Adaptive Initiation of AutoPas Tuning Phases for Efficient Particle Simulations

Bachelor's Thesis in Informatics

Niklas Ladurner

Technische Universität München
School of Computation, Information and Technology - Informatics
Chair of Scientific Computing in Computer Science

Munich, September 15th, 2025

This page is intentionally left blank.

Adaptive Initiation of AutoPas Tuning Phases for Efficient Particle Simulations

Adaptives Auslösen von Tuning-Phasen für effiziente Partikelsimulation in AutoPas

Bachelor's Thesis in Informatics

Niklas Ladurner

Supervisor: Univ.-Prof. Dr. Hans-Joachim Bungartz
Chair of Scientific Computing in Computer Science

Advisor: Manish Mishra, M.Sc.
Chair of Scientific Computing in Computer Science

Date: 15.09.2025

Technische Universität München
School of Computation, Information and Technology - Informatics
Chair of Scientific Computing in Computer Science

Munich, September 15th, 2025

This page is intentionally left blank.

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 15.09.2025 Niklas Ladurner

This page is intentionally left blank.

Acknowledgements

I would like to express my deepest gratitude to my advisor Manish Mishra. His guidance, patience, and insightful explanations during our meetings were invaluable in helping me understand AutoPas and its wide array of complex features. His continuous encouragement and his in-depth analysis of preliminary results pushed me to refine my ideas and improve the quality of this thesis.

Additionally, I would like to thank all my friends who kindly took the time to proofread parts of this work and provide thoughtful feedback. Their constructive criticism and suggestions helped shape this thesis into what it is now.

The evaluation of our proposed mechanisms could not have been performed without sizeable processing capabilities. Thus, for supporting this thesis by providing computing time on its Linux-Cluster, I gratefully acknowledge the Leibniz Supercomputing Centre.

Lastly, I want to thank José Areia and all contributors to the *IPLeiria-Thesis*[1] template, which was adapted and used in this work.

[1] https://github.com/joseareia/ipleiria-thesis

This page is intentionally left blank.

Abstract

Particle simulations have become an indispensable tool in research and are used across a wide range of applications. Depending on the specific scenario, different simulation configurations may be more suitable. AutoPas is a particle simulation library that offers a simple black-box interface for researchers. To achieve this, AutoPas reevaluates the optimal configuration at fixed intervals. This thesis proposes a dynamic approach to determine ideal points for the initiation of new tuning phases.

This page is intentionally left blank.

Zusammenfassung

TODO

This page is intentionally left blank.

Contents

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

1 Introduction

This chapter is intended to introduce the main ideas of molecular dynamics simulation. In Section 1.1 ...

1.1 Motivation

1.2 Molecular Dynamics

Molecular Dynamics (MD) simulation is one method of simulating a n-body problem on the atomic level. At that level, the interactions between atoms can be simulated based on Newton's laws of motion. The forces at play are interatomic potentials such as Lennard-Jones or Coloumb potentials.

The main simulation loop consists of calculating the forces between particles or atoms and integrating the equations of motion. These two steps are repeated until an equilibrium is reached, at which point the desired measurements can be taken. [FS02]

Such MD simulations are used in different fields ranging from drug discovery [HD18] to physics and material scieneces.

1.2.1 Newton's laws of motion

As referred to before, Newton's laws of motion are important in the context of MD simulation. ...

The mentioned laws are as follows. [New87; New34]

I. *Every object perseveres in its state of rest, or of uniform motion in a right line, unless it is compelled to change that state by forces impressed thereon.*
   In other words, if net force is zero for any body, its velocity is constant.
II. *The alteration of motion is ever proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed.*
   In other words, $F = m \cdot a$.
III. *To every action there is always opposed an equal reaction: or, the mutual actions of two bodies upon each other are always equal, and directed to contrary parts.*
   In other words, if one body exerts force F_a on another body, than the second body exerts force $F_b = -F_a$ on the first body.

