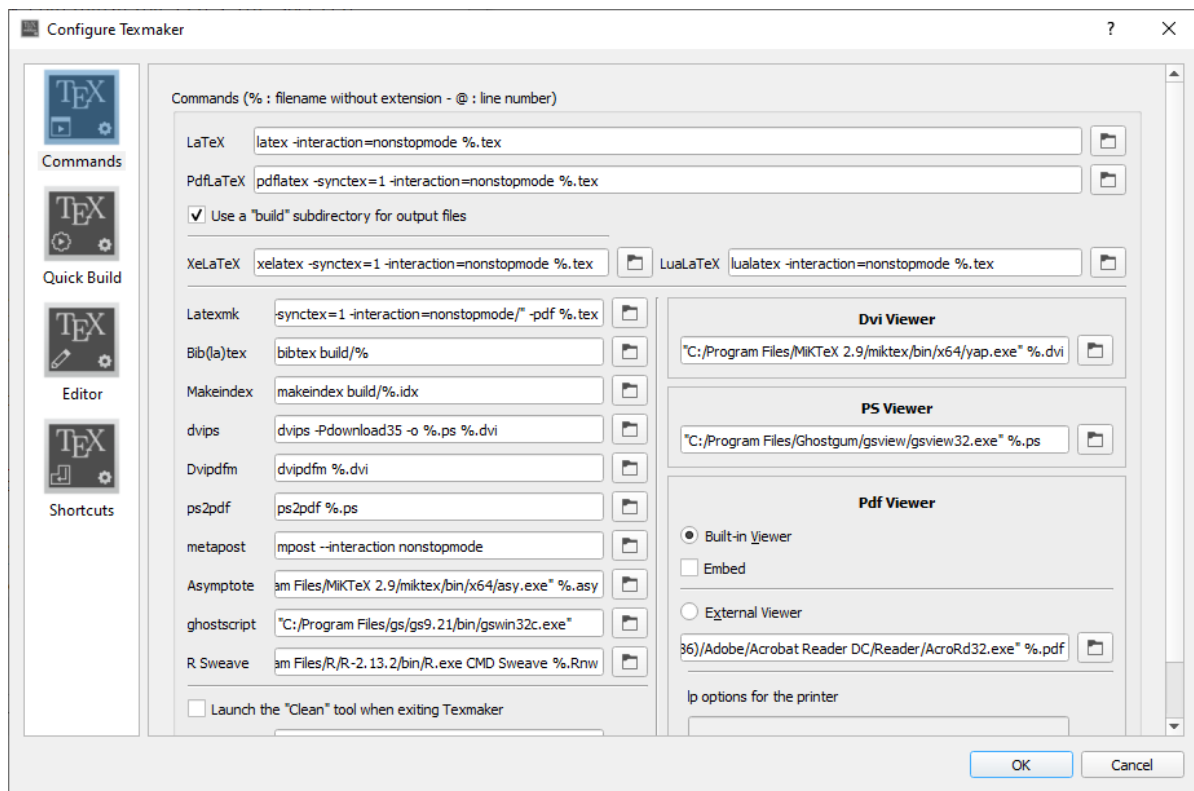


Figure 5.20.: Configuring TexMaker

Structure The documentation is structured in several sub-files for each chapter which can be compiled individually. Thereby, a lot of time is saved if not the whole project is compiled during writing.

It is recommended to avoid numbering the \LaTeX source files since it mainly generates work in case an additional file is added. A prefix is used instead ("s" for subfiles, "f" for figures, "t" for tables, etc.) and a clear name, ideally within a reasonably named, short folder. Otherwise, the included paths get infeasibly long and might even exceed the limit of the \LaTeX compiler. For example, in the present report, all include files of the *Theoretical Background* section are stored in the folder 'th' while imports of the *Implementation* section are stored in the folder 'im'.

Finally, the command `biblio` needs to be added at the end of every \LaTeX subfile to include the bibliography when individual chapters are compiled.

Screen shots Throughout this documentation, screenshots are used to illustrate some aspects of the software package. For consistent screenshots, 'ShareX' can be used, as illustrated in Figure 5.21. Before the screenshots are taken, the following command needs to be executed in Mathematica to hide the top bar.

```
SetOptions[InputNotebook[], "DockedCells" -> None]
```

Otherwise, the top bar would use a lot of space and the screenshots would become excessively large. Further, the screenshots shall be included using `width=0.9columnwidth` for good readability of texts.

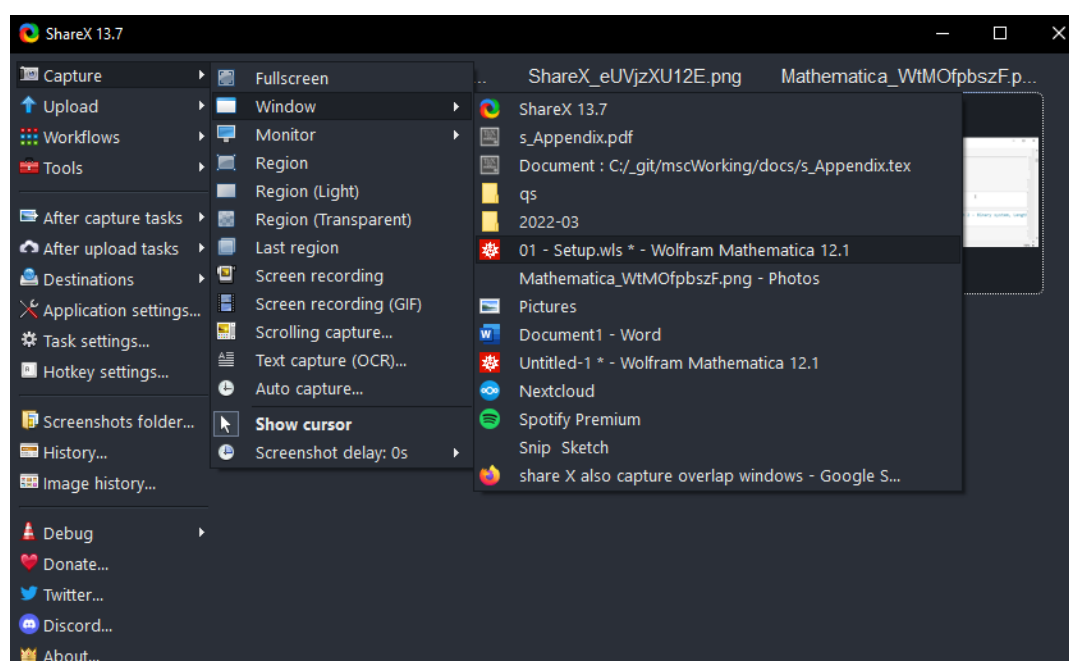


Figure 5.21.: How to use ShareX to make proper screenshots from any application.

Note: Ironically, ShareX cannot capture itself properly with the overlay windows. Thus, a regular screenshot had to be taken from ShareX.

6. Parameterization

In the following section, parameterization guidelines and exemplary results are given. Currently, Lennard-Jones (LJ) fluids, binary mixtures of LJ fluids and simple organic alkanes have been thoroughly investigated.

6.1. Pure Lennard-Jones Fluids ¹

In the following section, the main findings of the bachelor thesis of Andreas Schwarz are summarized.[39]

6.1.1. Guidelines

1. Initial Volume

The initial molar volume can be estimated using a suitable EoS like the EoS for LJ fluids by Thol.[46] For temperatures in the range of $1.00 < T^* < 1.30$ it is recommended to setup the boxes with equal volumes and an equal number of particles. For low temperatures ($T^* = 0.75$) a volume ratio with twice as much volume in the vapor box than the liquid box is necessary to ensure a safe phase split.

2. Number of Moves per Cycle

The initial trial move composition should be chosen depending on the total number of particles in the system, n_{tot} , and the LJ-reduced temperature T^* .

¹Andreas Schwarz contributed to this section.

Trial Move	T^*					Comment
	1.30	1.25	1.15	1.00	0.75	
Translations			n_{tot}			(number of total particles)
Volume Changes			1			
Insertions	0.04	0.04	0.1	0.2	1.5	$(n_{\text{Ins}} = X \cdot n_{\text{tot}})$
Widom Insertions			$n_{\text{tot}}/4$			

3. Number of Cycles

For the amount of cycles performed it is recommended to first execute simulations with more production cycles. If the results show a constant development of the system properties, a reduction of cycles is possible to save time. However, in order to keep a meaningful simulation course, a minimum amount of 5,000 production cycles is recommended.

warm-up cycles: 2,000
 equilibration cycles: 10,000
 production cycles: 5,000 - 10,000

4. The **cut-off radius** should be set to half the box length.

5. Custom Parameters

The following additional simulation details are recommended. They can be enabled in the simulation by setting the custom parameters as listed below:

- **Trial Move Composition**

It is recommended to auto-calibrate the number of insertions to yield approximately one successful insertion per cycle. No adaptation of translations and volume changes is necessary.

- "trialMoveAdoption" → "acceptedPerCycle"
- "trialMoveAdoptionSetpoint" → 1

- **Trial Move Limits**

The trial move limits should be adjusted after at least 10 trial moves of the respective type have been executed.

- "trialMoveLimitsAdoptionMode" → "numberMoves"
- "trialMoveAdoptionLimit" → 10

- **Trial Move Arrangement**

The arrangement of the trial moves should be random.

- "sampleType" → "random"

Parameters that are not discussed should not be executed for the first time.

To ensure statistical accuracy, enough particles must be present in one phase at the end of the simulation. At equilibrium, a rule of thumb is that at least 20% of the particles are in the vapor phase and at least 200 molecules are in the liquid phase. If those requirements are not met, the equilibrium can be shifted in further simulations with two approaches.

1. Change the total volume of the simulated equilibrium while setting up an extension with the volume change tool.
2. Set up a new simulation using the *Setup* notebook with the resulting densities from the finished simulation as the initial molar volumes. Set up the total number of molecules so that the vapor box has twice the side length of the liquid box using the 'vapour-DoubleVolumeLiquid' option.

6.1.2. Equation of State for pure LJ Fluids

The EoS for LJ fluids by Thol is used to evaluate the results of pure LJ fluids. It is derived from the reduced Helmholtz free energy and consists of two parts: An exact ideal-gas contribution and a residual contribution, consisting of 23 terms which are fitted to literature data and simulation results.[46]

Its validity ranges from $0.661 < T^* < 9$ and $P^* < 65$. That corresponds to $0.5 < T/T_c < 7$ and $P/P_c < 500$. Its uncertainty is estimated to be 1.8% in vapor pressure and saturated vapor density, and 0.15%–0.3% in saturated liquid density. Its uncertainty is 0.15%–0.3% in the supercritical region. The uncertainty in pressure amounts to 2% in the critical region. The residual internal energy is reproduced within 1% in the gaseous region, 0.1% in the liquid region, and 1% in the critical and supercritical region.[46]

6.1.3. Results

Figure 6.1 shows simulation results and standard deviation of a pure LJ fluid (argon) using the previously described guidelines. The LJ parameters used are $\sigma = 3.3952 \text{ \AA}$ and $\epsilon/k_B = 116.79 \text{ K}$. [44] Table 6.1 contains the numerical values of the diagram as well as the relative error of the ensemble averages to the EoS at each simulation point.

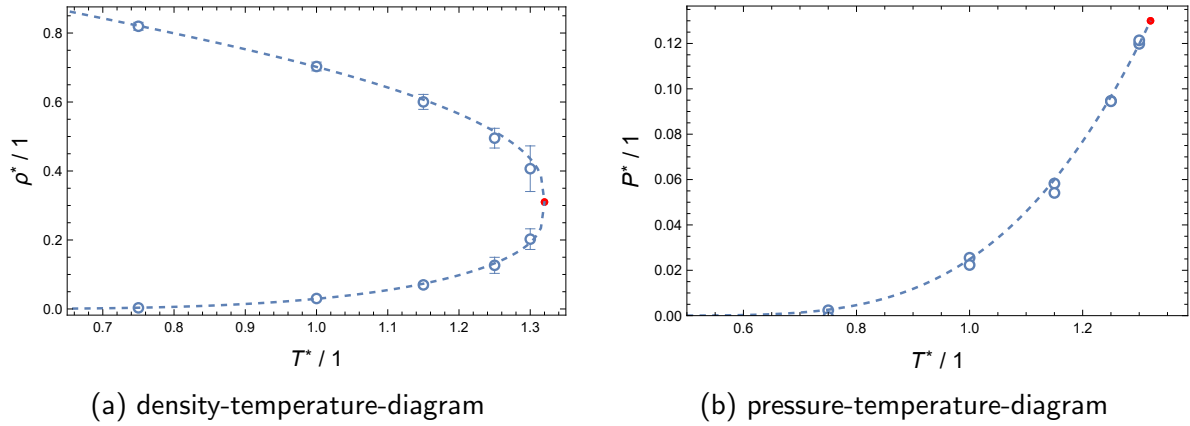


Figure 6.1.: Simulation results and standard deviation of argon using the guidelines for pure LJ fluids. The units are based on the LJ norm. (----) Thol EoS; • critical point

Table 6.1.: Ensemble averages and their relative error to the Thol EoS [46] from Figure 6.1

T^*	Unit	Vapor Phase			Liquid Phase		
		Sim	EoS	$\Delta_{\text{rel}} / \%$	Sim	EoS	$\Delta_{\text{rel}} / \%$
0.75	$\rho^* / 1$	0.00324	0.00362	10.6	0.820	0.821	0.188
	$P^* / 1$	0.00236	0.00263	10.3	0.00415	0.00263	57.9
1.00	$\rho^* / 1$	0.0303	0.0295	2.98	0.703	0.702	0.161
	$P^* / 1$	0.0255	0.0249	2.51	0.0223	0.0249	10.2
1.15	$\rho^* / 1$	0.0698	0.0733	4.75	0.600	0.607	1.07
	$P^* / 1$	0.0582	0.0600	2.91	0.0579	0.0600	9.79
1.25	$\rho^* / 1$	0.126	0.132	4.48	0.495	0.514	3.72
	$P^* / 1$	0.0947	0.0966	2.03	0.0945	0.0966	2.19
1.30	$\rho^* / 1$	0.202	0.192	5.26	0.407	0.436	6.68
	$P^* / 1$	0.120	0.120	0.07	0.121	0.120	1.37

6.2. Binary Lennard-Jones Fluids

In the following section, the main findings of the bachelor thesis of Michael Haring are summarized.[12]

6.2.1. Guidelines

The following suggestions are for simulations of binary LJ fluids with a total of 500 particles in the system. Simulations for mixtures ranging from $\sigma_{22}/\sigma_{11} = 1...0.5$ and $\epsilon_{22}/\epsilon_{11} = 0.75...0.5$ have been studied at LJ-reduced temperatures $T^* = 1.0$ and 0.75 . Only the Lorentz-Berthelot combining rules have been studied. Different amounts of total particles in the system will require an adjustment of the number of cycles and moves per cycle. Simulations at different reduced temperatures require an interpolation/extrapolation of recommended settings. The conventional GEMC insertion is used to facilitate particle transfers. Custom parameters need to be adjusted if the CFC method is used for insertions.

1. The **initial molar volume/number density** is set using the section '*Lennard-Jones reduced units*' of '*Temperature, Pressure, Molar Volume*'. The values for ρ^* in the liquid and vapor phase depend on the reduced Temperature of the system and are stated in Table 6.2.[31] The different initial densities of each box allow a safe phase split and prevent early emptying of a box.
2. The **molar fraction** should be set equal for both boxes ($x_{1L} = x_{1V}$).[31] The value for the molar fraction should be chosen to be in the center of the two-phase area in the Pxy-diagram to assure enough particles in both phases. The t-PR-LJ EoS can be used to estimate this value, which is implemented in the *Estimation* notebook.[15]
3. **Number of Molecules**
The molecules should initially be distributed equally over both boxes as suggested by Panagiotopoulos.[28] Choose the 'equalMolecule' option when determining the initial number of molecules for each box.
4. **Number of Cycles**
The number of production cycles can be increased in case the simulation does not give satisfactory results. An extension can be used for this purpose as well.

warm-up cycles: 2,000
equilibration cycles: 6,000
production cycles: 15,000 - 30,000

5. Number of Moves per Cycle

The recommended numbers of trial moves per cycle depend on the reduced temperature of the system and are stated in Table 6.2. The number of rotations was not studied in this work. The number of insertions is just a starting value which will be adjusted in the equilibration phase. The values for the starting values were determined by averaging the results of the studied systems. 125 Widom Insertions were sufficient for all studied systems since regular insertions acted as Widom insertions as well. More Widom insertions are necessary in case a different insertion method is used or low amounts of insertions take place in the simulation.

6. Trial Move Limits

- The initial **trial move limits** should be set as it is recommended by Thaler [45] with a maximum translation distance of 2 Å and a maximum volume change of 1% of the total volume in the system.
- The **cut-off radius** should be set to half the box length for each box, which can be done by choosing the 'boxspecific' option when calculating `rCutVec`.
- The **overlap radius** should be set as recommended by Panagiotopoulos [28], which can be calculated using the 'automatic' option when calculating `r0v1Vec` as discussed in section 5.2.4.

7. Custom Parameters

The following additional simulation details are recommended. They can be enabled in the simulation by setting the custom parameters as listed below:

- **Trial Move Composition**

It is recommended to adjust the number of insertions to yield approximately one successful transfer per cycle. The maximum number of insertions might need to be increased for strongly asymmetric mixtures at low temperatures. No adaptation of translations and volume changes is necessary.

- "trialMoveAdoption" → "acceptedPerCycle"
- "trialMoveAdoptionSetpoint" → 1

- **Trial Move Limits**

The adjustment of the trial move limits should be done as recommended by Thaler [45] with an adjustment after 10 trial moves of its type have been attempted.

- "trialMoveLimitMode" → "numberMoves"
- "trialMoveAdoptionLimit" → 10

- **Insertion Species Choice Probability**

It is recommended to adjust the probability for species choice in insertion trial moves to achieve equal amounts of successful trial moves per species as suggested by Harismiadis.[16] This can be done with the methods described in section 4.5.3. The following settings worked well for all studied systems:

- "speciesSwapProbabilityMode" → "auto"
- "speciesSwapWeightIncrease" → "sharedependent"
- "speciesSwapCyclesAdjustment" → 50

- **Volume Change Trial Move Limits**

For mixture simulations in the NPT ensemble it is recommended to use two separate trial move limits for the volume changes.[16]

- "boxSpecificVolumeLimits" → True

- **Equilibrium Phase Adjustment Averages**

Since auto-calibrated parameters can fluctuate near the end on the equilibration phase it is recommended to use the average of the last 1,000 documented states in the equilibration phase.

- "eqAdjAvrCycles" → 1000

- **Widom Insertion Overlap Radius**

It is recommended not to use an overlap radius for Widom insertions, as mentioned in Section 5.3.2.

- "widomrOvl" → 0

Table 6.2 summarizes the recommendation for the trial move composition and the initial reduced densities of binary mixture simulations.

