FORSCHUNGSZENTRUM JÜLICH GmbH
Jülich Supercomputing Centre
D-52425 Jülich, Tel. (02461) 61-6402

Technical Report

Benchmark of fast Coulomb Solvers for open and periodic boundary conditions

Sebastian Krumscheid

FZJ-JSC-IB-2010-01

February 2010
(last change: 19.02.2010)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays the simulation of particle systems (N-body systems) is an important task in a broad range of research areas, e.g. physics, chemistry and electrical engineering. A straightforward approach to compute all pairwise interactions for such an N-body system has complexity $\mathcal{O}\left(N^2\right)$. However, there exist methods, so-called Fast Summation Methods, which reduce the computational effort from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}\left(N \log N\right)$ or even to $\mathcal{O}\left(N\right)$ [1]. An additional obstacle occurs, if the interaction between particles is described by a long-range potential, e.g. gravitational or electrostatic potential. Then such simulations are even worse to tackle, since no short-range approximation can be applied. In the case of electrostatic interactions, the methods which can speed up the calculation of the Coulomb force in N-body systems are called fast Coulomb solvers. In fact the above mentioned gravitational interaction represents a special case of electrostatic interactions, since a mass is always positive, while a charge can be positive or negative. Since these kind of interactions are essential for simulations in a host of physical and chemical processes as well as in astrophysical simulations, there exist a growing demand for fast Coulomb solvers. The Fast Multipole Method (FMM), developed by Greengard and Rohklin and published in 1987 [2], is one famous method, which reduces the complexity of the electrostatic N-body problem to $\mathcal{O}\left(N\right)$.

In any case, there exist a wide range of methods able to compute the electrostatic interactions in N-body systems for both systems with open boundary conditions and systems with periodic boundary conditions, e.g. [3, 4, 2, 5]. However, this report is a first approach to evaluate the implementation of the Fast Multipole Method of our group at Forschungszentrum Jülich GmbH against other freely available methods.
The results presented here were obtained on JUMP [6] at the Jülich Supercomputing Centre (JSC). The main goal of this comparison was to determine how well the implementation of several different algorithms dealing with electrostatic interactions perform. Since our intention was to see how fast the underlying method is and not how well the algorithm can be parallelized, we used one single processor on a single node without any communication (neither `OMP` nor `MPI`). Additionally we limited the consumable memory to maximal 6 GB ($6/7$ of it for data and $1/7$ for stack) for each test case.
Each tested method is applied to a test particle distribution with different levels of required accuracy. All of these test cases were performed ten times and the minimum computation time was recorded. Additionally we applied a quadruple precision FMM on each test scenario, thus one could determine whether the method holds the required accuracy or not.

Chapter 2

Methods

2.1 Open Boundary Conditions

Since its publication, the FMM received much attention in the literature and a lot of efficient implementations have been published as well, e.g. [7]. However, some authors mention large prefactors and high crossover points as a drawback of the FMM, stating that FMMs are only preferable for large N, e.g. [8, 9]. Since the prefactors and crossover points strongly depend on the implementation of a method, the purpose of this report is to demonstrate that there exists a huge discrepancy even between different FMM implementations. Therefore we now want to compare different FMM implementations. Since a detailed description of each method would go beyond the scope of this report, we simply want to point out the main differences compared to the FMM implementation of our group (*libfm*) [10, 11].

2.1.1 A Fast Multipole Method using Cartesian Moment Representation

The FMM itself has an optimal linear complexity with respect to the number of particles N. However, crucial for an efficient code is its implementation. In 1994 White and Head-Gordon published an approach for an efficient implementation of the FMM [7]. Needless to say, there exist other implementation approaches as well. One alternative approach is used by the group of `multipole.org` [12]. The main goal of this group was to develop a method which is as independent of the computer architecture as possible. Due to this goal the method (referred to as *FMM-A*) is an ANSI C implementation of the three-dimensional FMM algorithm.
In contrast to *libfm* the method *FMM-A* uses a Cartesian representation for the moments. On one hand the usage of Cartesian coordinates simplifies the implementation but on the other hand it leads to a higher complexity with respect to the number of multipole moments. Another constraint of the actual implementation is, that *FMM-A* is only able to work with unit weights, i.e. charges or masses of equal strength.

2.1.2 A Kernel-Independent Fast Multipole Algorithm

In molecular dynamics simulations the particle interactions act through a potential. Many of these potentials are related to the fundamental solution of elliptic partial differential equations. For example the fundamental solution of the Laplace equation $\Delta u = 0$ is given by:

$$\Phi : \mathbb{R}^n \setminus \{0\} \ni x \mapsto \Phi(x) := \begin{cases} \frac{1}{2}\|x\| & \text{, if } n = 1 \\ \frac{1}{2\pi}\ln\|x\| & \text{, if } n = 2 \\ \frac{1}{(2-n)\omega_n}\frac{1}{\|x\|^{n-2}} & \text{, if } n \geq 3 \end{cases}, \quad \omega_n = \mathrm{vol}\left(S^{n-1}\right), \tag{2.1}$$

with $S^{n-1} := \partial B_1(0) \subset \mathbb{R}^n$ is the unit sphere of $(\mathbb{R}^n, \|\cdot\|)$, for derivation see reference [13]. For the three dimensional case $\omega_n = 4\pi$, the fundamental solution is $\Phi(x) = (-4\pi \|x\|)^{-1}$ which is proportional to the Coulomb potential, i.e. $1/\|x\|$.
Hence, if one solves the N-body problem for the potential of the electrostatic case, then one also obtains a proportional solution for the fundamental solution of the Laplace equation in the N-body problem, e.g. solving a Poisson equation by means of convolution of Φ and right-hand side ρ, whereas ρ is the particle density function of the corresponding particle system. Now the approach of the kernel-independent FMM (later on referred as *FMM-B*) is to be able to evaluate fundamental solutions of various partial differential equations, e.g. Laplace, Navier or Stokes equation, by applying an adaptive FMM-like algorithm [14]. This is achieved by using an equivalent (continuous) density which is obtained by solving local boundary value problems. Thus *FMM-B* is able to deal with an expanded range of different potentials. Due to the abilities of kernel-independence and the parallelized code, *FMM-B* requires the installation of several libraries, e.g. `FFTW-3`, `PETSc`, `BLAS` and `LAPACK` plus `MPI`, even if one wants to use it on only one processor. For more information on *FMM-B*, see reference [15].

2.2 Periodic Boundary Conditions

In addition to systems with open (non-periodic) boundary conditions systems with periodic boundary conditions (periodic systems) play a decisive role in research as well. In the case of periodic systems one distinguishes between the number of periodic spatial directions, and therefore we denote the system as 1D-, 2D- or 3D-periodic.
According to literature the FMM is mainly applied in non-periodic systems consisting of a large number of particles. Since *libfm* is able to handle periodic boundary conditions the intention is to show how systems with periodic boundary conditions can be solved efficiently using *libfm*.
Therefore we compare *libfm* against other methods to solve systems with periodic boundary conditions. Since some authors state, that FMM "*becomes comparable in speed to P^{3}M only for more than a million particles*" [16, p. 206], we additionally wanted to disprove the above mentioned statement. One famous software package to solve systems with periodic boundary conditions is `ESPResSo` [17, 18]. Since the `ESPResSo` package contains algorithms like *P^{3}M*, *MMM2D*, *ELC* and *MMM1D* for the simulation of electrostatic interactions we compared *libfm* against these methods. Besides a C-compiler `ESPResSo` requires the script language `Tcl`. Since the chosen methods utilize a Fourier transformation, the `FFTW-3` library is required too.

Again, a detailed description of the methods would go beyond the scope of this report, therefore we only mention some characteristics and refer to literature for further details.

2.2.1 Particle-Particle/Particle-Mesh Method (P^{3}M)

There exist a wide variety of methods to solve particle systems with periodic boundary conditions applied in all three spatial directions. One famous method is called Ewald summation. The approach of Ewald was to split the slowly and only conditionally converging sum of the Coulomb potential into two exponentially converging sums. Therefore one has to split the Coulomb potential into two parts

$$\frac{1}{r} = \frac{\operatorname{erfc}(\beta r)}{r} + \frac{1 - \operatorname{erfc}(\beta r)}{r}, \tag{2.2}$$

with $r := \|x\|$, $\beta \in \mathbb{R}_>$ being a precision control parameter and $\operatorname{erfc}(z) := \frac{2}{\sqrt{\pi}} \int_z^\infty \exp\left(-\tau^2\right) \mathrm{d}\tau$ being the complementary error function. The first summand decays rapidly and the second summand is smooth for all r. Due to these characteristics, the summands are treated differently. The

first summand can be evaluated efficiently in real space, while the second term can be treated in Fourier space efficiently. The complexity of the Ewald formula can be reduced to $\mathcal{O}\left(N^{3/2}\right)$, with respect to the number of particles inside the simulation box, using optimized parameters [16].
An efficient successor of the classic Ewald method is the so-called Particle-Particle/Particle-Mesh Method (P^3M). It reduces the complexity even further to $\mathcal{O}\left(N \log N\right)$. The asymptotical acceleration is achieved by smearing (interpolating) the charges onto a regular mesh and using the discrete Fast Fourier Transformations (FFT) in lieu of continuous Fourier Transformations (CFT). For further details on Ewald methods and its ancestors, see reference [19].

It should be noted, that one can formulate Ewald method for 1D- or 2D-periodic systems as well [16]. However, in these cases the Ewald method has a complexity of $\mathcal{O}\left(N^2\right)$ and thus other methods are preferable. Additionally one should note, that the P^3M approach is only possible for 3D-periodic systems.

2.2.2 MMM Methods and the Electrostatic Layer Correction

An alternative approach to make the conditionally convergent Coulomb sum convergent is by means of a convergence factor. Therefore one might use the obvious identity

$$\frac{1}{r} = \lim_{\gamma \to 0} \frac{\exp\left(-\gamma \cdot r\right)}{r} \,. \tag{2.3}$$

Replacing the summand r^{-1} of the Coulomb sum by $\left(r \exp\left(\gamma \cdot r\right)\right)^{-1}$, with $r := \|x\|$, and applying the limit outside the summation, one can show that this limit exists. Additionally, its value only differs from the value of the Ewald method by a term proportional to the dipole term for 3D-periodic systems. Like Ewald methods one splits the sum into two parts with respect to the distance of particles to each other. The sum of particles far apart is treated efficiently in Fourier space. Because of its slow convergence, the sum of nearby particles can not be treated efficiently in a Fourier space. Thus a less favorable computational approach is used. However, using optimized parameters for the method the complexity yields $\mathcal{O}\left(N^{7/5}\right)$, but it can be reduced further by a hierarchical implementation to $\mathcal{O}\left(N \log N\right)$ [16, 20, 21].
This briefly described method is called MMM. Like P^3M and Ewald summation the MMM method solves 3D-periodic systems. But the MMM approach can also be customized for 1D- and 2D-periodic systems. The resulting methods are named *MMM1D* and *MMM2D* respectively. The complexity of *MMM2D* is $\mathcal{O}\left(N^{5/3}\right)$ and is therefore preferable to Ewald methods for 2D-periodic systems. Also for 1D-periodic systems the *MMM1D* is preferable to Ewald methods although the complexity of $\mathcal{O}\left(N^2\right)$ is the same. This is due to a smaller prefactor. For further details on the MMM methods and especially on *MMM1D* and *MMM2D*, see references [20, 21].

Due to the complexity of $\mathcal{O}\left(N^{5/3}\right)$ *MMM2D* is only suitable for rather small systems. Another approach for larger 2D-periodic systems is to scale the particles inside the 3D simulation box, so that there is a sufficient large gap between the image boxes in the non-periodic direction. Now one applies a faster 3D method. For instance, consider a 2D-periodic N-body system. Let us denote z the non-periodic direction and assume further, that all particles are located in an original simulation box of size $L_x \times L_y \times L_z$. Now one replaces the original 2D-system by an approximate 3D-system by means of embedding the original box inside a new simulation box of size $L_x \times L_y \times (L_z + h)$, where h is sufficiently large. The empty space regions added to the system this way are called gaps. The new simulation box is now continued periodically in the prior non-periodic direction z. These modifications are shown schematically in figure 2.1. Thus the idea is to approximate a 2D-periodic system by a 3D-periodic system by means of introducing gaps. However the geometry of the original system has to be changed. Obviously this approach results in huge systematic errors. By

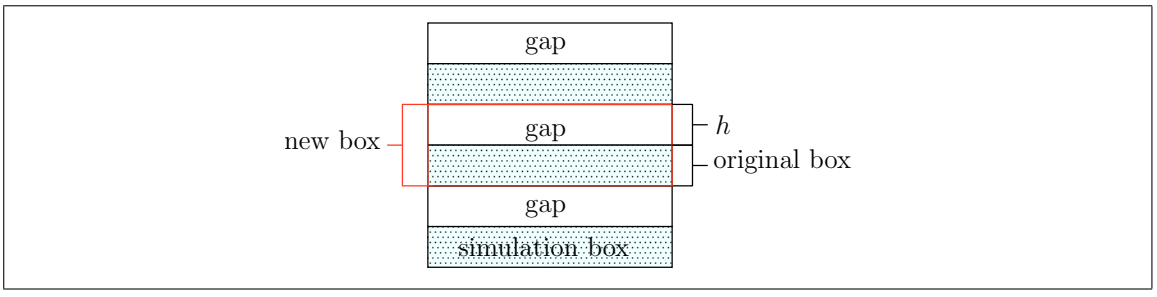


Figure 2.1: Schematic modification of the original simulation box in order to build a 3D-periodic system.

changing the Ewald summation order in a certain way and some further modifications, like *MMM2D* related approximations for particles far apart and analytical subtraction of unwanted interactions, one is able to reduce these errors. The resulting formula is called electrostatic layer correction (*ELC*). Hence it is not a real method itself, it rather allows us to use faster 3D methods, e.g. P^3M or MMM, for 2D-periodic systems. For further information, see reference [22].

Chapter 3

Benchmark Results

In this section we want to introduce the considered test particle distributions and present an excerpt of the benchmark results. The complete data of our performed benchmark is given in the appendix. Additionally the used input data is also available online.

3.1 Test Particle Distributions

3.1.1 Open Boundary Conditions

We used two different particle distributions to compare the different FMM implementations applied to systems with open boundary conditions. The first test case is a uniform distribution of particles inside the unit cube with varying number of particles. Therefore the x, y and z coordinates of each particle were assigned by uniformly, independently and identically on the interval $]0, 1[$ distributed random variables respectively. Hence, we consider a N-body system, where particle i, $1 \leq i \leq N$, is located at $(x_i, y_i, z_i)^T \in \mathbb{R}^3$. The values of x_i, y_i and z_i are defined by

$$\alpha_i := \xi_\alpha \ , \quad \text{with} \quad \xi_\alpha \sim \text{UNI}\,(]0,1[) \ , \quad \forall \alpha \in \{x, y, z\} \ , \ 1 \leq i \leq N \ . \tag{3.1}$$

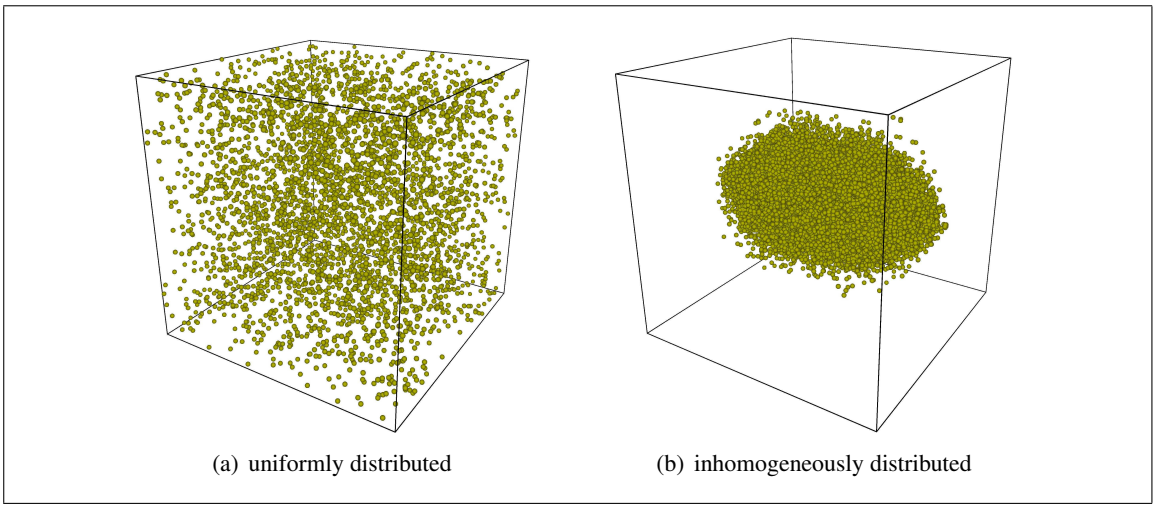
The input data were generated by a C program using a time-dependent seed and the default random number generator. Hence, the following lines yield the positions of the particles inside the unit cube:

```
1       time(&t);
2       srand((unsigned int)t);
3       for(i=0;i<N;++i)
4       {
5         x[i] = rand()/((double)RAND_MAX);
6         y[i] = rand()/((double)RAND_MAX);
7         z[i] = rand()/((double)RAND_MAX);
8       }
```

The second test case consists of 114537 inhomogeneously distributed particles inside the unit cube. Examples of these test cases are shown in figure 3.1. Due to the fact that *FMM-A* is only able to deal with unit-charges, the charges in both systems where set as $q = 1.0$. The largest test case consists of $8^8 \approx 16 \cdot 10^6$ particles. To anticipate, this limitation is due to the fact that only *libfm* was able to handle this many (and even more) particles with respect to the provided resources.