1.2.2 Lennard-Jones Potential

Simulating all pairwise interactions between atoms has complexity $O\left(N^2\right)$. To reduce this complexity, most MD simulations restrict themselves to short-range interactions. As the forces of these interactions are negligible if the interacting particles are far apart, a cutoff-radius can be introduced after which the forces can be assumed to be close to zero, without having to compute them. This significantly reduces the computational complexity, as only the interactions between close neighbors have to be calculated. [Gra+21]

The Lennard-Jones (LJ) potential is one such short-range interaction potential that acts on pairs of particles. It is a sufficiently good approximation such that macroscopic effects can be derived from simulating the interactions at an atomic level. It is most frequently used in the form of the 12-6 potential as defined in (1.1).

$$V_{\mathrm{LJ}}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{1.1}$$

In this equation, r is the distance between the two particles, ε the interaction strength and σ the distance at which the potential signs change (zero-crossing). Parameters ε, σ are dependent on the simulation context, e.g. the material which ought to be simulated. The potential function is illustrated in Figure 1.1. [Wan+20; LSH24]

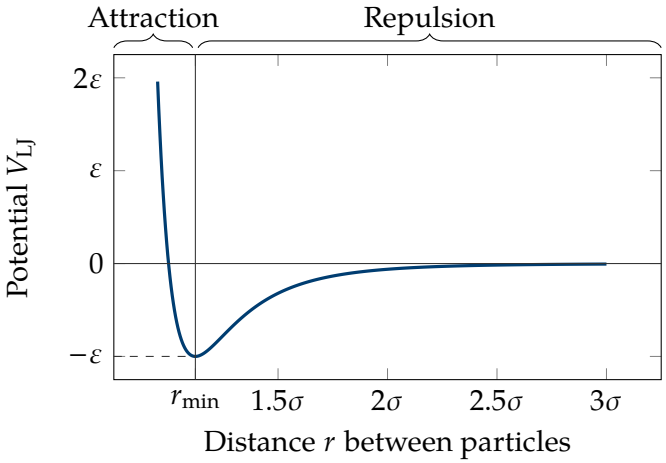


Figure 1.1: An illustration of the potential well of the 12-6 LJ-Potential, with the minimum of $-\varepsilon$ at $r_{\min} = \sigma \sqrt[6]{2}$. The figure is based on Lenhard et al. [LSH24]

1.2.3 Störmer-Verlet Algorithm

Using LJ potentials and Newton's laws of motion, we can construct a system of ordinary differential equations. To solve them analytically is practically infeasible for large systems, therefore numeric solvers have to be used in approximating a solution. The Störmer-Verlet algorithm is one such numeric approach for solving the systems derived by Newton's laws of motion. With $\mathbf{p}_i(t)$, $\mathbf{v}_i(t)$, $a_i(t)$, m_i, $\mathbf{F}_i(t)$ as the position, velocity, acceleration, mass and force acting on a particle i at time t, we can derive the algorithm by the summation of Taylor expansions. First, we deduce the position of particle i at time $t + \delta t$, as in (1.2). Secondly, we make a backwards step to $t - \delta t$, as in (1.3).

$$\mathbf{p}_i(t + \delta t) = \mathbf{p}_i(t) + \delta t \dot{\mathbf{p}}_i(t) + \frac{1}{2}\delta t^2 \ddot{\mathbf{p}}_i(t) + \frac{1}{6}\delta t^3 \dddot{\mathbf{p}}_i(t) + O(\delta t^4) \tag{1.2}$$

$$\mathbf{p}_i(t-\delta t) = \mathbf{p}_i(t) - \delta t\dot{\mathbf{p}}_i(t) + \frac{1}{2}\delta t^2\ddot{\mathbf{p}}_i(t) - \frac{1}{6}\delta t^3\dddot{\mathbf{p}}_i(t) + O(\delta t^4) \tag{1.3}$$