Table 6.2.: Initial settings for binary mixtures of LJ fluids at different LJ reduced temperatures

		T^*		Comment
Setting		1.00	0.75	
Initial densities	ρ_V^*	0.2	0.025	
	ρ_L^*	0.5	0.8	
Trial moves	Translations	500		equal to total number of particles
	Rotations	-		
	Volume changes	1	3	
	Insertions	250	1,000	
	Widom Insertions	125		1/4 of total number of particles

6.2.2. Equation of State for binary Mixtures of LJ Fluids

The volume-translated Peng-Robinson equation of state for LJ Fluids (t-PR-LJ EoS) was introduced by Harismiadis in 1994.[15] It is a modification of the cubic equation of state. The parameters for the EoS are calculated from the critical point which depends on the LJ parameters. The pure-component parameters are combined in accordance to the Lorenz-Berthelot combining rules. While the molar fractions can be well described it is known that it cannot accurately predict the liquid density of asymmetric mixtures where $\epsilon_{22}/\epsilon_{11}$ is smaller than 0.75.[15]

The functions of the t-PR-LJ EoS are part of the *ChemicalThermo* package. The pressure explicit EoS and the fugacity are used to calculate the vapor-liquid equilibrium. The function `bubblePressureVTPRLJ` calculates the bubble pressure (P_{bubble}) at a given temperature and molar fraction of the liquid phase. To calculate the bubble pressure, a good estimate for the pressure and vapor molar fraction is necessary to ensure convergence in the iterative calculation. It is assumed that one of the two components is below the critical temperature allowing for an initial calculation at $x_{1,\text{init}} = 1$ or $x_{1,\text{init}} = 0$ using the EoS for LJ fluids by

Thol [46] to estimate the pressure (P_{init}). The molar fraction is then adjusted according to Equation 6.2.1 until the bubble pressure is equal or greater than the specified pressure (P_{set}). The results for the pressure and vapor molar fraction of the previous iteration are used as estimates of the next calculation.

$$x_{\text{new}} = x_{\text{old}} + dx \cdot \frac{P_{\text{bubble}} - P_{\text{set}}}{P_{\text{init}} - P_{\text{set}}}, \quad (6.2.1)$$
$$\text{with } dx = \begin{cases} 0.05 & \text{if } x_{1,\text{init}} = 0 \\ -0.05 & \text{if } x_{1,\text{init}} = 1 \end{cases}$$

6.2.3. Results

Figure 6.2 shows the simulation results of the three binary mixtures of LJ fluids studied at $T^* = 1.0$ and $T^* = 0.75$ with the guidelines from the previous section. The LJ parameters of the first component are set to unity. The results are compared with estimations from the t-PR-LJ EoS.[15]

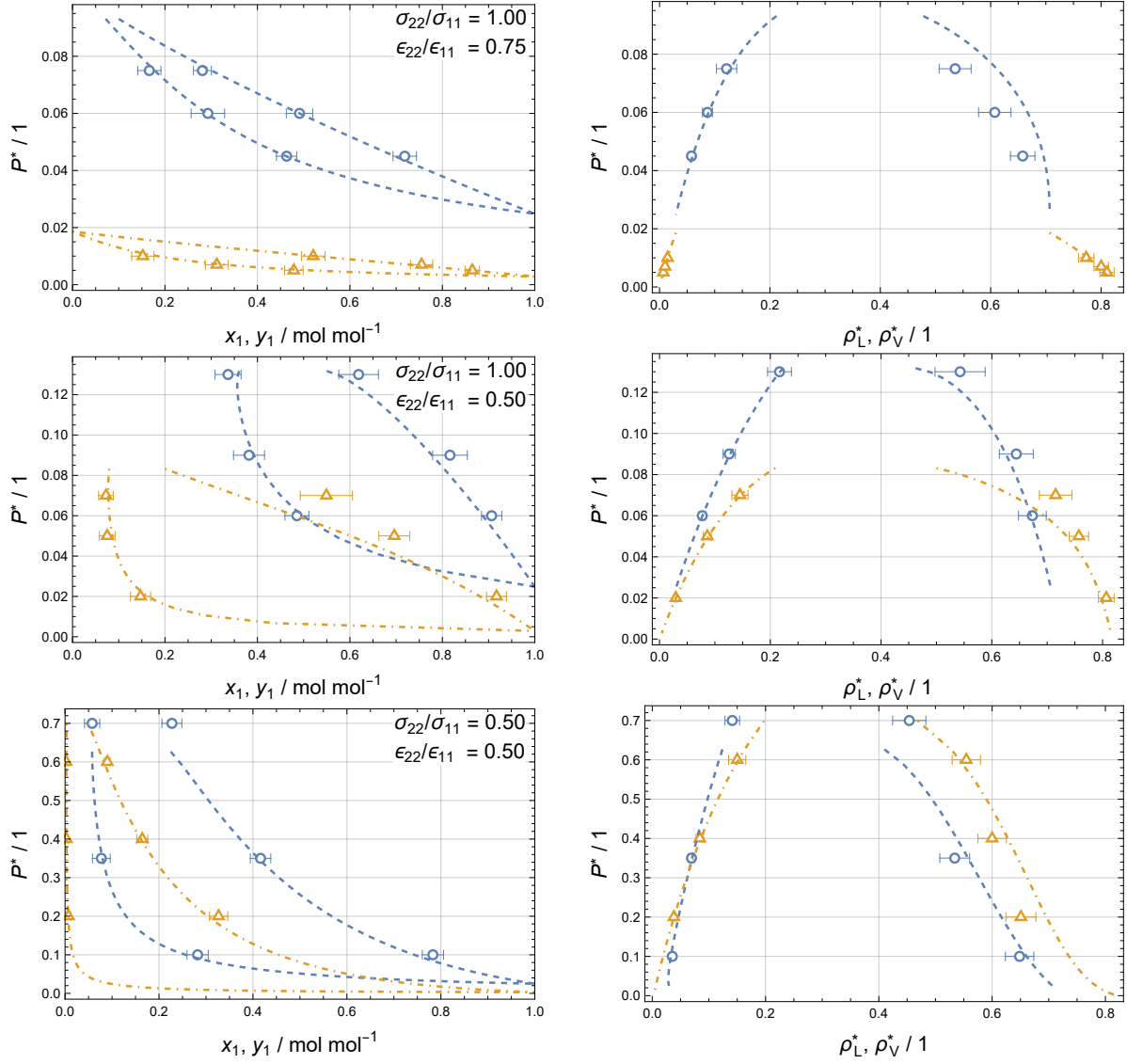


Figure 6.2.: Simulation results of binary mixtures of LJ fluids in the GEMC NPT ensemble. Molar fraction (left) and LJ reduced densities (right) of the liquid and vapor phase at different LJ reduced pressures. The LJ parameters of the first component are set to unity. (----) t-PR-LJ EoS at $T^* = 1.0$; (-·-·-) t-PR-LJ EoS at $T^* = 0.75$; \circ simulation results at $T^* = 1.0$; \triangle simulation results at $T^* = 0.75$

6.3. Rigid Molecules²

In the following section, the main findings of the bachelor thesis of Theresa Plesch [35] are summarized.

6.3.1. Guidelines

The following suggestions are for simulations involving components consisting of multiple interaction sites, complementary to the recommendations for pure components described in section 6.1.1. The recommended settings were derived in test series simulating ethane modeled as a rigid molecule with two centers, and binary mixture simulations including methane as a LJ fluid. All simulations were conducted with 500 particles in the system.

1. The **initial molar volume** should be chosen according to the prediction of the Soave-Redlich-Kwong equation of state if critical point data is available. If the prediction leads to an insufficient number of molecules in a box, extensions need to be conducted as described in section 6.1.1.

2. **Number of Moves per Cycle**

The number of translations and rotations should be equal ($n_{\text{trans}} = n_{\text{rot}}$). The total number of translations and rotations depends on the number of particles and whether a pure component or binary mixture is simulated.

- **Pure Component**

The number of translations and rotations should be 1,000 each.

$$n_{\text{trans}} = n_{\text{rot}} = 1,000 (= 2 \cdot n_{\text{tot}})$$

- **Binary Mixture**

The number of translations and rotations should be 2,000 each.

$$n_{\text{trans}} = n_{\text{rot}} = 2,000 (= 4 \cdot n_{\text{tot}})$$

²Theresa Plesch contributed to this section.

6.3.2. Results

Figure 6.3 shows the simulation results and standard deviations of ethane as a rigid molecule with two centers with the guidelines from the previous section. The LJ parameters of ethane are $\sigma = 3.75 \text{ \AA}$ and $\epsilon/k_B = 98 \text{ K}$ per interaction site and a bond length of $l = 1.53 \text{ \AA}$. [7] The results are compared to estimations from the equation of state by Friend. [10]

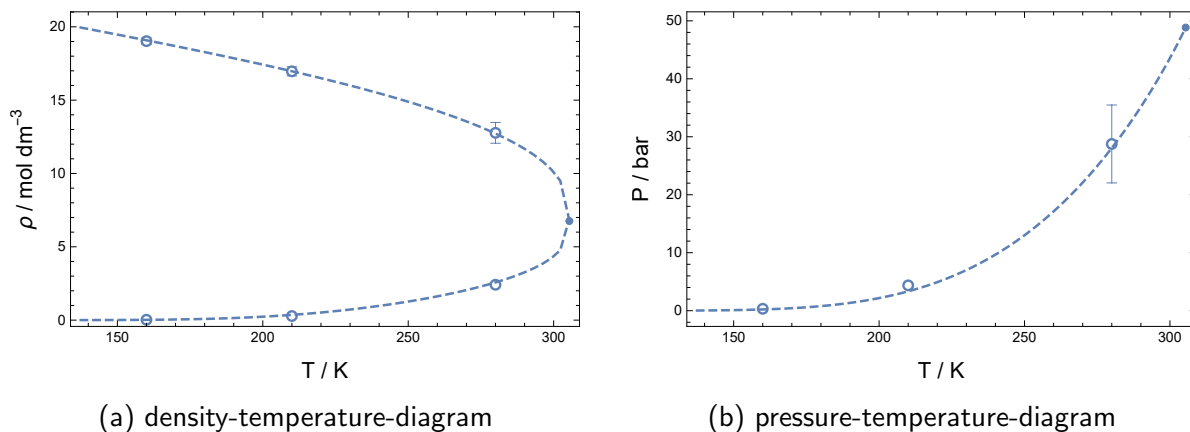


Figure 6.3.: Simulation results and standard deviations of ethane using the guidelines for rigid molecules.

(----) Friend 1991 EoS [10], \circ simulation results, \bullet critical point

Figure 6.4 shows the simulation results and standard deviation of the binary mixture ethane - methane. Methane is modeled as a LJ fluid with $\sigma = 3.73 \text{ \AA}$ and $\epsilon/k_B = 148 \text{ K}$. [7] The results are compared to estimations from the SRK EoS with a binary parameter of $k_{ij} = -0.01$. [22] The binary parameters was fitted to experimental data in the temperature range from $T = 144 \text{ K}$ to $T = 283 \text{ K}$.

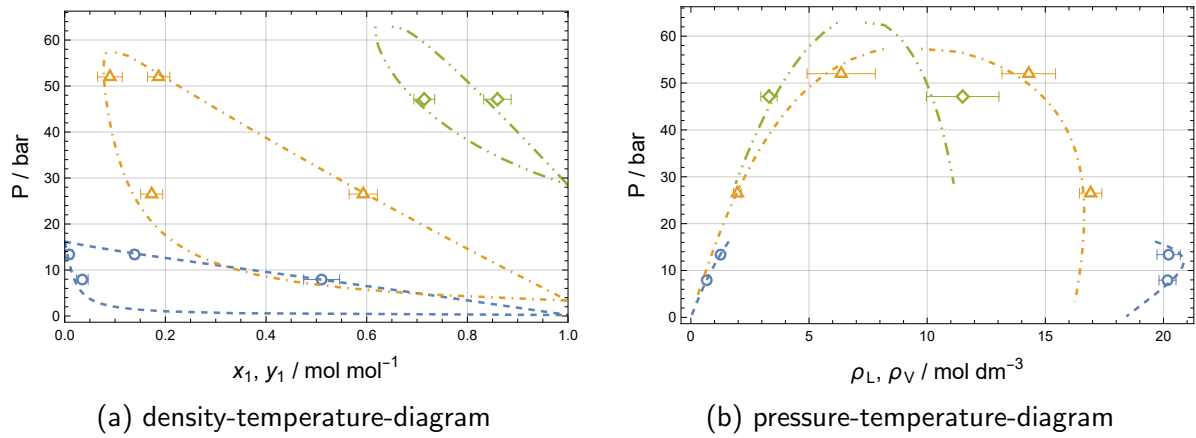


Figure 6.4.: Simulation results and standard deviations of the binary mixture ethane - methane using the guidelines for rigid molecules. The lines indicate the estimation from the SRK EoS with $k_{ij} = -0.01$. [22]
 SRK EoS at (----) $T = 160$ K, (-·-·-·) $T = 210$ K, (·-·-·) $T = 260$ K;
 simulation results at \circ $T = 160$ K, \triangle $T = 210$ K and \diamond $T = 260$ K

6.4. Continuous Fractional Component (CFC) Method

In the following section, the main findings of the master thesis of Niklas Mayr [24] are summarized. The GECFC method is an attractive choice when simulating pure LJ fluids at low temperatures ($T^* < 0.8$) as well as for asymmetric binary mixtures ($\sigma_{22}/\sigma_{11} < 1.0$; $\epsilon_{22}/\epsilon_{11} < 0.75$). The GECFC method increases the acceptance rate of insertions, reducing the effort where the conventional GEMC method would require a high number of attempted insertions. Because of the increased acceptance rate, it is not necessary to bias the species selection for insertions when simulating asymmetric binary mixtures.

The GECFC method is enabled by setting the following custom parameter:

- "particleInsertionMode" \rightarrow "CfcVlugt"

6.4.1. Guidelines

Based on the guidelines of sections 6.1.1 and 6.2.1, the following recommendations were deduced when using the CFC method for insertions:

Trial Move Composition In contrast to the recommendations for the conventional GEMC method, when using the GECFC method it is recommended to execute a constant amount of trial moves per cycle equal to the number of particles in the system. The trial moves should consist of 49.5% displacements, 1% volume changes and 49.5% GECFC trial moves. The CFC trial moves should further consist of 50% *Lambda changes*, 25% *CFC Exchanges* and 25% *CFC Swaps* which are described in section 4.4.3. For a discrete number of trial moves in a simulation with 500 particles, this would correspond to 248 displacements, 5 volume changes, and 247 GECFC trial moves. Additional trial moves that don't alter the system, like the Widom insertion, need to be executed in addition to this recommendation.

Number of Cycles For the conventional GEMC method, Panagiotopoulos [34] recommends that the number of successful insertions in the equilibration phase should be at least three to five times the number of molecules in the system. In case a simulation of 500 particles is executed, 500 moves are executed per cycle of which 49.5 % are CFC moves. Further, 25 %

of the CFC moves are *CFC Swap* moves with an average acceptance rate of 5 %. The number of equilibration cycles should therefore be $[3 - 5] \cdot 0.495^{-1} \cdot 0.25^{-1} \cdot 0.05^{-1} \cong [500 - 800]$.

In practice, however, it is recommended to conduct at least 2-3 times as many cycles compared to the conventional GEMC method because *CFC Swaps* are less selective. Indeed, the acceptance of conventional GEMC insertions is directly related to the chemical potential and, thus, the GEMC method almost solely accepts insertions towards equilibrium. By contrast, the energy differences of the *CFC Swaps*, especially at a low coupling factor ($\lambda \rightarrow 0$), are small, with moves towards equilibrium being accepted more often.

This can be clearly seen in Figure 6.5 where the course of the number of molecules and the molar volume using GECFC and the conventional GEMC method are compared. Mind that in each MC cycle of the GEMC method substantially more trial moves were attempted compared to the GECFC method.

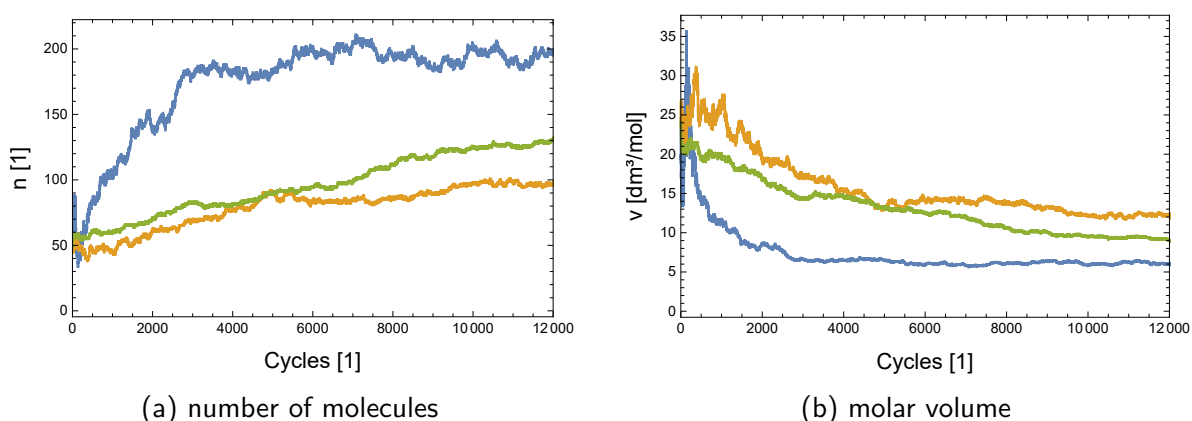


Figure 6.5.: Comparison of the simulation course of argon as a LJ fluid at $T^* = 0.75$.
 — GEMC algorithm, — unbiased CFC Pou., — biased CFC Pou.

Overlap Radius Since interactions of the fractional component are scaled, a different overlap radius than the one recommended by Panagiotopoulos [28] must be set. An overlap radius of 1 Å was used, as recommended in literature.[41, 36]

Biasing Function It is recommended to use a biasing function as stated in section 4.4.2. The following settings can be adjusted through custom parameters. However, note that the recommended values are all set by default once the GECFC method has been enabled.

- **Start value of the biasing increment**

If the start value was chosen too large, e.g. $\ln(v) = 1$, the simulation course was unstable because the effect of the biasing function was too strong, compared to an average energy change of a trial move ΔU . However, if the start value was too small, e.g. 0.001, it took too long until the biasing function had been adjusted to appropriate values yielding a flat λ distribution. A start value of 0.01 yielded plausible results.

- **Number of cycles until the biasing increment is evaluated**

If the biasing increment $\ln(v)$ was decreased too quickly, the biasing function was unstable. At the same time, the adjustment was infeasible if the biasing increment was evaluated too slowly. In practice, an evaluation every 200 GEMC cycles showed reasonable results.

- **Calculation of the biasing function**

If discrete bins were used, the recorded λ states showed an uneven, sawtooth-like course. Thus, it is necessary to use a continuous interpolation between the mesh points.

