Michael Pippig

Massively Parallel, Fast Fourier Transforms and Particle-Mesh Methods

Michael Pippig

Massively Parallel, Fast Fourier Transforms and Particle-Mesh Methods

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Universitätsverlag Chemnitz

2016

Impressum

Bibliografische Information der Deutschen Nationalbibliothek

Die deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über `http://dnb.d-nb.de` abrufbar.

Diese Arbeit wurde von der Fakultät für Mathematik der Technischen Universität Chemnitz als Dissertation zur Erlangung des akademischen Grades
Dr. rer. nat. genehmigt.
Die Arbeit wurde in englischer Sprache verfasst.

Tag der Einreichung: 25. Juni 2015

Betreuer: Prof. Dr. Daniel Potts, Technische Universität Chemnitz

1. Gutachter: Prof. Dr. Daniel Potts, Technische Universität Chemnitz
2. Gutachter: Prof. Dr. Christian Holm, Universität Stuttgart
3. Gutachter: Prof. Dr. Matthias Bolten, Universität Kassel

Tag der öffentlichen Prüfung: 13. Oktober 2015

Technische Universität Chemnitz/Universitätsbibiliothek
Universitätsverlag Chemnitz
09107 Chemnitz
`http://www.tu-chemnitz.de/ub/univerlag`

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
`http://www.mv-verlag.de`

ISBN 978–3–944640–76–1

`http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-197359`

Contents

1 Introduction

The development of efficient numerical algorithms can, without exaggeration, be called the fundamental basis of high performance computing. For sure, the permanent increase of computing power has changed our life in many ways. Problems that have been considered practically not computable for many years are now solvable within seconds. However, this enormous speedup would not have been possible without the development of adapted algorithms that are able to exploit the available computing power. Thereby, the continual change of computing architectures poses great challenges to the design of sustainable high performance software and requires repeated algorithmic rethinking.

The current trend of hardware design is mainly driven by massive parallelism. This includes shared memory parallelism, distributed memory parallelism, parallelism due to accelerators such as graphics processing units, and also any combination of these three. Still, programmers also have to consider the more or less traditional performance relevant problems such as optimal cache utilization and instruction set extensions like SSE and AVX - just to name a few. Programming on these environments gets increasingly sophisticated, which emphasizes the need for optimized software packages that encapsulate the solution of sub-problems within standalone modules. Note that typical applications in scientific computing are conglomerates of many different algorithmic modules. In most cases it is practically impossible to know all the performance relevant details of each algorithmic step. Instead, we may rely on low-level software modules that are highly optimized for these specific tasks. The philosophy behind this thesis is to reuse such low-level modules as much as possible and combine them to higher-level modules. The benefits of this so-called modularized approach are obvious. First of all, the modularized view on algorithms reveals a lot of common structure that simplifies programming and prevents code repetition. Second, improvements of low-level modules automatically inherit to higher-level modules, i.e., performance optimization can be focused on the modular level.

One can think of several attributes that such a software module should have as minimal requirements. Obviously, this list will strongly depend on the precise application of the software but we give a non-exhaustive list of important features that stand in the focus the present thesis:

- *Low complexity:* A focus on large problem sizes emphasizes the impact of fast algorithms. Within this thesis fast means that an algorithm scales almost linear in the number of degrees of freedom N, i.e., the number of arithmetic

operations increases at most proportional to $N \log N$.

- *Hardware adaptivity:* A software module should be portable to a broad variety of hardware architectures and result in high performance on all of these architectures.
- *Parallel scalability:* In order to exploit the computing power of massively parallel architectures, all software modules should be scalable in terms of runtime and memory consumption. This implies that the underlying algorithms must be adapted in order to circumvent scalability bottlenecks. Within this thesis we aim for distributed memory parallelism for hundreds of thousands of processes.
- *Error control and parameter selection:* Approximate algorithms often introduce a set of parameters that can be used to balance the required accuracy against the total computation time. In this case precise error estimates are desirable in order to assist selection of optimal parameters.
- *Flexibility:* The design of our algorithms should be abstract enough in order to suit a broad variety of applications. Optimizations that only effect rare use cases should not end up in a completely new software module.
- *Usability:* The sophisticated optimizations and algorithmic details should be hidden from the user by an easy to use interface.
- *Public availability:* All modules should be available as part of open source software libraries that are freely available. This makes it possible for other researchers to compare the underlying algorithms and contribute to the development process.

To find the modular structure of algorithms can not always be assumed to be trivial. Indeed, the use of different viewpoints and notation often leads to independent reinvention of algorithms in scientific literature. Also approaches that look remarkably different at the first sight may turn out to be closely related from an appropriate viewpoint. This is especially true for the broad variety of publications about the Fast Fourier Transform (FFT). In [127] the rigorous use of matrix-vector notation led to an unified framework that reveals the common structure and hidden beauty of supposedly distinct FFT approaches. The structured view not only helped to understand and classify existing FFT algorithms but also resulted in the discovery of new approaches. On the one hand, the vast variety of different FFT approaches makes it hard to choose the best suitable algorithm for a given problem; on the other hand, plurality gives the chance of algorithmic adaption with respect to changing hardware. A very elegant exploitation of this fact defines the design concept of the FFTW [57, 58] software library. Under the hood FFTW takes into account many different highly optimized FFT approaches and chooses the fastest one at runtime. However, the sophisticated implementation is hidden from the user by an easy to use interface. In this sense, FFTW fulfills all of the above mentioned criteria of a software module but one, namely, high scalability on distributed memory architectures. This led to the development of PFFT – an extension of FFTW

to massively parallel architectures that will be the topic of Chapter 2 in this thesis.

Another example that stands in the focus of this thesis is given by the diverse literature about particle-mesh methods. These are fast numerical methods for computing the Coulomb interactions between N charged particles in three-dimensional space. Loosely speaking, the basic concept of particle-mesh methods is to convert the given data in a suitable way such that fast Fourier transforms can be applied. Thereby, it is necessary to pre-process and post-processes the data in order to connect the irregularly distributed particle positions with the equispaced mesh points of the FFT. Within Chapter 4 of this thesis we show that so-called nonequispaced Fourier transforms (NFFTs) represent a powerful tool to reveal the common modularized structure of particle-mesh methods. NFFTs are generalizations of the FFT to nonequispaced sampling points and have been studied and optimized for a long time. Based on the NFFT framework we are able to summarize all particle-mesh methods within one unified framework called Particle-Particle–NFFT (P^2NFFT). However, before this unified picture can be drawn, it is necessary to extend the NFFT framework by several algorithmic modules that are motivated by particle-mesh methods but also recur in other applications of NFFT. This will be the main topic of Chapter 3 of this thesis. We will see that the modularized view of particle-mesh algorithms in terms of NFFTs leads to novel algorithmic developments of both – NFFT and particle-mesh methods.

To this end, the present thesis is about three algorithmic frameworks that are built on top of each other, namely FFT→NFFT→P^2NFFT. Each of these frameworks is designed with the above listed software module requirements in mind. Special emphasis will be placed on the parallel scalability on distributed memory architectures up to hundreds of thousands of processes. More precisely, this thesis is structured in the following three parts that are devoted to the three self-contained frameworks.

Outline of the Thesis

Chapter 2: Parallel fast Fourier transforms

At first, we derive the PFFT Framework2.5 – a highly scalable FFT framework designed for massive parallelism on distributed memory architectures. We start with some elementary definitions in Section 2.1. In Sections 2.2 and 2.3 we introduce the concept of pruned FFT with possibly shifted index sets in one and multiple dimensions. These FFT adaptions appear naturally during the approximation of non-periodic functions by finite Fourier series and will be an important ingredient of the algorithmic frameworks presented in the following chapters. We derive various representations of pruned FFT and FFT with shifted index sets in order to have the freedom to choose the most scalable approach for parallelization later on. As an interim result we get Corollary 2.3 that shows how one-dimensional pruned

FFT with shifted index sets can be computed without explicit data shifts. The multidimensional generalization of this fact is given in (2.9).

Next, in Section 2.4 we show how the algorithms that are already implemented within the FFTW software library can be turned into two powerful algorithmic modules for operating on parallel block-decomposed data. More precisely, these are the serial FFT and transposition module presented in Section 2.4.3 and the parallel multidimensional transposition module introduced in Section 2.4.7. Special attention is given to the generality of these modules. Especially, the pitfalls of unequal block decomposition in multiple dimensions are investigated in detail in Section 2.4.5.

We will see that the above mentioned two modules are sufficient to derive the main result of this chapter – the versatile PFFT Framework 2.5 for computing massively parallel FFT in a highly scalable way. Furthermore, we present the PFFT$^{\mathsf{H}}$ Framework 2.6 that essentially computes the parallel inverse FFT.

In Sections 2.6 and 2.7 we compare the different approaches for computing pruned FFT and FFT with shifted index sets with respect to their parallel scalability. It turns out that the most scalable approach is to apply pruning on a per-dimension basis, while shifted index sets should be incorporated by a suitable rescaling in Fourier space. Of course, our PFFT Framework 2.5 is based on these most scalable approaches. Furthermore, we give a brief overview of two ghost cell communication modules in Section 2.8. These modules accomplish the frequently occurring task of next neighbor communication in distributed memory parallelism.

As a major result of this chapter, a freely available implementation of PFFT is presented in Section 2.9. This exceptionally flexible software module is not only basis for all of the following frameworks in Chapters 3 and 4 but also deserves appreciation as a standalone library. Finally, we provide numerical evidence for the high scalability of our parallel FFT implementation with an extensive list of numerical tests in Section 2.10. These tests include investigation of strong and weak scaling on multiple hardware architectures as well as performance evaluations of parallel pruned FFT.

Parts of this chapter have already been published in condensed form in the peer-reviewed papers [4, 2]. A self-contained software library based on the PFFT framework is freely available at [11]. Most of the performance measurements have been published as parts of the proceedings [9, 7, 8].

Chapter 3: Parallel nonequispaced fast Fourier transforms

The second framework presented in this work is PNFFT – a highly scalable NFFT framework designed for massive parallelism on distributed memory architectures. It makes use of the previously derived PFFT framework from Chapter 2 but also incorporates many other algorithmic modules that are presented in this chapter.

Again, we start with some elementary definitions in Section 3.1 and review the

NFFT approximation ideas at the beginning of Section 3.2. Hereby, we introduce the new concept of pruned NFFT that can be understood as the nonequispaced analog of pruned FFT and becomes important for the evaluation of non-periodic Fourier approximations. Furthermore, we present two fast approaches for the computation of the NFFT's gradient that originate from particle-mesh methods but have never been considered as individual modules of the NFFT framework before. In Section 3.2.2 we introduce the new concept of shifted NFFT, i.e., an approximation similar to NFFT but with a mesh shifted by half the inverse mesh size. This enables us to transfer the concepts of interlacing from particle-mesh methods to the NFFT framework in Section 3.2.3. Interlacing can be understood as the average of standard NFFT outputs and shifted NFFT outputs. As far as we know, this is the first time that interlacing is considered in the context of the modularized NFFT framework. Another theoretical result on NFFT is presented in Section 3.2.4. There, we derive the formulas of optimal deconvolution coefficients of all presented NFFT types with respect to a mean square aliasing error. Also the interlaced NFFT is considered and the accuracy improvement of interlacing can be shown analytically. We emphasize that these optimized coefficients yield the missing link between NFFT and P^3M – a special kind of particle-mesh algorithm; cf. Chapter 4 for a detailed comparison of both methods.

In Section 3.3 we derive the PNFFT Framework 3.1 and the PNFFT$^{\mathsf{H}}$ Framework 3.2 for computing the NFFT and its adjoint in parallel. Thereby, we pay special attention to parallel scalability and make use of the parallel FFT frameworks that have been derived in the previous chapter. The parallel modules for the direct computation of nonequispaced discrete Fourier transforms in Section 3.4 further extend the flexibility of the PNFFT framework and provide the link between PNFFT and Ewald summation later on in Section 4.7.2. A freely available implementation of PNFFT is presented in Section 2.9. Finally, the numerical tests in Section 3.6 proof the high scalability of our parallel NFFT frameworks.

Parts of this chapter are based on the peer-reviewed paper [5]. The PNFFT framework has been published as a self-contained software library called PNFFT, which is freely available at [12]. We emphasize that the PNFFT library is the first open source, massively parallel software library for computing the NFFT.

Chapter 4: Parallel particle-mesh methods based on nonequispaced fast Fourier transforms

The last chapter is devoted to the Particle-Particle–NFFT (P^2NFFT) – an NFFT-based, fast framework for the computation of Coulomb interactions.

After some introductory part we define the Coulomb problem with mixed-periodic boundary conditions in Section 4.1 and review the Ewald splitting – a well know approach for splitting Coulomb interactions into short-range and smooth long-range contributions – in Section 4.2. Afterward, the derivation of an algorithmic module

for the computation of the short-range part is straightforward; see Section 4.3. In Section 4.4 we take a closer look at the smooth long-range part. Thereby, we present a new NFFT-based long-range interaction Module 4.3 for the fast computation of the long-range part in Section 4.4.1. In the following Sections 4.4.2–4.4.5, we show that this module can be applied for all types of periodicity. A crucial point is the correct treatment of non-periodic boundary conditions in order to get highly accurate Fourier approximations at modest mesh sizes. Our approach involves the embedding of non-periodic functions into smooth periodic functions that can be approximated well by finite Fourier series. Thereby, smoothness in the endpoints is ensured by continuation with a special kind of Hermite-Birkhoff interpolation. In Section 4.4.6 we present a new representation of such an interpolating polynomial in terms of Newton basis polynomials that allows an efficient evaluation. A detailed comparison of our continuation approach to other proposed Fourier approximations of non-periodic functions from the literature is given in Section 4.4.7.

Altogether, the modules for computing the short-range and long-range parts are combined in Section 4.5 and form the very flexible P^{2}NFFT Framework 4.4. The close connection of 3d-periodic P^{2}NFFT and the Particle-Particle–Particle-Mesh (P^{3}M) method is subject of investigation in Section 4.6. We show that P^{3}M is essentially a special case of P^{2}NFFT and that NFFTs provide a modularized view on P^{3}M. A unique characteristic of P^{3}M is its optimal deconvolution in Fourier space with respect to the mean square aliasing error. Almost the same deconvolution can be derived by P^{2}NFFT with the optimal NFFT deconvolution that was derived in Section 3.2.4. Furthermore, we show in Section 4.6.1 that application of interlaced NFFT to P^{2}NFFT results in four kinds of interlaced particle-mesh algorithms – only two of them have been considered in the literature for interlaced P^{3}M. However, we will also see that we can save one FFT in interlaced P^{3}M if we apply one of the two new interlacing approaches.

Beside P^{3}M, many other particle-mesh methods are included in the P^{2}NFFT framework. We review some of them in Section 4.7. These include NFFT-based fast summation, Ewald summation, NFFT-based fast Ewald summation, the Particle-Mesh Ewald method, the Smooth Particle-Mesh Ewald method, Gaussian Split Ewald and Spectrally Accurate Ewald.

Section 4.8 is devoted to a rigorous investigation of the asymptotic runtime of Ewald summation and P^{2}NFFT. Although it is very popular that Ewald summation scales as $\mathcal{O}(N^{3/2})$ for optimal choice of parameters, we show that this is only true for the computation of the Coulomb fields. Indeed, the computation of the Coulomb potential scales slightly worse as $\mathcal{O}((N \log N)^{3/2})$. On the other hand, optimal choice of parameters can turn the P^{2}NFFT into a method that scales as $\mathcal{O}(N\sqrt{\log N})$ for the field computation and as $\mathcal{O}(N\sqrt{\log N}(\log\log N)^{3/2})$ for the potential computation. Note that this is slightly better than the commonly cited $\mathcal{O}(N \log N)$ scaling of particle-mesh methods.

In Section 4.8.6 we present a heuristic approach for P^{2}NFFT parameter choice

that enables us to tune the 7 available P^{2}NFFT parameters for arbitrary prescribed root mean square error bounds. Thereby, 2 parameters can be eliminated by error bounds of the real space and Fourier space parts, one parameter can be assumed fixed in order to reach machine precision, 3 parameters have to be tuned for small test systems and can be kept constant with increasing N, and the last free parameter can be tuned in order to balance the real and Fourier space computation times.

The parallel P^{2}NFFT Framework 4.8 introduced in Section 4.9 is a generalization of P^{2}NFFT to massive parallelism on distributed memory architectures. The main ingredients are the parallel NFFT frameworks that have been derived in Chapter 3. A parallel implementation of P^{2}NFFT is part of the ScaFaCoS software library and will be presented in Section 4.10.

Finally, we provide numerical evidence for the validity of the P^{2}NFFT parameter tuning up to high precision in Section 4.11. Furthermore, we compare P^{2}NFFT to other fast Coulomb solvers with respect to runtime and parallel scalability.

Parts of this chapter are based on the peer-reviewed papers [5, 1, 3] and the proceedings [6]. A freely available implementation of the P^{2}NFFT framework has been published as part of the ScaFaCoS software library [10].

Acknowledgments

During the preparation of this thesis I enjoyed the help of many people. It is not an exaggeration to say, that this work would not have been possible without their support and I express my sincere gratitude to all of them. First of all, I like to thank my wife Jule for the continuous assistance and encouragement. I also thank my son Gustav and the rest of my family for their patience and tolerance regarding my limited free time due to long nights of programming.

I want to proclaim my deep appreciation to Prof. Dr. Daniel Potts – founder of the topic and outstanding adviser for many years. His expertise not only enriched my academic work but also my personality. Furthermore, I gratefully acknowledge the help of Prof. Dr. Christian Holm and Prof. Dr. Matthias Bolten, who agreed to examine this thesis.

A special thanks goes to my colleague Franziska Nestler. Her rigorous mathematical understanding really pushed forward the generalization of our particle-mesh methods to mixed-periodic boundary conditions. Beside Franziska I also like to thank all the other members of the working group on Applied Functional Analysis at Technische Universität Chemnitz for the nice atmosphere and fruitful discussions during the coffee breaks.

Many thanks go to all the contributors of the ScaFaCoS project. Especially, I want to acknowledge the great help of the following persons: Dr. Godehard Sutmann always managed to support me with computing time. I really adore him for his broad understanding of particle methods. From Dr. Axel Arnold and Dr. Olaf Lenz I gained a lot of experience on particle-mesh methods and code optimization. The competition with the FMM library authored by Dr. Holger Dachsel and Dr. Ivo Kabadshow initiated many performance improvements in my codes. Dr. Michael Hofmann did an excellent job in the support of massively parallel sorting, short-range interactions and build systems. I seriously admire his fast and helpful responses to my putative bug reports. Rene Halver put a lot of effort into the user friendliness of the ScaFaCoS library and Dr. Franz Gähler enforced invaluable improvements of our software packages due to his endless tests and applications to real world problems. Again, many thanks go to all of them.

This work was partly supported by the German Ministery of Science and Education (BMBF) under grant 01IH08001. Last but not least, I am grateful to the Jülich Supercomputing Center for providing the computational resources on Jülich Blue Gene/P (JUGENE), Jülich Blue Gene/Q (JUQUEEN) and Jülich Research on Petaflop Architectures (JUROPA).

2 Parallel fast Fourier transforms

Without doubt, the fast Fourier transform (FFT) is one of the most important algorithms in scientific computing. It provides the basis of many fast algorithms and there is no chance to give an exhaustive list of the tremendous number of applications. The crucial point in these applications is that the FFT reduces the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$, whereby N denotes the total number of degrees of freedom. For large N this comes with an enormous decrease in runtime. The original FFT algorithm was published in 1965 [32], although it was already known to Gauss [69]. Since then, an immense number of alternative algorithms and generalizations has been published, including FFT algorithms for arbitrary input size N, in-place algorithms, shared and distributed memory parallelism, FFTs on graphics processing units, optimized cache use, and many more hardware specific optimizations. A nice survey and abstract view is provided in [127]. But even this book does not include all the different approaches that have been used to find FFT algorithms. This variety of algorithms and the continuous change of hardware architectures made it practically impossible to find one FFT algorithm that is best suitable for all circumstances. This shortcoming led to the development of the FFTW [57] software library. Under the hood, FFTW compares a wide variety of different FFT algorithms and measures their runtime to find the most appropriate one for a given problem size and hardware architecture. The sophisticated implementation is hidden behind an easy interface structure. Therefore, users of FFTW are able to apply highly optimized FFT algorithms without knowing all the details about them. These algorithms have been continuously improved by many collaborators in order to support new hardware trends, such as SSE, AVX, graphics processing units, shared memory parallelism, distributed memory parallelism and so on.

Since the focus of this chapter is distributed memory parallelism for multidimensional FFTs we take a closer look at the available algorithms. There are two main approaches; the first is binary exchange algorithms, and the second is transpose algorithms. An introduction and theoretical comparison can be found in [66]. However, the second approach has been used in most implementations since it allows the use of highly optimized serial FFT kernels and offers a lot of flexibility. For a closer look at this approach we assume a three-dimensional input array of size $\hat{m}_0 \times \hat{m}_1 \times \hat{m}_2$ with $\hat{m}_0 \geq \hat{m}_1 \geq \hat{m}_2$. For the sake of simplicity, we assume that the array size $\hat{m}_0$ along the first dimension is divisible by the total number of parallel processes P. Then, the left hand side of Figure 2.1 gives an overview of the so-called slab decomposition

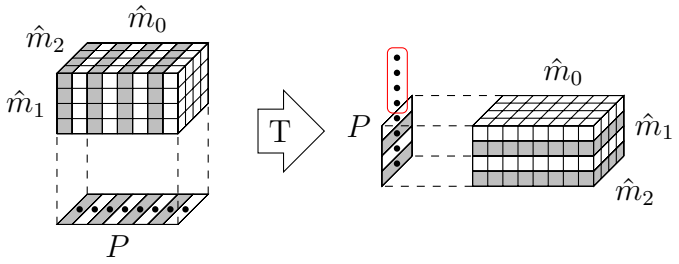

Figure 2.1: Decomposition of a three-dimensional array of size $\hat{m}_0 \times \hat{m}_1 \times \hat{m}_2 = 8 \times 4 \times 4$ on a one-dimensional process mesh of size $P = 8$. After the transposition (T) half of the processes remain idle.

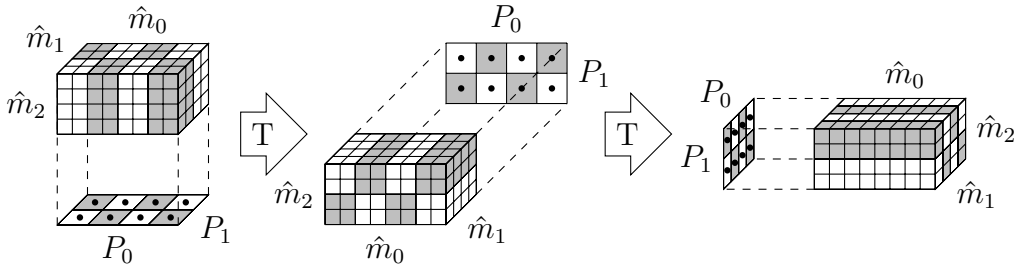

Figure 2.2: Distribution of a three-dimensional array of size $\hat{m}_0 \times \hat{m}_1 \times \hat{m}_2 = 8 \times 4 \times 4$ on a two-dimensional process mesh of size $P_0 \times P_1 = 4 \times 2$. None of the processes remains idle in any calculation step.

or one-dimensional decomposition. It is well known that a multidimensional FFT can be efficiently computed by a sequence of lower-dimensional FFTs. Therefore, a parallel FFT algorithm based on slab decomposition is given as follows. First, compute the $\hat{m}_0/P$ locally available, two-dimensional FFTs of size $\hat{m}_1 \times \hat{m}_2$ on each process. Second, perform a transposition such that only the second dimension of the data array is decomposed. Finally, compute the remaining $\hat{m}_1/P \times \hat{m}_2$ locally available, one-dimensional FFTs of size $\hat{m}_0$ on each process. This approach has been implemented in many software libraries including the IBM PESSL library [54], the Intel Math Kernel Library [75], and the FFTW [57] software package. The main drawback of one-dimensional decomposition is given by the fact that we can not use more than $\hat{m}_1$ parallel processes without sacrificing performance due to idle processes. This shortcoming is illustrated in Figure 2.1.

The main idea in overcoming this scalability bottleneck is to use a two-dimensional data decomposition. Assume a two-dimensional mesh of $P_0 \times P_1$ processes. Then, Figure 2.2 illustrates the so-called rod or pencil decomposition. This time, every process starts with the computation of $\hat{m}_0/P_0 \times \hat{m}_1/P_1$ one-dimensional FFTs of size $\hat{m}_2$, followed by a communication step that ensures a new two-dimensional data

decomposition along $\hat{m}_0$ and $\hat{m}_2$. After another $\hat{m}_0/P_0 \times \hat{m}_2/P_1$ one-dimensional FFTs of size $\hat{m}_1$ and one more parallel transposition, we end up with another $\hat{m}_1/P_0 \times \hat{m}_2/P_1$ one-dimensional FFTs of size $\hat{m}_0$. Note that the number of data transpositions is increased by one in comparison to the one-dimensional decomposition approach. However, these data transpositions are performed in smaller subgroups along the rows and columns of the process mesh, which results in a better latency bound.

The two-dimensional data decomposition allows us to increase the number of processes to at most $\hat{m}_1 \cdot \hat{m}_2$. It was first proposed in 1995 [40]. Eleftheriou et al. [47] implemented a software library for power-of-two FFTs customized to the Blue Gene/L architecture based on the two-dimensional data decomposition. They used to call it volumetric domain decomposition – which is a bit misleading. Indeed the a three-dimensional data decomposition is remapped to a two-dimensional one, before serial FFTs are computed. The first publicly available implementation of the two-dimensional decomposition approach came with Sandia-FFT [110, 109]. Afterward, several other packages appeared such as P3DFFT [107, 106], FFTE [124, 123] and 2DECOMP&FFT [91, 90]. Performance evaluations of two-dimensional decomposed parallel FFTs have been published in [51, 124, 104]. However, finding other approaches for highly scalable, parallel FFT is still a vital research topic [34, 22, 119, 60]. Although the two-dimensional decomposition allows to employ more processes for parallel computation, it also increases the amount of inherent communication due to multiple parallel data transpositions. In this sense, high scalability and low communication overhead are somehow contradictory. It was argued that the one-dimensional decomposition should be used as long as the total number of processes is less or equal to the FFT mesh size in each dimension [77] and switch to two-dimensional decomposition otherwise. An elegant approach came up with OpenFFT [44, 43]. Here, the transition from one- to two-dimensional decomposition is performed step by step resulting in a less regular domain decomposition that optimizes the locality of data blocks. Therefore, less data needs to be send during the parallel transposition. However, the less regular decomposition places an additional burden on the user and the implementation of the transposition is much more complicated. Especially, it is much harder to figure out neighboring processes with respect to this decomposition scheme.

All of these FFT implementations based on two-dimensional domain decomposition offer a different set of features, introduce their own interface and have several restrictions that the user must be aware of. Unfortunately, this inconvenience can not the avoided by using the highly appreciated FFTW library since it only supports one-dimensional domain decomposition. This problem naturally leads to the question if there is a way to extend FFTW to two-dimensional domain decompositions. Thereby, the important point is to implement all performance relevant steps of a two-dimensional distributed FFT with modules that can be completely implemented by FFTW calls. Following this route, one can combine the hardware adaptivity of FFTW with the improved scalability of the two-dimensional decomposition. This is

exactly what the PFFT [11] software library stands for and the precise algorithmic structure of the PFFT framework will be described in this chapter. Thereby, we pay special attention to the flexibility of our framework in the sense that it supports many features that are not available in other parallel FFT implementations. Theses include $(d-1)$-dimensional domain decomposition of d-dimensional FFTs, adapted parallel algorithm design for pruned FFTs and an efficient parallel implementation of FFTs with shifted index sets. Note that pruned FFTs with shifted index sets naturally appear in the approximation of non-periodic functions by finite Fourier series and will be an important ingredient of the algorithmic frameworks presented in Chapters 3 and 4.

2.1 Definitions

Throughout this thesis we use the common notations $\mathbb{N}, \mathbb{Z}, \mathbb{R}$, and $\mathbb{C}$, for the sets of all natural numbers, integers, real numbers and complex numbers, respectively. Let the Kronecker symbol δ_k be defined as $\delta_k := 0$ for $k \neq 0$ and $\delta_0 := 1$. For an arbitrary vector $\boldsymbol{d} = (d_0, d_1, \ldots, d_{n-1})^\mathsf{T} \in \mathbb{C}^n$ we denote the corresponding $n \times n$ diagonal matrix by $\operatorname{diag} \boldsymbol{d} = \operatorname{diag}(d_0, \ldots, d_{n-1}) := (d_i \delta_{i-j})_{i,j=0}^{n-1}$. Let $\mathbf{I}_n := \operatorname{diag}(1, 1, \ldots, 1)$ denote the $n \times n$ identity matrix, and $\mathbf{0}_{m,n}$ the $m \times n$ matrix with zero entries. For the square case $m = n$ we introduce the shorthand $\mathbf{0}_n$. The Kronecker product of two matrices $\mathbf{A} \in \mathbb{C}^{p \times q}$, $\mathbf{B} \in \mathbb{C}^{s \times t}$ is defined as

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} a_{0,0}\mathbf{B} & \cdots & a_{0,q-1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{p-1,0}\mathbf{B} & \cdots & a_{p-1,q-1}\mathbf{B} \end{pmatrix} \in \mathbb{C}^{ps \times qt}.$$

In the following, some basic properties of the Kronecker product are revisited. An extended list and proofs can be found in [120]. The Kronecker product is bi-linear, associative and the Kronecker product of two identity matrices yields an identity matrix $\mathbf{I}_m \otimes \mathbf{I}_n = \mathbf{I}_{m \cdot n}$. The transposition of a Kronecker product is the Kronecker product of the transpositions, i.e., $(\mathbf{A} \otimes \mathbf{B})^\mathsf{T} = \mathbf{B}^\mathsf{T} \otimes \mathbf{A}^\mathsf{T}$. Let $\mathbf{C} \in \mathbb{C}^{q \times r}$, and $\mathbf{D} \in \mathbb{C}^{t \times u}$, i.e., the products $\mathbf{AC}$ and $\mathbf{BD}$ are well defined. Then, the Kronecker product fulfills

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}).$$

Especially, we get

$$\mathbf{A} \otimes \mathbf{B} = (\mathbf{A} \otimes \mathbf{I}_s)\,(\mathbf{I}_q \otimes \mathbf{B}) = (\mathbf{I}_p \otimes \mathbf{B})\,(\mathbf{A} \otimes \mathbf{I}_t)\,. \tag{2.1}$$

This property allows us to separate the computation of algorithmic steps that operate on individual dimensions. Moreover, we see that we are free to choose the order of execution as long as we take care of the correct strides.

2.2 One-dimensional fast Fourier transforms

For any given mesh size $M \in \mathbb{N}$ we define the Fourier matrix

$$\mathbf{F}_M := \left(\mathrm{e}^{-2\pi \mathrm{i} kl/M}\right)_{k,l=0}^{M-1,M-1}.$$

The linear mapping $\mathrm{DFT} : \mathbb{C}^M \to \mathbb{C}^M$, that assigns a complex vector $\hat{\boldsymbol{g}} = (\hat{g}_k)_{k=0}^{M-1} \in \mathbb{C}^M$ to $\mathrm{DFT}(\hat{\boldsymbol{g}}) = \mathbf{F}_M \hat{\boldsymbol{g}} \in \mathbb{C}^M$ is known as discrete Fourier transform (DFT). Furthermore, the adjoint discrete Fourier transform ($\mathrm{DFT}^{\mathsf{H}}$) is given by the linear mapping $\mathrm{DFT}^{\mathsf{H}} : \mathbb{C}^M \to \mathbb{C}^M$ that assigns complex vectors $\boldsymbol{g} = (g_l)_{l=0}^{M-1} \in \mathbb{C}^M$ to $\mathrm{DFT}^{\mathsf{H}}(\boldsymbol{g}) = \mathbf{F}_M^{\mathsf{H}} \boldsymbol{g} \in \mathbb{C}^M$. Hereby, $\mathbf{F}_M^{\mathsf{H}}$ denotes the adjoint of $\mathbf{F}_M$. It is well known that the Fourier matrix fulfills $\mathbf{F}_M \mathbf{F}_M^{\mathsf{H}} = M \mathbf{I}_M$, i.e., $\mathbf{F}_M$ is unitary up to a normalization factor. A direct computation of a DFT requires $\mathcal{O}(M^2)$ arithmetic operations. In 1965 Cooley and Tukey [32] published a divide-and-conquer algorithm that reduces the arithmetic complexity to $\mathcal{O}(M \log M)$ by exploiting the structure of the Fourier matrix. This algorithm was restricted to highly composite mesh size M, i.e., the prime factors of M are sufficiently small. Later on, algorithms with the same complexity have been developed also for large prime sizes M, cf. [116, 27]. The whole class of fast algorithms that realize DFTs within $\mathcal{O}(M \log M)$ arithmetic operations are commonly known as fast Fourier transforms (FFTs). Similarly, we denote any fast algorithm for computing the $\mathrm{DFT}^{\mathsf{H}}$ as adjoint fast Fourier transform ($\mathrm{FFT}^{\mathsf{H}}$). In the following, we discuss the generalization of one-dimensional DFTs to pruned input and output as well as the definition of DFTs with shifted index sets. These transforms are building blocks of the algorithms given in Chapter 3.

2.2.1 Pruned fast Fourier transform

A DFT of size $\hat{m} \in \mathbb{N}$ can be interpreted as the evaluation of a one-dimensional trigonometric polynomial $g : \mathbb{R} \to \mathbb{C}$ given by

$$g(x) := \sum_{k=0}^{\hat{m}-1} \hat{g}_k \mathrm{e}^{-2\pi \mathrm{i} kx}$$

at equispaced nodes, i.e., $g\left({}^{l}/_{\hat{m}}\right)$, $l = 0, \ldots, \hat{m} - 1$. This corresponds to a distance of ${}^{1}/_{\hat{m}}$ between the equispaced nodes. In our context, a pruned DFT means that we want to sample the trigonometric polynomial g only at $m \in \mathbb{N}$ equispaced points with a possibly smaller sampling distance ${}^{1}/_{M} \leq {}^{1}/_{\hat{m}}$. More precisely, let $M, \hat{m}, m \in \mathbb{N}$ such that $\hat{m} \leq M$ and $m \leq M$. Then, the pruned DFT is given by the sums

$$g_l := g\left(\tfrac{l}{M}\right) = \sum_{k=0}^{\hat{m}-1} \hat{g}_k \mathrm{e}^{-2\pi \mathrm{i} kl/M} = \sum_{k=0}^{M-1} \hat{g}_k \mathrm{e}^{-2\pi \mathrm{i} kl/M}, \quad l = 0, \ldots, m-1. \tag{2.2}$$

Hereby, we extended the last sum by appending zero coefficients $\hat{g}_k := 0$, $k = \hat{m}, \dots, M-1$ at the end of the input vector $\hat{\boldsymbol{g}}$. In order to give a matrix representation of the pruned FFT we define the zero padding matrix $\mathbf{P}_{r,s} \in \mathbb{R}^{r\times s}$ for $r, s \in \mathbb{N}$ with $r \geq s$ by

$$\mathbf{P}_{r,s} := \begin{pmatrix} \mathbf{I}_s \\ \mathbf{0}_{r-s,s} \end{pmatrix}.$$

Note that the transpose $\mathbf{P}_{r,s}^{\mathsf{T}} \in \mathbb{R}^{s\times r}$ of the zero padding matrix accomplishes a truncation of the last $r-s$ rows. Now, the pruned FFT (2.2) reads in matrix-vector form

$$\boldsymbol{g} = \mathbf{P}_{M,m}^{\mathsf{T}} \mathbf{F}_M \mathbf{P}_{M,\hat{m}} \hat{\boldsymbol{g}},$$

with the vectors $\hat{\boldsymbol{g}} := (\hat{g}_k)_{k=0}^{\hat{m}-1} \in \mathbb{C}^{\hat{m}}$ and $\boldsymbol{g} = (g_l)_{l=0}^{m-1} \in \mathbb{C}^m$. From this representation it is clear that a pruned FFT can be computed by an FFT of size M that is preceded by a zero padding and followed by a truncation. The pruned $\mathrm{FFT}^{\mathsf{H}}$ is defined as the adjoint of the pruned FFT and, therefore, yields an adjoint matrix factorization.

2.2.2 Fast Fourier transform with shifted index sets

In many applications it is more convenient to define the DFT of even mesh size $M \in 2\mathbb{N}$ with index sets shifted by $M/2$, i.e., for $\hat{f}_k \in \mathbb{C}$, $k = -M/2, \dots, M/2 - 1$ compute the sums

$$f_l := \sum_{k=-M/2}^{M/2-1} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} kl/M} \in \mathbb{C}, \quad l = -\tfrac{M}{2}, \dots, \tfrac{M}{2} - 1. \tag{2.3}$$

In this case we define the input vector $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k=-M/2}^{M/2-1} \in \mathbb{C}^M$, the output vector $\boldsymbol{f} := (f_l)_{l=-M/2}^{M/2-1} \in \mathbb{C}^M$ and the shift matrix

$$\mathbf{S}_M := \begin{pmatrix} \mathbf{0}_{M/2} & \mathbf{I}_{M/2} \\ \mathbf{I}_{M/2} & \mathbf{0}_{M/2} \end{pmatrix}.$$

Then, we can rewrite (2.3) in matrix-vector form as

$$\boldsymbol{f} = \mathbf{S}_M \mathbf{F}_M \mathbf{S}_M \hat{\boldsymbol{f}}. \tag{2.4}$$

An application of the shift matrix $\mathbf{S}_M$ to a given vector is often called FFT shift. It moves the first half of the input vector to the end as following

$$\mathbf{S}_M \left(\hat{f}_{-M/2}, \dots, \hat{f}_{-1}, \hat{f}_0, \dots, \hat{f}_{M/2-1} \right)^{\mathsf{T}} = \left(\hat{f}_0, \dots, \hat{f}_{M/2-1}, \hat{f}_{-M/2}, \dots, \hat{f}_{-1} \right)^{\mathsf{T}}.$$

The matrix-vector representation (2.4) shows that computing the FFT with shifted index sets is essentially nothing else than computing an ordinary FFT preceded and followed by this kind of data shift.

2.2.3 Pruned fast Fourier transform with shifted index sets

The definition of the pruned FFT changes slightly for the case of shifted index sets. Let $M, \hat{m}, m \in 2\mathbb{N}$ with $\hat{m} \leq M$ and $m \leq M$. Then, the pruned DFT with shifted index sets is defined as

$$f_l = \sum_{k=-\hat{m}/2}^{\hat{m}/2-1} \hat{f}_k \mathrm{e}^{-2\pi \mathrm{i} kl/M}, \quad l = -\tfrac{m}{2}, \ldots, \tfrac{m}{2} - 1.$$

This can be written in matrix-vector form as

$$\boldsymbol{f} = \widetilde{\mathbf{P}}_{M,m}^{\mathsf{T}} \mathbf{S}_M \mathbf{F}_M \mathbf{S}_M \widetilde{\mathbf{P}}_{M,\hat{m}} \hat{\boldsymbol{f}}, \tag{2.5}$$

with the vectors $\boldsymbol{f} = (f_l)_{l=-m/2}^{m/2-1} \in \mathbb{C}^m$, $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k=-\hat{m}/2}^{\hat{m}/2-1} \in \mathbb{C}^{\hat{m}}$ and the shifted zero padding matrix $\widetilde{\mathbf{P}}_{r,s} \in \mathbb{R}^{r \times s}$ that is defined for $r, s \in 2\mathbb{N}$ with $r \geq s$ by

$$\widetilde{\mathbf{P}}_{r,s} := \begin{pmatrix} \mathbf{0}_{(r-s)/2,s} \\ \mathbf{I}_s \\ \mathbf{0}_{(r-s)/2,s} \end{pmatrix}.$$

Note that half of the zero padding and truncation is now applied to the first rows and only the other half is applied to the last rows. The pruned $\mathrm{DFT}^{\mathsf{H}}$ with shifted index sets is defined by the adjoint of (2.5).

In Sections 2.6 and 2.7 we will see that representation (2.5) is not well suited to design a scalable parallel algorithm. The main shortcoming are the explicit data movements due to FFT shifts that correspond to global communications on parallel environments. Therefore, we derive an alternative representation of the pruned FFT with shifted index sets that does not involve data reordering. Let the twiddle matrix $\mathbf{T}_M \in \mathbb{R}^{M \times M}$ of size M be given by

$$\mathbf{T}_M := \operatorname{diag}\left((-1)^k\right)_{k=0}^{M-1}.$$

Lemma 2.1. *For $M \in 2\mathbb{N}$ we have*

$$\begin{aligned} \mathbf{F}_M \mathbf{S}_M &= \mathbf{T}_M \mathbf{F}_M, \\ \mathbf{S}_M \mathbf{F}_M &= \mathbf{F}_M \mathbf{T}_M, \text{ and} \\ \mathbf{S}_M \mathbf{T}_M &= (-1)^{M/2} \mathbf{T}_M \mathbf{S}_M. \end{aligned}$$

Proof. Let $m := M/2 \in \mathbb{N}$ and block the Fourier matrix $\mathbf{F}_M = (\mathbf{F}_{M,m}^{\mathrm{L}} | \mathbf{F}_{M,m}^{\mathrm{R}})$ into the left hand side columns $\mathbf{F}_{M,m}^{\mathrm{L}} := (\mathrm{e}^{-2\pi \mathrm{i} kl/M})_{k=0,l=0}^{M-1,m-1}$ and the right hand side columns $\mathbf{F}_{M,m}^{\mathrm{R}} := (\mathrm{e}^{-2\pi \mathrm{i} kl/M})_{k=0,l=m}^{M-1,M-1}$. As a consequence of $\mathrm{e}^{-2\pi \mathrm{i} k(l+m)/M} = (-1)^k \mathrm{e}^{-2\pi \mathrm{i} kl/M}$ we

obtain $\mathbf{F}^{\mathrm{L}}_{M,m} = \mathbf{T}_M \mathbf{F}^{\mathrm{R}}_{M,m}$ and, equivalently, $\mathbf{F}^{\mathrm{R}}_{M,m} = \mathbf{T}_M \mathbf{F}^{\mathrm{L}}_{M,m}$. Now, the evidence of the first claim results from

$$\mathbf{F}_M \mathbf{S}_M = \left(\mathbf{F}^{\mathrm{R}}_{M,m} \middle| \mathbf{F}^{\mathrm{L}}_{M,m}\right) = \left(\mathbf{T}_M \mathbf{F}^{\mathrm{L}}_{M,m} \middle| \mathbf{T}_M \mathbf{F}^{\mathrm{R}}_{M,m}\right) = \mathbf{T}_M \mathbf{F}_M.$$

Because of the symmetries $\mathbf{F}_M = \mathbf{F}_M^{\mathsf{T}}$, $\mathbf{S}_M = \mathbf{S}_M^{\mathsf{T}}$ and $\mathbf{T}_M = \mathbf{T}_M^{\mathsf{T}}$ we obtain the second claim by $\mathbf{S}_M \mathbf{F}_M = (\mathbf{F}_M \mathbf{S}_M)^{\mathsf{T}} = (\mathbf{T}_M \mathbf{F}_M)^{\mathsf{T}} = \mathbf{F}_M \mathbf{T}_M$. Finally, we represent $\mathbf{T}_M$ by the following block structure

$$\mathbf{T}_M = \begin{pmatrix} \mathbf{T}_m & \mathbf{0}_m \\ \mathbf{0}_m & (-1)^m \mathbf{T}_m \end{pmatrix},$$

and get

$$\mathbf{S}_M \mathbf{T}_M = \begin{pmatrix} \mathbf{0}_m & (-1)^m \mathbf{T}_m \\ \mathbf{T}_m & \mathbf{0}_m \end{pmatrix} = (-1)^m \mathbf{T}_M \mathbf{S}_M.$$

■

Lemma 2.2. *Let $M, \hat{m} \in 2\mathbb{N}$ such that $\hat{m} \leq M$. Then,*

$$\mathbf{T}_M \widetilde{\mathbf{P}}_{M,\hat{m}} = (-1)^{(M+\hat{m})/2} \widetilde{\mathbf{P}}_{M,\hat{m}} \mathbf{T}_{\hat{m}}.$$

Proof. We have

$$\mathbf{T}_M \widetilde{\mathbf{P}}_{M,\hat{m}} = \begin{pmatrix} \mathbf{0}_{(M-\hat{m})/2,\hat{m}} \\ (-1)^{(M-\hat{m})/2} \mathbf{T}_{\hat{m}} \\ \mathbf{0}_{(M-\hat{m})/2,\hat{m}} \end{pmatrix} = (-1)^{(M-\hat{m})/2} \widetilde{\mathbf{P}}_{M,\hat{m}} \mathbf{T}_{\hat{m}}.$$

Since $\hat{m}$ is even, we can multiply this line with $1 = (-1)^{\hat{m}}$ and obtain the claim. ■

Finally, we yield an alternative representation of the pruned FFT with shifted index sets by straightforward combination of Lemma 2.1 and Lemma 2.2.

Corollary 2.3. *Let $M, \hat{m}, m \in 2\mathbb{N}$ such that $\hat{m} \leq M$ and $m \leq M$. Then, the matrix representation of the pruned DFT with shifted index sets can be rewritten as*

$$\widetilde{\mathbf{P}}^{\mathsf{T}}_{M,m} \mathbf{S}_M \mathbf{F}_M \mathbf{S}_M \widetilde{\mathbf{P}}_{M,\hat{m}} = (-1)^{(M+m+\hat{m})/2} \mathbf{T}_m \widetilde{\mathbf{P}}^{\mathsf{T}}_{M,m} \mathbf{F}_M \widetilde{\mathbf{P}}_{M,\hat{m}} \mathbf{T}_{\hat{m}}.$$

2.3 Multidimensional fast Fourier transform

The definitions from Section 2.2 can be generalized straightforward to $d \in \mathbb{N}$ dimensions by using Kronecker products. Let the d-dimensional mesh sizes $\boldsymbol{M} = (M_t)_{t=0}^{d-1}, \hat{\boldsymbol{m}} = (\hat{m}_t)_{t=0}^{d-1}, \boldsymbol{m} = (m_t)_{t=0}^{d-1} \in \mathbb{N}^d$ be given such that $\hat{\boldsymbol{m}} \leq \boldsymbol{M}$ and $\boldsymbol{m} \leq \boldsymbol{M}$

Algorithm 2.1 Multidimensional pruned FFT - Variant A

Input: $\hat{\boldsymbol{g}} \in \mathbb{C}^{\hat{m}_0 \cdots \hat{m}_{d-1}}$

1: Pad the d-dimensional input array with zeros: $\hat{\boldsymbol{g}} \leftarrow (\mathbf{P}_{M_0,\hat{m}_0} \otimes \cdots \otimes \mathbf{P}_{M_{d-1},\hat{m}_{d-1}})\hat{\boldsymbol{g}}$.
2: Compute the multidimensional FFT: $\boldsymbol{g} \leftarrow (\mathbf{F}_{M_0} \otimes \cdots \otimes \mathbf{F}_{M_{d-1}})\hat{\boldsymbol{g}}$.
3: Truncate the d-dimensional input array: $\boldsymbol{g} \leftarrow (\mathbf{P}^{\mathsf{T}}_{M_0,m_0} \otimes \cdots \otimes \mathbf{P}^{\mathsf{T}}_{M_{d-1},m_{d-1}})\boldsymbol{g}$.