By adding both (1.2) and (1.3) and reordering terms, we conclude (1.4).

$$\mathbf{p}_i(t+\delta t) = 2\mathbf{p}_i(t) - \mathbf{p}_i(t-\delta t) + \delta t^2\ddot{\mathbf{p}}_i(t) + O(\delta t^4) \tag{1.4}$$

As the second derivative of the position $\mathbf{p}_i(t)$ is the acceleration $\mathbf{a}_i(t)$, we can express (1.4) as (1.5). Where, by Newton's second law, $\mathbf{a}_i(t) = \frac{\mathbf{F}_i(t)}{m_i}$.

$$\mathbf{p}_i(t+\delta t) = 2\mathbf{p}_i(t) - \mathbf{p}_i(t-\delta t) + \delta t^2\mathbf{a}_i(t) + O(\delta t^4) \tag{1.5}$$

A more exact approach is the so-called Velocity-Verlet-Algorithm, ...
[FS13] [LR05]
[HLW03]

2 AutoPas

In this chapter, some key concepts of AutoPas are introduced. In particular, the auto-tuning system.

2.1 Background

2.2 Functors

2.3 Containers

2.4 Traversals

2.5 Simulation Loop

2.6 Data Layout

2.7 Tuning strategies

3 Implementation

To dynamically initiate new tuning phases, a strategy must be found such that they can be triggered at runtime on live simulation data. Depending on the scenario and available statistics provided by the simulation, different methods of finding these trigger points may be optimal. In this chapter therefore, the strategies investigated are presented.

3.1 Considerations

3.1.1 Computational overhead

Our trigger strategies introduce additional computations, as we have to make decisions based on data we can only collect at runtime. Therefore, we must keep this overhead as small as possible, otherwise gains made by retuning less often might easily be dwarfed by expensive computations. Some optimizations that were used in our implementation are as follows: ...

3.1.2 Available simulation statistics

3.1.3 Interaction with tuning strategies

3.2 Time-based Triggers

The simplest approach in detecting whether the current configuration might become suboptimal is to observe changes in iteration runtime. As a specific configuration becomes less suitable as the simulation state changes, one would expect the runtime to increase, as e.g. suboptimal containers lead to unfavorable access patterns. Therefore, the primary focus of this thesis lies on runtime-based strategies in finding trigger points.

3.2.1 Simple Trigger

Regarding the iteration runtime analysis mentioned, the naive strategy compares the runtime of only two iterations: The current one and the previous one. The ratio at

which a new tuning phase is triggered, is set by the user via the `triggerFactor` configuration variable, henceforth denoted as λ. In other words, if $t_i \geq \lambda \cdot t_{i-1}$, a new tuning phase is triggered. This is implemented as the `TimeBasedSimpleTrigger`.

3.2.2 Single-iteration averaging Trigger

The simple strategy descriped in Section 3.2.1 is quite unstable. Because of hardware heterogeneity, the iteration runtimes may have .....

The `TimeBasedAverage` method is different from the `TimeBasedSimple` trigger in that the first compares to the moving average of the last n runtime samples. The formula is provided in (3.1).

$$t_i \geq \frac{\lambda}{n} \cdot \sum_{k=i-n-1}^{i-1} t_k \tag{3.1}$$

3.2.3 Interval averaging Trigger

If we have scenario changes that happen gradually, the runtime might not increase drastically in a single iteration, but rather across a series of iterations. As the previous two triggers only compare to the current iteration's runtime, they might be suboptimal in such experiments. Therefore, triggers that take this effect into account are needed. One such approach is the `TimeBasedSplitTrigger`. This strategy splits the measurements of the last n iterations and the current iteration into two intervals A, B as in (3.2), and compares whether $\text{avg}(B) \geq \lambda \cdot \text{avg}(A)$.

$$A = \left[t_{i-n}, t_{i-j}\right], \quad B = \left[t_{i-j+1}, t_i\right], \quad j = \left\lfloor \frac{n}{2} \right\rfloor \tag{3.2}$$

3.2.4 Linear Regression Trigger

This approach is conceptually similar to the interval averaging trigger, although with one major difference. Instead of comparing the current interval of runtimes to a previous one, the comparison is based on an estimate of the runtime in the next interval based on data of the current interval.