- **Interpolation order & number of mesh points**

Given that an interpolation is needed, it was investigated how this interpolation shall be implemented. First, second and third order polynomials and different numbers of mesh points were investigated. If too few bins were used, the biasing function could not represent the steep course of the distribution at the boundaries $\lambda \rightarrow \{0, 1\}$. At the same time, too few states per bin were collected if too many bins were used. Thus, it is recommended to use 20 bins spanning the $[0, 1]$ λ space if the biasing function is evaluated every 200 cycles. Concerning the interpolation order, the higher order polynomials seemed to yield a slightly more continuous course of the biasing function, yet they often diverged at the boundary $\lambda \rightarrow \{0, 1\}$. Thus it is recommended to use a first order interpolation.

- **Box biasing**

The investigated approach to adjust the biasing function in a way that the fractionals are equally likely to be in both boxes yielded implausible results. Indeed, negative pressures were recorded due to a significant contribution of the virial. A possible explanation might be that, with the box biasing, fractionals were exchanged with full

molecules at implausible positions. Thus, further investigations are needed in this area.

- **Calculation of the chemical potential**

If the λ states were unbiased, the CFC method by Poursaeidesfahani [36] reproduced results of the Widom method qualitatively, however they differed significantly quantitatively. If the biasing function was used, neither qualitative nor quantitative agreement was achieved. A few possible aspects of future research could be the parametrization of the biasing function, a proper algorithm to bias the box distribution or simply a significantly longer simulation period.

Additional Details for Binary Mixtures

The algorithm was further investigated for the binary mixtures presented in Table 6.3.

Table 6.3.: Initial conditions for simulations of binary LJ mixtures (M1, M2, M3). σ_{22}/σ_{11} and $\epsilon_{22}/\epsilon_{11}$ are the ratios of the size and energy parameters of the potential wells of the two components, respectively. T^* is the LJ reduced temperature and P^* is the LJ reduced pressure.

Index	σ_{22}/σ_{11}	$\epsilon_{22}/\epsilon_{11}$	T^*	P_1^*	P_2^*	P_3^*
M1	1.00	0.75	0.75	0.005	0.007	0.010
			1.00	0.045	0.060	0.075
M2	1.00	0.50	0.75	0.020	0.050	0.070
			1.00	0.060	0.090	0.130
M3	0.50	0.50	0.75	0.200	0.400	0.600
			1.00	0.100	0.350	0.700

Acceptance Rate of Molecule Transfers In Figure 6.6, the acceptance rate of the bigger component 1 and the smaller component 2 are stated for the three binary mixtures. Mind that the scale of the y-axis is logarithmic. While the acceptance rate of the bigger component 1 is in the order of 0.1 % for the symmetric mixtures M1 and M2 and in the order of 0.01 % for the asymmetric mixture, the Poursaeidesfahani method [36] shows a drastically higher

acceptance rate in the order of 5 % for the insertion trial move. Mind, however, that this method consists of three types of moves - of which only one transfers the fractionals between the boxes.

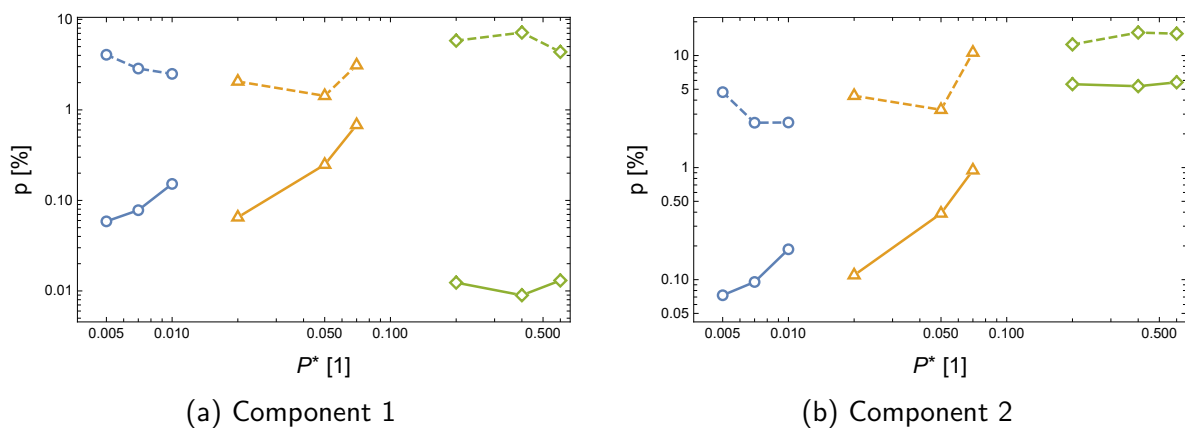


Figure 6.6.: Comparison of the acceptance rate of the bigger component 1 and the smaller component 2. Legend: — GEMC, ---- GECFC Pou. ; \circ M1, \triangle M2, \diamond M3

Species Choice Probability for Insertions Both components show a similar acceptance rate, as illustrated in Figure 6.6. When testing the algorithms for auto-calibrating the probability for species choice in insertions, results showed that the rate of the probabilities ranged around unity. Hence, it is not recommended to bias the species selection in GECFC trial moves for the investigated binary mixtures.

6.4.2. Results

In the following section, the investigated methods are compared. The GECFC method is currently implemented as suggested by Poursaeidesfahani et al. ('CFC Pou.'). The method is compared against the conventional GEMC algorithm, the improved insertion algorithm for binary mixtures by Panagiotopoulos ('Pan Swap', cf. section 4.3) and a previously suggested implementation by Shi et al. ('CFC Shi', see section 4.4).

Pure LJ Fluids In Figure 6.7, the coexisting densities and pressures are visualized. Clearly, all methods are in good qualitative and quantitative agreement. Specifically, the deviation

of the coexisting density from the Thol EoS is in the order of $0.03 \cdot \sigma^3$ for the liquid phase and $0.01 \cdot \sigma^3$ of the vapor phase. The pressure in the vapor phase deviates in the order of $3 \cdot 10^{-4} \sigma^3 / \epsilon$ and $5 \cdot 10^{-3} \sigma^3 / \epsilon$ in the liquid phase.

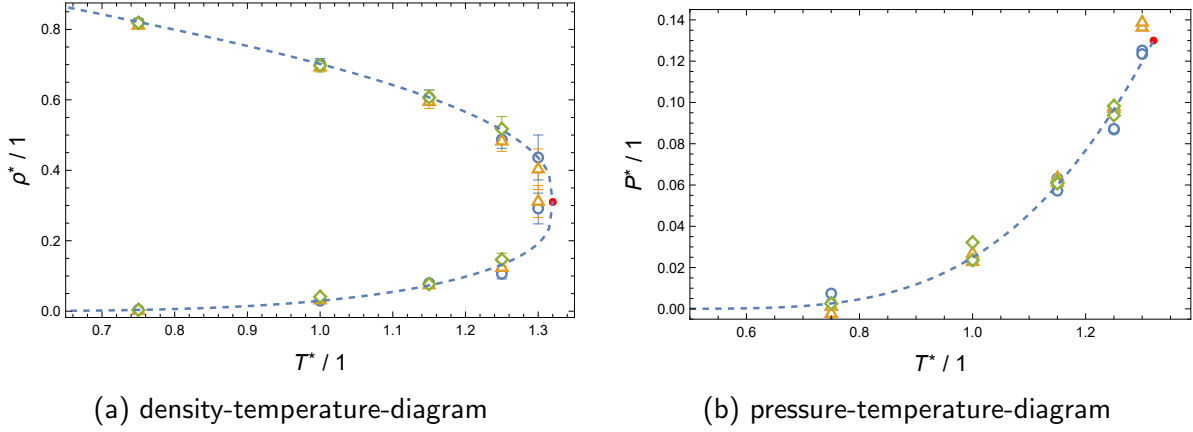


Figure 6.7.: Comparison of the investigated methods for argon as a LJ fluid covering the whole temperature range. Legend: \circ GEMC, \triangle GECFC Shi, \diamond GECFC Pou.

Binary LJ Mixtures In Figure 6.8, exemplary results of the conducted simulations are stated.

For the mixture M1 the results are in good quantitative agreement with the BWR EoS by Nicolas et al. [27]. A significant deviation from the t-PR-LJ EoS remains for liquid coexisting density. Still, the t-PR-LJ EoS provides a reasonable starting point since it describes the molar fractions well.

For the mixture M2, whose energy parameter ϵ deviates twice as much compared to M1, the different insertion methods are generally in good agreement. However, some outliers are observed, especially at higher pressures and for the lower temperature of $T^* = 0.75$. Still, the t-PR-LJ EoS predicts the molar fractions sufficiently well, and the guesses for the initial densities yield good results in a reasonable computation time.

For the most asymmetric mixture M3, whose components differ in both LJ parameters σ and ϵ the molar fractions are in good agreement. For the molar densities the different insertion methods are in agreement except for some outliers. The guess of the molar fractions using the t-PR-LJ EoS is sufficiently accurate.

In brief, all investigated insertion methods are in qualitative and quantitative agreement with respect to the average deviation from the BWR EoS: The vapor density, ρ_V^* , deviated in the order of $10^{-3}\sigma^3$, and the liquid density, ρ_L^* , deviated in the order of $10^{-2}\sigma^3$. The deviations of the molar fractions are in the order of 1-5 % for the simulated mixtures. The standard deviation of the number of molecules ranges between 10 and 30 for all algorithms and simulated states. This indicates that all algorithms are approximately similarly stable.

6.4.3. Conclusion

In the following section, the observations described in the previous section are summarized, and it is explained when each particular method should be used.

GEMC With the derived parameterization rules, simulations can be conducted conveniently. While the GEMC method is very reliable, its bottleneck is the low temperature region, where the acceptance rates of particle transfers, which equilibrate the chemical potential, are low. Thus, a lot of insertions have to be attempted in order to yield a sufficient number of successful exchanges.

Improved Insertion Algorithm for binary Mixtures The method by Panagiotopoulos [29] is a particularly attractive choice since it is very similar to the conventional GEMC method. Thereby, the derived parameterization rules can easily be used for this method. It is recommended to use this method for asymmetric mixtures at moderate temperatures. However, it cannot be used for pure components, and its computational performance deteriorates at low temperatures or dense phases.

CFC Method according to Shi et al. The trial moves of the first implementation of the GECFC method are rather complex and thus very slow since they involve a high computational effort. Thus, it has only been investigated for pure LJ components where it showed good agreement with the conventional GEMC method and the Thol EoS.

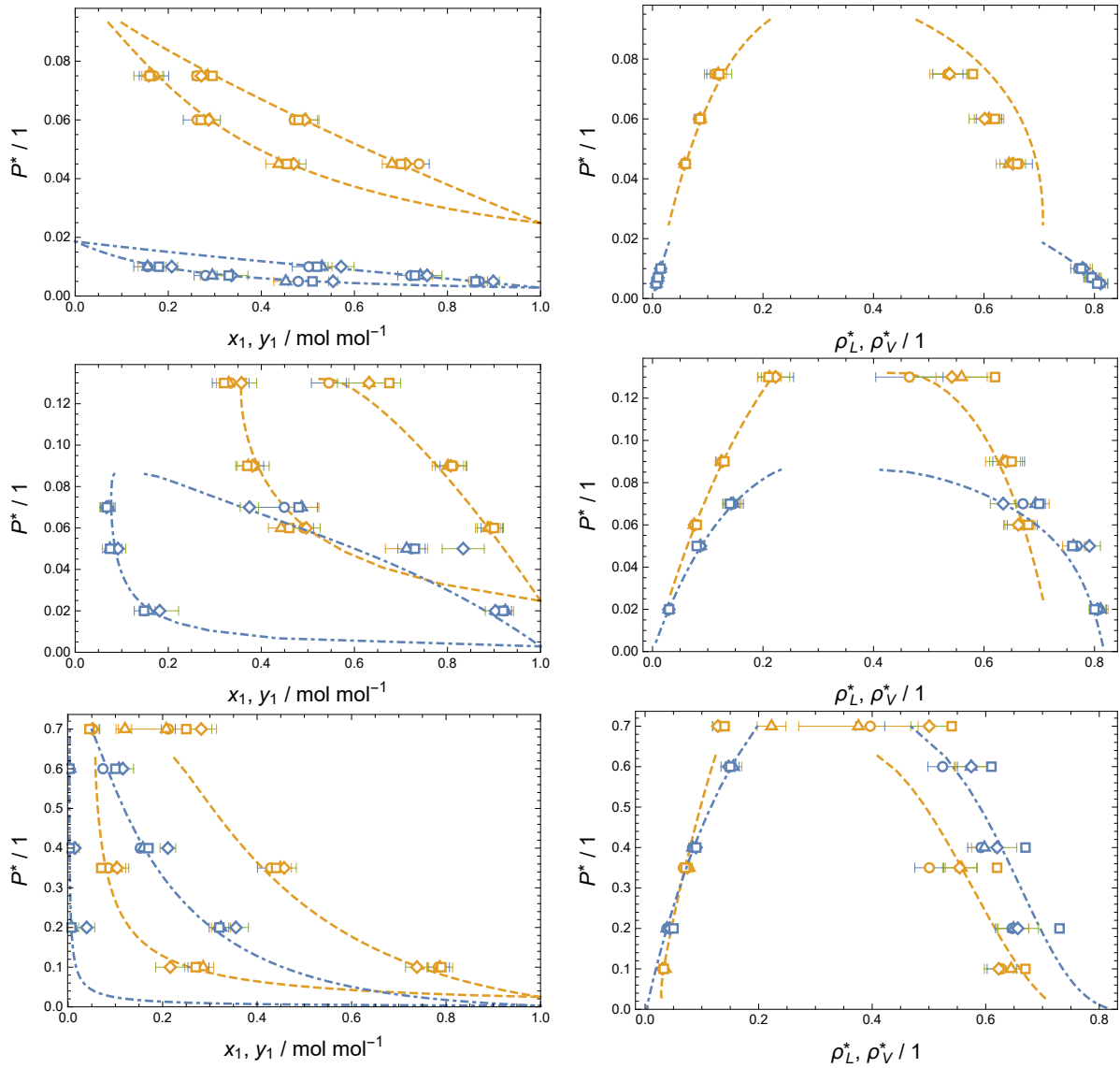


Figure 6.8.: Simulation results of the investigated systems M1 (Top), M2 (Middle) and M3 (Bottom) using different methods for insertions; \cdots t-PR-LJ EoS, $T^* = 0.75$; \cdots t-PR-LJ EoS, $T^* = 1.00$; \square EoS(BWR), \circ GEMC, \triangle PAN Swap, \diamond GECFC Pou.

CFC Method according to Poursaeidesfahani et al. In comparison to the GEMC method, the more recently proposed CFC method is still very complex, yet involving less computational effort compared to the original CFC Shi method, since only one fractional is used per component. Because of its gradual insertion, insertions are facilitated with a reasonable acceptance rate even in dense phases at low temperatures.

6.5. Charged Components

In the following section, the parameterization recommendations when using the Ewald and Wolf summation are summarized. The currently implemented set of terms for the electrostatic long-range correction only works on charge-neutral simulation boxes; in other words, it is only possible to simulate non-ionic molecules ($\sum q_i = 0$).

6.5.1. Initial Parameters for Ewald and Wolf Summation

The Ewald summation is operated with the 'range' and κ parameters, which cannot be chosen individually. The 'range' is the maximum magnitude of the vector n_k for the reciprocal space term which can only be set to positive integer values, and κ is the width parameter of the Gaussian distribution function.

Depending on the system and demanded accuracy for the energy, the ideal parameters can vary. The chosen parameters can be verified by fixing the 'range' and calculate the total electrostatic energy, U^C , at different values of κ . The ideal parameter can be determined from the resulting curve demonstrated in Figure 6.9.

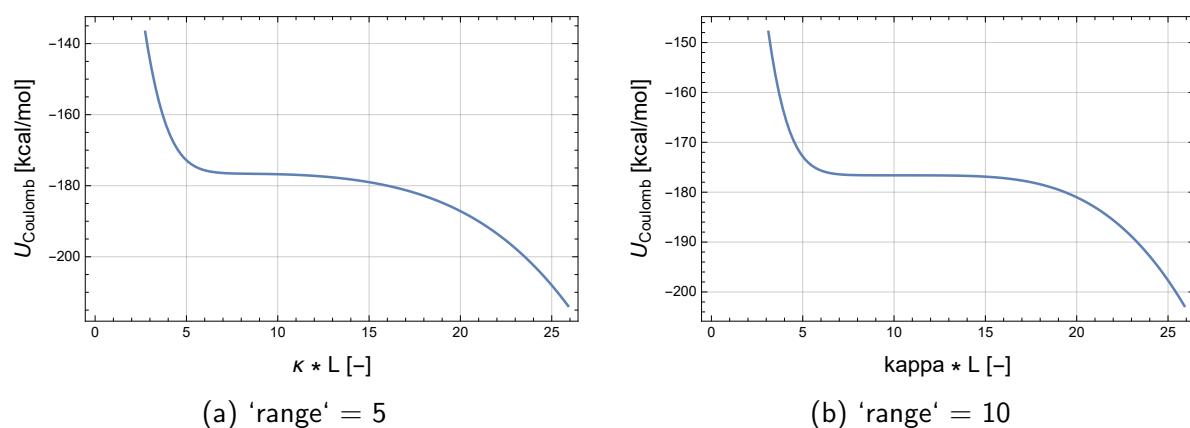


Figure 6.9.: Total electrostatic potential in a simulation box with 250 molecules of carbon dioxide at different values for κ . The value for κ should be chosen so that the total electrostatic energy barely changes around κ . A greater value for the 'range' increases the area for an ideal κ value. For example the ideal range for $\kappa \cdot L$ is in the case of (a) 7...10 and for (b) 7...15.

The Ewald summation is accurate in the range of κ where the total energy barely changes around values of the width parameter. This can be seen as an even section in the energy over κ curve. If no such section is visible, then the 'range' needs to be increased.

A good initial guess for the 'range' is 12 and $\kappa \cdot L = 10$, where L is the box side length. However, the 'range' can be reduced after verification to save computation time. The reason why κ is specified as the value of $\kappa \cdot L$ is that when the side length of the simulation box is changed, the parameter κ is adjusted so that the value of $\kappa \cdot L$ remains constant. The method can be enabled with the recommended parameters by setting the following custom parameters:

- "eLRMethod" \rightarrow "Ewald"
- "eLRkappa" \rightarrow {10,10}
- "eLRkRange" \rightarrow {12,12}

The Wolf summation is operated with the width parameter κ and further depends on the cut-off radius used in the simulation ($R_C = r_c$). Unlike in the Ewald summation, κ is kept constant throughout the simulation. The ideal value of κ is chosen by plotting the relative difference between the Wolf and Ewald summation at different values of κ and different cut-off radii. The parameter should be chosen where the relative difference is small for multiple values of the cut-off radius. An example of how the plot can look like for typical liquid and vapor phases is shown in Figure 6.10.

The method can be enabled with exemplary parameters for the simulation displayed in Figure 6.10 by setting the following custom parameters:

- "eLRMethod" \rightarrow "Wolf"
- "eLRkappa" \rightarrow {0.12,0.24}

The plots for checking the parameters can be generated in the section '*GEMC Simulator Initialization/Test specific Initialization/Check Ewald and Wolf Parameters*' of the *Execution* notebook. Execute the chapter '*Initialization*' and choose an existing simulation setup using Ewald or Wolf summation. In the subsection '*Check Ewald and Wolf Parameters*' set the first argument of the If-statement to `True` and execute the cell. Change the parameters in '*parameter.m*' and repeat the process until the parameters are sufficient.