Output: $\boldsymbol{g} = \mathbf{P}^{\mathsf{T}}_{\boldsymbol{M},\boldsymbol{m}}\mathbf{F}_{\boldsymbol{M}}\mathbf{P}_{\boldsymbol{M},\hat{\boldsymbol{m}}}\hat{\boldsymbol{g}} \in \mathbb{C}^{m_0 \cdots m_{d-1}}$

Algorithm 2.2 Multidimensional pruned FFT - Variant B

Input: $\hat{\boldsymbol{g}} \in \mathbb{C}^{\hat{m}_0 \cdots \hat{m}_{d-1}}$

1: $\boldsymbol{g} \leftarrow \hat{\boldsymbol{g}}$
2: **for** $t = d-1, \ldots, 0$ **do**
3: $\quad \boldsymbol{g} \leftarrow (\mathbf{I}_{m_0} \otimes \cdots \otimes \mathbf{I}_{m_{t-1}}) \otimes \mathbf{P}^{\mathsf{T}}_{M_t,m_t}\mathbf{F}_{M_t}\mathbf{P}_{M_t,\hat{m}_t} \otimes (\mathbf{I}_{\hat{m}_{t+1}} \otimes \cdots \otimes \mathbf{I}_{\hat{m}_{d-1}})\boldsymbol{g}$
4: **end for**

Output: $\boldsymbol{g} = \mathbf{P}^{\mathsf{T}}_{\boldsymbol{M},\boldsymbol{m}}\mathbf{F}_{\boldsymbol{M}}\mathbf{P}_{\boldsymbol{M},\hat{\boldsymbol{m}}}\hat{\boldsymbol{g}} \in \mathbb{C}^{m_0 \cdots m_{d-1}}$

holds component-wise. Then, the matrix representation of the d-dimensional pruned DFT is given by

$$\mathbf{P}^{\mathsf{T}}_{\boldsymbol{M},\boldsymbol{m}}\mathbf{F}_{\boldsymbol{M}}\mathbf{P}_{\boldsymbol{M},\hat{\boldsymbol{m}}} \in \mathbb{C}^{(m_0 \cdots m_{d-1}) \times (\hat{m}_0 \cdots \hat{m}_{d-1})}, \tag{2.6}$$

where all matrices are defined as their corresponding Kronecker products $\mathbf{P}^{\mathsf{T}}_{\boldsymbol{M},\boldsymbol{m}} := \bigotimes_{t=0}^{d-1} \mathbf{P}^{\mathsf{T}}_{M_t,m_t}$, $\mathbf{F}_{\boldsymbol{M}} := \bigotimes_{t=0}^{d-1} \mathbf{F}_{M_t}$, and $\mathbf{P}_{\boldsymbol{M},\hat{\boldsymbol{m}}} := \bigotimes_{t=0}^{d-1} \mathbf{P}_{M_t,\hat{m}_t}$. This representation directly leads to the simple three-step Algorithm 2.1 for computing the d-dimensional pruned FFT. However, the definitions

$$\mathbf{A}_t := (\mathbf{I}_{m_0} \otimes \cdots \otimes \mathbf{I}_{m_{t-1}}) \otimes \mathbf{P}^{\mathsf{T}}_{M_t,m_t}\mathbf{F}_{M_t}\mathbf{P}_{M_t,\hat{m}_t} \otimes (\mathbf{I}_{\hat{m}_{t+1}} \otimes \cdots \otimes \mathbf{I}_{\hat{m}_{d-1}}),$$

$t = 0, \ldots, d-1$ together with the Kronecker product property (2.1) directly lead to the alternative factorization

$$\mathbf{P}^{\mathsf{T}}_{\boldsymbol{M},\boldsymbol{m}}\mathbf{F}_{\boldsymbol{M}}\mathbf{P}_{\boldsymbol{M},\hat{\boldsymbol{m}}} = \mathbf{A}_0\mathbf{A}_1 \cdots \mathbf{A}_{d-1}. \tag{2.7}$$

This representation yields the theoretical foundation for separating multidimensional pruned FFTs into a sequence of one-dimensional pruned FFTs of multiple, non-unit stride input vectors as given in Algorithm 2.2. This formulation will be the starting point for the parallel pruned FFT algorithms presented in this work; cf. the comparison of Algorithms 2.1 and 2.2 for parallel scalability in Section 2.6.

Analogously, for $\boldsymbol{M}, \hat{\boldsymbol{m}}, \boldsymbol{m} \in 2\mathbb{N}^d$ with $\hat{\boldsymbol{m}} \leq \boldsymbol{M}$, $\boldsymbol{m} \leq \boldsymbol{M}$ the matrix representation of the d-dimensional pruned DFT with shifted index sets is given by

$\widetilde{\mathbf{P}}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}}\mathbf{S}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{S}_{\boldsymbol{M}}\widetilde{\mathbf{P}}_{\boldsymbol{M},\hat{\boldsymbol{m}}}$, with the tensor product matrices $\widetilde{\mathbf{P}}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}} := \bigotimes_{t=0}^{d-1}\widetilde{\mathbf{P}}_{M_t,m_t}^{\mathsf{T}}$, $\mathbf{S}_{\boldsymbol{M}} := \bigotimes_{t=0}^{d-1}\mathbf{S}_{M_t}$, and $\widetilde{\mathbf{P}}_{\boldsymbol{M},\hat{\boldsymbol{m}}} := \bigotimes_{t=0}^{d-1}\widetilde{\mathbf{P}}_{M_t,\hat{m}_t}$. Again, by Kronecker product property (2.1) we see that

$$\widetilde{\mathbf{P}}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}}\mathbf{S}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{S}_{\boldsymbol{M}}\widetilde{\mathbf{P}}_{\boldsymbol{M},\hat{\boldsymbol{m}}} = \mathbf{B}_0\mathbf{B}_1\cdots\mathbf{B}_{d-1}, \tag{2.8}$$

where for $t = 0,\ldots,d-1$ the matrix factors $\mathbf{B}_t$ are defined as

$$\mathbf{B}_t := (\mathbf{I}_{m_0}\otimes\cdots\otimes\mathbf{I}_{m_{t-1}})\otimes\widetilde{\mathbf{P}}_{M_t,m_t}^{\mathsf{T}}\mathbf{S}_{M_t}\mathbf{F}_{M_t}\mathbf{S}_{M_t}\widetilde{\mathbf{P}}_{M_t,\hat{m}_t}\otimes(\mathbf{I}_{\hat{m}_{t+1}}\otimes\cdots\otimes\mathbf{I}_{\hat{m}_{d-1}}).$$

Now, application of Corollary 2.3 to every matrix $\mathbf{B}_t$ in (2.8) and reorder of Kronecker products with the help of (2.1) gives the following d-dimensional analogue of Corollary 2.3

$$\widetilde{\mathbf{P}}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}}\mathbf{S}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{S}_{\boldsymbol{M}}\widetilde{\mathbf{P}}_{\boldsymbol{M},\hat{\boldsymbol{m}}} = \prod_{t=0}^{d-1}(-1)^{(M_t+m_t+\hat{m}_t)/2}\mathbf{T}_{\boldsymbol{m}}\widetilde{\mathbf{P}}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}}\mathbf{F}_{\boldsymbol{M}}\widetilde{\mathbf{P}}_{\boldsymbol{M},\hat{\boldsymbol{m}}}\mathbf{T}_{\hat{\boldsymbol{m}}}. \tag{2.9}$$

Hereby, the d-dimensional analogue of the twiddle matrix $\mathbf{T}_{\boldsymbol{m}} := \bigotimes_{t=0}^{d-1}\mathbf{T}_{m_t}$ is a diagonal matrix since it evolves as a tensor product of diagonal matrices. Note that the application of the diagonal matrices $\mathbf{T}_{\boldsymbol{m}}$, $\mathbf{T}_{\hat{\boldsymbol{m}}}$ can be easily performed as a point-wise scaling of the multidimensional input array. Analogously to (2.7) we can factorize

$$\widetilde{\mathbf{P}}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}}\mathbf{F}_{\boldsymbol{M}}\widetilde{\mathbf{P}}_{\boldsymbol{M},\hat{\boldsymbol{m}}} = \widetilde{\mathbf{A}}_0\widetilde{\mathbf{A}}_1\cdots\widetilde{\mathbf{A}}_{d-1}, \tag{2.10}$$

where for $t = 0,\ldots,d-1$ the matrix factors $\widetilde{\mathbf{A}}_t$ are defined as

$$\widetilde{\mathbf{A}}_t := (\mathbf{I}_{m_0}\otimes\cdots\otimes\mathbf{I}_{m_{t-1}})\otimes\widetilde{\mathbf{P}}_{M_t,m_t}^{\mathsf{T}}\mathbf{F}_{M_t}\widetilde{\mathbf{P}}_{M_t,\hat{m}_t}\otimes(\mathbf{I}_{\hat{m}_{t+1}}\otimes\cdots\otimes\mathbf{I}_{\hat{m}_{d-1}}).$$

Finally, combination of (2.9) and (2.10) yields Algorithm 2.3 for the computation of the multidimensional pruned FFT with shifted index sets. The benefits of Algorithm 2.3 for parallel computations will become clear when we come to parallel data decomposition. A detailed discussion can be found in Sections 2.6 and 2.7.

Remark 2.4. A comparison of Algorithm 2.2 and Algorithm 2.3 reveals that a multidimensional pruned FFT can be easily extended to shifted index sets by the following three steps. First, we need to add a simple scaling of the FFT inputs and outputs given by the twiddle matrices $\mathbf{T}_{\hat{\boldsymbol{m}}}$ and $\mathbf{T}_{\boldsymbol{m}}$ in Algorithm 2.3. Second, we replace $\mathbf{P}_{M_t,\hat{m}_t}$ by $\widetilde{\mathbf{P}}_{M_t,\hat{m}_t}$, which means that the zero padding takes place at another position in the input vectors. Last, the FFT outputs are truncated at a different position due to the exchange of $\widetilde{\mathbf{P}}_{M_t,m_t}^{\mathsf{T}}$ for $\mathbf{P}_{M_t,m_t}^{\mathsf{T}}$. Therefore, we focus in the following derivations on a parallel FFT framework based on the serial Algorithm 2.2. Once a parallel counterpart of Algorithm 2.2 is found, Algorithm 2.3 can be parallelized analogously by a simple exchange of $\mathbf{P}_{M_t,\hat{m}_t}^{\mathsf{T}}$ and $\mathbf{P}_{M_t,m_t}^{\mathsf{T}}$. □

Algorithm 2.3 Multidimensional pruned FFT with shifted index sets

Input: $\hat{\boldsymbol{f}} \in \mathbb{C}^{\hat{m}_0 \cdots \hat{m}_{d-1}}, \hat{m}_0, \ldots, \hat{m}_{d-1} \in 2\mathbb{N}$

1: $\boldsymbol{f} \leftarrow \mathbf{T}_{\hat{\boldsymbol{m}}} \hat{\boldsymbol{f}}$
2: **for** $t = d-1, \ldots, 0$ **do**
3: $\quad \boldsymbol{f} \leftarrow (\mathbf{I}_{m_0} \otimes \cdots \otimes \mathbf{I}_{m_{t-1}}) \otimes \widetilde{\mathbf{P}}^{\mathsf{T}}_{M_t, m_t} \mathbf{F}_{M_t} \widetilde{\mathbf{P}}_{M_t, \hat{m}_t} \otimes (\mathbf{I}_{\hat{m}_{t+1}} \otimes \cdots \otimes \mathbf{I}_{\hat{m}_{d-1}}) \boldsymbol{f}$
4: **end for**
5: $\boldsymbol{f} \leftarrow \prod_{t=0}^{d-1} (-1)^{(M_t + m_t + \hat{m}_t)/2} \cdot \mathbf{T}_{\boldsymbol{m}} \boldsymbol{f}$

Output: $\hat{\boldsymbol{f}} = \widetilde{\mathbf{P}}^{\mathsf{T}}_{M,\boldsymbol{m}} \mathbf{S}_M \mathbf{F}_M \mathbf{S}_M \widetilde{\mathbf{P}}_{M,\hat{\boldsymbol{m}}} \boldsymbol{f} \in \mathbb{C}^{m_0 \cdots m_{d-1}}, \hat{m}_0, \ldots, \hat{m}_{d-1} \in 2\mathbb{N}$

Algorithm 2.4 Multidimensional pruned FFT$^{\mathsf{H}}$ - Variant B

Input: $\boldsymbol{f} \in \mathbb{C}^{m_0 \cdots m_{d-1}}$

1: $\hat{\boldsymbol{f}} \leftarrow \boldsymbol{f}$
2: **for** $t = 0, \ldots, d-1$ **do**
3: $\quad \hat{\boldsymbol{f}} \leftarrow (\mathbf{I}_{m_0} \otimes \cdots \otimes \mathbf{I}_{m_{t-1}}) \otimes \mathbf{P}^{\mathsf{T}}_{M_t, \hat{m}_t} \mathbf{F}^{\mathsf{H}}_{M_t} \mathbf{P}_{M_t, m_t} \otimes (\mathbf{I}_{\hat{m}_{t+1}} \otimes \cdots \otimes \mathbf{I}_{\hat{m}_{d-1}}) \hat{\boldsymbol{f}}$
4: **end for**

Output: $\hat{\boldsymbol{f}} = \mathbf{P}^{\mathsf{T}}_{M,\hat{\boldsymbol{m}}} \mathbf{F}^{\mathsf{H}}_M \mathbf{P}_{M,\boldsymbol{m}} \boldsymbol{f} \in \mathbb{C}^{\hat{m}_0 \cdots \hat{m}_{d-1}}$

Remark 2.5. So far we considered matrix factorizations and fast algorithms for computing DFTs. Note that the corresponding adjoint algorithms are immediately given by the adjoint matrix factorization of the DFT. For example, the adjoint factorization of (2.7) is given by

$$\mathbf{P}^{\mathsf{T}}_{M,\hat{\boldsymbol{m}}} \mathbf{F}^{\mathsf{H}}_M \mathbf{P}_{M,\boldsymbol{m}} = \mathbf{A}^{\mathsf{H}}_{d-1} \cdots \mathbf{A}^{\mathsf{H}}_0.$$

and yields Algorithm 2.4 for computing the multidimensional pruned FFT$^{\mathsf{H}}$; cf. the non-adjoint Algorithm 2.2. □

2.4 Basic modules for multidimensional array transformation

Our parallel FFT framework will be a composition of several multidimensional array transformations. During these transformations the size of the data array may change and the dimensions of the array may be transposed. We will also see that all of these transformations can be realized in-place and out-of-place. In order to focus on the algorithmic workflow we introduce an appropriate shorthand notation that concentrates on the absolutely essential details about the multidimensional transformations

and data layout. The starting point is that we abbreviate a fixed d-dimensional array $\boldsymbol{g} \in \mathbb{C}^{M_0 \times \cdots \times M_{d-1}}$ simply by its size $M_0 \times \cdots \times M_{d-1}$. Furthermore, we declare that arrays given in this notation are stored contiguously in memory following a linearized row-major order, i.e., the entry of $\boldsymbol{g}$ at position $\boldsymbol{k} = (k_t)_{t=0}^{d-1}$, $0 \leq k_t < M_t$ can be found in memory at position $\sum_{t=0}^{d-1} k_t \prod_{r=t+1}^{d-1} M_r$. In the following, we will see that this notation can be extended naturally to express the effects of several multidimensional array transformations.

2.4.1 The serial FFT module

Assume a three-dimensional input array $\hat{\boldsymbol{g}} \in \mathbb{C}^{h_0 \times \hat{m} \times h_1}$. We introduce the notation

$$h_0 \times \hat{m} \times h_1 \overset{\mathsf{FFT}}{\rightarrow} h_0 \times m \times h_1 \tag{2.11}$$

to abbreviate an application of the matrix $\mathbf{I}_{h_0} \otimes \mathbf{P}_{M,m}^{\mathsf{T}} \mathbf{F}_M \mathbf{P}_{M,\hat{m}} \otimes \mathbf{I}_{h_1}$, i.e., we compute $h_0 \cdot h_1$ pruned one-dimensional FFTs along the second dimension of $\hat{\boldsymbol{g}}$ with stride h_1. Note that we do not compute the one-dimensional FFTs along the first dimension h_0. Later, we will use h_0 to store the parallel distributed dimensions. The additional dimension h_1 at the end of the array allows us to compute a set of h_1 serial FFTs at once. We will see that h_1 can be used in a higher-dimensional setting to store all the dimensions where the FFT was already performed. Analogously, we define

$$h_0 \times m \times h_1 \overset{\mathsf{FFT}}{\rightarrow} h_0 \times \hat{m} \times h_1$$

as a set of $h_0 \cdot h_1$ pruned one-dimensional adjoint FFTs along the second dimension with stride h_1. Note that the position of the hat distinguishes non-adjoint and adjoint FFTs in our shorthand notation. This module can be implemented by one appropriately chosen FFTW plan. At this point it is important to recall that FFTW chooses the best suitable plan for a given problem and hardware out of a large collection of different FFT algorithms. Therefore, our serial FFT module inherits the automatic hardware adaptivity. Moreover, the application of FFTW introduces much flexibility to our serial FFT module. For example, this module can be executed in-place and applies fast $\mathcal{O}(M \log M)$ algorithms also for prime size FFTs.

Remark 2.6. For sake of simplicity, we restrict the serial FFT module to complex input data. However, the serial FFT transforms can be easily replaced by specialized FFT algorithms that take into account the additional symmetries of real input data as shown in [4]. In principle any one-dimensional transformation can be applied instead of an FFT. □

Remark 2.7. Note that the serial FFT module can be easily extended to support pruned FFT with shifted index sets. In this case it is sufficient to implement the matrix operation given by $\mathbf{I}_{h_0} \otimes \widetilde{\mathbf{P}}_{M,m}^{\mathsf{T}} \mathbf{F}_M \widetilde{\mathbf{P}}_{M,\hat{m}} \otimes \mathbf{I}_{h_1}$, see also Remark 2.4. □

2.4.2 The local transposition module

The following shorthand notation describes a transposition of a three-dimensional array $\hat{\boldsymbol{g}} \in \mathbb{C}^{h_0 \times \hat{m} \times h_1}$ along its first two dimensions

$$\hat{m} \times h_0 \times h_1 \underset{\mathsf{TI}}{\rightarrow} h_0 \times \hat{m} \times h_1 \tag{2.12}$$

and

$$h_0 \times \hat{m} \times h_1 \underset{\mathsf{TO}}{\rightarrow} \hat{m} \times h_0 \times h_1. \tag{2.13}$$

Thereby, we can choose whether the input (TI) or the output (TO) array should be transposed, respectively.

This module can be realized by a single appropriately chosen FFTW plan. Although array transposition is a standard task in matrix computations, its efficient implementation is indeed a nontrivial task. Especially, one has to think of many details about the memory hierarchy of current computer architectures. At this point, we benefit of the cache oblivious array transpositions [59] that are implemented within FFTW. This means that the asymptotic number of cache misses is minimized independently of the cache size.

2.4.3 Combination of serial FFT and local transposition

A combination of the serial FFT module (2.11) and local transpositions (2.12), (2.13) results in the following two three-dimensional array transformations

$$\begin{aligned} \hat{m} \times h_0 \times h_1 &\overset{\mathsf{FFT}}{\underset{\mathsf{TI}}{\rightarrow}} h_0 \times m \times h_1, \\ h_0 \times \hat{m} \times h_1 &\overset{\mathsf{FFT}}{\underset{\mathsf{TO}}{\rightarrow}} m \times h_0 \times h_1. \end{aligned} \tag{2.14}$$

Both realize a set of pruned one-dimensional FFTs. However, the first one starts with an input array that is transposed in the first two dimensions, while the second one ends up in a transposed output array. Note that we do not specify the order of execution between the transposition and the FFTs a priori. For example, we can implement the first transform with only one appropriate chosen FFTW plan. But we are also free to use two successive plans to perform the serial FFT first and the transpose in the second step as

$$\hat{m} \times h_0 \times h_1 \overset{\mathsf{FFT}}{\rightarrow} m \times h_0 \times h_1 \underset{\mathsf{TI}}{\rightarrow} h_0 \times m \times h_1$$

or the other way around as follows

$$\hat{m} \times h_0 \times h_1 \underset{\mathsf{TI}}{\rightarrow} h_0 \times \hat{m} \times h_1 \overset{\mathsf{FFT}}{\rightarrow} h_0 \times m \times h_1.$$

We leave it to a planner to find the fastest of these three algorithms at runtime. The serial FFT with transposed output is handled analogously.

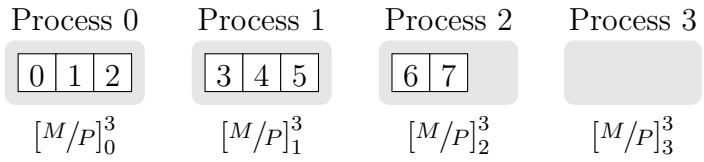

Figure 2.3: An array of size $M = 8$ gets block decomposed on $P = 4$ processes with block size $B = 3$. The picture shows the resulting local blocks $[M/P]_j^B$ for all processes $j \in \{0, 1, 2, 3\}$.

2.4.4 Parallel block decomposition

In the following, we introduce the data decomposition that is used for our parallel FFT algorithms. We start with the definition of a one-dimensional block decomposition. Afterward, we discuss the straightforward generalization to higher-dimensional settings.

Assume a one-dimensional array $\hat{\boldsymbol{g}} = (\hat{g}_k)_{k=0}^{M-1}$ of complex numbers that should be distributed on $P \in \mathbb{N}$ processes. We use the notation $[M/P]_j^B$ to symbolize that all data elements $\hat{g}_k$ with $jB \leq k < (j+1)B$ belong to process $j \in \{0, \ldots, P-1\}$. Hereby, the block size $B \in \mathbb{N}$ must be chosen $B \geq M/P$ such that no data remains undistributed. We see that the first $\lceil M/B \rceil$ processes get contiguous data blocks of block length B. Thereby, we used the notation $\lceil x \rceil := \min\{z \in \mathbb{Z} : x \leq z\}$ for rounding $x \in \mathbb{R}$ up. For the sake of convenience, we skip the index j in the aforementioned block notations whenever a statement is valid for all processes $j = 0, \ldots, P-1$. This means $[M/P]^B$ can be read as $[M/P]_j^B$ for all $j = 0, \ldots, P-1$. Analogously, we skip the block size B whenever a statement is valid for any integer block size $B \geq M/P$. Figure 2.3 shows an example of a one-dimensional block decomposition.

Of course, we are interested in block sizes that imply an almost equal data distribution on all processes. One way to choose the block size automatically is to set the default value $B = \lceil M/P \rceil$ in correspondence to the definition of the parallel FFTW interface. If M is divisible by P, the default block size is simply $B = M/P$ and results in P equal pieces each consisting of M/P contiguous data elements of $\hat{\boldsymbol{g}}$ as illustrated in Figure 2.4.

Multidimensional block decompositions are defined straightforward by the blocks that result from one-dimensional block decomposition along each dimension. In combination with the shorthand array notation that was introduced at the beginning of Section 2.4 we are now able to express multidimensional arrays that are block distributed along several dimensions. For example, assume a three-dimensional array $(\hat{g}_{k_0,k_1,k_2})_{k_0,k_1,k_2=0}^{M_0-1,M_1-1,M_2-1}$ of size $M_0 \times M_1 \times M_2$ that is block distributed on a Cartesian process mesh of size $P_0 \times P_1$ along the first two dimensions. Then, we write $[M_0/P_0]_i^{B_0} \times [M_1/P_1]_j^{B_1} \times M_2$ to express that process $(i, j) \in \{0, \ldots, P_0 - 1\} \times \{0, \ldots, P_1 - 1\}$ owns

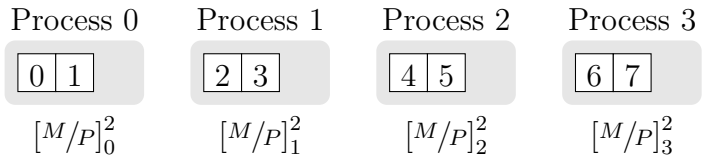

Figure 2.4: An array of size $M = 8$ gets block decomposed on $P = 4$ processes with default block size $B = 2$. The picture shows the resulting local blocks $[M/P]_j^B$ for all processes $j \in \{0, 1, 2, 3\}$. Since M is divisible by P, the default block size decomposition yields perfect load balancing.

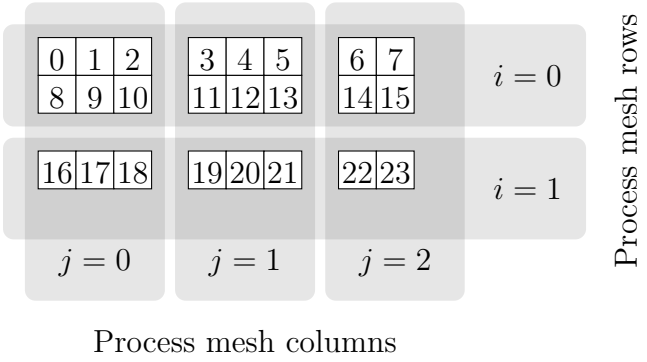

Figure 2.5: A two-dimensional array of size $M_0 \times M_1 = 3 \times 8$ gets block decomposed on a two-dimensional process mesh of size $P_0 \times P_1 = 2 \times 3$ with default block sizes $B_0 = 2$ and $B_1 = 3$. The picture shows the mapping of the local array blocks $[M_0/P_0]_i^{B_0} \times [M_1/P_1]_j^{B_1}$ to each process (i, j). Hereby, i and j are running indexes along the process mesh rows and columns, respectively.

all data $\hat{g}_{k_0,k_1,k_2}$ with $iB_0 \leq k_0 < (i+1)B_0$, $jB_1 \leq k_1 < (j+1)B_1$, and $0 \leq k_2 < M_2$. Thereby, we declare all data blocks to be stored in row-major order. In Figure 2.5 we present an illustration of a two-dimensional block decomposition with unequal block sizes. Furthermore, we define the notation $[(M_0 \times M_1)/P]_j^B$ by a one-dimensional block decomposition that results from the following two steps. First, linearize the two-dimensional array $M_0 \times M_1$ in row-major order and, second, block decompose this linearized array on P processes with block size B. In this case, block size $B \geq M_0 M_1/P$ must be fulfilled in order to distribute all data.

2.4.5 Folding multidimensional arrays in row-major order

An important observation is that an one-dimensional block decomposition of a two-dimensional array can be interpreted as a block decomposed one-dimensional array with suitably chosen block size. More precisely, let a size $M_0 \times M_1$ array be block

decomposed on P processes with block size B. Then,

$$\left[\tfrac{M_0}{P}\right]^B \times M_1 = \left[\tfrac{M_0 \times M_1}{P}\right]^{BM_1}. \tag{2.15}$$

Recall that $\left[{}^{(M_0 \times M_1)}/_P\right]^{BM_1}$ means the block decomposition of the one-dimensional array that is generated by the linearization of $M_0 \times M_1$ in row-major order. Thereby, the block size is chosen as a multiple of M_1 in order to avoid fragmentation along the second dimension. Figure 2.6 gives a graphical illustration of (2.15). Note

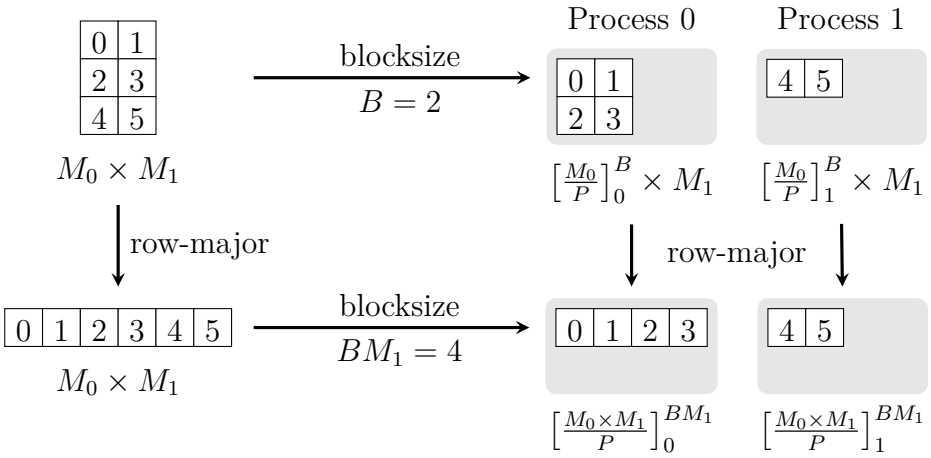

Figure 2.6: A two-dimensional array of size $M_0 \times M_1 = 3 \times 2$ is block decomposed along the first dimension on $P = 2$ processes with default block size $B = \lceil {}^{M_0}/_P \rceil = 2$. Afterward, the local blocks are stored in row-major order. This is equivalent to the block decomposition of the linearized array with block size $BM_1 = 4$.

that the row-major memory layout implies that this relation is only valid for block decompositions along the first dimension. In general we have

$$M_0 \times \left[\tfrac{M_1}{P}\right]^B \neq \left[\tfrac{M_0 \times M_1}{P}\right]^{BM_0} \tag{2.16}$$

as illustrated in Figure 2.7. However, if we used column-major memory order for linearization of multidimensional arrays, the validity of (2.16) and (2.15) would be the other way around.

2.4.6 Parallel matrix transposition

In the following, we introduce the algorithms for parallel transposition of a bunch of $h \in \mathbb{N}$ block decomposed matrices. A generalization to higher-dimensional settings is given in the next section. Assume a three-dimensional array of $M_0 \times M_1 \times h$ complex numbers. This can be interpreted as the a bunch of h interleaved matrices

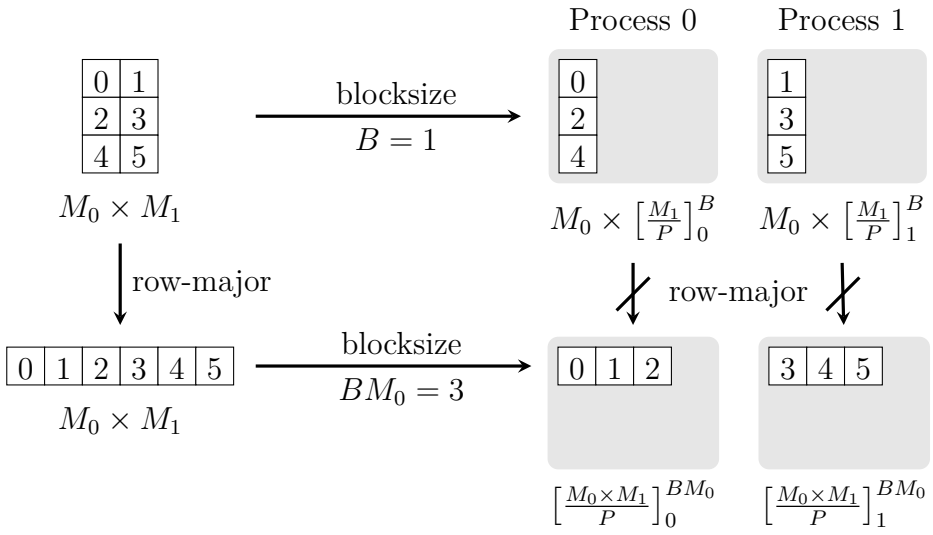

Figure 2.7: A two-dimensional array of size $M_0 \times M_1 = 3 \times 2$ is block decomposed along the second dimension on $P = 2$ processes with default block size $B = \lceil M_1/P \rceil = 1$. Afterward, the local blocks are stored in row-major order. In contrast to Figure 2.6 this is not equivalent to the block decomposition of the linearized array with block size $BM_0 = 3$.

each of size $M_0 \times M_1$. A parallel matrix transposition on P processes is defined as the transfer between the two following block decompositions

$$\left\lceil \frac{M_0}{P} \right\rceil^{B_0} \times M_1 \times h \xrightarrow{\mathsf{T}} \left\lceil \frac{M_1}{P} \right\rceil^{B_1} \times M_0 \times h, \tag{2.17}$$

with some integer block sizes $B_0 \geq M_0/P$ and $B_1 \geq M_1/P$. Note that h matrices of size $M_0 \times M_1$ are transposed at once and the block decomposition switches to the second matrix dimension. Furthermore, we define so-called parallel matrix transpositions with transposed input as

$$M_1 \times \left\lceil \frac{M_0}{P} \right\rceil^{B_0} \times h \xrightarrow[\mathsf{TI}]{\mathsf{T}} \left\lceil \frac{M_1}{P} \right\rceil^{B_1} \times M_0 \times h, \tag{2.18}$$

and with transposed output as

$$\left\lceil \frac{M_0}{P} \right\rceil^{B_0} \times M_1 \times h \xrightarrow[\mathsf{TO}]{\mathsf{T}} M_0 \times \left\lceil \frac{M_1}{P} \right\rceil^{B_1} \times h. \tag{2.19}$$

The FFTW library provides a variety of algorithms that are suitable to perform the parallel matrix transpositions (2.17), (2.18), and (2.19). These algorithms are also used for the one-dimensional decomposed parallel FFT implementations within FFTW. We stress that this collection of algorithms also includes in-place matrix transpositions, which means that per process only one additional data buffer of size

$(M_0 \times M_1 \times h)/P^2$ is necessary. Again, a planner searches for the fastest, applicable algorithm at runtime. Although it is not obvious, we will show that it is possible to reduce all the global communication of a parallel multidimensional FFT to the aforementioned parallel matrix transposition. Therefore, the high performance and flexibility of the FFTW matrix transposition is elementary for our parallel FFT. This provides us with portable and hardware adaptive communication functions.

2.4.7 The parallel multidimensional transposition module

Now, we show that the parallel matrix transpositions defined in Section 2.4.6 can be used to perform even more sophisticated parallel multidimensional array transpositions. Therefore, we will be able to perform more complex array transformations but still benefit from the high performance matrix transpositions. Assume a five dimensional array of size $L_0 \times h_0 \times L_1 \times h_1 \times h_2$ that is block distributed on P processes along the first dimension with block size B_0, i.e.,

$$\left[\tfrac{L_0}{P}\right]^{B_0} \times h_0 \times L_1 \times h_1 \times h_2.$$

By using (2.15) we can fold this array into a three-dimensional array of size

$$\left[\tfrac{M_0}{P}\right]^{B_0 h_0} \times M_1 \times h.$$

with $M_0 = L_0 \times h_0$, $M_1 = L_1 \times h_1$, $h = h_2$. Note that the block size must be a multiple of h_0 in order to fit to the row-major memory order; cf. Figures 2.6 and 2.7. Application of the parallel matrix transposition algorithms given in (2.17) with block size $B_1 h_1$ for the output decomposition yields

$$\left[\tfrac{M_0}{P}\right]^{B_0 h_0} \times M_1 \times h \xrightarrow{\mathsf{T}} \left[\tfrac{M_1}{P}\right]^{B_1 h_1} \times M_0 \times h.$$

Using (2.15) again, we can unfold the output array into

$$\left[\tfrac{L_1}{P}\right]^{B_1} \times h_1 \times L_0 \times h_0 \times h_2.$$

In summary, we have shown that an implementation of the parallel matrix transposition is sufficient to perform the following more general parallel multidimensional array transposition

$$\left(\left[\tfrac{L_0}{P}\right]^{B_0} \times h_0\right) \times \left(L_1 \times h_1\right) \times h_2 \xrightarrow{\mathsf{T}} \left(\left[\tfrac{L_1}{P}\right]^{B_1} \times h_1\right) \times \left(L_0 \times h_0\right) \times h_2.$$

Analogously, we can derive the parallel multidimensional array transposition with transposed input

$$\left(L_1 \times h_1\right) \times \left(\left[\tfrac{L_0}{P}\right]^{B_0} \times h_0\right) \times h_2 \xrightarrow[\mathsf{TI}]{\mathsf{T}} \left(\left[\tfrac{L_1}{P}\right]^{B_1} \times h_1\right) \times \left(L_0 \times h_0\right) \times h_2, \qquad (2.20)$$

and with transposed output

$$\left(\left[\tfrac{L_0}{P}\right]^{B_0} \times h_0\right) \times \left(L_1 \times h_1\right) \times h_2 \xrightarrow[\mathsf{TO}]{\mathsf{T}} \left(L_0 \times h_0\right) \times \left(\left[\tfrac{L_1}{P}\right]^{B_1} \times h_1\right) \times h_2 \tag{2.21}$$

from the parallel matrix transpositions (2.18) and (2.19), respectively. Later on, we will see that (2.20) and (2.21) are sufficient to handle all the global communication within our parallel FFT frameworks.

2.5 Parallel FFT with multidimensional data decomposition

Now, we have collected all the ingredients to formulate the parallel FFT framework that allow us to calculate a bunch of $h \in \mathbb{N}$ multidimensional pruned FFTs of size

$$\hat{m}_0 \times \cdots \times \hat{m}_{d-1} \times h \overset{\mathsf{FFT}}{\rightarrow} m_0 \times \cdots \times m_{d-1} \times h$$

in parallel on a process mesh of size $P_0 \times \cdots \times P_{r-1}$ with $r < d$. Let a r-dimensional block decomposition of the input array be given by

$$\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-1}}{P_{r-1}}\right] \times \hat{m}_r \times \hat{m}_{r+1} \times \cdots \times \hat{m}_{d-1} \times h.$$

We see that the last $d - r + 1$ dimensions of the input array are locally available on each process. Therefore, the first step of our framework consists of the computation of all pruned FFTs corresponding to dimensions $r + 1$ until $d - 1$ which results in the partially transformed multidimensional array

$$\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-1}}{P_{r-1}}\right] \times \hat{m}_r \times m_{r+1} \times \cdots \times m_{d-1} \times h.$$

After this step, only one non-transformed dimension remains locally available on each process. In the following, we compute the serial pruned FFT along this dimension and transpose it to the front, i.e.,

$$\begin{aligned} &\left(\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-1}}{P_{r-1}}\right]\right) \times \hat{m}_r \times (m_{r+1} \times \cdots \times m_{d-1} \times h) \\ \xrightarrow[\mathsf{TO}]{\mathsf{FFT}} &m_r \times \left(\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-1}}{P_{r-1}}\right]\right) \times (m_{r+1} \times \cdots \times m_{d-1} \times h)\,. \end{aligned}$$

Hereby, the extra brackets denote that we call the three-dimensional array transformation (2.14) with the substitutions $h_0 \leftarrow [\hat{m}_0/P_0] \times \cdots \times [\hat{m}_{r-1}/P_{r-1}]$, $\hat{m} \leftarrow \hat{m}_r$, and $h_1 \leftarrow m_{r+1} \times \cdots \times m_{d-1} \times h$. Next, we apply a parallel transposition with transposed input

$$\begin{aligned} &m_r \times \left(\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-2}}{P_{r-2}}\right]\right) \times \left[\tfrac{\hat{m}_{r-1}}{P_{r-1}}\right] \times (m_{r+1} \times \cdots \times m_{d-1} \times h) \\ \xrightarrow[\mathsf{TI}]{\mathsf{T}} &\left[\tfrac{m_r}{P_{r-1}}\right] \times \left(\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-2}}{P_{r-2}}\right]\right) \times \hat{m}_{r-1} \times (m_{r+1} \times \cdots \times m_{d-1} \times h) \end{aligned}$$

according to (2.20) with the substitutions $L_1 \leftarrow m_r$, $h_1 \leftarrow [\hat{m}_0/P_0] \times \cdots \times [\hat{m}_{r-2}/P_{r-2}]$, $L_0 \leftarrow \hat{m}_{r-1}$, $h_0 \leftarrow 1$, $h_2 \leftarrow m_{r+1} \times \cdots \times m_{d-1} \times h$, and $P \leftarrow P_{r-1}$. The last two steps are repeated analogously for another $r-1$ times to end up with the following partially transformed array

$$\left[\tfrac{m_1}{P_0}\right] \times \cdots \times \left[\tfrac{m_r}{P_{r-1}}\right] \times \hat{m}_0 \times m_{r+1} \times \cdots \times m_{d-1} \times h.$$

Finally, a non-transposed, serial pruned FFT (2.11) with the substitutions $h_0 \leftarrow [m_1/P_0] \times \cdots \times [m_r/P_{r-1}]$, $\hat{m} \leftarrow \hat{m}_0$, and $h_1 \leftarrow m_{r+1} \times \cdots \times m_{d-1} \times h$ finishes the d-dimensional FFTs

$$\begin{aligned} &\left[\tfrac{m_1}{P_0}\right] \times \cdots \times \left[\tfrac{m_r}{P_{r-1}}\right] \times \hat{m}_0 \times m_{r+1} \times \cdots \times m_{d-1} \times h \\ \xrightarrow{\mathsf{FFT}} &\left[\tfrac{m_1}{P_0}\right] \times \cdots \times \left[\tfrac{m_r}{P_{r-1}}\right] \times m_0 \times m_{r+1} \times \cdots \times m_{d-1} \times h. \end{aligned} \tag{2.22}$$

Note that the first r dimensions of the output array are transposed. For convenience, we introduce the notation

$$\bigtimes_{s=l}^{u} m_s := \begin{cases} m_l \times \cdots \times m_u & : l \leq u, \\ 1 & : l > u. \end{cases}$$

In summary, we get the Parallel Fast Fourier Transform (PFFT) Framework 2.5 for computing a parallel multidimensional, pruned FFT with multidimensional data decomposition. For the sake of completeness, we also present the Parallel Adjoint Fast Fourier Transform (PFFT$^{\mathsf{H}}$) Framework 2.6 for computing multidimensional, pruned FFT$^{\mathsf{H}}$. Note that the latter Framework is simply the adjoint of Framework 2.6, cf. Remark 2.5. Consequently, the adjoint Framework starts with the output decomposition of the PFFT Framework

$$\left[\tfrac{m_1}{P_0}\right] \times \cdots \times \left[\tfrac{m_{r-2}}{P_{r-1}}\right] \times m_0 \times m_r \times \cdots \times m_{d-1} \times h$$

and ends with the initial data decomposition of PFFT

$$\left[\tfrac{\hat{m}_0}{P_0}\right] \times \cdots \times \left[\tfrac{\hat{m}_{r-1}}{P_{r-1}}\right] \times \hat{m}_r \times \cdots \times \hat{m}_{d-1} \times h.$$

In the following, we present the most important special cases that are included in our parallel FFT Frameworks 2.5 and 2.6, i.e., one-, two-, and three-dimensional parallel data decomposition.

Framework 2.5 PFFT – Parallel, multidimensional, pruned FFT

Input: $\hat{\boldsymbol{g}} \in \mathbb{C}^{\hat{m}_0 \times \cdots \times \hat{m}_{d-1} \times h}$ block decomposed as
$\left[\hat{m}_0/P_0\right] \times \cdots \times \left[\hat{m}_{r-1}/P_{r-1}\right] \times \hat{m}_r \times \cdots \times \hat{m}_{d-1} \times h$

1: **for** $t \leftarrow 0, \ldots, d-r-2$ **do**
2: ▷ Initiate serial FFT module
3: $h_0 \leftarrow \bigtimes_{s=0}^{r-1} \left[\hat{m}_s/P_s\right] \times \bigtimes_{s=r}^{d-2-t} \hat{m}_s$
4: $\hat{m} \leftarrow \hat{m}_{d-1-t}$
5: $h_1 \leftarrow \bigtimes_{s=d-t}^{d-1} m_s \times h$
6: $h_0 \times \hat{m} \times h_1 \xrightarrow{\mathsf{FFT}} h_0 \times m \times h_1$
7: **end for**
8: **for** $t \leftarrow 0, \ldots, r-1$ **do**
9: ▷ Initiate serial FFT module
10: $h_0 \leftarrow \bigtimes_{s=r-t}^{r-1} \left[m_{s+1}/P_s\right] \times \bigtimes_{s=0}^{r-t-1} \left[\hat{m}_s/P_s\right]$
11: $\hat{m} \leftarrow \hat{m}_{r-t}$
12: $h_1 \leftarrow \bigtimes_{s=r+1}^{d-1} m_s \times h$
13: $h_0 \times \hat{m} \times h_1 \xrightarrow[\mathsf{TO}]{\mathsf{FFT}} m \times h_0 \times h_1$
14: ▷ Initiate global transposition module
15: $L_1 \leftarrow m_{r-t}$
16: $h_1 \leftarrow \bigtimes_{s=r-t}^{r-1} \left[m_{s+1}/P_s\right] \times \bigtimes_{s=0}^{r-t-2} \left[\hat{m}_s/P_s\right]$
17: $L_0 \leftarrow \hat{m}_{r-t-1}$
18: $h_0 \leftarrow 1$
19: $h_2 \leftarrow \bigtimes_{s=r+1}^{d-1} m_s \times h$
20: $P \leftarrow P_{r-t-1}$
21: $L_1 \times h_1 \times \left[L_0/P\right] \times h_0 \times h_2 \xrightarrow[\mathsf{TI}]{\mathsf{T}} \left[L_1/P\right] \times h_1 \times L_0 \times h_0 \times h_2$
22: **end for**
23: ▷ Initiate serial FFT module
24: $h_0 \leftarrow \bigtimes_{s=0}^{r-1} \left[m_{s+1}/P_s\right]$
25: $\hat{m} \leftarrow \hat{m}_0$
26: $h_1 \leftarrow \bigtimes_{s=r+1}^{d-1} m_s \times h$
27: $h_0 \times \hat{m} \times h_1 \xrightarrow{\mathsf{FFT}} h_0 \times m \times h_1$

Output: $\boldsymbol{g} = \mathbf{P}_{\boldsymbol{M},\boldsymbol{m}}^{\mathsf{T}} \mathbf{F}_{\boldsymbol{M}} \mathbf{P}_{\boldsymbol{M},\hat{\boldsymbol{m}}} \hat{\boldsymbol{g}} \in \mathbb{C}^{m_0 \times \cdots \times m_{d-1} \times h}$ block decomposed as
$\left[m_1/P_0\right] \times \cdots \times \left[m_{r-2}/P_{r-1}\right] \times m_0 \times m_r \times \cdots \times m_{d-1} \times h$

Framework 2.6 PFFT$^{\mathsf{H}}$ – Parallel, multidimensional, pruned FFT$^{\mathsf{H}}$

Input: $\boldsymbol{g} \in \mathbb{C}^{m_0 \times \cdots \times m_{d-1} \times h}$ block decomposed as
$[m_1/P_0] \times \cdots \times [m_{r-2}/P_{r-1}] \times m_0 \times m_r \times \cdots \times m_{d-1} \times h$

1: ▷ Initiate serial FFT module
2: $h_0 \leftarrow \bigtimes_{s=0}^{r-1} [m_{s+1}/P_s]$
3: $m \leftarrow m_0$
4: $h_1 \leftarrow \bigtimes_{s=r+1}^{d-1} m_s \times h$
5: $h_0 \times m \times h_1 \overset{\mathsf{FFT}}{\rightarrow} h_0 \times \hat{m} \times h_1$
6: **for** $t \leftarrow r-1, \ldots, 0$ **do**
7: ▷ Initiate global transposition module
8: $L_0 \leftarrow m_{r-t}$
9: $h_0 \leftarrow \bigtimes_{s=r-t}^{r-1} [m_{s+1}/P_s] \times \bigtimes_{s=0}^{r-t-2} [\hat{m}_s/P_s]$
10: $L_1 \leftarrow \hat{m}_{r-t-1}$
11: $h_1 \leftarrow 1$
12: $h_2 \leftarrow \bigtimes_{s=r+1}^{d-1} m_s \times h$
13: $P \leftarrow P_{r-t-1}$
14: $[L_0/P] \times h_0 \times L_1 \times h_1 \times h_2 \underset{\mathsf{TO}}{\overset{\mathsf{T}}{\rightarrow}} L_0 \times h_0 \times [L_1/P] \times h_1 \times h_2$
15: ▷ Initiate serial FFT module
16: $h_0 \leftarrow \bigtimes_{s=r-t}^{r-1} [m_{s+1}/P_s] \times \bigtimes_{s=0}^{r-t-1} [\hat{m}_s/P_s]$
17: $m \leftarrow m_{r-t}$
18: $h_1 \leftarrow \bigtimes_{s=r+1}^{d-1} m_s \times h$
19: $m \times h_0 \times h_1 \underset{\mathsf{TI}}{\overset{\mathsf{FFT}}{\rightarrow}} h_0 \times \hat{m} \times h_1$
20: **end for**
21: **for** $t \leftarrow d-r-2, \ldots, 0$ **do**
22: ▷ Initiate serial FFT module
23: $h_0 \leftarrow \bigtimes_{s=0}^{r-1} [\hat{m}_s/P_s] \times \bigtimes_{s=r}^{d-2-t} \hat{m}_s$
24: $m \leftarrow m_{d-1-t}$
25: $h_1 \leftarrow \bigtimes_{s=d-t}^{d-1} m_s \times h$
26: $h_0 \times m \times h_1 \overset{\mathsf{FFT}}{\rightarrow} h_0 \times \hat{m} \times h_1$
27: **end for**

Output: $\hat{\boldsymbol{g}} = \mathbf{P}^{\mathsf{T}}_{\boldsymbol{M},\hat{\boldsymbol{m}}} \mathbf{F}^{\mathsf{H}}_{\boldsymbol{M}} \mathbf{P}_{\boldsymbol{M},\boldsymbol{m}} \boldsymbol{g} \in \mathbb{C}^{\hat{m}_0 \times \cdots \times \hat{m}_{d-1} \times h}$ block decomposed as
$[\hat{m}_0/P_0] \times \cdots \times [\hat{m}_{r-1}/P_{r-1}] \times \hat{m}_r \times \cdots \times \hat{m}_{d-1} \times h$

Example 2.8. At first we present the PFFT Framework 2.5 for a three-dimensional FFT with one-dimensional data decomposition. Note that for this setting we essentially yield the framework that is implemented within the FFTW software library. Assume a three-dimensional array of size $\hat{m}_0 \times \hat{m}_1 \times \hat{m}_2$ that is distributed on a one-dimensional process mesh of size P_0. For this setting the PFFT Framework 2.5 becomes

$$\begin{aligned}\left[\tfrac{\hat{m}_0}{P_0}\right] \times \hat{m}_1 \times \hat{m}_2 &\overset{\mathsf{FFT}}{\rightarrow} \left[\tfrac{\hat{m}_0}{P_0}\right] \times \hat{m}_1 \times m_2 \overset{\substack{\mathsf{FFT}\\ \rightarrow\\ \mathsf{TO}}}{} m_1 \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times m_2 \\ &\overset{\substack{\mathsf{T}\\ \rightarrow\\ \mathsf{TI}}}{} \left[\tfrac{m_1}{P_0}\right] \times \hat{m}_0 \times m_2 \overset{\mathsf{FFT}}{\rightarrow} \left[\tfrac{m_1}{P_0}\right] \times m_0 \times m_2.\end{aligned}$$

The PFFT$^{\mathsf{H}}$ Framework 2.6 starts with the transposed input data and returns to the initial data distribution by simply reverting all steps of the PFFT Framework as follows

$$\begin{aligned}\left[\tfrac{m_1}{P_0}\right] \times m_0 \times m_2 &\overset{\mathsf{FFT}}{\rightarrow} \left[\tfrac{m_1}{P_0}\right] \times \hat{m}_0 \times m_2 \\ \overset{\substack{\mathsf{T}\\ \rightarrow\\ \mathsf{TO}}}{} m_1 \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times m_2 &\overset{\substack{\mathsf{FFT}\\ \rightarrow\\ \mathsf{TI}}}{} \left[\tfrac{\hat{m}_0}{P_0}\right] \times \hat{m}_1 \times m_2 \overset{\mathsf{FFT}}{\rightarrow} \left[\tfrac{\hat{m}_0}{P_0}\right] \times \hat{m}_1 \times \hat{m}_2.\end{aligned}$$

□

Example 2.9. This examples shows the work flow of our parallel FFT frameworks with two-dimensional data decomposition. In order to use two-dimensional data decomposition, the FFT of interest must be at least three-dimensional. However, note that this framework works analogously with four- or higher-dimensional FFTs. Assume a three-dimensional array of size $\hat{m}_0 \times \hat{m}_1 \times \hat{m}_2$ that is distributed on a two-dimensional process mesh of size $P_0 \times P_1$. For this setting the PFFT Framework 2.5 becomes

$$\begin{aligned}\left[\tfrac{\hat{m}_0}{P_0}\right] \times \left[\tfrac{\hat{m}_1}{P_1}\right] \times \hat{m}_2 &\overset{\substack{\mathsf{FFT}\\ \rightarrow\\ \mathsf{TO}}}{} m_2 \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times \left[\tfrac{\hat{m}_1}{P_1}\right] \\ \overset{\substack{\mathsf{T}\\ \rightarrow\\ \mathsf{TI}}}{} \left[\tfrac{m_2}{P_1}\right] \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times \hat{m}_1 &\overset{\substack{\mathsf{FFT}\\ \rightarrow\\ \mathsf{TO}}}{} m_1 \times \left[\tfrac{m_2}{P_1}\right] \times \left[\tfrac{\hat{m}_0}{P_0}\right] \\ \overset{\substack{\mathsf{T}\\ \rightarrow\\ \mathsf{TI}}}{} \left[\tfrac{m_1}{P_0}\right] \times \left[\tfrac{m_2}{P_1}\right] \times \hat{m}_0 &\overset{\mathsf{FFT}}{\rightarrow} \left[\tfrac{m_1}{P_0}\right] \times \left[\tfrac{m_2}{P_1}\right] \times m_0.\end{aligned}$$

The PFFT$^{\mathsf{H}}$ Framework 2.6 starts with the transposed input data and returns to the initial data distribution by simply reverting all steps of the PFFT Framework. □

Example 2.10. Now, we show the work flow of our parallel FFT framework with three-dimensional domain decomposition. In order to use three-dimensional data decomposition, the FFT of interest must be at least four-dimensional. However, note that this framework works analogously with higher-dimensional FFTs. As far as we know, our implementation of Framework 2.5 and Framework 2.6 is the only publicly available parallel FFT software that allows three-dimensional data decomposition at

all. Assume a four-dimensional array of size $\hat{m}_0 \times \hat{m}_1 \times \hat{m}_2 \times \hat{m}_3$ that is distributed on a three-dimensional process mesh of size $P_0 \times P_1 \times P_2$. For this setting the PFFT Framework 2.5 becomes

$$\begin{aligned}
&\left[\tfrac{\hat{m}_0}{P_0}\right] \times \left[\tfrac{\hat{m}_1}{P_1}\right] \times \left[\tfrac{\hat{m}_2}{P_2}\right] \times \hat{m}_3 \overset{\substack{\mathsf{FFT}\\ \mathsf{TO}}}{\rightarrow} m_3 \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times \left[\tfrac{\hat{m}_1}{P_1}\right] \times \left[\tfrac{\hat{m}_2}{P_2}\right] \\
\overset{\substack{\mathsf{T}\\ \mathsf{TI}}}{\rightarrow} &\left[\tfrac{m_3}{P_2}\right] \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times \left[\tfrac{\hat{m}_1}{P_1}\right] \times \hat{m}_2 \overset{\substack{\mathsf{FFT}\\ \mathsf{TO}}}{\rightarrow} m_2 \times \left[\tfrac{m_3}{P_2}\right] \times \left[\tfrac{\hat{m}_0}{P_0}\right] \times \left[\tfrac{\hat{m}_1}{P_1}\right] \\
\overset{\substack{\mathsf{T}\\ \mathsf{TI}}}{\rightarrow} &\left[\tfrac{m_2}{P_1}\right] \times \left[\tfrac{m_3}{P_2}\right] \times \left[\tfrac{m_0}{P_0}\right] \times \hat{m}_1 \overset{\substack{\mathsf{FFT}\\ \mathsf{TO}}}{\rightarrow} m_1 \times \left[\tfrac{m_2}{P_1}\right] \times \left[\tfrac{m_3}{P_2}\right] \times \left[\tfrac{\hat{m}_0}{P_0}\right] \\
\overset{\substack{\mathsf{T}\\ \mathsf{TI}}}{\rightarrow} &\left[\tfrac{m_1}{P_0}\right] \times \left[\tfrac{m_2}{P_1}\right] \times \left[\tfrac{m_3}{P_2}\right] \times \hat{m}_0 \overset{\mathsf{FFT}}{\rightarrow} \left[\tfrac{m_1}{P_0}\right] \times \left[\tfrac{m_2}{P_1}\right] \times \left[\tfrac{m_3}{P_2}\right] \times m_0.
\end{aligned}$$

The PFFT$^{\mathsf{H}}$ Framework 2.6 starts with the transposed input data and returns to the initial data distribution by simply reverting all steps of the PFFT Framework. □

Remark 2.11. The PFFT Framework 2.5 ends up with a transposed data decomposition (2.22). Using the same transposed data layout as input for the PFFT$^{\mathsf{H}}$ Framework 2.6 returns to the initial data layout after one FFT and FFT$^{\mathsf{H}}$. For most applications it is acceptable to deal with this transposed data layout between the two FFT executions. For example, a common use case for parallel FFT is the fast convolution of two signals. Thereby, we compute the FFT of both signals, multiply both results point-wise, and compute the FFT$^{\mathsf{H}}$. The point-wise multiplication can be trivially performed as long as both arrays share the same data layout in Fourier space. However, if transposed data layout is not acceptable we are able to reorder the data with a framework very similar to the PFFT$^{\mathsf{H}}$ Framework 2.6. Thereby, we simply skip the computation of all local FFTs but remain the local and global transpositions in Framework 2.6. Obviously, the non-transposed data layout comes at the cost of doubling the amount of communication and should be avoided whenever possible. □

2.6 Parallel pruned FFT

We have seen that the PFFT Framework 2.5 incorporates the computation of pruned FFTs in the flavor of the serial Algorithm 2.2, i.e., array dimensions are zero padded and truncated whenever they are locally available. An alternative would have been to combine the serial Algorithm 2.1 with a parallel block decomposition. This would have meant that we called the PFFT Framework on the zero padded input array and truncated its outputs all at once. However, this second approach yields several drawbacks for parallel execution as the following example will illustrate.