Figure 3.1: An overview of the `TimeBasedSplitTrigger` and `TimeBasedRegression` trigger intervals.

The general idea is to fit a simple linear regression, adapted to our use case, on the last n runtime samples. Using the linear regression we obtain a slope estimator $\hat{\beta}_1$, by which we can predict the runtime in the next interval. In the following, t_k is the runtime at iteration k, i the current iteration and $\bar{t}$, $\bar{k}$ the average runtime and iteration respectively. Then the slope estimator $\hat{\beta}_1$ in the standard simple linear regression model is presented in (3.3).

$$\hat{\beta}_1 = \frac{\sum_{k=i-n-1}^{i}(k-\bar{k})(t_k-\bar{t})}{\sum_{k=i-n-1}^{i}(k-\bar{k})^2}, \quad \bar{t} = \frac{1}{n}\sum_{k=i-n-1}^{i} t_k, \quad \bar{k} = \frac{1}{n}\sum_{k=i-n-1}^{i} k \tag{3.3}$$

The value of the estimator $\hat{\beta}_0$, i.e. the intersection at $y = 0$, is not of interest. Similarly, as the samples are taken in constant steps of one iteration, the values of k can

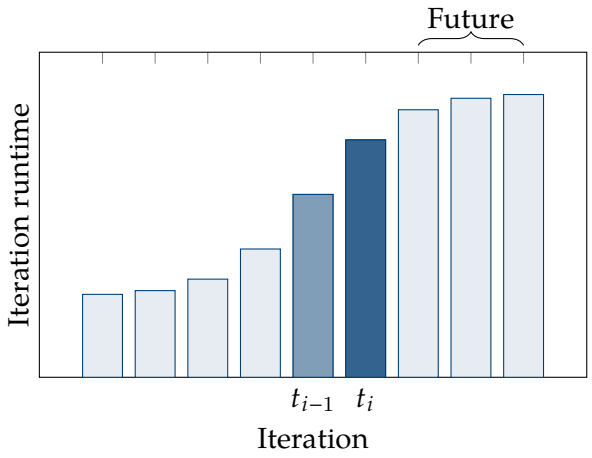


Figure 3.2: asdf

be shifted to the interval $[0, n]$. Considering these two points, the model can be transformed to (3.4), where A, B are constants that can be precomputed at initialization, as given in (3.5).

$$\hat{\beta}_1' = \frac{\sum_{k=0}^{n-1} \left(k - \frac{n(n-1)}{2n}\right)\left(t_{i-n-1+k} - \bar{t}\right)}{\sum_{k=0}^{n-1} \left(k - \frac{n(n-1)}{2n}\right)^2} = \frac{1}{B}\sum_{k=0}^{n-1}(k-A)\left(t_{i-n-1+k} - \bar{t}\right) \tag{3.4}$$

$$A = \frac{n-1}{2}, \quad B = \sum_{k=0}^{n-1}(k-A)^2 \tag{3.5}$$

$\hat{\beta}_1'$ can thus be interpreted as "in each iteration, the runtime is projected to increase $\hat{\beta}_1'$ nanoseconds". This however, is not a sensible metric to compare to a user-set triggerFactor, as it heavily depends on the scenario and would require to know rough iteration runtime estimates beforehand. Therefore, we use a normalization function, such that a factor of 1.0 is roughly equal to "no runtime increase". The resulting value is also consistent to other triggers. The normalization implemented is presented in (3.6).

$$\hat{\beta}_{\text{norm}} = \frac{n \cdot \hat{\beta}_1'}{\bar{t}} \tag{3.6}$$

In particular, we have:

(i) $\hat{\beta}_{\text{norm}} = 1$ if there is no projected change in iteration runtime
(ii) $\hat{\beta}_{\text{norm}} > 1$ if there is a projected increase in iteration runtime
(iii) $\hat{\beta}_{\text{norm}} < 1$ if there is a projected decrease in iteration runtime
(iv) $\hat{\beta}_{\text{norm}} = 2$ if there the runtime of the next interval is projected to be double the current interval's runtime.