3.1.2 Periodic Boundary Conditions

For each case of periodic boundary conditions we used two different particle distributions to compare *libfm* against the corresponding methods of the `ESPResSo` package. Just as for systems with



(a) uniformly distributed (b) inhomogeneously distributed

Figure 3.1: Test particle distributions: (a) shows an example of a system with particles distributed randomly inside the unit cube and (b) displays an inhomogeneous distribution of particles.

open boundaries we used a random distribution of particles inside the unit cube with varying number of particles for each situation of periodicity. In contrast to the test case of systems with no periodicity we assigned a charge $q \in \{-1, +1\}$ to each particle, so that $\sum_i^N q_i = 0$ (i.e. N is even). The charges were assigned alternately to the particles. One example of such a particle distribution is shown in figure 3.2(a). The second test case for each situation of periodic boundary conditions was a Madelung particle distribution inside the unit cube with varying number of particles. In this test case the particles are located on a regular mesh with alternating charges. The detailed structure depends on the kind of periodicity and is shown in 3.2(b) - (d). An advantage of a Madelung systems is, that either the analytical Coulomb energy is known (1D and 2D) or at least a rapidly convergent sum-representation is given (3D), and thus a direct measure for the accuracy is available. The number of particles inside both systems is limited by 46656. To anticipate, this limitation is due to the fact that only *libfm* was able to handle this many (and even more) particles with respect to the provided resources.

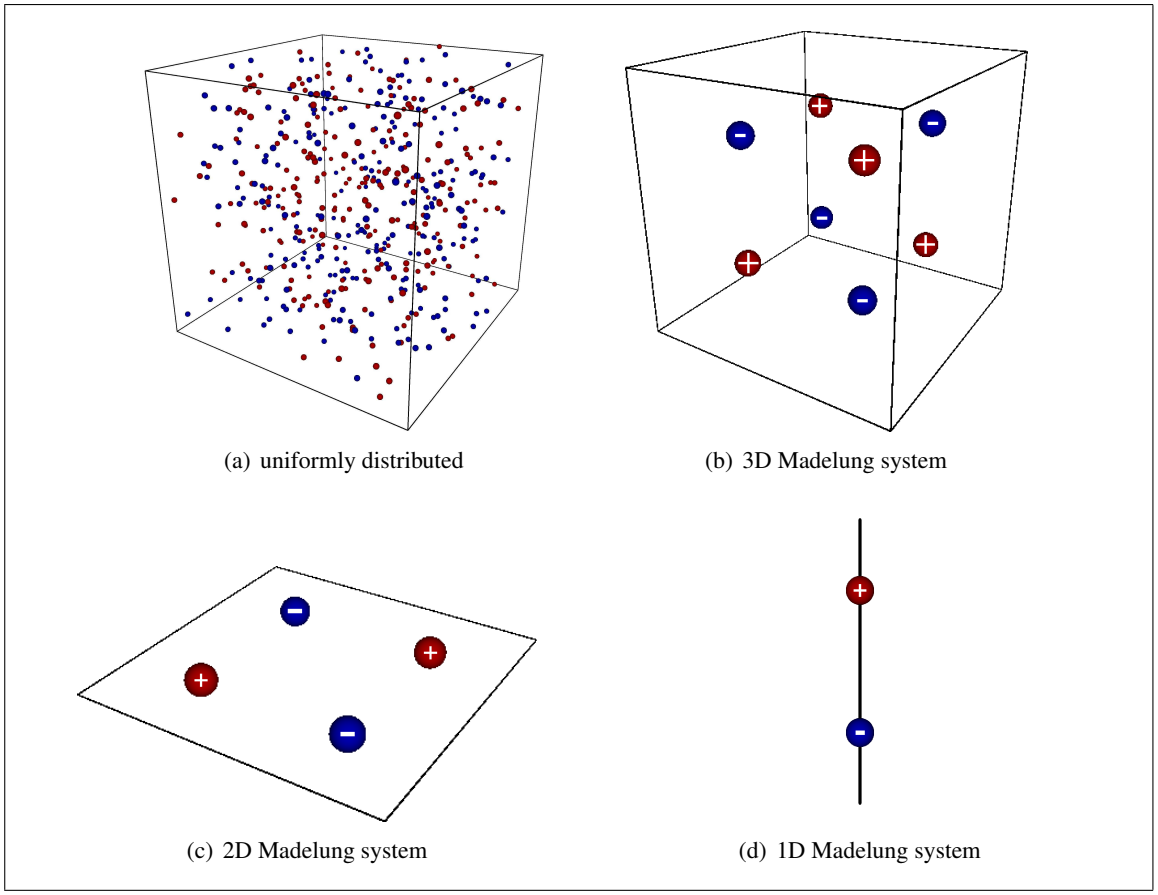
3.2 Open Boundary Conditions

In the case of particle systems with open boundaries, we compared our FMM implementation (*libfm*) against some other FMM implementations, namely *FMM-A* and *FMM-B*.

3.2.1 Benchmark

All considered methods enable the user to specify a required energy accuracy. Hence, a first comparative criterion was whether a method fulfills the requested accuracy goal for the energy of the system or not. We determined that there has not been an issue with the energy accuracy neither for *FMM-B* nor for *libfm*. However, for different levels of accuracy *FMM-A* did not reach the requested accuracy goal. Table 3.1 shows exemplarily the results for the random particle distribution. The first column contains the name of each method and the second column the number of particles inside the considered system. The following columns show the measured computation time in seconds for different accuracies. The emphasized numbers mark the cases, in which the accuracy goal was not reached. E.g. if the required relative error was lower or equal than 10^{-7}, it points out that *FMM-A* was not able to fulfill the accuracy goal independently of the considered test cases[1]. A result,

[1]Needless to say, a relative error of 10^{-5} is a suitable result for some applications. But if the program allows the user to enter a required relative error and the program exits normally without any hint, that the accuracy goal could not be reached, then the method is not reliable.



(a) uniformly distributed (b) 3D Madelung system

(c) 2D Madelung system (d) 1D Madelung system

Figure 3.2: Test particle distributions used for systems with periodic boundary conditions: (a) is an example of a systems with particles uniformly distributed inside the unit cube and (b) - (d) are examples of Madelung systems for 1D-, 2D- and 3D-periodic systems respectively.

		Requested Accuracy ΔE						
Method	N	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
FMM-A	512	0.0	-	0.02	-	**0.05**	-	**0.12**
FMM-B	512	-	0.02	-	0.06	-	0.24	-
libfm	512	0.01	-	0.0	-	0.0	-	0.0
FMM-A	4096	0.15	-	0.46	-	**0.97**	-	**2.55**
FMM-B	4096	-	0.35	-	0.55	-	1.15	-
libfm	4096	0.07	-	0.09	-	0.08	-	0.12

Table 3.1: Excerpt of the timing results in seconds for the homogeneous particle distribution for different levels of required relative energy error. The emphasized numbers of measured timings and indicate scenarios, where a certain method was not able to fulfill the accuracy goal.

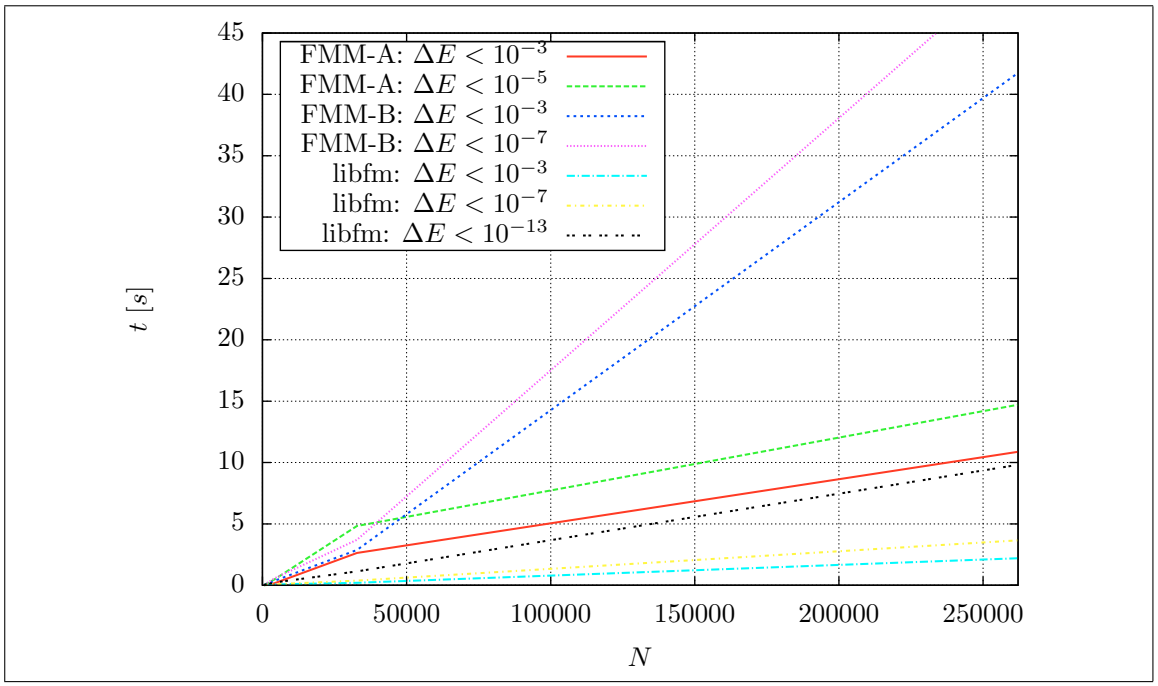


Figure 3.3: Excerpt of the results for the test system containing uniformly distributed particles for different levels of reached relative energy error. In the plots the timings in seconds versus the number of particles $N < 8^6 = 262144$ are displayed.

which did not fulfill the required accuracy goal, can however fulfill a lower accuracy goal. Hence, the following results are based on the a-posteriori determined (reached) relative energy error ΔE.

> *Remark.* As mentioned above, we denote the reached relative energy error by ΔE. Below, we investigate the timings of the different methods for various N and different ΔE. Therein we use the notation $\Delta E < \varepsilon$ to denote the best timing result for each N, so that the reached relative energy error $\Delta E < \varepsilon$ for the considered N.

An excerpt of the results for the test system containing uniformly distributed particles is shown in figure 3.3, 3.4 and 3.5. Therein the measured computation times in seconds versus the number of particles N are displayed. Figure 3.3 shows a comparison of the methods with different reached relative energy errors ΔE for rather small particle systems. The number of particles is lower than $2.6 \cdot 10^5$. The figure shows, that for very small systems all methods consumed approximately the same amount of time to solve the considered particle system. Especially if one takes into account that there might exist an inaccuracy in the determined time, due to inaccuracy in the measurement or due to rounding. Since error bars for this figure would be comparable small ($< 10^{-2}\ s$) in contrast to the measured timings and thus would not yield to a better understanding, they are neglected here. However, it is also apparent, that the measured timings increase differently for each method with increasing number of particles. *FMM-B* is the method where the time and accordingly the prefactor increases most rapidly. Also *FMM-A* can not catch up to *libfm*, since one has to compare the low accuracy result ($\Delta E < 10^{-3}$) of *FMM-A* against the high accuracy result ($\Delta E < 10^{-13}$) of *libfm* to see similar timing results. Thus, one already identifies significant discrepancies between the different methods for systems containing up to 10^5 particles. To sum up, even for rather small systems *libfm* is significantly faster for all level of accuracy than the methods compared against and *FMM-A* is faster than *FMM-B* for systems with more than $50\,000$ particles.
Figure 3.4 shows the same benchmark results as figure 3.3 but it covers a wider range of number of particles, up to approximately one million particles. We see the same qualitative behavior as

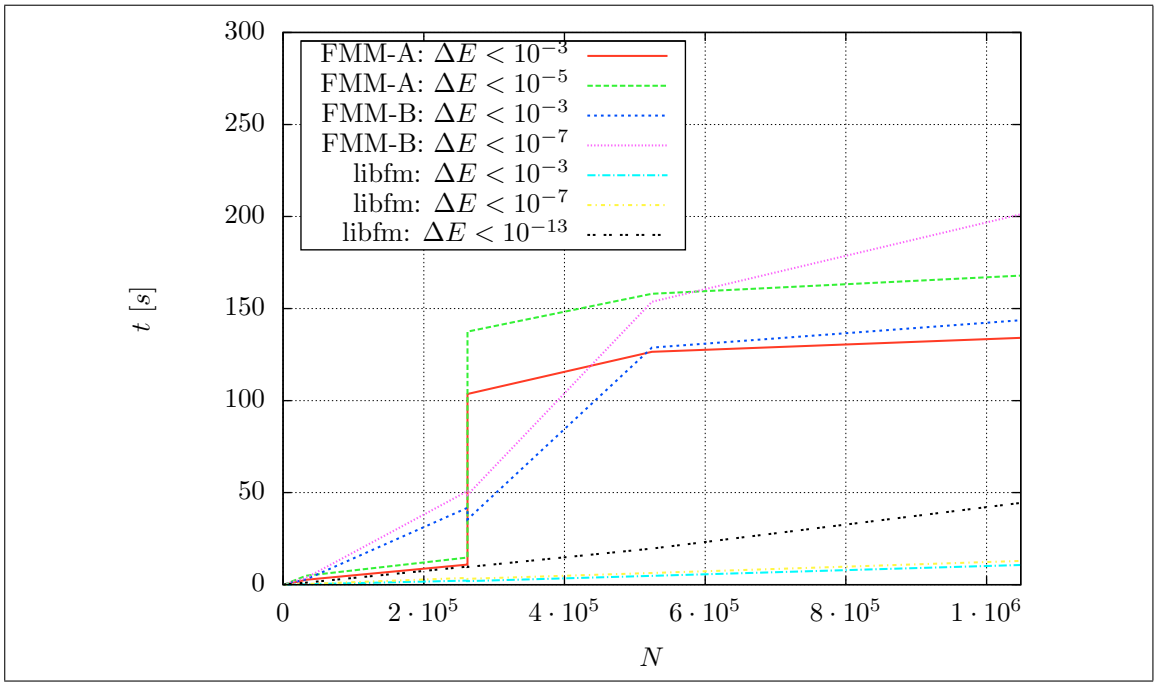


Figure 3.4: Excerpt of the results for the test system containing uniformly distributed particles for different levels of reached relative energy error. In the plots the timings in seconds versus the number of particles $N \leq 4 \cdot 8^6 = 1048576$ are displayed.

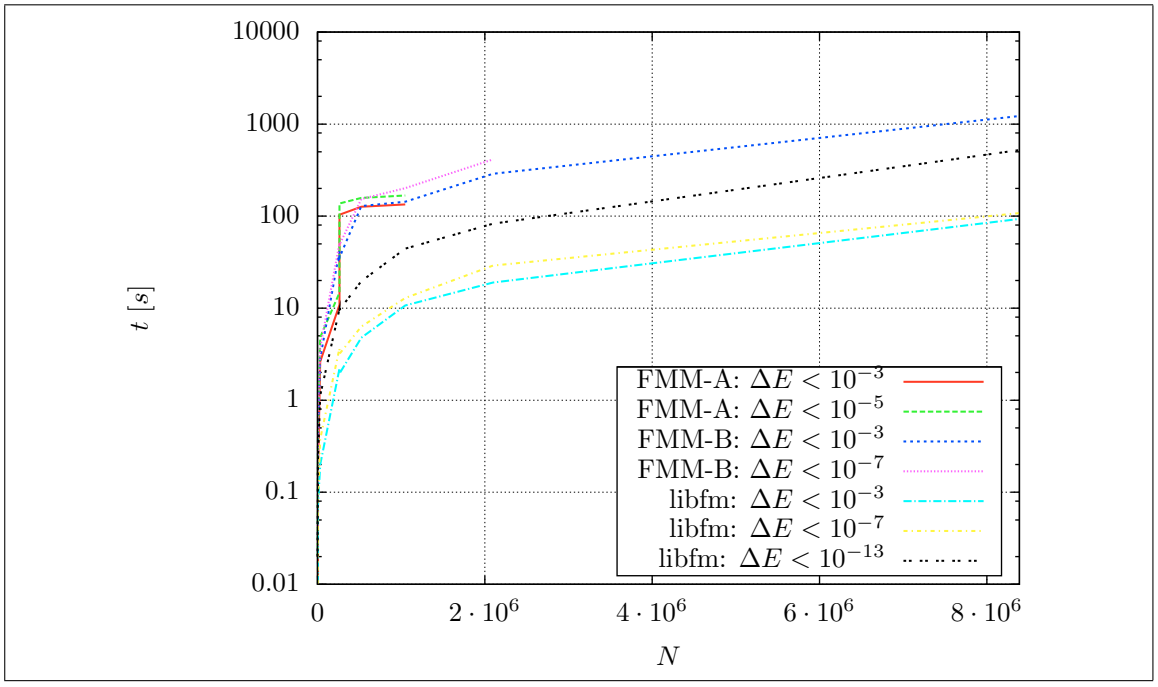


Figure 3.5: Excerpt of the results for the test system containing uniformly distributed particles for different levels of reached relative energy error. In the plots the timings in seconds versus the number of particles $N \leq 4 \cdot 8^7 = 8388608$ are displayed in a logarithmic scale.