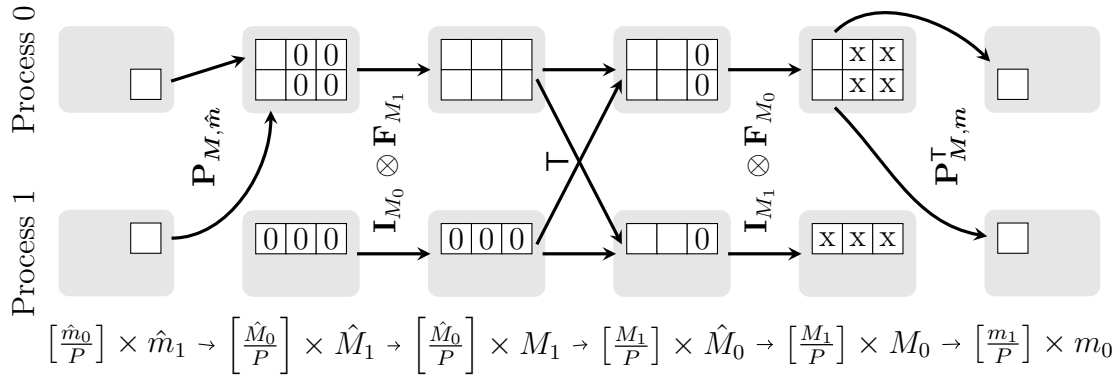

Figure 2.8: Parallel workflow of a two-dimensional pruned FFT on $P = 2$ processes with pruned input size $\hat{\boldsymbol{m}} = (2, 1)^{\mathsf{T}}$, FFT size $\boldsymbol{M} = (3, 3)^{\mathsf{T}}$, and pruned output size $\boldsymbol{m} = (1, 2)^{\mathsf{T}}$ based on the serial Algorithm 2.1.

Example 2.12. In the following, we compare two parallel approaches for computing the parallel pruned FFT (2.6). Therefore, assume a two-dimensional input array of size $\hat{\boldsymbol{m}} = (2, 1)^{\mathsf{T}}$. We are interested in the parallel computation of the size $\boldsymbol{m} = (1, 2)^{\mathsf{T}}$ output sub-array of a size $\boldsymbol{M} = (3, 3)^{\mathsf{T}}$ FFT on $P = 2$ processes. Note that although this example looks superficially small it already reveals all the important differences of the algorithms. At first, we discuss the parallel computation in correspondence to the serial pruned FFT Algorithm 2.1, see Figure 2.8. Algorithm 2.1 starts with an input array of size $[\hat{m}_0/P] \times \hat{m}_1$. First, it applies zero padding of both dimensions at once. Especially, this includes the first dimension that is parallel block decomposed at this stage. Therefore, the parallel block decomposition must be adjusted in order to fit the block sizes $[\hat{M}_0/P] \times \hat{M}_1$, where the notations $\hat{M}_0 := M_0$ and $\hat{M}_1 := M_1$ are introduced in order to symbolize the non-transformed array size before the FFT. In general, this data rearrangement implies a global communication as can be seen in Figure 2.8. Next, $\hat{M}_0$ one-dimensional FFTs of size $\hat{M}_1$ along the second dimension are computed. However, $\hat{M}_0 - \hat{m}_0$ of these FFTs can be skipped since they have zero inputs. Unfortunately, the redistribution of the first dimension implied that process 1 gets all of these unnecessary FFTs. Therefore, we have a 50% drop of computational efficiency at this stage. After the parallel matrix transposition, we want to compute M_1 one-dimensional FFTs of size $\hat{M}_0$ along the second dimension. In fact, only the first m_1 FFTs produce relevant outputs since the last $M_1 - m_1$ columns of the output array will be truncated due to the pruning size $\boldsymbol{m}$. In the picture above, truncated elements are symbolized with a cross. Again, we see that process 1 does not take part in the relevant computations. Finally, we redistribute the block decomposed FFT output array $[M_1/P] \times M_0$ in order to fulfill the block decomposition $[m_1/P] \times m_0$ of the truncated output array. In general, this implies another global communication. In summary, we lost 50% of

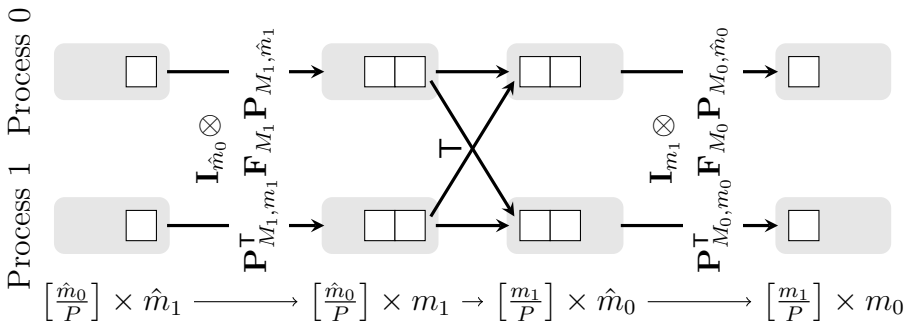

Figure 2.9: Parallel workflow of a two-dimensional pruned FFT on $P = 2$ processes with pruned input size $\hat{\boldsymbol{m}} = (2, 1)^{\mathsf{T}}$, FFT size $\boldsymbol{M} = (3, 3)^{\mathsf{T}}$, and pruned output size $\boldsymbol{m} = (1, 2)^{\mathsf{T}}$ based on the serial Algorithm 2.2.

the overall computational efficiency and got two additional global communications.

In contrast, Figure 2.9 illustrates the parallel approach that is part of our parallel FFT Frameworks 2.5 and 2.6. This approach is based on the serial Algorithm 2.2. At a glance we can see that the second approach looks more ordered and yields better load balancing. In fact the computational workload is perfectly balanced between the two processes and we do not have the two additional global communication steps of the first approach. Furthermore, we see that the memory consumption of the second approach is lower due to the partially pruned arrays and the perfect load balancing. Especially, the parallel matrix transposition of the first approach works on a larger array of size $\hat{M}_0 \times M_1$ while the second approach requires only a parallel matrix transposition of size $\hat{m}_0 \times m_1$. This means that the pruned PFFT approach is also superior in terms of data amount that must be communicated. All in all, we have seen that the second approach offers much more parallel scalability than the naive first approach for computing pruned FFTs in parallel. However, the preferable approach must be implemented at the library level, i.e., pruning has to be part of the parallel FFT algorithm and can not be added as a simple wrapper of an existing parallel FFT library. □

2.7 Parallel FFT with shifted index sets

We already pointed out in Remark 2.4 that we can use (2.9) in order to extend our parallel pruned FFT frameworks to shifted index sets. Now, we present an example that illustrates the differences of the two representations given in (2.9) when they are combined with parallel block decomposition.

Example 2.13. For the sake of simplicity we restrict ourselves to a two-dimensional FFT of size $\hat{\boldsymbol{m}} = \boldsymbol{m} = \boldsymbol{M} = (2, 4)^{\mathsf{T}}$, i.e., no pruning is applied and the size of the

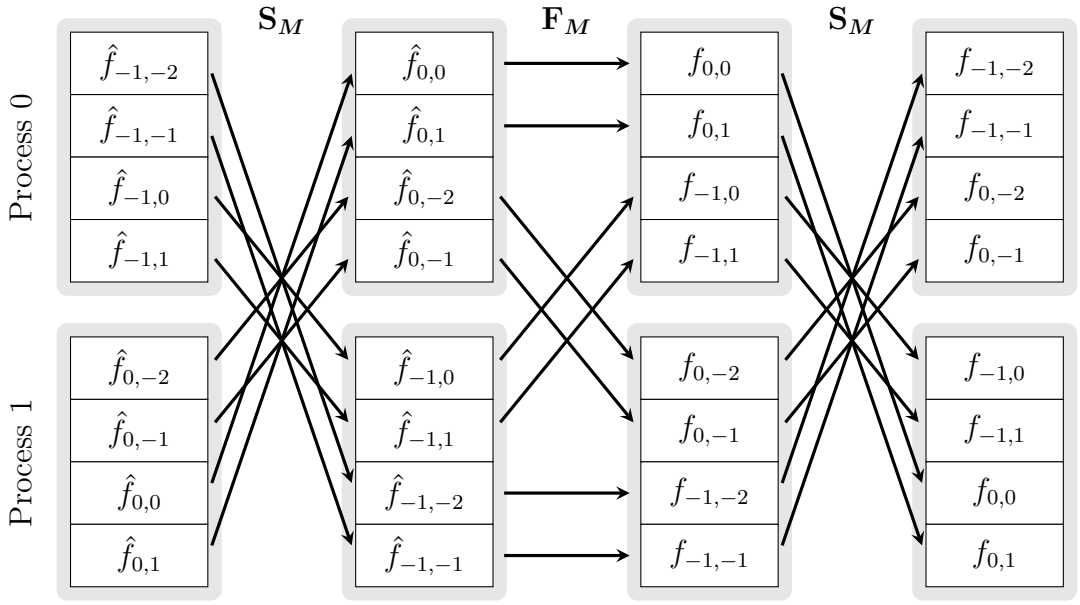

Figure 2.10: Parallel workflow of a two-dimensional FFT with shifted index sets on $P = 2$ processes with FFT size $\boldsymbol{M} = (2, 4)^{\mathsf{T}}$ based on the factorization $\mathbf{S}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{S}_{\boldsymbol{M}}$.

input array $\hat{\boldsymbol{f}} \in \mathbb{C}^{2\times 4}$ and output array $\boldsymbol{f} \in \mathbb{C}^{2\times 4}$ are equal to the FFT size. In this setting (2.9) can be simplified to

$$\mathbf{S}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{S}_{\boldsymbol{M}} = (-1)\mathbf{T}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{T}_{\boldsymbol{M}}. \tag{2.23}$$

We now apply a one-dimensional parallel block decomposition on $P = 2$ processes to both sides of this equation and compare the resulting parallel algorithms. Starting with the left hand side we get the algorithmic workflow presented in Figure 2.10. Thereby, we track the data movement of the one-dimensional decomposed array during each of the three steps. Note that the application of the shift matrix $\mathbf{S}_{\boldsymbol{M}}$ requires the two processes to exchange all their data before and after the parallel FFT framework is applied. This stands for two additional global communication steps. In contrast the right hand side of (2.23) only incorporates the diagonal matrices $\mathbf{T}_{\boldsymbol{M}}$ that do not change the parallel data decomposition. Mapping this representation to parallel data decomposition results in the parallel counterpart of Algorithm 2.3. Figure 2.11 shows an example of the parallel workflow for this approach. Note that the application of the twiddle matrix $\mathbf{T}_{\boldsymbol{M}}$ does not change the data order and, therefore, avoids any additional communication. The extra amount of computation due to the multiplications with (-1) will be more than compensated by the savings in communication, especially if we scale to large numbers of parallel processes. At this point it is clear why we choose Algorithm 2.3 as starting point for our parallel FFT frameworks with shifted index sets. □

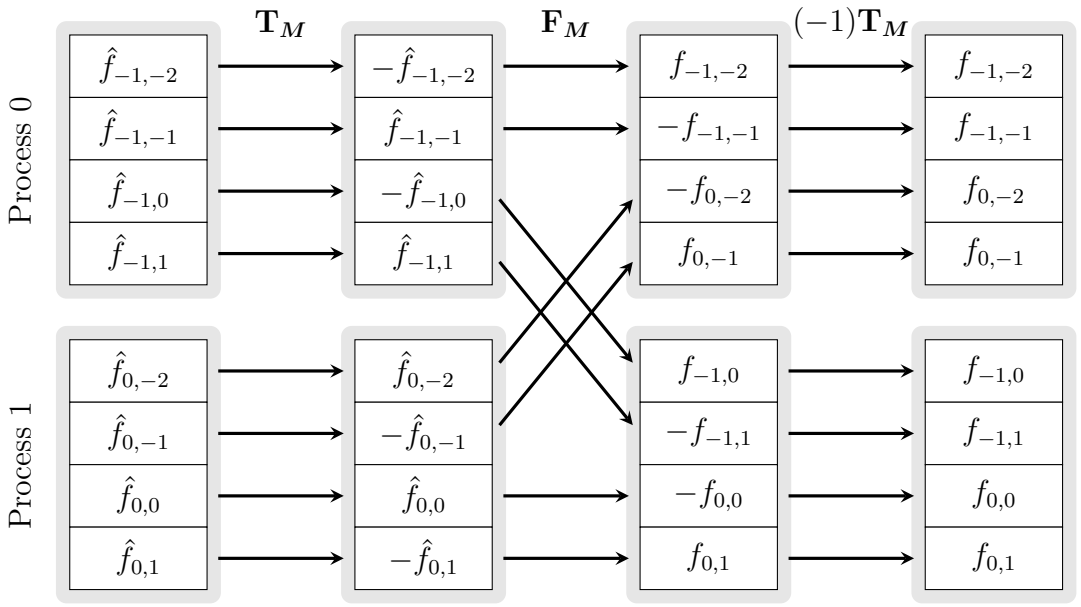

Figure 2.11: Parallel workflow of a two-dimensional FFT with shifted index sets on $P = 2$ processes with FFT size $\boldsymbol{M} = (2, 4)^{\mathsf{T}}$ based on the factorization $(-1)\mathbf{T}_{\boldsymbol{M}}\mathbf{F}_{\boldsymbol{M}}\mathbf{T}_{\boldsymbol{M}}$.

2.8 The ghost cell communication modules

The data arrays of our parallel FFT are distributed about the processes in distinct blocks. However, in many application one wants to perform calculations with a small set of data points that are direct neighbors, e.g., in the calculation of finite differences or the computation of interpolation values as we will need in Section 3.3. This becomes problematic when the border of the local data blocks is exceeded and the data is not locally available. Therefore, it is common practice to generate a thin layer of duplicated data around the local blocks. This operation is well known as ghost cell creation [64, Chap. 5.6.1]. Figure 2.12 gives an example of the corresponding matrix-vector operation for a one-dimensional block decomposition of $M = 4$ data points on $P = 2$ processes with default block size.

Closely related to this operation is the transposed ghost cell creation, also known as ghost cell reduction. The ghost cell reduction computes the sums of each original data element with all of its ghost cells. This operation is needed whenever one data point is computed as the sum of operations that must be performed on distinct processes due to data locality. Figure 2.13 shows the transposed counterpart of the ghost cell send example that was given in Figure 2.12.

We implemented several algorithms to perform the parallel ghost cell creation and reduction. Similar to FFTW a planner decides at runtime which algorithm is preferable. Note that our implementation also suits the case in which the number of

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ \hline c \\ d \end{pmatrix} = \begin{pmatrix} d \\ a \\ b \\ \hline c \\ b \\ c \\ d \\ a \end{pmatrix} \begin{matrix} \} \text{ Process 0} \\ \\ \\ \\ \\ \} \text{ Process 1} \\ \\ \\ \end{matrix}$$

Figure 2.12: Matrix-vector notation of the ghost cell send operation for a one-dimensional block decomposition of $M = 4$ data points on $P = 2$ processes with default block size $B = 2$ and one ghost cell at each side. Thereby, matrix and vector entries corresponding to ghost cell values are highlighted in gray.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ \hline e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} b+h \\ c+e \\ \hline f+d \\ g+a \end{pmatrix} \begin{matrix} \} \text{ Process 0} \\ \\ \\ \} \text{ Process 1} \end{matrix}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{Process 0} \quad \text{Process 1}}$$

Figure 2.13: The transposed ghost cell send for a one-dimensional block decomposition of $M = 4$ data points on $P = 2$ processes with default block size $B = 2$ and one ghost cell at each side. Thereby, matrix and vector entries corresponding to ghost cell values are highlighted in gray.

ghost cells exceeds the block size of the next neighboring process. This is especially important for unequal block sizes, where some processes get less data then others.

2.9 The PFFT software library

We implemented the parallel multidimensional FFT Frameworks 2.5 and 2.6 as part of a publicly available software library called Parallel Fast Fourier Transform (PFFT). The development code is distributed under a GNU General Public License and is freely available within the Git repository [11]. Parallelism has been incorporated by the Message Passing Interface (MPI) standard. PFFT was designed as an extension of FFTW to multidimensional data decompositions. Therefore, the user interface is very similar to the FFTW-MPI interface and PFFT inherits most of FFTWs features. In the following, we list the most important features of our parallel FFT implementation in order to give an impression of its flexibility:

- We employ fast $\mathcal{O}(M \log M)$ algorithms of FFTW to compute arbitrary-size discrete Fourier transforms of complex data, real data, and even- or odd-symmetric real data. This also includes prime size FFTs.
- The dimension of the FFT can be arbitrary. Note that other publicly available, parallel FFT implementations based on two-dimensional data decomposition [123, 106, 90, 43] are restricted to three-dimensional FFTs.
- PFFT supports r-dimensional data decomposition of d-dimensional FFTs for any $r < d$. Especially, PFFT includes three-dimensional data decomposition of four-dimensional FFTs. We are not aware of any other publicly available parallel FFT implementation based on three- or higher-dimensional data decomposition.
- The application of PFFT is split into a time consuming planning step and a high performance execution step. Thereby, the amount of planning can be adjusted via flags.
- Because of the hardware adaptive planning phase PFFT offers portable performance; i.e., it will perform well on most platforms. This also includes the MPI based communication of parallel FFT – a feature not available in any other parallel FFT implementations based on two-dimensional data decomposition [123, 106, 90, 43].
- Instal of the library is easy. It is based on the Autotools and follows the common sequence of configure, make, and make install.
- The interface of PFFT is very close to the MPI interface of FFTW. In fact, we tried to add as few extra parameters as possible. Porting code from FFTW-MPI to PFFT is straightforward, i.e., there is no need for complicated code refactoring in order to switch from one-dimensional to multidimensional data decomposition.
- PFFT is written in C but also offers a Fortran 2003 interface.

- Since FFTW offers shared memory parallelism, we can run PFFT in a hybrid parallel environment as a mix of distributed and shared memory parallelism.
- All steps of our parallel FFT can be performed completely in-place. This is especially remarkable for the global transposition routines. Note that all other parallel FFT implementation based on multidimensional data decomposition need at least twice the memory amount of the input array.
- The block distribution of PFFT does not require the FFT size to be divisible by the number of processes. However, unequal block distributions naturally lead to load balancing problems that should be avoided if possible.
- PFFT includes a very flexible ghost cell exchange module, cf. Section 2.8. It includes ghost cell operations that exceed nearest neighbor communication and supports unequal block decompositions. Moreover, the transposed ghost cell send module turns out to be a unique feature of PFFT.
- PFFT allows three-dimensional data decomposition even for three-dimensional FFTs. In this case the three-dimensional data decomposition is redistributed into a two-dimensional data decomposition and the underlying parallel FFT framework is still based on two-dimensional decomposition. A more detailed description of this module was published in [4].
- PFFT has been the first publicly available library that introduced support for the parallel calculation of pruned FFTs at the library level; see Section 2.6. Later on, this feature has been added up to some extend to the P3DFFT library [106], i.e., P3DFFT allows pruning in the inputs of the forward transforms and the outputs of the backward transforms only. Moreover, pruned FFTs are not supported by any other freely available parallel FFT implementation. In Section 2.10.4 we present some performance results of PFFTs pruned FFTs.
- PFFT supports FFTs with shifted index sets, cf. Section 2.7. We are not aware of any other parallel FFT software library that supports this feature at the moment.
- All performance relevant computation and communication steps are automatically timed throughout the execution of PFFT. The user can access these runtime measurements through the PFFT timer interface. This is especially helpful for benchmark and runtime tuning purposes. If timing of each PFFT step is considered to be waste of performance, the user has the possibility to switch off the PFFT timer interface during configuration of the library instal.
- PFFT supports parallel FFT with transposed input and output, cf. Section 2.5. This saves half of the communication amount in comparison to a parallel FFT that restores the initial data ordering after each FFT.

2.10 Numerical results

In [4] we published extensive numerical tests that give numerical evidence for the high scalability of our parallel FFT frameworks. We compared our PFFT software library with the MPI-parallel algorithms of the FFTW software library as well as P3DFFT [107, 106]. In the following, we present selected results from [8] and [4]. We refer the reader to these references for more elaborate performance tests.

2.10.1 Description of the parallel computing architectures

The following three large scale, high performance platforms have been taken into consideration.

1. *JUGENE – Blue Gene/P in Research Center Jülich [14]*: One node of a Blue Gene/P consists of 4 IBM PowerPC 450 cores that run at 850 MHz. These 4 cores share 2 GB of main memory. Therefore, we have 0.5 GB RAM per core, whenever all the cores per node are used. The nodes are connected by a three-dimensional torus network with 425 MB/s bandwidth per link. In total JUGENE consisted of 73 728 nodes, i.e., 294 912 cores. This system went out of production in August, 2012.
2. *JUQUEEN – Blue Gene/Q in Research Center Jülich [15]*: One node of a Blue Gene/Q consists of 16 IBM PowerPC A2 cores that run at 1.6 GHz. These 16 cores share 16 GB SDRAM-DDR3. Therefore, we have 1 GB RAM per core, whenever all the cores per node are used. The nodes are connected by a five-dimensional torus network. In total JUGENE consists of 24 576 nodes, i.e., 393 216 cores.
3. *JUROPA – Jülich Research on Petaflop Architectures [16]*: One node of JUROPA consists of 2 Intel Xeon X5570 (Nehalem-EP) quad-core processors that run at 2.93 GHz. These 8 cores share 24 GB DDR3 main memory. Therefore, we have 3 GB RAM per core whenever all the cores per node are used. The nodes are connected by an Infiniband QDR with non-blocking fat tree topology. In total JUROPA consists of 2208 nodes, i.e., 17 664 cores.

2.10.2 Strong scaling behavior of PFFT on JUGENE

We investigated the strong scaling behavior of PFFT [11] and P3DFFT [106] on the JUGENE machine in Research Center Jülich. A complex to complex FFT of size 512^3 has been run out-of-place with 64 of the available 72 racks, i.e., 262 144 cores. Since P3DFFT supports only real to complex FFTs, we applied P3DFFT to the real and imaginary parts of a complex input array to get times comparable to those of the complex to complex FFTs of the PFFT package. The test runs consisted of 10 alternate calculations of FFT and $\mathrm{FFT}^{\mathsf{H}}$. Since these two transforms are inverse

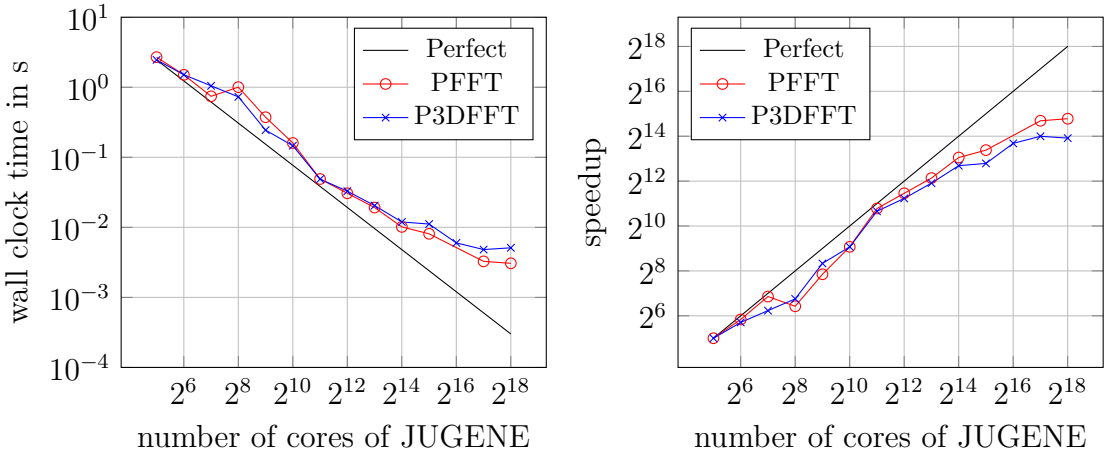

Figure 2.14: Required wall clock time (left) and speedup (right) for FFT of size 512^3 up to 262 144 cores on JUGENE.

except for a constant factor, it is easy to check the results after each run. The average wall clock time and the average speedup of one FFT and FFT^{H} can be seen in Figure 2.14 for an FFT of size 512^3. Memory restrictions force P3DFFT to utilize at least 32 cores on JUGENE. Therefore, we chose the associated wall clock times as references for speedup and efficiency calculations. Note that PFFT can perform these FFTs on half the cores because of less memory consumption. However, we only recorded times on core counts which both algorithms were able to utilize to get comparable results. It turns out that both libraries are comparable in speed. However, from our point of view the flexibility of PFFT is a great advantage over P3DFFT.

2.10.3 Comparison of PFFT and FFTW on JUROPA

We tested our PFFT library on JUROPA and compared the scaling behavior with a one-dimensional decomposed parallel FFTW. The runtime of a three-dimensional FFT of size 256^3 given in Figure 2.15 shows a good scaling behavior of our two-dimensional decomposed PFFT up to 2048 cores, while the one-dimensional data decomposition of FFTW cannot make use of more than 256 cores.

2.10.4 Parallel pruned FFT on JUGENE

We have seen in Section 2.6 that the naive computation of parallel pruned FFTs via zero padded FFTs may lead to serious load imbalance since some processes calculate one-dimensional FFTs on vectors that are full of zeros. In addition, the data has to be redistributed before and after the FFT in order to match the parallel domain decomposition. Our parallel FFT Frameworks 2.5 and 2.6 completely avoid the data

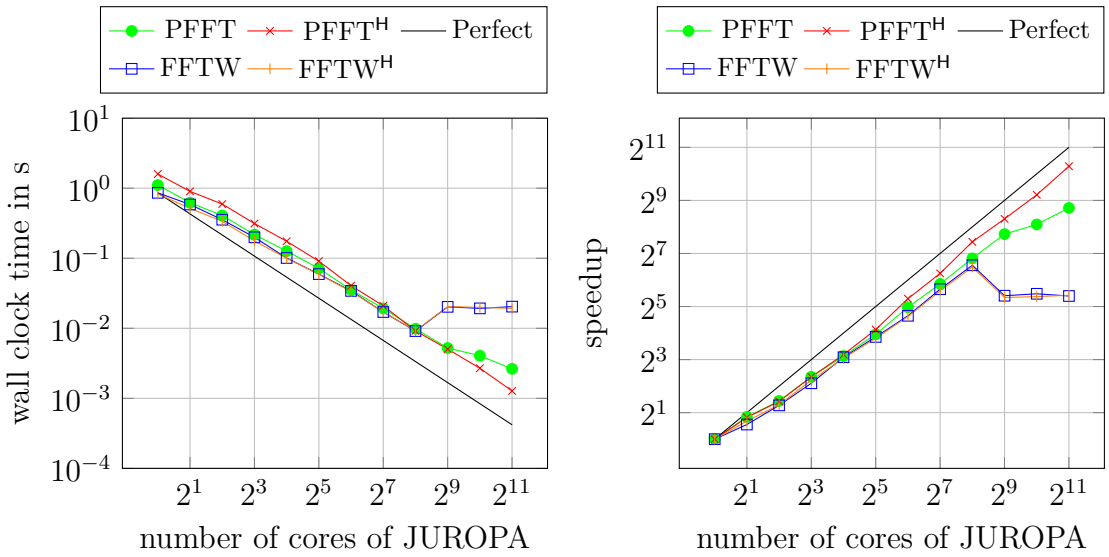

Figure 2.15: Required wall clock time (left) and speedup (right) for FFT of size 256^3 up to 2048 cores on JUROPA.

redistribution and the load imbalances, since they apply one-dimensional pruned FFTs row-wise whenever the corresponding data dimension is locally available on the processes.

We want to illustrate the possible performance gain with an example. Therefore, we compute a three-dimensional pruned FFT of size $M \times M \times M = 256^3$ on 256 cores of a Blue Gene/P architecture. The data decomposition scheme is based on a two-dimensional process mesh of size 16×16. We alter the pruned input size $m \times m \times m$ and the pruned output size $\hat{m} \times \hat{m} \times \hat{m}$ between 32 and 256. Figure 2.16 shows the runtime of pruned PFFT for different values of m and $\hat{m}$. We observe an increasing performance benefit for decreasing input array size m and also for decreasing output array size $\hat{m}$. Without the pruned FFT support, we would have to pad the input array of size $m \times m \times m$ with zeros to the full three-dimensional FFT size $M \times M \times M$ and calculate this FFT in parallel. The time for computing an FFT of size 256^3 corresponds to the time in Figure 2.16 for $m = \hat{m} = 256$.

2.10.5 Weak scaling behavior of PFFT on JUQUEEN and JUROPA

In order to investigate the weak scaling behavior on a Blue Gene/Q we performed parallel FFTs of size 512^3, 1024^3, 2048^3, 4096^3, and 8192^3 on 8, 64, 512, 4096, and 32 768 cores, respectively. This gives a constant local array size of 256^3 per process. We measured the average time over 10 loops of FFT and FFT^{H} with transposed input/output on each process and plotted the maximum over all processes in Fig-

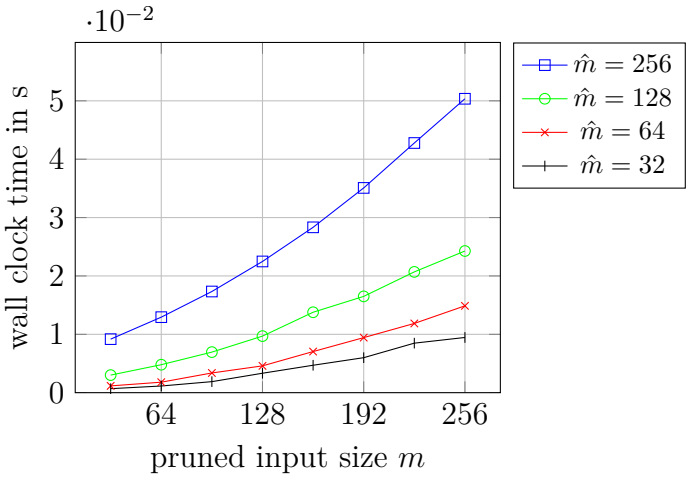

Figure 2.16: Required wall clock time of pruned FFT with FFT size $M^3 = 256^3$ and varying FFT input size m^3 and FFT output size $\hat{m}^3$. All test have been computed with $P = 16^2$ cores on JUGENE.

ure 2.17. We used exactly the same setting on JUROPA but stopped at 4096 cores. The results are also given in Figure 2.17. In addition, we show the time that is spent for communication and computation. Note that the computational part also includes the local transposes of our serial FFT module.

2.10.6 Strong scaling behavior of PFFT on JUQUEEN and JUROPA

Finally, we compare the strong scaling behavior of our parallel in-place and out-of-place FFTs for different FFT sizes on JUQUEEN and JUROPA. Again, we performed 10 loops of a FFT and FFT^{H} with transposed input/output. The maximum average time for FFTs of size 512^3 and 1024^3 with up to 32 768 cores on JUQUEEN are given in Figure 2.18, and Figure 2.19, respectively. In addition, we show the time that is spent for communication and computation. Note that the computational part also includes the local transposes of our serial FFT module. For every test run we chose the minimal possible core count to start the benchmark. We observe that the in-place transforms are indeed more memory efficient, since they allow us to run the benchmarks with smaller core counts. The out-of-place transforms are slightly faster for large core counts. However, the in-place transforms are most important for small numbers of cores, where less memory is available. There is no difference in the performance of in-place and out-of-place FFTs for small core counts. Our parallel FFT framework provides an overall good scaling behavior. For large numbers of cores we observe some jumps of the runtime due to the communication part.

The maximum average time of 10 FFT and FFT^{H} of size 512^3, and 1024^3 with up

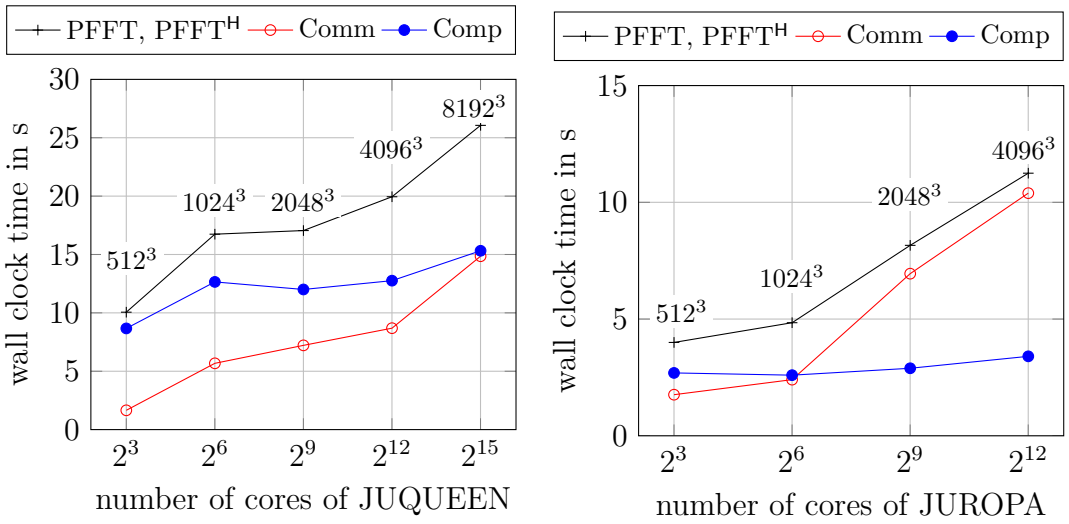

Figure 2.17: Required wall clock time for FFT of constant local array size 256^3 per core up to $P = 32\,768$ cores on JUQUEEN (left) and up to $P = 2048$ cores on JUROPA (right). The figures include the whole runtime of one pair of PFFT and PFFT$^{\mathsf{H}}$ that splits into the time spent for communication (Comm) and computation (Comp). The numbers next to data points indicate the total FFT size.

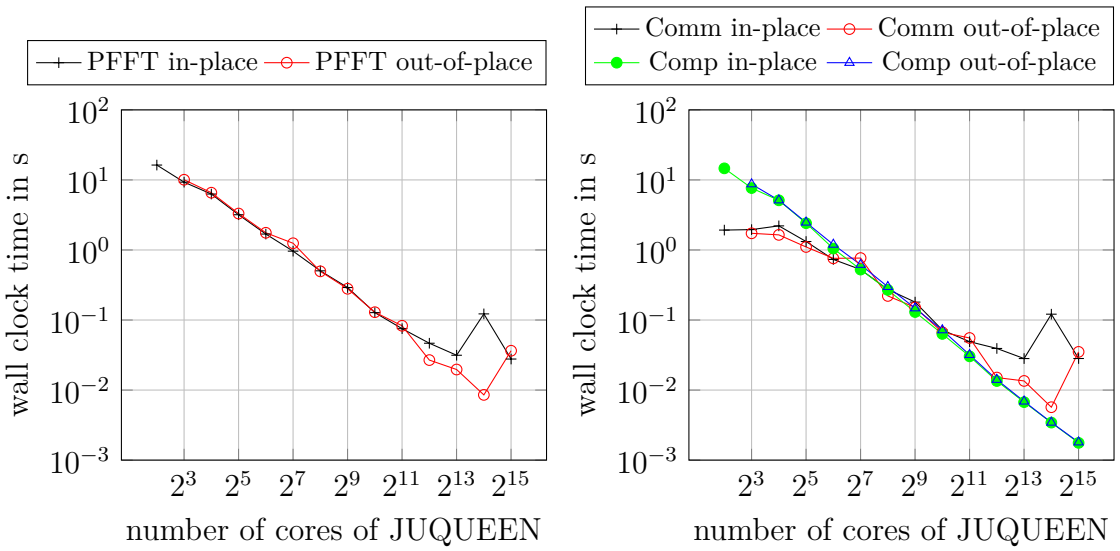

Figure 2.18: Required wall clock time for in-place and out-of-place FFT of size 512^3 up to $P = 32\,768$ cores on JUQUEEN. The figure includes the whole runtime of one PFFT and its adjoint. This time splits into the time spent for communication (Comm) and computation (Comp).

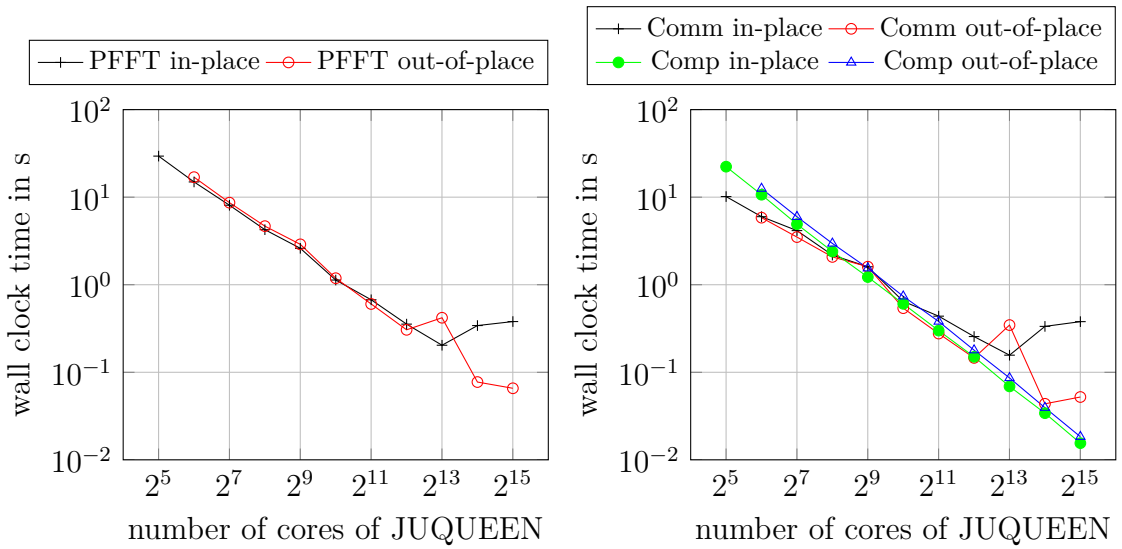

Figure 2.19: Required wall clock time for in-place and out-of-place FFT of size 1024^3 up to $P = 32\,768$ cores on JUQUEEN. The figure includes the whole runtime of one PFFT and its adjoint. This time splits into the time spent for communication (Comm) and computation (Comp).

to 2048 cores on JUROPA are given in Figure 2.20, and Figure 2.21, respectively. Here we see nearly the same behavior. There is even less difference in the performance of in-place and out-of-place FFTs on JUROPA. The big jump in Figure 2.20 results from the fact that an in-place transposition with one single core can be totally omitted, while the out-of-place transposition needs at least one copy of the local memory.

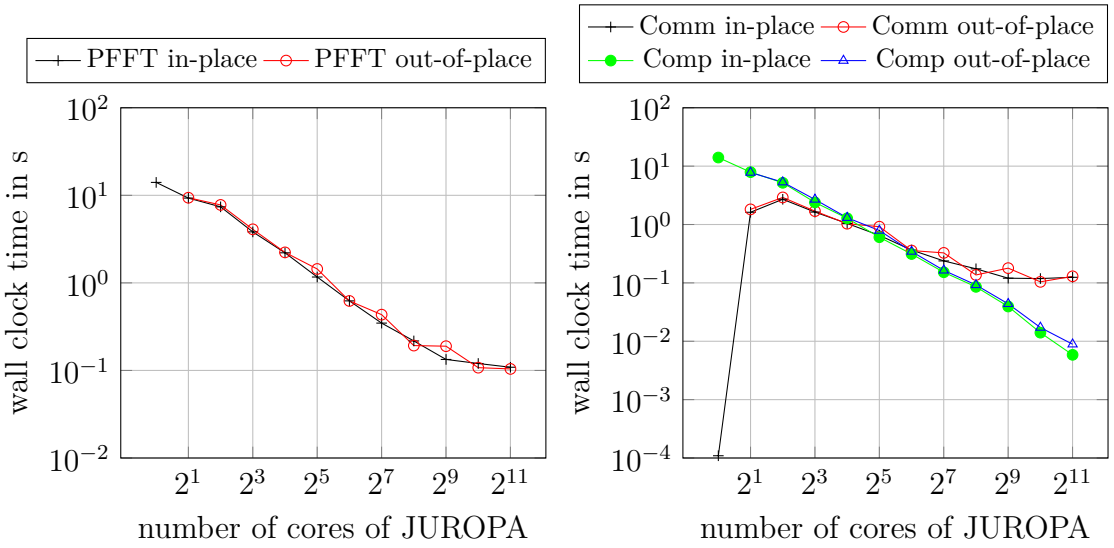

Figure 2.20: Required wall clock time for in-place and out-of-place FFT of size 512^3 up to $P = 2048$ cores on JUROPA. The figure includes the whole runtime of one PFFT and its adjoint. This time splits into the time spent for communication (Comm) and computation (Comp).

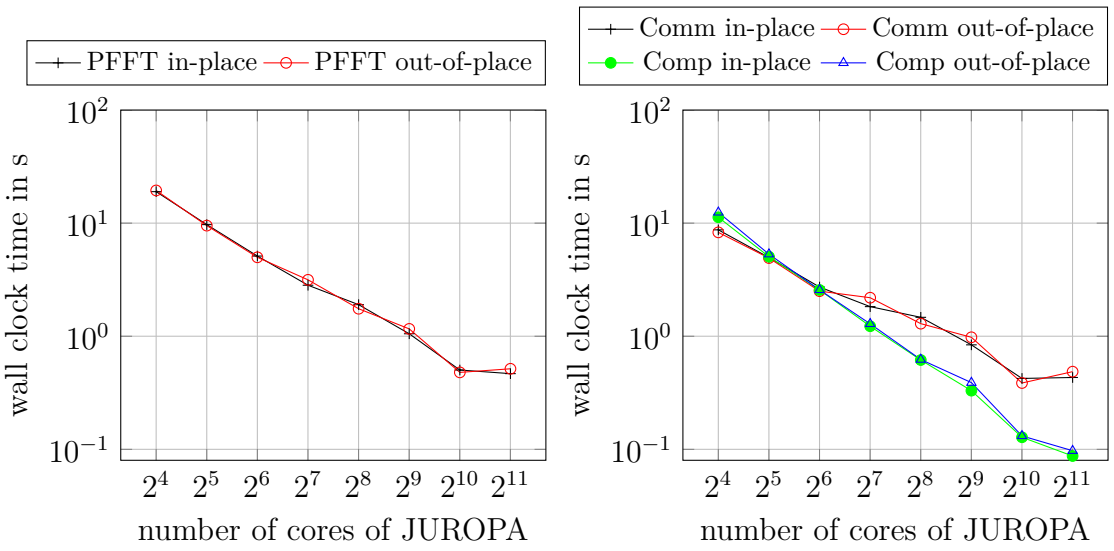

Figure 2.21: Required wall clock time for in-place and out-of-place FFT of size 1024^3 up to $P = 2048$ cores on JUROPA. The figure includes the whole runtime of one PFFT and its adjoint. This time splits into the time spent for communication (Comm) and computation (Comp).