3.3 Hybrid Triggers

As will be discussed later, time-based approaches are not suitable for all scenarios. In these scenarios, iteration runtimes alone might not be a good enough indicator for scenario change. As AutoPas provides additional live simulation statistics through its `liveinfo` interface, these can be used in combination with iteration runtimes to find better strategies in detecting scenario change.

4 Evaluation

This chapter presents the scenarios and criteria employed in the evaluation of our implementation. Section 4.1 introduces a series of benchmarking scenarios, which have been chosen to reflect distinct simulation characteristics appearing in real-world applications.
Subsequently, Section 4.2 defines the evaluation metrics applied to these benchmarks. The metrics are intended to provide comparability between simulation runs with dynamic tuning intervals and to the baseline runs with static tuning intervals.
Together, the benchmarking scenarios and evaluation metrics provide a framework for assessing the performance and reliability of the proposed strategies.

4.1 Benchmarking Scenarios

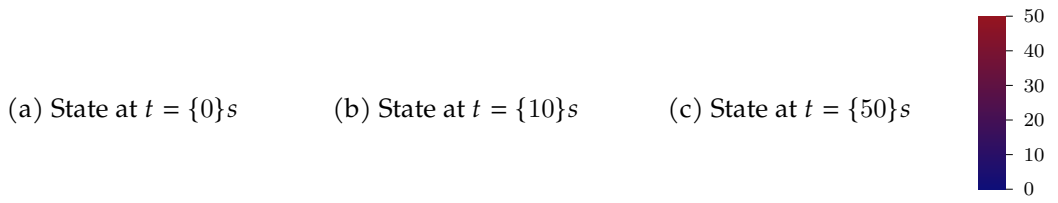
As to not limit our analysis to one specific simulation setting, we use a selection of benchmarking scenarios. These represent different structures as they may be used in real-world applications. The heating-sphere and exploding-liquid scenarios are identical to the ones given by Newcome et al., the configuration files have been adapted and parametrized for use in this thesis [New+25]. The other scenarios are are taken from the AutoPas `md-flexible` example.

4.1.1 Equilibrium

In the equilibrium scenario, particles with initial velocity 0 are packed tightly into a cube with periodic boundary conditions. The particles interact with each other and the grid structure loosens up, but ultimately an equilibrium is reached in which no rapid changes in velocity occur anymore. After some initial relaxation of the grid structure, there is no further scenario change expected. Therefore, no additional tuning phases should be needed, as the optimal configuration is not expected to change.

4.1.2 Exploding Liquid

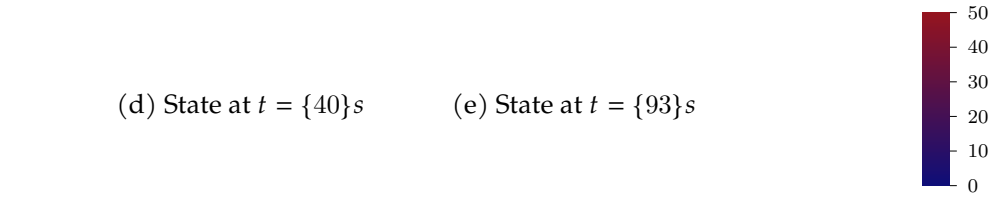
Similarly to the equilibrium scenario, the exploding liquid scenario starts of with the particles packed into a cuboid with periodic boundaries. The cuboid explodes in y-direction, collides with the boundary and finally settles into an equilibrium state spread out over the whole simulation space. If a single AutoTuner instance is used for the



