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, September 15th, 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 introduces some fundamental concepts necessary to understand Molecular Dynamics (MD) simulations. We begin with a discussion of the motivation behind the general n-body problem and the goals of this thesis in Section 1.1.
Afterwards, the components of a simple MD simulation loop are presented in Section 1.2. These include Newton's laws of motion for providing the governing equations of particle trajectories, the Lennard-Jones potential as a model for pairwise interactions, and the Störmer-Verlet algorithms as a numerical scheme for integrating the equations of motion.

1.1 Motivation

The n-body problem is a foundational challenge of classical physics. It concerns the interaction and movement of bodies, like the trajectories of masses in the solar system. At such astronomic scales, general relativity additionally introduces a high degree of complexity. Yet, even in classical Newtonian physics, the systems of equations tend to no longer be solvable by analytic means if $n > 2$ bodies are involved, except for certain special cases. Hence, numerical algorithms have become essential in finding approximate solutions. [1]

With the advent of computer-based simulation, the feasibility of numerical solutions to the n-body problem has increased significantly. Over the past few decades, advances in high performance computing (HPC) have further increased both scale and efficiency of such simulations, allowing for the modeling of large systems at unprecedented resolution. For instance, the TianNu project simulated 2.97×10^{12} particles in 2017 on the Tianhe-2 supercomputer [3]. These days, particle simulations have established themselves as indispensable tools across a wide range of scientific fields. Applications reach from drug discovery [10] to plasma physics [17] and materials science [16].

One example of a software framework enabling n-body simulations is the AutoPas library [6]. Its internal mechanisms will be discussed in detail in Chapter 2; for the motivation of this thesis it suffices to know, that AutoPas seeks to dynamically select optimal simulation configurations without requiring expert knowledge during setup ("autotuning").

To achieve this, so-called tuning phases are initiated at fixed intervals. During each tuning phase, different configurations are sampled for a predetermined number of it-

erations, after which the best performing configuration is selected to simulate the remaining iterations until the next tuning phase. Naturally, these static intervals do not necessarily align with the points at which it would be most advantageous to switch configurations. Consider a scenario, in which the optimal configuration changes rapidly in the beginning, but stabilizes and settles into an equilibrium later on. Having one uniform static interval, it would be either too short – resulting in unnecessary tuning phases during equilibrium, or too long – resulting in suboptimal performance in the early phase.

This thesis proposes a method to resolve this problem by dynamically initiating tuning phases based on live simulation data.

1.2 Molecular Dynamics

Molecular Dynamics (MD) simulation is one method of solving the classical n-body problem on the molecular level. At that level, the interactions between atoms are subject to the of laws quantum mechanics, in particular the Schrödinger equation. That equation however is unsuitable for the simulation of larger systems due to its complexity by nature of being a partial differential equation.

Therefore, simplifications such as the Born-Oppenheimer approximation have to be employed. This approximation is based on the fact that the nuclei of atoms have much greater mass than the electrons surrounding them. Under the additional assumption, that the nuclei can be considered static relative to the movements of the electrons, we can separate the Schrödinger equation into two parts coupled by an interaction potential. Using further simplifications, we obtain (1.1), which directly corresponds to the classical laws of motion as stated by Newton (cf. Section 1.2.1). In this equation, $\mathbf{p}_i(t), \mathbf{a}_i(t), m_i, V(\mathbf{p}_i(t))$ are the position, acceleration, mass and potential acting on a particle i at time t. [2, 8, 18]

$$m_i \mathbf{a}_i(t) = -\nabla_{\mathbf{p}_i} V(\mathbf{p}(t)) \tag{1.1}$$

The simplest interatomic potentials one could apply here describe the interactions between only two particles, such as the Gravitational, Lennard-Jones or Coloumb potentials. [8]

Considering the formula presented in (1.1), the main MD simulation loop is rather simple. It 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. [4]

1.2.1 Newton's laws of motion

As referred to before, Newton's laws of motion can be applied to MD simulation in approximating particle behavior. These well-known laws of classical mechanics are as follows. [15]

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 the net force on any body is zero, 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.