3 Parallel nonequispaced fast Fourier transforms

The equispaced sampling of a trigonometric polynomial at N points can be performed in a computational efficient way by the fast Fourier transform (FFT) within $\mathcal{O}(N \log N)$ operations. However, many applications such as computerized tomography [53, 45] and particle simulation [112, 68] require a nonequispaced sampling and, therefore, the FFT is not applicable anymore. This shortcoming led to the development of several generalizations to nonequispaced sampling [42, 26, 121, 130, 114, 61, 84]. In the following, we denote this class of algorithms as Nonequispaced Fast Fourier Transform (NFFT) and focus on the notation and modularized approach used in [121, 114, 84]. Loosely speaking, the NFFT consists of the following three steps. First, it applies a deconvolution in Fourier space that can be performed by a simple point-wise product. Second, an FFT is computed and, finally, a discrete convolution in spatial domain is performed in the form of a sparse matrix-vector product. The deconvolution in Fourier space and the discrete convolution in real space are performed with a window function that is well localized in both domains. Therefore, the overall complexity of the NFFT with N nonequispaced samples and the total number of frequencies M is given by $\mathcal{O}(N + M \log M)$. This is a tremendous improvement in comparison to the $\mathcal{O}(NM)$ scaling of the naive computation. In this sense the NFFT is a fast algorithm with a modularized structure. A publicly available implementation is given by the NFFT software library [84, 83], which also offers support of shared memory parallelism [128] and a parallel implementation for graphic processing units [87, 134]. In contrast, distributed memory parallelism of NFFT has not been considered for a long time and to our knowledge the PNFFT library presented in this chapter is the only publicly available NFFT implementation that combines the fast approximate NFFT algorithm with high scalability on distributed memory architectures. We emphasize that a highly scalable butterfly algorithm for the computation of Fourier integral operators has been presented in [115]. The Fourier integral operator contains the NFFT as a special case. However, the runtime of this algorithm is rather high due to its generality and can not compete with the NFFT algorithms presented in this thesis.

In this chapter we derive a highly scalable framework for computing the NFFT on massively parallel architectures called PNFFT [5]. Since the FFT plays a central role in the modular structure of NFFT, we will exploit the parallel FFT frameworks that were presented in the previous chapter.

Furthermore, we derive various new advancements of the NFFT algorithm itself. More precisely, we introduce the concepts of pruned NFFT, fast NFFT gradient computation, shifted NFFT, interlaced NFFT and optimized NFFT deconvolution with respect to the mean square aliasing error. Most of these concepts have their origin in particle-mesh methods but have never been considered as individual modules of the NFFT. Altogether, these algorithmic enhancements will bridge the gap between NFFT and particle-mesh methods in Chapter 4.

3.1 Definitions

In this section we introduce the new notation, basic definitions and assumptions that are related to nonequispaced Fourier transforms. Assume a mesh size $\hat{\boldsymbol{m}} = (\hat{m}_0, \hat{m}_1, \hat{m}_2)^{\mathsf{T}} \in 2\mathbb{N}^3$. We define the shifted multi-index set of frequencies $\mathcal{I}_{\hat{\boldsymbol{m}}} := \{-\hat{m}_0/2, \dots, \hat{m}_0/2 - 1\} \times \{-\hat{m}_1/2, \dots, \hat{m}_1/2 - 1\} \times \{-\hat{m}_2/2, \dots, \hat{m}_2/2 - 1\}$, and the total number of frequencies $|\mathcal{I}_{\hat{\boldsymbol{m}}}| = \hat{m}_0 \cdot \hat{m}_1 \cdot \hat{m}_2$. For $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ complex numbers $\hat{f}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$, the trigonometric polynomial $f\colon \mathbb{R}^3 \to \mathbb{C}$ is given by

$$f(\boldsymbol{x}) := \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}}. \tag{3.1}$$

Note that this function is 1-periodic in each dimension of space. Furthermore, we introduce the node scaling matrix $\mathbf{S} = \operatorname{diag}(S_0, S_1, S_2)$ with entries $S_t \in (0, 1]$ and define the truncated unit cube $\mathbf{S}\big[-1/2, 1/2\big)^3 := \big[-S_0/2, S_0/2\big) \times \big[-S_1/2, S_1/2\big) \times \big[-S_2/2, S_2/2\big)$. The evaluation of f at $N \in \mathbb{N}$ arbitrarily chosen nodes $\boldsymbol{x}_j \in \mathbf{S}\big[-1/2, 1/2\big)^3$, $j \in \mathcal{N} := \{1, 2, \dots, N\}$ can be written as

$$f_j := f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}_j}, \quad \boldsymbol{x}_j \in \mathbf{S}\left[-\tfrac{1}{2}, \tfrac{1}{2}\right)^3, \quad j \in \mathcal{N}, \tag{3.2}$$

and is denoted as three-dimensional, pruned Nonequispaced Discrete Fourier Transform (NDFT). Note that most authors define the NDFT without pruning, i.e., they use a fixed node scaling matrix $\mathbf{S} = \operatorname{diag}(1, 1, 1)$ and allow the nodes to fill the whole unit cube $\boldsymbol{x}_j \in \big[-1/2, 1/2\big)^3$. The most important difference is that for $\mathbf{S} \neq \operatorname{diag}(1, 1, 1)$ the box lengths of $\mathbf{S}\big[-1/2, 1/2\big)^3$ do not match the period lengths of the trigonometric polynomial (3.1). We will see that this fact offers some opportunities for adapted algorithm design and requires special attention when it comes to parallelization. For example, the pruned NDFT naturally appears when we want to evaluate Fourier approximations of non-periodic functions, cf. Section 4.4.7.

The NDFT (3.2) can be written as a matrix-vector product

$$\boldsymbol{f} = \mathbf{A}_{N,\hat{\boldsymbol{m}}} \hat{\boldsymbol{f}},$$

with the vectors $\boldsymbol{f} := (f_j)_{j\in\mathcal{N}} \in \mathbb{C}^N$, $\hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \in \mathbb{C}^{|\mathcal{I}_{\hat{\boldsymbol{m}}}|}$, and the nonequispaced Fourier matrix $\mathbf{A}_{N,\hat{\boldsymbol{m}}} := (\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j})_{j\in\mathcal{N};\,\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \in \mathbb{C}^{N\times|\mathcal{I}_{\hat{\boldsymbol{m}}}|}$. For the sake of simplicity, we write the multi-index $\boldsymbol{k}$ in order to address elements of vectors and matrices at the linearized index $k_2 + \hat{m}_2 k_1 + \hat{m}_2\hat{m}_1 k_0$, i.e., the multi-indexes $\boldsymbol{k}$ are ordered in row-major format. In general, the matrix $\mathbf{A}_{N,\hat{\boldsymbol{m}}}$ is not square. Even for the square case, it is usually not orthogonal and $|\mathcal{I}_{\hat{\boldsymbol{m}}}|^{-1}\mathbf{A}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}}\mathbf{A}_{N,\hat{\boldsymbol{m}}}$ does not yields the identity matrix in contrast to the equispaced case. Therefore, the definition of an inverse NFFT is not canonical, but can be realized, e.g., by an iterative conjugate gradient method; cf. [88]. Nevertheless, it is customary to define the Adjoint Nonequispaced Discrete Fourier Transform ($\mathrm{NDFT}^{\mathsf{H}}$) by the matrix-vector product

$$\hat{\boldsymbol{h}} = \mathbf{A}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}}\boldsymbol{f},$$

that is equivalent to the sums

$$\hat{h}_{\boldsymbol{k}} = \sum_{j=1}^{N} f_j \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j}, \qquad \boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}, \tag{3.3}$$

with the vector $\hat{\boldsymbol{h}} := (\hat{h}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}}$. Furthermore, applications of the NDFT in particle simulation motivate the calculation of the gradients

$$\nabla f_j := \nabla f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j} = -2\pi\mathrm{i}\sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}}\boldsymbol{k}\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j}, \quad j\in\mathcal{N}. \tag{3.4}$$

This can be written as the matrix-vector product

$$\nabla\boldsymbol{f} = \nabla\mathbf{A}_{N,\hat{\boldsymbol{m}}}\hat{\boldsymbol{f}},$$

with the formal definition of the vector $\nabla\boldsymbol{f} := (\nabla f_j)_{j\in\mathcal{N}} \in \mathbb{C}^{3N}$, and the matrix $\nabla\mathbf{A}_{3N,\hat{\boldsymbol{m}}} := \left(\nabla\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j}\right)_{j\in\mathcal{N};\,\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \in \mathbb{C}^{3N\times|\mathcal{I}_{\hat{\boldsymbol{m}}}|}$. Hereby, each gradient spans over three successive rows of the vector $\nabla\boldsymbol{f}$ and the matrix $\nabla\mathbf{A}_{3N,\hat{\boldsymbol{m}}}$, respectively.

In the following, we give a brief overview of fast approximate algorithms for computing the NDFT (3.2), its adjoint (3.3), and the gradient NDFT (3.4). We denote the class of fast algorithms for computing the NDFT as Nonequispaced Fast Fourier Transforms (NFFT). Analogously, fast algorithms for computing the $\mathrm{NDFT}^{\mathsf{H}}$ will be called Adjoint Nonequispaced Fast Fourier Transform ($\mathrm{NFFT}^{\mathsf{H}}$).

3.2 The three-dimensional NFFT algorithm

In this section we give a brief overview of mathematical theory and ideas behind the NFFT based on [121, 114, 84]. This will also be the starting point of our parallel

algorithms in Section 3.3. For alternative NFFT approaches we refer to the nice overview in [84, Appendix D].

Assume an FFT mesh size $\boldsymbol{M} = (M_0, M_1, M_2)^{\mathsf{T}} \in 2\mathbb{N}^3$ that fulfills the componentwise inequality $\boldsymbol{M} \geq \hat{\boldsymbol{m}}$ in comparison to the NDFT mesh size $\hat{\boldsymbol{m}}$ in (3.2). We define the shifted multi-index set of possible frequencies as $\mathcal{I}_{\boldsymbol{M}} := \{-M_0/2, \ldots, M_0/2 - 1\} \times \{-M_1/2, \ldots, M_1/2 - 1\} \times \{-M_2/2, \ldots, M_2/2 - 1\}$ and the total number of frequencies is given by $|\mathcal{I}_{\boldsymbol{M}}| = M_0 \cdot M_1 \cdot M_2$. Let $\psi\colon \mathbb{R} \to \mathbb{R}$ be a smooth window function, i.e., a function that is well localized in spatial domain and in frequency domain. Moreover, its continuous Fourier transform $\hat{\psi}(v) := \int_{\mathbb{R}} \psi(x) \mathrm{e}^{+2\pi \mathrm{i} v x} \mathrm{d}x$, $v \in \mathbb{R}$ may exist and its 1-periodic version $\tilde{\psi}\colon \mathbb{R} \to \mathbb{R}$, $\tilde{\psi}(x) := \sum_{r \in \mathbb{Z}} \psi(x + r)$ may have a uniformly convergent Fourier series. Then, we define a multivariate window function $\varphi\colon \mathbb{R}^3 \to \mathbb{R}$ by the tensor product $\varphi(\boldsymbol{x}) := \tilde{\psi}(x_0)\, \tilde{\psi}(x_1)\, \tilde{\psi}(x_2)$. Therefore, the Fourier coefficients of φ are given by

$$\hat{\varphi}_{\boldsymbol{k}} := \int_{\left[-1/2, 1/2\right)^3} \varphi(\boldsymbol{x}) \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}} \mathrm{d}\boldsymbol{x} = \hat{\psi}(k_0) \hat{\psi}(k_1) \hat{\psi}(k_2), \quad \boldsymbol{k} = (k_0, k_1, k_2)^{\mathsf{T}} \in \mathbb{Z}^3,$$

and the gradient $\nabla\varphi$ can be easily computed by

$$\nabla\varphi(x_0, x_1, x_2) = \left(\tilde{\psi}'(x_0)\tilde{\psi}(x_1)\tilde{\psi}(x_2),\, \tilde{\psi}(x_0)\tilde{\psi}'(x_1)\tilde{\psi}(x_2),\, \tilde{\psi}(x_0)\tilde{\psi}(x_1)\tilde{\psi}'(x_2)\right)^{\mathsf{T}}. \tag{3.5}$$

A selection of possible window functions will be given later on in Section 3.2.1.

We follow the general approach of [121, 114] and approximate the trigonometric polynomial f by a linear combination of translated window functions

$$f(\boldsymbol{x}) \approx \breve{s}(\boldsymbol{x}) := \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} g_{\boldsymbol{l}} \varphi(\boldsymbol{x} - \boldsymbol{l} \odot \boldsymbol{M}^{-1}). \tag{3.6}$$

Hereby, we used the reciprocal $\boldsymbol{M}^{-1} := \left(M_0^{-1}, M_1^{-1}, M_2^{-1}\right)^{\mathsf{T}}$ of a vector $\boldsymbol{M}$ with nonzero components and $\boldsymbol{l} \odot \boldsymbol{M}^{-1} := \left(l_0 M_0^{-1}, l_1 M_1^{-1}, l_2 M_2^{-1}\right)^{\mathsf{T}}$ denotes a componentwise vector product. A straightforward Fourier series expansion yields

$$\breve{s}(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}}} \hat{g}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}} + \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}}} \sum_{\boldsymbol{r} \in \mathbb{Z}^3 \setminus \{\boldsymbol{0}\}} \hat{g}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M}} \mathrm{e}^{-2\pi \mathrm{i} (\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M})^{\mathsf{T}} \boldsymbol{x}}, \tag{3.7}$$

with the $\boldsymbol{M}$-periodic discrete Fourier coefficients

$$\hat{g}_{\boldsymbol{k}} = \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} g_{\boldsymbol{l}} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}, \quad \boldsymbol{k} \in \mathbb{Z}^3. \tag{3.8}$$

At a first glance, a comparison of coefficients in (3.1) and (3.7) suggests to set $\hat{g}_{\boldsymbol{k}}$ in compliance to [121, 114, 84] as

$$\hat{g}_{\boldsymbol{k}} := \begin{cases} \hat{f}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}}^{-1} & \text{for } \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}, \\ 0 & \text{for } \boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}} \setminus \mathcal{I}_{\hat{\boldsymbol{m}}}. \end{cases} \tag{3.9}$$

Thereby, we assume a fast decay of $\hat{\varphi}_{\boldsymbol{k}}$ such that the terms for $\boldsymbol{k} \notin \mathcal{I}_{\boldsymbol{M}}$ in (3.7) are negligible. However, this is not the only possibility for setting $\hat{g}_{\boldsymbol{k}}$. In Section 3.2.4 we will return to the question of how to choose $\hat{g}_{\boldsymbol{k}}$ in an optimal way. For the moment we remain with this simple scheme and define the first step of our NFFT algorithm, namely the deconvolution in Fourier space

$$\hat{g}_{\boldsymbol{k}} = \hat{f}_{\boldsymbol{k}} \hat{d}_{\boldsymbol{k}}, \quad \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}, \quad \text{with} \quad \hat{d}_{\boldsymbol{k}} := \hat{\varphi}_{\boldsymbol{k}}^{-1} \tag{3.10}$$

and we implicitly assume $\hat{g}_{\boldsymbol{k}} = 0$, for $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}} \setminus \mathcal{I}_{\hat{\boldsymbol{m}}}$. Obviously, this step consists of $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ multiplications.

Once the coefficients $\hat{g}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$, are given, we can recover the coefficients $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}$, from (3.8) by a three-dimensional pruned FFT with shifted index sets

$$g_{\boldsymbol{l}} = \frac{1}{|\mathcal{I}_{\boldsymbol{M}}|} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{g}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \left(\boldsymbol{l} \odot \boldsymbol{M}^{-1} \right)}, \quad \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}, \tag{3.11}$$

cf. Algorithm 2.3. Thereby, the pruned FFT input size is given by $\hat{\boldsymbol{m}}$ and the FFT mesh size is equal to $\boldsymbol{M}$. This will be the second step of our NFFT algorithm. It requires $\mathcal{O}(|\mathcal{I}_{\boldsymbol{M}}| \log |\mathcal{I}_{\boldsymbol{M}}|)$ arithmetic operations.

Finally, we want to truncate the sums in (3.6) due to the fast decay of the window function φ in real space. Therefore, we introduce the multi-index set

$$\mathcal{I}_{\boldsymbol{M}, c_\varphi}(\boldsymbol{x}) := \left\{ \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}} \colon \exists \boldsymbol{z} \in \mathbb{Z}^3 \text{ with } -c_\varphi \boldsymbol{1} \leq \boldsymbol{l} + \boldsymbol{z} - \boldsymbol{M} \odot \boldsymbol{x} \leq c_\varphi \boldsymbol{1} \right\}$$

that collects all the indexes where the $\boldsymbol{1}$-periodic window function $\varphi(\cdot - \boldsymbol{M} \odot \boldsymbol{x})$ is mostly concentrated. Note that we abbreviate the vector $\boldsymbol{1} := (1,1,1)^{\mathsf{T}}$, all inequalities between vectors are understood component-wise, and $c_\varphi \in \mathbb{N}$ is a small window cutoff parameter, which depends on the particular choice of the window function. Now, we can write an approximation of (3.6) in terms of the discrete convolution sum

$$f(\boldsymbol{x}_j) \approx s_j := \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}, c_\varphi}(\boldsymbol{x}_j)} g_{\boldsymbol{l}} \, \varphi\left(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1} \right), \quad j \in \mathcal{N}. \tag{3.12}$$

Each sum in (3.12) has at most $(2c_\varphi + 1)^3$ terms since the window function φ is sampled only in the neighborhood of each node $\boldsymbol{x}_j$. The discrete convolution step (3.12) represents the last step of our NFFT algorithm and requires $\mathcal{O}((2c_\varphi + 1)^3 N)$ arithmetic operations. In summary, the three NFFT approximation steps (3.10), (3.11) and (3.12) can be written altogether as

$$f(\boldsymbol{x}) \approx \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}, c_\varphi}(\boldsymbol{x}_j)} \left(\sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \frac{\hat{f}_{\boldsymbol{k}} \hat{d}_{\boldsymbol{k}}}{|\mathcal{I}_{\boldsymbol{M}}|} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \left(\boldsymbol{l} \odot \boldsymbol{M}^{-1} \right)} \right) \varphi\left(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1} \right). \tag{3.13}$$

We point out that the approximation error introduced by the NFFT decomposes into an aliasing error due to (3.10) and a truncation error due to (3.12); see [114, 84] for details. Most important, for appropriate choice of window functions φ the approximation error decays exponentially which makes the NFFT a computational efficient algorithm. Error estimates for the multivariate NFFT were presented in [46]; see also [84, Appendix C].

Remark 3.1. An alternative viewpoint on the derivation of the NFFT is used in the context of particle-mesh Ewald methods [37, 48]. Thereby, the authors start with an approximation of the complex exponential in the form

$$\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}} \approx \frac{1}{|\mathcal{I}_{\boldsymbol{M}}|\hat{\varphi}_{\boldsymbol{k}}} \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x})} \varphi\left(\boldsymbol{x}-\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right)\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\left(\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right)}. \tag{3.14}$$

Plugging (3.14) into the trigonometric polynomial (3.1) directly leads to (3.13). □

There are also several fast ways for computing the gradients (3.4). Here we concentrate on the two most common ones. The first approach is to apply one NFFT for each component of the vector sum (3.4). This is also known as i$\boldsymbol{k}$-derivative in the community of particle-mesh algorithms, cf. [70, 38]. The resulting approximation is denoted as gradient NFFT with derivative in Fourier space (i$\boldsymbol{k}$-NFFT) and reads as

$$\nabla f(\boldsymbol{x}) \approx -2\pi\mathrm{i} \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{m},c_\varphi}(\boldsymbol{x}_j)} \left(\sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \boldsymbol{k}\frac{\hat{f}_{\boldsymbol{k}}\hat{d}_{\boldsymbol{k}}}{|\mathcal{I}_{\boldsymbol{M}}|}\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\left(\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right)} \right) \varphi\left(\boldsymbol{x}_j-\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right). \tag{3.15}$$

Note that this approach triples the number of Fourier transforms and discrete convolutions in comparison to the computation of the function values by (3.13).

Alternatively, one can substitute the discrete convolution step (3.12) of the NFFT by

$$\nabla f(\boldsymbol{x}_j) \approx \nabla s_j := \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}\,\nabla\varphi\left(\boldsymbol{x}_j-\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right), \quad j\in\mathcal{N}. \tag{3.16}$$

This approximation is also known as analytic differentiation [48]. Therefore, we denote the resulting approximation of the gradient NDFT

$$\nabla f(\boldsymbol{x}) \approx \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{m},c_\varphi}(\boldsymbol{x}_j)} \left(\sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \frac{\hat{f}_{\boldsymbol{k}}\hat{d}_{\boldsymbol{k}}}{|\mathcal{I}_{\boldsymbol{M}}|}\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\left(\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right)} \right) \nabla\varphi\left(\boldsymbol{x}_j-\boldsymbol{l}\odot\boldsymbol{M}^{-1}\right). \tag{3.17}$$

as gradient NFFT with analytic derivative (ad-NFFT). The main advantage of this approach is that it only requires one FFT for evaluating the trigonometric polynomial and its gradient, while the i$\boldsymbol{k}$-NFFT requires 3 additional FFTs for computing the gradient.

So far we did not take into account that the nodes are located in the possibly truncated unit cube $\mathbf{S}\big[-1/2, 1/2\big)^3$. In the following, we present an adaption of the NFFT algorithm that offers reduced memory consumption and arithmetic operations whenever the truncation is sufficiently large. In general, the computation of the convolution sums (3.12) and (3.16) requires all FFT outputs $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}$. However, the index set of necessary FFT outputs can be substantially reduced, if we have the more restrictive assumption $\boldsymbol{x} \in \mathbf{S}\big[-1/2, 1/2\big)^3 \subset \big[-1/2, 1/2\big)^3$. Therefore, we define a grid size $\boldsymbol{m} = (m_0, m_1, m_2)^{\mathsf{T}} \in 2\mathbb{N}$ with

$$m_t := \min\left\{M_t, 2\left\lceil \tfrac{1}{2} S_t M_t + c_\varphi \right\rceil\right\}, \quad t = 0, 1, 2. \tag{3.18}$$

Then, one can easily verify that the FFT outputs $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}$, are sufficient to compute any convolution sum appearing in (3.12) and (3.16). This fact is illustrated in Figure 3.1. Therefore, we can state the FFT step (3.11) more precisely by the following pruned FFT with shifted index sets

$$g_{\boldsymbol{l}} = \frac{1}{|\mathcal{I}_{\boldsymbol{M}}|} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{g}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \left(\boldsymbol{l} \odot \boldsymbol{M}^{-1}\right)}, \quad \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}. \tag{3.19}$$

By definition of $\boldsymbol{m}$ this becomes equal to (3.11) whenever the node scaling matrix $\mathbf{S} \approx \mathbf{I}_3$ is close to the identity matrix. The additional pruning of the FFT outputs can already give a noticeable speedup. Even more, it is a mandatory prerequisite for parallel scalability as we will discuss in Section 3.3. Figure 3.1 shows a two-dimensional illustration of the additional condition $\boldsymbol{x}_j \in \mathbf{S}\big[-1/2, 1/2\big)^3$ and its implications on the serial computation of the convolution sums (3.12). Furthermore, we present a two-dimensional illustration of the adapted serial NFFT algorithm in Figure 3.2.

In matrix-vector notation, the NFFT approximation idea can be summarized as

$$\mathbf{A}_{N,\hat{\boldsymbol{m}}} \hat{\boldsymbol{f}} \approx \mathbf{C}_{N,\boldsymbol{m}} \widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} \mathbf{D}_{\hat{\boldsymbol{m}}} \hat{\boldsymbol{f}}, \tag{3.20}$$

where $\mathbf{A}_{N,\hat{\boldsymbol{m}}}$ denotes a nonequispaced Fourier matrix and the matrix factors are defined as following. The deconvolution step (3.10) corresponds to the real valued $|\mathcal{I}_{\hat{\boldsymbol{m}}}| \times |\mathcal{I}_{\hat{\boldsymbol{m}}}|$ diagonal matrix

$$\mathbf{D}_{\hat{\boldsymbol{m}}} := \operatorname{diag}\left(|\mathcal{I}_{\boldsymbol{M}}|^{-1} \hat{d}_{\boldsymbol{k}}\right)_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \quad \text{with} \quad \hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}}^{-1}. \tag{3.21}$$

The matrix representation of the pruned FFT with shifted index sets (3.11) is given by $\widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} := (\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})})_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}; \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}}$, i.e., a Fourier matrix of original size $\boldsymbol{M}$ that was shifted and truncated to size $|\mathcal{I}_{\boldsymbol{m}}| \times |\mathcal{I}_{\hat{\boldsymbol{m}}}|$. Note that the prefactor $|\mathcal{I}_{\boldsymbol{M}}|^{-1}$ from (3.19) moved into the definition of the diagonal matrix $\mathbf{D}_{\hat{\boldsymbol{m}}}$. Furthermore, $\mathbf{C}_{N,\boldsymbol{m}}$ denotes the sparse, real valued $N \times |\mathcal{I}_{\boldsymbol{m}}|$ matrix

$$\mathbf{C}_{N,\boldsymbol{m}} := (c_{j,\boldsymbol{l}})_{j \in \mathcal{N}; \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}}, \quad c_{j,\boldsymbol{l}} := \begin{cases} \varphi\left(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1}\right) & : \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}, c_\varphi}(\boldsymbol{x}_j) \\ 0 & : \boldsymbol{l} \notin \mathcal{I}_{\boldsymbol{M}, c_\varphi}(\boldsymbol{x}_j) \end{cases} \tag{3.22}$$

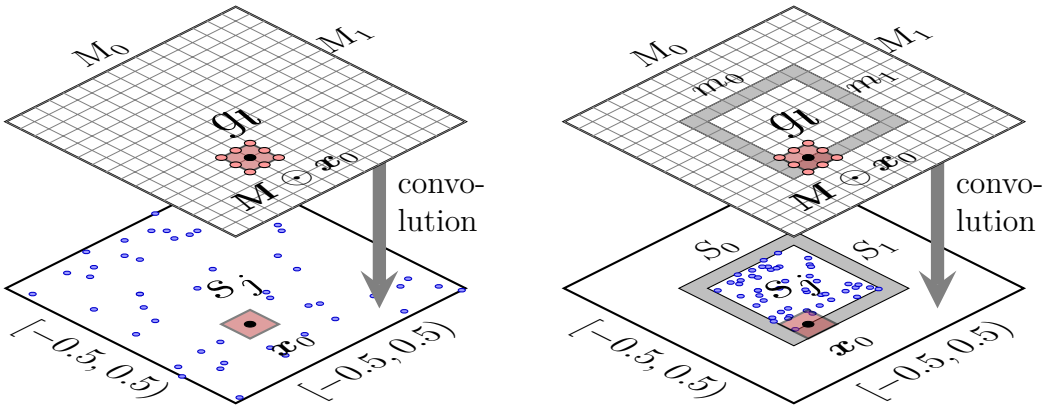

Figure 3.1: Two-dimensional illustration of the serial convolution workflow according to (3.12) for node scaling matrix $\mathbf{S} = \operatorname{diag}(1, 1)$ on the left and node scaling matrix $\mathbf{S} = \operatorname{diag}(3/8, 3/8)$ on the right. For both illustrations we chose $N = 50$ nonequispaced nodes, FFT size $\boldsymbol{M} = (16, 16)^\mathsf{T}$, pruned FFT output size $\boldsymbol{m} = (8, 8)^\mathsf{T}$, and window cutoff parameter $c_\varphi = 1$. The computation of the convolution sums (3.12) is illustrated at the example of a single black colored node $\boldsymbol{x}_0 = (-2/16, -3/16)^\mathsf{T}$. A red rectangle indicates the support of the truncated window function centered at node $\boldsymbol{x}_0$. Note that each sum s_j depends on at most $(2c_\varphi + 1)^2$ grid values $g_{\boldsymbol{l}}$. In general, for every grid value $g_{\boldsymbol{l}}$ there may exist a node $\boldsymbol{x}_j$ that depends on $g_{\boldsymbol{l}}$ for the computation of the convolution sum s_j. On the right hand side, no grid value $g_{\boldsymbol{l}}$ outside the gray colored border takes part in the computation of the convolution sums s_j. Therefore, it is sufficient to compute the innermost $m_0 \times m_1$ grid values $g_{\boldsymbol{l}}$. Note that the gray border of width c_φ results from the support of the window function that may exceed the range of nodes.

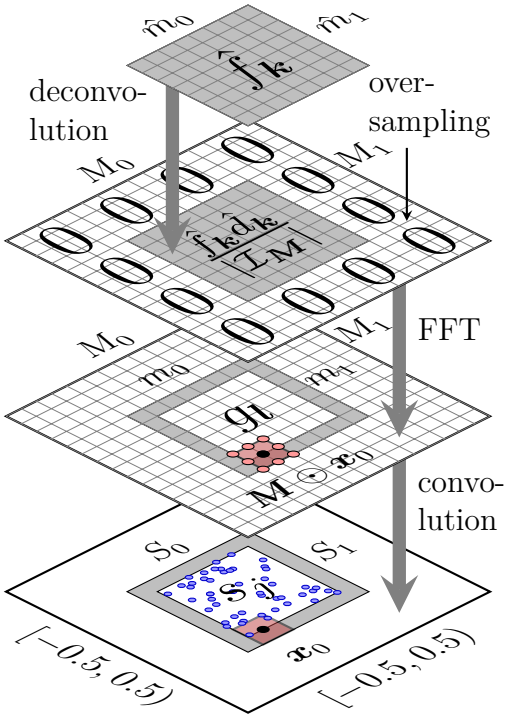

Figure 3.2: Two-dimensional illustration of the serial NFFT workflow for $\hat{\boldsymbol{m}} = (8, 8)^{\mathsf{T}}$ given Fourier coefficients, FFT size $\boldsymbol{M} = (16, 16)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (8, 8)^{\mathsf{T}}$, node scaling matrix $\mathbf{S} = \operatorname{diag}(3/8, 3/8)$, $N = 50$ nonequispaced nodes, and window cutoff parameter $c_\varphi = 1$. At the beginning, the given Fourier coefficients $\hat{f}_{\boldsymbol{k}}$ are point-wise multiplied according to the deconvolution formula (3.10) and mapped into an FFT array of size $\boldsymbol{M}$. Afterward, an FFT of size $\boldsymbol{M}$ is performed according to (3.11). The computation of the convolution sums (3.12) is illustrated at the example of a single black colored node $\boldsymbol{x}_0 = (-2/16, -3/16)^{\mathsf{T}}$. A red rectangle indicates the support of the truncated window function centered at node $\boldsymbol{x}_0$. Note that $(2c_\varphi + 1)^2$ grid points are sufficient to compute each convolution sum s_j.

that represents the discrete convolution step (3.12). From this matrix representation we get immediately an approximate adjoint NFFT by

$$\mathbf{A}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}}\boldsymbol{f} \approx \mathbf{D}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}^{\mathsf{H}}_{\boldsymbol{m},\hat{\boldsymbol{m}}}\mathbf{C}^{\mathsf{T}}_{N,\boldsymbol{m}}\boldsymbol{f}. \tag{3.23}$$

Analogously, the ad-NFFT (3.17) is given by

$$\nabla\mathbf{A}_{3N,\hat{\boldsymbol{m}}}\hat{\boldsymbol{f}} \approx \nabla\mathbf{C}_{3N,\boldsymbol{m}}\widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}}\mathbf{D}_{\hat{\boldsymbol{m}}}\hat{\boldsymbol{f}} \tag{3.24}$$

with the sparse, real valued $3N \times |\mathcal{I}_{\boldsymbol{m}}|$ matrix $\nabla\mathbf{C}_{3N,\boldsymbol{m}} := (\nabla c_{j,\boldsymbol{l}})_{j\in\mathcal{N};\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{m}}}$,

$$\nabla c_{j,\boldsymbol{l}} := \begin{cases} \nabla\varphi\left(\boldsymbol{x}_j - \boldsymbol{l}\odot\boldsymbol{M}^{-1}\right) & : \boldsymbol{l}\in\mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j), \\ \mathbf{0} & : \boldsymbol{l}\notin\mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j). \end{cases} \tag{3.25}$$

Note that this matrix representation differs from the NFFT only in the multiplication with the last matrix $\mathbf{C}_{N,\boldsymbol{m}}$ or $\nabla\mathbf{C}_{3N,\boldsymbol{m}}$, respectively. Since the window function φ is defined as a tensor product, the evaluation of function values for both matrices can be easily combined. As we can see in (3.5) for a given node $\boldsymbol{x} = (x_0, x_1, x_2)^{\mathsf{T}} \in \left[-1/2, 1/2\right)^3$ it is sufficient to evaluate the one-dimensional window function $\tilde{\psi}$ and its derivative $\tilde{\psi}'$ at the three coordinates x_0, x_1, x_2.

The i$\boldsymbol{k}$-NFFT (3.15) can be written as

$$\nabla\mathbf{A}_{3N,\hat{\boldsymbol{m}}}\hat{\boldsymbol{f}} \approx \left(\mathbf{C}_{N,\boldsymbol{m}}\widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} \otimes \mathbf{I}_3\right)\mathbf{K}_{3\hat{\boldsymbol{m}},\hat{\boldsymbol{m}}}\mathbf{D}_{\hat{\boldsymbol{m}}}\hat{\boldsymbol{f}} \tag{3.26}$$

with the block-diagonal matrix

$$\mathbf{K}_{3\hat{\boldsymbol{m}},\hat{\boldsymbol{m}}} := (-2\pi\mathrm{i}\boldsymbol{k}\delta_{\boldsymbol{k}-\boldsymbol{l}})_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}};\boldsymbol{l}\in\mathcal{I}_{\hat{\boldsymbol{m}}}}.$$

Hereby, $\delta_{\boldsymbol{k}}$ denotes the Kronecker symbol that equals 1 for $\boldsymbol{k} = \mathbf{0}$ and becomes zero otherwise.

3.2.1 Window functions

To keep the approximation error small, several window functions φ with good localization in spatial and frequency domain have been proposed in the literature. In the following, we give an overview of five univariate window functions together with their first derivative and their continuous Fourier transform. Thereby, $c_\varphi \in \mathbb{N}$ is a given window cutoff parameter and W_p denotes the centered cardinal B-Spline of order p. The cardinal sine function is defined by $\operatorname{sinc}(x) := \sin(x)/x$ for $x \neq 0$ and $\operatorname{sinc}(0) := 1$. For the sake of simplicity, we also introduce the analogue definition $\operatorname{sinhc}(x) := \sinh(x)/x$ for $x \neq 0$ and $\operatorname{sinhc}(0) := 1$. I_0 and I_1 denote the zeroth and first order modified Bessel function of the first kind, respectively. Furthermore, for a given number of Fourier coefficients $\hat{m}_t$ and FFT mesh size M_t along dimension $t = 0, 1, 2$, we define the oversampling factors $\sigma_t := M_t/\hat{m}_t \in \mathbb{R}$.

1. For a shape parameter $b_t = \frac{2\sigma_t}{2\sigma_t - 1}\frac{c_\varphi}{\pi}$ the dilated *Gaussian window* [42, 121, 41] is given by

$$\begin{aligned} \psi_t(x) &= (\pi b_t)^{-1/2} \exp\left(-(M_t x)^2 / b_t\right), \\ \psi_t'(x) &= -2M_t^2 x \psi_t(x)/b_t, \\ \hat{\psi}_t(k) &= M_t^{-1} \exp\left(-(\pi k/M_t)^2 b_t\right). \end{aligned} \tag{3.27}$$

2. The dilated *B-Spline window* [26, 121] is given by

$$\begin{aligned} \psi_t(x) &= W_{2c_\varphi}(M_t x), \\ \psi_t'(x) &= M_t \left(W_{2c_\varphi - 1}(M_t x + 1/2) - W_{2c_\varphi - 1}(M_t x - 1/2)\right), \\ \hat{\psi}_t(k) &= M_t^{-1} \operatorname{sinc}^{2c_\varphi}(k\pi/M_t). \end{aligned} \tag{3.28}$$

3. For a shape parameter $b_t = \frac{2\sigma_t - 1}{2\sigma_t}\frac{\pi}{c_\varphi} M_t$ the dilated *Sinc window* is given by

$$\begin{aligned} \psi_t(x) &= \tfrac{b_t}{\pi} \operatorname{sinc}^{2c_\varphi}(b_t x), \\ \psi_t'(x) &= \frac{b_t}{\pi} \begin{cases} 0 & \text{for } x = 0, \\ \frac{2c_\varphi}{x}\left(\cos(b_t x) - \operatorname{sinc}(b_t x)\right) \operatorname{sinc}^{2c_\varphi - 1}(b_t x) & \text{otherwise,} \end{cases} \\ \hat{\psi}_t(k) &= W_{2c_\varphi}(k\pi/b_t). \end{aligned} \tag{3.29}$$

4. For a shape parameter $b_t = \pi(2 - \frac{1}{\sigma_t})$ the dilated *Bessel-I0 window* [76] is given by

$$\psi_t(x) = \frac{1}{2} \begin{cases} \mathrm{I}_0\left(b_t \sqrt{c_\varphi^2 - M_t^2 x^2}\right) & \text{for } |M_t x| \le c_\varphi, \\ 0 & \text{otherwise,} \end{cases} \tag{3.30}$$

$$\psi_t'(x) = -\frac{1}{2} \begin{cases} b_t M_t^2 x \left(c_\varphi^2 - M_t^2 x^2\right)^{-1/2} \mathrm{I}_1\left(b_t \sqrt{c_\varphi^2 - M_t^2 x^2}\right) & \text{for } |M_t x| < c_\varphi, \\ \frac{1}{2} b_t^2 M_t c_\varphi & \text{for } |M_t x| = c_\varphi, \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{\psi}_t(k) = \frac{c_\varphi}{M_t} \begin{cases} \operatorname{sinhc}\left(c_\varphi \sqrt{b_t^2 - (2\pi k/M_t)^2}\right) & \text{for } |k| < M_t\left(1 - \frac{1}{2\sigma_t}\right), \\ \operatorname{sinc}\left(c_\varphi \sqrt{(2\pi k/M_t)^2 - b_t^2}\right) & \text{otherwise.} \end{cases}$$

5. For a shape parameter $b_t = \pi(2 - \frac{1}{\sigma_t})$, and the definitions $u_1(x) := \cosh(x) - \operatorname{sinhc}(x)$, $u_2(x) := \cos(x) - \operatorname{sinc}(x)$ the dilated *Kaiser-Bessel window* [56, 112,

96] is given by

$$\psi_t(x) = \frac{b_t}{\pi} \begin{cases} \operatorname{sinhc}\left(b_t\sqrt{c_\varphi^2 - M_t^2x^2}\right) & \text{for } |M_tx| < c_\varphi, \\ \operatorname{sinc}\left(b_t\sqrt{M_t^2x^2 - c_\varphi^2}\right) & \text{otherwise,} \end{cases} \tag{3.31}$$

$$\psi_t'(x) = \frac{b_t}{\pi}\frac{M_t^2x}{M_t^2x^2 - c_\varphi^2} \begin{cases} u_1\left(b_t\sqrt{c_\varphi^2 - M_t^2x^2}\right) & \text{for } |M_tx| < c_\varphi, \\ 0 & \text{for } |M_tx| = c_\varphi, \\ u_2\left(b_t\sqrt{M_t^2x^2 - c_\varphi^2}\right) & \text{otherwise,} \end{cases}$$

$$\hat{\psi}_t(k) = \frac{1}{M_t} \begin{cases} I_0\left(c_\varphi\sqrt{b_t^2 - (2\pi k/M_t)^2}\right) & \text{for } |k| \le M_t\left(1 - \frac{1}{2\sigma_t}\right), \\ 0 & \text{otherwise.} \end{cases}$$

Note that (3.28) and (3.29) as well as (3.30) and (3.31) are closely related due to an interchange of time and frequency domain. Furthermore, we see that (3.28) and (3.30) have compact support in time domain while (3.29) and (3.31) have compact support in frequency domain. Note that the choice of the shape parameter b is nontrivial and can be further optimized if the input Fourier coefficients $\hat{f}_{\boldsymbol{k}}$ in (3.2) provide fast decay [102].

In the case of the Gaussian window function (3.27), the evaluations of the exponential function `exp()` can be reduced substantially by fast Gaussian gridding; see [61] and [84, Appendix C]. However, our numerical experiments showed that the evaluation of the Gaussian window can be realized even more efficiently by interpolation from short precomputed interpolation tables as described in the following. Another benefit of the interpolation approach is that it is not restricted to the Gaussian window. This means, beside a short precomputation of one-dimensional function value tables, all window functions can be evaluated with the same efficiency that even competes with fast Gaussian gridding.

In order to reduce the computational cost of the evaluation of the window functions, we use tensor structure based precomputation and interpolation from look-up tables [84]. Therefore, we reduce the computation of the $(2c_\varphi+1)^3$ window function values per node in (3.12) to $3(2c_\varphi+1)$ interpolations of the one-dimensional window function $\tilde{\psi}$ and $(2c_\varphi+1)^3$ multiplications in order to compute the tensor products $\varphi(\boldsymbol{x} - \boldsymbol{l} \odot \boldsymbol{M}^{-1}) = \tilde{\psi}(x_0 - {}^{l_0}/_{M_0})\tilde{\psi}(x_1 - {}^{l_1}/_{M_1})\tilde{\psi}(x_2 - {}^{l_2}/_{M_2})$. Note that the computing time of the window function with tensor product based interpolation does not depend on the particular choice of the window function. In our implementation we support constant, linear, quadratic, and cubic interpolation of the one-dimensional window function $\psi(x)$.

Additionally, it may be advantageous to precompute the Fourier coefficients $\hat{\varphi}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$. Hereby, we do not store the full set of $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ Fourier coefficients. Instead, we exploit the tensor structure of the three-dimensional window function $\hat{\varphi}_{\boldsymbol{k}} = \hat{\psi}(k_0)\hat{\psi}(k_1)\hat{\psi}(k_2)$, $\boldsymbol{k} = (k_0, k_1, k_2)^\mathsf{T} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$ and store the precomputed $\hat{m}_0 +$

$\hat{m}_1 + \hat{m}_2$ Fourier coefficients of the one-dimensional window functions $\hat{\psi}_t(k_t)$, $k_t = -\hat{m}_t/2, \ldots, \hat{m}_t/2 - 1$, $t = 0, 1, 2$. Therefore, the evaluation of the three-dimensional Fourier coefficients requires $2|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ multiplications.

3.2.2 Shifted NFFT

In this section we introduce a slightly modified NFFT approximation that we denote as shifted NFFT. This idea is closely connected to interlaced particle-mesh algorithms, as we will see in Section 4.6. The starting point is the following substitution in the definition (3.2) of the NDFT. The NDFT can be reinterpreted as the evaluation of a trigonometric polynomial $f^{\mathsf{s}}\colon \mathbb{R}^3 \to \mathbb{C}$,

$$f^{\mathsf{s}}(\boldsymbol{y}) := \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}^{\mathsf{s}}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{y}}, \quad \hat{f}^{\mathsf{s}}_{\boldsymbol{k}} := \hat{f}_{\boldsymbol{k}} \mathrm{e}^{\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{M}^{-1}}, \tag{3.32}$$

at the nonequispaced nodes $\boldsymbol{y}_j := \boldsymbol{x}_j + 1/2 \boldsymbol{M}^{-1}$. Note that we are free to wrap the nodes $\boldsymbol{y}_j$ periodically into $\left[-1/2, 1/2\right)^3$. The alternative formulation (3.32) is of the exactly the same structure as (3.2). Therefore, we simply apply the same approximation scheme as for the NFFT and, afterward, invert the substitutions. This directly leads to the matrix notation of the shifted NFFT

$$\mathbf{A}_{N,\hat{\boldsymbol{m}}} \approx \widetilde{\mathbf{C}}_{N,\boldsymbol{m}} \widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} \widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}} \tag{3.33}$$

with the following definitions of the matrix factors. Due to the substitution $\hat{f}^{\mathsf{s}}_{\boldsymbol{k}} = \mathrm{e}^{\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{M}^{-1}} \hat{f}_{\boldsymbol{k}}$, the deconvolution matrix (3.21) turns into the complex valued $|\mathcal{I}_{\hat{\boldsymbol{m}}}| \times |\mathcal{I}_{\hat{\boldsymbol{m}}}|$ diagonal matrix

$$\widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}} := \operatorname{diag}\left(|\mathcal{I}_{\boldsymbol{M}}|^{-1} \hat{d}_{\boldsymbol{k}} \mathrm{e}^{\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{M}^{-1}} \right)_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \quad \text{with} \quad \hat{d}_{\boldsymbol{k}} = \hat{\varphi}^{-1}_{\boldsymbol{k}}.$$

The matrix representation $\widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} := (\mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})})_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}; \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}}$ of the pruned FFT with shifted index sets is exactly the same as for the non-shifted NFFT. Furthermore, $\widetilde{\mathbf{C}}_{N,\boldsymbol{m}} := (\tilde{c}_{j,\boldsymbol{l}})_{j \in \mathcal{N}; \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}}$ denotes a sparse, real valued $N \times |\mathcal{I}_{\boldsymbol{m}}|$ matrix with

$$\tilde{c}_{j,\boldsymbol{l}} := \begin{cases} \varphi\left(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1} + \frac{1}{2}\boldsymbol{M}^{-1}\right) & : \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}, c_\varphi}(\boldsymbol{x}_j), \\ 0 & : \boldsymbol{l} \notin \mathcal{I}_{\boldsymbol{M}, c_\varphi}(\boldsymbol{x}_j) \end{cases} \tag{3.34}$$

that represents the discrete convolution step (3.12) with a window function φ shifted by half the inverse mesh size $1/2 \boldsymbol{M}^{-1}$. From this matrix representation we get immediately an adjoint shifted NFFT by

$$\mathbf{A}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} \approx \widetilde{\mathbf{D}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}} \widetilde{\mathbf{F}}^{\mathsf{H}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} \widetilde{\mathbf{C}}^{\mathsf{T}}_{N,\boldsymbol{m}}.$$

Analogously, the shifted version of i$\boldsymbol{k}$-NFFT reads as

$$\nabla \mathbf{A}_{3N,\hat{\boldsymbol{m}}} \approx \left(\widetilde{\mathbf{C}}_{N,\boldsymbol{m}} \widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} \otimes \mathbf{I}_3\right) \mathbf{K}_{3\hat{\boldsymbol{m}},\hat{\boldsymbol{m}}} \widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}$$

and shifted ad-NFFT can be written as

$$\nabla \mathbf{A}_{3N,\hat{\boldsymbol{m}}} \approx \nabla \widetilde{\mathbf{C}}_{3N,\boldsymbol{m}} \widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}} \widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}$$

with the sparse, real valued $3N \times |\mathcal{I}_{\boldsymbol{m}}|$ matrix $\nabla \widetilde{\mathbf{C}}_{3N,\boldsymbol{m}} := \left(\nabla \tilde{c}_{j,\boldsymbol{l}}\right)_{j \in \mathcal{N}; \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}}$,

$$\nabla \tilde{c}_{j,\boldsymbol{l}} := \begin{cases} \nabla \varphi\left(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1} + \frac{1}{2}\boldsymbol{M}^{-1}\right) & : \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j), \\ \mathbf{0} & : \boldsymbol{l} \notin \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j). \end{cases} \tag{3.35}$$

It is instructive to have a look at the following alternative derivation of the shifted NFFT. This time, we start with an ansatz similar to (3.6)

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}} \\ \approx \breve{t}(\boldsymbol{x}) &:= \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} h_{\boldsymbol{l}} \varphi(\boldsymbol{x} - \boldsymbol{l} \odot \boldsymbol{M}^{-1} + \tfrac{1}{2}\boldsymbol{M}^{-1}) \qquad (3.36) \\ &= \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{h}_{\boldsymbol{k}} \mathrm{e}^{-\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{M}^{-1}} \sum_{\boldsymbol{r} \in \mathbb{Z}^3} (-1)^{|\boldsymbol{r}|} \hat{\varphi}_{\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M}} \mathrm{e}^{-2\pi \mathrm{i} (\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M})^{\mathsf{T}} \boldsymbol{x}}. \qquad (3.37) \end{aligned}$$

Hereby, $|\boldsymbol{r}| := r_0 + r_1 + r_2$ denotes the sum over all components of the vector $\boldsymbol{r} \in \mathbb{Z}^3$ and $\hat{h}_{\boldsymbol{k}} := \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} h_{\boldsymbol{l}} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}$, $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}}$, are the $\boldsymbol{M}$-periodic discrete Fourier coefficients of $h_{\boldsymbol{l}}$. Note that the important difference between (3.6) and (3.36) is the shift of all window functions by half the inverse mesh size $1/2\boldsymbol{M}^{-1}$. Now, one can repeat the derivation of the NFFT in an analogous manner to the non-shifted case, cf. (3.7)–(3.12). Again, this leads to matrix representation (3.33) and we skip the details. Our main benefit of this equivalent formulation (3.36) comes from the Fourier series (3.37) of $\breve{t}(\boldsymbol{x})$. We will use it later on for the investigation of the aliasing error.

On its own the shifted NFFT will not give any advantage over the non-shifted NFFT. In the average case both algorithms will give the same approximation error, see Section 3.2.4. This is not very surprising since for an arbitrary trigonometric polynomial $f(\boldsymbol{x})$ and nodes $\boldsymbol{x}_j$ there is no reason to favor one of the two methods. However, we will see in the next section that both methods can be coupled in a way that indeed reduces the NFFT approximation error.