	Accuracy ΔE				
Method	$< 10^{-3}$	$< 10^{-5}$	$< 10^{-7}$	$< 10^{-9}$	$< 10^{-11}$
FMM-A	41.69	55.37	-	-	-
FMM-B	-	119.57	129.15	142.36	-
libfm	7.49	18.17	12.08	-	14.61

Table 3.2: Excerpt of the results for the inhomogeneous particle distribution containing 114537 particles for different levels of reached relative energy error and different methods. The units of the displayed timings is seconds.

discovered in the previous figure, but there is a conspicuous gap[2] in the measured curves of *FMM-A* at approximately $2.6 \cdot 10^5$. Due to this huge gap of the measured timings of *FMM-A*, there is a range where *FMM-B* is faster than *FMM-A* with respect to the same reached relative energy error. But since the curves of *FMM-B* increase so rapidly, *FMM-A* again becomes faster for increasing number of particles. However, *libfm* provides the fastest implementation for all levels of accuracy compared with both *FMM-A* and *FMM-B*. E.g. for $N = 1\,048\,576$ and $\Delta E < 10^{-3}$ *libfm* is approximately 13 times faster than *FMM-A* and 14 times faster than *FMM-B* respectively.
Figure 3.5 shows an overview of the complete range of considered number of particles N. The measured timings are displayed in a logarithmic scale. The overall view confirms, that *libfm* is significantly faster than all the methods compared with. For instance, one can infer from figure 3.5 that one would be able to solve a system containing of approximately $8 \cdot 10^6$ particles using *libfm* in about the same time, solving a system of $5 \cdot 10^5$ particles using *FMM-B*, with respect to a reached relative energy error $\Delta E < 10^{-3}$. Additionally one detects that only the low accuracy *FMM-B* ($\Delta E < 10^{-3}$) and *libfm* (all level of accuracy) provide results for particle systems containing approximately 8 million particles. As mentioned above, it turned out that no method, apart from *libfm*, was able to solve the test system with more than 8 million particles. Therefore, only results of particle systems up to approximately 8 million particles are considered in the plots. *FMM-A* as well as medium and high accuracy *FMM-B* were not even able to reach this 8 million particles barrier. Medium and high accuracy *FMM-B* ran into memory issues with particle systems containing more than two million particles[3]. The issue of *FMM-A* again is related to the gap we already mentioned. Analog to the appearance of the first gap in the measured time of *FMM-A* there occurs another gap at approximately 2 million particles. In this situation the gap was so huge, that is was not reasonable to continue comparing *FMM-A* against the other methods. Due to these gaps *FMM-A* is not feasible for rather large systems.
However, the comparison indicates that for rather small particle systems, i.e. $N < 4100$ (see table A.1 for details), all three FMM implementations consumed approximately the same amount of time. The benchmark points out, that for larger N, *libfm* is the only feasible method, independently of the accuracy.

The excerpt of the results for the inhomogeneous particle distribution are presented in table 3.2. The first column denotes the used method and the following columns contain the measured computation time for several reached relative energy errors. As already mentioned, the particle system contains 114537 particles. The obtained results show a similar behavior compared to the results of a homogeneous particle distribution. However, due to the clustered particle distribution the computational load is higher. As for systems with uniformly distributed particles of comparable size we see that *libfm* is faster than *FMM-A*, which is faster than *FMM-B*. It is also obvious that the gaps between the determined timings are already big, indicating significant performance differences between the

[2]This gap is due to a huge time consumption of the methods internal procedure, where the Near-Field is defined. The gap occurs at the crossing from $8^6 - 1$ to 8^6 particles. The developer told us, that this issue is known and they are going to resolve it in a parallel version.

[3]The program crashes with a fatal PETSC error, most likely due to memory allocating issues.

different methods. Assuming a similar monotonicity for different sizes of inhomogeneous particle distributions as for the homogenous case mentioned above, one infers that *libfm* is the only feasible method for rather large inhomogeneously distributed systems.

To outline the above presented, *libfm* is the only feasible method for dealing with arbitrary particle distributions with open boundary conditions among the methods compared with. This is due to the facts, that *libfm* can achieve any (reasonable) accuracy and is on that level of accuracy the fastest method.

3.3 Periodic Boundary Conditions

For the case of periodic boundaries, we compared our FMM implementation (*libfm*) against some methods of the `ESPResSo` package. In detail, that is *P*^{3}*M* for 3D-periodic, *MMM2D* and *ELC*/*P*^{3}*M* for 2D-periodic and *MMM1D* for 1D-periodic systems respectively. Every method provided by this package for the electrostatic case comes along with an automatic tuning procedure. This automatic tuning procedure allows the user to specify an absolute root mean square (RMS) force error. The RMS force error F_{RMS} is defined by

$$F_{\text{RMS}} := \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(F_i - \hat{F}_i\right)^2} \equiv \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\Delta F_i\right)^2}\,, \tag{3.2}$$

where $\hat{F}_i$ is the estimator for the force on particle i and hence, F_i is the real force respectively. For the methods of the `ESPResSo` package estimators for ΔF_i are available [16], so that the tuning procedure can compute a set of parameters. Using this set of parameters, the corresponding method solves the particle system in the minimum of time, with respect to the required RMS force error. Since we want to compare measured timings of different methods for the same relative energy error, we performed tests of a variety of required absolute RMS errors. Due to the fact, that the automatic tuning procedures are quite time consuming, we excluded these timings in our comparisons, i.e. we only considered the time, a method itself consumed to solve the particle system, but not the time needed to generate the optimized parameters.[4] It is also worth to mention that the introduced particle limit of approximately $50\,000$ particles is due to the high time consumption of the automatic tuning procedures as well as the poor complexity of some of the methods of the `ESPResSo` package.

3.3.1 3D-Periodic Benchmark

An excerpt of the results for test systems with periodic boundary condition applied in all spatial directions is shown in figure 3.6 and 3.7. Therein the measured time in seconds versus the number of particles N for each method and each test case are displayed. Additionally both the 3D Madelung particle distribution and the uniform particle distribution inside the unit cube (marked as *Cube* in the figures) are analyzed. Figure 3.6 shows the comparison of *P*^{3}*M* and *libfm* for both the 3D Madelung systems and the homogeneous system, where the number of particles is lower or equal 4250. Up to 1500 particles all displayed test scenarios consumed approximately the same amount of time. For systems containing more than 1500 particles the obtained timings of the *P*^{3}*M* method increase more rapidly than these of *libfm* with increasing N. Thus, there exists a significant gap between the results obtained by *libfm* and these obtained by *P*^{3}*M* already for systems with approximately 4000 particles.
In figure 3.7 an overview of the complete range of considered number of particles is shown, i.e.

[4] In contrast, the time for obtaining optimized parameter for our FMM implementation (*libfm*) was not excluded in the measured timings.

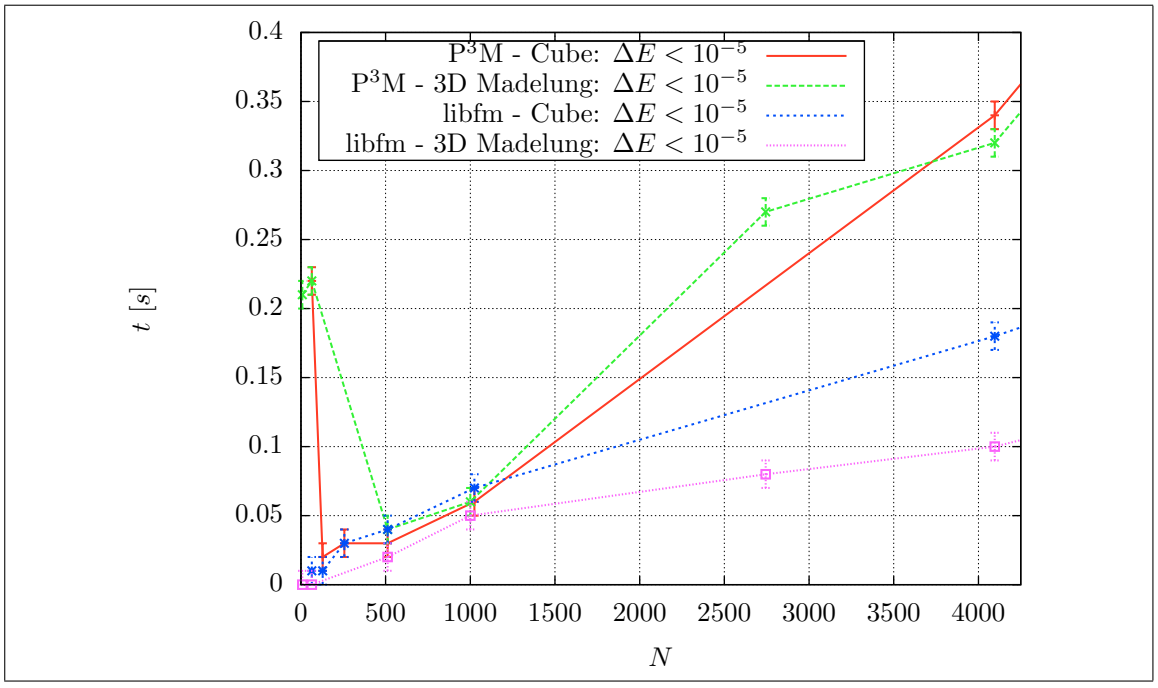


Figure 3.6: Excerpt of the results for test systems with periodic boundary condition applied in all spatial directions for a reached relative energy error $\Delta E < 10^{-5}$. In the plots the timings in seconds versus the number of particles $N \leq 4250$ are displayed.

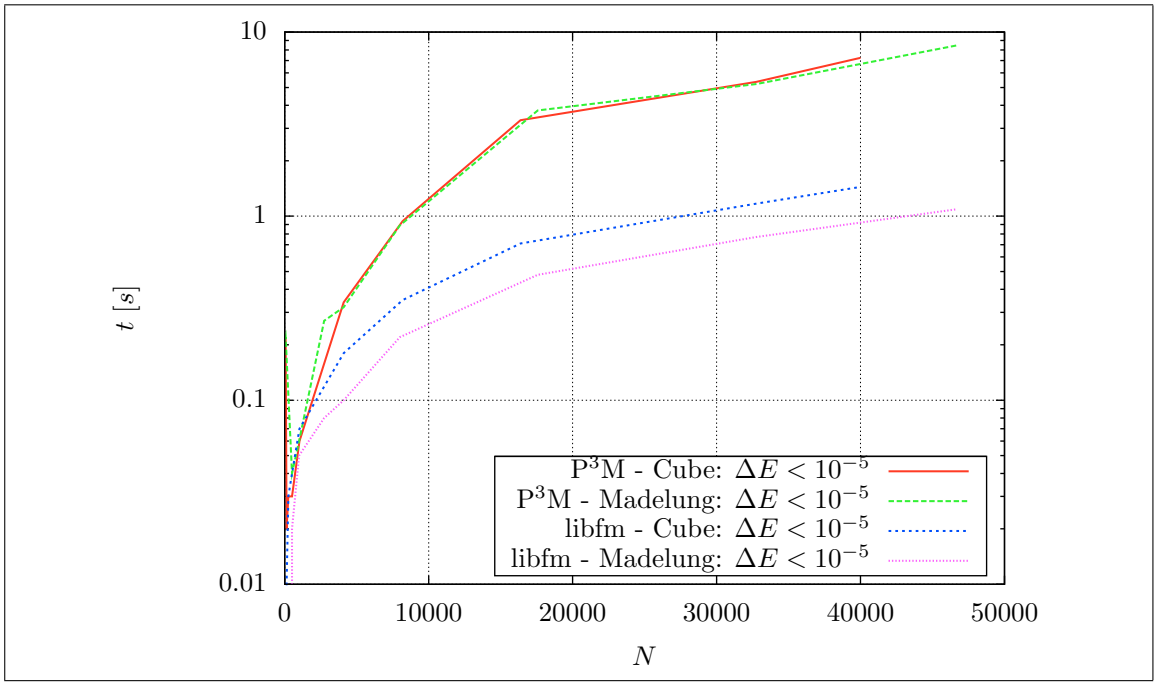


Figure 3.7: Excerpt of the results for test systems with periodic boundary condition applied in all spatial directions for a reached relative energy error $\Delta E < 10^{-5}$. In the plots the timings in seconds versus the number of particles $N \leq 46656$ are displayed in a logarithmic scale.

$N \leq 40000$ for the uniform distribution and $N \leq 46656$ for the 3D Madelung system respectively. The measured timings are displayed in a logarithmic scale. This combined view confirms, that *libfm* is significantly faster than P^3M, especially for particle systems with more than 5000 particles. The presented results in figure 3.6 and 3.7 correspond to calculations with a reached relative energy error of $\Delta E < 10^{-5}$. It turned out in our benchmark, that P^3M was not able to provide solutions with a much higher accuracy goal, what coincides with reference [16]. In contrast to P^3M, *libfm* is able to provide solutions on a higher level of accuracy, e.g. *libfm* with a reached relative energy error of $\Delta E < 10^{-13}$ is faster than P^3M with $\Delta E < 10^{-5}$ starting from approximately 11500 particles.
Since the Madelung system consists of particles located on a regular mesh and thus, no local inhomogeneity exists, one would expect that P^3M as well as *libfm* are faster for the 3D Madelung system, than for systems with randomly distributed particles. However, figure 3.7 demonstrates that the results of P^3M do not show any significant runtime difference between the different particle distributions, whereas the results of *libfm* do.

3.3.2 2D-Periodic Benchmark

The available methods of the `ESPResSo` package for 2D periodicity are *MMM2D* and *ELC*. While performing different test scenarios, we obtained for both methods some constraints. The constraint of *MMM2D* is, that there is no automatic tuning procedure for all relevant parameters available. Thus, the presented results might not be optimal. This constraint is not relevant if one requires low accuracy results, e.g. $\Delta E \leq 10^{-3}$ or $\Delta E \leq 10^{-5}$, since in this situation *ELC* with automatic parameter tuning with a better complexity is available. However, *ELC* using P^3M is not suitable for high accuracy results. Therefore, the missing parameter tuning and accordingly the (maybe) non-optimal set of parameters becomes relevant. But due to the poor complexity of *MMM2D*, this constraint is only be apparent for rather small particle systems. *ELC* only works properly, if there is a sufficient large gap in the non-periodic direction between the surface of the simulation box and the closest particle in one of the periodic images. Otherwise the original particle system has to be modified (see also figure 2.1). Therefore the test cases of particles distributed uniformly inside the unit cube have to be modified by *ELC*, since in these cases such a gap might not exist. However, the Madelung test cases already contain a sufficiently large gap. It turned out that the timing results of *ELC* in the case of a non Madelung system is comparable to the timings of the Madelung system. However, the reached relative energy error was very poor, i.e. $\Delta E \approx 10^{-1}$ in some cases. Therefore this rescaling due to *ELC* is a real constraint and hence, we only consider the Madelung system from now on. For details on the timings of *ELC* in the case of non Madelung systems and the related relative energy error, see table A.3.

Figure 3.8 and 3.9 show an excerpt of the results for 2D-periodic Madelung systems. The organization of the plots is analog to the plots in the previous section. In figure 3.8 the measured timings for the different methods for the 2D Madelung scenario with up to 3000 particles and a reached relative energy error $\Delta E < 10^{-5}$ are displayed. Due to the poor complexity the displayed data of the *MMM2D* runs show the most rapid slope already for small systems. Additionally, it is noticeable, that there already exist a significant gap between measured timings of *ELC* and *libfm* starting from $N = 1000$. If $N < 1000$ both methods are comparable in speed but with an advantage for *libfm*. From approximately 500 particles forward, the timings of *ELC* increases more rapidly than the timings of *libfm* with increasing number of particles. Thus, there exists a significant difference in the execution time between both methods. Figure 3.9 shows an overview for the complete range of considered number of particles. Therein the time is displayed in a logarithmic scale. It is noticeable that the displayed curve of *ELC* stops at approximately 32500 particles. This is due to the fact, that *ELC* did not finish any scenario with more particles. Additionally this figure shows how *libfm* again outclasses the other methods.
Since *ELC* was not applicable for the uniform particle distribution inside the unit cube and due to

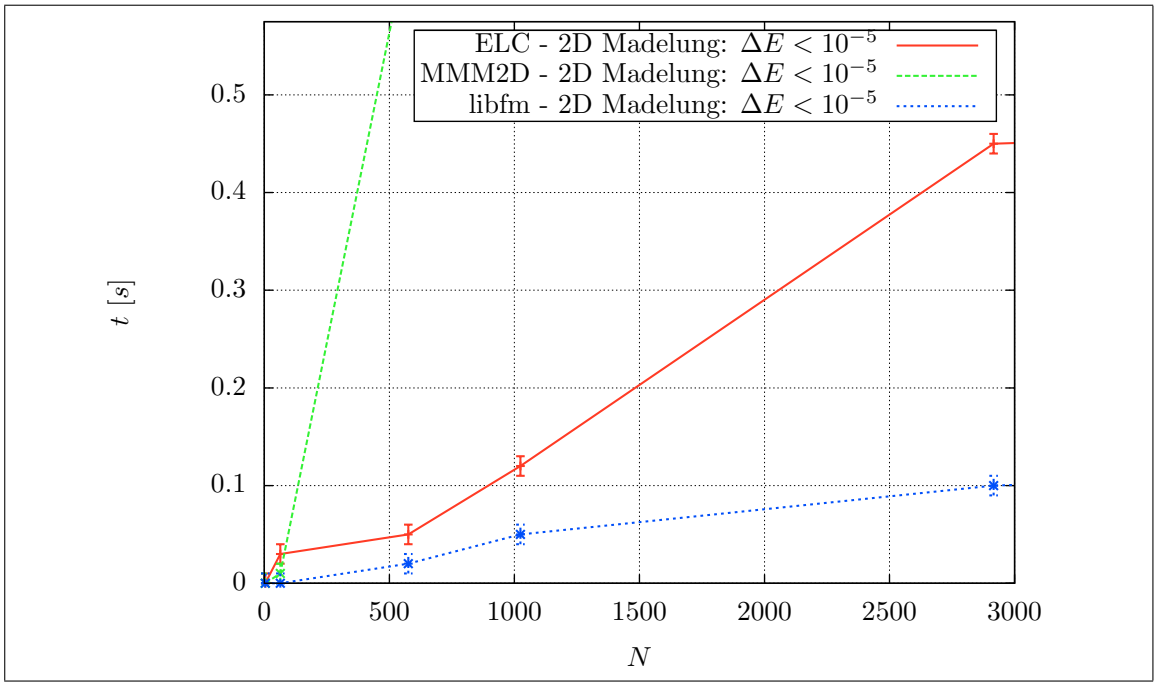


Figure 3.8: Excerpt of the results for 2D-periodic Madelung systems with a reached relative energy error $\Delta E < 10^{-5}$. In the plots the timings in seconds versus the number of particles $N \leq 3000$ are displayed.