The second law is of utmost importance, as it allows to compute trajectories of particles based on the force acting upon them. The third law, while secondary in dynamics, is useful especially regarding optimizations: for pairwise interactions, the force needs to be evaluated only once, since the second particle experiences a force of the same magnitude in opposed direction. [7]

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. This significantly reduces the computational complexity, as only the interactions between close neighbors have to be computed. [7]

The Lennard-Jones (LJ) potential is one such short-range interaction potential that acts on pairs of particles. It is based on empirical data and 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 (1.2).

$$V_{\mathrm{LJ}}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{1.2}$$

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. [12, 19]

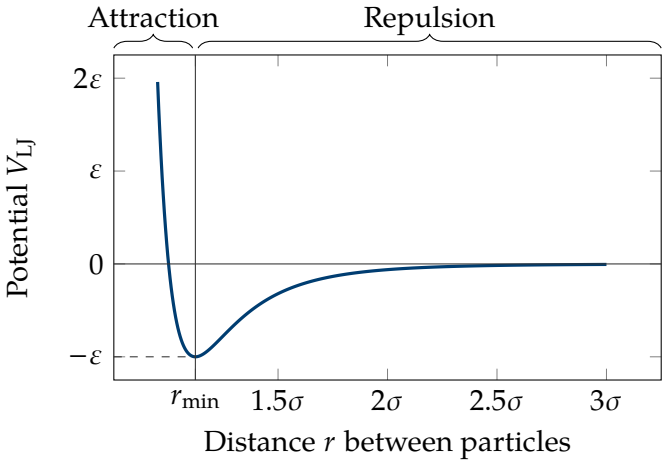


Figure 1.1: An illustration of the potential well of the 12-6 LJ potential, with the minimum of $-\varepsilon$ at $r_{\min} = \sigma\sqrt[6]{2}$ and zero-crossing at σ. The figure is based on Lenhard et al. [12]

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, as stated before.

The Störmer-Verlet algorithm is one such numeric technique for solving the systems constructed as mentioned. With $\mathbf{p}_i(t), \mathbf{v}_i(t), \mathbf{a}_i(t), m_i, \mathbf{F}_i(t)$ 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 (1.3). Secondly, we take a backwards step to $t - \delta t$, as in (1.4).

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

$$\mathbf{p}_i(t - \delta t) = \mathbf{p}_i(t) - \delta t\dot{\mathbf{p}}_i(t) + \frac{1}{2}\delta t^2\ddot{\mathbf{p}}_i(t) - \frac{1}{6}\delta t^3\dddot{\mathbf{p}}_i(t) + O(\delta t^4) \tag{1.4}$$

By adding both (1.3) and (1.4) and reordering terms, we conclude (1.5).

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

As the second derivative of the position $\mathbf{p}_i(t)$ is the acceleration $\mathbf{a}_i(t)$, we can express (1.5) as (1.6). Where, by Newton's second law, $\mathbf{a}_i(t) = \frac{\mathbf{F}_i(t)}{m_i}$.

$$\mathbf{p}_i(t + \delta t) = 2\mathbf{p}_i(t) - \mathbf{p}_i(t - \delta t) + \delta t^2\mathbf{a}_i(t) + O(\delta t^4) \tag{1.6}$$

By this formulation we could already calculate the velocities of the particles, however, there are some drawbacks – e.g. high error propagation. A more exact and efficient approach, sometimes referred to as the Velocity-Verlet algorithm, can be derived similarly. For that, considering (1.7), we can rearrange and substitute into (1.5) to conclude (1.8) and finally (1.9).

$$\mathbf{v}_i(t) = \frac{\mathbf{p}_i(t + \delta t) - \mathbf{p}_i(t - \delta t)}{2\delta t} \rightsquigarrow \mathbf{p}_i(t - \delta t) = \mathbf{p}_i(t + \delta t) - 2\delta t\mathbf{v}_i(t) \tag{1.7}$$

$$\mathbf{p}_i(t + \delta t) = \mathbf{p}_i(t) + \delta t\mathbf{v}_i(t) + \frac{\delta t^2}{2}\mathbf{a}_i(t) + O(\delta t^4) \tag{1.8}$$

$$\mathbf{v}_i(t + \delta t) = \mathbf{v}_i(t) + \frac{\delta t}{2}\left[\mathbf{a}_i(t) + \mathbf{a}_i(t + \delta t)\right] + O(\delta t^4) \tag{1.9}$$

Because of the aforementioned improved properties of this method, it is often preferred in MD simulations. [5, 9, 11]

2 AutoPas

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

2.1 Background

2.2 Configuration Parameters

2.2.1 Containers

2.2.2 Traversals

2.2.3 Simulation Loop

2.2.4 Additional Parameters

2.3 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 that can only be collected at runtime. Therefore, the overhead must be kept as small as possible, otherwise gains made by triggering less tuning phases might easily be dwarfed by expensive computations. Additionally, there might be feedback of our method to itself, as our strategies may affect iteration runtime which in turn alters the trigger behavior.