3.2.3 Interlaced NFFT

In the following, we couple the shifted and non-shifted NFFT in order to reduce the approximation error significantly. The main idea comes from the fact that largest

approximation errors of the NFFT can be observed at nodes that maximize the distance to their nearest mesh points. However, these nodes will be very close to a mesh shifted by $1/2\boldsymbol{M}^{-1}$ and the approximation error will be smaller with a shifted NFFT for these nodes. Therefore, we conjecture that an average of the NFFT and shifted NFFT results will give an improved approximation. We start with an ansatz

$$f(\boldsymbol{x}) \approx \tfrac{1}{2}\left(\breve{s}(\boldsymbol{x}) + \breve{t}(\boldsymbol{x})\right),$$

that is an average of the NFFT ansatz (3.6) and the shifted NFFT ansatz (3.36). The derivation of the corresponding NFFT algorithm is straightforward and we only summarize its matrix notation

$$\mathbf{A}_{N,\hat{\boldsymbol{m}}} \approx \tfrac{1}{2}\left(\mathbf{C}_{N,\boldsymbol{m}}\widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}}\mathbf{D}_{\hat{\boldsymbol{m}}} + \widetilde{\mathbf{C}}_{N,\boldsymbol{m}}\widetilde{\mathbf{F}}_{\boldsymbol{m},\hat{\boldsymbol{m}}}\widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}\right). \tag{3.38}$$

Similar matrix factorizations result for the i$\boldsymbol{k}$-NFFT and ad-NFFT. Again, the adjoint transform is derived by taking the adjoint of (3.38) and results in

$$\mathbf{A}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} \approx \tfrac{1}{2}\left(\mathbf{D}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}^{\mathsf{H}}_{\boldsymbol{m},\hat{\boldsymbol{m}}}\mathbf{C}^{\mathsf{T}}_{N,\boldsymbol{m}} + \widetilde{\mathbf{D}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}^{\mathsf{H}}_{\boldsymbol{m},\hat{\boldsymbol{m}}}\widetilde{\mathbf{C}}^{\mathsf{T}}_{N,\boldsymbol{m}}\right).$$

We call these algorithms interlaced NFFT and interlaced NFFT$^{\mathsf{H}}$, respectively. This name is a tribute to [70, page 260 ff.], where interlacing was presented in the context of particle-mesh methods. Thereby, a specialized version of interlacing was presented under the name harmonic averaging. It turns out that this algorithm has a close connection to the interlaced NFFT$^{\mathsf{H}}$. Details on this connection will be given in Section 4.6. However, to the best of our knowledge this is the first time that interlacing is applied in the context of the standalone NFFT modules. Furthermore, we stress that [70, page 260 ff.] did not include some kind of (non-adjoint) interlaced NFFT, i$\boldsymbol{k}$-NFFT or ad-NFFT.

3.2.4 Optimized deconvolution

As already mentioned, the choice (3.10) of deconvolution $\hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}}^{-1}$ for the NFFT is not unique. In the following, we derive distinct optimal choices of $\hat{d}_{\boldsymbol{k}}$ in the sense that certain mean square aliasing errors are minimized.

Minimizing the mean square aliasing error of the NFFT

One may look for those coefficients $\hat{g}_{\boldsymbol{k}}$ that minimize the mean square aliasing error of the NFFT approximation given by

$$\begin{aligned}\|f - \breve{s}\|_2^2 &= \int_{\left[-1/2,1/2\right)^3} |f(\boldsymbol{x}) - \breve{s}(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \\ &= \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}} - \hat{g}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}}|^2 + \sum_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{M}}\setminus\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{g}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}}|^2 + \sum_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{M}}} \sum_{\boldsymbol{r}\in\mathbb{Z}^3\setminus\{\boldsymbol{0}\}} |\hat{g}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}|^2.\end{aligned}$$

Note that this minimization does not include the truncation error that arises in the NFFT approximation for window functions that miss compact support in real space. Obviously, the choice $\hat{g}_{\boldsymbol{k}} = 0$ is optimal for $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}} \setminus \mathcal{I}_{\hat{\boldsymbol{m}}}$. Furthermore, we assume a linear dependency $\hat{g}_{\boldsymbol{k}} = \hat{d}_{\boldsymbol{k}} \hat{f}_{\boldsymbol{k}}$ with $\hat{d}_{\boldsymbol{k}} \in \mathbb{R}$ and get

$$\sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - 2\hat{d}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}} + \hat{d}_{\boldsymbol{k}}^2 \sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}^2 \right).$$

Now computing the derivative with respect to $\hat{d}_{\boldsymbol{k}}$ gives the optimal values

$$\hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}} \Big(\sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}^2 \Big)^{-1}, \quad \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}. \tag{3.39}$$

The same formula was also found in [41, Equation (A-7)] using a randomized approach. There the authors already mentioned that the approximation error resulting from (3.10) and (3.39) are practically equivalent, since for typical window functions the $\boldsymbol{r} = \boldsymbol{0}$ terms dominate the aliasing sum in (3.39). These finding are supported by recent numerical examples in [102, 103]. By the choice (3.39) the optimal mean square aliasing error becomes

$$\sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - \hat{\varphi}_{\boldsymbol{k}}^2 \Big(\sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}^2 \Big)^{-1} \right). \tag{3.40}$$

Exactly the same results (3.39) and (3.40) apply for the shifted NFFT. Furthermore, note that the same choice of optimal deconvolution (3.39) results for the NFFT$^{\mathsf{H}}$.

Minimizing the mean square aliasing error of the ad-NFFT

The ad-NFFT approximation can be written as

$$\nabla f(\boldsymbol{x}) = -2\pi \mathrm{i} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}} \boldsymbol{k} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}} \approx \nabla \breve{s}(\boldsymbol{x}) = \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} g_{\boldsymbol{l}} \nabla \varphi(\boldsymbol{x} - \boldsymbol{l} \odot \boldsymbol{M}^{-1}).$$

and its Fourier series is given by

$$\nabla \breve{s}(\boldsymbol{x}) = -2\pi \mathrm{i} \sum_{\boldsymbol{k} \in \mathbb{Z}^3} \boldsymbol{k} \hat{g}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}}$$

with the $\boldsymbol{M}$-periodic discrete Fourier coefficients $\hat{g}_{\boldsymbol{k}} = \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} g_{\boldsymbol{l}} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}$, $\boldsymbol{k} \in \mathbb{Z}^3$. This time, we want to compute those coefficients $\hat{g}_{\boldsymbol{k}}$ that minimize the mean square aliasing error of the ad-NFFT approximation given by

$$\begin{aligned} &\int_{\left[-1/2,1/2\right)^3} \|\nabla f(\boldsymbol{x}) - \nabla \breve{s}(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x} \\ &= 4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \|\boldsymbol{k}\|^2 |\hat{f}_{\boldsymbol{k}} - \hat{g}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}}|^2 + 4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \sum_{\boldsymbol{r} \in \mathbb{Z}^3 \setminus \{\boldsymbol{0}\}} \|\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M}\|^2 |\hat{g}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}|^2. \end{aligned}$$

Hereby, we already set $\hat{g}_{\boldsymbol{k}} = 0$ for $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}} \setminus \mathcal{I}_{\hat{\boldsymbol{m}}}$. Assuming a linear dependency $\hat{g}_{\boldsymbol{k}} = \hat{d}_{\boldsymbol{k}} \hat{f}_{\boldsymbol{k}}$ with $\hat{d}_{\boldsymbol{k}} \in \mathbb{R}$ for all $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$ we get

$$4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \left(\|\boldsymbol{k}\|^2 - 2\hat{d}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}} \|\boldsymbol{k}\|^2 + \hat{d}_{\boldsymbol{k}}^2 \sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}^2 \|\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M}\|^2 \right),$$

which becomes optimal for

$$\hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}} \|\boldsymbol{k}\|^2 \Big(\sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}^2 \|\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M}\|^2 \Big)^{-1}. \tag{3.41}$$

In the case of a fast decreasing window function the optimal value (3.41) is dominated by the $\boldsymbol{r} = \boldsymbol{0}$ terms and will be close to the simpler deconvolution (3.10). With the optimal choice (3.41) the mean square ad-NFFT aliasing error becomes

$$4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \|\boldsymbol{k}\|^2 \left(1 - \hat{\varphi}_{\boldsymbol{k}}^2 \|\boldsymbol{k}\|^2 \Big(\sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}^2 \|\boldsymbol{k} + \boldsymbol{r} \odot \boldsymbol{M}\|^2 \Big)^{-1} \right). \tag{3.42}$$

Exactly the same results (3.41) and (3.42) apply for the shifted ad-NFFT.

Minimizing the mean square aliasing error of the ik-NFFT

For $t = 0, 1, 2$ we can write the i$\boldsymbol{k}$-NFFT approximation in the t-th vector component as

$$f_{x_t}(\boldsymbol{x}) := \frac{\partial}{\partial x_t} f(\boldsymbol{x}) = -2\pi \mathrm{i} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{f}_{\boldsymbol{k}} k_t \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}} \approx \breve{s}_t(\boldsymbol{x}) = -2\pi \mathrm{i} \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} g_{\boldsymbol{l},t} \varphi(\boldsymbol{x} - \boldsymbol{l} \odot \boldsymbol{M}^{-1}).$$

Thereby, the Fourier series of $\breve{s}_t(\boldsymbol{x})$ is given by

$$\breve{s}_t(\boldsymbol{x}) = -2\pi \mathrm{i} \sum_{\boldsymbol{k} \in \mathbb{Z}^3} \hat{g}_{\boldsymbol{k},t} \hat{\varphi}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}}$$

with the $\boldsymbol{M}$-periodic discrete Fourier coefficients $\hat{g}_{\boldsymbol{k},t} := \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M}}} g_{\boldsymbol{l},t} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}$, $\boldsymbol{k} \in \mathbb{Z}^3$. In the following, we want to minimize the mean square aliasing error of the i$\boldsymbol{k}$-NFFT approximation for each component $t = 0, 1, 2$

$$\begin{aligned}
&\int_{\left[-1/2,1/2\right)^3} |f_{x_t}(\boldsymbol{x}) - \breve{s}_t(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \\
&= 4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} |k_t \hat{f}_{\boldsymbol{k}} - \hat{g}_{\boldsymbol{k},t} \hat{\varphi}_{\boldsymbol{k}}|^2 + 4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}}} \sum_{\boldsymbol{r} \in \mathbb{Z}^3 \setminus \{\boldsymbol{0}\}} |\hat{g}_{\boldsymbol{k},t} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}|^2.
\end{aligned}$$

Again, we already set $\hat{g}_{\boldsymbol{k},t} = 0$ for $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}} \setminus \mathcal{I}_{\hat{\boldsymbol{m}}}$ and assume a linear dependency $\hat{g}_{\boldsymbol{k},t} = \hat{d}_{\boldsymbol{k}} k_t \hat{f}_{\boldsymbol{k}}$ with $\hat{d}_{\boldsymbol{k}} \in \mathbb{R}$ for all $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$. Then, we get

$$4\pi^2 \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} k_t^2 |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - 2\hat{d}_{\boldsymbol{k}} \hat{\varphi}_{\boldsymbol{k}} + \hat{d}_{\boldsymbol{k}}^2 \sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}} \right)$$

which becomes optimal for

$$\hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}} \Big(\sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}} \Big)^{-1}. \tag{3.43}$$

Note that the optimal value $\hat{d}_{\boldsymbol{k}}$ is equal for all dimensions $t = 0, 1, 2$. Especially, it is equal to the optimal deconvolution coefficients (3.39) of the plain NFFT. In the case of a fast decreasing window function the optimal value (3.43) is dominated by the $\boldsymbol{r} = \boldsymbol{0}$ terms and will be close to the simpler deconvolution (3.10). With the optimal choice (3.43) the minimal mean square i$\boldsymbol{k}$-NFFT aliasing error becomes

$$4\pi^2 \sum_{t=0}^{2} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} k_t^2 |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - \hat{\varphi}^2_{\boldsymbol{k}} \Big(\sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}} \Big)^{-1} \right). \tag{3.44}$$

Exactly the same results (3.43) and (3.44) apply for the shifted i$\boldsymbol{k}$-NFFT.

Minimizing the mean square aliasing error of the interlaced NFFT

We start with the Fourier series of the NFFT and shifted NFFT approximations given in (3.7) and (3.37) as

$$\begin{aligned}
\breve{s}(x) &= \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{g}_{\boldsymbol{k}} \sum_{\boldsymbol{r} \in \mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}} \mathrm{e}^{-2\pi \mathrm{i} (\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M})^{\mathsf{T}} \boldsymbol{x}}, \\
\breve{t}(x) &= \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{h}_{\boldsymbol{k}} \mathrm{e}^{-\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{M}^{-1}} \sum_{\boldsymbol{r} \in \mathbb{Z}^3} (-1)^{|\boldsymbol{r}|} \hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}} \mathrm{e}^{-2\pi \mathrm{i} (\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M})^{\mathsf{T}} \boldsymbol{x}}.
\end{aligned}$$

Hereby, $|\boldsymbol{r}| := r_0 + r_1 + r_2$ denotes the sum over all component of the vector $\boldsymbol{r} \in \mathbb{Z}^3$. Motivated by the structure of the optimal NFFT deconvolution (3.39) we make the ansatz $\hat{g}_{\boldsymbol{k}} = \hat{d}_{\boldsymbol{k}} \hat{f}_{\boldsymbol{k}}$, $\hat{h}_{\boldsymbol{k}} = \hat{d}_{\boldsymbol{k}} \mathrm{e}^{\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{M}^{-1}} \hat{f}_{\boldsymbol{k}}$ with a real coefficient $\hat{d}_{\boldsymbol{k}}$. Then, the mean

square aliasing error of the interlaced NFFT is given by

$$\begin{aligned}
&\int_{\left[-1/2,1/2\right)^3} |f(\boldsymbol{x}) - \tfrac{1}{2}\breve{s}(\boldsymbol{x}) - \tfrac{1}{2}\breve{t}(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \\
&= \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}} - \tfrac{1}{2}(\hat{g}_{\boldsymbol{k}} + \hat{h}_{\boldsymbol{k}}\mathrm{e}^{-\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{M}^{-1}})\hat{\varphi}_{\boldsymbol{k}}|^2 \\
&\quad + \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \sum_{\boldsymbol{r}\in\mathbb{Z}^3\setminus\{\boldsymbol{0}\}} |\tfrac{1}{2}(\hat{g}_{\boldsymbol{k}} + \hat{h}_{\boldsymbol{k}}\mathrm{e}^{-\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{M}^{-1}}(-1)^{|\boldsymbol{r}|})\hat{\varphi}_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}|^2 \\
&= \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - \hat{d}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}}\right)^2 + \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \hat{d}_{\boldsymbol{k}}^2 \sum_{\boldsymbol{r}\in\mathbb{Z}^3\setminus\{\boldsymbol{0}\}} \tfrac{1}{4}\left[(1+(-1)^{|\boldsymbol{r}|})\right]^2 \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}} \\
&= \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - 2\hat{\varphi}_{\boldsymbol{k}}\hat{d}_{\boldsymbol{k}} + \hat{d}_{\boldsymbol{k}}^2 \sum_{\boldsymbol{r}\in\mathbb{Z}^3} \tfrac{1}{2}\left[1+(-1)^{|\boldsymbol{r}|}\right] \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}\right).
\end{aligned}$$

Hereby, we set $\hat{g}_{\boldsymbol{k}} = 0$ for all $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{M}}\setminus\mathcal{I}_{\hat{\boldsymbol{m}}}$ in the second line and used $\left[1+(-1)^{|\boldsymbol{r}|}\right]^2 = 2\left[1+(-1)^{|\boldsymbol{r}|}\right]$ in the last line. Now, optimization of $\hat{d}_{\boldsymbol{k}}$ for each $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$ results in

$$\hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}}\Big(\sum_{\boldsymbol{r}\in\mathbb{Z}^3} \tfrac{1}{2}\left[1+(-1)^{|\boldsymbol{r}|}\right] \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}\Big)^{-1}. \tag{3.45}$$

With this choice of deconvolution the aliasing error reaches its minimal value

$$\sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} |\hat{f}_{\boldsymbol{k}}|^2 \left(1 - \hat{\varphi}^2_{\boldsymbol{k}}\Big(\sum_{\boldsymbol{r}\in\mathbb{Z}^3} \tfrac{1}{2}\left[1+(-1)^{|\boldsymbol{r}|}\right] \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}\Big)^{-1}\right).$$

By comparison to the optimal aliasing error (3.40) of the NFFT we see that only the aliasing sum in the denominator changed. While the terms for even $|\boldsymbol{r}| = r_0+r_1+r_2$ coincide for both cases, the terms for odd $|\boldsymbol{r}|$ vanish in the interlaced case. Therefore, the term $\hat{\varphi}^2_{\boldsymbol{k}}\big(\sum_{\boldsymbol{r}\in\mathbb{Z}^3} 1/2\left[1+(-1)^{|\boldsymbol{r}|}\right] \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}\big)^{-1}$ is closer to 1 and the approximation error is smaller for the interlaced NFFT. Again, for fast decreasing windows the $\boldsymbol{r} = \boldsymbol{0}$ term will dominate the aliasing sum and a good approximation of the optimal deconvolution coefficients (3.45) is given by the simpler choice (3.10).

Analogously, we can derive the optimal deconvolution coefficients for the computing the gradients with interlaced NFFT. As we have already seen before, the optimal deconvolution of the NFFT and i$\boldsymbol{k}$-NFFT are the same. This is also the case for the interlaced NFFT and interlaced i$\boldsymbol{k}$-NFFT. In contrast, the mean square aliasing error of the interlaced ad-NFFT is minimized for

$$\hat{d}_{\boldsymbol{k}} = \hat{\varphi}_{\boldsymbol{k}}\|\boldsymbol{k}\|^2\Big(\sum_{\boldsymbol{r}\in\mathbb{Z}^3} \tfrac{1}{2}\left[1+(-1)^{|\boldsymbol{r}|}\right] \hat{\varphi}^2_{\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}}\|\boldsymbol{k}+\boldsymbol{r}\odot\boldsymbol{M}\|^2\Big)^{-1}. \tag{3.46}$$

3.3 The parallel three-dimensional NFFT

In this section we describe a parallel algorithm for computing the three-dimensional NFFT on massively parallel, distributed memory architectures. The parallel implementation of this algorithm is based on the massage passing interface (MPI) [100]. In principle, our Parallel Nonequispaced Fast Fourier Transform (PNFFT) framework combines the serial three-dimensional NFFT algorithm from Section 3.2 with a three-dimensional block domain decomposition, cf. Section 2.4.4.

3.3.1 Parallel data decomposition

In the following, we describe parallel algorithms for computing the NDFT and NFFT. Thereby, we assume a parallel hardware architecture that can be described as a three-dimensional process mesh $\mathcal{P}_{\mathbf{P}} := \{0,\dots,P_0-1\} \times \{0,\dots,P_1-1\} \times \{0,\dots,P_2-1\}$ of size $\boldsymbol{P} = (P_0,P_1,P_2)^{\mathsf{T}} \in \mathbb{N}^3$. Each parallel process will be identified with its corresponding multi-index $\boldsymbol{r} = (r_0,r_1,r_2)^{\mathsf{T}} \in \mathcal{P}_{\mathbf{P}}$. For every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ we define the multi-index set

$$\mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}} = \left\{ (k_0,k_1,k_2)^{\mathsf{T}} \in \mathcal{I}_{\hat{\boldsymbol{m}}} : 0 \leq k_t - r_t \left\lceil \tfrac{\hat{m}_t}{P_t} \right\rceil + \tfrac{\hat{m}_t}{2} < \left\lceil \tfrac{\hat{m}_t}{P_t} \right\rceil,\ t = 0,1,2 \right\}.$$

At the beginning of our parallel algorithm, we assume the NFFT input array of $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ complex numbers to be distributed among the three-dimensional process mesh $\mathcal{P}_{\mathbf{P}}$ such that every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ holds the input data $\hat{f}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}$, in its local memory. Note that this corresponds to the three-dimensional block decomposition $\lceil \hat{m}_0/P_0 \rceil \times \lceil \hat{m}_1/P_1 \rceil \times \lceil \hat{m}_2/P_2 \rceil$ with default block sizes $\lceil \hat{m}_t/P_t \rceil$ as described in Section 2.4.4.

3.3.2 Description of the algorithm

The serial NFFT algorithm starts with the deconvolution step (3.10) that consists of $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ independent point-wise multiplications. These operations can be distributed over all processes according to the parallel block decomposition, i.e., every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ computes

$$\hat{g}_{\boldsymbol{k}} = |\mathcal{I}_{\boldsymbol{M}}|^{-1} \hat{d}_{\boldsymbol{k}} \hat{f}_{\boldsymbol{k}}, \quad \boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}. \tag{3.47}$$

Figure 3.3 shows a two-dimensional illustration of the serial deconvolution workflow according to (3.10) and the parallel deconvolution workflow according to (3.47).

The second stage (3.19) of the serial NFFT consists of a three-dimensional pruned FFT with shifted index sets. Thereby, the FFT size equals $\boldsymbol{M}$, the FFT inputs are pruned to size $\hat{\boldsymbol{m}} \leq \boldsymbol{M}$, and the FFT outputs are pruned to size $\boldsymbol{m} \leq \boldsymbol{M}$. Obviously, this stage can be performed in parallel with the PFFT Framework 2.5 that we derived in Section 2.5. The parallel pruned FFT with shifted index sets can

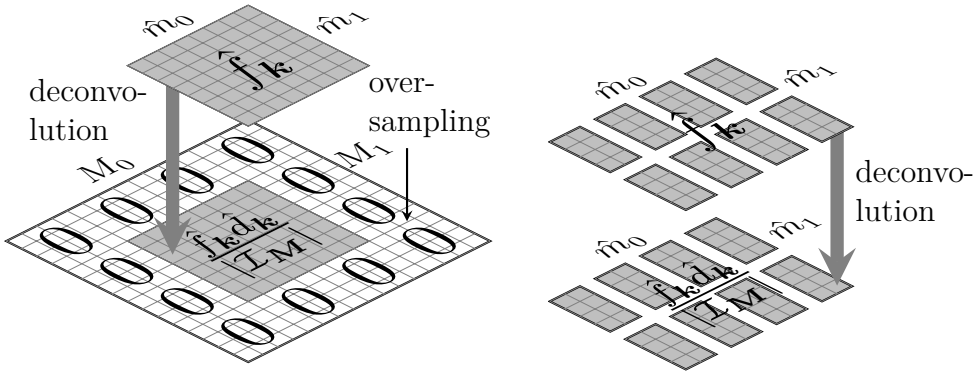

Figure 3.3: Two-dimensional illustration of the serial deconvolution workflow according to (3.10) on the left and the parallel deconvolution workflow according to (3.47) on the right. We chose $\hat{\boldsymbol{m}} = (8,8)^{\mathsf{T}}$ given Fourier coefficients, FFT size $\boldsymbol{M} = (16,16)^{\mathsf{T}}$, and a process mesh of size $\boldsymbol{P} = (4,2)^{\mathsf{T}}$. The serial algorithm uses explicit mapping of the incoming Fourier coefficients $\hat{f}_{\boldsymbol{k}}$ into an FFT array of size $\boldsymbol{M}$ filled with zeros. In contrast, our parallel data distribution avoids the distribution of zeros and, therefore, does not depend on the FFT size $\boldsymbol{M}$.

be formally written as

$$g_{\boldsymbol{l}} = \sum_{\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{s}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}} \hat{g}_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}, \quad \boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{m},\boldsymbol{P}}. \tag{3.48}$$

Thereby, the formal order of summation was chosen to symbolize the parallel data decomposition of our block distributed PFFT. The inner sum reflects that every process $\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}$ starts with calculations on its locally available input data block of size $[\hat{m}_0/P_0] \times [\hat{m}_1/P_1] \times [\hat{m}_2/P_2]$. The outer sum stands for the global communication that must be performed somehow within the parallel FFT algorithm. After the parallel FFT the output data $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}$, is distributed on the process mesh in a similar way as the input data set, i.e., every process owns a block of $[m_0/P_0] \times [m_1/P_1] \times [m_2/P_2]$ complex numbers. Again, for every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ the multi-index set

$$\mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{m},\boldsymbol{P}} := \left\{ (k_0,k_1,k_2)^{\mathsf{T}} \in \mathcal{I}_{\boldsymbol{M}} : 0 \leq k_t - r_t \left\lceil \tfrac{m_t}{P_t} \right\rceil + \tfrac{m_t}{2} < \left\lceil \tfrac{m_t}{P_t} \right\rceil, \; t = 0,1,2 \right\}$$

collects all the multi-indexes of locally available data. At this point we see that the ability of PFFT to handle pruned FFT with shifted index sets in a scalable manner is mandatory for parallel NFFT. This feature is crucial in order to assure a good load balancing of our parallel NFFT framework. It is important to recall that PFFT is based on a two-dimensional domain decomposition, i.e., the three-dimensional decomposed FFT input $\hat{g}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}$, and output $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{m},\boldsymbol{P}}$ is redistributed before and after every parallel FFT. Therefore, an upper limit for the

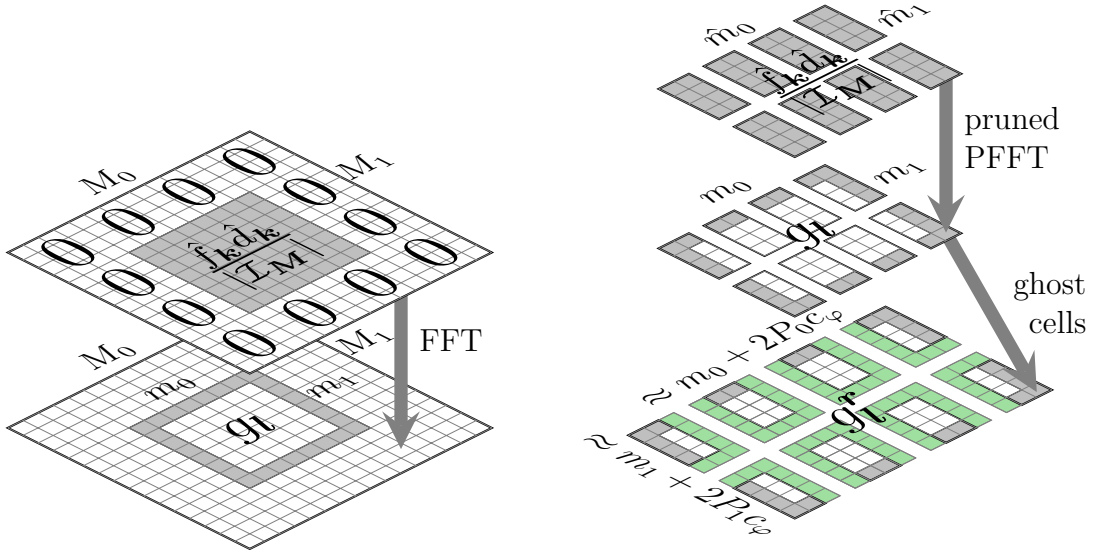

Figure 3.4: Two-dimensional illustration of the serial FFT workflow according to (3.11) on the left and the parallel pruned FFT workflow according to (3.48) on the right. We chose $\hat{\boldsymbol{m}} = (8,8)^{\mathsf{T}}$ given Fourier coefficients, FFT size $\boldsymbol{M} = (16,16)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (8,8)^{\mathsf{T}}$, window cutoff parameter $c_\varphi = 1$, and a process mesh of size $\boldsymbol{P} = (4,2)^{\mathsf{T}}$. A naive block decomposition of the FFT on the left would lead to several processes that own input blocks full of zeros before the FFT and output blocks full with unnecessary data $g_{\boldsymbol{l}}$ outside the gray colored border after the FFT. Instead, our algorithm uses a parallel pruned FFT that works with a block distribution of the necessary input and output data. Note that the parallel NFFT needs to communicate a border of c_φ (green colored) ghost cells according to (3.50) in order to prepare the parallel convolution according to (3.51).

number of processes is given by the two-dimensional decomposition of the parallel FFT. Figure 3.4 shows a two-dimensional illustration of the serial FFT workflow according to (3.19) and the parallel pruned FFT workflow according to (3.48). Note that the parallel ghost cell duplication step (3.50) within this illustration will be explained in the next paragraphs.

The block data distribution of the FFT output $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m}}$, naturally implies a block decomposition of the truncated unit cube $\mathbf{S}\big[-1/2, 1/2\big)^3$. This motivates the definition of the index sets

$$\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}} := \big\{ j \in \mathcal{N} : \exists \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{r}} \text{ with } \boldsymbol{l} \le \boldsymbol{M} \odot \boldsymbol{x}_j < \boldsymbol{l} + \boldsymbol{1} \big\} \tag{3.49}$$

for every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$. We assign all nodes $\boldsymbol{x}_j$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$, to process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$. As one can already see, heterogeneous distributions of the nodes $\boldsymbol{x}_j$, $j \in \mathcal{N}$, may lead

to imbalances in memory consumption and workload between the processes, which is a typical problem of mesh based domain decompositions.

According to the discrete convolution step of the serial NFFT, we compute the sums (3.12), which run over the local multi-index sets $\mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$. Our choice (3.18) of parameter $\boldsymbol{m}$ assures $\mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j) \subset \mathcal{I}_{\boldsymbol{m}}$ for every $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$, i.e., the output of the pruned FFT is sufficient. But in general, not all sufficient data $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)$, is located on a single process $\boldsymbol{r}$. Therefore, we perform a communication step in order to gather all the additionally needed data. This step equals a ghost cell communication that was described in Section 2.8 and only involves nearest neighbor communication. We introduce the multi-index sets

$$\mathcal{I}_{\boldsymbol{m},\boldsymbol{P},c_\varphi}^{\boldsymbol{r}} := \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{r}} + \{-c_\varphi, \ldots, c_\varphi\}^3$$

for all processes $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$. Thereby, the sum is understood as the common set operation $A + B := \{a + b \colon a \in A, b \in B\}$. However, we declare that all additions are performed modulus $\boldsymbol{M}$ such that we never leave $\mathcal{I}_{\boldsymbol{M}}$. Then, we symbolize the ghost cell communication for every parallel process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ by

$$g_{\boldsymbol{l}}^{\boldsymbol{r}} = g_{\boldsymbol{l}}, \quad \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P},c_\varphi}^{\boldsymbol{r}}. \tag{3.50}$$

Note that every process must gather the data $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P},c_\varphi}^{\boldsymbol{r}} \setminus \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{r}}$, from its nearest neighbors. In our implementation we use the ghost cell support of the PFFT software library; cf. Section 2.8. Figure 3.4 also includes a two-dimensional illustration of the parallel ghost cell communication according to (3.50). Finally, the sums

$$s_j = \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^{\boldsymbol{r}}\, \varphi(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1}), \quad j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}, \tag{3.51}$$

are calculated locally on all processes $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$. Figure 3.5 shows a two-dimensional illustration of the serial convolution workflow according to (3.12) and the parallel convolution workflow according to (3.51). Framework 3.1 summarizes the PNFFT Framework in pseudo code and Figure 3.6 gives a summarizing two-dimensional illustration of the serial and parallel NFFT workflow.

The Parallel Adjoint Nonequispaced Fast Fourier Transform (PNFFT$^{\mathsf{H}}$) can be derived analogously from the serial NFFT$^{\mathsf{H}}$ algorithm (3.23). Note that the transposed counterpart of the ghost cell communication (3.50) is a sum over all ghost cells, i.e.,

$$g_{\boldsymbol{l}} = \sum_{\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}} g_{\boldsymbol{l}}^{\boldsymbol{s}}, \quad \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{r}}.$$

The resulting pseudo code of the PNFFT$^{\mathsf{H}}$ is given by Framework 3.2.

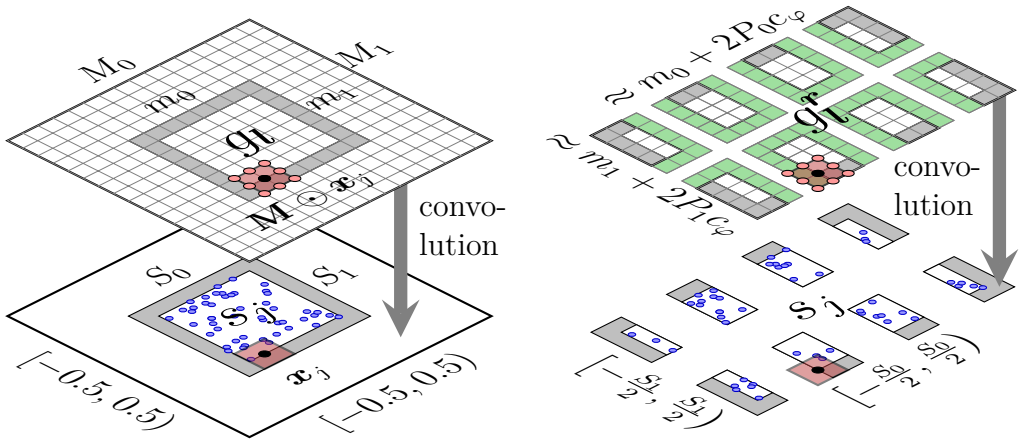

Figure 3.5: Two-dimensional illustration of the serial convolution workflow according to (3.12) on the left and the parallel convolution workflow according to (3.51) on the right. We chose FFT size $\boldsymbol{M} = (16, 16)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (8, 8)^{\mathsf{T}}$, node scaling matrix $\mathbf{S} = \operatorname{diag}(3/8, 3/8)$, $N = 50$ nonequispaced nodes, window cutoff parameter $c_\varphi = 1$, and a process mesh of size $\boldsymbol{P} = (4, 2)^{\mathsf{T}}$. In both cases the computation of the convolution sums (3.12) is illustrated at the example of a single black circled node $\boldsymbol{x}_j$. The support of the truncated window function centered at node $\boldsymbol{x}_j$ is given by the red rectangle. Again, a small number of $(2c_\varphi + 1)^2$ (red circled) grid points is sufficient in order to compute the convolution sum (3.12) corresponding to this node. In the parallel case, the nonequispaced nodes have been block distributed according to the block distribution of the FFT outputs $g_{\boldsymbol{l}}$. Every process is able to compute the convolution sums s_j for all locally available nodes $\boldsymbol{x}_j$ from the locally available data $g_{\boldsymbol{l}}^r$.

Framework 3.1 PNFFT – Parallel NFFT for each process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$

Input: $\boldsymbol{x}_j \in \mathbf{S}\big[-1/2, 1/2\big)^3$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$, and $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$

1: For $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$ compute $\hat{g}_{\boldsymbol{k}} \leftarrow |\mathcal{I}_{\boldsymbol{M}}|^{-1} \hat{d}_{\boldsymbol{k}} \hat{f}_{\boldsymbol{k}}$.

2: For $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{r}}$ compute $g_{\boldsymbol{l}} \leftarrow \sum_{\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{s}}} \hat{g}_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}$ by a parallel, three-dimensional, pruned FFT with shifted index sets.

3: For $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P},c_\varphi}^{\boldsymbol{r}}$ gather $g_{\boldsymbol{l}}^{\boldsymbol{r}} \leftarrow g_{\boldsymbol{l}}$ by a ghost cell communication.

4: For $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$ compute $s_j \leftarrow \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^{\boldsymbol{r}} \, \varphi(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1})$.

5: For $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$ compute $\nabla s_j \leftarrow \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^{\boldsymbol{r}} \, \nabla\varphi(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1})$.

Output: Approximate function values $s_j \approx f_j$ and gradients $\nabla s_j \approx \nabla f_j$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$

Arithmetic cost: $\mathcal{O}\left(|\mathcal{I}_{\boldsymbol{M}}| \log |\mathcal{I}_{\boldsymbol{M}}| + (2c_\varphi + 1)^3 N\right)$ + evaluations of window function

Framework 3.2 PNFFT$^{\mathsf{H}}$ – Parallel NFFT$^{\mathsf{H}}$ for each process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$

Input: $\boldsymbol{x}_j \in \mathbf{S}\big[-1/2, 1/2\big)^3$, and $f_j \in \mathbb{C}$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$

1: For $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P},c_\varphi}^{\boldsymbol{r}}$ compute $g_{\boldsymbol{l}}^{\boldsymbol{r}} \leftarrow \sum_{\substack{\{j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}: \\ \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{M},c_\varphi}(\boldsymbol{x}_j)\}}} f_j \, \varphi(\boldsymbol{x}_j - \boldsymbol{l} \odot \boldsymbol{M}^{-1})$.

2: For $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{r}}$ accumulate $g_{\boldsymbol{l}} \leftarrow \sum_{\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}} g_{\boldsymbol{l}}^{\boldsymbol{s}}$ by a transposed ghost cell communication.

3: For $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$ compute $\hat{g}_{\boldsymbol{k}} \leftarrow \sum_{\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}} \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{m},\boldsymbol{P}}^{\boldsymbol{s}}} g_{\boldsymbol{l}} \, \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \boldsymbol{M}^{-1})}$ by an parallel, three-dimensional, pruned FFT$^{\mathsf{H}}$ with shifted index sets.

4: For $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$ compute $\hat{s}_{\boldsymbol{k}} \leftarrow |\mathcal{I}_{\boldsymbol{M}}|^{-1} \hat{d}_{\boldsymbol{k}} \hat{g}_{\boldsymbol{k}}$.

Output: Approximate coefficients $\hat{s}_{\boldsymbol{k}} \approx \hat{h}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$

Arithmetic cost: $\mathcal{O}\left(|\mathcal{I}_{\boldsymbol{M}}| \log |\mathcal{I}_{\boldsymbol{M}}| + (2c_\varphi + 1)^3 N\right)$ + evaluations of window function

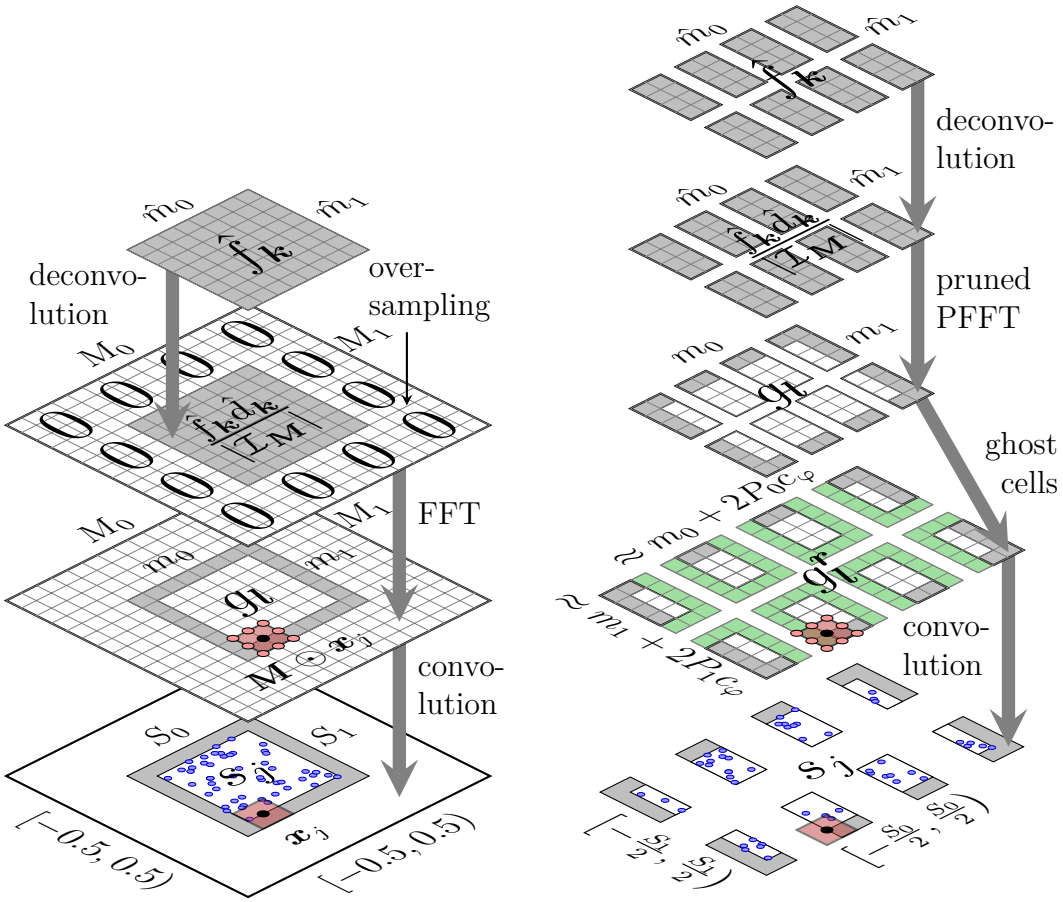

Figure 3.6: Two-dimensional illustration of the serial NFFT workflow without pruned FFT on the left and the parallel NFFT workflow of the PNFFT Framework 3.1 on the right. We chose $\hat{\boldsymbol{m}} = (8, 8)^{\mathsf{T}}$ given Fourier coefficients, FFT size $\boldsymbol{M} = (16, 16)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (8, 8)^{\mathsf{T}}$, node scaling matrix $\mathbf{S} = \operatorname{diag}(3/8, 3/8)$, $N = 50$ nonequispaced nodes, window cutoff parameter $c_\varphi = 1$, and a process mesh of size $\boldsymbol{P} = (4, 2)^{\mathsf{T}}$. For detailed explanations of the separate steps, see Figures 3.3, 3.4, and 3.5.

3.4 The parallel three-dimensional NDFT

In Section 3.3.1 we defined a block domain decomposition that led to the PNFFT Framework. Based on the same decomposition it is very simple to construct a parallel algorithm for the direct computation of the NDFT. Although the NDFT yields a worse complexity, it is still useful for comparison reasons and can be used to investigate the approximation error of the NFFT for small problem sizes. Furthermore, the NDFT will recur as a module of plain Ewald summation in Section 4.7.2. Therefore, it comes handy to have algorithmic modules for the computation of the Parallel Nonequispaced Discrete Fourier Transform (PNDFT) and its adjoint (PNDFT$^{\mathsf{H}}$). Later on, these modules will be the main ingredient for parallel Ewald summation.

In principle, our parallel NDFT algorithm starts with a block decomposition of the Fourier mesh and a partition of the nodes $\boldsymbol{x}_j$ into $|\mathcal{P}_{\mathbf{P}}|$ disjoint subsets $\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}} \subset \mathcal{N}$. One possibility to construct the subsets $\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$ was given in (3.49) by a block decomposition of the unit cube. However, the following two algorithms will not depend on this particular choice of decomposition. The pseudo code of the parallel computation of the NDFT is given by Module 3.3 and it operates as follows. After an initialization the algorithm iterates over all $|\mathcal{P}_{\boldsymbol{P}}|$ processes. In each step, one process broadcasts its local block of Fourier coefficients to all the other processes. Thereby, we denote the locally buffered copy of coefficient $\hat{f}_{\boldsymbol{k}}$ on process $\boldsymbol{r}$ by $\hat{f}_{\boldsymbol{k}}^{\boldsymbol{r}}$. Afterward, each process can locally compute the subtotal that belongs to this block of coefficients. Note that this algorithm is perfectly scalable in terms of memory requirements and arithmetic operations as long as $|\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}|$ is almost equal over all processes $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$. However, the number of expensive broadcast communications increases with $|\mathcal{P}_{\boldsymbol{P}}|$.

A similar parallel algorithm for computing the NDFT$^{\mathsf{H}}$ is given by Module 3.4. Loosely speaking it evolves from taking the adjoint of all the steps in the PNDFT module. Thereby, the adjoint of a broadcast is a reduce summation. The parallel scalability is exactly the same as for the PNDFT module.

3.5 The PNFFT software library

We implemented the Frameworks 3.1 and 3.2 for computing the parallel NFFT and its adjoint as part of a publicly available software library called PNFFT. The development code is distributed under a GNU General Public License and is freely available within the Git repository [12]. As far as we know this is the first publicly available implementation of nonequispaced fast Fourier transforms for massively parallel, distributed memory architectures. Note that a highly scalable butterfly algorithm for the computation of Fourier integral operators has been presented in [115]. Although this algorithm contains the NFFT as a special case, its runtime is much higher due to its generality. In the following, we highlight selected features of our parallel NFFT implementation in order to give an impression of its flexibility.

Module 3.3 PNDFT – Parallel NDFT for each process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$

Input: $\boldsymbol{x}_j \in \left[-1/2, 1/2\right)^3$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$, and $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$

1: **for** $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$ **do**
2: $\quad s_j \leftarrow 0$
3: **end for**
4: **for** $\boldsymbol{p} \in \mathcal{P}_{\mathbf{P}}$ **do**
5: $\quad$ Receive $\hat{f}_{\boldsymbol{k}}^{\boldsymbol{r}} \leftarrow \hat{f}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{p}}$, from process $\boldsymbol{p}$ by a parallel broadcast.
6: $\quad$ **for** $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$ **do**
7: $\quad\quad s_j \leftarrow s_j + \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{p}}} \hat{f}_{\boldsymbol{k}}^{\boldsymbol{r}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}_j}$
8: $\quad\quad \nabla s_j \leftarrow \nabla s_j - 2\pi \mathrm{i} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{p}}} \hat{f}_{\boldsymbol{k}}^{\boldsymbol{r}} \boldsymbol{k} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{x}_j}$
9: $\quad$ **end for**
10: **end for**

Output: Function values $s_j \approx f_j$ and gradients $\nabla s_j \approx \nabla f_j$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$
Arithmetic cost: $\mathcal{O}\left(|\mathcal{I}_{\hat{\boldsymbol{m}}}| N\right)$

Module 3.4 PNDFT$^{\mathsf{H}}$ – Parallel NDFT$^{\mathsf{H}}$ for each process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$

Input: $\boldsymbol{x}_j \in \left[-1/2, 1/2\right)^3$, and $f_j \in \mathbb{C}$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$

1: **for** $\boldsymbol{p} \in \mathcal{P}_{\mathbf{P}}$ **do**
2: $\quad$ **for** $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{p}}$ **do**
3: $\quad\quad \hat{h}_{\boldsymbol{k}}^{\boldsymbol{r}} \leftarrow \sum_{j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}} f_j \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}$
4: $\quad$ **end for**
5: $\quad$ Compute $\hat{s}_{\boldsymbol{k}} \leftarrow \sum_{\boldsymbol{s} \in \mathcal{P}_{\mathbf{P}}} \hat{h}_{\boldsymbol{k}}^{\boldsymbol{s}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{p}}$, on process $\boldsymbol{p}$ by a parallel reduce.
6: **end for**

Output: Fourier coefficients $\hat{s}_{\boldsymbol{k}} \approx \hat{h}_{\boldsymbol{k}}, \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}},\boldsymbol{P}}^{\boldsymbol{r}}$
Arithmetic cost: $\mathcal{O}\left(|\mathcal{I}_{\hat{\boldsymbol{m}}}| N\right)$

- The application of PNFFT is split into a time consuming planning step and a high performance execution step. Thereby, the amount of planning can be adjusted via flags.
- Installing the library is easy. It is based on the common sequence of configure, make, and make install.
- The interface of PNFFT is a mixture between the MPI interface of FFTW and the NFFT [83] interface. Porting code from NFFT to PNFFT is straightforward.
- PNFFT is written in C but also offers a Fortran 2003 interface.
- PNFFT supports 5 different window functions that can be chosen at the planing step via flags, see Section 3.2.1 for details about the available windows. All window functions can be evaluated at the same efficiency via interpolation from precomputed look-up tables. The user can choose between interpolation of degree 0, 1, 2, and 3. However, also direct evaluation of the windows and fast Gaussian gridding is available.
- PNFFT supports the parallel computation of interlaced NFFTs. To the best knowledge of the authors there is no other publicly available NFFT implementation that supports interlacing at all.
- PNFFT supports fast gradient computation via analytic and i$\boldsymbol{k}$-differentiation, cf. (3.24) and (3.26).
- An adapted algorithm design ensures that PNFFT can efficiently compute pruned NFFTs in parallel. This is especially important when NFFT is applied to non-periodic functions as we will see in Chapter 4.
- We integrated a timer interface into PNFFT. This means that all performance relevant computation and communication is automatically timed throughout the execution of PNFFT. For example, the user can easily access the runtime of the NFFT matrix factors in (3.20) separately. This is especially helpful for benchmark and tuning purposes. If the timer interface is not needed, the user has the possibility to switch it off during the configuration of the library.
- The amount of global communication can be halved, if the underlying parallel FFT is allowed to end up with transposed data layout (2.22), cf. Remark 2.11. In this case PNFFT starts with a transposed data layout in Fourier space.