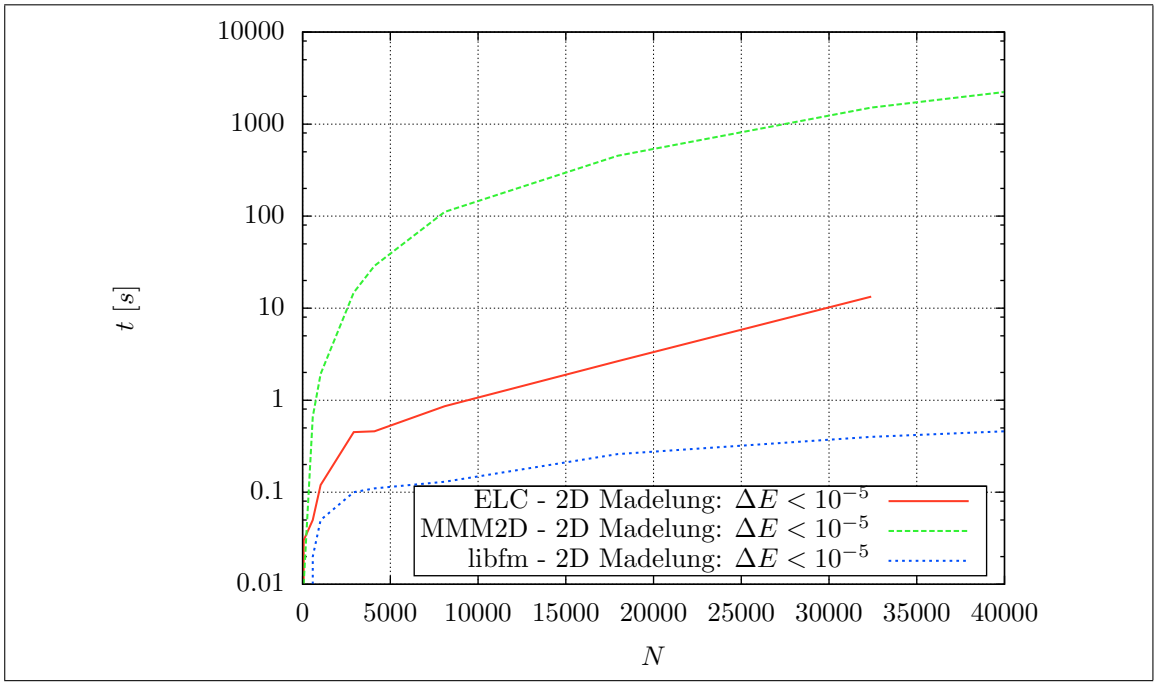


Figure 3.9: Excerpt of the results for 2D-periodic Madelung systems with a reached relative energy error $\Delta E < 10^{-5}$. In the plots the timings in seconds versus the number of particles $N \leq 40000$ are displayed in a logarithmic scale.

	N					
Method	64	128	256	1024	4096	8192
MMM2D	0.01	0.03	0.1	0.65	8.21	31.25
libfm	0.01	0.01	0.03	0.07	0.13	0.27

Table 3.3: Excerpt of the result for the random particle distribution with a reached relative energy error $\Delta E < 10^{-5}$. Due to the *MMM2D* complexity of $\mathcal{O}\left(N^{5/3}\right)$, the number of particles in the table is limited by 8192. Displayed timings for different N are in seconds.

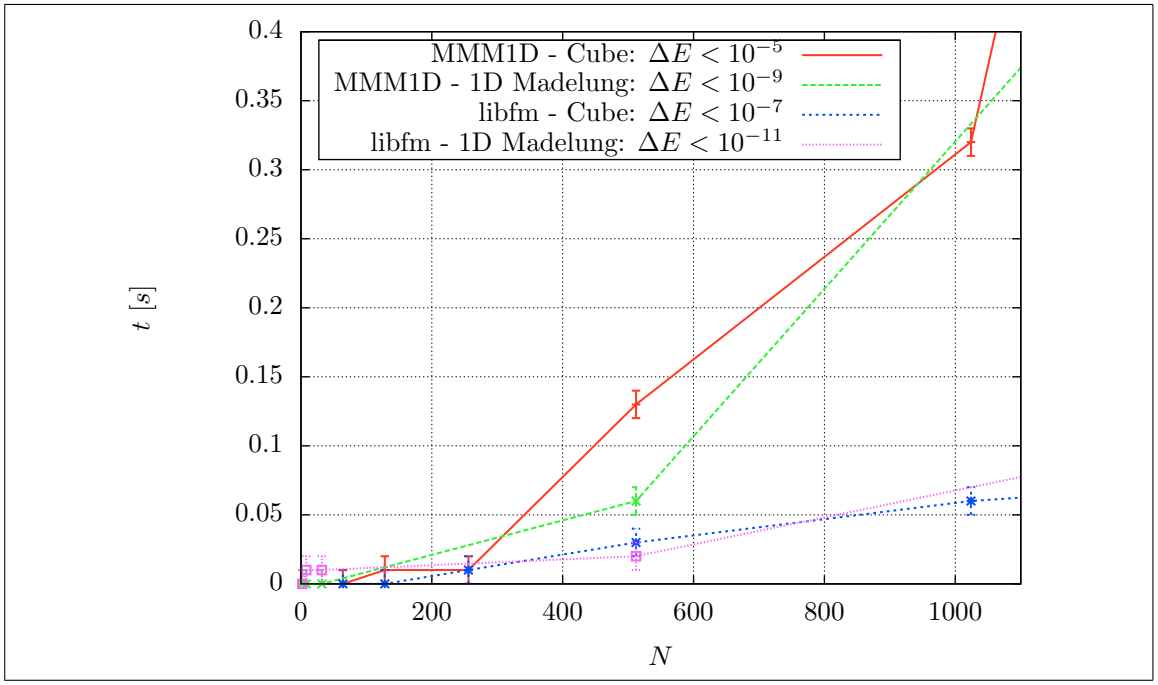


Figure 3.10: Excerpt of the results for test systems with periodic boundary condition applied in only one spatial direction with different levels of reached relative energy error. In the plots the timings in seconds versus the number of particles $N \leq 1100$ are displayed.

the poor complexity of *MMM2D* an excerpt of the result for this test case is shown in table 3.3. The first column denotes the used method to solve the uniform particle system and the following columns contain the measured time for different number of particles inside the system. The timings were obtained with a relative energy error of $\Delta E < 10^{-5}$. It is quite obvious that *MMM2D* is only comparable in speed to *libfm* for very small particle systems, i.e. $N \leq 256$.
As mentioned above, *libfm* is able to determine results of all (reasonable) level of accuracy. Since *ELC* enables the usage of a 3D periodic method, e.g. P^3M, the level of accuracy depends directly on the 3D solver. Due to the fact that P^3M is the recommended method (see reference [16]), the obtained accuracy results of *ELC* are only available for low accuracy calculations. However, *MMM2D* was able to provide solutions with a higher accuracy goal than *ELC*, e.g. for both the system of uniformly distributed particles and the 2D Madelung system *MMM2D* obtained a relative energy error of $\Delta E < 10^{-7}$. But due to the poor complexity *MMM2D* was not competitive with *libfm*.

3.3.3 1D-Periodic Benchmark

The last presented benchmark results are for systems with periodic boundary condition in one spatial direction. Again we used a 1D Madelung system and a uniformly distributed particle system, each with various numbers of particles, to compare *libfm* against *MMM1D*. An excerpt of the benchmark is shown in figure 3.10 and 3.11. The measured timings of *MMM1D* and *libfm* for both the 1D Madelung systems and uniformly distributed systems with up to 1100 particles with different

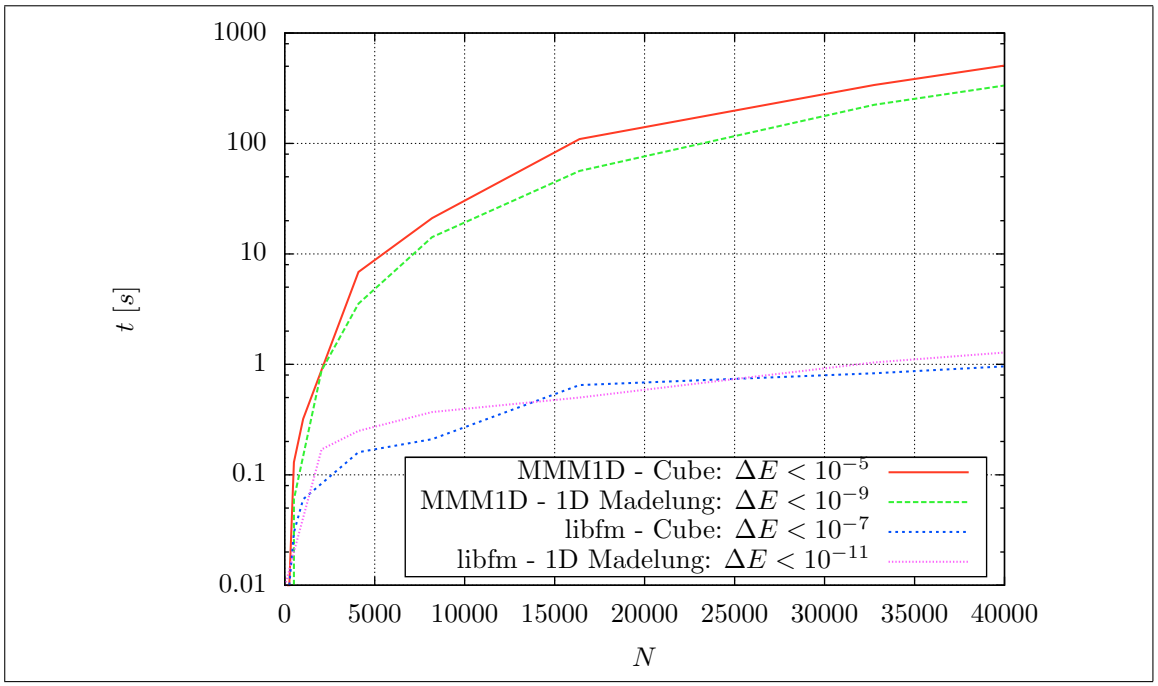


Figure 3.11: Excerpt of the results for test systems with periodic boundary condition applied in only one spatial direction with different levels of reached relative energy error. In the plots the timings in seconds versus the number of particles $N \leq 40000$ are displayed.

levels of accuracy are shown in figure 3.10. Up to approximately 400 particles the methods are comparable in speed. But due to the poor complexity of *MMM1D* the curves of *MMM1D* increase more rapidly than these of *libfm*. This behavior is perpetuated for increasing number of particles, as shown in figure 3.11. Hence, there exists a huge discrepancy between the timing curves of *MMM1D* and these of *libfm* respectively.
With respect to the accuracy, we additionally obtained a dependency of *MMM1D* from the simulated particle system. On one hand *MMM1D* produces solutions with a high accuracy for the 1D Madelung scenario, independently of the required accuracy. This might be due to the fact, that in the 1D Madelung scenario the particles are located on a regular grid. On the other hand, if the particles are not located on a regular grid, *MMM1D* generates solutions of low and intermediate accuracy with respect to the same required RMS force error. However, *libfm* was able to determine results of all levels of accuracy.

3.3.4 Remarks on Optimized Parameters for ESPResSo

Finally we want to return to the mentioned issue of obtaining optimized parameters for the methods of the `ESPResSo` package. As already mentioned, for the methods of the `ESPResSo` package we did not consider the consumed time for tuning the parameters. Even with respect to this advantage for the methods of the `ESPResSo` package *libfm* is preferable. Table 3.4 shows exemplarily a comparison between the consumed time only for tuning the parameters, the consumed time for solving the problem by the according method of the `ESPResSo` package and the overall consumed time for solving the problem by *libfm* (the last three columns). The first three columns determine the method of the `ESPResSo` package, the test case and the number of particles therein respectively. All methods of the `ESPResSo` package require a huge amount of time to obtain optimized parameters in comparison to the time to solve the corresponding problem, especially *ELC* and P^3M. It is worth to mention, that the time used to tune the parameters for *MMM2D* is comparable small for the systems presented in table 3.4 and is therefore not included in the table. However, since *libfm* optimizes its own parameters on the fly, one would have to add column four and five to compare against the timings of *libfm* and thus, one can infer, that *libfm* is the only feasible method for periodic systems

			Timing Approximation t $[s]$		
Method	Test Case	N	only Tuning	only Solving	complete *libfm*
MMM1D	1D Madelung	512	0.6	0.06	0.02
MMM1D	Cube	512	1.4	0.11	0.03
MMM1D	1D Madelung	4096	43.6	3.54	0.12
MMM1D	Cube	4096	89.7	4.34	0.16
ELC/*P*^{3}*M*	2D Madelung	576	9.9	0.03	0.02
ELC/*P*^{3}*M*	2D Madelung	4096	328.0	0.46	0.18
P^3*M*	3D Madelung	512	73.3	0.04	0.02
P^3*M*	Cube	512	57.2	0.03	0.04
P^3*M*	3D Madelung	4096	311.8	0.32	0.1
P^3*M*	Cube	4096	429.1	0.34	0.18

Table 3.4: Exemplary comparison between the time for tuning the parameter and the time for solving the particle system by *MMM1D*, *ELC* or *P*^{3}*M* and the overall timing of *libfm*.

of any kind among the compared methods.

Chapter 4

Conclusion

The aim of this report was a first evaluation of the FMM implementation *libfm*, provided by our group at JSC. Hence, we compared freely available methods, with the ability to compute the electrostatic interactions in N-body systems against *libfm*. Therefore we differentiated between particle systems with open boundary conditions and particle systems with periodic boundary conditions.

In the case of open boundary conditions applied to an N-body system, we compared different FMM implementations against each other. This is due to the optimal asymptotical complexity of $\mathcal{O}(N)$ of the Fast Multipole Method itself. However, crucial for the practicability of an implementation are prefactors and crossover points as well as the accuracy. Hence, our goal was to demonstrate, that there exist a huge discrepancy even between different FMM implementations, although the asymptotical complexity is the same (compare table 4.1). To benchmark the different implementations, we used two different kinds of particle distributions. The first kind is a random particle distribution with varying N and the second one is an inhomogeneous particle distribution with 114537 particles. Since some methods were not able to handle 8^8 or more particles with respect to the provided resources, we limited the number of particles in our benchmark. In addition to the timings we also considered the obtained relative energy error ΔE. If $N < 4100$, then the benchmark pointed out, that the timings for all FMM implementations were approximately equal (with respect to the same ΔE), with an advantage for *libfm*. Hence, there exist no reason, not to choose *libfm* for an application. For larger particle systems, there exist significant timing differences between the methods. These differences amplify with increasing N. However, *libfm* obtained by far the best timing results and outclassed the other FMM implementations. In addition to the discrepancy in the timings, our benchmark showed, that some methods had trouble to reach the required accuracy goal, in particular with increasing number of particles N. However, *libfm* was able to provide results for any accuracy. Altogether, the benchmark illustrates the advantage of *libfm* compared to the other considered FMM implementations for particle systems with open boundary conditions.

In the case of particle systems with periodic boundary conditions, the methods for electrostatic interactions of the `ESPResSo` package receive much attention in literature and in applications. Hence, we compared *libfm* against these methods. The `ESPResSo` package provides different methods

Method	Asymptotic Complexity
FMM-A	$\mathcal{O}(N)$
FMM-B	$\mathcal{O}(N)$
libfm	$\mathcal{O}(N)$

Table 4.1: Overview of the asymptotic complexity of the considered methods for particle systems with open boundary conditions.