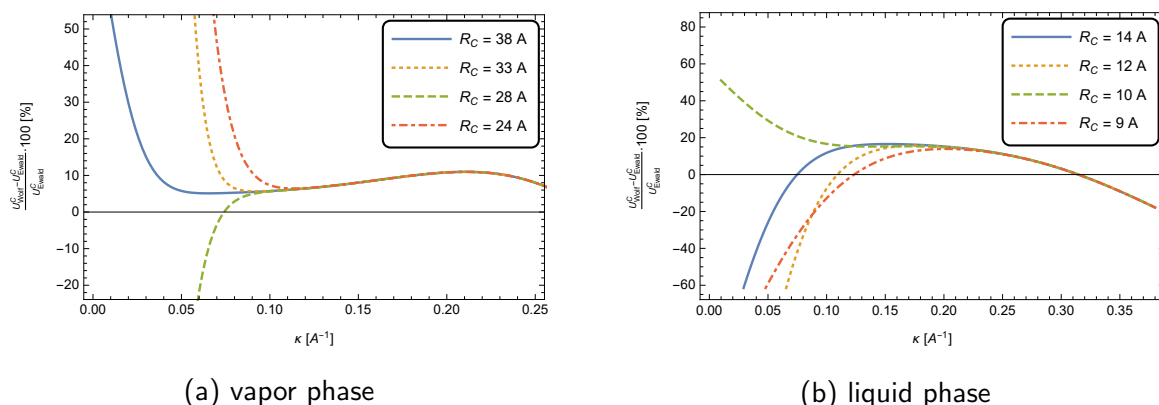


Figure 6.10.: Relative error between the Wolf and Ewald summation for a typical liquid and vapor phase of carbon dioxide. The value for κ should be chosen so that the difference is small for multiple values of the cut-off radius R_C . For example the ideal range for κ is in case of (a) 0.09...0.14 Å and for (b) 0.24...0.30 Å.

6.5.2. Guidelines

In the following, additional recommendations for simulations of partially charged molecules are given. Carbon dioxide and methane have been studied with settings as recommended for alkanes.[13]

- **Ewald and Wolf parameter**

To be chosen as described in section 6.5.1.

- **Boundary condition**

It is recommended to use metallic boundary conditions ($\epsilon_s = \text{infinity}$) because the results are closer to the literature data and the calculation time is shorter. This can be set by adding the following custom parameter:

– "eLRCes" → Infinity

- **GECFC**

It is necessary to use an overlap radius of at least $0.2 \cdot \sigma$ to avoid unrealistically low energy states.

6.5.3. Results

Carbon dioxide and methane were simulated as partially charged molecules. Carbon dioxide is modeled with three interaction sites representing two oxygen and one carbon atom. Methane is modeled with five interaction sites representing a carbon and four hydrogen atoms. The LJ and Coulomb parameters as well as the rigid geometries are summarized in Table 6.4. Simulations were conducted with Ewald and Wolf summations. The results are displayed in

Table 6.4.: Parameters and geometries for methane and carbon dioxide as a partially charged rigid molecules. σ is the size and ϵ is the energy parameter of the LJ potential, q is the electrostatic charge parameter, l is the bond length, and ϕ is the bond angle.

Component	Element	$\sigma[\text{\AA}]$	$\epsilon/k_B[\text{K}]$	$q[e^-]$	$l[\text{\AA}]$	$\phi[\text{deg}]$	Reference
CH ₄	C	3.50	33.21	-0.24	-	-	[20]
	H	2.50	15.10	0.06	-	-	
	C-H	-	-	-	1.09	-	
	H-C-H	-	-	-	-	109.47	
CO ₂	C	2.80	27.00	0.70	-	-	[17]
	O	3.05	79.00	-0.35	-	-	
	C=O	-	-	-	1.16	-	
	C=O=C	-	-	-	-	180.00	

Figure 6.11 with a comparison to literature data and an estimate using the Soave-Redlich-Kwong (SRK) equation of state. A comparison between metallic and vacuum boundary conditions was only done using the Ewald summation.

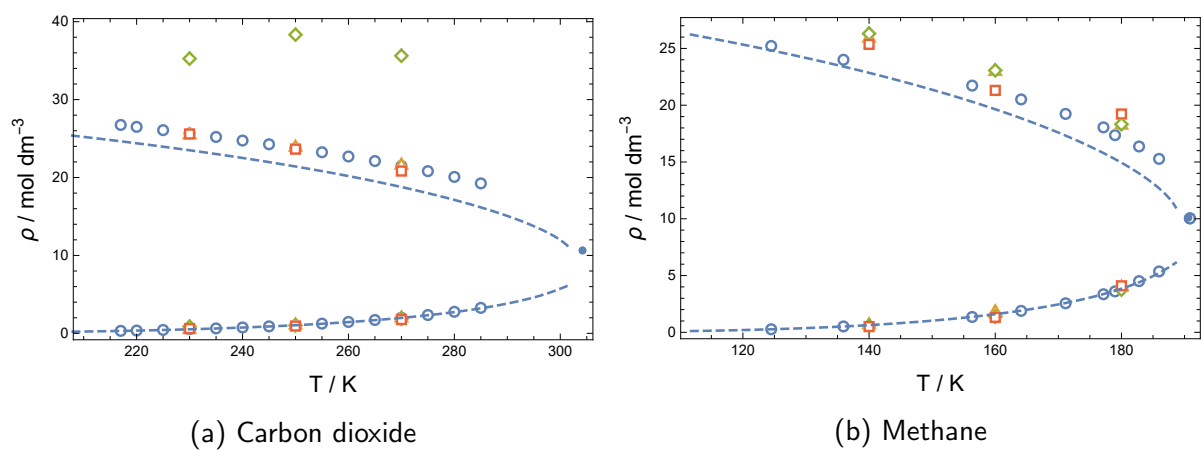


Figure 6.11.: Simulation results for carbon dioxide and methane as partially charged rigid molecules. (---) SRK EoS; \circ literature data; \triangle Ewald with $\epsilon_s = \infty$; \diamond Ewald with $\epsilon_s = 0$; \square Wolf with $\epsilon_s = \infty$; \bullet critical point

6.6. Intramolecular Flexible Components

In the following section, the main findings of the master thesis of Michael Haring [14] are summarized. The components methane, methanol, ethanol, ammonia and acetone were simulated using both united-atoms and all-atoms force field models. To allow intramolecular flexibility the three additional trial moves bond stretching, angle bending and dihedral torsion were performed. The work focused on the influence of intramolecular flexibility on the vapor-liquid equilibrium and used protocol parameters from literature.

The number of cycles, translations, rotations, insertions, volume changes and further parameters were taken from the recommendations for rigid molecules (see subsection 6.3.1). In addition to those settings, bond stretch and angle bend trial moves were executed equal to the amount of atoms in the system per cycle. Torsion trial moves were executed equal to the number of independent rotation axes in the system. For example, since methanol using the OPLS-AA force field consists of six atoms and one dihedral axis, a MC cycle included 3,000 bond stretch, 3,000 angle bending and 500 dihedral torsion trial moves.

Intramolecular trial moves were performed with a fixed trial move limit Δl_{\max} , $\Delta \phi_{\max}$ and $\Delta \theta_{\max}$ specified for each constraint in a molecule. The value of the trial move limit was chosen depending on the energy parameter, using a high value if the energy constant was low and a low value if the energy constant was high. The parameter was chosen between 0.1 and 0.2 Å for Δl_{\max} and between 10 and 30 degrees for $\Delta \phi_{\max}$. Dihedral torsions were always performed with $\Delta \theta_{\max}$ equal to 180 degrees. These trial move limits were not adjusted to yield a 50% acceptance rate with the trial move type.

6.6.1. Custom Parameter

It is possible to enable further analysis features for simulations with intramolecular trial moves through the following custom parameters.

- **Intramolecular Configuration Distribution Analysis**

This feature allows for the sampling of the distribution of bond lengths, bond angles and dihedral angles. The values are taken from every molecule in the system at the end of each MC cycle. The bin sizes for the individual attributes can be specified as well with default values as follows.

- "intraConfigDocu" → True
- "intraConfigDocudr" → 0.01 Å
- "intraConfigDocudphi" → 0.5 deg
- "intraConfigDocudtheta" → 2.0 deg

- **Jacobian**

The acceptance rule for bond stretches and angle bending trial moves includes a Jacobian term as described in subsection 5.3.2. The following custom parameter specifies whether this factor is calculated (True) or set to 1 (False).

- "intraJacobian" → True

- **Radial Distribution Function Sampling**

The following custom parameter enables the sampling of the radial distribution function (RDF) after documenting the system state in the production phase. The RDF can be sampled per molecule or per atom type in the system with each possible interaction partner within the cut-off radius. The parameter 'rdfDocudr' specifies the width of a bin used to sample the RDF. By default, a bin width of 0.01 Å is used.

- "rdfDocuMolec" → True
- "rdfDocuMolec" → True
- "rdfDocudr" → 0.1 Å

6.6.2. Simulation Results

Some of the intramolecular conformational distribution analysis plots of methanol are shown in Figure 6.12. The simulation of methanol was performed using the OPLS-AA model and settings as in the example simulation. Figure 6.13 shows the simulation results of ammonia simulated as a pure component and the binary mixture acetone-methanol simulated in the NVT ensemble. The OPLS-AA force field model was used and simulations were performed both with and without intramolecular trial moves. The results showed that the greatest influence on the VLE comes from the dihedral potential and a change in the dipole moment through different angles.

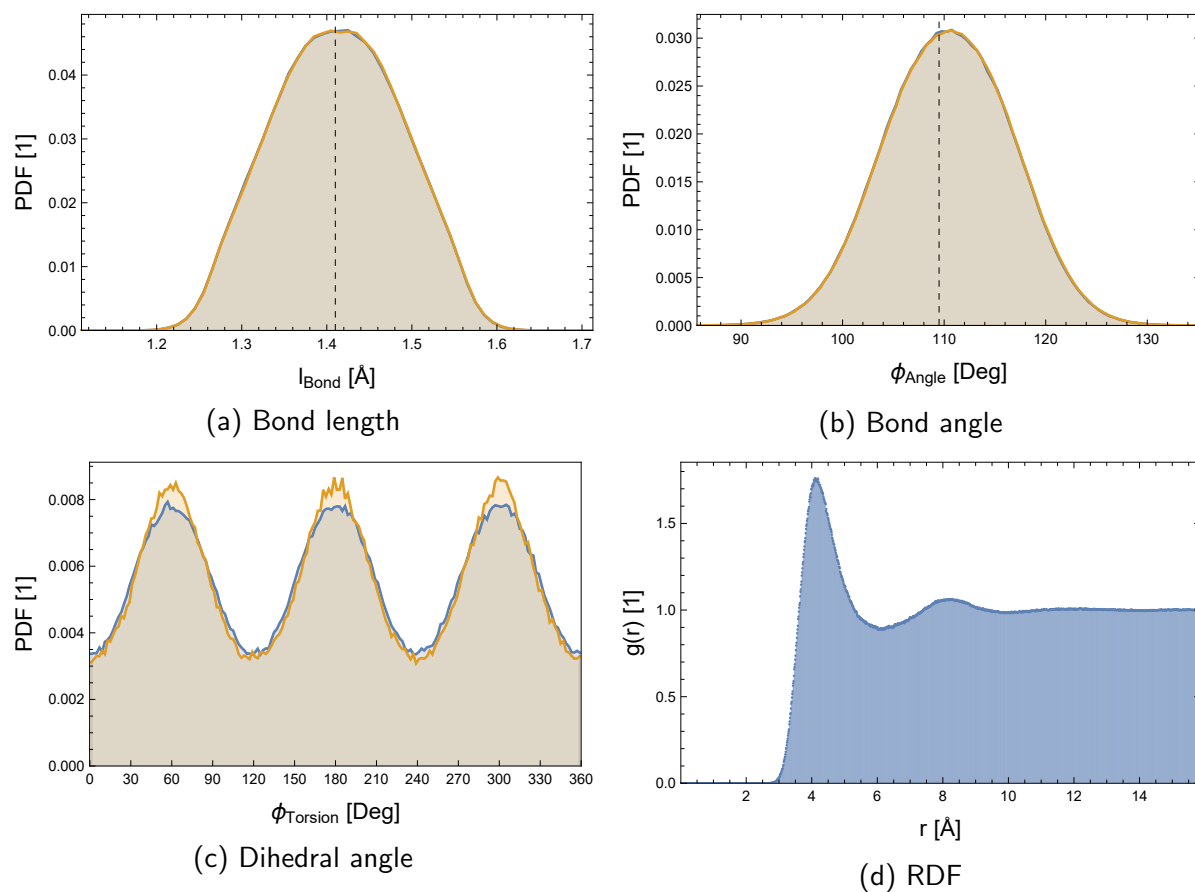


Figure 6.12.: Simulation results of methanol using the OPLS-AA model. The figure shows the distribution of the “C-O” bond length (a), the “C-O-H” bond angle (b) the “H-C-O-H” dihedral angle (c) and the radial distribution function of the molecule centers (d).

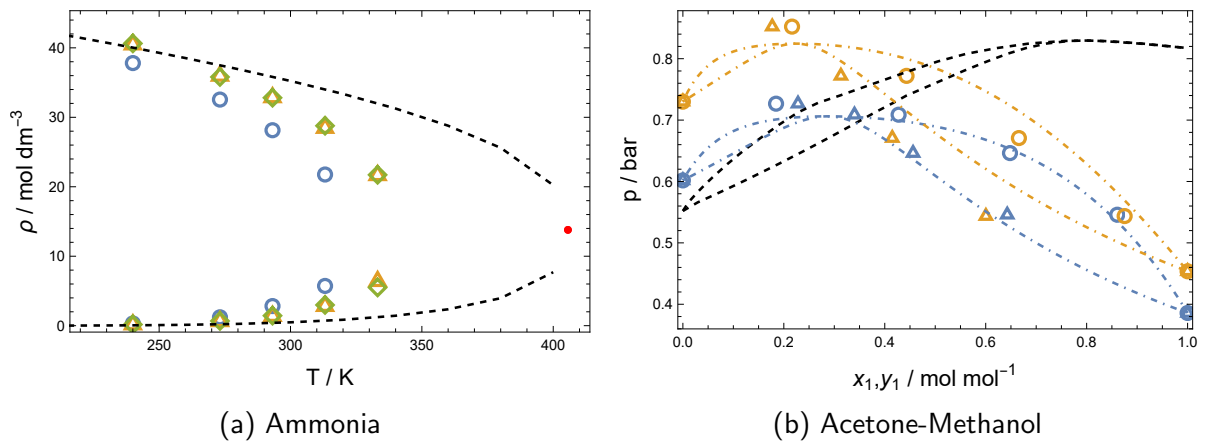


Figure 6.13.: Simulation results compared to literature data (----). (left) Vapor and liquid density of ammonia using the OPLS-AA model. Simulations were performed without intramolecular trial moves (\circ), with intramolecular trial moves (\triangle) and with intramolecular trial moves including the Jacobian term in the acceptance rule (\diamond). \bullet critical point (right) Vapor (\triangle) and liquid (\circ) mole fractions of the binary mixture acetone-methanol at $T = 323.15 \text{ K}$ using the OPLS-AA model. Simulations were performed without intramolecular trial moves (----) and with intramolecular trial moves (----). The results are visually linked through a fit of the NRTL model using a constant non-randomness parameter $\alpha_{12} = 0.47$.

Appendices

I. Step by Step Instructions for the first Simulations³

In the following section, a guide to configure, execute and evaluate three simulations is given. As an example, argon as a pure LJ fluid is simulated at medium and high temperature. Further, carbon dioxide is simulated using the Ewald summation for electrostatic long-range correction.

The simulation setups and results are stored in the folder ‘_examples’ on the ‘docs’ branch.

I.I. First Example - Pure LJ fluid at moderate temperature

In the following section the required steps for the pure LJ fluid simulation are given.

Setup of the Simulation

Most things are documented in the notebook and in section 5.2. Thus, except for the molecule and OPLS-AA setup, for which also screenshots are provided, only the necessary settings are given below.

Step by step First of all, it is recommend NOT to use the "Run Package" button in the toolbar. Each section should be executed separately, because this makes it easier to check the inputs. Since running a simulation can take up to several hours, it is highly recommended to double check all inputs.

Init This section only needs to be executed. It should be checked that no errors occur during evaluation. Thereby, it is assured that all packages are initialized properly.

³Theresa Plesch contributed to this section.

Molecules & their parameters

- **Molecule types**

Here, the molecule that is going to be simulated is defined as shown in Figure 14. First, the molecule types/names need to be set in the section '*Molecule Types*'. It is important to note that the name of the system must correspond to the available '*.mol'-files, where the molecular data is stored. Otherwise, the molecules cannot be visualized to check if the OPLS-AA parameters have been properly set.

Apart from that, one general principle can be seen in Figure 14: Throughout the notebook, `Switch[]` structures are used in case different procedures are available: First, the preferred option must be set. Usually, one option is 'manual', where the setting is set right below. In case a function is called - in this example `AutomaticSetupMoleculeData[]` - more information can be found in the definition of the function, which is stored in the subsection above the switch statement.

- **Molecule bonds**

The molecule bonds can be imported, similar to the molecule data, either manually or automatically from a '*.mol'-file. As discussed above, the procedure can be specified in the `Switch[]` statement.

- **OPLS-AA parameters**

This section is maybe the most tricky one since it is about the actual model parameters for the simulation. Apart from the physical applicability, the set parameters need to be the same as in the *ForceFieldOPLS* package. The associations in the sections '*OPLS-AA bond stretching*', '*-Angle Bending*' and '*-Torsion*' need to be empty because neither bond stretching, angle bending or torsion occur for a LJ fluid. Exemplary input is shown in Figure 15 for the non-bonded intermolecular interactions.

- **Combining rule for pair interactions and extract OPLS-parameters**

The combining rule in the section '*Combining Rule for Pair Interaction and Extract OPLS-Parameters*' can be set to `"Default"` since we simulate a pure component system. Once every variable is set, execute the '*OPLS-AA Parameters*' section. Execute the '*Consistency Check*' section to get feedback whether the OPLS-AA parameters are set in a decent way or not.