3.6 Numerical results

In the following, we present selected performance results of our PNFFT library that have been published in [5]. We refer the reader to this reference for more elaborate performance tests. All runtime measurements have been performed on JUGENE – a Blue Gene/P architecture that has been explained in detail in Section 2.10.1. For our numerical experiments we chose the Kaiser-Bessel window function (3.31) with cubic interpolation from precomputed look-up tables, cf. Section 3.2.1. Note

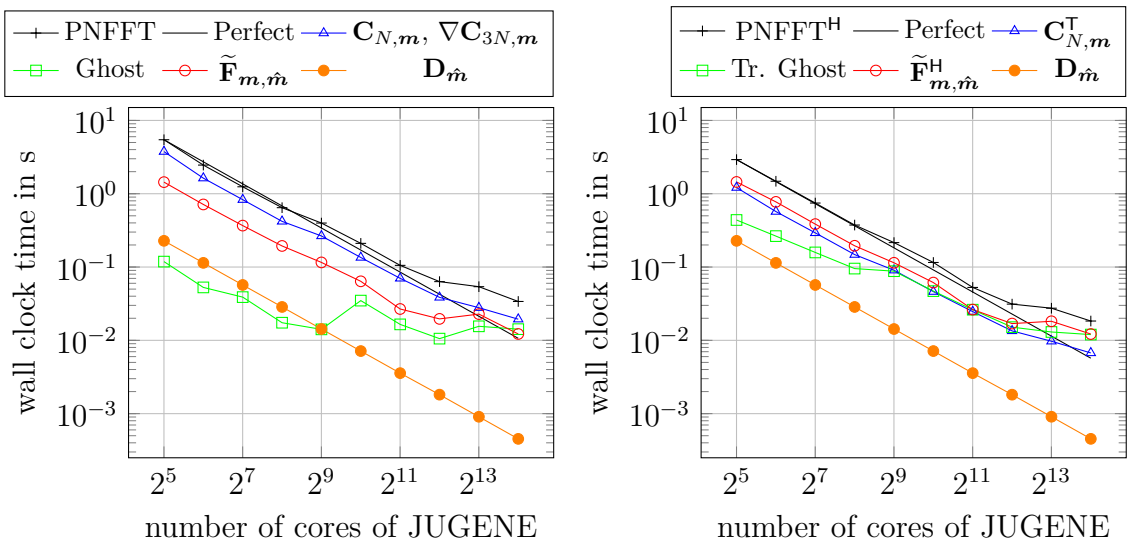

Figure 3.7: Required wall clock time of PNFFT Framework 3.1 (left) and PNFFT$^{\mathsf{H}}$ Framework 3.2 (right) on up to 16 384 cores of JUGENE with $N = 829\,440$ nonequispaced nodes, pruned FFT input size $\hat{\boldsymbol{m}} = (512, 512, 512)^{\mathsf{T}}$, oversampled FFT size $\boldsymbol{M} = (576, 576, 576)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (174, 174, 174)^{\mathsf{T}}$, and window cutoff parameter $c_\varphi = 4$. Timing of PNFFT includes the calculation of the gradient.

that the following results are connected with the numerical results in Section 4.11.4, where PNFFT is executed as part of an application in particle simulation.

3.6.1 Strong scaling of pruned PNFFT on JUGENE

In Figure 3.7 we show the wall clock time measurements of the PNFFT Framework 3.1 and the PNFFT$^{\mathsf{H}}$ Framework 3.2 for 512^3 Fourier coefficients and 829 440 nonequispaced nodes up to 16 384 cores of a Blue Gene/P architecture. For comparison purposes we show the perfect strong scaling times of the first recorded time. In addition, we add the wall clock time of the most time consuming parts of these algorithms. These are the deconvolution step 1 ($\mathbf{D}_{\hat{m}}$), the pruned FFT step 2 with shifted index sets ($\widetilde{\mathbf{F}}_{m,\hat{m}}$) , the ghost cell communication step 3 (ghost), and the convolution steps 4, 5 ($\mathbf{C}_{N,m}$,$\nabla\mathbf{C}_{3N,m}$) of Framework 3.1 and their adjoint counterparts of Framework 3.2. Both plots are scaled equally such that a direct comparison of the time measurement is possible.

The deconvolution steps ($\mathbf{D}_{\hat{m}}$) scale perfectly and only represent a small amount of the overall runtime. The FFT steps ($\widetilde{\mathbf{F}}_{m,\hat{m}}$, $\widetilde{\mathbf{F}}^{\mathsf{H}}_{m,\hat{m}}$) show good strong scaling up to 4096 cores, which corresponds to one rack of JUGENE. We observe a performance penalty for computing the parallel FFT on more than one rack. Moreover, note

that the pruned FFT output is only of size 174^3. The FFT-internal two-dimensional distribution of 174^3 complex numbers on 128^2 processes results in very small workload per process. We stress that the pruned FFT output size $\boldsymbol{m} = (174, 174, 174)^{\mathsf{T}}$ is significantly smaller than the oversampled FFT size $\boldsymbol{M} = (576, 576, 576)^{\mathsf{T}}$ for this test case, i.e., only 2.8% of the FFT output needs to be computed. However, the adapted algorithm design of our parallel NFFT Frameworks takes care of this fact since we apply a parallel pruned FFT and use the data decomposition of the truncated unit cube $\mathbf{S}\left[-1/2, 1/2\right)^3$ instead of the full unit cube $\left[-1/2, 1/2\right)^3$; see also Figure 3.1 for details. Without the availability of pruned parallel FFT there would be no chance to scale this test on such a massively parallel environment. Note that this test case arises from an application of PNFFT for particle simulation that will be described in more detail in Section 4.11.4. This means that the considered test case is not artificial but demonstrates the real problems of parallel FFT and NFFT load balancing.

The discrete convolution of the PNFFT Framework 3.1 include the calculation of the function values ($\mathbf{C}_{N,\boldsymbol{m}}$) and the gradients ($\nabla\mathbf{C}_{3N,\boldsymbol{m}}$), while the discrete convolution of PNFFT$^{\mathsf{H}}$ ($\mathbf{C}^{\mathsf{T}}_{N,\boldsymbol{m}}$) only deals with function values. Therefore, the runtime of PNFFT convolution is larger than for PNFFT$^{\mathsf{H}}$. However, both show good strong scaling behavior. Note that every process gets only 51 nodes $\boldsymbol{x}_j$ in average, if we use 16 384 processes in total.

In contrast, the ghost cell communication shows bad strong scaling for more than 512 processes. The transposed ghost cell communication is more expensive than plain ghost cell communication since it additionally involves the computation and synchronization of the partial sums of ghost cells. For 16 384 processes the transposed ghost cell communication turns out to be the most time consuming part of the PNFFT$^{\mathsf{H}}$. We observe that the ghost cell communication is the most limiting factor for strong scaling. This improves for larger test cases, where the ratio between the ghost cell communication and the overall computing time decreases.

In Figure 3.8 we show the wall clock time measurements of PNFFT Framework 3.1 and PNFFT$^{\mathsf{H}}$ Framework 3.2 for a medium sized test case with 1024^3 Fourier coefficients and 9 447 840 nonequispaced nodes. Furthermore, we present in Figure 3.9 the wall clock time measurements of the same Frameworks 3.1 and 3.2 for a large test case with 2048^3 Fourier coefficients and 103 680 000 nonequispaced nodes.

There, the computing time of the discrete convolution step, the FFT$^{\mathsf{H}}$, and the transposed ghost cell communication are nearly the same. Also the strong scaling behavior of all three steps is good. Although the ghost cell communication on PNFFT does not provide good scaling behavior, it only takes 4% of the overall PNFFT runtime with 65 536 processes. Once more, the deconvolution steps show perfect strong scaling. The parallel pruned FFT shows a performance penalty at 8192 and 16 384 processes. Presumably this comes from the fact, that the underlying two-dimensional process grid exceeds the physical dimensions of one rack along the

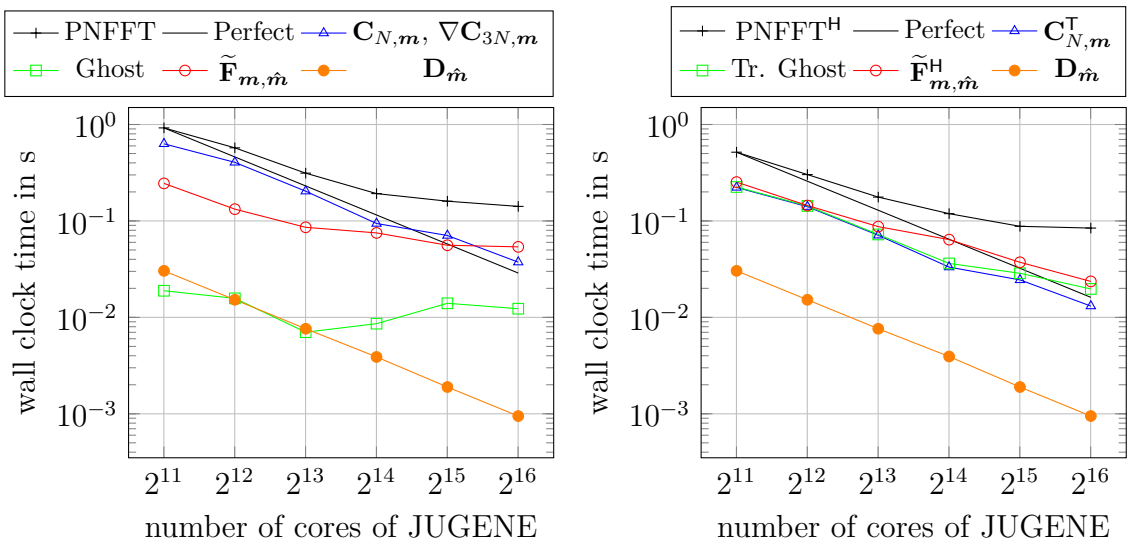

Figure 3.8: Required wall clock time of PNFFT Framework 3.1 (left) and PNFFT$^{\mathsf{H}}$ Framework 3.2 (right) on up to 65 536 cores of JUGENE with $N = 9\,447\,840$ nonequispaced nodes, pruned FFT input size $\hat{\boldsymbol{m}} = (1024, 1024, 1024)^{\mathsf{T}}$, oversampled FFT size $\boldsymbol{M} = (1152, 1152, 1152)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (340, 340, 340)^{\mathsf{T}}$, and window cutoff parameter $c_\varphi = 4$. Timing of PNFFT includes the calculation of the gradient.

first dimension for 8192 processes and along the first and second dimension for more than 8192 processes. We observe a good scaling of the overall wall clock time of PNFFT and PNFFT$^{\mathsf{H}}$ that reflects the good scaling of all their most time consuming steps.

Note that the pruned FFT output sizes used in these examples are not divisible by the number of processes. Therefore, the block decomposition of the parallel FFT does not result in equal blocks and gives rise to load imbalances. These load imbalances become especially obvious for small FFT output sizes $\boldsymbol{m}$ and large process counts. Therefore, the parallel FFTs of the medium sized 9 447 840 particle test case presented in Figure 3.8 show underwhelming strong scaling behavior, whereas the parallel FFTs corresponding to the 103 680 000 particle test case presented in Figure 3.9 show a nearly perfect scaling behavior.

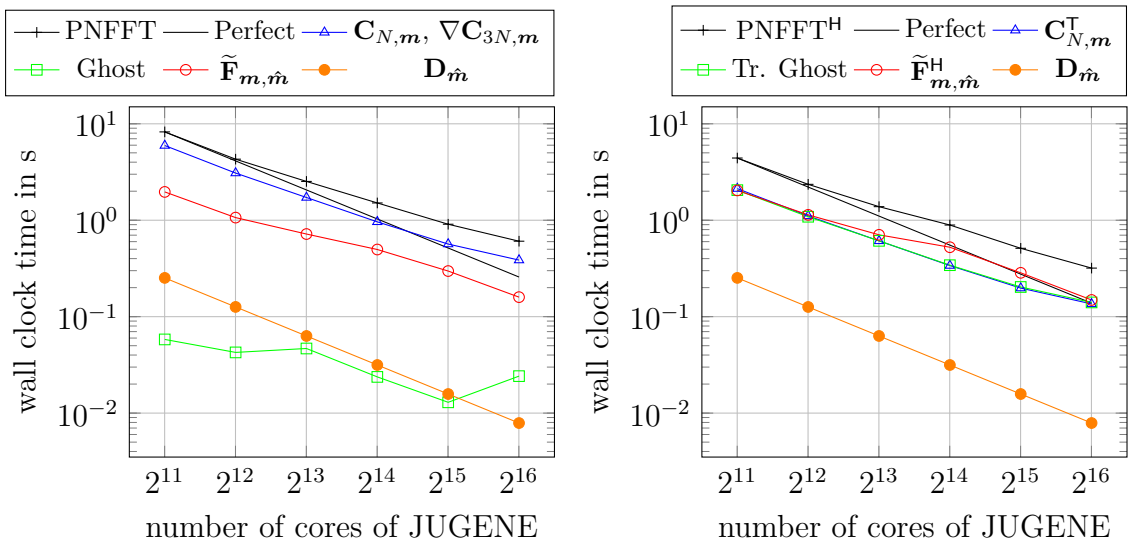

Figure 3.9: Required wall clock time of PNFFT Framework 3.1 (left) and PNFFT$^{\mathsf{H}}$ Framework 3.2 (right) on up to 65 536 cores of JUGENE with $N = 103\,680\,000$ nonequispaced nodes, pruned FFT input size $\hat{\boldsymbol{m}} = (2048, 2048, 2048)^{\mathsf{T}}$, oversampled FFT size $\boldsymbol{M} = (2304, 2304, 2304)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (674, 674, 674)^{\mathsf{T}}$, and window cutoff parameter $c_\varphi = 4$. Timing of PNFFT includes the calculation of the gradient.

4 Parallel particle-mesh methods based on nonequispaced fast Fourier transforms

The computation of electrostatic interactions is an important task in molecular dynamics. Because of the long-range character of the Coulomb potential this becomes a computational demanding task. Simple truncation schemes can not be applied since they produce artifacts [73]. Therefore, the interactions between all N particles have to be considered, which results in a asymptotic complexity of $\mathcal{O}(N^2)$ for open boundary conditions. The situation gets even worse if some kind of periodic boundary conditions are applied. Then, the sum over all infinitely many interactions yields conditional and very slow convergence. Fully 3d-periodic boundary conditions are often assumed in simulations of bulk systems with small particle numbers in order suppress boundary effects that stem from the limited system size. Periodicity in one or two dimension of the three-dimensional space occur in the simulation of nanotubes or thin films, respectively.

A traditional way to sum up the infinite terms under 3d-periodic boundary conditions is known as Ewald summation [49]. This method splits the total interaction into a short-range and a long-range part. Summing up the short-range part in real space and the long-range part in Fourier space leads to two rapidly converging sums. These sums can be truncated and the error can be estimated very well [85]. For optimal choice of parameters, Ewald summation result in $\mathcal{O}(N^{3/2})$ complexity [108, 55] for computing the forces. The same splitting is also the basis for a variety of fast algorithms including P^3M [70, 38], PME [37], SPME [48], NFFT-based fast Ewald summation [68] and the spectrally accurate Ewald method [92]. These methods compute the Fourier space sum with FFTs and result in $\mathcal{O}(N \log N)$ scaling.

Similar Ewald formulas also exist for 2d- and 1d-periodic geometries [65, 111]. However, these formulas do not separate the particle coordinates in the non-periodic dimensions and, therefore, do not straightforwardly lead to fast methods. Therefore, many algorithms with higher complexity than $\mathcal{O}(N \log N)$ have been proposed in the past [131, 133, 18, 30, 29, 19]. Two remarkable exceptions are given by the Electrostatic Layer Correction [20] and the spectrally accurate Ewald summation [93]. Both have complexity $\mathcal{O}(N \log N)$ for 2d-periodic boundary conditions. We emphasize that a common FFT-based framework was presented in [126] that is able to

handle 2d-periodic [99], 1d-periodic [97], and 0d-periodic [98] boundary conditions. However, the approximation idea used in [126] is based on a Fourier approximation of a function that is only continuous. Therefore, the resulting precision of this method is not very high. Last but not least, the NFFT-based fast summation method [112, 113] uses an alternative splitting into a short-range and long-range part that yields $\mathcal{O}(N \log N)$ complexity for 0d-periodic boundary conditions.

It is not obvious which of these methods should be preferred. A closer look shows that most of these methods have a very similar structure, which can be expressed in terms of NFFTs. Indeed, in [9] we pointed out that the building blocks of the fast summation for non-periodic boundary conditions and the NFFT-based fast Ewald summation [68] for periodic boundary conditions are very similar. However, at this point the splitting functions of the two approaches did not match. Later on, we applied the approximation ideas of fast NFFT-based summation to the Ewald splittings for 2d- and 1d-periodic boundary conditions [3] and got fast $\mathcal{O}(N \log N)$ methods for these cases. Finally, the same approximation idea resulted in a fast method in 0d-periodic boundary conditions [6]. All of these methods can be summarized within a common framework called P^{2}NFFT that we are going to describe in this chapter. The main advantage of P^{2}NFFT is its modularized structure that strongly depends on NFFTs. Beside the easy parallelization of the short-range part we only need to apply our highly scalable PNFFT framework from the previous chapter in order to get a parallel version of the whole P^{2}NFFT framework at once. This includes all types of boundary conditions and other algorithmic variations that come with the various interchangeable NFFT modules presented in Chapter 3. Furthermore, we will see that a lot of the above-mentioned fast methods can be interpreted as special cases of P^{2}NFFT. This is especially true for the P^{3}M method, as we discuss in detail in Section 4.6.

Finally, we emphasize that there are a lot of other fast methods for computing the Coulomb interactions, including the Fast Multipole Method [62], multigrid-based methods [125, 28], and the Barnes-Hut tree method [24]. A comparison of P^{2}NFFT with these methods has been published in [1] and some selected results will be presented in Section 4.11.5.

4.1 Definitions

We start with a formal definition of the Coulomb problem subject to mixed periodic boundary conditions. Assume an invertible matrix $\mathbf{L} \in \mathbb{R}^{3\times 3}$ whose columns span the primary box $\mathbf{L}\big[-1/2, 1/2\big)^3 := \{\mathbf{L}\boldsymbol{r} : \boldsymbol{r} \in \big[-1/2, 1/2\big)^3\}$. Furthermore, let N charges $q_j \in \mathbb{R}$ be located within the primary box at positions $\boldsymbol{r}_j \in \mathbf{L}\big[-1/2, 1/2\big)^3$, $j = 1, \ldots, N$. Now, the total Coulomb energy of the particle system can be written

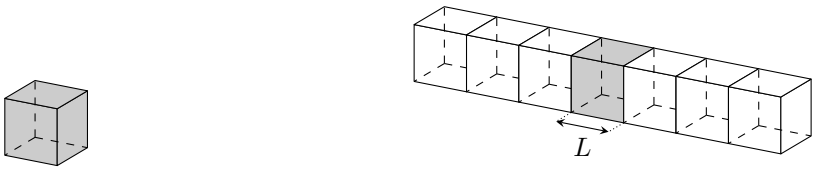

Figure 4.1: In the 0d-periodic case the particles are distributed within a finite box in $\mathbb{R}^3$ (left). For 1d-periodic boundary conditions the simulation box is duplicated with period L along the first dimension (right).

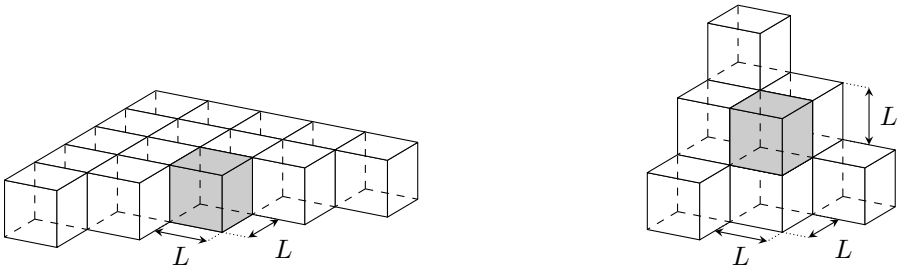

Figure 4.2: In the 2d-periodic case the simulation box is duplicated with period L along two of three dimensions (left). For the 3d-periodic case the simulation box with edge length L is duplicated along all three dimensions (right).

as

$$U_{\mathcal{S}_p} := \frac{1}{2} \sum_{j=1}^{N} q_j \phi_{\mathcal{S}_p}(\boldsymbol{r}_j),$$

where for each particle $\boldsymbol{r}_j$ the potential $\phi_{\mathcal{S}_p}(\boldsymbol{r}_j)$ is formally given by

$$\phi_{\mathcal{S}_p}(\boldsymbol{r}_j) = \sum_{\boldsymbol{n} \in \mathcal{S}_p} {\sum_{i=1}^{N}}' \frac{q_i}{\|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\|}. \tag{4.1}$$

Thereby, we use $\|\cdot\|$ to denote the Euclidean norm and introduce the replicated difference vectors $\boldsymbol{r}_{ij}^{\boldsymbol{n}} := \boldsymbol{r}_i - \boldsymbol{r}_j + \mathbf{L}\boldsymbol{n}$. For the sake of convenience we use Gaussian units, i.e., we skip the prefactor $(4\pi\varepsilon_0)^{-1}$ in definition (4.1). The set of translation vectors $\mathcal{S}_p \subseteq \mathbb{Z}^3$ is defined according to the given boundary conditions as follows. For fixed $p \in \{0, 1, 2, 3\}$ we assume periodic boundary conditions in the first p dimensions and non-periodic (open) boundary conditions in the remaining $3 - p$ dimensions. Then, we set $\mathcal{S}_p := \mathbb{Z}^p \times \{0\}^{3-p}$, i.e., the sum over $\mathcal{S}_p$ in (4.1) can be interpreted as a replication of the primary box along all dimensions subject to periodic boundary conditions; see also Figures 4.1 and 4.2 for graphical illustrations.

Note that the prime on the inner sum in (4.1) indicates that all terms with zero denominator are omitted. In many applications we are also interested in the forces

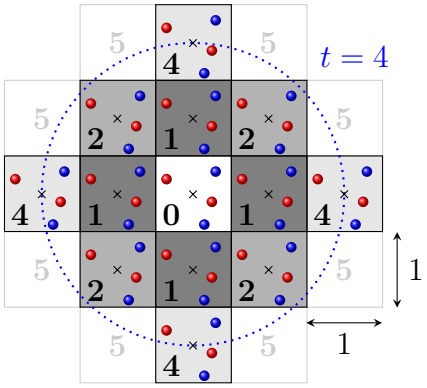

Figure 4.3: Illustration of the spherical summation order (4.3) up to the 4th partial sum of a 2d-periodic unit box. The interactions with a box at distance $\sqrt{t}$ from the primary box are introduced in the t-th partial sum. Box distances are computed between the centers of the boxes that are illustrated as black crosses. Note that the 3rd partial sum is only apparent with 3d-periodic boundary conditions.

acting on the particles, which are given by

$$\boldsymbol{F}_{\mathcal{S}_p}(\boldsymbol{r}_j) := q_j \boldsymbol{E}_{\mathcal{S}_p}(\boldsymbol{r}_j) \quad \text{with the fields} \quad \boldsymbol{E}_{\mathcal{S}_p}(\boldsymbol{r}_j) := -\nabla \phi_{\mathcal{S}_p}(\boldsymbol{r}_j). \tag{4.2}$$

If periodic boundary conditions are apparent in at least one dimension, we claim charge neutrality $\sum_{j=1}^{N} q_j = 0$ in order to assure convergence of the potential (4.1). However, it is important to note that also for charge neutral systems the sums in (4.1) are conditionally convergent, i.e., the values of the potentials $\phi_{\mathcal{S}_p}(\boldsymbol{r}_j)$ depend on the order of summation. We follow [89] to sum up the interactions box wise in a spherically increasing order, i.e., the precise definition of (4.1) is given by

$$\phi_{\mathcal{S}_p}(\boldsymbol{r}_j) := \sum_{t=0}^{\infty} \sum_{\substack{\boldsymbol{n} \in \mathcal{S}_p: \\ \|\boldsymbol{n}\|^2 = t}} {\sum_{i=1}^{N}}' \frac{q_i}{\|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\|}. \tag{4.3}$$

This order of summation is illustrated for the 2d-periodic case in Figure 4.3.

4.2 The Ewald splitting

As the starting point we take the function splitting

$$\frac{1}{r} = \frac{\operatorname{erfc}(\alpha r)}{r} + \frac{\operatorname{erf}(\alpha r)}{r} \tag{4.4}$$

that was proposed by Ewald [49] for the 3d-periodic setting. Hereby, we use the well known error function $\operatorname{erf}(r) := 2\pi^{-1/2} \int_0^r \mathrm{e}^{-t^2} \mathrm{d}t$ and the complementary error

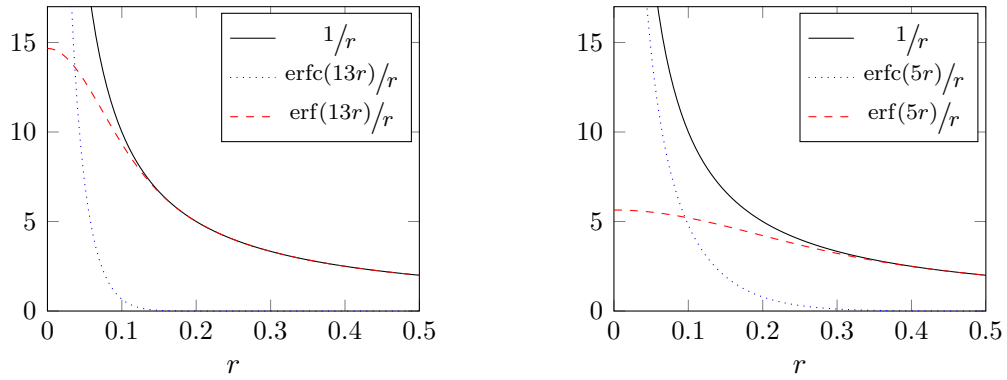

Figure 4.4: Illustration of the Ewald splitting $r^{-1} = r^{-1}\operatorname{erfc}(\alpha r) + r^{-1}\operatorname{erf}(\alpha r)$ for two different values of the splitting parameter α. For $\alpha = 13$ (left) the erfc part converges faster to zero for $r \to \infty$, while for $\alpha = 5$ the peak of the erf part decreases, i.e., it gets less high frequency components.

function $\operatorname{erfc}(r) := 1 - \operatorname{erf}(r)$. The splitting parameter $\alpha > 0$ can be used to balance the speed of convergence between the erf and erfc parts. As one can already see, for increasing α the erfc-part converges faster to zero for $r \to \infty$ while the erf-part gets closer to the original kernel function $1/r$, see also Figure 4.4 for two illustrations of the Ewald splitting at different values of α. Note that the erf-term does not have a singularity anymore. Instead it yields the finite limit

$$\lim_{r \to 0} \frac{\operatorname{erf}(\alpha r)}{r} = \frac{2\alpha}{\sqrt{\pi}}. \tag{4.5}$$

Moreover, the erf part is indefinitely often differentiable, which makes it a good candidate for Fourier approximations – as we will confirm later on.

Now, plugging the Ewald splitting (4.4) into the spherical summation (4.3) yields the decomposition $\phi_{\mathcal{S}_p} = \phi^{\mathsf{sr}}_{\mathcal{S}_p} + \phi^{\mathsf{lr}}_{\mathcal{S}_p} + \phi^{\mathsf{self}}$ with the short-range part

$$\phi^{\mathsf{sr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) := \sum_{\boldsymbol{n} \in \mathcal{S}_p} {\sum_{i=1}^{N}}' q_i \frac{\operatorname{erfc}(\alpha \|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|)}{\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|}, \tag{4.6}$$

the long-range part

$$\phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) := \sum_{t=0}^{\infty} \sum_{\substack{\boldsymbol{n} \in \mathcal{S}_p: \\ \|\boldsymbol{n}\|^2 = t}} \sum_{i=1}^{N} q_i \frac{\operatorname{erf}(\alpha \|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|)}{\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|}, \tag{4.7}$$

and the self interaction

$$\phi^{\mathsf{self}}(\boldsymbol{r}_j) := -\frac{2\alpha}{\sqrt{\pi}} q_j. \tag{4.8}$$

Thereby, we define $\operatorname{erf}(\alpha\|\mathbf{0}\|)/\|\mathbf{0}\| := 2\alpha\pi^{-1/2}$ by the limit (4.5). Note that the short-range part converges exponentially. Therefore, it does not depend on the order of summation. Analogously, we can use

$$\nabla\frac{1}{\|\boldsymbol{r}\|} = \nabla\frac{\operatorname{erfc}(\alpha\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|} + \nabla\frac{\operatorname{erf}(\alpha\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}$$

to split the field $\boldsymbol{E}_{\mathcal{S}_p}(\boldsymbol{r}_j) = \boldsymbol{E}^{\mathsf{sr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) + \boldsymbol{E}^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j)$ into the exponentially convergent, short-range part

$$\boldsymbol{E}^{\mathsf{sr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) := -\sum_{\boldsymbol{n}\in\mathcal{S}_p}\sum_{i=1}^{N}{}' q_i\nabla\frac{\operatorname{erfc}(\alpha\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|)}{\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|} \tag{4.9}$$

and the conditionally convergent, long-range part

$$\boldsymbol{E}^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) := -\sum_{t=0}^{\infty}\sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p:\\ \|\boldsymbol{n}\|^2=t}}\sum_{i=1}^{N} q_i\nabla\frac{\operatorname{erf}(\alpha\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|)}{\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|}. \tag{4.10}$$

In the next section we explain the fast computation of the fast converging short-range parts and the self interactions. The conditionally convergent, long-range parts will be subject to further investigations in Section 4.4.

4.3 The short-range and self interaction modules

Since the short-range part (4.6) converges exponentially, we can use a simple finite radius cutoff scheme for its computation. This means we choose a small short-range cutoff $r_{\mathrm{c}} > 0$ and compute the approximation

$$\phi^{\mathsf{sr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \approx \sum_{i=1}^{N} q_i \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p:\\ \|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|\le r_{\mathrm{c}}}}{}' \psi^{\mathsf{sr}}(\boldsymbol{r}^{\boldsymbol{n}}_{ij}) \tag{4.11}$$

with the short-range potential $\psi^{\mathsf{sr}}\colon \mathbb{R}^3 \to \mathbb{R}$ given by $\psi^{\mathsf{sr}}(\boldsymbol{r}) := \|\boldsymbol{r}\|^{-1}\operatorname{erfc}(\alpha\|\boldsymbol{r}\|)$. Similarly, the short-range part (4.9) of the field can be approximated by a simple radial cutoff scheme

$$\boldsymbol{E}^{\mathsf{sr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \approx -\sum_{i=1}^{N} q_i \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p:\\ \|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|\le r_{\mathrm{c}}}}{}' \nabla\psi^{\mathsf{sr}}(\boldsymbol{r}^{\boldsymbol{n}}_{ij}) \tag{4.12}$$

with the short-range field

$$\nabla\psi^{\mathsf{sr}}(\boldsymbol{r}) = -\frac{\boldsymbol{r}}{\|\boldsymbol{r}\|^2}\left(\frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2\|\boldsymbol{r}\|^2} + \frac{\operatorname{erfc}(\alpha\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}\right).$$

Module 4.1 Short-range interactions

Input:
- number of particles $N \in \mathbb{N}$
- primary box shape $\mathbf{L} \in \mathbb{R}^{3\times 3}$, $\det(\mathbf{L}) \neq 0$
- particle positions $\boldsymbol{r}_j \in \mathbf{L}\big[-1/2, 1/2\big)^3$ and sources $q_j \in \mathbb{R}$, $j = 1, \dots, N$
- number of periodic dimensions p
- short-range cutoff $r_\mathrm{c} > 0$
- short-range potential $\psi^{\mathsf{sr}} \colon \mathbb{R}^3 \to \mathbb{R}$

Assumptions: homogenous particle distribution

1: For all particle positions $\boldsymbol{r}_j$ compute

$$\phi_j \leftarrow \sum_{i=1}^{N} q_i \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p:\\ \|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\|\le r_\mathrm{c}}}{}' \psi^{\mathsf{sr}}(\boldsymbol{r}_{ij}^{\boldsymbol{n}}), \qquad \boldsymbol{E}_j \leftarrow -\sum_{i=1}^{N} q_i \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p:\\ \|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\|\le r_\mathrm{c}}}{}' \nabla\psi^{\mathsf{sr}}(\boldsymbol{r}_{ij}^{\boldsymbol{n}}),$$

by a linked cell algorithm; see [70, Chap. 8.4] or [63, Chap. 3].

Output:
- approximated short-range potential $\phi_j \approx \phi_{\mathcal{S}_p}^{\mathsf{sr}}(\boldsymbol{r}_j) \in \mathbb{R}$, $j = 1, \dots, N$
- approximated short-range field $\boldsymbol{E}_j \approx \boldsymbol{E}_{\mathcal{S}_p}^{\mathsf{sr}}(\boldsymbol{r}_j) \in \mathbb{R}^3$, $j = 1, \dots, N$

Arithmetic cost: $\mathcal{O}(N)$

If we assume a sufficiently homogenous particle distribution, the number of terms in the short-range approximations (4.11) and (4.12) will be bounded by a constant. Therefore, we can compute these approximations for all particle positions $\boldsymbol{r}_j$ within $\mathcal{O}(N)$ arithmetic operations. However, a naive search for all neighboring particles would still require the computation of all particle-to-particle distances and is of order $\mathcal{O}(N^2)$. Instead we use a linked cell algorithm to perform the search for next neighboring particles within $\mathcal{O}(N)$ operations; see [70, Chap. 8.4] or [63, Chap. 3]. We summarize the computation of the short-range parts in the algorithmic Module 4.1.

The computation of the short-range module 4.1 requires the repetitive evaluation of the short-range potential ψ^{sr} and its derivative $\frac{\mathrm{d}}{\mathrm{d}r}\psi^{\mathsf{sr}}(r)$ for $0 \le r \le r_\mathrm{c}$. Since these two functions are smooth, we can use interpolation from precomputed look-up tables to speed up their evaluation. Once the look-up tables are generated, the evaluations of the short-range interaction become independent from the precise choice of short-range potential ψ^{sr}. This offers an easy way to interchange the short-range potential ψ^{sr} without effecting performance. Note that the same approach was used for the implementation of NFFT window functions; cf. Section 3.2.1. In our

Module 4.2 Self interactions

Input:

- number of particles $N \in \mathbb{N}$
- sources $q_j \in \mathbb{R}$, $j = 1, \ldots, N$
- self potential $\psi^{\mathsf{self}} \in \mathbb{R}$

1: For all $j = 1, \ldots, N$ compute

$$\phi_j^{\mathsf{self}} \leftarrow \psi^{\mathsf{self}} q_j.$$

Output:

- self potential $\phi_j^{\mathsf{self}} = \phi^{\mathsf{self}}(\boldsymbol{r}_j) \in \mathbb{R}$, $j = 1, \ldots, N$

Arithmetic cost: $\mathcal{O}(N)$

implementation the user is free to chose between direct evaluation and interpolation of degree 0, 1, 2, or 3.

As we can easily see, all self interactions (4.8) can be computed directly within the optimal complexity of $\mathcal{O}(N)$ operations. In order to define a more general algorithmic module we rewrite (4.8) as

$$\phi^{\mathsf{self}}(\boldsymbol{r}_j) = \psi^{\mathsf{self}} q_j,$$

with $\psi^{\mathsf{self}} := -2\alpha\pi^{-1/2}$. Then, Module 4.2 summarizes the computation of the self interactions for all particle positions $\boldsymbol{r}_j$.

4.4 The long-range interaction module

In this section we present different approximation ideas that lead to fast algorithms for computing the long-range potential (4.7). Obviously, the number p of periodic dimensions will have a great impact on the precise definition of the approximations. However, we will see that it is possible to summarize the most important ideas within a common approximation framework. In order to emphasize this fact, we start with the presentation of this framework and show that it yields a fast algorithm. Afterward, we give the precise definition of the approximations for all kinds of mixed periodic boundary conditions and show that they fit into the common framework.

4.4.1 The common approximation framework and fast algorithm

We call a matrix $\mathbf{H} \in \mathbb{R}^{3\times 3}$ an extended box shape matrix if it fulfills $\det(\mathbf{H}) \neq 0$ and $\mathbf{L}\big[-1/2, 1/2\big)^3 \subset \mathbf{H}\big[-1/2, 1/2\big)^3$, i.e., the columns of $\mathbf{H}$ span a box that includes

the primary box $\mathbf{L}$. Assume that we can find such an extended box shape matrix $\mathbf{H} \in \mathbb{R}^{3\times 3}$, a finite mesh size $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$ and Fourier coefficients $\hat{R}_{\boldsymbol{k}} \in \mathbb{C}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$, such that

$$\phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) = \sum_{t=0}^{\infty} \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p:\\ \|\boldsymbol{n}\|^2=t}} \sum_{i=1}^{N} q_i \frac{\operatorname{erf}(\alpha\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|)}{\|\boldsymbol{r}^{\boldsymbol{n}}_{ij}\|} \approx \sum_{i=1}^{N} q_i \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}_{ij}} \tag{4.13}$$

yields a sufficiently small approximation error. Then, a simple change in the order of summation yields

$$\phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \approx \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \left(\sum_{i=1}^{N} q_i \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}_i} \right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}_j}. \tag{4.14}$$

The expression in the inner brackets can be approximated by a three-dimensional NFFT$^{\mathsf{H}}$ with nodes $\boldsymbol{x}_i := \mathbf{H}^{-1}\boldsymbol{r}_i \in \left[-1/2, 1/2\right)^3$ and total number of frequencies $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$. Afterward, $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ point-wise multiplications with the Fourier coefficients $\hat{R}_{\boldsymbol{k}}$ are performed. Finally, a three-dimensional NFFT with nodes $\boldsymbol{x}_j := \mathbf{H}^{-1}\boldsymbol{r}_j \in \left[-1/2, 1/2\right)^3$ and total number of frequencies $|\mathcal{I}_{\hat{\boldsymbol{m}}}|$ gives an approximation of the outer sum. If c_φ is the NFFT window cutoff parameter, then the proposed evaluation at all nodes $\boldsymbol{r}_j$, $j = 1, \ldots, N$, requires $\mathcal{O}(c_\varphi^3 N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ arithmetic operations.

Analogously, (4.13) yields an fast way to approximate the long-range part (4.10) of the field as

$$\boldsymbol{E}^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) = -\nabla\phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \approx -\sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \left(\sum_{i=1}^{N} q_i \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}_i} \right) \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}_j}.$$

Thereby, the only difference to (4.14) is the outer vector sum that can be computed by a gradient NFFT with nodes $\boldsymbol{x}_j := \mathbf{H}^{-1}\boldsymbol{r}_j \in \left[-1/2, 1/2\right)^3$. Note that the substitution $\boldsymbol{x}_j = \mathbf{H}^{-1}\boldsymbol{r}_j$ results in a rescaling of the gradient due to the formula

$$\nabla_{\boldsymbol{r}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}} = \mathbf{H}^{-\mathsf{T}} \nabla_{\boldsymbol{x}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}}.$$

Module 4.3 summarizes the fast algorithm for computing the long-range potential and field that result from approximation (4.13). In the following, we show the precise parameter settings of the common approximation framework for the different kinds of mixed periodic boundary conditions.

4.4.2 Periodicity in three dimensions

The limit of the conditionally convergent, long-range part with 3d-periodic boundary conditions was given in [49]. A rigorous and elegant proof can be given using convergence factors, see [89]. Furthermore, we point on [72] were most of the notational

Module 4.3 Long-range interactions

Input:

- number of particles $N \in \mathbb{N}$
- primary box shape $\mathbf{L} \in \mathbb{R}^{3\times 3}$, $\det(\mathbf{L}) \neq 0$
- particle positions $\boldsymbol{r}_j \in \mathbf{L}\big[-1/2, 1/2\big)^3$ and sources $q_j \in \mathbb{R}$, $j = 1, \ldots, N$
- mesh size $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$
- Fourier coefficients $\hat{R}_{\boldsymbol{k}} \in \mathbb{C}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$
- extended box shape $\mathbf{H} \in \mathbb{R}^{3\times 3}$, $\det(\mathbf{H}) \neq 0$

1: For all particle positions $\boldsymbol{r}_j$ compute the rescaled coordinates

$$\boldsymbol{x}_j \leftarrow \mathbf{H}^{-1}\boldsymbol{r}_j \in \big[-1/2, 1/2\big)^3.$$

2: For all mesh points $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$ compute

$$\hat{S}_{\boldsymbol{k}} \leftarrow \sum_{i=1}^{N} q_i \mathrm{e}^{+2\pi \mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_i}$$

by a three-dimensional NFFT$^{\mathsf{H}}$; cf. (3.23).

3: For all mesh points $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$ compute the point-wise products

$$\hat{a}_{\boldsymbol{k}} \leftarrow \hat{R}_{\boldsymbol{k}}\hat{S}_{\boldsymbol{k}}.$$

4: For all rescaled particle positions $\boldsymbol{x}_j$ compute

$$\phi_j \leftarrow \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{a}_{\boldsymbol{k}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j}, \qquad \boldsymbol{E}_j \leftarrow -\mathbf{H}^{-\mathsf{T}} \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{a}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j},$$

by a three-dimensional (gradient) NFFT; cf. (3.20), (3.24), (3.26).

Output:

- approximated long-range potential $\phi_j \approx \phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \in \mathbb{R}$, $j = 1, \ldots, N$
- approximated long-range field $\boldsymbol{E}_j \approx \boldsymbol{E}^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \in \mathbb{R}^3$, $j = 1, \ldots, N$

Arithmetic cost: $\mathcal{O}\left(N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|\right)$

convenience comes from. For an arbitrary tricline box shape $\mathbf{L} \in \mathbb{R}^{3\times3}, \det(\mathbf{L}) \neq 0$, the long-range part becomes

$$\phi^{\mathsf{lr}}_{\mathcal{S}_3}(\boldsymbol{r}_j) = \frac{1}{|\det(\mathbf{L})|} \sum_{i=1}^{N} q_i \sum_{\boldsymbol{k}\in\mathbb{Z}^3} \hat{R}_{\boldsymbol{k}} \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^T\mathbf{L}^{-1}\boldsymbol{r}_{ij}}$$

with $\hat{R}_{\mathbf{0}} := 0$ and

$$\hat{R}_{\boldsymbol{k}} := \frac{1}{\pi\|\mathbf{L}^{-\mathsf{T}}\boldsymbol{k}\|^2} \mathrm{e}^{-\pi^2\|\mathbf{L}^{-\mathsf{T}}\boldsymbol{k}\|^2/\alpha^2}, \quad \boldsymbol{k} \neq \mathbf{0}. \tag{4.15}$$

Because of the exponential decay of $\hat{R}_{\boldsymbol{k}}$ it is nearby to choose a finite mesh size $\hat{\boldsymbol{m}} \in 2\mathbb{N}^2$ and truncate the infinite sum as

$$\phi^{\mathsf{lr}}_{\mathcal{S}_3}(\boldsymbol{r}_j) \approx \frac{1}{|\det(\mathbf{L})|} \sum_{i=1}^{N} q_i \sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^T\mathbf{L}^{-1}\boldsymbol{r}_{ij}}.$$

If we set $\mathbf{H} \leftarrow \mathbf{L}$, this approximation is exactly of the form (4.11). Therefore, we can use Module 4.3 for its fast evaluation with $\mathcal{O}(N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ operations.

4.4.3 Periodicity in two dimensions

In the following, we briefly present the approximation ideas of [3] that lead to a fast algorithm for 2d-periodic boundary conditions. Note that the definition of our problem (4.1) used the more general case of a tricline box shape $\mathbf{L}$. However, without loss of generality we may assume a box shape of type

$$\mathbf{L} = \begin{pmatrix} L_{00} & L_{01} & 0 \\ L_{10} & L_{11} & 0 \\ 0 & 0 & L_{22} \end{pmatrix} \in \mathbb{R}^{3\times3}, \quad \det(\mathbf{L}) \neq 0, \quad L_{22} > 0.$$

This means, the two periodic dimensions can always be rotated into the x-y-plane and we have the freedom to choose the non-periodic dimension along the z-axis. For the case of an orthogonal box shape $\mathbf{L}$ ($L_{01} = L_{10} = 0$) the limit of the conditionally convergent, long-range part with 2d-periodic boundary conditions can be found in [65]. We gave a rigorous proof of the corresponding formulas using convergence factors in [3]. A generalization to arbitrary L_{01}, L_{10} is straightforward and yields

$$\phi^{\mathsf{lr}}_{\mathcal{S}_2}(\boldsymbol{r}_j) = \frac{1}{|L_{00}L_{11} - L_{01}L_{10}|} \sum_{\boldsymbol{k}\in\mathcal{S}_2} \sum_{i=1}^{N} q_i \Theta_{\mathcal{S}_2}\left(\|\mathbf{L}^{-\mathsf{T}}\boldsymbol{k}\|, z_{ij}\right) \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{L}^{-1}\boldsymbol{r}_{ij}}, \tag{4.16}$$

where z_{ij} is the last coordinate of $\boldsymbol{r}_{ij} \in 2\mathbf{L}\left[-1/2, 1/2\right)^3$. Thereby, $\Theta_{\mathcal{S}_2} \colon [0,+\infty)^2 \to \mathbb{R}$ is defined for the $k = 0$ term by

$$\Theta_{\mathcal{S}_2}(0,r) := -2\sqrt{\pi}\left[\frac{1}{\alpha}\mathrm{e}^{-\alpha^2r^2} + \sqrt{\pi}r\,\mathrm{erf}(\alpha r)\right]$$

and otherwise as

$$\Theta_{\mathcal{S}_2}(k,r) := \frac{1}{2k}\left[\mathrm{e}^{+2\pi kr}\operatorname{erfc}\left(\frac{\pi k}{\alpha}+\alpha r\right)+\mathrm{e}^{-2\pi kr}\operatorname{erfc}\left(\frac{\pi k}{\alpha}-\alpha r\right)\right].$$

The main problem of (4.16) is its dependency on $z_{ij} = z_i - z_j$ within $\Theta_{\mathcal{S}_2}$. Computing all of these distances for all particle pairs would require $\mathcal{O}(N^2)$ operations. Therefore, we introduce another approximation in the flavor of fast summation algorithms [112, 113]. Let $H > 2L_{22}$. For any $(k_0,k_1)^{\mathsf{T}} \in \mathbb{Z}^2$ we introduce a regularization $R_{k_0,k_1}\colon \mathbb{R}\to\mathbb{R}$ by the H-periodic continuation of

$$R_{k_0,k_1}(z) := \begin{cases} \Theta_{\mathcal{S}_2}\left(\left\|\left(\begin{smallmatrix} L_{00} & L_{01} \\ L_{10} & L_{11}\end{smallmatrix}\right)^{-\mathsf{T}}\left(\begin{smallmatrix} k_0 \\ k_1\end{smallmatrix}\right)\right\|, r\right) & \text{if } |z|\le L_{22}, \\ T_{k_0,k_1}(r) & \text{if } L_{22} < |z| \le \frac{H}{2}. \end{cases} \tag{4.17}$$

The transition T_{k_0,k_1} is chosen such that R_{k_0,k_1} is s times differentiable for an appropriately chosen degree of smoothness $s\in\mathbb{N}$. One can think of several transition function types, e.g., algebraic polynomials, splines, trigonometric polynomials, or two point Taylor interpolation; see [52]. Here, we construct T as the unique polynomial of degree $2s+1$ that fulfills the $2s+2$ interpolation conditions

$$T^{(j)}_{k_0,k_1}(L_{22}) = (-1)^j T^{(j)}_{k_0,k_1}(H-L_{22}) = \Theta^{(j)}_{\mathcal{S}_2}\left(\left\|\left(\begin{smallmatrix} L_{00} & L_{01} \\ L_{10} & L_{11}\end{smallmatrix}\right)^{-\mathsf{T}}\left(\begin{smallmatrix} k_0 \\ k_1\end{smallmatrix}\right)\right\|, L_{22}\right),$$

for $j=0,\dots,s$. The construction and evaluation of such interpolating polynomials T_{k_0,k_1} can be performed with a recursive scheme of divided differences [122, Section 2.1] and requires $\mathcal{O}(s^2)$ operations per evaluation, see also Section 4.4.6 for details. Since R_{k_0,k_1} is periodic and smooth of order s, we expect a good approximation by the finite Fourier series

$$R_{k_0,k_1}(z) \approx \frac{1}{H}\sum_{k_2\in\mathcal{I}_{\hat{m}_2}} \hat{R}_{k_0,k_1,k_2}\mathrm{e}^{+2\pi\mathrm{i}k_2 z/H}, \tag{4.18}$$

where $\hat{m}_2\in 2\mathbb{N}$ is an appropriately chosen finite mesh size and

$$\hat{R}_{k_0,k_1,k_2} := \frac{H}{\hat{m}_2}\sum_{l_2\in\mathcal{I}_{\hat{m}_2}} R_{k_0,k_1}\left(H\frac{l_2}{\hat{m}_2}\right)\mathrm{e}^{-2\pi\mathrm{i}k_2 l_2/\hat{m}_2}, \quad k_2\in\mathcal{I}_{\hat{m}_2},$$

are the discrete Fourier coefficients that can be precomputed by a one-dimensional FFT.

Note that $\Theta_{\mathcal{S}_2}(k,z)$ decays exponentially with $|k|\to\infty$. Therefore, we may truncate the infinite sum in (4.16) and plug in the approximations (4.18). Then, we get

$$\phi^{\mathsf{lr}}_{\mathcal{S}_2}(\boldsymbol{r}_j) \approx \frac{1}{\det(\mathbf{H})}\sum_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}}\sum_{i=1}^{N} q_i \hat{R}_{k_0,k_1,k_2}\mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\mathbf{H}^{-1}\boldsymbol{r}_{ij}},$$

with the extended box shape $\mathbf{H} \in \mathbb{R}^{3\times3}$ given by

$$\mathbf{H} \leftarrow \begin{pmatrix} L_{00} & L_{01} & 0 \\ L_{10} & L_{11} & 0 \\ 0 & 0 & H \end{pmatrix} \in \mathbb{R}^{3\times3}.$$

Again, this approximation is exactly of the form (4.11). Therefore, we can use Module 4.3 for its fast evaluation with $\mathcal{O}(N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ operations.