Method	Periodicity	Asymptotic Complexity
MMM1D	1D	$\mathcal{O}\left(N^2\right)$
MMM2D	2D	$\mathcal{O}\left(N^{5/3}\right)$
ELC/P^{3}M	2D	$\mathcal{O}\left(N \log N\right)$
P^{3}M	3D	$\mathcal{O}\left(N \log N\right)$
libfm	1D,2D,3D	$\mathcal{O}\left(N\right)$

Table 4.2: Overview of the asymptotic complexity of the considered methods for particle systems with periodic boundary conditions.

for different kinds of periodicity. Therefore, the asymptotic complexity of the considered methods of the `ESPResSo` package vary. Table 4.2 shows an overview of the asymptotic complexity of the methods considered in the benchmark. The table shows the best asymptotic complexity for *libfm*. Again, crucial for a fast code is it's implementation, since an implementation of an $\mathcal{O}\left(N\right)$ algorithm can be worse for some N, than an implementation of an algorithm with a disadvantaged complexity. Hence, our goal was to demonstrate, that *libfm* is able to handle particle systems with periodic boundary conditions efficiently. To benchmark the different methods, we again used two different kinds of particle distributions. The first kind is a random particle distribution with varying N and the second one is a Madelung system with varying N. The Madelung system depends on the periodicity. Since some methods were not able to handle 46656 or more particles with respect to the provided resources, we limited the number of particles in our benchmark. Again, in addition to the timings we also considered the obtained relative energy error ΔE. For each case of periodicity the benchmark showed, that there exists a crossover point N_c, so that the timings of the considered methods are approximately equal, if $N \leq N_c$ ($N_c \approx 1000$, for details see appendix A). For particle systems larger than N_c, there exist significant timing differences between the methods. These differences are even amplified with increasing N. However, *libfm* obtained by far the best timing results and outclassed the other methods in each case of periodicity. In addition to the discrepancy in the timings, our benchmark showed, that some methods were only applicable for a rather small range of ΔE. However, *libfm* was able to provide results for any arbitrary accuracy. Altogether, the benchmark illustrates also the advantage of *libfm* compared to the other considered methods for particle systems with periodic boundary conditions.

Hence, if considering the complete benchmark, then one can sum up, that not any of the methods compared with can be considered to be competitive against *libfm*, neither in the sense of speed nor in the sense of accuracy.

Funding

This work was supported by a JARA-SIM [23] seed fund within the project "Performance Classification, Parallelization and Algorithmic Enhancement of a Single-Parameter FMM Implementation".

Appendix A

Benchmark Results - Overview

A.1 Open Boundary Conditions

The following table A.1 contains the results of all successfully performed test cases with open boundary conditions. If a method did not exit normally or did not finish, then this case is not reported below. Thus there might not be a result in the table, where one might expect an entry. For example, in section 3.2.1 we mentioned that *FMM-A* was not able to solve the N-body system with respect to the used resources if N is larger than 2 million and therefore no columns for *FMM-A* and $N > 2.0 \cdot 10^6$ exist in table A.1. The cases where $N = 16777216$ are also noticeable, since only *libfm* was able to handle systems with this many particles. Additionally, the high accuracy calculation to determine the reached relative energy error would be too time consuming, we did not verify the required accuracy of *libfm*.
The first column in table A.1 shows the considered particle distribution as introduced in section 3.1.1. The following two columns display the used method and the size of the particle system respectively. Since all considered methods allow the user to specify an accuracy, column four shows this user defined value. The last two columns show the outcomes of the benchmark defined by the previous columns, namely the reached accuracy ΔE and the measured computational time respectively.

Table A.1: Benchmark results for particle systems with open boundary conditions. The timings are rounded at two digits after the decimal dot.

Distribution	Method	N	Requested ΔE	Reached ΔE	$t\ [s]$
Cube	*FMM-A*	512	1.0e-03	8.8e-05	0.00
Cube	*FMM-A*	512	1.0e-05	5.2e-06	0.02
Cube	*FMM-A*	512	1.0e-07	8.8e-07	0.05
Cube	*FMM-A*	512	1.0e-09	1.8e-07	0.12
Cube	*FMM-B*	512	1.0e-04	1.2e-15	0.02
Cube	*FMM-B*	512	1.0e-06	1.2e-15	0.07
Cube	*FMM-B*	512	1.0e-08	1.2e-15	0.24
Cube	*libfm*	512	1.0e-03	1.2e-04	0.01
Cube	*libfm*	512	1.0e-05	6.0e-17	0.00
Cube	*libfm*	512	1.0e-07	6.0e-17	0.00
Cube	*libfm*	512	1.0e-09	6.0e-17	0.00
Cube	*libfm*	512	1.0e-11	6.0e-17	0.00
Cube	*libfm*	512	1.0e-13	6.0e-17	0.00
Cube	*FMM-A*	4096	1.0e-03	1.3e-04	0.15
Cube	*FMM-A*	4096	1.0e-05	3.5e-06	0.46

continued on next page

Table A.1 – continued from previous page

Distribution	Method	N	Requested ΔE	Reached ΔE	$t\ [s]$
Cube	*FMM-A*	4096	1.0e-07	2.5e-06	0.97
Cube	*FMM-A*	4096	1.0e-09	4.6e-07	2.55
Cube	*FMM-B*	4096	1.0e-04	6.5e-06	0.35
Cube	*FMM-B*	4096	1.0e-06	2.1e-08	0.55
Cube	*FMM-B*	4096	1.0e-08	3.7e-10	1.15
Cube	*libfm*	4096	1.0e-03	1.4e-04	0.07
Cube	*libfm*	4096	1.0e-05	1.9e-08	0.09
Cube	*libfm*	4096	1.0e-07	1.6e-10	0.08
Cube	*libfm*	4096	1.0e-09	3.6e-12	0.12
Cube	*libfm*	4096	1.0e-11	2.2e-14	0.22
Cube	*libfm*	4096	1.0e-13	1.4e-16	0.07
Cube	*FMM-A*	32768	1.0e-03	1.4e-04	2.63
Cube	*FMM-A*	32768	1.0e-05	3.6e-06	4.83
Cube	*FMM-A*	32768	1.0e-07	2.6e-06	11.78
Cube	*FMM-A*	32768	1.0e-09	4.9e-07	30.01
Cube	*FMM-B*	32768	1.0e-04	1.4e-05	2.87
Cube	*FMM-B*	32768	1.0e-06	4.6e-08	3.70
Cube	*FMM-B*	32768	1.0e-08	1.1e-09	5.07
Cube	*libfm*	32768	1.0e-03	1.5e-04	0.20
Cube	*libfm*	32768	1.0e-05	1.7e-08	0.38
Cube	*libfm*	32768	1.0e-07	1.5e-11	0.50
Cube	*libfm*	32768	1.0e-09	1.8e-12	0.86
Cube	*libfm*	32768	1.0e-11	6.9e-15	1.13
Cube	*libfm*	32768	1.0e-13	2.2e-15	2.12
Cube	*FMM-A*	262143	1.0e-03	1.4e-04	10.87
Cube	*FMM-A*	262143	1.0e-05	3.5e-06	14.71
Cube	*FMM-A*	262143	1.0e-07	2.6e-06	23.11
Cube	*FMM-A*	262143	1.0e-09	4.9e-07	40.00
Cube	*FMM-B*	262143	1.0e-04	2.2e-05	41.75
Cube	*FMM-B*	262143	1.0e-06	7.1e-08	50.84
Cube	*FMM-B*	262143	1.0e-08	1.7e-09	66.85
Cube	*libfm*	262143	1.0e-03	1.3e-04	2.20
Cube	*libfm*	262143	1.0e-05	2.0e-09	3.66
Cube	*libfm*	262143	1.0e-07	3.8e-11	4.38
Cube	*libfm*	262143	1.0e-09	1.3e-12	6.47
Cube	*libfm*	262143	1.0e-11	6.9e-15	9.81
Cube	*libfm*	262143	1.0e-13	3.9e-15	15.23
Cube	*FMM-A*	262144	1.0e-03	1.4e-04	103.61
Cube	*FMM-A*	262144	1.0e-05	3.5e-06	137.44
Cube	*FMM-A*	262144	1.0e-07	2.7e-06	211.93
Cube	*FMM-A*	262144	1.0e-09	5.0e-07	349.40
Cube	*FMM-B*	262144	1.0e-04	2.2e-05	35.41
Cube	*FMM-B*	262144	1.0e-06	7.1e-08	48.77
Cube	*FMM-B*	262144	1.0e-08	1.8e-09	63.04
Cube	*libfm*	262144	1.0e-03	1.3e-04	1.95
Cube	*libfm*	262144	1.0e-05	5.1e-09	3.19
Cube	*libfm*	262144	1.0e-07	3.4e-11	3.73
Cube	*libfm*	262144	1.0e-09	1.3e-12	6.40

continued on next page

Table A.1 – continued from previous page

Distribution	Method	N	Requested ΔE	Reached ΔE	$t\ [s]$
Cube	*libfm*	262144	1.0e-11	1.5e-14	9.60
Cube	*libfm*	262144	1.0e-13	1.3e-14	15.05
Cube	*FMM-A*	524287	1.0e-03	1.4e-04	126.51
Cube	*FMM-A*	524287	1.0e-05	3.5e-06	158.00
Cube	*FMM-A*	524287	1.0e-07	2.7e-06	234.02
Cube	*FMM-A*	524287	1.0e-09	5.0e-07	389.74
Cube	*FMM-B*	524287	1.0e-04	2.6e-05	128.79
Cube	*FMM-B*	524287	1.0e-06	8.3e-08	153.65
Cube	*FMM-B*	524287	1.0e-08	2.2e-09	198.45
Cube	*libfm*	524287	1.0e-03	1.9e-04	4.78
Cube	*libfm*	524287	1.0e-05	2.2e-08	6.30
Cube	*libfm*	524287	1.0e-07	2.0e-10	9.98
Cube	*libfm*	524287	1.0e-09	8.3e-13	12.86
Cube	*libfm*	524287	1.0e-11	2.0e-14	19.62
Cube	*libfm*	524287	1.0e-13	9.7e-16	31.28
Cube	*FMM-A*	1048576	1.0e-03	1.4e-04	134.11
Cube	*FMM-A*	1048576	1.0e-05	3.5e-06	167.93
Cube	*FMM-A*	1048576	1.0e-07	2.7e-06	246.55
Cube	*FMM-A*	1048576	1.0e-09	4.9e-07	386.75
Cube	*FMM-B*	1048576	1.0e-04	3.0e-05	143.63
Cube	*FMM-B*	1048576	1.0e-06	9.5e-08	201.20
Cube	*FMM-B*	1048576	1.0e-08	2.5e-09	305.46
Cube	*libfm*	1048576	1.0e-03	2.7e-05	10.71
Cube	*libfm*	1048576	1.0e-05	1.1e-08	12.90
Cube	*libfm*	1048576	1.0e-07	1.6e-10	18.48
Cube	*libfm*	1048576	1.0e-09	5.3e-13	28.92
Cube	*libfm*	1048576	1.0e-11	2.8e-14	44.39
Cube	*libfm*	1048576	1.0e-13	2.1e-15	63.73
Cube	*FMM-B*	2097152	1.0e-04	3.0e-05	288.28
Cube	*FMM-B*	2097152	1.0e-06	9.6e-08	413.73
Cube	*FMM-B*	2097152	1.0e-08	2.5e-09	545.37
Cube	*libfm*	2097152	1.0e-03	1.4e-04	18.99
Cube	*libfm*	2097152	1.0e-05	3.7e-09	29.05
Cube	*libfm*	2097152	1.0e-07	3.5e-11	38.33
Cube	*libfm*	2097152	1.0e-09	1.5e-12	55.12
Cube	*libfm*	2097152	1.0e-11	2.2e-15	82.65
Cube	*libfm*	2097152	1.0e-13	6.8e-15	123.66
Cube	*FMM-B*	8388608	1.0e-04	3.8e-05	1222.58
Cube	*libfm*	8388608	1.0e-03	2.7e-05	93.11
Cube	*libfm*	8388608	1.0e-05	1.4e-08	107.74
Cube	*libfm*	8388608	1.0e-07	1.9e-10	160.80
Cube	*libfm*	8388608	1.0e-09	6.2e-13	251.79
Cube	*libfm*	8388608	1.0e-11	1.1e-13	391.90
Cube	*libfm*	8388608	1.0e-13	3.9e-14	523.08
Cube	*libfm*	16777216	1.0e-03	not verified	162.88
Cube	*libfm*	16777216	1.0e-05	not verified	249.47
Cube	*libfm*	16777216	1.0e-07	not verified	315.42
Cube	*libfm*	16777216	1.0e-09	not verified	465.33

continued on next page

Table A.1 – continued from previous page

Distribution	Method	N	Requested ΔE	Reached ΔE	$t\,[s]$
Cube	*libfm*	16777216	1.0e-11	not verified	726.22
Cube	*libfm*	16777216	1.0e-13	not verified	1119.58
Cluster	*FMM-A*	114537	1.0e-03	2.6e-05	41.69
Cluster	*FMM-A*	114537	1.0e-05	1.6e-05	49.14
Cluster	*FMM-A*	114537	1.0e-07	7.3e-06	55.37
Cluster	*FMM-A*	114537	1.0e-09	2.5e-06	73.88
Cluster	*FMM-B*	114537	1.0e-04	5.9e-06	119.57
Cluster	*FMM-B*	114537	1.0e-06	2.4e-08	129.15
Cluster	*FMM-B*	114537	1.0e-08	9.0e-11	142.36
Cluster	*libfm*	114537	1.0e-03	7.8e-05	7.49
Cluster	*libfm*	114537	1.0e-05	1.3e-07	18.17
Cluster	*libfm*	114537	1.0e-07	2.1e-09	12.08
Cluster	*libfm*	114537	1.0e-09	7.1e-12	14.61
Cluster	*libfm*	114537	1.0e-11	2.2e-13	17.98
Cluster	*libfm*	114537	1.0e-13	2.1e-14	22.13

A.2 Periodic Boundary Conditions

The following tables A.2,A.3 and A.4 contain the results of all successfully performed test cases with 3D-periodic, 2D-periodic and 1D-periodic boundary conditions respectively. If a method did not exit normally or did not finish, then this case is not reported below. Thus there might be a result missing in one of the tables, where one might expect an entry. The first column in each table A.2,A.3 or A.4 shows the considered particle distribution as introduced in section 3.1.2. The following two columns report the used method and the size of the particle system. The next column contains a so-called precision control parameter ε, used to tune the parameters of the corresponding method for an accuracy goal. This parameter depends on the method. In the case of *libfm*, the precision control parameter ε represents the required relative energy error ΔE. The meaning of ε for the other methods is explained in the following sections. Finally the last two columns show the outcomes of the experiment defined by the previous columns, i.e. the reached accuracy ΔE and the measured computational time respectively.
To enable Coulomb interactions in the `ESPResSo` package and to define how these interactions are treated, one has to set up one of the methods of the `ESPResSo` package. Besides the choice of method, the necessary Bjerrum length λ_B, a scaling factor for the strength of the electrostatic interaction, has to be defined. All the simulations of each method provided by the `ESPResSo` package were performed using $\lambda_B = 1$.

A.2.1 Periodic Boundary Conditions - 3D

For the method P^3M of the `ESPResSo` package we used the automatic tuning feature `tunev2` to obtain a set of optimized parameters with respect to a required accuracy ε. Hence the precision control parameter ε in the following table A.2 represents the required RMS force error. We used the same values for all test scenarios, $\varepsilon \in \left\{10^{-7}, 10^{-5}, 10^{-3}\right\}$.

Table A.2: Benchmark results for particle systems with 3D periodic boundary conditions. The measured timings are rounded at two digits after the decimal dot.