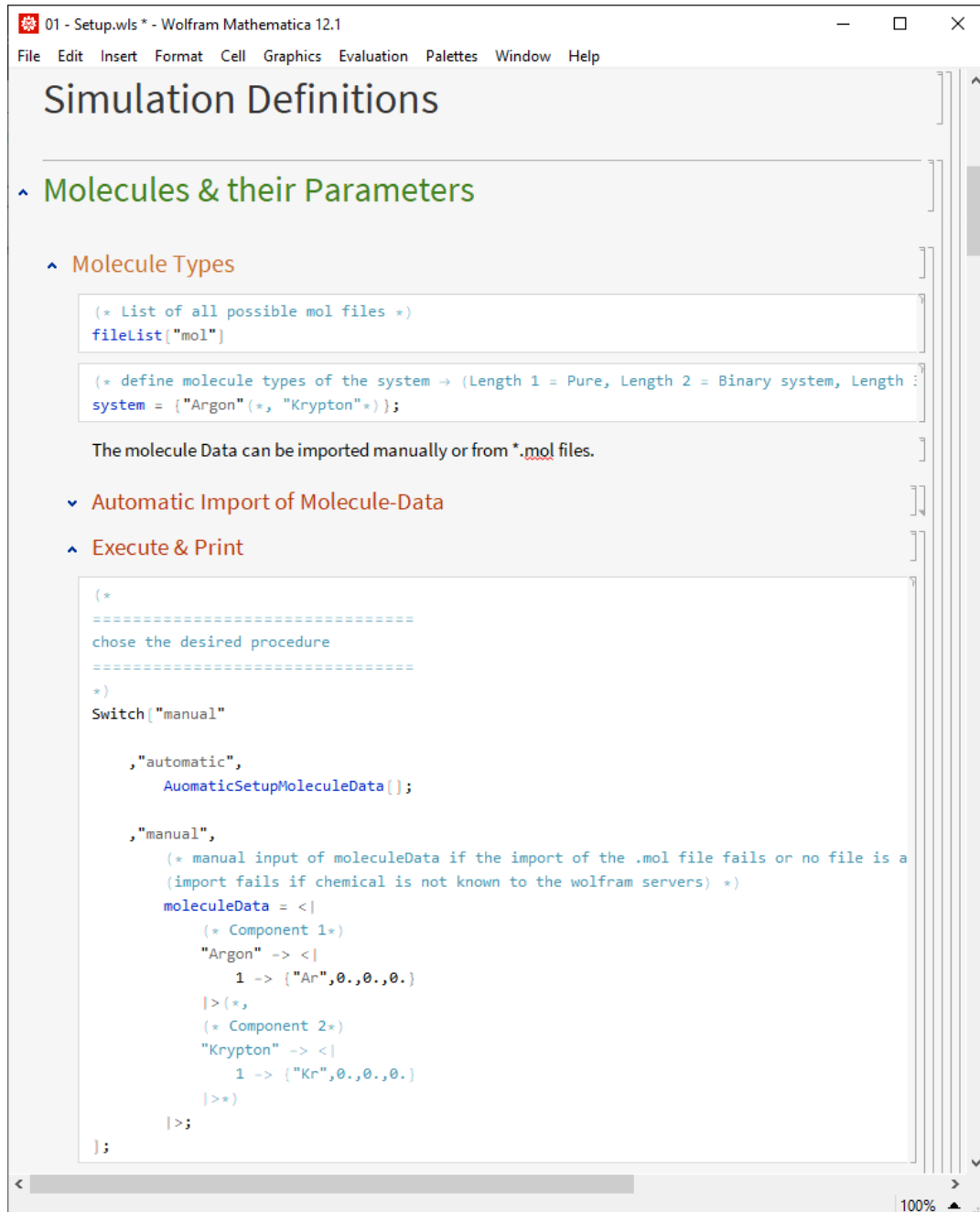


Figure 14.: Setup of molecule types



Figure 15.: Setup of OPLS-AA parameters.

System Properties Here, the number of molecules, temperature, pressure and other system relevant properties are set.

- **Ensemble, number of molecules**

The ensemble type for a single component needs to be set to `"1"` indicating the NVT-ensemble. The total number of molecules need to be set. Usually, a simulation is set up with a total number of `ntot=500` molecules.

- **Molar fraction and box set up**

For a single component the molar fractions `x1V` and `x1L` need to be set to `1`.

- **Temperature, pressure, and molar volume**

The system properties can be set using LJ reduced units since argon is modeled as a LJ fluid. Therefore, the '*Lennard-Jones reduced units*' section is used and needs to be adjusted. The data is reduced with the LJ parameters σ and ϵ . Set `TRed=1.00` as well as `rhoRedV=0.06` and `rhoRedL=0.6`. Afterwards, the function needs to be called in the following section '*Execute & Print*'.

System properties implementation

- **System properties implementation**

The molecule distribution can be set with several procedures as described in subsection 5.2.2. In our example we use '*vapourDoubleVolumeLiquid*'. Afterwards, the entire section needs to be executed. If everything worked out, the output is the `ncompVec` which shows how many molecules are in each box.

- **Create coordinates**

The '*Create Coordinates*' section only needs to be executed. Thereby, the actual initial positions of the atoms are created.

- **Visual check**

The visual check shows how the simulation boxes look like. After executing the section, the output may look like Figure 16a. The visual check is a plausibility check, if it looks like Figure 16b for example, an error has arisen somewhere in the set-up before.

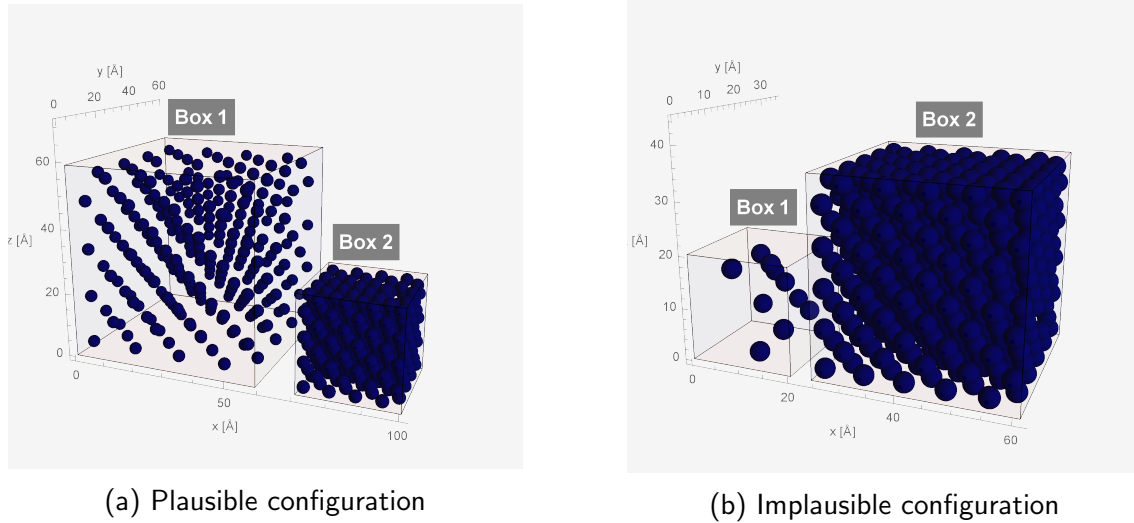


Figure 16.: Comparison of a (a) plausible and (b) implausible example. The second example is not plausible since the liquid box is larger than the vapor box. Consequently, there would be too few molecules in the vapor box which would result in a very unstable simulation course.

Number of cycles and trial moves For choosing an appropriate number of cycles and trial moves, the recommendations by Andreas Schwarz given in section 6.1.1 can be used: 2000 warm-up cycles, 2000 equilibration cycles and 8000 production cycles are executed. Regarding the trail move composition, 500 (n_{tot}) translations, 0 rotations, 1 volume change, 100 ($0.2 \cdot n_{tot}$) insertions and 125 ($n_{tot} / 4$) Widom insertions are attempted.

- **Trial move limits**

The trial move limits can be left as they are set by default since these values are a plausible start value for pure component LJ fluid simulations: 2.0 \AA for the maximum translation distance and 0.01 times the volume of the smaller box for the maximum volume change.

- **Cut-off radius**

For the cut-off radius the switch needs to be using the 'boxspecific' procedure. Thereby, the cut-off radius is set to half of the individual box side length.

- **Overlap radius**

For the overlap radius, the 'automatic' procedure should be used. Make sure to evaluate the whole section '*Overlap Radius*'.

Final check The execution of this section shows an overview of all the set values to check if the values were set correctly in the previous sections (Figure 17). Make sure to double check the number of molecules and the initial molar volumes!

Export To export the simulation definitions, a folder, where the file should be saved at, needs to be created. The first folder (in this example 'export') is the main folder. The other folders are subfolders. The folders should be named in a logical way so it can be identified easily. After the execution of '*Try to export Variables*' the output must be "Export successful".

Custom parameters for pure substances To set the custom parameters, a separate section called '*Test specific parameters*' is used. It is located right after '*Export*' in the *Setup* notebook. For pure LJ fluids, only a few subsections need to be executed:

- **Setup**

First the '*Setup*' section initializes the `custParam` association.

- **Trial move composition**

Using this custom parameter, the number of attempted swaps is adjusted to yield - on average - one successful insertion per cycle. Again, the number of attempted moves is adjusted once 10 cycles have been executed. Thereby, a stable performance of the simulation is ensured.

- **Trial move limits**

Using this custom parameter results in a more stable adjustment of the volume change trial move limit. The background is that only 1-3 volume changes are executed each cycle. In consequence, it is hard to achieve an acceptance rate of 50 % over each cycle. Thus, the trial move limit is adjusted once at least 10 moves of the respective move were executed.

- **Export**

Finally, the export section needs to be executed to save the set custom parameters. Check that the custom parameter file '*parameter.m*' was created in the simulation folder.

Now the simulation is ready to be executed!

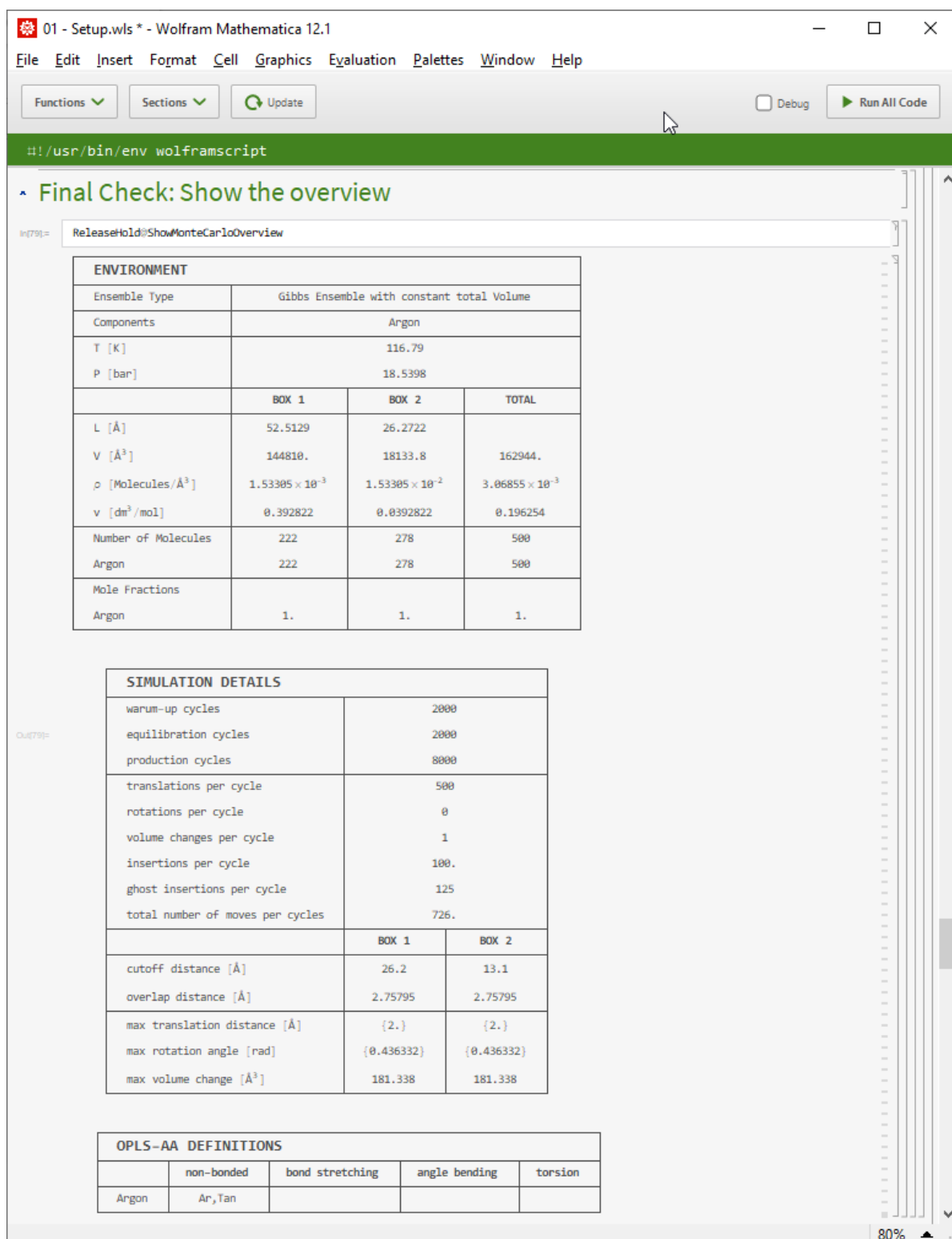


Figure 17.: The overview

Execution of the simulation

Now it is time to run the simulation. Open the *Execution* notebook and evaluate all cells using the "Run All Code" button. Mathematica will open an explorer, where you need to select the simulation to be executed. Navigate to the previously set-up simulation folder in 'export' and continue.

During the simulation, the progress is indicated in a live-visualization in the section '*Progress Indication*'. The elapsed cycles and the estimated remaining time are displayed in a grid. The course of the number of molecules, the molar volume and the number density in each box are plotted. However, the update of the `Dynamic[]` graphic causes a significant memory leak. Thus it is recommended not to display the graphic constantly, but rather check it a couple of times during the simulation (just scroll up to the top of the notebook or minimize the window). Each time the graphic is displayed, all previous cycles are then loaded.

Analysis of the Simulation

Once the simulation has finished, its results can be evaluated. First, it needs to be assessed if the simulation was sufficiently stable to reasonably evaluate the results. Therefore, the Frenkel plot, displayed in Figure 18, is checked. Indeed, the successful simulation quickly converged towards the equilibrium configuration and sampled the phase space efficiently. This can be seen since the equilibrium configuration is substantially more frequently sampled (cf. the probability indicator in the bar on the right side of the plot per phase) compared to the path of the convergence. By contrast, the unsuccessful simulation resulted in a state where most molecules are in one of the boxes. Since both states are almost on the diagonal, it is obvious that the molar volume in both boxes are very similar. Thus we can expect that no phase split occurred.

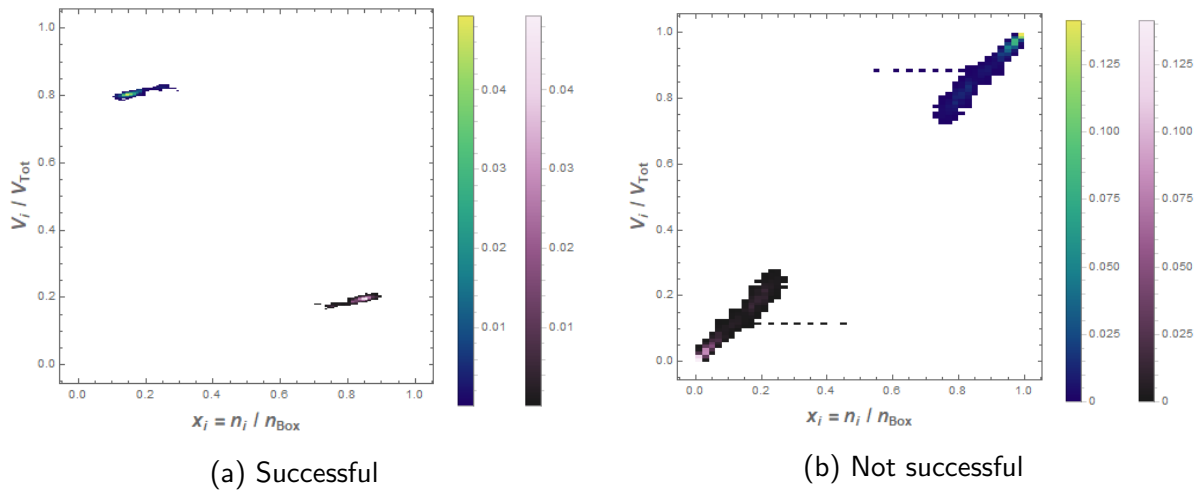


Figure 18.: Comparison of the Frenkel plot of a (a) successful (b) not successful simulation.

The simulation results are further analysed by comparing the number of molecules per box, as displayed in Figure 19. As expected, the number of molecules in the successful simulation converged quickly (within the equilibration phase!) while the number of molecules in the unsuccessful simulation did not. As described above, almost no molecules remain in the second box - it appears that sometimes the second box was even completely empty!

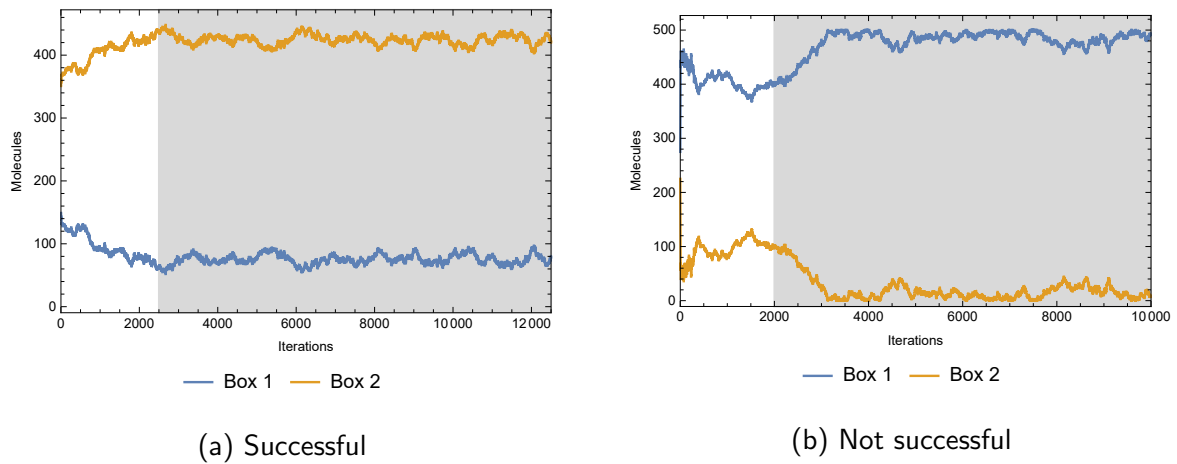
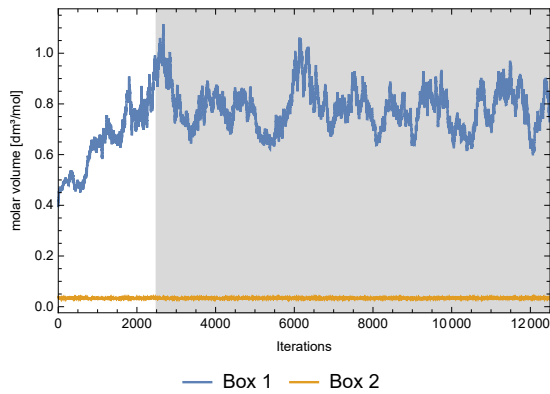


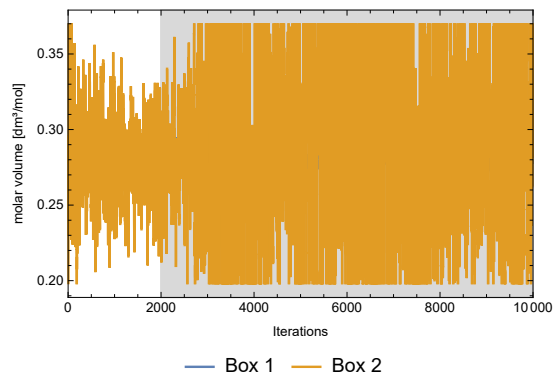
Figure 19.: Comparison of the course of the number of molecules for a (a) successful and (b) not successful simulation.