(a) State at $t = \{0\}s$ (b) State at $t = \{10\}s$ (c) State at $t = \{50\}s$

Figure 4.1: Evolution of the simulation state in the equilibrium scenario.

whole domain, the rapid changes in particle positions and heterogeneous particle distribution make finding an optimal configuration very hard. However, if the domain is split up into multiple independent AutoPas instances on different MPI nodes, each AutoTuning instance can independently find an optimal configuration for its part of the domain. By this, the simulation domain can be split up into regions that have been affected by the "explosion" and regions that no particles have entered yet.



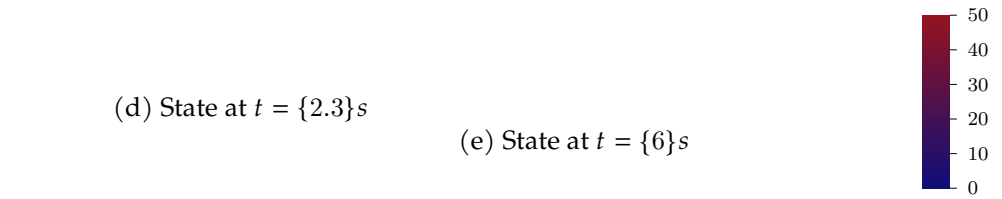
(a) State at $t = \{0\}s$ (b) State at $t = \{5.5\}s$ (c) State at $t = \{20\}s$

(d) State at $t = \{40\}s$ (e) State at $t = \{93\}s$

Figure 4.2: Evolution of the simulation state in the exploding liquid scenario.

4.1.3 Heating Sphere

The heating sphere scenario consists of a dense, small sphere of particles. Reflective boundary conditions apply as in the previous scenarios. In the course of the simulation, the temperature rises from 0.1 to 100 with a $\Delta T = 0.1$ every 100 iterations. Additionaly, brownian motion, i.e. random fluctuations in particle positions are applied [MP10]. The sphere expands with the inrcease in temperature and particles are radiating out from the sphere center.



(a) State at $t = \{0\}s$ (b) State at $t = \{0.40\}s$ (c) State at $t = \{1.2\}s$

(d) State at $t = \{2.3\}s$ (e) State at $t = \{6\}s$

Figure 4.3: Evolution of the simulation state in the heating sphere scenario.

4.1.4 Falling Drop

4.1.5 Spinodial Decomposition

4.2 Evaluation Metrics

To compare results between dynamic and static tuning intervals, different metrics can be used. Firstly, the primary goal is to reduce the total simulation runtime for a range of typical scenarios. As tuning phases spend time in quite suboptimal configurations, a reduction in total runtime is the expected result if our approach reduces the number of tuning phases without spending too many iterations using a suboptimal configuration outside tuning phases.

The metric of total runtime is not particularly fine-grained however, as it only takes into account entire simulation runs. To achieve a more detailed benchmark, we also consider the number of iterations that were running on an optimal configuration. As an approximation to the optimal configuration per iteration we use simulation run with static tuning, a high number of tuning samples and a short tuning interval. Based on this static data we can then rank the configuration our dynamic run chose in terms of "optimality".

5 Results

...

5.1 Trigger Parameters

All presented trigger strategies are based on a trigger factor λ. The averaging and regression triggers additionally take into account the number of samples to inspect, denoted as n. For any dynamic tuning trigger to be useful, sensible default values for these parameters are needed, as the performance of the whole simulation is dependent on the trigger's behavior.

Therefore, we first inspect the relation between these parameters and the total simulation runtime for a range of combinations to find parameter defaults for further evaluation.