4.4.4 Periodicity in one dimension

Without loss of generality we assume an orthogonal box shape $\mathbf{L} = \operatorname{diag}(L_0, L_2, L_2)$, $L_0, L_2 > 0$. This means, the periodic dimension can be rotated onto the x-axis and we are free to chose the box vectors of both non-periodic dimensions orthogonal and of equal length L_2. A rigorous proof for the limit of the conditionally convergent, long-range part with 1d-periodic boundary conditions is given in [3]. Therein, the 1d-periodic long-range parts are presented as

$$\phi^{\mathsf{lr}}_{\mathcal{S}_1}(\boldsymbol{r}_j) = \frac{1}{L_0} \sum_{\boldsymbol{k}\in\mathcal{S}_1} \sum_{i=1}^{N} q_i \Theta_{\mathcal{S}_1}\left(\frac{k_0}{L_0}, \sqrt{y_{ij}^2 + z_{ij}^2}\right) \mathrm{e}^{+2\pi\mathrm{i}k_0 x_{ij}/L_0}, \tag{4.19}$$

where $\boldsymbol{r}_{ij} = (x_{ij}, y_{ij}, z_{ij})^{\mathsf{T}} \in 2\mathbf{L}\big[-1/2, 1/2\big)^3$. Furthermore, $\Theta_{\mathcal{S}_1} \colon [0, +\infty)^2 \to \mathbb{R}$ is defined for the $k = 0$ term by

$$\Theta_{\mathcal{S}_1}(0, r) := -\frac{1}{L_0}\left[\gamma + \Gamma(0, \alpha^2 r^2) + \ln(\alpha^2 r^2)\right],$$

and otherwise as

$$\Theta_{\mathcal{S}_1}(k, r) := \frac{1}{L_0} K_0\left(\tfrac{\pi^2 k_x^2}{\alpha^2 L_0^2}, \alpha^2 r^2\right).$$

Thereby, $\gamma \approx 0.5772\ldots$ is the Euler-Mascheroni constant, $\Gamma(s, x) := \int_x^\infty t^{s-1}\mathrm{e}^{-t}\mathrm{d}t$ is the upper incomplete gamma function and

$$K_\nu(x, y) := \int_1^\infty t^{-\nu-1}\mathrm{e}^{-xt-y/t}\mathrm{d}t, \quad \nu \in \mathbb{R}, x \geq 0, y \geq 0,$$

denotes the ν-th order incomplete modified Bessel function of the second kind [67].

Obviously, direct evaluation of all the distances y_{ij}, z_{ij} within (4.19) would require $\mathcal{O}(N^2)$ operations. Again, we introduce an approximation in the flavor of fast summation algorithms [112, 113]. Let $H > 2\sqrt{2}L_2$. For any $k_0 \in \mathbb{Z}$ we introduce a regularization $R_{k_0} \colon \mathbb{R}^2 \to \mathbb{R}$ by the two-dimensional H-periodic continuation of

$$R_{k_0}(y, z) := \begin{cases} \Theta_{\mathcal{S}_1}\left(\frac{k_0}{L_0}, \sqrt{y^2+z^2}\right) & \text{if } \sqrt{y^2+z^2} \leq \sqrt{2}L_2, \\ T_{k_0}(\sqrt{y^2+z^2}) & \text{if } \sqrt{2}L_2 < \sqrt{y^2+z^2} \leq \frac{H}{2}, \\ T_{k_0}(\frac{H}{2}) & \text{if } \frac{H}{2} < \sqrt{y^2+z^2}. \end{cases} \tag{4.20}$$

Again, the transition T_{k_0} is chosen such that $R(k_0, \cdot, \cdot)$ is s times differentiable for an appropriately chosen degree of smoothness $s \in \mathbb{N}$. This time we construct T_{k_0} as the unique polynomial of degree $2s$ that fulfills the $2s+1$ interpolation conditions

$$\begin{aligned} T_{k_0}^{(j)}\left(\sqrt{2}L\right) &= \Theta_{\mathcal{S}_1}^{(j)}\left(\tfrac{k_0}{L_0}, \sqrt{2}L\right), \quad j = 0, \ldots, s, \\ T_{k_0}^{(j)}\left(\tfrac{H}{2}\right) &= 0, \quad j = 1, \ldots, s. \end{aligned}$$

Note that the classical recursive scheme of divided differences [122, Section 2.1] is no more applicable, since the function value $T_{k_0}(H/2)$ is not part of the interpolation conditions. However, in Section 4.4.6 we present an adapted scheme for the construction and evaluation of T_{k_0} that preserves the $\mathcal{O}(s^2)$ complexity per evaluation. Note that the construction by Lagrangian basis polynomials given in [3] requires $\mathcal{O}(s^3)$ operations. Since R_{k_0} is periodic and smooth of order s, we expect a good approximation by the finite Fourier series

$$R_{k_0}(y, z) \approx \frac{1}{H^2} \sum_{k_1 \in \mathcal{I}_{\hat{m}_1}} \sum_{k_2 \in \mathcal{I}_{\hat{m}_2}} \hat{R}_{k_0,k_1,k_2} \mathrm{e}^{+2\pi \mathrm{i} k_1 y/H} \mathrm{e}^{+2\pi \mathrm{i} k_2 z/H}, \tag{4.21}$$

where $\hat{m}_1, \hat{m}_2 \in 2\mathbb{N}$ are appropriately chosen finite mesh sizes and for $(k_1, k_2)^\mathsf{T} \in \mathcal{I}_{\hat{m}_1} \times \mathcal{I}_{\hat{m}_2}$ the discrete Fourier coefficients

$$\hat{R}_{k_0,k_1,k_2} := \frac{H^2}{\hat{m}_1 \hat{m}_2} \sum_{l_1 \in \mathcal{I}_{\hat{m}_1}} \sum_{l_2 \in \mathcal{I}_{\hat{m}_2}} R_{k_0}\left(H\frac{l_1}{\hat{m}_1}, H\frac{l_2}{\hat{m}_2}\right) \mathrm{e}^{-2\pi \mathrm{i} k_1 l_1/\hat{m}_1} \mathrm{e}^{-2\pi \mathrm{i} k_2 l_2/\hat{m}_2},$$

can be precomputed by a two-dimensional FFT. Note that function $\Theta_{\mathcal{S}_1}$ asymptotically tends to zero as $k^{-2}\mathrm{e}^{-k^2}$ for $k \to \infty$, which justifies truncation of the Fourier series (4.19) at a finite mesh size $\hat{m}_0 \in 2\mathbb{N}$ along the periodic dimension, cf. [3]. Then, plugging in approximation (4.21) yields

$$\phi_{\mathcal{S}_1}^{\mathsf{lr}}(\boldsymbol{r}_j) \approx \frac{1}{\det(\mathbf{H})} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \sum_{i=1}^{N} q_i \hat{R}_{k_0,k_1,k_2} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^\mathsf{T} \mathbf{H}^{-1} \boldsymbol{r}_{ij}},$$

with the extended box shape $\mathbf{H} \leftarrow \mathrm{diag}(L_0, H, H) \in \mathbb{R}^{3\times 3}$. Again, this approximation is exactly of the form (4.11). Therefore, we can use Module 4.3 for its fast evaluation with $\mathcal{O}(N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ operations.

4.4.5 Periodicity in no dimension

Finally, we give the analogous approximation for the Ewald splitting with open boundary conditions, i.e., the 0d-periodic case in (4.4). Without loss of generality

we assume a cubic box shape $\mathbf{L} = \operatorname{diag}(L, L, L)$, $L \neq 0$. In the 0d-periodic case the long-range potential (4.7) reads as

$$\phi^{\mathsf{lr}}_{\mathcal{S}_0}(\boldsymbol{r}_j) = \sum_{i=1}^{N} q_i \Theta_{\mathcal{S}_0}(\|\boldsymbol{r}_{ij}\|). \tag{4.22}$$

where $\boldsymbol{r}_{ij} := \boldsymbol{r}_i - \boldsymbol{r}_j \in 2\mathbf{L}\big[-1/2, 1/2\big)^3$ and $\Theta_{\mathcal{S}_0}(t) := t^{-1}\operatorname{erf}(\alpha t)$. Assume $H > 2\sqrt{3}L$. A suitable smooth regularization $R\colon \mathbb{R}^3 \to \mathbb{R}$ is constructed by by the three-dimensional H-periodic continuation of

$$R(\boldsymbol{r}) := \begin{cases} \Theta_{\mathcal{S}_0}(\|\boldsymbol{r}\|) & \text{if } \|\boldsymbol{r}\| \leq \sqrt{3}L, \\ T(\|\boldsymbol{r}\|) & \text{if } \sqrt{3}L < \|\boldsymbol{r}\| \leq \frac{H}{2}, \\ T(\frac{H}{2}) & \text{if } \frac{H}{2} < \|\boldsymbol{r}\|. \end{cases} \tag{4.23}$$

The transition T is chosen such that R is s times differentiable for an appropriately chosen degree of smoothness $s \in \mathbb{N}$. Analogously to the 1d-periodic case we construct T as the unique algebraic polynomial of degree $2s$ that fulfills the $2s+1$ interpolation conditions

$$\begin{aligned} T^{(j)}\left(\sqrt{3}L\right) &= \Theta^{(j)}_{\mathcal{S}_0}\left(\sqrt{3}L\right), \quad j = 0, \ldots, s, \\ T^{(j)}\left(\tfrac{H}{2}\right) &= 0, \quad j = 1, \ldots, s. \end{aligned}$$

Evaluation of T is performed with the adapted scheme of divided differences presented in Section 4.4.6 an takes $\mathcal{O}(s^2)$ operations per evaluation. Since R is periodic and smooth of order s we expect a good approximation by the finite Fourier series

$$R(\boldsymbol{r}) \approx \frac{1}{H^3} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{r}/H}, \tag{4.24}$$

where $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$ is an appropriately chosen finite mesh size and

$$\hat{R}_{\boldsymbol{k}} := \frac{H^3}{|\mathcal{I}_{\hat{\boldsymbol{m}}}|} \sum_{\boldsymbol{l} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} R(H\boldsymbol{l} \odot \hat{\boldsymbol{m}}^{-1}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \hat{\boldsymbol{m}}^{-1})}, \quad \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}},$$

are the discrete Fourier coefficients that can be precomputed by a three-dimensional FFT. Again, plugging approximation (4.24) into (4.22) yields an approximation of the form (4.13) with $\mathbf{H} := \operatorname{diag}(H, H, H)$. Therefore, we can use Module 4.3 for its fast evaluation with $\mathcal{O}(N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ operations.

4.4.6 Two-point Taylor interpolation using Newton basis polynomials

We have seen that the construction of smooth regularizations for 1d- and 0d-periodic boundary conditions requires the evaluation of a special class of interpolating polynomials. In the following, we present an adapted scheme of divided differences that

realizes the evaluation in an efficient way. More precisely, we want to construct an algebraic polynomial $Q\colon \mathbb{R} \to \mathbb{R}$ of degree $2s$ that fulfills the $2s+1$ Hermite-Birkhoff interpolation conditions

$$Q(x_0) = a_0 \quad \text{and} \quad Q^{(j)}(x_0) = a_j, \quad Q^{(j)}(x_1) = b_j \quad \text{for} \quad j = 1, \dots, s. \tag{4.25}$$

Thereby, $a_j, b_j \in \mathbb{R}$ are arbitrary given numbers. In [3, Appendix C] we presented a representation of Q based on integrated Lagrangian basis polynomials. However, a direct evaluation of this construction requires $\mathcal{O}(s^3)$ operations. This stays in contrast to the $\mathcal{O}(s^2)$ operations that would be sufficient if the function value $Q(x_1)$ was interpolated too, see [17, Corollary 2.2.6] or [3, Appendix C]. In the following, we present an alternative construction based on Newton basis polynomials that enables us to construct the coefficients of Q with $\mathcal{O}(s^2)$ operations. At first, we assume that Q is given in terms of a Newton basis as

$$Q(x) = \sum_{k=0}^{s+1} q_k (x - x_0)^k + (x - x_0)^{s+1} \sum_{k=s+2}^{2s} q_k (x - x_1)^{k-s-1}.$$

Furthermore, we consider the unique polynomial

$$P(x) = \sum_{k=0}^{s} p_k (x - x_0)^k + (x - x_0)^s \sum_{k=s+1}^{2s-1} p_k (x - x_1)^{k-s}$$

of degree $2s-1$ satisfying the Hermite interpolation conditions $P^{(j)}(x_0) = a_{j+1}$ and $P^{(j)}(x_1) = b_{j+1}$ for all $j = 0, \dots, s-1$. It is well known that the coefficients p_k can be computed with a recursive scheme of divided differences within $\mathcal{O}(s^2)$ operations, see [122, Section 2.1] for details. Therefore, the ansatz $P(x) = Q'(x)$ may yield all coefficients q_k, $k = 1, \dots, 2s$. Together with $q_0 := a_0$ this will solve the interpolation problem (4.25).

The derivative of Q can be written as $Q'(x) = Q_1'(x) + Q_2'(x)$ with

$$\begin{aligned} Q_1'(x) &:= \sum_{k=1}^{s+1} k q_k (x - x_0)^{k-1} \\ &= \sum_{k=1}^{s} k q_k (x - x_0)^{k-1} + (x - x_0)^s \sum_{k=s+1}^{s+1} k q_k (x - x_1)^{k-s-1}, \\ Q_2'(x) &:= \sum_{k=s+2}^{2s} q_k \frac{\mathrm{d}}{\mathrm{d}x} (x - x_0)^{s+1} (x - x_1)^{k-s-1}. \end{aligned}$$

For the second part we apply the identity

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}x} (x - x_0)^{s+1} (x - x_1)^{k-s-1} \\ &= (x - x_0)^s \left[k (x - x_1)^{k-s-1} + (k - s - 1)(x_1 - x_0)(x - x_1)^{k-s-2} \right] \end{aligned}$$

and obtain

$$Q_2'(x) = 2sq_{2s}(x-x_0)^s(x-x_1)^{s-1} + (x-x_0)^s \sum_{k=s+2}^{2s-1} kq_k(x-x_1)^{k-s-1} + (x-x_0)^s \sum_{k=s+1}^{2s-1} q_{k+1}(k-s)(x_1-x_0)(x-x_1)^{k-s-1}.$$

Thereby, we extracted the term for $k = 2s$ out of the first sum and shifted the index $k \to k-1$ in the second sum. Now, we can add Q_1' and yield

$$Q'(x) = \sum_{k=1}^{s} kq_k(x-x_0)^k + (x-x_0)^s \sum_{k=s+1}^{2s-1} \left[kq_k + (k-s)(x_1-x_0)q_{k+1}\right](x-x_1)^{k-s-1} + 2sq_{2s}(x-x_0)^s(x-x_1)^{s-1}.$$

Finally, comparison of the coefficients between $P(x)$ and $Q'(x)$ yields the recursion

$$\begin{aligned} q_{2s} &:= \frac{1}{2s}p_{2s-1}, \\ q_k &:= \frac{1}{k}\left[p_{k-1} - (k-s)(x_1-x_0)q_{k+1}\right], && k = 2s-1, \ldots, s+1, \\ q_k &:= \frac{1}{k}p_{k-1}, && k = s, \ldots, 1, \\ q_0 &:= a_0. \end{aligned}$$

Note that this recursion can be computed with $\mathcal{O}(s)$ operations. Together with the complexity of the divided differences scheme for computing the coefficients p_k we end up with $\mathcal{O}(s^2)$ operation for computing all coefficients q_k. Afterward, the evaluation of $Q(x)$ can be performed in $\mathcal{O}(s)$ operations with a Horner scheme.

4.4.7 Some notes on Fourier approximations of non-periodic functions

The long-range interaction Module 4.3 incorporates non-periodic boundary conditions in the Ewald long-range part using a regularization technique that results in fast convergent Fourier approximations; cf. (4.17), (4.20), (4.23). However, a lot of other fast $\mathcal{O}(N \log N)$ algorithms for computing the Ewald sum with non-periodic boundary conditions have been proposed earlier. For example, an algorithm for computing the non-periodic Ewald sum has already been proposed in [70]. Later on, 0d-periodic [98], 1d-periodic [97], and 2d-periodic [99] boundary conditions have been handled within one common framework [126]. More recently, a spectrally convergent approach [93] for the 2d-periodic Ewald sum was published. Essentially,

the above mentioned algorithms only differ in the approach for constructing Fourier approximations of non-periodic functions. Thereby, the applied approximation approaches fall into three categories that will be compared in the following. For the sake on convenience we restrict the discussion to non-periodic functions $\Theta\colon \mathbb{R} \to \mathbb{R}$ of one variable. Then, we can choose one of the following Variants I-III in order to construct a Fourier approximation on the interval $[-L, L]$.

Variant I (Truncation):

We take a sufficiently large cutoff $H \geq 2L$ and approximate the function Θ on the interval $[-H/2, H/2]$ by a finite Fourier series $\Theta(x) \approx \sum_{k=M/2}^{M/2-1} c_k \mathrm{e}^{-2\pi \mathrm{i} kx/h}$, where we compute the coefficients c_k by

$$c_k := \frac{1}{H} \int_{-H/2}^{H/2} \Theta(x) \mathrm{e}^{+2\pi \mathrm{i} lx/H} \mathrm{d}x.$$

This means, we compute the Fourier coefficients of a H-periodic function that is equal to Θ on the interval $[-H/2, H/2]$, see Figure 4.5 for an illustration. Note that the approximated H-periodic function is only smooth of order zero in $x = H/2$, which results in a rather slow second order convergence in Fourier space. Thus, one may have to choose a very large mesh size M in order to achieve a good approximation.

This approach was used in [99] for the 2d-periodic Ewald sum. There, the coefficients c_k are known analytically; cf. equation (2.9) in [99]. A generalization to non-periodic functions Θ of two or three variables is straightforward and yields the 0d- and 1d-periodic algorithms published in [98, 126, 97].

Variant II (Periodization):

The continuous Fourier transform of a function $\Theta \in L^1(\mathbb{R})$ is given by

$$\hat{\Theta}(\xi) = \int_{\mathbb{R}} \Theta(x) \mathrm{e}^{+2\pi \mathrm{i} x\xi} \mathrm{d}x.$$

If Θ is sufficiently small outside the interval $[-L, L]$, we may approximate Θ by its H-periodic version $\sum_{n\in\mathbb{Z}} \Theta(\cdot + Hn)$, where $H \geq 2L$, apply the Poisson summation formula and truncate the resulting infinite sum in order to obtain an approximation of the form

$$\begin{aligned} \Theta(x) &\approx \sum_{n=-\infty}^{\infty} \Theta(x + Hn) = \tfrac{1}{H} \sum_{k=-\infty}^{\infty} \hat{\Theta}(\tfrac{k}{H}) \mathrm{e}^{-2\pi \mathrm{i} kx/H} \\ &\approx \tfrac{1}{H} \sum_{k=-M/2}^{M/2-1} \hat{\Theta}(\tfrac{k}{H}) \mathrm{e}^{-2\pi \mathrm{i} kr/H}, \end{aligned} \tag{4.26}$$

where the mesh size $M \in 2\mathbb{N}$ has to be chosen sufficiently large. Alternatively, some authors argue as follows. First, we truncate the Fourier integral and, second, we approximate the resulting finite integral via the trapezoidal quadrature rule

$$\begin{aligned}\Theta(x) &= \int_{\mathbb{R}} \hat{\Theta}(\xi) \mathrm{e}^{-2\pi \mathrm{i} x\xi} \mathrm{d}\xi \approx \int_{-A/2}^{A/2} \hat{\Theta}(\xi) \mathrm{e}^{-2\pi \mathrm{i} x\xi} \mathrm{d}\xi \\ &\approx \frac{A}{M} \sum_{k=-M/2}^{M/2-1} \hat{\Theta}(\tfrac{kA}{M}) \mathrm{e}^{-2\pi \mathrm{i} x k A/M}. \end{aligned} \tag{4.27}$$

Comparison of (4.26) and (4.27) shows that the latter approach is equivalent to considering a $H = \frac{M}{A}$ periodization of Θ, as described above.

Note that Variant II is limited to functions that decay sufficiently fast in the interval $[-H/2, H/2)$. In other words, whenever Θ is not sufficiently small we need to choose a relatively large period $H \gg 2L$, which results in the choice of a large mesh size M. We give a graphical illustration of Variant II in Figure 4.5. Hereby, the approximation error due to the periodization can be seen as the difference between the dashed function Θ and its blue colored periodic version. The main advantage of Variant II is that the periodic function $\sum_{n\in\mathbb{Z}} \Theta(\cdot + Hn)$ is indefinitely often differentiable and, therefore, its Fourier series yields spectral convergence.

In the context of fast Ewald summation this approach was first mentioned in [70, Equation (6-113)] in combination with 0d-periodic boundary conditions. Therein, the authors simply doubled the extend of the primary box in each dimension and used the Fourier coefficients of the 3d-periodic Ewald sum. This can be interpreted as a straightforward generalization of Variant II to three dimensions and setting $H = 2L$.

More recently, Variant II was used in [93] for constructing a spectrally convergent 2d-periodic fast Ewald summation. In contrast to Variant I, this approximation approach can not be used for the non-decreasing function $\Theta_{\mathcal{S}_2}(0, r)$ in the 2d-periodic Ewald. Therefore, another non-FFT based approximation of this term was proposed in [93].

Variant III (Regularization):

In [3] we presented a third approach for the approximation of the function $\Theta(x) \approx \sum_{k=-M/2}^{M/2-1} c_k \mathrm{e}^{-2\pi \mathrm{i} k x/H}$ by a finite Fourier series. The key idea is to cutoff Θ outside the interval $[-L, L]$ but use a Fourier approximation on the slightly larger interval $[-H/2, H/2]$. In the resulting gap $[L, H-L]$ we construct a transition function $T\colon [L, H-L] \to \mathbb{R}$ that interpolates the derivatives of Θ at $x = L$ and $x = H - L$ up to order $s \in \mathbb{N}$, see Figure 4.6 for a graphical illustration. Therefore, we get a Fourier approximation of a s-times differentiable function which yields convergence order $(s+2)$ in Fourier space. Note that we are free to choose an arbitrary order of

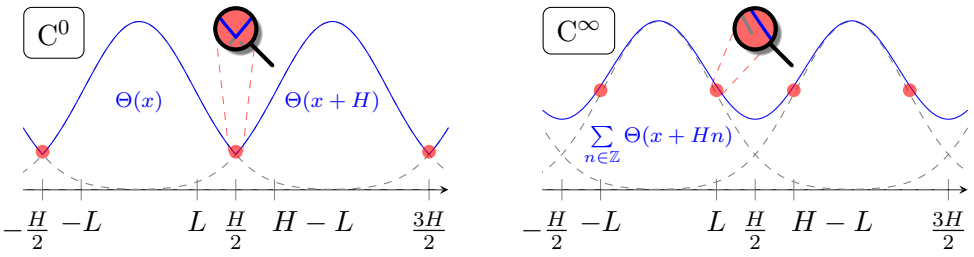

Figure 4.5: Variant I (truncation) on the left and Variant II (periodization) on the right side. While the truncation approach results in a kink at $x = {}^{H}/_{2}$, the approximation error of the periodization mainly depends on the fast decay of Θ.

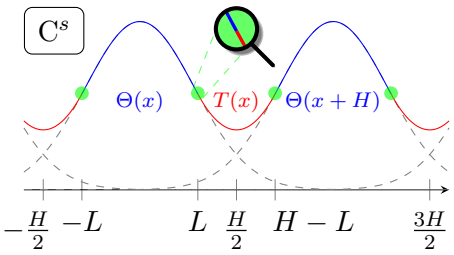

Figure 4.6: Variant III (regularization). The smooth transition function T interpolates the derivatives of Θ at $x = L$ and $x = H - L$ up to order $s \in \mathbb{N}$.

convergence $s \in \mathbb{N}$ within this approach. In general, the Fourier coefficients c_k are computed approximately by a fast Fourier transform from the equispaced samples of the regularized function at $x = lHM^{-1}$, $l = -{}^{M}/_{2}, \ldots, {}^{M}/_{2} - 1$.

The main advantage of Variant III is that we are able construct a function of arbitrary smoothness $s \in \mathbb{N}$ while the period H can be chosen relatively small compared to the doubled interval length $2L$. In contrast, when applying Variant I the functions are only continuous and of no higher smoothness. Thus, the Fourier coefficients decrease rather slow, which results in the choice of a relatively large mesh size M. On the other hand, using Variant II often implies a very large extension interval $[-{}^{H}/_{2}, {}^{H}/_{2}]$ in order to ensure sufficient decay of $\Theta({}^{H}/_{2})$. Again, this has to be compensated by a large number of sampling nodes M.

Loosely speaking, Variant III can be understood as a compromise between the fixed second order convergence of Variant I and the spectral convergence of Variant II. The benefit of this compromise is that Variant III can be applied to non-decreasing functions Θ, while Variant II was restricted to fast decreasing functions. Thereby, the degree of smoothness $s \in \mathbb{N}$ can be understood as a parameter that adjusts the balance between Variant I and Variant II.

We remark that in our application we always know the function values and the

Framework 4.4 P²NFFT – Particle-Particle–NFFT

Input:

- short-range potential $\psi^{\mathsf{sr}}\colon \mathbb{R}^3 \to \mathbb{R}$
- self potential $\psi^{\mathsf{self}} \in \mathbb{R}$
- precomputed Fourier coefficients $\hat{R}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}$

1: Apply the short-range interaction Module 4.1.

2: Apply the self interaction Module 4.2.

3: Apply the long-range interaction Module 4.3.

Output:

- approximated potential $\phi_{\mathcal{S}_p}(\boldsymbol{r}_j)$, $j = 1, \ldots, N$
- approximated field $\boldsymbol{E}_{\mathcal{S}_p}(\boldsymbol{r}_j)$, $j = 1, \ldots, N$

derivatives in the boundary points. Methods without this knowledge are known as Fourier extensions [74] or Fourier continuations [94].

4.5 The P²NFFT framework

The Particle-Particle–NFFT (P²NFFT) framework is a composition of the short-range interaction Module 4.1, the self interaction Module 4.2, and the long-range interaction Module 4.3. We give a brief summary of P²NFFT in Framework 4.4. Thereby, we only concentrate on the input parameters that depend on the particular choice of splitting function. In the case of the Ewald splitting we were able to derive fast algorithms for all kinds of mixed periodic boundary conditions. However, it is also possible to use other splitting functions. For example, we show in Section 4.7.1 that the modularized structure allows us to include the NFFT-based fast summation algorithm [112, 113] within the P²NFFT framework.

Even more, we are able to construct a broad variety of particle-mesh algorithms by considering all the possible combinations of algorithmic modules that are part of P²NFFT. Especially, we are free to choose any of the available PNFFT modules that were presented in Section 3.5. In the following, we discuss the relation of the P²NFFT to a selection of well known particle-mesh algorithms. Special emphasize will be given to the strong connection of P²NFFT and P³M.

4.6 The relation of P²NFFT and P³M

The Particle-Particle–Particle-Mesh method (P³M) [70, 38, 39] is the oldest example of a mesh-based Ewald summation. In the following, we show that the P³M can

be essentially interpreted as a special case of P^{2}NFFT with 3d-periodic boundary conditions. However, we will also see that P^{2}NFFT profits a lot from the experiences of the P^{3}M community. Both methods are based on the Ewald splitting and share exactly the same short-range evaluations. Therefore, we only concentrate on the long-range interactions of these two methods. For the sake of notational convenience, we assume a unit box shape $\mathbf{L} = \mathbf{I}_3$. We emphasize that all of the following comparisons apply analogously for tricline box shapes. The P^{3}M long-range part for computing the potentials is given in our notation by the following matrix decomposition

$$\mathbf{C}_{N,\hat{\boldsymbol{m}}} \widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}} \operatorname{diag}\left(\hat{G}_{\mathrm{opt}}(\boldsymbol{k})\right)_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}^{\mathsf{H}} \mathbf{C}_{N,\hat{\boldsymbol{m}}}^{\top}, \tag{4.28}$$

where $\mathbf{C}_{N,\hat{\boldsymbol{m}}}$ denotes a discrete convolution with the B-Spline window function (3.28) as given in (3.22) and $\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}} = (\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}(\boldsymbol{l}\odot\hat{\boldsymbol{m}}^{-1})})_{\boldsymbol{l},\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}}$ denotes the (non-pruned) Fourier matrix of size $\hat{\boldsymbol{m}}$ with shifted index sets. The function $\hat{G}_{\mathrm{opt}} : \mathbb{R}^3 \to \mathbb{R}$ is often called optimal influence function in the P^{3}M terminology and will be defined later on. For the sake of convenience we abbreviate the vector $\hat{\boldsymbol{G}} := \left(\hat{G}_{\mathrm{opt}}(\boldsymbol{k})\right)_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}}$ in all what follows. Note that the matrix decomposition (4.28) already shares a lot of matrix factors with the matrix representations (3.20), (3.23) of the NFFT and its adjoint. The P^{3}M with analytic differentiation can be written as

$$\nabla\mathbf{C}_{3N,\hat{\boldsymbol{m}}} \widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{G}}) \widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}^{\mathsf{H}} \mathbf{C}_{N,\hat{\boldsymbol{m}}}^{\top}, \tag{4.29}$$

where $\nabla\mathbf{C}_{3N,\hat{\boldsymbol{m}}}$ denotes a discrete convolution with the gradient of the B-Spline window function as given in (3.25). The i$\boldsymbol{k}$-derivative is given by

$$\left(\mathbf{C}_{N,\boldsymbol{m}} \widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}} \otimes \mathbf{I}_3\right) \mathbf{K}_{3\hat{\boldsymbol{m}},\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{G}}) \widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}^{\mathsf{H}} \mathbf{C}_{N,\hat{\boldsymbol{m}}}^{\top}. \tag{4.30}$$

Hereby, $\mathbf{K}_{3\hat{\boldsymbol{m}},\hat{\boldsymbol{m}}} := (-2\pi\mathrm{i}\boldsymbol{k}\delta_{\boldsymbol{k}-\boldsymbol{l}})_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}};\,\boldsymbol{l}\in\mathcal{I}_{\hat{\boldsymbol{m}}}}$ accomplishes the derivative in Fourier space. In compliance to the NFFT naming scheme we denote by ad-P^{3}M and i$\boldsymbol{k}$-P^{3}M the P^{3}M method in combination with the corresponding differentiation scheme.

Analogously, we denote the P^{2}NFFT with analytic derivative (ad-P^{2}NFFT) and with derivative in Fourier space (i$\boldsymbol{k}$-P^{2}NFFT).

In general, the optimal influence function $\hat{G}_{\mathrm{opt}}$ will differ for all of the above mentioned P^{3}M algorithms. The values $\hat{G}_{\mathrm{opt}}(\boldsymbol{k})$ are interpreted as parameters of the method that can be freely chosen. The philosophy of P^{3}M is to choose these values such that the mean square aliasing error of the whole method is minimized. Formulas of the optimal influence functions have been derived for many types of P^{3}M methods. We refer to [23, Equations (4.10), (4.11)] for a nice overview. In the following, we summarize these formulas in our notation and compare them with the NFFT deconvolution. In order to keep notation short, we abbreviate $\boldsymbol{k}_{\boldsymbol{r}} := \boldsymbol{k}+\boldsymbol{r}\odot\hat{\boldsymbol{m}}$ in all what follows. Then, the numerator $B(\boldsymbol{k})$ and denominator $A(\boldsymbol{k})$ of the optimal

influence function $\hat{G}_{\mathrm{opt}}(\boldsymbol{k}) = B(\boldsymbol{k})/A(\boldsymbol{k})$ are given by

$$A(\boldsymbol{k}) := \begin{cases} |\mathcal{I}_{\hat{\boldsymbol{m}}}|^4 \Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \Big)^2 & \text{for opt. potential,} \\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^4 \Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \Big)^2 \|\boldsymbol{k}\|^2 & \text{for opt. i}\boldsymbol{k}\text{-field,} \\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^4 \Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \Big) \Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \|\boldsymbol{k}_{\boldsymbol{r}}\|^2 \Big) & \text{for opt. ad-field,} \end{cases} \tag{4.31}$$

and

$$B(\boldsymbol{k}) := \begin{cases} |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2 \sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{R}_{\boldsymbol{k}_{\boldsymbol{r}}} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} & \text{for opt. potential,} \\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2 \sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{R}_{\boldsymbol{k}_{\boldsymbol{r}}} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \boldsymbol{k}^{\mathsf{T}} \boldsymbol{k}_{\boldsymbol{r}} & \text{for opt. i}\boldsymbol{k}\text{-field,} \\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2 \sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{R}_{\boldsymbol{k}_{\boldsymbol{r}}} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \|\boldsymbol{k}_{\boldsymbol{r}}\|^2 & \text{for opt. ad-field.} \end{cases} \tag{4.32}$$

Thereby, the coefficients $\hat{R}_{\boldsymbol{k}} = \pi^{-1}\|\boldsymbol{k}\|^{-2} \mathrm{e}^{-\pi^2\|\boldsymbol{k}\|^2/\alpha^2}$ are given by the long-range part (4.15) of the Ewald splitting. Because of the fast decay of $\hat{R}_{\boldsymbol{k}}$ we get a very good approximation of $B(\boldsymbol{k})$ by only considering the $\boldsymbol{r} = \boldsymbol{0}$ terms of the sums, i.e.,

$$B(\boldsymbol{k}) \approx \hat{R}_{\boldsymbol{k}} \begin{cases} |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2 \hat{\varphi}^2_{\boldsymbol{k}} & \text{for opt. potential,} \\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2 \hat{\varphi}^2_{\boldsymbol{k}} \|\boldsymbol{k}\|^2 & \text{for opt. ad- and i}\boldsymbol{k}\text{-field.} \end{cases} \tag{4.33}$$

Then, the optimal influence functions for computing the potential and the fields via i$\boldsymbol{k}$-derivative simplify to the same expression

$$\hat{G}^{\mathsf{ik}}_{\mathrm{opt}}(\boldsymbol{k}) = \frac{B(\boldsymbol{k})}{A(\boldsymbol{k})} \approx \frac{\hat{\varphi}_{\boldsymbol{k}}}{|\mathcal{I}_{\hat{\boldsymbol{m}}}| \sum_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}} \hat{R}_{\boldsymbol{k}} \frac{\hat{\varphi}_{\boldsymbol{k}}}{|\mathcal{I}_{\hat{\boldsymbol{m}}}| \sum_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}}. \tag{4.34}$$

This expression exhibits a remarkable agreement with the NFFT deconvolution steps. The factors on the right hand side of (4.34) can be interpreted from right to left as follows. First, the optimal deconvolution (3.39) of the NFFT$^{\mathsf{H}}$ is applied. Next, the Fourier coefficients of the Ewald splitting are multiplied. Finally, the optimal deconvolution (3.39) of the NFFT is applied. Thereby, the last step can also be interpreted as the optimal deconvolution (3.43) of the i$\boldsymbol{k}$-NFFT. Note that these are exactly the deconvolutions that are applied in the long-range interaction Module 4.3 due to the NFFT$^{\mathsf{H}}$ in Step 2 and the NFFT in Step 4. Analogously, the optimal P^{3}M influence function for computing the ad-derivative simplifies to

$$\hat{G}^{\mathsf{ad}}_{\mathrm{opt}}(\boldsymbol{k}) = \frac{B(\boldsymbol{k})}{A(\boldsymbol{k})} \approx \frac{\hat{\varphi}_{\boldsymbol{k}}\|\boldsymbol{k}\|^2}{|\mathcal{I}_{\hat{\boldsymbol{m}}}| \sum_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} \|\boldsymbol{k}_{\boldsymbol{r}}\|^2} \hat{R}_{\boldsymbol{k}} \frac{\hat{\varphi}_{\boldsymbol{k}}}{|\mathcal{I}_{\hat{\boldsymbol{m}}}| \sum_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}}.$$

Hereby, the only difference to (4.34) is the leftmost factor, which can be interpreted as the optimal deconvolution (3.41) of the ad-NFFT.

$$\text{P}^2\text{NFFT:} \quad \overbrace{\mathbf{C}_{N,\hat{\boldsymbol{m}}}\,\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\,\mathbf{D}_{\hat{\boldsymbol{m}}}}^{\text{NFFT}}\;\underbrace{\operatorname{diag}(\hat{\boldsymbol{R}})}_{\text{optimal influence function}}\;\overbrace{\mathbf{D}_{\hat{\boldsymbol{m}}}\,\widetilde{\mathbf{F}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\,\mathbf{C}^{\top}_{N,\hat{\boldsymbol{m}}}}^{\text{NFFT}^{\mathsf{H}}} \quad \text{P}^3\text{M:}$$

Figure 4.7: For 3d-periodic boundary conditions P^{3}M and P^{2}NFFT differ only in the viewpoint on the same algorithm.

Now, the matrix representation (4.28) of the P^{3}M long-range part becomes by application of (4.34)

$$\mathbf{C}_{N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\mathbf{D}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{D}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\mathbf{C}^{\top}_{N,\hat{\boldsymbol{m}}},$$

where we abbreviate $\hat{\boldsymbol{R}} := (\hat{R}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_{\hat{\boldsymbol{m}}}} \in \mathbb{C}^{|\mathcal{I}_{\hat{\boldsymbol{m}}}|}$. This representation reveals the matrix factorization of the non-pruned NFFT $\mathbf{A}_{N,\hat{\boldsymbol{m}}} \approx \mathbf{C}_{N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\mathbf{D}_{\hat{\boldsymbol{m}}}$ and its adjoint. This means, that 3d-periodic P^{2}NFFT and P^{3}M can be understood as two different viewpoints on the same algorithm as shown in Figure 4.7. We have also seen that an analogous connection exists between i$\boldsymbol{k}$-P^{3}M (4.30) and i$\boldsymbol{k}$-P^{2}NFFT as well as between ad-P^{3}M (4.29) and ad-P^{2}NFFT. This means that the only difference between P^{2}NFFT and P^{3}M is approximation (4.33) in the numerator of the optimal P^{3}M influence function. Since P^{2}NFFT looks at the NFFT steps as uncoupled modules, it implicitly applies this additional approximation. In contrast, P^{3}M looks at the whole algorithm at once and gets the exact influence function. However, we also see that the denominators, which give the main contribution of the P^{3}M influence function, are completely recovered by the NFFT based splitting.

This means, as long as the Ewald coefficients $\hat{R}_{\boldsymbol{k}}$ decay fast enough such that the $\boldsymbol{r} = \boldsymbol{0}$ alias term dominates the numerator (4.32), P^{3}M can be interpreted as a special case of P^{2}NFFT. However, if this requirement is not fulfilled we can turn the 3d-periodic P^{2}NFFT efficiently into P^{3}M by a simple rescale in Fourier space. The main advantages of P^{2}NFFT are the free choice of window function, the possibility to use oversampling and the generalization to mixed- and non-periodic boundary conditions.

Remark 4.1. Within this thesis we only investigated the aliasing error of the NFFT and optimized the deconvolution steps for this error. As long as NFFT applies a window function that exhibits compact support in real space this will be the only source of NFFT approximation error. Especially, this is valid for the B-Spline (3.28) and the Bessel-I0 window (3.30). For these windows it is correct that the P^{3}M influence function is the best choice in the sense that it optimizes the overall mean square approximation error. However, for all the other windows introduced in Section 3.2.1 we have an additional truncation error that is not considered in P^{3}M and also not in

the optimal NFFT deconvolution presented in this thesis. For a more general view on the optimal influence function of P²NFFT with aliasing and truncation error we refer to the recent work [103]. □

4.6.1 Interlaced P²NFFT and P³M

Interlacing in the context of P³M has been introduced in [70]. Thereby, the authors distinguished two forms of interlacing, namely the force averaging scheme [70, Sec. 7-8-1] and the harmonic average [70, Sec. 7-8-2]. The key idea of the force averaging approach is to shift the whole particle system by half the inverse mesh size $1/2\hat{\boldsymbol{m}}^{-1}$, compute the Coulomb field, and average with the field resulting from the non-shifted particle system. Analogously, this can be done for the Coulomb potential, energy and force. This scheme applied to the Coulomb field can be written in matrix form as

$$\tfrac{1}{2}\left[\nabla\mathbf{C}_{3N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{G}})\widetilde{\mathbf{F}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\mathbf{C}^{\top}_{N,\hat{\boldsymbol{m}}} + \nabla\widetilde{\mathbf{C}}_{3N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{G}})\widetilde{\mathbf{F}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{C}}^{\top}_{N,\hat{\boldsymbol{m}}}\right], \tag{4.35}$$

where $\widetilde{\mathbf{C}}_{N,\hat{\boldsymbol{m}}}$ and $\nabla\widetilde{\mathbf{C}}_{3N,\hat{\boldsymbol{m}}}$ stand for the discrete convolution steps with shifted particle positions that were defined in (3.34) and (3.35). The optimal influence functions $\hat{G}_{\mathrm{opt}}(\boldsymbol{k}) = B(\boldsymbol{k})/A(\boldsymbol{k})$ for interlaced P³M, ad-P³M, and i$\boldsymbol{k}$-P³M are given in [101] as follows. While the numerator $B(\boldsymbol{k})$ remains the same as in the non-interlaced case, the denominator is given by $A(\boldsymbol{k}) = \frac{1}{2}[A_1(\boldsymbol{k}) + A_2(\boldsymbol{k})]$. Hereby, $A_1(\boldsymbol{k})$ is the denominator (4.31) from the non-interlaced case and $A_2(\boldsymbol{k})$ is defined as

$$A_2(\boldsymbol{k}) := \begin{cases} |\mathcal{I}_{\hat{\boldsymbol{m}}}|^4\Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3}(-1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}\Big)^2 & \text{for opt. potential,}\\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^4\Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3}(-1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}\Big)^2\|\boldsymbol{k}\|^2 & \text{for opt. i}\boldsymbol{k}\text{-field,}\\ |\mathcal{I}_{\hat{\boldsymbol{m}}}|^4\Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3}(-1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}\Big)\Big(\sum\limits_{\boldsymbol{r}\in\mathbb{Z}^3}(-1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}\|\boldsymbol{k}_{\boldsymbol{r}}\|^2\Big) & \text{for opt. ad-field.}\end{cases}$$

In the following, we are going to investigate the connection of interlaced ad-P³M and ad-P²NFFT. We emphasize that it is straightforward to find similar results for interlaced P³M and i$\boldsymbol{k}$-P³M. In order to get a compact representation of the rather long influence function denominators we introduce the following abbreviations

$$\begin{aligned} A^{\mathsf{pot}}_{\pm}(\boldsymbol{k}) &:= |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2\sum_{\boldsymbol{r}\in\mathbb{Z}^3}(\pm1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}} && \text{for opt. potential,}\\ A^{\mathsf{ik}}_{\pm}(\boldsymbol{k}) &:= |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2\Big(\sum_{\boldsymbol{r}\in\mathbb{Z}^3}(\pm1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}\Big)\|\boldsymbol{k}\|^2 && \text{for opt. i}\boldsymbol{k}\text{-field,}\\ A^{\mathsf{ad}}_{\pm}(\boldsymbol{k}) &:= |\mathcal{I}_{\hat{\boldsymbol{m}}}|^2\Big(\sum_{\boldsymbol{r}\in\mathbb{Z}^3}(\pm1)^{|\boldsymbol{r}|}\hat{\varphi}^2_{\boldsymbol{k}_{\boldsymbol{r}}}\|\boldsymbol{k}_{\boldsymbol{r}}\|^2\Big) && \text{for opt. ad-field.}\end{aligned}$$

Using the same approximation (4.33) of $B(\boldsymbol{k})$ as above, the optimal influence function of interlaced ad-P^{3}M can be written as

$$\frac{2B(\boldsymbol{k})}{A_1(\boldsymbol{k})+A_2(\boldsymbol{k})} \approx \hat{R}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}}^2\|\boldsymbol{k}\|^2|\mathcal{I}_{\hat{\boldsymbol{m}}}|^2\frac{2}{A_+^{\mathsf{ad}}A_+^{\mathsf{pot}}+A_-^{\mathsf{ad}}A_-^{\mathsf{pot}}}. \tag{4.36}$$

From the P^{2}NFFT point of view it is more natural to incorporate interlacing in the form of interlaced NFFT, cf. Section 3.2.3. In order to keep notation short, we introduce the abbreviations

$$\nabla\mathbf{N}_{3N,\hat{\boldsymbol{m}}} := \nabla\mathbf{C}_{3N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\mathbf{D}_{\hat{\boldsymbol{m}}}^{\mathsf{ad}}, \qquad \mathbf{N}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} := \mathbf{D}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}^{\mathsf{H}}\mathbf{C}_{N,\hat{\boldsymbol{m}}}^{\top},$$

for the matrix notation of the ad-NFFT and NFFT$^{\mathsf{H}}$ from Section 3.2 as well as

$$\nabla\widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}} := \nabla\widetilde{\mathbf{C}}_{3N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}^{\mathsf{ad}}, \qquad \widetilde{\mathbf{N}}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} := \widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}^{\mathsf{H}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}^{\mathsf{H}}\widetilde{\mathbf{C}}_{N,\hat{\boldsymbol{m}}}^{\top},$$

for the matrix representation of the shifted ad-NFFT and shifted NFFT$^{\mathsf{H}}$ from Section 3.2.2. Thereby, the entries of the diagonal matrices $\mathbf{D}_{\hat{\boldsymbol{m}}}$ and $\mathbf{D}_{\hat{\boldsymbol{m}}}^{\mathsf{ad}}$ are given by the optimal deconvolution coefficients (3.39) and (3.41), respectively. Then, the interlaced ad-NFFT reads as $\nabla\mathbf{A}_{3N,\hat{\boldsymbol{m}}} \approx (\nabla\mathbf{N}_{3N,\hat{\boldsymbol{m}}} + \nabla\widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}})/2$ and the interlaced NFFT$^{\mathsf{H}}$ can be written as $\mathbf{A}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} \approx (\mathbf{N}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} + \widetilde{\mathbf{N}}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}})/2$. Now, we have the freedom to choose between three different schemes in order to incorporate interlacing in P^{2}NFFT.

As a first approach, we apply interlacing for ad-NFFT and NFFT$^{\mathsf{H}}$ at once. This results in the following matrix representation of P^{2}NFFT

$$\tfrac{1}{2}\left[\nabla\mathbf{N}_{3N,\hat{\boldsymbol{m}}} + \nabla\widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}}\right]\operatorname{diag}(\hat{\boldsymbol{R}})\tfrac{1}{2}\left[\mathbf{N}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} + \widetilde{\mathbf{N}}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}}\right]. \tag{4.37}$$

Then, the P^{2}NFFT influence function is given by the product $\mathbf{D}_{\hat{\boldsymbol{m}}}^{\mathsf{ad}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{D}_{\hat{\boldsymbol{m}}}$, where the entries of $\mathbf{D}_{\hat{\boldsymbol{m}}}^{\mathsf{ad}}$ are given by the optimal deconvolution (3.45) for interlaced ad-NFFT and the entries of $\mathbf{D}_{\hat{\boldsymbol{m}}}$ are given by the optimal deconvolution (3.46) for interlaced NFFT$^{\mathsf{H}}$. Altogether, this yields

$$\begin{aligned}&\frac{4\hat{R}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}}^2\|\boldsymbol{k}\|^2}{|I_{\hat{\boldsymbol{m}}}|^2}\left[\sum_{\boldsymbol{r}\in\mathbb{Z}^3}[1+(-1)^{|\boldsymbol{r}|}]\hat{\varphi}_{\boldsymbol{k}_{\boldsymbol{r}}}^2\sum_{\boldsymbol{s}\in\mathbb{Z}^3}[1+(-1)^{|\boldsymbol{s}|}]\hat{\varphi}_{\boldsymbol{k}_{\boldsymbol{s}}}^2\|\boldsymbol{k}_{\boldsymbol{s}}\|^2\right]^{-1}\\ &= \hat{R}_{\boldsymbol{k}}\hat{\varphi}_{\boldsymbol{k}}^2\|\boldsymbol{k}\|^2|I_{\hat{\boldsymbol{m}}}|^2\frac{4}{A_+^{\mathsf{ad}}A_+^{\mathsf{pot}}+A_-^{\mathsf{ad}}A_+^{\mathsf{pot}}+A_+^{\mathsf{ad}}A_-^{\mathsf{pot}}+A_-^{\mathsf{ad}}A_-^{\mathsf{pot}}}.\end{aligned}$$

Again, this can be written in the form $B(\boldsymbol{k})/A(\boldsymbol{k})$, where $B(\boldsymbol{k})$ is given by approximation (4.33) and $A(\boldsymbol{k}) = [A_+^{\mathsf{ad}}A_+^{\mathsf{pot}}+A_-^{\mathsf{ad}}A_+^{\mathsf{pot}}+A_+^{\mathsf{ad}}A_-^{\mathsf{pot}}+A_-^{\mathsf{ad}}A_-^{\mathsf{pot}}]/4$. On the other hand, an expansion of (4.37) shows that it can be interpreted as an average over four different types of shifted P^{2}NFFT

$$\begin{aligned}&\tfrac{1}{4}\nabla\mathbf{N}_{3N,\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{N}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} + \tfrac{1}{4}\nabla\mathbf{N}_{3N,\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\widetilde{\mathbf{N}}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}}\\ &+\tfrac{1}{4}\nabla\widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{N}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}} + \tfrac{1}{4}\nabla\widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\widetilde{\mathbf{N}}_{N,\hat{\boldsymbol{m}}}^{\mathsf{H}}.\end{aligned} \tag{4.38}$$

Note that the simple substitutions

$$\begin{aligned} \nabla \mathbf{N}_{3N,\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{R}}) \mathbf{N}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} &\to A^{\mathsf{ad}}_{+} A^{\mathsf{pot}}_{+}, \quad \nabla \mathbf{N}_{3N,\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{R}}) \widetilde{\mathbf{N}}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} \to A^{\mathsf{ad}}_{+} A^{\mathsf{pot}}_{-}, \\ \nabla \widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{R}}) \mathbf{N}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} &\to A^{\mathsf{ad}}_{-} A^{\mathsf{pot}}_{+}, \quad \nabla \widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{R}}) \widetilde{\mathbf{N}}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} \to A^{\mathsf{ad}}_{-} A^{\mathsf{pot}}_{-} \end{aligned} \tag{4.39}$$

turn (4.38) into the denominator $A(\boldsymbol{k})$, which gives us a deep insight in the structure of the influence function.