Finally, the courses of the molar volume of the two simulations are analysed in Figure 20. Again, we see that the simulation results of the first simulation are stable, since they fluctuate

around the equilibrium and thereby sample the phase space efficiently. Mind that a Monte Carlo simulation will *never* be totally stable since also uphill moves are accepted to prevent that the simulation is trapped in local minima. However, by calculating the ensemble averages, the actual values are determined. Regarding the unsuccessful simulation, it is observed that the molar volume of the almost empty second box fluctuates extremely. This can be explained by the very low number of molecules in this box - mind that the molar volume of the box is defined as $v_{\text{Box}} = V_{\text{Box}} \cdot n_{\text{Box}}$. Consequently, the molar volume fluctuates drastically, since the number of molecules is in the denominator and the number of molecules per box is very low.



(a) Successful



(b) Not successful

Figure 20.: Comparison of the course of the molar volume for a (a) successful and (b) not successful simulation.

I.II. Second Example - Pure LJ fluid at high temperature

The second example is also about the simulation of argon as a pure LJ fluid. Thus, the OPLS-AA parameter can be set as described in subsection I.I. However, the initial configuration needs to be altered, likewise the recommendations of Panagiotopoulos [28] are followed:

- **Temperature**

The LJ reduced temperature is set to $T^* = 1.30$.

- **Molar volume**

The LJ reduced molar volume is set to $\rho_V^* = \rho_L^* = 0.35$.

- **Number of molecules**

The number of molecules is set to $n_V = n_L = 250$.

- **Number of insertions**

The number of attempted insertions per cycle needs to be set to 20 ($0.04 \cdot n_{\text{tot}}$).

The course of the number of molecules and the molar volume of this simulation are displayed in Figure 21.

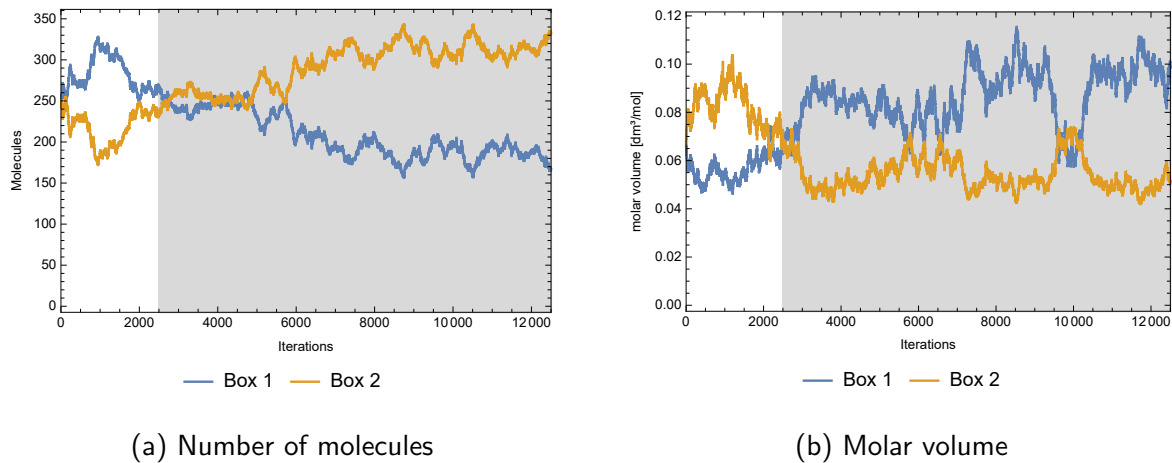


Figure 21.: Comparison of the course of (a) the number of molecules and (b) the molar volume of argon at $T^* = 1.30$.

While the number of molecules seems to be quite stable in the end of the production phase, the molar volume clearly remains unstable. That indicates that the system is not properly

equilibrated during the executed cycles. Thus, it was decided to extend the simulation with another 10^4 production cycles. The results are displayed in Figure 22.

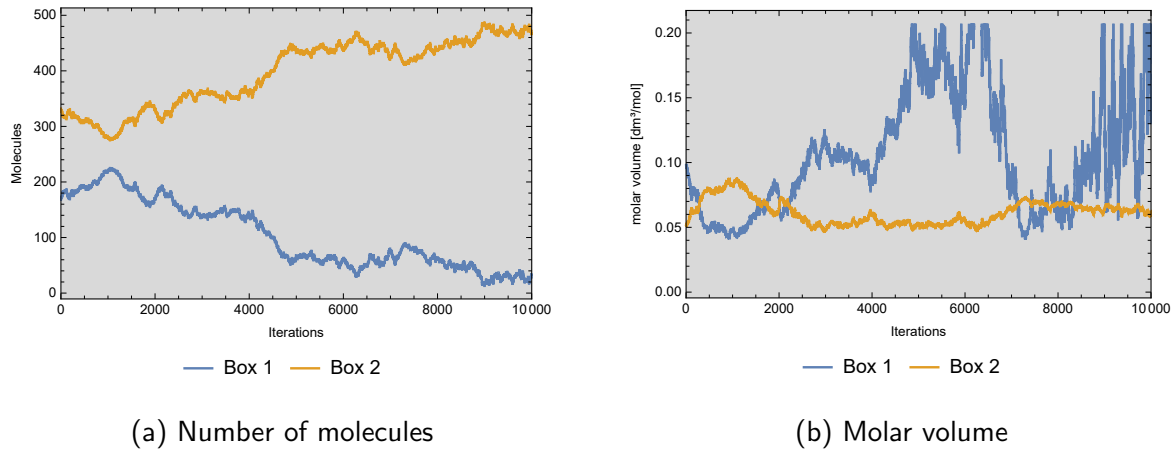


Figure 22.: Comparison of the course of (a) the number of molecules and (b) the molar volume of Argon at $T^* = 1.30$ of the extended simulation.

It is observed that too many molecules are transferred in the liquid box and only few remain in the vapor box. Consequently, the molar volume of the vapor box becomes very unstable. Thus, it is decided to set up a new simulation with the following settings:

- **Temperature**

The temperature remains at $T^* = 1.30$ which is equivalent to $T = 151.827$ K.

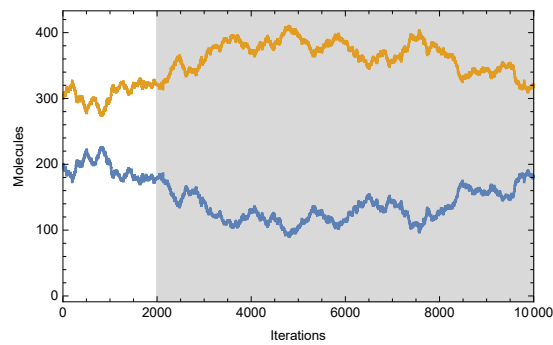
- **Molar volume**

The molar volume is set to $v_V = 0.15$ dm³/mol and $v_L = 0.05$ dm³/mol.

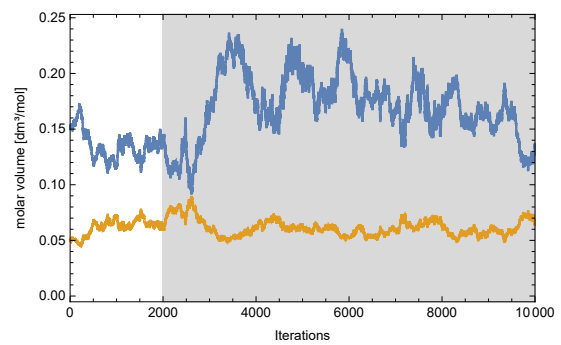
- **Number of molecules**

The number of molecules is set to $n_V = 200$ and $n_L = 300$.

The results are displayed in Figure 23. With these settings, a sufficient number of molecules remains in the liquid phase, whereby the molar volume is substantially more stable than before. Still one could consider executing another simulation with even more molecules in the vapor phase (resulting in a higher total system volume) - it would most likely further improve the stability of the system and thus the simulation results!



(a) Number of molecules



(b) Molar volume

Figure 23.: Comparison of the course of (a) the number of molecules and (b) the molar volume of argon at $T^* = 1.30$ of the simulation setup with the new configuration.

I.III. Third Example - Carbon Dioxide

The third example is about carbon dioxide as a partially charged rigid molecule. Since electrostatic interactions need to be considered, a long-range correction method for electrostatic interactions needs to be used. The main aspects about executing a simulation are already discussed above. The setup of this simulation only differs while setting up the molecule, the system temperature and the custom parameters. In this example, a short simulation using the Ewald summation will be set up.

Molecules & their parameters

- **Molecule types**

Since the file '*CarbonDioxide.mol*' already exists for carbon dioxide, the 'automatic' procedure can be used to define the arrangement of sites in the section '*Molecule Types*'. Make sure however, that the name in the system is "CarbonDioxide", as it needs to be the same name as the '*.mol'-file and the `ChemicalData` association for later extraction of critical data.

- **Molecule bonds**

Use the 'automatic' option to import the molecule bonds as well.

- **OPLS-AA parameters**

The labels for the parameters of carbon dioxide are already defined in the *ForceField-OPLS* package. The order in which the labels are stated corresponds to the order of atoms defined in '*Molecule Types*'. Since we used the 'automatic' option it is important to check the order by visualizing the molecule in the section '*Visualize Components*' as shown in Figure 24. The input of OPLS labels is shown in Figure 25. The associations in the sections '*OPLS-AA bond stretching*', '*-Angel Bending*' and '*-Torsion*' need to be empty because carbon is considered a rigid molecule in this setup.

- **Combining rule for pair interaction and extract OPLS-parameters**

The combining rule in the section '*Combining Rule for Pair Interaction and Extract OPLS-Parameters*' can either be set to `"Default"` or `"Lorentz-Berthelot"`. Once every variable is set, execute the '*OPLS-AA Parameters*' section. The '*Consistency check*' will give a warning that not enough labels for bonds have been entered. Do not be alarmed because this is intended since we use a rigid molecule.

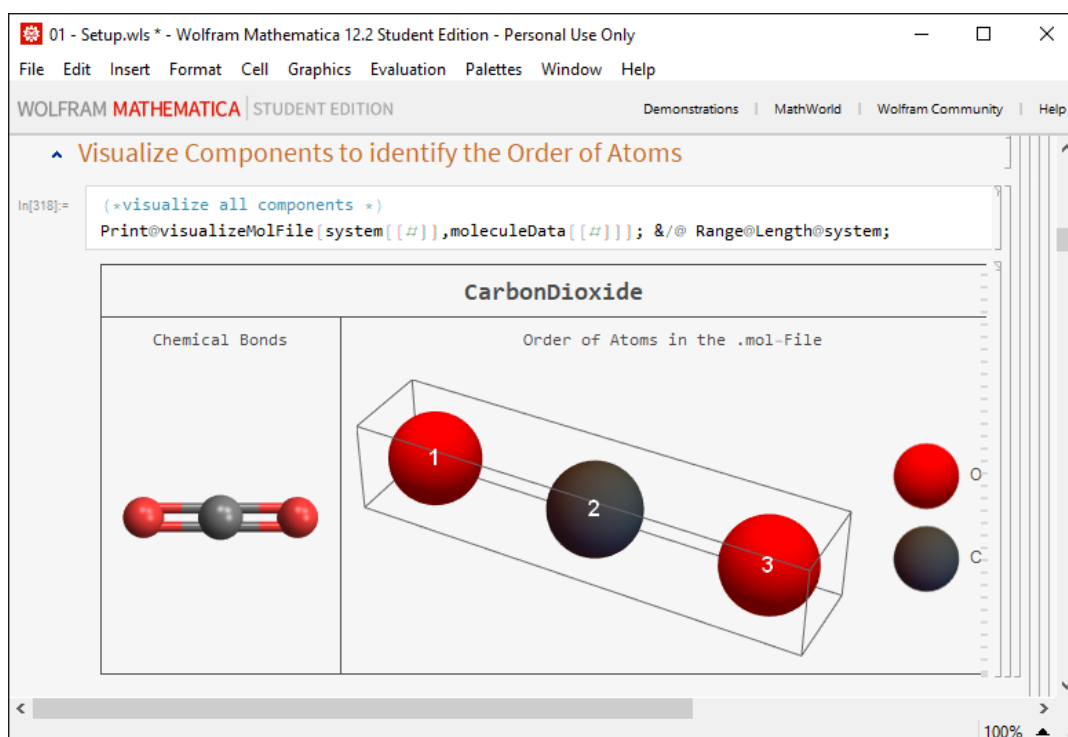


Figure 24.: Visualization of molecule sites

System properties The ensemble type is set to 1, the number of molecules `ntot = 500` and the molar fraction `x1V` as well as `x1L` are set to 1. Since the LJ-reduced units are not properly defined for poly-atomic components, the temperature and pressure need to be entered using the 'conventional' procedure. The initial values for the molar volume can be estimated with the SRK EoS in the '*Pure Component*' section of the *Estimation* notebook. An example of how to set the system properties is shown in Figure 26.

System properties implementation The initial distribution of the molecules can be done using the 'equalMolecule' procedure. Make sure to execute the entire section. The output `ncompVec` should show that 250 molecules are in each box.

Number of cycles and trial moves For choosing the appropriate number of cycles and trial moves the recommendations in section 6.3.1 can be used. In this example 2000 warm-up cycles, 5000 equilibration cycles and 5000 production cycles are set.

01 - Setup.wls * - Wolfram Mathematica 12.2 Student Edition - Personal Use Only

File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

WOLFRAM MATHEMATICA STUDENT EDITION

Demonstrations MathWorld Wolfram Community Help

^ OPLS-AA Parameters

^ OPLS-AA Parameters -> non-bonded Interactions

In[320]:= (*if necessary print the available properties*)
MatrixForm[ArrayReshape[Keys@oplsNonBonded, {Ceiling[Length@Keys@oplsNonBonded/#], #} &@ 4, "

In[321]:= (*
define OPLS-labels for every atom of the molecules
(the OPLS-parameters will be extracted from the molecularSampling package with those labels)
the labels of the atoms must be set in the order of occurrence in the .mol-file!
to show the occurrence of the atoms use the function visualizeMolFile[]
)

ffLabelsNonBonded = <|

(*
example molecule:

"Methanol" -> {
 "O,ROH", (* atom 1 *)
 "C,CH3OH and RCH2OH", (* atom 2 *)
 "H(C),CH3OH", (* atom 3 *)
 "H(C),CH3OH", (* atom 4 *)
 "H(C),CH3OH", (* atom 5 *)
 "H(O),ROH" (* atom 6 *)
}

*)

(* Component 1*)
"CarbonDioxide" -> {
 "O,C02", (* atom 1 *)
 "C,C02", (* atom 2 *)
 "O,C02" (* atom 3 *)
} (*,
(* Component 2*)
"Krypton" -> {
 "Kr,Tan" (* atom 1 *)
} *)
|>;

100%

Figure 25.: Setup of OPLS-AA parameters

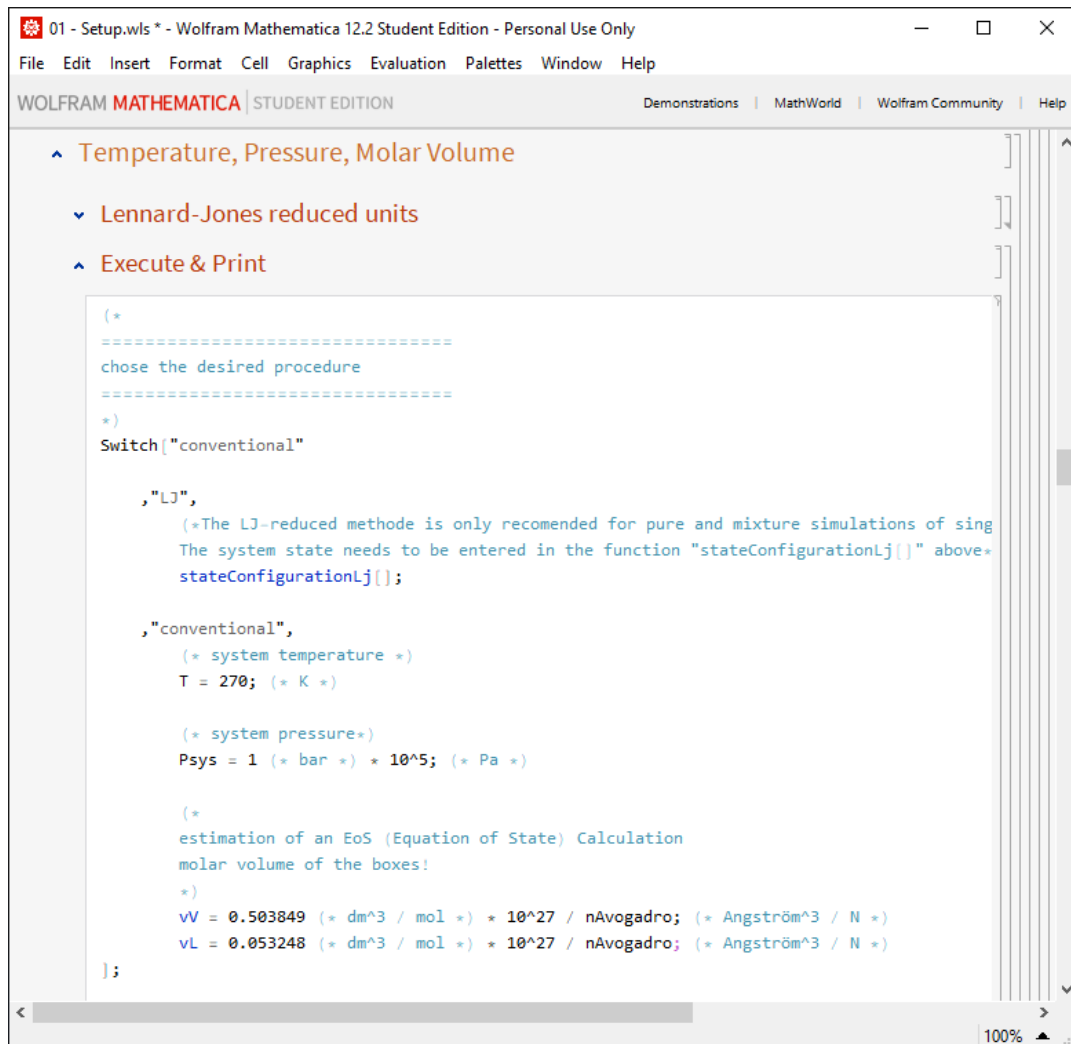


Figure 26.: Setup of system properties

A total of 1000 ($2 \cdot n_{\text{tot}}$) translations, 1000 ($2 \cdot n_{\text{tot}}$) rotations, 1 volume change, 100 ($0.2 \cdot n_{\text{tot}}$) insertions and 125 ($n_{\text{tot}} / 4$) Widom insertions are attempted per cycle.