To exemplify the importance of optimizing the trigger routines, Figure 3.1 illustrates the runtime differences between naive and optimized triggers in the equilibrium scenario. The naive version recalculates the average over all samples each iteration, whereas the optimized version uses a ring buffer and running summation to reduce computational cost.

3.1.2 Available simulation statistics

3.1.3 Interaction with tuning strategies

As introduced in Section 2.3, AutoPas offers multiple different tuning strategies. Depending on the specific scenario settings, one strategy might be more efficient – e.g. the `slow-config-filter` setting in the heating-sphere example. To keep results comparable between scenarios therefore, all experiments were executed using the `full-search` strategy. As this strategy is expected to sample more suboptimal configurations than others, the effect of tuning iterations on the whole simulation runtime is higher. Using more tailored tuning strategies, the improvements as presented in this thesis might not be as visible.

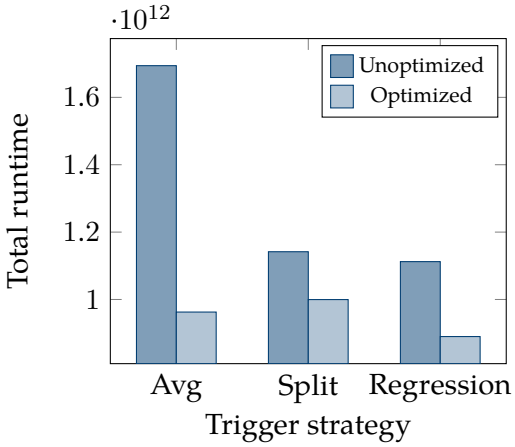


Figure 3.1: asdf

3.2 Time-based Triggers

The simplest approach in detecting whether the current configuration might have 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.

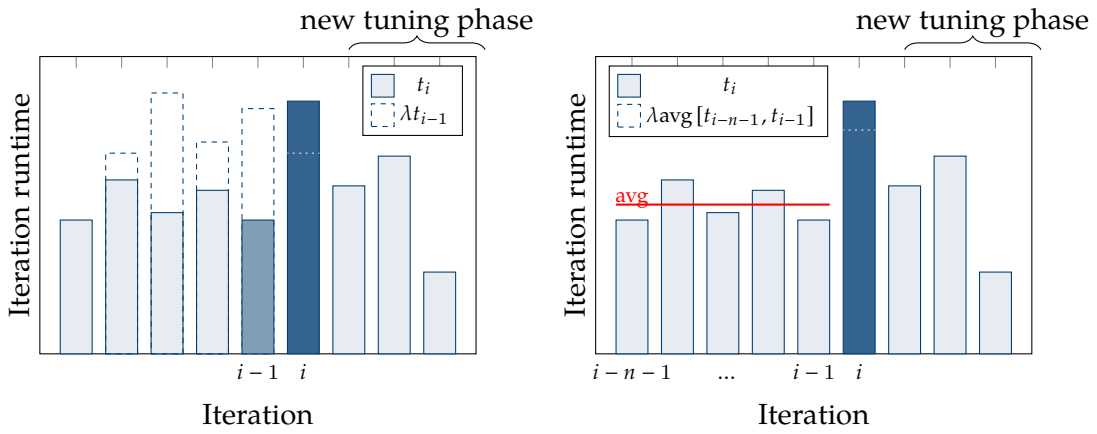


Figure 3.2: Comparison between the `TimeBasedSimpleTrigger` and `TimeBasedAverageTrigger` strategies for $\lambda = 1.5$.

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 3.3: An overview of the `TimeBasedSplitTrigger` and `TimeBasedRegressionTrigger` intervals.