Distribution	Method	N	Control Parameter	Reached ΔE	$t\,[s]$
3D Madelung	*libfm*	8	1.0e-03	6.9e-06	0.00
continued on next page					

Table A.2 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
3D Madelung	*libfm*	8	1.0e-05	1.2e-06	0.00
3D Madelung	*libfm*	8	1.0e-07	3.7e-10	0.00
3D Madelung	*libfm*	8	1.0e-09	7.0e-11	0.01
3D Madelung	*libfm*	8	1.0e-11	3.9e-14	0.00
3D Madelung	*libfm*	8	1.0e-13	6.6e-16	0.01
3D Madelung	P^3M	8	1.0e-03	8.8e-05	0.00
3D Madelung	P^3M	8	1.0e-05	5.3e-07	0.21
3D Madelung	*libfm*	64	1.0e-03	6.2e-06	0.00
3D Madelung	*libfm*	64	1.0e-05	6.6e-10	0.01
3D Madelung	*libfm*	64	1.0e-07	8.5e-10	0.01
3D Madelung	*libfm*	64	1.0e-09	2.1e-12	0.01
3D Madelung	*libfm*	64	1.0e-11	7.8e-15	0.02
3D Madelung	*libfm*	64	1.0e-13	1.3e-15	0.01
3D Madelung	P^3M	64	1.0e-03	1.1e-05	0.02
3D Madelung	P^3M	64	1.0e-05	4.3e-07	0.22
3D Madelung	*libfm*	512	1.0e-03	1.2e-08	0.03
3D Madelung	*libfm*	512	1.0e-05	5.0e-09	0.02
3D Madelung	*libfm*	512	1.0e-07	3.2e-10	0.04
3D Madelung	*libfm*	512	1.0e-09	9.3e-13	0.06
3D Madelung	*libfm*	512	1.0e-11	3.2e-15	0.09
3D Madelung	*libfm*	512	1.0e-13	5.5e-15	0.12
3D Madelung	P^3M	512	1.0e-03	6.4e-08	0.04
3D Madelung	P^3M	512	1.0e-05	9.8e-07	3.27
3D Madelung	*libfm*	1000	1.0e-03	2.6e-07	0.05
3D Madelung	*libfm*	1000	1.0e-05	1.2e-08	0.05
3D Madelung	*libfm*	1000	1.0e-07	3.7e-10	0.05
3D Madelung	*libfm*	1000	1.0e-09	2.7e-11	0.12
3D Madelung	*libfm*	1000	1.0e-11	7.5e-13	0.20
3D Madelung	*libfm*	1000	1.0e-13	4.5e-15	0.18
3D Madelung	P^3M	1000	1.0e-03	4.4e-06	0.06
3D Madelung	P^3M	1000	1.0e-05	5.3e-07	3.16
3D Madelung	*libfm*	2744	1.0e-03	8.3e-08	0.08
3D Madelung	*libfm*	2744	1.0e-05	4.8e-08	0.15
3D Madelung	*libfm*	2744	1.0e-07	1.8e-10	0.24
3D Madelung	*libfm*	2744	1.0e-09	5.8e-13	0.30
3D Madelung	*libfm*	2744	1.0e-11	3.7e-13	0.41
3D Madelung	*libfm*	2744	1.0e-13	1.5e-15	0.55
3D Madelung	P^3M	2744	1.0e-03	9.9e-07	0.27
3D Madelung	*libfm*	4096	1.0e-03	1.4e-07	0.10
3D Madelung	*libfm*	4096	1.0e-05	6.4e-09	0.17
3D Madelung	*libfm*	4096	1.0e-07	1.2e-10	0.30
3D Madelung	*libfm*	4096	1.0e-09	1.4e-13	0.56
3D Madelung	*libfm*	4096	1.0e-11	5.3e-15	0.74
3D Madelung	P^3M	4096	1.0e-03	4.9e-07	0.32
3D Madelung	*libfm*	8000	1.0e-03	2.6e-07	0.22
3D Madelung	*libfm*	8000	1.0e-05	2.9e-09	0.29
3D Madelung	*libfm*	8000	1.0e-07	1.3e-10	0.41
3D Madelung	*libfm*	8000	1.0e-09	3.9e-12	0.64

continued on next page

Table A.2 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
3D Madelung	*libfm*	8000	1.0e-11	2.3e-14	1.74
3D Madelung	P^3M	8000	1.0e-03	1.3e-06	0.89
3D Madelung	*libfm*	17576	1.0e-03	1.9e-07	0.48
3D Madelung	*libfm*	17576	1.0e-05	3.1e-09	0.89
3D Madelung	*libfm*	17576	1.0e-07	2.9e-10	1.07
3D Madelung	*libfm*	17576	1.0e-09	5.4e-12	1.50
3D Madelung	*libfm*	17576	1.0e-11	4.4e-14	2.18
3D Madelung	P^3M	17576	1.0e-03	1.6e-06	3.75
3D Madelung	*libfm*	32768	1.0e-03	7.8e-09	0.77
3D Madelung	*libfm*	32768	1.0e-05	2.2e-10	1.25
3D Madelung	*libfm*	32768	1.0e-07	1.1e-12	2.88
3D Madelung	*libfm*	32768	1.0e-09	4.7e-13	3.37
3D Madelung	*libfm*	32768	1.0e-11	1.7e-15	4.19
3D Madelung	P^3M	32768	1.0e-03	4.4e-07	5.22
3D Madelung	*libfm*	46656	1.0e-03	8.6e-08	1.09
3D Madelung	*libfm*	46656	1.0e-05	4.4e-09	1.69
3D Madelung	*libfm*	46656	1.0e-07	2.1e-10	3.09
3D Madelung	*libfm*	46656	1.0e-09	3.2e-13	5.83
3D Madelung	*libfm*	46656	1.0e-11	6.2e-15	6.73
3D Madelung	P^3M	46656	1.0e-03	1.6e-07	8.45
Cube	*libfm*	64	1.0e-03	2.0e-05	0.00
Cube	*libfm*	64	1.0e-05	1.7e-07	0.01
Cube	*libfm*	64	1.0e-07	1.3e-10	0.01
Cube	*libfm*	64	1.0e-09	2.1e-12	0.02
Cube	*libfm*	64	1.0e-11	1.5e-14	0.03
Cube	*libfm*	64	1.0e-13	6.1e-16	0.04
Cube	P^3M	64	1.0e-03	2.3e-05	0.02
Cube	P^3M	64	1.0e-05	1.8e-06	0.22
Cube	*libfm*	128	1.0e-03	1.1e-05	0.02
Cube	*libfm*	128	1.0e-05	2.1e-08	0.01
Cube	*libfm*	128	1.0e-07	2.4e-10	0.01
Cube	*libfm*	128	1.0e-09	6.1e-12	0.02
Cube	*libfm*	128	1.0e-11	1.1e-14	0.03
Cube	*libfm*	128	1.0e-13	3.6e-15	0.05
Cube	P^3M	128	1.0e-03	1.6e-07	0.02
Cube	P^3M	128	1.0e-05	2.9e-07	0.22
Cube	*libfm*	256	1.0e-03	9.9e-06	0.03
Cube	*libfm*	256	1.0e-05	1.2e-07	0.05
Cube	*libfm*	256	1.0e-07	6.3e-10	0.03
Cube	*libfm*	256	1.0e-09	3.4e-13	0.04
Cube	*libfm*	256	1.0e-11	7.6e-15	0.06
Cube	*libfm*	256	1.0e-13	3.0e-15	0.07
Cube	P^3M	256	1.0e-03	1.2e-06	0.03
Cube	P^3M	256	1.0e-05	1.8e-07	3.14
Cube	*libfm*	512	1.0e-03	1.8e-06	0.04
Cube	*libfm*	512	1.0e-05	2.4e-08	0.06
Cube	*libfm*	512	1.0e-07	2.2e-10	0.11
Cube	*libfm*	512	1.0e-09	6.1e-14	0.08

continued on next page

Table A.2 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
Cube	*libfm*	512	1.0e-11	6.8e-15	0.10
Cube	*libfm*	512	1.0e-13	2.2e-16	0.13
Cube	P^3M	512	1.0e-03	4.8e-07	0.03
Cube	P^3M	512	1.0e-05	1.5e-07	3.16
Cube	*libfm*	1024	1.0e-03	7.9e-07	0.07
Cube	*libfm*	1024	1.0e-05	2.6e-09	0.10
Cube	*libfm*	1024	1.0e-07	3.2e-10	0.17
Cube	*libfm*	1024	1.0e-09	2.4e-12	0.27
Cube	*libfm*	1024	1.0e-11	4.3e-14	0.46
Cube	P^3M	1024	1.0e-03	8.3e-07	0.06
Cube	P^3M	1024	1.0e-05	1.5e-07	3.30
Cube	*libfm*	4096	1.0e-03	3.8e-07	0.18
Cube	*libfm*	4096	1.0e-05	3.4e-09	0.31
Cube	*libfm*	4096	1.0e-07	1.3e-10	0.57
Cube	*libfm*	4096	1.0e-09	1.6e-13	0.73
Cube	*libfm*	4096	1.0e-11	7.0e-14	1.03
Cube	P^3M	4096	1.0e-03	5.3e-07	0.34
Cube	*libfm*	8192	1.0e-03	1.4e-07	0.35
Cube	*libfm*	8192	1.0e-05	1.9e-10	0.50
Cube	*libfm*	8192	1.0e-07	1.1e-10	0.84
Cube	*libfm*	8192	1.0e-09	2.0e-12	1.46
Cube	P^3M	8192	1.0e-03	6.4e-07	0.94
Cube	*libfm*	16384	1.0e-03	3.6e-08	0.71
Cube	*libfm*	16384	1.0e-05	4.0e-09	0.97
Cube	*libfm*	16384	1.0e-07	4.2e-11	1.32
Cube	*libfm*	16384	1.0e-09	4.4e-13	1.92
Cube	P^3M	16384	1.0e-03	5.7e-07	3.32
Cube	*libfm*	32768	1.0e-03	3.9e-08	1.17
Cube	*libfm*	32768	1.0e-05	9.2e-10	2.19
Cube	*libfm*	32768	1.0e-07	1.0e-11	3.29
Cube	*libfm*	32768	1.0e-09	9.4e-15	4.04
Cube	P^3M	32768	1.0e-03	8.1e-08	5.37
Cube	*libfm*	40000	1.0e-03	1.8e-08	1.44
Cube	*libfm*	40000	1.0e-05	9.9e-11	2.38
Cube	*libfm*	40000	1.0e-07	2.2e-11	4.30
Cube	*libfm*	40000	1.0e-09	7.0e-13	5.44
Cube	P^3M	40000	1.0e-03	5.1e-07	7.25

A.2.2 Periodic Boundary Conditions - 2D

For its internal particle data organization the method *MMM2D* of the `ESPResSo` package requires the so-called layered cell system. This cell system and therefore the performance of the method depends on the number of layers n_{layer}. The parameter needs to be tuned manually. For the remaining parameter we also used the automatic tuning feature to obtain a set of optimized parameters with respect to a required accuracy ε. Hence the precision control parameter ε in the following table A.3 represents the maximal pairwise error. We used the same values for all test scenarios, $\varepsilon \in \left\{10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}\right\}$. The value for n_{layer}, which needs to be set manually, was

chosen with respect to ε:

$$n_{\text{layer}} := \begin{cases} 24 & \text{, if} \quad \infty > \varepsilon > 10^{-3} \\ 10 & \text{, if} \quad 10^{-3} \geq \varepsilon > 10^{-7} \\ 5 & \text{, if} \quad 10^{-7} \geq \varepsilon > 0 \end{cases} . \tag{A.1}$$

However, this choice of n_{layer} is arbitrary, and therefore the obtained results might not be optimal. In order to perform simulations using *ELC*, we used *P*^{3}*M* again with `tuneV2` as a 3D periodic method. We used the precision control parameter ε for *ELC* as a maximal pairwise error and for *P*^{3}*M* as accuracy goal to tune the parameters. As for *MMM2D*, we used the same values for all test scenarios, $\varepsilon \in \{10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

Table A.3: Benchmark results for particle systems with 2D periodic boundary conditions. The measured timings are rounded at two digits after the decimal dot.

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
2D Madelung	*libfm*	4	1.0e-03	2.0e-05	0.00
2D Madelung	*libfm*	4	1.0e-05	3.4e-07	0.00
2D Madelung	*libfm*	4	1.0e-07	1.8e-09	0.00
2D Madelung	*libfm*	4	1.0e-09	1.0e-10	0.01
2D Madelung	*libfm*	4	1.0e-11	1.4e-12	0.01
2D Madelung	*libfm*	4	1.0e-13	2.4e-16	0.01
2D Madelung	*MMM2D*	4	1.0e-01	1.4e-04	0.07
2D Madelung	*MMM2D*	4	1.0e-02	4.9e-05	0.10
2D Madelung	*MMM2D*	4	1.0e-03	6.6e-06	0.00
2D Madelung	*MMM2D*	4	1.0e-05	4.2e-08	0.00
2D Madelung	*MMM2D*	4	1.0e-07	1.2e-07	0.00
2D Madelung	*ELC*/*P*^{3}*M*	4	1.0e-01	1.4e-02	0.00
2D Madelung	*ELC*/*P*^{3}*M*	4	1.0e-02	1.9e-03	0.00
2D Madelung	*ELC*/*P*^{3}*M*	4	1.0e-03	1.5e-04	0.00
2D Madelung	*ELC*/*P*^{3}*M*	4	1.0e-05	8.7e-07	0.22
2D Madelung	*libfm*	64	1.0e-03	2.7e-05	0.01
2D Madelung	*libfm*	64	1.0e-05	1.3e-09	0.00
2D Madelung	*libfm*	64	1.0e-07	3.7e-11	0.01
2D Madelung	*libfm*	64	1.0e-09	2.6e-13	0.01
2D Madelung	*libfm*	64	1.0e-11	6.5e-14	0.01
2D Madelung	*libfm*	64	1.0e-13	3.8e-16	0.02
2D Madelung	*MMM2D*	64	1.0e-01	1.5e-04	0.34
2D Madelung	*MMM2D*	64	1.0e-02	2.4e-05	0.48
2D Madelung	*MMM2D*	64	1.0e-03	3.1e-06	0.02
2D Madelung	*MMM2D*	64	1.0e-05	7.7e-09	0.03
2D Madelung	*MMM2D*	64	1.0e-07	1.1e-10	0.01
2D Madelung	*ELC*/*P*^{3}*M*	64	1.0e-01	6.1e-05	0.00
2D Madelung	*ELC*/*P*^{3}*M*	64	1.0e-02	1.0e-05	0.01
2D Madelung	*ELC*/*P*^{3}*M*	64	1.0e-03	3.1e-06	0.03
2D Madelung	*ELC*/*P*^{3}*M*	64	1.0e-05	5.0e-07	0.23
2D Madelung	*libfm*	576	1.0e-03	6.0e-07	0.02
2D Madelung	*libfm*	576	1.0e-05	6.8e-08	0.02
2D Madelung	*libfm*	576	1.0e-07	1.8e-09	0.04
2D Madelung	*libfm*	576	1.0e-09	6.8e-11	0.08
2D Madelung	*libfm*	576	1.0e-11	5.7e-13	0.13

continued on next page

Table A.3 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
2D Madelung	*libfm*	576	1.0e-13	3.0e-16	0.05
2D Madelung	*MMM2D*	576	1.0e-01	4.7e-05	3.08
2D Madelung	*MMM2D*	576	1.0e-02	8.1e-06	4.16
2D Madelung	*MMM2D*	576	1.0e-03	3.9e-07	0.66
2D Madelung	*MMM2D*	576	1.0e-05	1.9e-09	0.95
2D Madelung	*MMM2D*	576	1.0e-07	1.3e-10	0.86
2D Madelung	*ELC/P^{3}M*	576	1.0e-01	1.6e-04	0.03
2D Madelung	*ELC/P^{3}M*	576	1.0e-02	2.6e-06	0.05
2D Madelung	*ELC/P^{3}M*	576	1.0e-03	3.6e-07	0.08
2D Madelung	*ELC/P^{3}M*	576	1.0e-05	3.0e-07	3.38
2D Madelung	*libfm*	1024	1.0e-03	1.0e-06	0.05
2D Madelung	*libfm*	1024	1.0e-05	3.7e-08	0.04
2D Madelung	*libfm*	1024	1.0e-07	6.3e-10	0.07
2D Madelung	*libfm*	1024	1.0e-09	1.0e-11	0.12
2D Madelung	*libfm*	1024	1.0e-11	6.0e-16	0.19
2D Madelung	*MMM2D*	1024	1.0e-01	3.3e-05	5.98
2D Madelung	*MMM2D*	1024	1.0e-02	4.8e-06	7.91
2D Madelung	*MMM2D*	1024	1.0e-03	1.6e-08	1.93
2D Madelung	*MMM2D*	1024	1.0e-05	1.4e-08	2.77
2D Madelung	*MMM2D*	1024	1.0e-07	1.4e-10	2.67
2D Madelung	*ELC/P^{3}M*	1024	1.0e-01	3.6e-05	0.07
2D Madelung	*ELC/P^{3}M*	1024	1.0e-02	7.0e-06	0.12
2D Madelung	*ELC/P^{3}M*	1024	1.0e-03	2.2e-07	0.15
2D Madelung	*ELC/P^{3}M*	1024	1.0e-05	3.0e-07	3.37
2D Madelung	*libfm*	2916	1.0e-03	1.1e-07	0.07
2D Madelung	*libfm*	2916	1.0e-05	1.3e-08	0.10
2D Madelung	*libfm*	2916	1.0e-07	2.4e-10	0.13
2D Madelung	*libfm*	2916	1.0e-09	4.4e-12	0.20
2D Madelung	*libfm*	2916	1.0e-11	2.2e-13	0.31
2D Madelung	*MMM2D*	2916	1.0e-01	4.0e-05	23.31
2D Madelung	*MMM2D*	2916	1.0e-02	3.0e-06	29.24
2D Madelung	*MMM2D*	2916	1.0e-03	5.3e-07	14.76
2D Madelung	*MMM2D*	2916	1.0e-05	1.9e-09	20.84
2D Madelung	*MMM2D*	2916	1.0e-07	1.2e-11	21.29
2D Madelung	*ELC/P^{3}M*	2916	1.0e-01	1.6e-05	0.36
2D Madelung	*ELC/P^{3}M*	2916	1.0e-02	2.7e-06	0.45
2D Madelung	*ELC/P^{3}M*	2916	1.0e-03	2.2e-07	0.55
2D Madelung	*libfm*	4096	1.0e-03	1.0e-06	0.09
2D Madelung	*libfm*	4096	1.0e-05	1.0e-08	0.11
2D Madelung	*libfm*	4096	1.0e-07	4.0e-10	0.18
2D Madelung	*libfm*	4096	1.0e-09	7.6e-12	0.24
2D Madelung	*MMM2D*	4096	1.0e-01	1.5e-05	37.98
2D Madelung	*MMM2D*	4096	1.0e-02	2.7e-06	46.70
2D Madelung	*MMM2D*	4096	1.0e-03	2.9e-07	28.79
2D Madelung	*MMM2D*	4096	1.0e-05	4.8e-09	40.64
2D Madelung	*MMM2D*	4096	1.0e-07	4.7e-11	41.85
2D Madelung	*ELC/P^{3}M*	4096	1.0e-01	7.8e-06	0.46
2D Madelung	*ELC/P^{3}M*	4096	1.0e-02	2.1e-06	0.59