Custom parameters As mentioned earlier, the use of an electrostatic long-range correction method is important. For this reason, add the custom parameters stated in '*Electrostatic Long Range Correction*' using the recommended values for the Ewald summation as shown in Figure 27. The parameters can be checked in the *Execution* notebook as described in section 6.5.1.

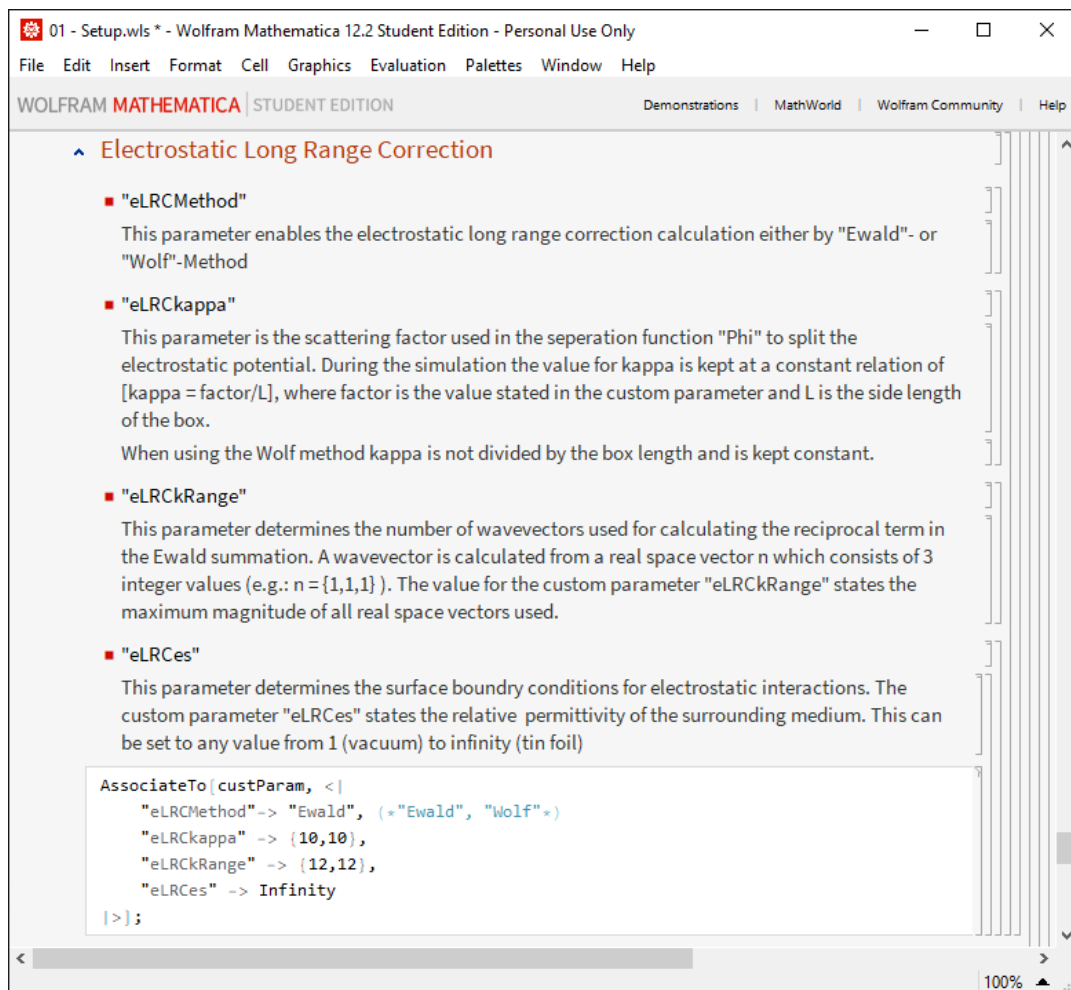
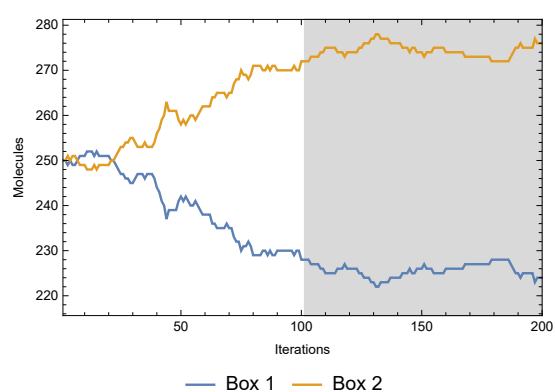
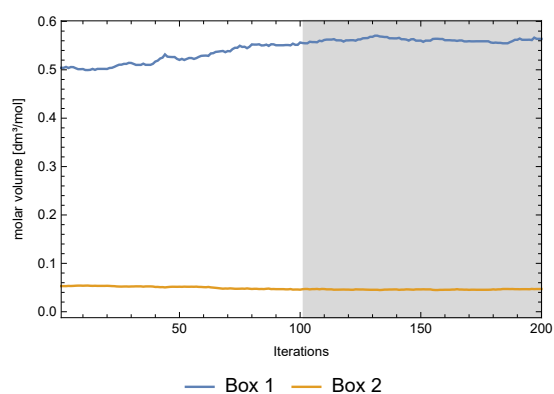


Figure 27.: Add Ewald parameter to custom parameters

Execution and analysis The course of a simulation at a temperature of 270 K with 100 cycles per phase is shown in Figure 28. For quicker evaluation, the amount of cycles was kept lower than recommended.



(a) Number of molecules



(b) Molar volume

Figure 28.: Comparison of the course of (a) the number of molecules and (b) the molar volume of carbon dioxide at $T = 270$ K.

I.IV. Usage of the "00-Estimation Notebook"

As a first step, execute the top section that is shown in Figure 29. It is possible that a message-box opens (Figure 30) and asks if all initialization cells should automatically be evaluated. Since all initialization cells are currently attempted to be executed, simply select "no" and continue.

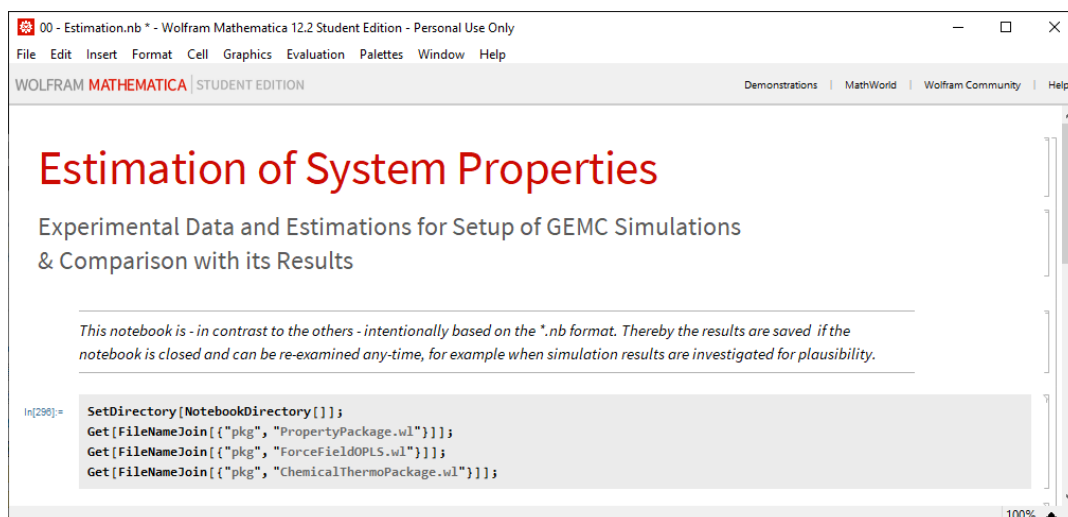


Figure 29.: Start of the Estimation Notebook

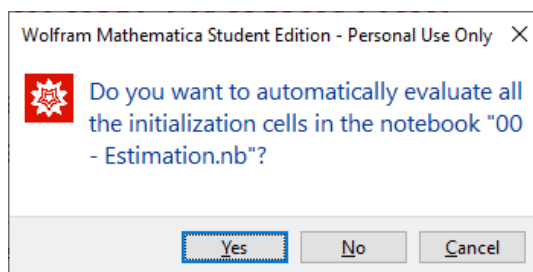


Figure 30.: Pop up - click "no"

Next, open the '*Binary Mixture LJ-fluid*' section. All cells with necessary user input are highlighted in yellow. In the subsection '*Components in System*' enter the names of the components through the variable `component` (Figure 31). The components need to be named according to their OPLS-label in the *ForceFieldOPLS* package. The component that is bigger or has the lower saturation pressure (in this example ethane) needs to be named first. Make sure to also evaluate the section '*Helper Functions*' and '*Extracting OPLSA - Parameters*'.

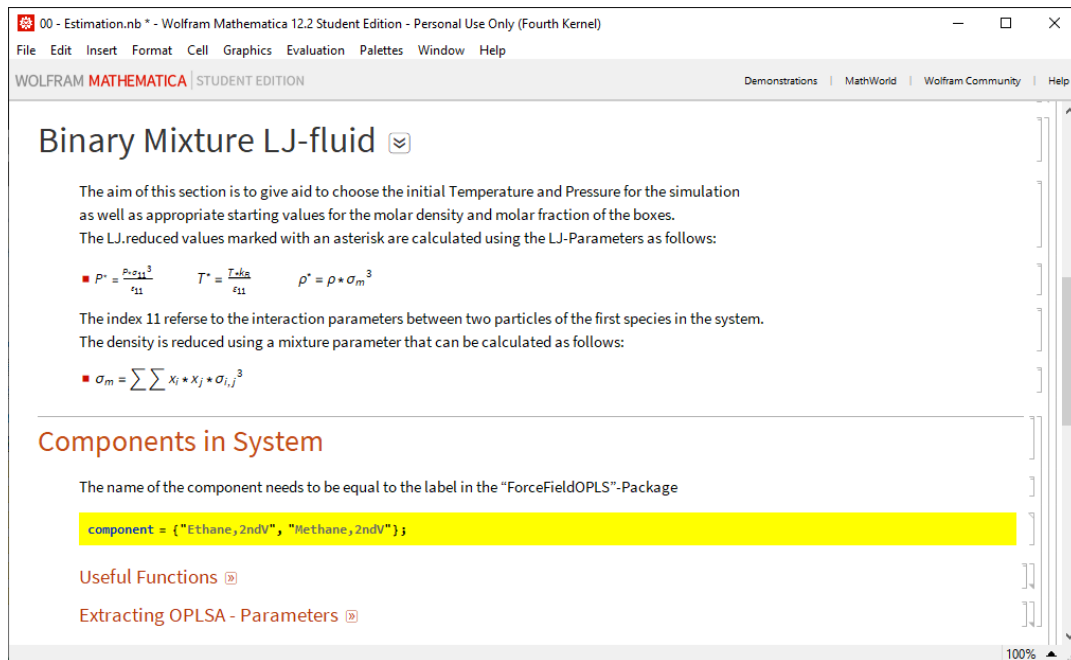


Figure 31.: Components in the System

Afterwards, the simulation temperature and pressure need to be specified. The sections '*Normal Boiling Temperature*' and '*Saturation Pressure at Simulation Temperature*' can help find appropriate values for temperature and pressure.

The temperature and pressure can be entered in LJ-reduced values or in absolute/caloric values by opening and executing the respective section (Figure 32). Multiple entries for pressure are possible. Afterwards, a summary of the specified data is provided. For an overview, the '*Summary*' section needs to be evaluated.

To estimate the molar fractions section '*t-PR-LJ EoS*' in '*Estimation of molar fraction and molar density with EoS*' (Figure 33) needs to be executed, which may take a few minutes. At the end of the calculation, the results are presented in grids stating the molar fractions of the first component (x_1, y_1) as shown in Figure 33.

Afterwards, the section '*Visualisation*' (Figure 34) can be executed to get diagrams showing the Pxy and P- ρ values of the specified system. The '*Visualisation*' also calculates the $x_{m50\%}$ values and displays them below the Pxy-diagram (Figure 34). Those values are necessary for the '*Molar Fraction and Box Setup*' section of the *Setup* notebook, where $x_m = x_L = x_V$.

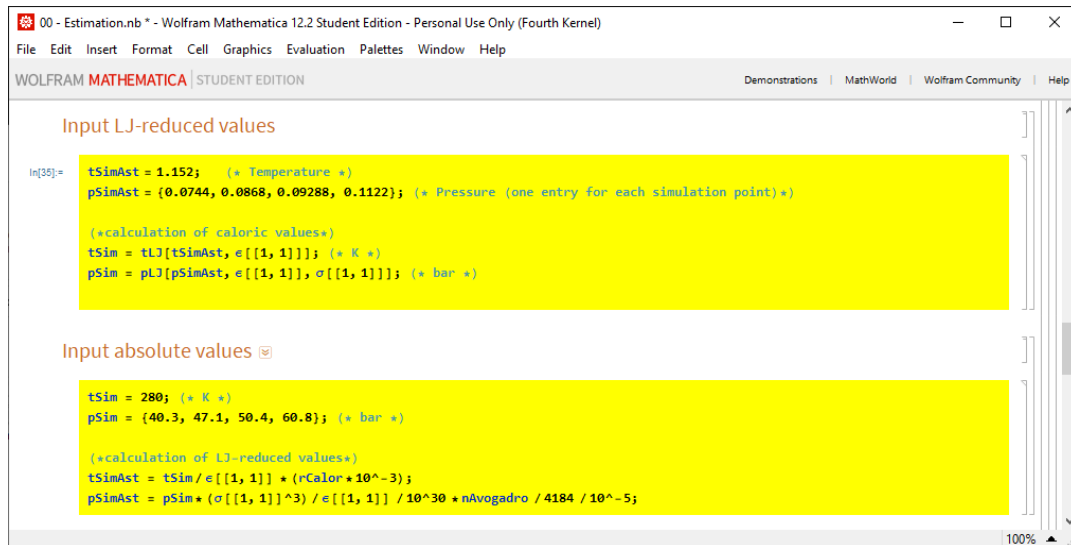


Figure 32.: Simulation Input Data

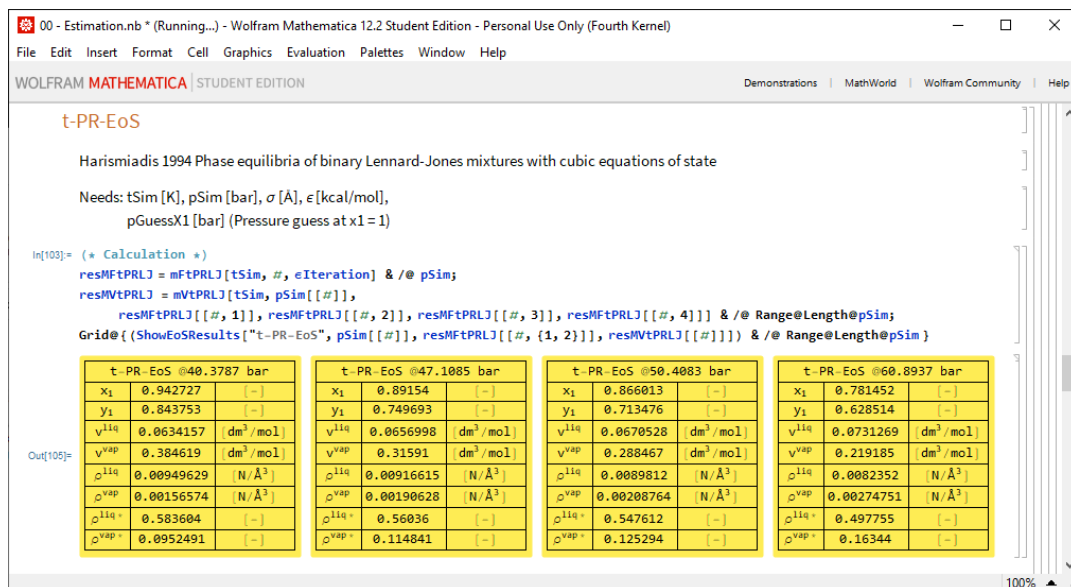


Figure 33.: Results of the calculation

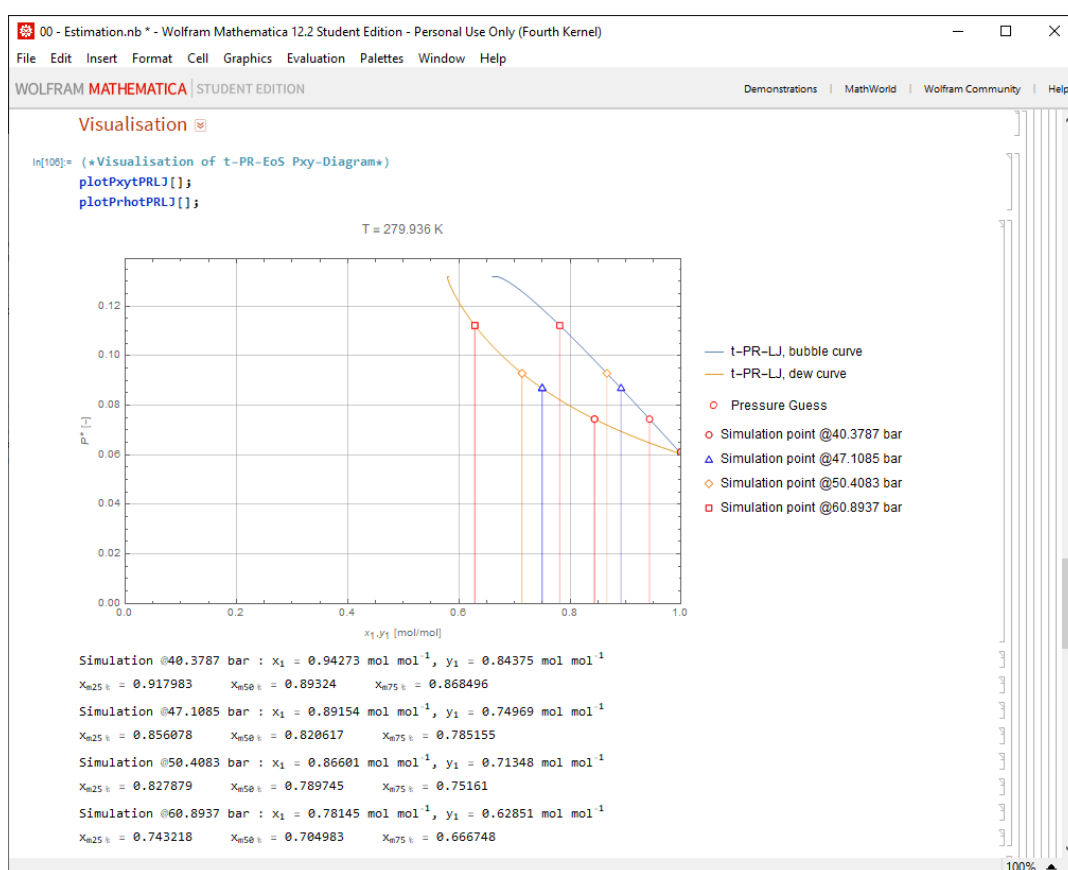


Figure 34.: Visualization of the calculated molar fractions

Future Extensions and Adaptations

There are many possible ways to improve and extend the notebooks. The following list presents some important modifications that should or can be made:

- Implementation of a trial move for bond stretching, angle bending and torsions for the OPLS force field.
- Selection of the PBC Method for Lennard-Jones interactions: Truncated and without a correction, truncated with a tail correction (currently implemented) or shifted Lennard-Jones potential
- Adjust trial move limits with a scaling factor that depends on the acceptance ratio of the last cycle instead of a fixed factor
- Possibility to automatically select pre-defined configurations in the Setup-Notebook.
- Minor optimizations are listed in the section `Possible Improvements` in some notebooks.