5.2 Trigger Behavior

The blue bars in the graphs represent the runtime of that particular iteration. In the configuration plots, the colored background identifies the used configuration: same configurations map to the same color. The gaps in the plot are where tuning iterations have been logged – as their runtime is not relevant for the scenario change and would distort the actual runtime plot, they are not reported here. The red vertical lines indicate the start of a tuning phase.

6 Conclusion

This chapter will ...

6.1 Dynamic Initiation of Tuning Intervals

6.2 Future Work

This page is intentionally left blank.

Bibliography

[FS02] Daan Frenkel and Berend Smit. "Chapter 4 - Molecular Dynamics Simulations". In: *Understanding Molecular Simulation (Second Edition)*. Ed. by Daan Frenkel and Berend Smit. Second Edition. San Diego: Academic Press, 2002, pp. 63–107. ISBN: 978-0-12-267351-1. DOI: https://doi.org/10.1016/B978-012267351-1/50006-7. URL: https://www.sciencedirect.com/science/article/pii/B9780122673511500067.

[FS13] Alexander Fulst and Christian Schwermann. *Molekulardynamiksimulation*. Accessed: 2025-08-23. 2013. URL: https://www.uni-muenster.de/Physik.TP/archive/fileadmin/lehre/TheorieAKkM/ws13/Fulst-Schwermann.pdf.

[Gra+21] Fabio Alexander Gratl et al. "N Ways to Simulate Short-Range Particle Systems: Automated Algorithm Selection with the Node-Level Library AutoPas". en. In: *Computer Physics Communications* 273 (2021), p. 108262. DOI: 10.1016/j.cpc.2021.108262. URL: https://www.researchgate.net/publication/357143093_N_Ways_to_Simulate_Short-Range_Particle_Systems_Automated_Algorithm_Selection_with_the_Node-Level_Library_AutoPas.

[HLW03] Ernst Hairer, Christian Lubich, and Gerhard Wanner. "Geometric numerical integration illustrated by the Störmer–Verlet method". In: *Acta Numerica* 12 (2003), pp. 399–450. DOI: 10.1017/S0962492902000144. URL: https://www.math.kit.edu/ianm3/lehre/geonumint2009s/media/gni_by_stoermer-verlet.pdf.

[HD18] Scott A. Hollingsworth and Ron O. Dror. "Molecular Dynamics Simulation for All". In: *Neuron* 99.6 (2018), pp. 1129–1143. DOI: 10.1016/j.neuron.2018.08.011.

[LR05] Benedict Leimkuhler and Sebastian Reich. "Geometric integrators". In: *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005, pp. 70–104.

[LSH24] Johannes Lenhard, Simon Stephan, and Hans Hasse. "On the History of the Lennard-Jones Potential". In: *Annalen der Physik* 536.6 (2024), p. 2400115. DOI: https://doi.org/10.1002/andp.202400115. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.202400115. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.202400115.

[MP10] Peter Mörters and Yuval Peres. *Brownian motion*. Vol. 30. Cambridge University Press, 2010.

[New+25] Samuel James Newcome et al. "Algorithm Selection in Short-Range Molecular Dynamics Simulations". In: (May 2025). DOI: 10.48550/ARXIV.2505.03438. arXiv: 2505.03438 [cs.CE].

[New87] Isaac Newton. *Philosophiæ Naturalis Principia Mathematica*. London: Royal Society, 1687.

[New34] Isaac Newton. *Mathematical Principles of Natural Philosophy*. Ed. by Florian Cajori. Trans. by Andrew Motte. First English translation 1729; revised edition. Berkeley: University of California Press, 1934.

[Wan+20] Xipeng Wang et al. "The Lennard-Jones potential: when (not) to use it". In: *Phys. Chem. Chem. Phys.* 22 (19 2020), pp. 10624–10633. DOI: 10.1039/C9CP05445F. URL: http://dx.doi.org/10.1039/C9CP05445F.

This page is intentionally left blank.