continued on next page

Table A.3 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
2D Madelung	*ELC/P^{3}M*	4096	1.0e-03	3.2e-07	0.76
2D Madelung	*libfm*	8100	1.0e-03	9.1e-07	0.13
2D Madelung	*libfm*	8100	1.0e-05	1.5e-08	0.10
2D Madelung	*libfm*	8100	1.0e-07	5.8e-11	0.28
2D Madelung	*libfm*	8100	1.0e-09	9.4e-13	0.40
2D Madelung	*libfm*	8100	1.0e-11	3.2e-14	0.64
2D Madelung	*MMM2D*	8100	1.0e-01	1.3e-05	112.01
2D Madelung	*MMM2D*	8100	1.0e-02	5.2e-06	132.05
2D Madelung	*MMM2D*	8100	1.0e-03	1.1e-07	111.38
2D Madelung	*MMM2D*	8100	1.0e-05	1.8e-09	156.72
2D Madelung	*MMM2D*	8100	1.0e-07	1.1e-10	163.10
2D Madelung	*ELC/P^{3}M*	8100	1.0e-01	2.5e-06	0.86
2D Madelung	*ELC/P^{3}M*	8100	1.0e-02	1.3e-07	1.16
2D Madelung	*ELC/P^{3}M*	8100	1.0e-03	4.5e-08	2.21
2D Madelung	*libfm*	17956	1.0e-03	1.9e-07	0.26
2D Madelung	*libfm*	17956	1.0e-05	1.1e-08	0.28
2D Madelung	*libfm*	17956	1.0e-07	2.4e-10	0.54
2D Madelung	*libfm*	17956	1.0e-09	1.3e-12	0.71
2D Madelung	*MMM2D*	17956	1.0e-01	5.6e-06	452.86
2D Madelung	*MMM2D*	17956	1.0e-02	3.6e-06	513.07
2D Madelung	*MMM2D*	17956	1.0e-03	1.5e-08	543.09
2D Madelung	*MMM2D*	17956	1.0e-05	7.3e-09	763.68
2D Madelung	*MMM2D*	17956	1.0e-07	5.3e-11	801.00
2D Madelung	*ELC/P^{3}M*	17956	1.0e-01	6.3e-06	2.65
2D Madelung	*ELC/P^{3}M*	17956	1.0e-02	5.8e-07	3.99
2D Madelung	*ELC/P^{3}M*	17956	1.0e-03	7.8e-08	6.77
2D Madelung	*libfm*	32400	1.0e-03	4.0e-07	0.40
2D Madelung	*libfm*	32400	1.0e-05	2.8e-09	0.51
2D Madelung	*libfm*	32400	1.0e-07	3.3e-11	0.77
2D Madelung	*libfm*	32400	1.0e-09	4.5e-13	1.28
2D Madelung	*MMM2D*	32400	1.0e-01	2.3e-05	1357.42
2D Madelung	*MMM2D*	32400	1.0e-02	4.7e-06	1508.97
2D Madelung	*MMM2D*	32400	1.0e-03	2.1e-07	1772.17
2D Madelung	*MMM2D*	32400	1.0e-05	1.2e-08	2485.86
2D Madelung	*MMM2D*	32400	1.0e-07	1.1e-10	2612.26
2D Madelung	*ELC/P^{3}M*	32400	1.0e-03	6.3e-08	13.36
2D Madelung	*libfm*	40000	1.0e-03	2.0e-07	0.46
2D Madelung	*libfm*	40000	1.0e-05	1.2e-10	0.82
2D Madelung	*libfm*	40000	1.0e-07	6.0e-11	0.98
2D Madelung	*libfm*	40000	1.0e-09	1.9e-13	1.48
2D Madelung	*MMM2D*	40000	1.0e-01	2.7e-05	2021.91
2D Madelung	*MMM2D*	40000	1.0e-02	4.0e-06	2232.89
2D Madelung	*MMM2D*	40000	1.0e-03	1.9e-07	2703.13
2D Madelung	*MMM2D*	40000	1.0e-05	7.4e-09	3790.86
2D Madelung	*MMM2D*	40000	1.0e-07	2.7e-12	3988.07
Cube	*libfm*	64	1.0e-03	1.6e-05	0.00
Cube	*libfm*	64	1.0e-05	6.8e-08	0.01
Cube	*libfm*	64	1.0e-07	1.6e-10	0.01

continued on next page

Table A.3 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
Cube	*libfm*	64	1.0e-09	3.1e-12	0.02
Cube	*libfm*	64	1.0e-11	1.4e-14	0.02
Cube	*libfm*	64	1.0e-13	3.0e-17	0.04
Cube	*MMM2D*	64	1.0e-01	3.3e-05	0.34
Cube	*MMM2D*	64	1.0e-02	4.1e-06	0.47
Cube	*MMM2D*	64	1.0e-03	9.9e-07	0.01
Cube	*MMM2D*	64	1.0e-05	1.9e-08	0.02
Cube	*MMM2D*	64	1.0e-07	3.3e-09	0.01
Cube	*ELC/P^{3}M*	64	1.0e-01	6.6e-04	0.00
Cube	*ELC/P^{3}M*	64	1.0e-02	1.3e-04	0.01
Cube	*ELC/P^{3}M*	64	1.0e-03	1.1e-05	0.03
Cube	*ELC/P^{3}M*	64	1.0e-05	1.4e-06	0.24
Cube	*libfm*	128	1.0e-03	2.6e-06	0.01
Cube	*libfm*	128	1.0e-05	5.0e-08	0.01
Cube	*libfm*	128	1.0e-07	1.3e-11	0.01
Cube	*libfm*	128	1.0e-09	3.8e-12	0.02
Cube	*libfm*	128	1.0e-11	3.4e-14	0.03
Cube	*libfm*	128	1.0e-13	2.7e-15	0.05
Cube	*MMM2D*	128	1.0e-01	9.2e-05	0.63
Cube	*MMM2D*	128	1.0e-02	1.8e-05	0.88
Cube	*MMM2D*	128	1.0e-03	1.0e-06	0.03
Cube	*MMM2D*	128	1.0e-05	1.8e-08	0.05
Cube	*MMM2D*	128	1.0e-07	1.2e-08	0.03
Cube	*ELC/P^{3}M*	128	1.0e-01	4.5e-04	0.01
Cube	*ELC/P^{3}M*	128	1.0e-02	1.3e-04	0.01
Cube	*ELC/P^{3}M*	128	1.0e-03	1.7e-04	0.03
Cube	*ELC/P^{3}M*	128	1.0e-05	5.3e-05	0.24
Cube	*libfm*	256	1.0e-03	1.3e-06	0.03
Cube	*libfm*	256	1.0e-05	4.9e-09	0.04
Cube	*libfm*	256	1.0e-07	2.3e-10	0.03
Cube	*libfm*	256	1.0e-09	1.3e-11	0.04
Cube	*libfm*	256	1.0e-11	1.2e-13	0.05
Cube	*MMM2D*	256	1.0e-01	1.2e-04	1.25
Cube	*MMM2D*	256	1.0e-02	6.7e-07	1.73
Cube	*MMM2D*	256	1.0e-03	1.1e-05	0.07
Cube	*MMM2D*	256	1.0e-05	1.1e-07	0.12
Cube	*MMM2D*	256	1.0e-07	6.6e-08	0.10
Cube	*ELC/P^{3}M*	256	1.0e-01	5.6e-03	0.01
Cube	*ELC/P^{3}M*	256	1.0e-02	5.0e-03	0.02
Cube	*ELC/P^{3}M*	256	1.0e-03	2.5e-03	0.05
Cube	*ELC/P^{3}M*	256	1.0e-05	2.1e-03	3.25
Cube	*libfm*	512	1.0e-03	3.1e-07	0.03
Cube	*libfm*	512	1.0e-05	1.4e-08	0.04
Cube	*libfm*	512	1.0e-07	4.9e-10	0.09
Cube	*libfm*	512	1.0e-09	2.7e-12	0.06
Cube	*libfm*	512	1.0e-11	2.0e-13	0.09
Cube	*ELC/P^{3}M*	512	1.0e-01	9.5e-02	0.03
Cube	*ELC/P^{3}M*	512	1.0e-02	9.7e-02	0.04

continued on next page

Table A.3 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
Cube	*ELC/P^{3}M*	512	1.0e-03	9.7e-02	0.07
Cube	*ELC/P^{3}M*	512	1.0e-05	9.8e-02	3.31
Cube	*libfm*	1024	1.0e-03	6.6e-08	0.07
Cube	*libfm*	1024	1.0e-05	1.6e-08	0.07
Cube	*libfm*	1024	1.0e-07	3.4e-11	0.11
Cube	*libfm*	1024	1.0e-09	1.1e-11	0.17
Cube	*libfm*	1024	1.0e-11	1.8e-13	0.30
Cube	*MMM2D*	1024	1.0e-01	9.1e-06	4.80
Cube	*MMM2D*	1024	1.0e-02	9.5e-06	6.63
Cube	*MMM2D*	1024	1.0e-03	1.8e-08	0.65
Cube	*MMM2D*	1024	1.0e-05	6.0e-09	0.97
Cube	*MMM2D*	1024	1.0e-07	3.8e-09	1.42
Cube	*ELC/P^{3}M*	1024	1.0e-01	1.0e-03	0.07
Cube	*ELC/P^{3}M*	1024	1.0e-02	8.7e-05	0.11
Cube	*ELC/P^{3}M*	1024	1.0e-03	8.7e-04	0.14
Cube	*ELC/P^{3}M*	1024	1.0e-05	1.8e-04	3.47
Cube	*libfm*	4096	1.0e-03	1.1e-06	0.13
Cube	*libfm*	4096	1.0e-05	3.9e-09	0.27
Cube	*libfm*	4096	1.0e-07	1.7e-10	0.45
Cube	*libfm*	4096	1.0e-09	2.7e-12	0.51
Cube	*libfm*	4096	1.0e-11	2.7e-15	0.72
Cube	*MMM2D*	4096	1.0e-01	2.9e-05	20.66
Cube	*MMM2D*	4096	1.0e-02	7.1e-07	27.98
Cube	*MMM2D*	4096	1.0e-03	9.0e-07	8.21
Cube	*MMM2D*	4096	1.0e-05	2.6e-08	11.88
Cube	*MMM2D*	4096	1.0e-07	9.1e-09	21.63
Cube	*ELC/P^{3}M*	4096	1.0e-01	6.7e-03	0.40
Cube	*ELC/P^{3}M*	4096	1.0e-02	5.9e-03	0.50
Cube	*ELC/P^{3}M*	4096	1.0e-03	2.4e-03	0.64
Cube	*libfm*	8192	1.0e-03	7.7e-08	0.27
Cube	*libfm*	8192	1.0e-05	4.1e-09	0.40
Cube	*libfm*	8192	1.0e-07	4.1e-11	0.60
Cube	*libfm*	8192	1.0e-09	4.2e-13	0.91
Cube	*libfm*	8192	1.0e-11	5.1e-15	1.42
Cube	*MMM2D*	8192	1.0e-01	1.0e-06	45.74
Cube	*MMM2D*	8192	1.0e-02	2.3e-05	60.88
Cube	*MMM2D*	8192	1.0e-03	5.2e-08	31.25
Cube	*MMM2D*	8192	1.0e-05	8.9e-09	44.92
Cube	*MMM2D*	8192	1.0e-07	1.2e-08	85.71
Cube	*ELC/P^{3}M*	8192	1.0e-01	1.2e-02	0.63
Cube	*ELC/P^{3}M*	8192	1.0e-02	5.6e-03	0.85
Cube	*ELC/P^{3}M*	8192	1.0e-03	9.2e-03	1.54
Cube	*libfm*	16384	1.0e-03	8.3e-09	0.69
Cube	*libfm*	16384	1.0e-05	8.8e-11	0.79
Cube	*libfm*	16384	1.0e-07	3.1e-11	1.05
Cube	*libfm*	16384	1.0e-09	7.5e-13	1.53
Cube	*ELC/P^{3}M*	16384	1.0e-01	1.3e-02	1.25
Cube	*ELC/P^{3}M*	16384	1.0e-02	1.1e-02	1.87

continued on next page

Table A.3 – continued from previous page

Distribution	Method	N	Control Parameter	Reached ΔE	$t\ [s]$
Cube	*ELC/P^{3}M*	16384	1.0e-03	1.1e-02	4.59
Cube	*libfm*	32768	1.0e-03	3.9e-08	0.94
Cube	*libfm*	32768	1.0e-05	3.2e-10	1.64
Cube	*libfm*	32768	1.0e-07	4.2e-12	3.00
Cube	*libfm*	32768	1.0e-09	3.5e-13	3.15
Cube	*MMM2D*	32768	1.0e-01	6.3e-06	306.34
Cube	*MMM2D*	32768	1.0e-02	1.4e-07	383.99
Cube	*MMM2D*	32768	1.0e-03	1.6e-06	485.89
Cube	*MMM2D*	32768	1.0e-05	2.8e-09	693.80
Cube	*MMM2D*	32768	1.0e-07	4.5e-09	1360.79
Cube	*ELC/P^{3}M*	32768	1.0e-01	2.1e-04	3.20
Cube	*ELC/P^{3}M*	32768	1.0e-02	2.6e-04	5.14
Cube	*ELC/P^{3}M*	32768	1.0e-03	1.5e-03	8.09
Cube	*libfm*	40000	1.0e-03	4.6e-08	1.42
Cube	*libfm*	40000	1.0e-05	1.6e-09	2.62
Cube	*libfm*	40000	1.0e-07	2.6e-11	4.59
Cube	*libfm*	40000	1.0e-09	1.4e-12	4.62
Cube	*MMM2D*	40000	1.0e-01	4.4e-04	409.05
Cube	*MMM2D*	40000	1.0e-02	1.2e-04	503.42
Cube	*MMM2D*	40000	1.0e-03	2.7e-06	719.80
Cube	*MMM2D*	40000	1.0e-05	1.3e-08	1028.20
Cube	*MMM2D*	40000	1.0e-07	7.9e-08	2022.17
Cube	*ELC/P^{3}M*	40000	1.0e-01	5.2e-02	4.28
Cube	*ELC/P^{3}M*	40000	1.0e-03	3.0e-02	10.57

A.2.3 Periodic Boundary Conditions - 1D

The method *MMM1D* of the `ESPResSo` package requires for its internal particle data organization the so-called N-squared cell system, which is independent of any additional parameter. Thus a full automatic tuning feature (`tune`) is available. The precision control parameter ε is shown in table A.4. It is used to specify the required maximal pairwise error for the tuning process. We used the same values for all test scenarios, $\varepsilon \in \left\{10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}\right\}$.

Table A.4: Benchmark results for particle systems with 1D periodic boundary conditions. The measured timings are rounded at two digits after the decimal dot.