II. Changelog and Credits

Version 1.0

Thomas Steiner (24.02.2020)

- Basic implementation of the Gibbs Ensemble Monte Carlo Simulation

Version 1.7

Sebastian Thaler (31.08.2020)

- Notebook structure redesign
- Import & export of simulation settings and results
- Implement a grid system to flexibly initialize molecule positions
- Code optimizations (e.g., energy calculation, trial move functions)
- Progress indication during simulations
- Tail correction for a varying number of molecule types
- Implementation of warm-up, equilibration and production cycles
- Improved results analysis (e.g., ensemble averages, improved plots, exports)
- Notebook to extend simulations
- Implementation of the Widom insertion method
- Implementation of intramolecular energy calculation

Version 2.0

Niklas Mayr (28.11.2021)

- Automatic execution of sets of simulations
- Custom parameters to investigate the behavior of the algorithm like an automatic calibration of the number of swaps in the equilibration phase
- Systematic evaluation with an Excel-export & the combined analysis
- Improved swapping algorithm for binary mixtures
- Continuous fractional component (CFC) method

Version 2.3

Michael Haring (28.09.2022)

- Automatic '*.mol'-file import
- Checks & comparisons in the analysis
- Update the progress indication of all previous cycles
- Lorentz-Berthelot combining rule for binary mixtures
- Combine the simulation results of an extension with the simulation results of the base simulation
- Custom parameter to influence the choice of species for swap moves
- Change of virial calculation from updated list to separate routine
- Option to pad molecules to create atom coordinate and indicator tensors
- Implementation of Ewald and Wolf summation for electrostatic long-range correction
- Backwards compatibility check for simulation setups and results

Version 2.4

Michael Haring (06.04.2023)

- Documentation of result specific trial move evaluation time
- Code optimization (energy calculation functions with 'Listable' attribute)
- Removal of the 'ArrayPadding'-feature (no longer necessary due to code optimization)
- Improved PackedArray handling for position and energy variables
- Documentation of CPU time usage per cycle

Version 3.0

Michael Haring (15.04.2023)

- Implementation of intramolecular trial moves for bond stretching, angle bending and dihedral torsion
- Code optimization (intermolecular interactions are stored as an upper triangular matrix in a square array)
- Option to execute simulation from the command line using 'WolframScript'
- Custom parameter for documentation of intramolecular configuration distribution
- Custom parameter for documentation of RDF per atom and/or molecule center pair
- Custom parameter to specify different seeds for the random number generator
- Correction of the intramolecular non-bonded interactions further than three bonds apart

Version 3.1

Michael Haring (21.11.2023)

- Custom parameter to specify whether the Jacobian should be considered in the acceptance rule for intramolecular trial moves
- Changed configuration distribution sampling (at the end of a cycle)

III. Reference Tables

III.1. List of Figures

4.4.	Schematic illustration of the exchange step for highly asymmetric mixtures.[29] The blue and green spheres represent different species.	25
4.5.	Schematic illustration of the molecule exchange and molecule swap trial moves of the GECFC method.[36] The green spheres represent full molecules and the blue sphere represents the fractional.	26
4.6.	Scaled Lennard-Jones potential function for different values of λ	27
4.7.	Simulation results of argon simulated as a LJ fluid at different LJ-reduced temperatures using the new algorithm for adjusting trial move limits. (a) Course of the calibrated maximum volume change; (b) Comparison of the final value with the total volume. In the equilibrated state, about 100 molecules are in the vapor and 400 molecules in the liquid phase.	33
4.8.	(a) Course of the calibrated number of attempted transfers in the equilibration phase. (b) Comparison to values stated in literature.[28] The data points and error bars indicate the mean and single standard deviation of the last 500 cycles.	34
5.4.	Proposed mole fractions of $x_1 = 0.5303 \text{ mol mol}^{-1}$, $y_1 = 0.8256 \text{ mol mol}^{-1}$ for a simulation of the two-component system argon (1) + krypton (2) at 143.15 K and 20 bar.	49
5.20.	Configuring TexMaker	103
6.1.	Simulation results and standard deviation of argon using the guidelines for pure LJ fluids. The units are based on the LJ norm. (----) Thol EoS; ● critical point	108
6.3.	Simulation results and standard deviations of ethane using the guidelines for rigid molecules. (----) Friend 1991 EoS [10], ○ simulation results, ● critical point	117

- 6.4. Simulation results and standard deviations of the binary mixture ethane - methane using the guidelines for rigid molecules. The lines indicate the estimation from the SRK EoS with $k_{ij} = -0.01$. [22] SRK EoS at (----) $T = 160$ K, (-----) $T = 210$ K, (-----) $T = 260$ K; simulation results at $\circ T = 160$ K, $\triangle T = 210$ K and $\diamond T = 260$ K 118
- 6.5. Comparison of the simulation course of argon as a LJ fluid at $T^* = 0.75$.
— GEMC algorithm, — unbiased CFC Pou., — biased CFC Pou. . . 120
- 6.6. Comparison of the acceptance rate of the bigger component 1 and the smaller component 2. Legend: — GEMC, ---- GECFC Pou. ; \circ M1, \triangle M2, \diamond M3 123
- 6.7. Comparison of the investigated methods for argon as a LJ fluid covering the whole temperature range. Legend: \circ GEMC, \triangle GECFC Shi, \diamond GECFC Pou. 124

III.II. List of Tables

5.1. Overview of tasks executed in every phase including the execution of <u>T</u> ranslations, <u>R</u> otations, <u>V</u> olume changes, <u>I</u> nsertions and <u>W</u> idom insertions	76
5.2. Description and units of symbols used in Equation 5.4.1	80
5.3. Conversion factors used in Equation 5.4.1	81
5.4. Description and units of symbols used in Equations 5.4.2, 5.4.3 and 5.4.4	81
5.5. Description and units of symbols used in Equation 5.4.5	82
5.6. Description and units of symbols used in Equation 5.4.6	82
5.7. Conversion factors used in Equation 5.4.6	83
5.8. Description and units of symbols used in Equation 5.4.7 and 5.4.8	84
5.9. Conversion factors used in Equation 5.4.7	84
5.10. Conversion factors used in Equation 5.4.8	84
5.11. Overview of the packages (<u>A</u> nalysis, <u>C</u> hemical Thermodynamics, <u>F</u> orceFieldOPLS, <u>M</u> olecularSampling, <u>P</u> ackages, <u>P</u> roperty - Package))	90
6.1. Ensemble averages and their relative error to the Thol EoS [46] from Figure 6.1109	
6.2. Initial settings for binary mixtures of LJ fluids at different LJ reduced temperatures	113
6.3. Initial conditions for simulations of binary LJ mixtures (M1, M2, M3). σ_{22}/σ_{11} and $\epsilon_{22}/\epsilon_{11}$ are the ratios of the size and energy parameters of the potential wells of the two components, respectively. T^* is the LJ reduced temperature and P^* is the LJ reduced pressure.	122
6.4. Parameters and geometries for methane and carbon dioxide as a partially charged rigid molecules. σ is the size and ϵ is the energy parameter of the LJ potential, q is the electrostatic charge parameter, l is the bond length, and ϕ is the bond angle.	130

Bibliography

- [1] Allen, M.P., Tildesley, D.J., Tildesley, D.J., 2017: *Computer Simulation of Liquids*, Oxford science publications, Oxford University Press, URL: <https://books.google.at/books?id=nlExDwAAQBAJ>.
- [2] Binder, K., Ceperley, D.M., Hansen, J.P., Kalos, M., Landau, D., Levesque, D., Mueller-Krumbhaar, H., Stauffer, D., Weis, J.J., 2012: *Monte Carlo methods in statistical physics*, Band 7, Springer Science & Business Media.
- [3] Bondi, A., 1964: *van der Waals Volumes and Radii*, The Journal of Physical Chemistry, 68(3), 441–451, URL: <https://doi.org/10.1021/j100785a001>.
- [4] Chen, B., Martin, M.G., Siepmann, J.I., 1998: *Thermodynamic properties of the williams, opls-aa, and mmff94 all-atom force fields for normal alkanes*, The Journal of Physical Chemistry B, 102(14), 2578–2586.
- [5] Cortés Morales, A.D., Economou, I.G., Peters, C.J., Ilja Siepmann, J., 2013: *Influence of simulation protocols on the efficiency of Gibbs ensemble Monte Carlo simulations*, Molecular Simulation, 39(14-15), 1135–1142.
- [6] Di Pierro, M., Elber, R., Leimkuhler, B., 2015: *A stochastic algorithm for the isobaric–isothermal ensemble with Ewald summations for all long range forces*, Journal of chemical theory and computation, 11(12), 5624–5637.
- [7] Errington, J.R., Boulougouris, G.C., Economou, I.G., Panagiotopoulos, A.Z., Theodorou, D.N., 1998: *Molecular simulation of phase equilibria for water- methane and water- ethane mixtures*, The Journal of Physical Chemistry B, 102(44), 8865–8873.
- [8] Ewald, P.P., 1921: *Die Berechnung optischer und elektrostatischer Gitterpotentiale*, Annalen der Physik, 369(3), 253–287, URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19213690304>.

- [9] Frenkel, D., Smit, B., 2002: *Understanding molecular simulation*, Nr. AR-RAY(0x55ed32a6d0d8) in Computational science series, Academic Press, San Diego [u.a.], 2. ed.. Auflage.
- [10] Friend, D.G., Ingham, H., Fly, J.F., 1991: *Thermophysical properties of ethane*, Journal of physical and chemical reference data, 20(2), 275–347.
- [11] Gowers, T., Barrow-Green, J., Leader, I., 2008: *The Princeton companion to mathematics*, Princeton University Press.
- [12] Haring, M., 2021: *Parametrization of Gibbs Ensemble Monte Carlo Simulations*, TU Graz.
- [13] Haring, M., 2022: *Implementation of Ewald and Wolf Summations for a GEMC algorithm*, TU Graz.
- [14] Haring, M., 2023: *Intramolecular Trial Moves in GEMC Simulations*, TU Graz.
- [15] Harismiadis, V.I., Panagiotopoulos, A.Z., Tassios, D.P., 1994: *Phase equilibria of binary Lennard-Jones mixtures with cubic equations of state*, Fluid Phase Equilibria, 94, 1–18.
- [16] Harismiadis, V., Koutras, N., Tassios, D., Panagiotopoulos, A.Z., 1991: *How good is conformal solutions theory for phase equilibrium predictions?: Gibbs ensemble simulations of binary Lennard-Jones mixtures*, Fluid Phase Equilibria, 65, 1–18.
- [17] Hens, R., Vlugt, T.J., 2017: *Molecular simulation of vapor–liquid equilibria using the Wolf method for electrostatic interactions*, Journal of Chemical & Engineering Data, 63(4), 1096–1102.
- [18] Herce, H.D., Garcia, A.E., Darden, T., 2007: *The electrostatic surface term:(I) periodic systems*, The Journal of chemical physics, 126(12), 124106.
- [19] Jablonka, K.M., Ongari, D., Smit, B., 2019: *Applicability of Tail Corrections in the Molecular Simulations of Porous Materials*, Journal of Chemical Theory and Computation, 15(10), 5635–5641, URL: <https://doi.org/10.1021/acs.jctc.9b00586>, pMID: 31442035.
- [20] Jorgensen, W.L., Maxwell, D.S., Tirado-Rives, J., 1996: *Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of*

- Organic Liquids*, Journal of the American Chemical Society, 118(45), 11225–11236, URL: <https://doi.org/10.1021/ja9621760>.
- [21] Kalos, M.H., Whitlock, P.A., 2009: *Monte carlo methods*, John Wiley & Sons.
 - [22] Knapp, H., Döring, R., Oellrich, L., Plöcker, U., Prausnitz, J.M., Langhorst, R., Zeck, S., 1982: *Vapor-liquid equilibria for mixtures of low boiling substances. Pt. 1. Binary systems*, Band 6, DECHEMA.
 - [23] Lorentz, H.A., 1881: *Ueber die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase*, Annalen der physik, 248(1), 127–136.
 - [24] Mayr, N., 2022: *Assessment of Methods to Improve Particle Swap Rates in GEMC*, TU Graz.
 - [25] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953: *Equation of State Calculations by Fast Computing Machines*, The Journal of Chemical Physics, 21(6), 1087–1092, URL: <https://doi.org/10.1063/1.1699114>.
 - [26] de Miguel, E., Jackson, G., 2006: *The nature of the calculation of the pressure in molecular simulations of continuous models from volume perturbations*, The Journal of Chemical Physics, 125(16), 164109, URL: <https://doi.org/10.1063/1.2363381>.
 - [27] Nicolas, J., Gubbins, K., Streett, W., Tildesley, D., 1979: *Equation of state for the Lennard-Jones fluid*, Molecular Physics, 37(5), 1429–1454.
 - [28] Panagiotopoulos, A.Z., 1987: *Direct determination of phase coexistence properties of fluids by Monte Carlo simulation in a new ensemble*, Molecular Physics, 61(4), 813–826, URL: <https://doi.org/10.1080/00268978700101491>.
 - [29] Panagiotopoulos, A.Z., 1989: *Exact calculations of fluid-phase equilibria by Monte Carlo simulation in a new statistical ensemble*, International Journal of Thermophysics, 10(2), 447–457, URL: <https://doi.org/10.1007/BF01133541>.
 - [30] Panagiotopoulos, A.Z., 1999: *Monte Carlo methods for phase equilibria of fluids*, Journal of Physics: Condensed Matter, 12(3), R25–R52, URL: <https://doi.org/10.1088/0953-8984/12/3/R25>.

- [31] Panagiotopoulos, A.Z., Quirke, N., Stapleton, M., Tildesley, D.J., 1988: *Phase equilibria by simulation in the Gibbs ensemble*, Molecular Physics, 63(4), 527–545, URL: <https://doi.org/10.1080/00268978800100361>.
- [32] Panagiotopoulos, A.Z., Stapleton, M.R., 1989: *The gibbs method for molecular-based computer simulations of phase equilibria*, Fluid Phase Equilibria, 53, 133–141, URL: <http://www.sciencedirect.com/science/article/pii/0378381289800809>, proceedings of the Fifth International Conference.
- [33] Panagiotopoulos, A.Z., Suter, U.W., Reid, R.C., 1986: *Phase diagrams of nonideal fluid mixtures from Monte Carlo simulation*, Ind. Eng. Chem. Fundam., 25, 525–535.
- [34] Panagiotopoulos, A.Z., 1992: *Direct determination of fluid phase equilibria by simulation in the Gibbs ensemble: a review*, Molecular simulation, 9(1), 1–23.
- [35] Plesch, T., 2022: *GEMC Simulation des binären Gemisches Methan + Ethan*, TU Graz.
- [36] Poursaeidesfahani, A., Torres-Knoop, A., Dubbeldam, D., Vlugt, T.J., 2016: *Direct free energy calculation in the Continuous Fractional Component Gibbs ensemble*, Journal of chemical theory and computation, 12(4), 1481–1490.
- [37] Raabe, G., 2017: *Molecular Simulation Studies on Thermophysical Properties*, With Application to Working Fluids (Molecular Modeling and Simulation).
- [38] Rahbari, A., 2020: *Thermodynamics of Industrially Relevant Systems: Method Development and Applications*, , Ph. D. thesis, Delft University of Technology, Delft, The Netherlands.
- [39] Schwarz, A., 2021: *Parametrization of Gibbs Ensemble Monte Carlo Simulations*, TU Graz.
- [40] Shi, W., Maginn, E.J., 2007: *Continuous fractional component Monte Carlo: an adaptive biasing method for open system atomistic simulations*, Journal of chemical theory and computation, 3(4), 1451–1463.
- [41] Shi, W., Maginn, E.J., 2008: *Improvement in molecule exchange efficiency in Gibbs ensemble Monte Carlo: development and implementation of the continuous fractional component move*, Journal of computational chemistry, 29(15), 2520–2530.

- [42] Silva Fernandes, F.M., Fartaria, R.P., 2015: *Gibbs Ensemble Monte Carlo*, American Journal of Physics, 83(9), 809–816.
- [43] Steiner, T., Thaler, S., Mayr, N., Haring, M., Wallek, T., 2022: *Gibbs ensemble Monte-Carlo (GEMC) simulation algorithm*, URL: <https://notebookarchive.org/2022-11-6ger7tx>.
- [44] Tan, S., Do, D.D., Nicholson, D., 2018: *Molecular simulation of volume of mixing, Helmholtz free energy of mixing and entropy of mixing in bulk fluid mixtures*, Molecular Simulation, 44(16), 1312–1324.
- [45] Thaler, S., Mayr, N., 2021: *Gibbs Ensemble Monte Carlo Simulations with Mathematica*, TU Graz.
- [46] Thol, M., Rutkai, G., Köster, A., Lustig, R., Span, R., Vrabec, J., 2016: *Equation of state for the Lennard-Jones fluid*, Journal of Physical and Chemical Reference Data, 45(2), 023101.
- [47] Vlugt, T., 2021: private communication.
- [48] Vrabec, J., Lotfi, A., Fischer, J., 1995: *Vapour liquid equilibria of Lennard-Jones model mixtures from the NpT plus test particle method*, Fluid phase equilibria, 112(2), 173–197.
- [49] Wang, F., Landau, D.P., 2001: *Efficient, multiple-range random walk algorithm to calculate the density of states*, Physical review letters, 86(10), 2050.
- [50] Weeks, J.D., Chandler, D., Andersen, H.C., 1971: *Role of repulsive forces in determining the equilibrium structure of simple liquids*, The Journal of chemical physics, 54(12), 5237–5247.
- [51] Widom, B., 1963: *Some Topics in the Theory of Fluids*, The Journal of Chemical Physics, 39(11), 2808–2812, URL: <https://doi.org/10.1063/1.1734110>.
- [52] Wolf, D., Keblinski, P., Phillpot, S.R., Eggebrecht, J., 1999: *Exact method for the simulation of Coulombic systems by spherically truncated, pairwise $1/r$ summation*, The Journal of Chemical Physics, 110(17), 8254–8282, URL: <https://doi.org/10.1063/1.478738>.
- [53] Wolfram Research, Inc., : *Mathematica, Version 13.1*, Champaign, IL, 2022.