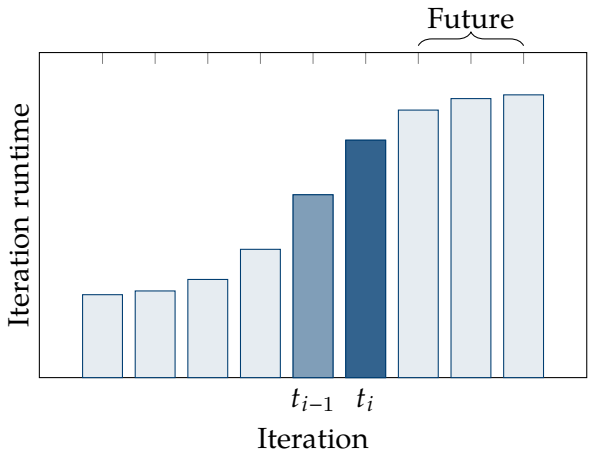


Figure 3.4: asdf

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 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).

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

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

$\hat{\beta}_1'$ 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).

$$\hat{\beta}_{\text{norm}} = 1 + \frac{n \cdot \hat{\beta}_1'}{2\bar{t}} \tag{3.6}$$

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

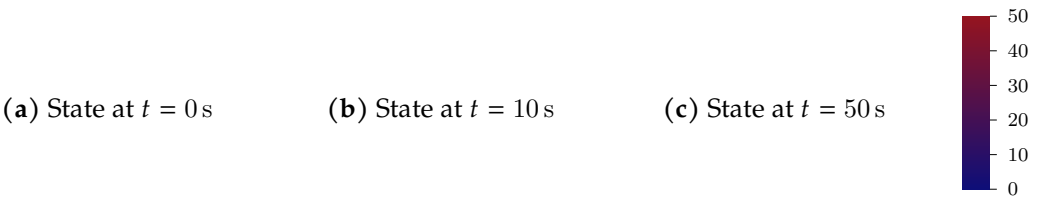
As to not limit our analysis to one specific simulation setting, we use a selection of benchmarking scenarios. These represent different structures as they may be used in real-world applications. The heating-sphere and exploding-liquid scenarios are identical to the ones given by Newcome et al., the configuration files have been adapted and parametrized for use in this thesis [14]. 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

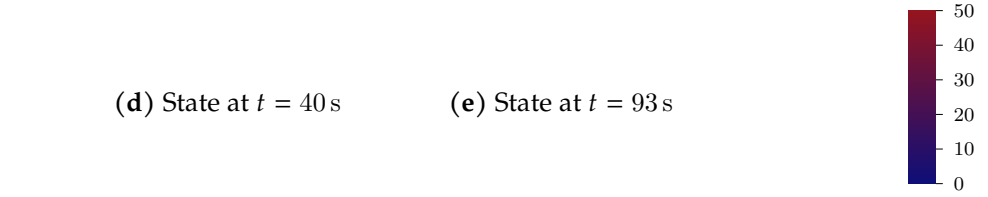
Similarly to the equilibrium scenario, the exploding liquid scenario starts of with the particles packed into a cuboid with periodic boundaries. The cuboid explodes in y-direction, collides with the boundary and finally settles into an equilibrium state spread out over the whole simulation space. If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



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

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.



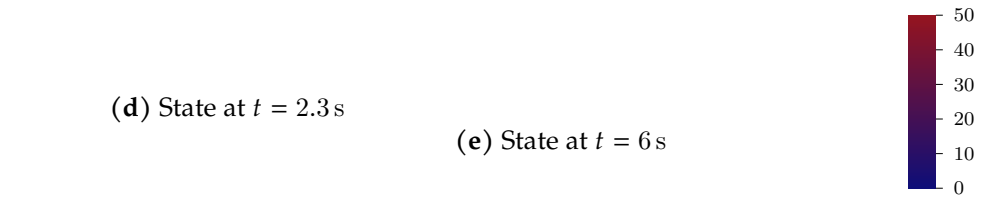
(**a**) State at $t = 0\,\mathrm{s}$ (**b**) State at $t = 5.5\,\mathrm{s}$ (**c**) State at $t = 20\,\mathrm{s}$

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

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 [13]. The sphere expands with the inrcease in temperature and particles are radiating out from the sphere center.



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

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

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, split 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 suitable default parameters for further evaluation.