Distribution	Method	N	control parameter	reached ΔE	$t\ [s]$
1D Madelung	*libfm*	2	1.0e-03	1.4e-05	0.00
1D Madelung	*libfm*	2	1.0e-05	1.2e-08	0.00
1D Madelung	*libfm*	2	1.0e-07	1.3e-10	0.00
1D Madelung	*libfm*	2	1.0e-09	3.9e-12	0.00
1D Madelung	*libfm*	2	1.0e-11	5.8e-14	0.00
1D Madelung	*libfm*	2	1.0e-13	6.1e-16	0.00
1D Madelung	*MMM1D*	2	1.0e-01	1.5e-15	0.00
1D Madelung	*MMM1D*	2	1.0e-02	1.5e-15	0.00
1D Madelung	*MMM1D*	2	1.0e-03	1.5e-15	0.00
1D Madelung	*MMM1D*	2	1.0e-05	1.5e-15	0.00
1D Madelung	*MMM1D*	2	1.0e-07	1.5e-15	0.00

continued on next page

Table A.4 – continued from previous page

Distribution	Method	N	control parameter	reached ΔE	$t\ [s]$
1D Madelung	*libfm*	8	1.0e-03	5.3e-05	0.00
1D Madelung	*libfm*	8	1.0e-05	4.3e-07	0.00
1D Madelung	*libfm*	8	1.0e-07	3.6e-08	0.01
1D Madelung	*libfm*	8	1.0e-09	2.5e-10	0.01
1D Madelung	*libfm*	8	1.0e-11	1.7e-12	0.01
1D Madelung	*libfm*	8	1.0e-13	1.2e-14	0.01
1D Madelung	*MMM1D*	8	1.0e-01	4.5e-16	0.00
1D Madelung	*MMM1D*	8	1.0e-02	4.5e-16	0.00
1D Madelung	*MMM1D*	8	1.0e-03	4.5e-16	0.00
1D Madelung	*MMM1D*	8	1.0e-05	4.5e-16	0.00
1D Madelung	*MMM1D*	8	1.0e-07	4.5e-16	0.00
1D Madelung	*libfm*	32	1.0e-03	1.5e-06	0.01
1D Madelung	*libfm*	32	1.0e-05	5.1e-08	0.00
1D Madelung	*libfm*	32	1.0e-07	5.6e-09	0.01
1D Madelung	*libfm*	32	1.0e-09	6.5e-11	0.01
1D Madelung	*libfm*	32	1.0e-11	7.0e-12	0.01
1D Madelung	*libfm*	32	1.0e-13	8.0e-14	0.01
1D Madelung	*MMM1D*	32	1.0e-01	4.2e-16	0.00
1D Madelung	*MMM1D*	32	1.0e-02	4.2e-16	0.00
1D Madelung	*MMM1D*	32	1.0e-03	4.2e-16	0.00
1D Madelung	*MMM1D*	32	1.0e-05	4.2e-16	0.00
1D Madelung	*MMM1D*	32	1.0e-07	4.2e-16	0.00
1D Madelung	*libfm*	512	1.0e-03	1.8e-06	0.04
1D Madelung	*libfm*	512	1.0e-05	7.0e-09	0.05
1D Madelung	*libfm*	512	1.0e-07	5.2e-11	0.08
1D Madelung	*libfm*	512	1.0e-09	3.8e-12	0.12
1D Madelung	*libfm*	512	1.0e-11	3.1e-14	0.02
1D Madelung	*libfm*	512	1.0e-13	1.3e-16	0.02
1D Madelung	*MMM1D*	512	1.0e-01	6.3e-14	0.06
1D Madelung	*MMM1D*	512	1.0e-02	6.3e-14	0.06
1D Madelung	*MMM1D*	512	1.0e-03	6.3e-14	0.06
1D Madelung	*MMM1D*	512	1.0e-05	6.3e-14	0.06
1D Madelung	*MMM1D*	512	1.0e-07	6.3e-14	0.06
1D Madelung	*libfm*	2048	1.0e-03	2.0e-06	0.05
1D Madelung	*libfm*	2048	1.0e-05	4.8e-08	0.08
1D Madelung	*libfm*	2048	1.0e-07	6.3e-10	0.12
1D Madelung	*libfm*	2048	1.0e-09	5.0e-12	0.17
1D Madelung	*MMM1D*	2048	1.0e-01	3.4e-13	0.88
1D Madelung	*MMM1D*	2048	1.0e-02	3.4e-13	0.88
1D Madelung	*MMM1D*	2048	1.0e-03	3.4e-13	0.89
1D Madelung	*MMM1D*	2048	1.0e-05	3.4e-13	0.89
1D Madelung	*MMM1D*	2048	1.0e-07	3.4e-13	0.89
1D Madelung	*libfm*	4096	1.0e-03	2.0e-06	0.07
1D Madelung	*libfm*	4096	1.0e-05	6.7e-08	0.12
1D Madelung	*libfm*	4096	1.0e-07	6.3e-10	0.17
1D Madelung	*libfm*	4096	1.0e-09	5.0e-12	0.25
1D Madelung	*MMM1D*	4096	1.0e-01	1.0e-12	3.54
1D Madelung	*MMM1D*	4096	1.0e-02	1.0e-12	3.54

continued on next page

Table A.4 – continued from previous page

Distribution	Method	N	control parameter	reached ΔE	$t\ [s]$
1D Madelung	*MMM1D*	4096	1.0e-03	1.0e-12	3.54
1D Madelung	*MMM1D*	4096	1.0e-05	1.0e-12	3.54
1D Madelung	*MMM1D*	4096	1.0e-07	1.0e-12	3.55
1D Madelung	*libfm*	8192	1.0e-03	6.5e-06	0.12
1D Madelung	*libfm*	8192	1.0e-05	6.7e-08	0.17
1D Madelung	*libfm*	8192	1.0e-07	6.3e-10	0.19
1D Madelung	*libfm*	8192	1.0e-09	2.0e-12	0.37
1D Madelung	*MMM1D*	8192	1.0e-01	3.3e-13	14.22
1D Madelung	*MMM1D*	8192	1.0e-02	3.3e-13	14.21
1D Madelung	*MMM1D*	8192	1.0e-03	3.3e-13	14.20
1D Madelung	*MMM1D*	8192	1.0e-05	3.3e-13	14.18
1D Madelung	*MMM1D*	8192	1.0e-07	3.3e-13	14.19
1D Madelung	*libfm*	16384	1.0e-03	6.5e-06	0.19
1D Madelung	*libfm*	16384	1.0e-05	1.7e-08	0.20
1D Madelung	*libfm*	16384	1.0e-07	1.6e-10	0.36
1D Madelung	*libfm*	16384	1.0e-09	2.0e-12	0.50
1D Madelung	*MMM1D*	16384	1.0e-01	1.6e-11	56.82
1D Madelung	*MMM1D*	16384	1.0e-02	1.6e-11	56.32
1D Madelung	*MMM1D*	16384	1.0e-03	1.6e-11	56.77
1D Madelung	*MMM1D*	16384	1.0e-05	1.6e-11	56.83
1D Madelung	*MMM1D*	16384	1.0e-07	1.6e-11	56.86
1D Madelung	*libfm*	32768	1.0e-03	6.5e-06	0.47
1D Madelung	*libfm*	32768	1.0e-05	1.7e-08	0.41
1D Madelung	*libfm*	32768	1.0e-07	1.1e-10	0.61
1D Madelung	*libfm*	32768	1.0e-09	2.0e-12	1.04
1D Madelung	*MMM1D*	32768	1.0e-01	5.2e-11	227.60
1D Madelung	*MMM1D*	32768	1.0e-02	5.2e-11	228.94
1D Madelung	*MMM1D*	32768	1.0e-03	5.2e-11	226.75
1D Madelung	*MMM1D*	32768	1.0e-05	5.2e-11	224.02
1D Madelung	*MMM1D*	32768	1.0e-07	5.2e-11	225.25
1D Madelung	*libfm*	40000	1.0e-03	1.0e-07	0.53
1D Madelung	*libfm*	40000	1.0e-05	2.1e-08	0.58
1D Madelung	*libfm*	40000	1.0e-07	1.8e-11	0.98
1D Madelung	*libfm*	40000	1.0e-09	7.1e-13	1.28
1D Madelung	*MMM1D*	40000	1.0e-01	4.5e-11	334.90
1D Madelung	*MMM1D*	40000	1.0e-02	4.5e-11	334.35
1D Madelung	*MMM1D*	40000	1.0e-03	4.5e-11	338.47
1D Madelung	*MMM1D*	40000	1.0e-05	4.5e-11	338.51
1D Madelung	*MMM1D*	40000	1.0e-07	4.5e-11	339.92
Cube	*libfm*	64	1.0e-03	7.5e-06	0.00
Cube	*libfm*	64	1.0e-05	1.7e-08	0.00
Cube	*libfm*	64	1.0e-07	1.0e-09	0.00
Cube	*libfm*	64	1.0e-09	1.9e-11	0.02
Cube	*libfm*	64	1.0e-11	5.6e-14	0.01
Cube	*libfm*	64	1.0e-13	2.0e-15	0.04
Cube	*MMM1D*	64	1.0e-01	5.0e-06	0.01
Cube	*MMM1D*	64	1.0e-02	2.6e-07	0.01
Cube	*MMM1D*	64	1.0e-03	2.4e-07	0.00

continued on next page

Table A.4 – continued from previous page

Distribution	Method	N	control parameter	reached ΔE	$t\ [s]$
Cube	*MMM1D*	64	1.0e-05	8.7e-10	0.01
Cube	*MMM1D*	64	1.0e-07	7.4e-11	0.01
Cube	*libfm*	128	1.0e-03	4.0e-06	0.01
Cube	*libfm*	128	1.0e-05	1.9e-09	0.00
Cube	*libfm*	128	1.0e-07	8.0e-10	0.01
Cube	*libfm*	128	1.0e-09	9.4e-13	0.01
Cube	*libfm*	128	1.0e-11	3.1e-14	0.03
Cube	*libfm*	128	1.0e-13	8.0e-16	0.03
Cube	*MMM1D*	128	1.0e-01	7.2e-05	0.00
Cube	*MMM1D*	128	1.0e-02	1.4e-05	0.00
Cube	*MMM1D*	128	1.0e-03	1.0e-05	0.01
Cube	*MMM1D*	128	1.0e-05	6.6e-08	0.01
Cube	*MMM1D*	128	1.0e-07	6.5e-09	0.01
Cube	*libfm*	256	1.0e-03	3.1e-06	0.01
Cube	*libfm*	256	1.0e-05	2.3e-08	0.02
Cube	*libfm*	256	1.0e-07	7.3e-11	0.01
Cube	*libfm*	256	1.0e-09	8.2e-13	0.01
Cube	*libfm*	256	1.0e-11	6.2e-14	0.02
Cube	*libfm*	256	1.0e-13	1.2e-15	0.07
Cube	*MMM1D*	256	1.0e-01	1.1e-03	0.01
Cube	*MMM1D*	256	1.0e-02	7.3e-06	0.02
Cube	*MMM1D*	256	1.0e-03	5.5e-06	0.02
Cube	*MMM1D*	256	1.0e-05	1.7e-07	0.03
Cube	*MMM1D*	256	1.0e-07	1.2e-10	0.04
Cube	*libfm*	512	1.0e-03	2.3e-06	0.04
Cube	*libfm*	512	1.0e-05	1.6e-09	0.03
Cube	*libfm*	512	1.0e-07	8.9e-11	0.09
Cube	*libfm*	512	1.0e-09	4.3e-12	0.05
Cube	*libfm*	512	1.0e-11	8.1e-14	0.09
Cube	*MMM1D*	512	1.0e-01	1.9e-01	0.06
Cube	*MMM1D*	512	1.0e-02	1.9e-02	0.08
Cube	*MMM1D*	512	1.0e-03	2.3e-03	0.08
Cube	*MMM1D*	512	1.0e-05	3.4e-05	0.11
Cube	*MMM1D*	512	1.0e-07	1.4e-07	0.13
Cube	*libfm*	1024	1.0e-03	1.4e-07	0.06
Cube	*libfm*	1024	1.0e-05	1.1e-08	0.06
Cube	*libfm*	1024	1.0e-07	3.8e-10	0.10
Cube	*libfm*	1024	1.0e-09	5.7e-13	0.09
Cube	*libfm*	1024	1.0e-11	2.0e-13	0.11
Cube	*MMM1D*	1024	1.0e-01	3.3e-04	0.23
Cube	*MMM1D*	1024	1.0e-02	4.5e-05	0.30
Cube	*MMM1D*	1024	1.0e-03	3.1e-06	0.32
Cube	*MMM1D*	1024	1.0e-05	1.1e-07	0.43
Cube	*MMM1D*	1024	1.0e-07	8.1e-10	0.53
Cube	*libfm*	4096	1.0e-03	3.4e-08	0.16
Cube	*libfm*	4096	1.0e-05	1.3e-09	0.26
Cube	*libfm*	4096	1.0e-07	4.5e-11	0.45
Cube	*libfm*	4096	1.0e-09	4.2e-12	0.42

continued on next page

Table A.4 – continued from previous page

Distribution	Method	N	control parameter	reached ΔE	$t\ [s]$
Cube	*libfm*	4096	1.0e-11	7.3e-15	0.58
Cube	*MMM1D*	4096	1.0e-01	1.5e-02	3.65
Cube	*MMM1D*	4096	1.0e-02	8.9e-04	4.34
Cube	*MMM1D*	4096	1.0e-03	4.5e-05	5.27
Cube	*MMM1D*	4096	1.0e-05	1.4e-07	6.86
Cube	*MMM1D*	4096	1.0e-07	3.8e-09	8.56
Cube	*libfm*	8192	1.0e-03	3.6e-08	0.21
Cube	*libfm*	8192	1.0e-05	2.4e-10	0.31
Cube	*libfm*	8192	1.0e-07	2.5e-11	0.48
Cube	*libfm*	8192	1.0e-09	1.1e-12	0.79
Cube	*libfm*	8192	1.0e-11	5.8e-14	1.29
Cube	*MMM1D*	8192	1.0e-01	2.3e-04	14.68
Cube	*MMM1D*	8192	1.0e-02	7.3e-05	17.35
Cube	*MMM1D*	8192	1.0e-03	6.4e-06	21.09
Cube	*MMM1D*	8192	1.0e-05	2.0e-07	27.41
Cube	*MMM1D*	8192	1.0e-07	9.4e-10	34.22
Cube	*libfm*	16384	1.0e-03	5.9e-08	0.65
Cube	*libfm*	16384	1.0e-05	2.7e-09	1.21
Cube	*libfm*	16384	1.0e-07	6.9e-12	0.89
Cube	*libfm*	16384	1.0e-09	3.5e-13	1.40
Cube	*MMM1D*	16384	1.0e-01	2.5e-03	58.23
Cube	*MMM1D*	16384	1.0e-02	1.6e-04	69.58
Cube	*MMM1D*	16384	1.0e-03	3.2e-05	84.61
Cube	*MMM1D*	16384	1.0e-05	2.1e-07	109.62
Cube	*MMM1D*	16384	1.0e-07	4.3e-09	137.08
Cube	*libfm*	32768	1.0e-03	2.1e-08	0.83
Cube	*libfm*	32768	1.0e-05	4.4e-10	1.37
Cube	*libfm*	32768	1.0e-07	3.2e-11	2.38
Cube	*libfm*	32768	1.0e-09	2.6e-13	2.51
Cube	*MMM1D*	32768	1.0e-01	1.1e-03	233.49
Cube	*MMM1D*	32768	1.0e-02	1.6e-04	277.86
Cube	*MMM1D*	32768	1.0e-03	8.0e-06	338.72
Cube	*MMM1D*	32768	1.0e-05	7.2e-08	440.09
Cube	*MMM1D*	32768	1.0e-07	1.5e-09	549.79
Cube	*libfm*	40000	1.0e-03	6.5e-08	0.96
Cube	*libfm*	40000	1.0e-05	2.7e-10	1.62
Cube	*libfm*	40000	1.0e-07	2.3e-11	2.72
Cube	*libfm*	40000	1.0e-09	4.1e-13	3.34
Cube	*MMM1D*	40000	1.0e-01	9.3e-04	348.83
Cube	*MMM1D*	40000	1.0e-02	3.1e-05	416.85
Cube	*MMM1D*	40000	1.0e-03	3.9e-06	506.95
Cube	*MMM1D*	40000	1.0e-07	1.4e-09	818.86

Bibliography

[1] L. Greengard. Fast Algorithms for Classical Physics. *Science*, 265(5174):909–914, August 1994.

[2] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *The Journal of Computational Physics*, 73(2):325–348, December 1987.

[3] P. P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik*, 369(3):253–287, 1921.

[4] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(6096):446–949, December 1986.

[5] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics*, 44(2):141–183, 1991.

[6] JUMP - Jülich Multi Processor. `http://www.fz-juelich.de/jsc/jump/`, 2010. [Online; accessed 19-February-2010].

[7] C. A. White and M. Head-Gordon. Derivation and efficient implementation of the fast multipole method. *The Journal of Chemical Physics*, 101(8):6593–6605, October 1994.

[8] S. A. Ghasemi, A. Neelov, and S. Goedecker. A particle-particle, particle-density algorithm for the calculation of electrostatic interactions of particles with slablike geometry. *The Journal of Chemical Physics*, 127(22):224102, 2007.

[9] A. Neelov, S. A. Ghasemi, and S. Goedecker. Particle-particle, particle-scaling function algorithm for electrostatic problems in free boundary conditions. *The Journal of Chemical Physics*, 127(2):024109, 2007.

[10] Fast multipole method implementation of JSC. `http://www.fz-juelich.de/jsc/fmm/`, 2010. [Online; accessed 19-February-2010].

[11] Holger Dachsel. An error-controlled fast multipole method. *The Journal of Chemical Physics*, 131(24):244102, 2009.

[12] Computational simulation of multi-body interactions with O(N) scaling. `http://www.multipole.org/`, 2010. [Online; accessed 19-February-2010].

[13] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 6 1998.

[14] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole method in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.

[15] Kernel-independent parallel 3d fast multipole method. `http://www.mrl.nyu.edu/~harper/kifmm3d/documentation/index.html`, 2010. [Online; accessed 19-February-2010].

[16] A. Arnold, B. A. Mann, and C. Holm. Simulating charged systems with ESPResSo. In M. Ferrario, G. Ciccotti, and K. Binder, editors, *Computer Simulations in Condensed Matter: from Materials to Chemical Biology*, volume 703 of *Lecture Notes in Physics*, pages 193–222. Springer, 2006.

[17] ESPREesSo - extensible simulation package for research on soft matter. `http://espressowiki.mpip-mainz.mpg.de/`, 2010. [Online; accessed 19-February-2010].

[18] H.-J. Limbach, A. Arnold, B. A. Mann, and C. Holm. ESPResSo – an extensible simulation package for research on soft matter systems. *Comput. Phys. Commun.*, 174(9):704–727, 2006.

[19] M. Deserno and C. Holm. How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines. *The Journal of Chemical Physics*, 109(18):7678–7693, 1998.

[20] A. Arnold and C. Holm. MMM1D: A method for calculating electrostatic interactions in one-dimensional periodic geometries. *The Journal of Chemical Physics*, 123(14):144103, 2005.

[21] A. Arnold and C. Holm. MMM2D: A fast and accurate summation method for electrostatic interactions in 2d slab geometries. *Computer Physics Communications*, 148(3):327–348, 2002.

[22] A. Arnold, J. de Joannis, and C. Holm. Electrostatics in periodic slab geometries. I. *The Journal of Chemical Physics*, 117(6):2496–2502, 2002.

[23] JARA-SIM website. `http://www.jara-excellence.de/cms/front_content.php?idcat=4&changelang=2`, 2010. [Online; accessed 19-February-2010].