5.1.1 Equilibrium

As can be seen in Figure 5.1, a trigger factor of $\lambda = 1.5$ leads to increased speedup compared to $\lambda = 1.25$. This is however mainly due to the nature of the equilibrium scenario: after the initial configuration selection, the optimal configuration is not expected to change. Therefore, not initiating any new tuning phases will lead to a decrease in total simulation runtime. That the speedup is indeed a result of the decreased number of tuning iterations can be verified by looking at the right-hand side plots; for the simple and averaging trigger it is most noticeable. Additionally, triggers with a larger sample size will typically trigger less frequently, as more of the variability in iteration runtime is smoothed out. For a too large number of samples, the speedup decreases however, computational overhead is directly proportional to the number of samples. This is best seen in the plots for the regression trigger, as the share of tuning iterations remains constant for all sample sizes, but the speedup decreases. Especially for the regression triggers, the computations required per sample and in each iteration are significant.

The collected data suggests default parameters as presented in Table 5.1.

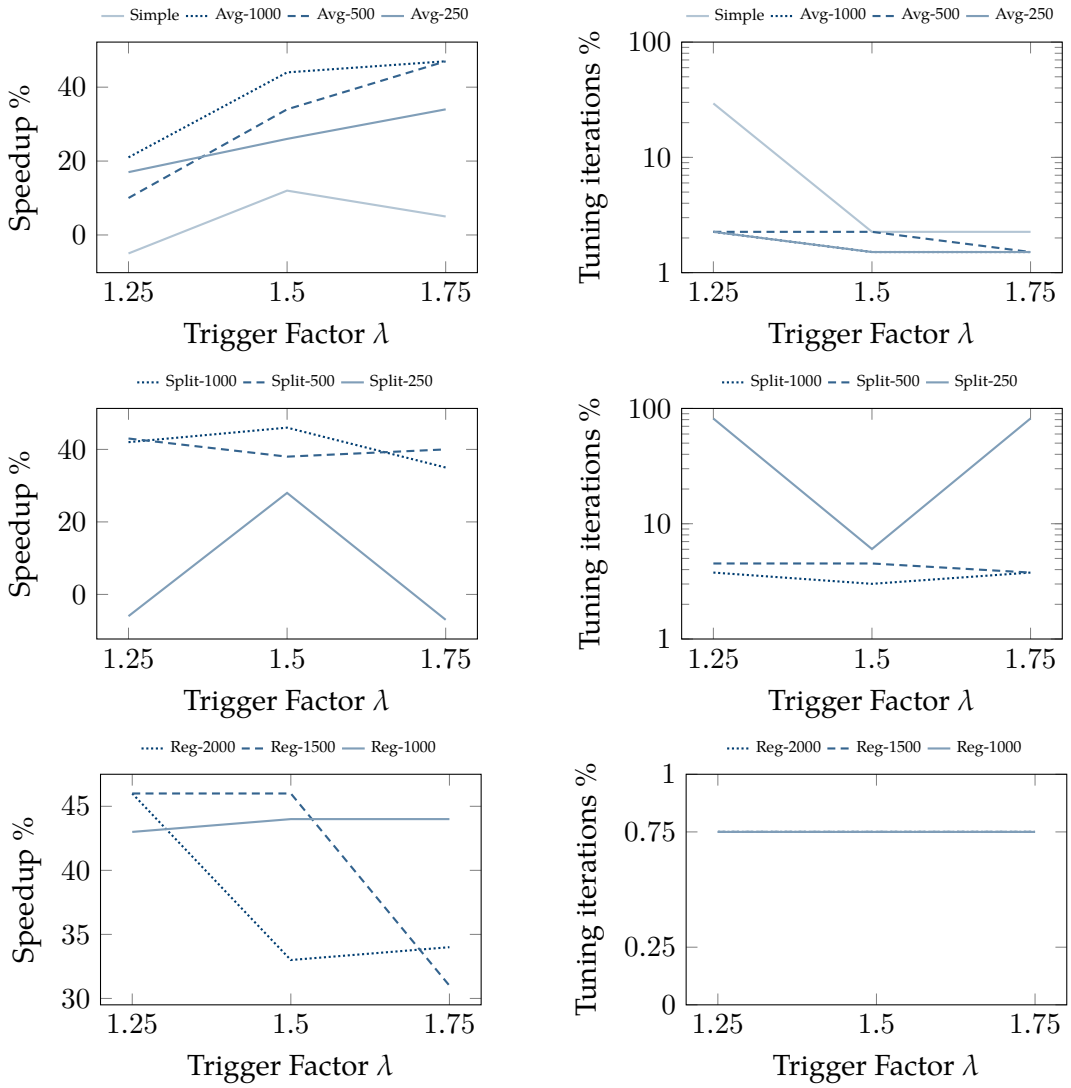


Figure 5.1: Trigger behavior in the equilibrium scenario, the numbers in the legends refer to the number of samples n considered. Note the logarithmic scale in the plots for the tuning iterations.

Trigger	**Trigger factor λ**	**Number of samples n**
`TimeBasedSimple`	1.75	-
`TimeBasedAverage`	1.75	500
`TimeBasedSplit`	1.5	1000
`TimeBasedRegression`	1.5	1500

Table 5.1: Suggested default parameters for the equilibrium scenario.

5.1.2 Exploding Liquid

5.1.3 Heating Sphere

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.

5.2.1 Simple Trigger

5.2.2 Single-iteration averaging Trigger

5.2.3 Interval averaging Trigger

5.2.4 Linear Regression Trigger

5.3 Optimality

5.4 Runtime

5.5 Share of tuning iterations

6 Conclusion

This chapter will ...

6.1 Dynamic Initiation of Tuning Intervals

6.2 Future Work

This page is intentionally left blank.

Bibliography

[1] V. Arnold, V. Kozlov, and A. Neishtadt. "Mathematical aspects of classical and celestial mechanics. Transl. from the Russian by A. Iacob. 2nd printing of the 2nd ed. 1993". In: *Itogi Nauki i Tekhniki Seriia Sovremennye Problemy Matematiki* (Jan. 1985).

[2] M. Born and R. Oppenheimer. "Zur Quantentheorie der Molekeln". In: *Annalen der Physik* 389.20 (1927), pp. 457–484. DOI: https://doi.org/10.1002/andp.19273892002.

[3] J. D. Emberson et al. "Cosmological neutrino simulations at extreme scale". In: *Research in Astronomy and Astrophysics* 17.8 (Aug. 2017), p. 085. ISSN: 1674-4527. DOI: 10.1088/1674-4527/17/8/85. URL: http://dx.doi.org/10.1088/1674-4527/17/8/85.

[4] 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.

[5] 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.

[6] Fabio Alexander Gratl et al. "AutoPas: Auto-Tuning for Particle Simulations". en. In: *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, May 2019. ISBN: 9781728135106. DOI: 10.1109/ipdpsw.2019.00125. URL: https://ieeexplore.ieee.org/document/8778280.

[7] Fabio Alexander Gratl et al. "N Ways to Simulate Short-Range Particle Systems: Automated Algorithm Selection with the Node-Level Library AutoPas". In: *Computer Physics Communications* 273 (2021), p. 108262. DOI: 10.1016/j.cpc.2021.108262.

[8] Michael Griebel, Gerhard Zumbusch, and Stephan Knapek. *Numerical Simulation in Molecular Dynamics: Numerics, Algorithms, Parallelization, Applications*. Vol. 5. Texts in Computational Science and Engineering. Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-68094-9. DOI: 10.1007/978-3-540-68095-6.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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].

[15] 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.

[16] Eric J.R. Parteli and Thorsten Pöschel. "Particle-based simulation of powder application in additive manufacturing". In: *Powder Technology* 288 (2016), pp. 96–102. ISSN: 0032-5910. DOI: https://doi.org/10.1016/j.powtec.2015.10.035.

[17] J P Verboncoeur. "Particle simulation of plasmas: review and advances". In: *Plasma Physics and Controlled Fusion* 47.5A (Apr. 2005), A231. DOI: 10.1088/0741-3335/47/5A/017.

[18] Troy Van Voorhis. *XII. The Born–Oppenheimer Approximation*. MIT OpenCourseWare, *Introductory Quantum Mechanics I (5.73), Fall 2005*. Lecture notes, Section XII (The Born–Oppenheimer Approximation). 2005. URL: https://ocw.mit.edu/courses/5-73-introductory-quantum-mechanics-i-fall-2005/bf19f723f60f6baeba12abcb6b97f6f5_sec12.pdf.

[19] 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.

This page is intentionally left blank.