If we only apply interlacing to the NFFT$^{\mathsf{H}}$, P^{2}NFFT turns into

$$\nabla \mathbf{N}_{3N,\hat{\boldsymbol{m}}} \operatorname{diag}(\hat{\boldsymbol{R}}) \tfrac{1}{2} \left[\mathbf{N}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} + \widetilde{\mathbf{N}}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}} \right]. \tag{4.40}$$

and its corresponding influence function is given by

$$\hat{R}_{\boldsymbol{k}} \hat{\varphi}^2_{\boldsymbol{k}} \|\boldsymbol{k}\|^2 |I_{\hat{\boldsymbol{m}}}|^2 \frac{2}{A^{\mathsf{ad}}_{+}(A^{\mathsf{pot}}_{+} + A^{\mathsf{pot}}_{-})} \approx \frac{B(\boldsymbol{k})}{A(\boldsymbol{k})},$$

with $A(\boldsymbol{k}) = [A^{\mathsf{ad}}_{+} A^{\mathsf{pot}}_{+} + A^{\mathsf{ad}}_{+} A^{\mathsf{pot}}_{-})]/2$. The latter results from the product of the optimal deconvolution coefficients (3.41), (3.45) of the ad-NFFT and the interlaced NFFT$^{\mathsf{H}}$. Again, we see that the structure of the denominator $A(\boldsymbol{k})$ recapitulates the structure of the algorithms' matrix representation with the substitutions (4.39). Note that (4.40) is exactly the so-called harmonic average P^{3}M presented in [70, Sec. 7-8-2]. However, within the P^{2}NFFT framework we were also able to derive the optimal influence function of this algorithm for the first time.

Numerical experiments show that the accuracy benefit of ad-P^{3}M with interlacing is much higher than for interlaced P^{3}M and i$\boldsymbol{k}$-P^{3}M [101]. Therefore, the ad-NFFT seems to be the main source of error. Following this line of argumentation it might be better to avoid the interlacing in the NFFT$^{\mathsf{H}}$ and incorporate only the interlaced ad-NFFT as follows

$$\tfrac{1}{2} \left[\nabla \mathbf{N}_{3N,\hat{\boldsymbol{m}}} + \nabla \widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}} \right] \operatorname{diag}(\hat{\boldsymbol{R}}) \mathbf{N}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}}.$$

This scheme has the advantage that it saves one application of the NFFT$^{\mathsf{H}}$ in comparison to the standard interlaced P^{3}M (4.35) but still offers the interlacing during the computation of the analytic derivatives. Therefore, the overall number of FFTs is reduced from 4 to 3 in comparison to the interlaced ad-P^{3}M. The corresponding influence function is given by

$$\hat{R}_{\boldsymbol{k}} \hat{\varphi}^2_{\boldsymbol{k}} \|\boldsymbol{k}\|^2 |I_{\hat{\boldsymbol{m}}}|^2 \frac{2}{(A^{\mathsf{ad}}_{+} + A^{\mathsf{ad}}_{-}) A^{\mathsf{pot}}_{+}} \approx \frac{B(\boldsymbol{k})}{A(\boldsymbol{k})},$$

with $A(\boldsymbol{k}) = [A^{\mathsf{ad}}_{+} A^{\mathsf{pot}}_{+} + A^{\mathsf{ad}}_{-} A^{\mathsf{pot}}_{+}]/2$. It can be derived from the product of the optimal deconvolution coefficients (3.46), (3.39) of the interlaced ad-NFFT and the NFFT$^{\mathsf{H}}$.

Again, the matrix decomposition directly leads to the structure of the denominator $A(\boldsymbol{k})$ with the substitutions (4.39).

So far none of the three interlaced P^{2}NFFT algorithms got the same matrix decomposition as the interlaced P^{3}M. The NFFT based approach that is closest to interlaced P^{3}M would look like

$$\begin{aligned}&\tfrac{1}{2}\left[\nabla\mathbf{N}_{3N,\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{N}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}}+\nabla\widetilde{\mathbf{N}}_{3N,\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\widetilde{\mathbf{N}}^{\mathsf{H}}_{N,\hat{\boldsymbol{m}}}\right]\\ =&\tfrac{1}{2}\left[\nabla\mathbf{C}_{3N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\mathbf{D}^{\mathsf{ad}}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{D}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\mathbf{C}^{\top}_{N,\hat{\boldsymbol{m}}}+\nabla\widetilde{\mathbf{C}}_{3N,\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{D}}^{\mathsf{ad}}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{F}}^{\mathsf{H}}_{\hat{\boldsymbol{m}}}\widetilde{\mathbf{C}}^{\top}_{N,\hat{\boldsymbol{m}}}\right].\end{aligned}$$

The resulting influence function $\hat{R}_{\boldsymbol{k}}\hat{\varphi}^2_{\boldsymbol{k}}\|\boldsymbol{k}\|^2|I_{\hat{\boldsymbol{m}}}|^2/(A^{\mathsf{ad}}_+A^{\mathsf{pot}}_+)$ of this approach is given by the entries of $\mathbf{D}^{\mathsf{ad}}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\mathbf{D}_{\hat{\boldsymbol{m}}}=\widetilde{\mathbf{D}}^{\mathsf{ad}}_{\hat{\boldsymbol{m}}}\operatorname{diag}(\hat{\boldsymbol{R}})\widetilde{\mathbf{D}}_{\hat{\boldsymbol{m}}}$, cf. (3.41) for $\mathbf{D}^{\mathsf{ad}}_{\hat{\boldsymbol{m}}}$ and (3.39) for $\mathbf{D}_{\hat{\boldsymbol{m}}}$. Obviously, this does not match the exact version (4.36) and also was not suspected if we used the substitutions (4.39) to derive the denominator $A(\boldsymbol{k})$ directly from the matrix representation of the algorithm. Once more, this is due to the fact that P^{2}NFFT does not look at the whole algorithm at once. The NFFT module does not consider at all that there is another instance of NFFT with shifted mesh. Loosely speaking, the P^{3}M viewpoint adds the missing link that was invisible for the modularized approach of NFFT. On the other hand, from the NFFT point of view we were able to find three alternative algorithms that incorporate interlacing in P^{3}M and found the corresponding optimal influence functions. In this sense the interlacing approaches of P^{3}M and P^{2}NFFT are complementary. Altogether, we got four different schemes that incorporate interlacing in P^{2}NFFT and derived the corresponding optimal influence functions.

Remark 4.2. In this section we assumed a unit box shape $\mathbf{L}=\mathbf{I}_3$ in order to get a unified look at P^{3}M and P^{2}NFFT. Nevertheless, all of the above presented formulas can be generalized to tricline box shapes. For P^{3}M with tricline box geometry we refer to [101, 23]. Loosely speaking, all vectors $\boldsymbol{k}_{\boldsymbol{r}}$ have to be replaced by $\mathbf{L}^{-\mathsf{T}}\boldsymbol{k}_{\boldsymbol{r}}$ in order to get the tricline case. We emphasize, that the P^{2}NFFT framework introduces the substitution $\boldsymbol{x}=\mathbf{L}^{-1}\boldsymbol{r}$ before it calls the NFFT. Together with the resulting substitution in Fourier space $\boldsymbol{k}=\mathbf{L}^{\mathsf{T}}\boldsymbol{k}_{\boldsymbol{r}}$ this causes that the matrix representation of the NFFT does not change at all for the tricline case. Therefore, all ingredients of the influence function that originate from the NFFT deconvolution steps look less complicated than the corresponding parts of the P^{3}M formulas. This makes the equivalence of the two methods harder to see, although it is still valid. □

4.7 The relation of P^{2}NFFT to other particle-mesh methods

Beside P^{3}M, the P^{2}NFFT Framework 4.4 additionally includes many other well known particle-mesh algorithms. In the following, we present a selected list of

algorithms that are part of the framework.

4.7.1 Fast summation with non-periodic boundary conditions

We start with an outline the NFFT-based fast summation algorithm [112, 113] for the fast computation of (4.1) and (4.2) with non-periodic boundary conditions. It requires $\mathcal{O}(N \log N)$ arithmetic operations for uniformly distributed source nodes $\boldsymbol{r}_j$. At the end, we will see that this method is part of the P²NFFT framework.

Assume $H > 2\sqrt{3}L$, $\mathbf{H} := \operatorname{diag}(H, H, H)$ and non-periodic boundary conditions. We construct a smooth regularization $R\colon \mathbf{H}\big[-1/2, 1/2\big)^3 \to \mathbb{R}$ of the kernel function $\|\boldsymbol{r}\|^{-1}$ as

$$R(\boldsymbol{r}) := \begin{cases} T^{\mathsf{sr}}(\|\boldsymbol{r}\|) & \text{if } \|\boldsymbol{r}\| \leq r_{\mathrm{c}}, \\ \widetilde{\Theta}_{\mathcal{S}_0}(\|\boldsymbol{r}\|) & \text{if } r_{\mathrm{c}} < \|\boldsymbol{r}\| \leq \sqrt{3}L, \\ T^{\mathsf{lr}}(\|\boldsymbol{r}\|) & \text{if } \sqrt{3}L < \|\boldsymbol{r}\| \leq \frac{H}{2}, \\ T^{\mathsf{lr}}(\frac{H}{2}) & \text{if } \frac{H}{2} < \|\boldsymbol{r}\|, \end{cases}$$

where $\widetilde{\Theta}_{\mathcal{S}_0}(t) := t^{-1}$. The short-range transition T^{sr} and the long-range transition T^{lr} are chosen such that the regularization R is s times differentiable for an appropriately chosen degree of smoothness $s \in \mathbb{N}$. In contrast to (4.23) an additional short-range transition T^{sr} is necessary since t^{-1} has a singularity at zero. We define T^{sr} by the unique polynomial of degree $2s+2$ that fulfills the $2s$ interpolation conditions

$$\tfrac{\mathrm{d}}{\mathrm{d}x^j} T^{\mathsf{sr}}(r_{\mathrm{c}}) = (-1)^j \tfrac{\mathrm{d}}{\mathrm{d}x^j} T^{\mathsf{sr}}(-r_{\mathrm{c}}) = \widetilde{\Theta}^{(j)}_{\mathcal{S}_0}(r_{\mathrm{c}}),$$

similar to the definition in Section 4.4.3. Furthermore, T^{lr} can be defined by the $2s+1$ interpolation conditions in the same way as for the Ewald splitting

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}x^j} T^{\mathsf{lr}}\left(\sqrt{3}L\right) &= \widetilde{\Theta}^{(j)}_{\mathcal{S}_0}\left(\sqrt{3}L\right), && j = 0, \dots, s, \\ \tfrac{\mathrm{d}}{\mathrm{d}x^j} T^{\mathsf{lr}}\left(\tfrac{H}{2}\right) &= 0, && j = 1, \dots, s, \end{aligned}$$

cf. Section 4.4.5. Since R is periodic and smooth of order s we expect a good approximation by the finite Fourier series

$$R(\boldsymbol{r}) \approx \frac{1}{H^3} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \mathbf{H}^{-1} \boldsymbol{r}}, \tag{4.41}$$

where $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$ is an appropriately chosen finite mesh size and

$$\hat{R}_{\boldsymbol{k}} := \frac{H^3}{|\mathcal{I}_{\hat{\boldsymbol{m}}}|} \sum_{\boldsymbol{l} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} R(H\boldsymbol{l} \odot \hat{\boldsymbol{m}}^{-1}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} (\boldsymbol{l} \odot \hat{\boldsymbol{m}}^{-1})}, \quad \boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}},$$

are the discrete Fourier coefficients that can be precomputed by a three-dimensional FFT. Now, the initial problem of calculating $\phi_{\mathcal{S}_0}(\boldsymbol{r}_j)$ can be rewritten as

$$\sum_{i=1}^{N}{}' \frac{q_i}{\|\boldsymbol{r}_{ij}\|} = \sum_{i=1}^{N}{}' q_i \left(\frac{1}{\|\boldsymbol{r}_{ij}\|} - R(\boldsymbol{r}_{ij}) \right) + \sum_{i=1}^{N} q_i R(\boldsymbol{r}_{ij}) - q_j R(\boldsymbol{0}).$$

The first term on the right hand side can be computed with the short-range interaction Module 4.1. Thereby, we simply set $\psi^{\mathsf{sr}}(\boldsymbol{r}) \leftarrow \frac{1}{\|\boldsymbol{r}\|} - T^{\mathsf{sr}}(\|\boldsymbol{r}\|)$. Furthermore, approximation (4.41) turns the second term into

$$\sum_{i=1}^{N} q_i R(\boldsymbol{r}_{ij}) \approx \sum_{i=1}^{N} q_i \frac{1}{|\det(\mathbf{H})|} \sum_{\boldsymbol{k} \in \mathcal{I}_{\hat{\boldsymbol{m}}}} \hat{R}_{\boldsymbol{k}} \mathrm{e}^{+2\pi \mathrm{i} \boldsymbol{k}^{\mathsf{T}} \mathbf{H}^{-1} \boldsymbol{r}_{ij}},$$

which can be computed by the long-range interaction Module 4.3. Finally, the third term can be computed using the self interaction Module 4.2 with $\psi^{\mathsf{self}} \leftarrow T^{\mathsf{sr}}(0)$. In summary, we see that the fast summation algorithm of [112, 113] is included in the P^{2}NFFT framework by a simple change of the splitting function.

4.7.2 Ewald summation

The P^{2}NFFT framework offers the opportunity to compute all the NFFTs within the long-range interaction Module 4.3 directly by the PNDFT Modules 3.3 and 3.4. In this case we use the name P^{2}NDFT to denote the direct computation of the NFFT parts. Then, the complexity of Module 4.3 raises from $\mathcal{O}(N + |\mathcal{I}_{\hat{\boldsymbol{m}}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ to $\mathcal{O}(N|\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ for all kinds of mixed-periodic boundary conditions. In the case of 3d-periodic boundary conditions the P^{2}NDFT is nothing else than the Ewald summation technique [49]. It is well known that the Ewald splitting parameter α can be chosen in a way that leads to an optimal coupling between the number of particles N and the mesh size $\hat{\boldsymbol{m}}$. For this optimal choice of the splitting parameter $\alpha(N)$ the overall complexity for computing the Coulomb forces via Ewald summation results in $\mathcal{O}(N^{3/2})$. We present a detailed discussion about the optimal choice of α in Section 4.8.1. Thereby, we will also see that optimizing α for computing the potential results in a slightly worse scaling of $\mathcal{O}((N \log N)^{3/2})$.

However, P^{2}NDFT also supports partially periodic and non-periodic boundary conditions. Therefore, it can be seen as the natural generalization of Ewald summation to these kinds of periodicity. The most important step towards this generalization was the approximation with fast convergent Fourier series along non-periodic dimensions, cf. Section 4.4.7. We stress that a naive evaluation of the 2d-periodic long-range part (4.16) would take $\mathcal{O}(N^2 \hat{m}_0 \hat{m}_1)$ operations. Analogously, the 1d-periodic long-range part (4.19) would require $\mathcal{O}(N^2 \hat{m}_0)$ operations. Note that in both cases the computational work increases with N^2 no matter how the Ewald splitting parameter α is chosen. In contrast, the $\mathcal{O}(N|\mathcal{I}_{\hat{\boldsymbol{m}}}|)$ scaling of the P^{2}NDFT

long-range part makes it possible to derive an $N^{3/2}$ algorithms by optimal coupling of N and $\hat{\boldsymbol{m}}$. However, using NFFTs the P²NFFT yields a much better scaling also for mixed-periodic boundary conditions at the price of the NFFT approximation error and should be the method of choice for usual use cases.

4.7.3 Fast Ewald summation based on NFFT

The fast Ewald summation based on NFFT [68, 129] simply applies NFFTs for the fast computation of the long-range part in the the 3d-periodic Ewald splitting. Therefore, it can be interpreted as P²NFFT with 3d-periodic boundary conditions. Indeed, [68] has been the starting point for the development of P²NFFT with periodic boundary conditions. However, we stress that the complete P²NFFT framework includes many features that were not included in this previous work, especially mixed-periodic boundary conditions. Furthermore, our versatile PNFFT implementation [12] includes many features that were not available at former times. This includes interlacing, adapted algorithmic design for scaled nodes $\boldsymbol{x} \in \mathbf{S}\left[-1/2, 1/2\right)^3 \subset \left[-1/2, 1/2\right)^3$, fast NFFT based computation of the gradient and parallelization.

4.7.4 Particle-mesh Ewald

The Particle-Mesh Ewald (PME) method [37] can be interpreted as P²NFFT with 3d-periodic boundary conditions. Thereby, a non-oversampled NFFT is computed that uses Lagrangian interpolation instead of a window function. In addition, the NFFT deconvolution step (3.10) is omitted completely. Note that the Lagrangian interpolation can be interpreted as a non-smooth window function in the NFFT setting. This results in relatively large errors, especially in the gradients. Lagrangian interpolation is not part of the implemented windows in our PNFFT library [12]. Nowadays, particle-mesh Ewald is superseded by the smooth particle-mesh Ewald method that shows better convergence, especially in the gradients.

4.7.5 Smooth particle-mesh Ewald

The Smooth Particle-Mesh Ewald (SPME) method [48] can be interpreted as 3d-periodic P²NFFT with a special set of NFFT parameters. In their original paper [48] a non-oversampled NFFT with a B-spline window function was proposed. Furthermore, they used the analytic differentiation scheme for computing the field. It can be shown [23, Equation (4.1)] that the SPME influence function is equivalent to

$$\frac{\hat{R}_{\boldsymbol{k}}}{(\sum_{\boldsymbol{r}\in\mathbb{Z}^3} \hat{\varphi}_{\boldsymbol{k}_{\boldsymbol{r}}})^2} \approx \frac{\hat{R}_{\boldsymbol{k}}}{\hat{\varphi}_{\boldsymbol{k}}^2},$$

which shows its close connection to both P²NFFT and P³M. For the non-interlaced case the differences in accuracy between P³M and SPME have been found to be

practically irrelevant [117], see also [101, Figure 1]. Because of the very similar structure of P^{3}M and P^{2}NFFT we suppose the same behavior for our method. A combination of SPME and interlacing was presented in [31] and is called staggered mesh Ewald. Numerical tests in [101] showed, that the specialized form of the interlaced P^{3}M influence function gives rise to a remarkable accuracy gain in comparison to SPME. The same can be expected for P^{2}NFFT. A parallel version of SPME has been described in [63, Section 7.5].

4.7.6 Gaussian split Ewald

The k-space Gaussian split Ewald method [118] splits the exponential of $\hat{R}_{\boldsymbol{k}}$ into several factors and derives an algorithm that recapitulates exactly the steps of an NFFT with Gaussian window function. In our notation Gaussian split Ewald can be interpreted as an NFFT based fast Ewald summation with non-oversampled NFFT, the Gaussian window function (3.27) and analytic differentiation. Furthermore, the authors propose an adaption called real-space Gaussian split Ewald method that shifts the deconvolution step of the NFFT into real space where it is performed on the mesh. This variation of deconvolution is not yet included in the PNFFT module and, therefore, not yet part of the P^{2}NFFT framework.

4.7.7 Spectrally accurate Ewald

The Spectrally Accurate Ewald method [92, 93] can be interpreted as a P^{2}NFFT with oversampled NFFT and Gaussian window function. This method was proposed for 3d-periodic and 2d-periodic boundary conditions. In the 2d-periodic case our P^{2}NFFT differs from [93] in the choice of the regularization (4.17), see also Section 4.4.7 for a comparison of the different regularization approaches. Note that the Spectrally Accurate Ewald method only supports 3d- and 2d-periodic boundary conditions, whereas P^{2}NFFT is a common framework for 3d-, 2d-, 1d- and 0d-periodic boundary conditions. In [3] we presented extensive numerical tests of P^{2}NFFT that recapitulate the high accuracy tests of the spectrally accurate Ewald method in [92, 93]. In summary, P^{2}NFFT reaches the same accuracy at smaller grid sizes.

4.8 Complexity of Ewald summation and parameter selection

It seems to be common knowledge that plain Ewald summation scales as $\mathcal{O}(N^{3/2})$ for optimal choice of the Ewald splitting parameter α and particle-mesh methods scale as $\mathcal{O}(N \log N)$. In the following, we give a rigorous derivation of the asymptotic runtime for plain Ewald summation and P^{2}NFFT based on the error estimates of

[85]. Thereby, we strictly assure that the approximation errors of our algorithms do not increase with the system size N. In addition, we find some easy rules for P^{2}NFFT parameter selection in order to fulfill a prescribed error bound.

We assume a uniform distribution of N particles with uncorrelated positions in a cubic box of size $L \times L \times L$. It is natural to scale up the system size by increasing the box length L. Then, the uniform particle distribution implies $Q := \sum_{i=1}^{N} q_i^2 \sim N \sim L^3$ and, therefore, QL^{-3} is constant. Let $\Delta\phi_i = \phi_{i,\text{approx}} - \phi_{i,\text{exact}}$ and $\Delta\boldsymbol{E}_i = \boldsymbol{E}_{i,\text{approx}} - \boldsymbol{E}_{i,\text{exact}}$ be the potential and field errors per particle position $\boldsymbol{r}_i$. Then, we define the root mean square (rms) errors in the potentials and fields as

$$\Delta\phi = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\Delta\phi_i^2}, \qquad \Delta\boldsymbol{E} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\Delta\boldsymbol{E}_i\|^2}.$$

The rms errors of the energies and forces are defined analogously.

Let a prescribed rms potential accuracy $\varepsilon > 0$ given. In [85] the rms potential error is decomposed as $(\Delta\phi)^2 = (\Delta\phi^{\mathsf{sr}})^2 + (\Delta\phi^{\mathsf{lr}})^2$ and estimates of the short-range rms potential error $\Delta\phi^{\mathsf{sr}}$ and the long-range rms potential error $\Delta\phi^{\mathsf{lr}}$ are given as follows

$$\Delta\phi^{\mathsf{sr}} \approx \sqrt{\frac{Qr_\mathrm{c}}{L^3}}\frac{\exp(-\alpha^2 r_\mathrm{c}^2)}{\alpha^2 r_\mathrm{c}^2}, \qquad \Delta\phi^{\mathsf{lr}} \approx \frac{\alpha}{\pi^2}\sqrt{\frac{2Q}{k_\mathrm{c}^3}}\exp\left(-\frac{\pi^2 k_\mathrm{c}^2}{\alpha^2 L^2}\right). \tag{4.42}$$

Analogously, the rms field error is written as $(\Delta\boldsymbol{E})^2 = (\Delta\boldsymbol{E}^{\mathsf{sr}})^2 + (\Delta\boldsymbol{E}^{\mathsf{lr}})^2$. Estimates for the short- and long-range rms field errors are also given in [85] as

$$\Delta\boldsymbol{E}^{\mathsf{sr}} \approx 2\sqrt{\frac{Q}{L^3 r_\mathrm{c}}}\exp(-\alpha^2 r_\mathrm{c}^2), \qquad \Delta\boldsymbol{E}^{\mathsf{lr}} \approx \frac{\alpha}{\pi^2 L}\sqrt{\frac{32Q}{k_\mathrm{c}}}\exp\left(-\frac{\pi^2 k_\mathrm{c}^2}{\alpha^2 L^2}\right). \tag{4.43}$$

Remark 4.3. Error estimates for the energies and forces are due to [85] given by

$$\Delta U \approx \sqrt{\frac{Q}{N}}\Delta\phi, \qquad \Delta\boldsymbol{F} \approx \sqrt{\frac{Q}{N}}\Delta\boldsymbol{E}. \tag{4.44}$$

Since we claim QN^{-1} to be constant, we get the same asymptotic behavior as for the potentials and fields. Therefore, we restrict the following investigations to potentials and fields. □

The above mentioned error estimates can be explicitly solved for α, r_c or k_c using the Lambert-W function [33]. For our purposes it is enough to consider the positive branch of the Lambert-W function restricted to the interval $(0, \infty)$, which is given as the inverse function of $f\colon (0,\infty) \to (0,\infty)$, $f(x) = x\mathrm{e}^x$. Note that the function f and, therefore, its inverse W are strictly monotonically increasing. In the following, we summarize some important properties of the Lambert-W function that will become useful in the investigation of the error estimates afterward.

Lemma 4.4. *Let $A, B, n > 0$. The unique positive solution $t > 0$ of the equation $A = t^n \log(Bt)$ is given by*

$$t = \left(\frac{nA}{\mathrm{W}[nAB^n]}\right)^{1/n} = \frac{1}{B^n} \exp\left(\frac{1}{n}\,\mathrm{W}[nAB^n]\right)$$

Proof. The equation $A = t^n \log(Bt)$ can be transformed equivalently into

$$nAt^{-n} \exp(nAt^{-n}) = nAB^n$$

which is equivalent to

$$nAt^{-n} = \mathrm{W}[nAB^n].$$

Then, the first representation of the solution t follows immediately. Now, for $z = nAB^n > 0$ the implicit definition of the Lambert-W function $W[z]\exp(W[z]) = z$ can be rearranged to $\frac{z}{W[z]} = \exp(W[z])$, which implies the second representation. Since we only used equivalent transformations, the solution t is unique. ■

Lemma 4.5. *For all $a > 0$ we have $\lim_{x\to\infty} \mathrm{W}[ax](\log x)^{-1} = 1$.*

Proof. Let $f\colon (0,\infty) \to (0,\infty)$, $f(x) = x\mathrm{e}^x$. Note that f is strictly monotonically increasing and its inverse function is given by the positive branch of the Lambert-W function restricted to the interval $(0,\infty)$. For $x \geq 1$ we have $(1 - x^{-1}\log x)\mathrm{e}^x \leq \mathrm{e}^x \leq x\mathrm{e}^x$ which can be rewritten as

$$f(x - \log x) \leq \mathrm{e}^x \leq f(x), \qquad \text{for all } x \geq 1.$$

Application of the strictly increasing function W to the above inequality yields

$$x - \log x \leq \mathrm{W}[\mathrm{e}^x] \leq x, \qquad \text{for all } x \geq 1.$$

Since x is positive, we can divide this inequality by x and take the limit $x \to \infty$. Then, the upper and lower bound tend to 1 and we get $\lim_{x\to\infty} x^{-1}\,\mathrm{W}[\mathrm{e}^x] = 1$. Now, for arbitrary $a > 0$ we substitute $y = \log x + \log a$ and yield

$$\lim_{x\to\infty} \frac{\mathrm{W}[ax]}{\log x} = \lim_{y\to\infty} \frac{\mathrm{W}[\mathrm{e}^y]}{y - \log a} = \lim_{y\to\infty} \underbrace{\frac{\mathrm{W}[\mathrm{e}^y]}{y}}_{\to 1} \underbrace{\frac{y}{y - \log a}}_{\to 1} = 1.$$

■

Lemma 4.6. *For all $a > 0$ we get the right-sided limit* $\lim_{x\to 0+} x^{-1}\,\mathrm{W}[ax] = a$.

Proof. A simple rearrangement of the implicit definition $W[ax]\mathrm{e}^{W[ax]} = ax$ of $W[ax]$ yields $\lim_{x\to 0+} x^{-1}\,\mathrm{W}[ax] = \lim_{x\to 0+} a\exp(-\mathrm{W}[ax]) = a$. Thereby, the last step follows from $\lim_{x\to 0+}\mathrm{W}[x] = 0$ and the continuity of the exponential function. ■

4.8.1 Runtime model

In the following, we are mainly interested in the asymptotic behavior of the Ewald summation runtime for $N \to \infty$. Therefore, we define for two positive functions $f\colon (0,\infty) \to (0,\infty)$ and $g\colon (0,\infty) \to (0,\infty)$ the relations

$$\begin{aligned} f(x) \sim g(x) \quad \text{for } x \to 0+ \quad &:\Leftrightarrow \quad 0 < \lim_{x\to 0+} \frac{f(x)}{g(x)} < \infty, \\ f(x) \sim g(x) \quad \text{for } x \to +\infty \quad &:\Leftrightarrow \quad 0 < \lim_{x\to +\infty} \frac{f(x)}{g(x)} < \infty. \end{aligned}$$

Hereby, the notation $x \to 0+$ stands for the right-sided limit. Note that these two conditions imply in Landau notation the less restrictive property $f(x) \in \Theta(g(x))$ for $x \to 0+$ or $x \to +\infty$, respectively. Our complexity estimates are based on the commonly used runtime model of the Ewald summation [85, Appendix B]. Thereby, we assume a homogenous particle distribution with a constant particle density $\rho := NL^{-3}$. Then, the number of particles within a sphere of radius r_c is given by $\frac{4}{3}\pi r_\mathrm{c}^3\rho$. This gives the number of terms that must be summed up for each particle in order to compute the short-range interactions (4.11). This means, that the asymptotic runtime of all the short-range interactions can be written as

$$t_N^{\mathsf{sr}} \sim \frac{4}{3}\pi r_\mathrm{c}^3 \rho N \sim r_\mathrm{c}^3 N \quad \text{for } N \to \infty. \tag{4.45}$$

Furthermore, for $N \to \infty$ the asymptotic runtime of the long-range part with a spherical Fourier space cutoff k_c is given by

$$t_N^{\mathsf{lr}} \sim \begin{cases} N k_\mathrm{c}^3 & \text{for plain Ewald summation,} \\ N + k_\mathrm{c}^3 \log k_\mathrm{c}^3 + k_\mathrm{c}^3 & \text{for fast (NFFT-based) Ewald summation.} \end{cases} \tag{4.46}$$

Note that the runtime of the fast Ewald summation results from the complexity of the NFFT and its adjoint. Thereby, we assume that we can find a constant window cutoff parameter c_φ and oversampling $\boldsymbol{M} \odot \hat{\boldsymbol{m}}^{-1}$ such that the NFFT approximation error is negligible in comparison to $\Delta\phi^{\mathsf{lr}}$ and $\Delta\boldsymbol{E}^{\mathsf{lr}}$ for all $N \to \infty$.

4.8.2 Potential computation via Ewald summation

Let ε be a prescribed error bound. Using the error estimates (4.42) we get $\Delta\phi^{\mathsf{sr}} \leq \varepsilon$ and $\Delta\phi^{\mathsf{lr}} \leq \varepsilon$ for

$$r_{\mathrm{c}} \geq \frac{\sqrt{3}}{2\alpha}\sqrt{\mathrm{W}\left[\frac{4}{3}\left(\frac{Q}{L^3}\right)^{2/3}\frac{1}{\varepsilon^{4/3}}\frac{1}{\alpha^{2/3}}\right]} \qquad \sim \frac{1}{\alpha}\sqrt{\log\alpha^{-1}} \qquad \text{for } \alpha \to 0+, \tag{4.47}$$

$$k_{\mathrm{c}} \geq \frac{\sqrt{3}}{2}\frac{L\alpha}{\pi}\sqrt{\mathrm{W}\left[\frac{4}{3}\left(\frac{2\pi Q}{L^3}\right)^{2/3}\frac{1}{\varepsilon^{4/3}}\frac{1}{\alpha^{2/3}}\right]} \sim N^{1/3}\alpha\sqrt{\log\alpha^{-1}} \quad \text{for } \alpha \to 0+. \tag{4.48}$$

Hereby, we used that ε and QL^{-3} are constants. Furthermore, we substituted $L \sim N^{1/3}$ into the last relation. The limits for $\alpha \to 0+$ are given by Lemma 4.5. A crucial point is that we assume $\alpha \to 0+$ for $\mathbb{N} \to \infty$. It can be easily verified that constant α and $\alpha \to \infty$ result in a worse complexity at the end. Loosely speaking, the additional factor L^{-2} in the exponential of the long-range error (4.42) causes the Fourier space cutoff k_{c} to increase faster with $N \to \infty$ than the short-range cutoff r_{c}. Therefore, α has to decrease in order to compensate the growth of L. Plugging the bounds (4.47), (4.48) into the runtime model (4.45), (4.46) yields the asymptotic total runtime

$$t_N = t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim \alpha^{-3}\left(\log\alpha^{-1}\right)^{3/2} N + \alpha^3 N\left(\log\alpha^{-1}\right)^{3/2} N \quad \text{for } \alpha \to 0+.$$

Choosing $t_N^{\mathsf{sr}} \sim t_N^{\mathsf{lr}}$ gives the asymptotic optimal value if $\alpha \sim N^{-1/6}$, for $N \to \infty$. Note that the optimal choice of α indeed fulfills $\alpha \to 0+$ for $N \to \infty$. Finally, plugging the asymptotic optimal α into the total runtime $t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}}$ gives a complexity estimate of the Ewald summation at constant rms potential error

$$t_N \sim (N \log N)^{3/2} \quad \text{for } N \to \infty.$$

Note that this term scales worse than $N^{3/2}$ as proposed in [55]. The additional logarithmic factor originates from the prefactors of the exponential in (4.42) that depend on r_{c} and k_{c}. These prefactors have not been taken into consideration in [55].

4.8.3 Potential computation via P^{2}NFFT

The total runtime of P^{2}NFFT is given by (4.45), (4.46) as

$$t_N = t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim r_{\mathrm{c}}^3 N + N + k_{\mathrm{c}}^3 \log k_{\mathrm{c}}^3 + k_{\mathrm{c}}^3.$$

Note that N does not depend on α and k_c^3 is of lower order in comparison to $k_c^3 \log k_c^3$. Therefore, we need to set $r_c^3 N \sim k_c^3 \log k_c^3$ in order to get the minimal asymptotic runtime in dependence on α. Using (4.47) and (4.48) this is equivalent to

$$1 \sim \alpha^6 \log\left[\alpha^3 N \left(\log \alpha^{-1}\right)^{3/2}\right] \sim \alpha^6 \log(\alpha^3 N) + \alpha^6 \log\left(\log \alpha^{-1}\right)^{3/2},$$

where the last term is of lower order as long as α does not decay exponentially with N. Therefore, we use Lemma 4.4 to solve the equation $1 = \alpha^6 \log(\alpha^3 N)$ for α. This results in the asymptotic optimal α as

$$\alpha \sim \left(\frac{2}{\mathrm{W}[2N^2]}\right)^{1/6} \sim \frac{1}{\sqrt[6]{\log N}} \quad \text{for } N \to \infty.$$

Hereby, the limit of the Lambert-W function was computed with Lemma 4.5. Finally, this choice of α gives the complexity estimates

$$t_N^{\mathsf{sr}} \sim \sqrt{\log N} N (\log\log N)^{3/2},$$
$$t_N^{\mathsf{lr}} \sim \frac{1}{\sqrt{\log N}} N (\log\log N)^{3/2} \left[\log N + \underbrace{\log(\log N)^{-1/2} + \log(\log\log N)^{3/2}}_{\text{lower order}}\right],$$

for $N \to \infty$. We see that the asymptotic runtime of the short-range and long-range part is balanced and the complexity estimate of the total runtime is given by $t_N \sim N\sqrt{\log N}(\log\log N)^{3/2}$. Note that this is better than the commonly cited $\mathcal{O}(N \log N)$ scaling of particle-mesh methods.

4.8.4 Field computation via Ewald summation

Let ε be a prescribed error bound. Using the error estimates (4.43) we get $\Delta \boldsymbol{E}^{\mathsf{sr}} \leq \varepsilon$ and $\Delta \boldsymbol{E}^{\mathsf{lr}} \leq \varepsilon$ for

$$r_c \geq \frac{1}{2\alpha}\sqrt{\mathrm{W}\left[64\left(\frac{Q}{L^3}\right)^2 \frac{1}{\varepsilon^4}\alpha^2\right]} \sim 1 \qquad \text{for } \alpha \to 0+, \tag{4.49}$$
$$k_c \geq \frac{L\alpha}{2\pi}\sqrt{\mathrm{W}\left[4096\left(\frac{Q}{\pi L^3}\right)^2 \frac{1}{\varepsilon^4}\alpha^2\right]} \sim N^{1/3}\alpha^2 \quad \text{for } \alpha \to 0+.$$

Again, we used that ε and QL^{-3} are constants and substituted $L \sim N^{1/3}$ into the last relation. The limits for $\alpha \to 0+$ are given by Lemma 4.6. Therefore, the complexity of the total runtime is given by

$$t_N = t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim r_c^3 N + k_c^3 N \sim N + \alpha^6 N^2 \quad \text{for } \alpha \to 0+.$$

Choosing $t_N^{\mathsf{sr}} \sim t_N^{\mathsf{lr}}$ yields the asymptotic optimal value for $\alpha \sim N^{-1/6}$, $N \to \infty$. With this choice of α we get the following complexity estimate of the Ewald summation

$$t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim N \quad \text{for } N \to \infty.$$

This result is surprising since it seems to be much better than the well known $\mathcal{O}(N^{3/2})$ scaling of the Ewald field computation. However, application of Lemma 4.6 to (4.49) reveals that

$$\lim_{\alpha \to 0+} r_{\mathrm{c}} = 4QL^{-3}\varepsilon^{-2}. \tag{4.50}$$

Although this limit is constant in N, it increases tremendously for higher accuracy. We emphasize that the short-range computation time scales as $t_N^{\mathsf{sr}} \sim r_{\mathrm{c}}^3 N$, i.e., this part increases with ε^{-6} and is almost not computable also for modest accuracy ε. For example, if we wanted to increase accuracy by one digit, we would have to multiply the work of the short-range part by one million. Therefore, the theoretical $\mathcal{O}(N)$ complexity of the Ewald summation is unreachable in practical computations.

In the following, we show that it is possible to obtain a smaller constant, if we are willing to sacrifice the optimal $\mathcal{O}(N)$ complexity. At first, we introduce some less strict bounds for r_{c} and k_{c} that are more focused on the exponential decay of the errors. We assume that the argument in the exponential are at least 1 and get

$$\begin{aligned}
\Delta \boldsymbol{E}^{\mathsf{sr}} &\approx 2\sqrt{\frac{Q}{L^3 r_{\mathrm{c}}}} \exp(-\alpha^2 r_{\mathrm{c}}^2) && \overset{\alpha r_{\mathrm{c}} \geq 1}{\leq} \sqrt{\frac{Q \max\{1,\alpha\}}{L^3}} \exp(-\alpha^2 r_{\mathrm{c}}^2), \\
\Delta \boldsymbol{E}^{\mathsf{lr}} &\approx \frac{\alpha}{\pi^2 L}\sqrt{\frac{32Q}{k_{\mathrm{c}}}} \exp\left(-\frac{\pi^2 k_{\mathrm{c}}^2}{\alpha^2 L^2}\right) && \overset{\frac{\pi k_{\mathrm{c}}}{\alpha L} \geq 1}{\leq} \sqrt{\frac{32Q \max\{1,\alpha\}}{\pi^3 L^3}} \exp\left(-\frac{\pi^2 k_{\mathrm{c}}^2}{\alpha^2 L^2}\right).
\end{aligned}$$

Using these bounds we have $\Delta \boldsymbol{E}^{\mathsf{sr}} \leq \varepsilon$ and $\Delta \boldsymbol{E}^{\mathsf{lr}} \leq \varepsilon$ if we choose

$$r_{\mathrm{c}}(\alpha) \geq \frac{1}{\alpha} \max\left\{1, \sqrt{\log\left[2\left(\frac{Q}{L^3}\right)^{1/2}\frac{1}{\varepsilon}\right] + \tfrac{1}{2}\log\max\{1,\alpha\}}\right\} \sim \frac{1}{\alpha}, \tag{4.51}$$

$$k_{\mathrm{c}}(\alpha) \geq \frac{\alpha L}{\pi} \max\left\{1, \sqrt{\log\left[\left(\frac{32}{\pi^3}\frac{Q}{L^3}\right)^{1/2}\frac{1}{\varepsilon}\right] + \tfrac{1}{2}\log\max\{1,\alpha\}}\right\} \sim N^{1/3}\alpha. \tag{4.52}$$

Hereby, the asymptotics are considered for $\alpha \to 0+$. Then, the asymptotic runtime (4.45), (4.46) of the short-range and long-range interactions for $\alpha \to 0+$ are given by $t_N^{\mathsf{sr}} \sim \alpha^{-3} N$ and $t_N^{\mathsf{lr}} \sim \alpha^3 N^2$, respectively. Choosing $t_N^{\mathsf{sr}} \sim t_N^{\mathsf{lr}}$ yields the asymptotic optimal value with $\alpha \sim N^{-1/6}$, $N \to \infty$, and we get the well known complexity [85, Appendix B] of the Ewald field computations

$$t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim N^{3/2} \quad \text{for } N \to \infty.$$

For this choice of α the short-range cutoff increases like $r_{\mathrm{c}} \sim \sqrt{\log \varepsilon^{-1}} N^{1/6}$. Note the important differences to (4.50). This time r_{c} grows with $N^{1/6}$ but the dependency on the accuracy is dramatically reduced.

	Plain Ewald		Fast Ewald	
	Potential	Field	Potential	Field
$\alpha \sim$	$N^{-1/6}$	$N^{-1/6}$	$(\log N)^{-1/6}$	$(\log N)^{-1/6}$
$r_\mathrm{c} \sim$	$N^{1/6}\sqrt{\log N}$	$N^{1/6}$	$(\log N)^{1/6}\sqrt{\log\log N}$	$(\log N)^{1/6}$
$k_\mathrm{c} \sim$	$N^{1/6}\sqrt{\log N}$	$N^{1/6}$	$N^{1/3}(\log N)^{-1/6}\sqrt{\log\log N}$	$N^{1/3}(\log N)^{-1/6}$
$t_N \sim$	$(N\log N)^{3/2}$	$N^{3/2}$	$N\sqrt{\log N}(\log\log N)^{3/2}$	$N\sqrt{\log N}$

Table 4.1: Comparison of the asymptotic optimal parameters for computing the Ewald potential and field by plain Ewald summation and fast (NFFT-based) Ewald summation. These parameters are the Ewald splitting parameter α, the short-range cutoff r_c, the Fourier space cutoff k_c and the total runtime t_N. For all cases we assume a homogenous particle distribution with $N \sim Q \sim L^3$ for $N \to \infty$.

4.8.5 Field compuation via P^{2}NFFT

The runtime of P^{2}NFFT is given by (4.45), (4.46) as

$$t_N = t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim r_\mathrm{c}^3 N + N + k_\mathrm{c}^3 \log k_\mathrm{c}^3 + k_\mathrm{c}^3.$$

As in Section 4.8.3 we get the optimal complexity for $r_\mathrm{c}^3 N \sim k_\mathrm{c}^3 \log k_\mathrm{c}^3$. Using (4.51) and (4.52) this is equivalent to $1 \sim \alpha^6 \log(\alpha^3 N)$. This can be solved for α with Lemma 4.4 and yields the asymptotic optimal α as

$$\alpha \sim \left(\frac{2}{\mathrm{W}[2N^2]}\right)^{1/6} \sim \frac{1}{\sqrt[6]{\log N}} \quad \text{for } N \to \infty.$$

Hereby, the limit of the Lambert-W function was calculated with Lemma 4.5. Finally, this choice of α gives the asymptotic runtime

$$t_N^{\mathsf{sr}} \sim \sqrt{\log N} N, \qquad t_N^{\mathsf{lr}} \sim \frac{1}{\sqrt{\log N}} N \big(\log N - \underbrace{\tfrac{1}{2}\log\log N}_{\text{lower order}} \big).$$

We see that the asymptotic runtime of the short-range and long-range part is balanced and the complexity estimate of the total runtime is given by $t_N \sim t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}} \sim N\sqrt{\log N}$. Again, this is better than the commonly cited $\mathcal{O}(N \log N)$ scaling of particle-mesh methods.

Finally, Table 4.1 summarizes the asymptotic optimal parameters that we found in Sections 4.8.2–4.8.5.

4.8.6 Selection of P^{2}NFFT parameters

A crucial point for the success of a numerical algorithm is the determination of good parameters. This means that we aim to fulfill some prescribed error bounds at the

shortest runtime, which emphasizes the need for accurate error estimates. Error estimates for the P^{3}M have been derived in [39] and can also be used to determine the parameters of the 3d-periodic P^{2}NFFT. However, in this section we present an alternative parameter tuning based on the Ewald summation error estimates (4.42), (4.43) presented in Section 4.8. Our approach enables us to reach very high accuracy and can be generalized to non-periodic boundary conditions. For the sake of simplicity, we assume a cubic box of size $L \times L \times L$. The list of P^{2}NFFT parameters is given as follows:

- short-range cutoff $r_\mathrm{c} > 0$,
- Ewald splitting parameter $\alpha > 0$,
- NFFT grid size $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$,
- FFT grid size $\boldsymbol{M} = \boldsymbol{\sigma} \odot \hat{\boldsymbol{m}} \in 2\mathbb{N}^3$,
- NFFT window cutoff $c_\varphi \in \mathbb{N}$,
- extended box shape $\mathbf{H} \in \mathbb{R}^{3\times 3}$, and
- degree of smoothness $s \in \mathbb{N}$ of the regularization.

Thereby, we introduced the vector valued oversampling factor $\boldsymbol{\sigma} := \hat{\boldsymbol{m}} \odot \boldsymbol{M}^{-1} \in [1,\infty)^3$. We will see that tuning $\boldsymbol{\sigma}$ instead of $\boldsymbol{M}$ is more convenient since it stays constant for $N \to \infty$.

In the following, we present our parameter tuning for 3d-periodic boundary conditions. Afterward, we will extend this approach to non-periodic boundary conditions. The starting point of our parameter tuning is some sensible choice of short-range cutoff r_c. Thereby, r_c should be chosen such that the computation of the short-range interaction part does not get overwhelming. Next, we compute α and k_c from (4.42) such that $\Delta\phi^{\mathsf{sr}} = \varepsilon/\sqrt{2}$ and $\Delta\phi^{\mathsf{lr}} = \varepsilon/\sqrt{2}$ is fulfilled with a prescribed rms potential error bound ε. The resulting values of α and k_c are given by

$$\alpha = \frac{1}{r_\mathrm{c}}\sqrt{\mathrm{W}\left(\sqrt{\frac{Qr_\mathrm{c}}{L^3}}\frac{1}{\tilde{\varepsilon}}\right)}, \qquad k_\mathrm{c} = \frac{\sqrt{3}}{2}\frac{\alpha L}{\pi}\sqrt{\mathrm{W}\left[\frac{4}{3}\left(\frac{2}{\alpha\pi}\frac{Q}{L^3}\right)^{2/3}\frac{1}{\tilde{\varepsilon}^{4/3}}\right]}, \tag{4.53}$$

where $\tilde{\varepsilon} := \varepsilon/\sqrt{2}$. Note that (4.44) implies that the same formulas can be used with $\tilde{\varepsilon} := \sqrt{NQ^{-1}\varepsilon}/\sqrt{2}$ in order to ensure a rms energy bound $\Delta U \leq \varepsilon$. Alternatively, if ε was the rms field error bound we would have computed α and k_c from (4.43) such that $\Delta\boldsymbol{E}^{\mathsf{sr}} = \varepsilon/\sqrt{2}$ and $\Delta\boldsymbol{E}^{\mathsf{lr}} = \varepsilon/\sqrt{2}$, i.e.,

$$\alpha = \frac{1}{r_\mathrm{c}}\sqrt{\ln\left(\sqrt{\frac{Q}{r_\mathrm{c}L^3}}\frac{2}{\tilde{\varepsilon}}\right)}, \qquad k_\mathrm{c} = \frac{\alpha L}{2\pi}\sqrt{\mathrm{W}\left[\left(\frac{64\alpha}{\pi^3}\frac{Q}{L^3}\right)^2\frac{1}{\tilde{\varepsilon}^4}\right]}. \tag{4.54}$$

Again by (4.44) the same formulas can be used with $\tilde{\varepsilon} := \sqrt{NQ^{-1}\varepsilon}/\sqrt{2}$ to ensure the rms field error bound $\Delta\boldsymbol{F} \leq \varepsilon$. In [93, Figure 7] it was shown numerically that these formulas also hold for the mixed-periodic case. Note that k_c represents a radial Fourier space cutoff while the NFFT grid size $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$ represents the edge

lengths of a box shaped cutoff scheme. Therefore, we choose the components of $\hat{\boldsymbol{m}} = (\hat{m}_0, \hat{m}_1, \hat{m}_2)$ as

$$\hat{m}_t = 2\lceil k_\mathrm{c} \rceil, \quad t = 0, 1, 2. \tag{4.55}$$

Next, we need to tune the NFFT parameters such that the NFFT approximation error is negligible in comparison to the prescribed approximation error ε of the truncated Ewald sum. More precisely, these parameters are the FFT mesh size $\boldsymbol{M} \in 2\mathbb{N}^3$ and the window cutoff parameter c_φ. We claim that these two parameters must be tuned only for a small test system. For larger particle numbers the same accuracy can be achieved by keeping the oversampling factor $\boldsymbol{\sigma} = \boldsymbol{M} \odot \hat{\boldsymbol{m}}^{-1} \in [1, \infty)^3$ and the window cutoff c_φ constant. In Section 4.11.2 we give some numerical evidence that this is true. At the moment the parameters of the NFFT are not tuned automatically but the development of an automatic NFFT parameter tuning is subject to recent research [102, 103]. Finally, the extended box shape can be chosen as $\mathbf{H} = \operatorname{diag}(L, L, L)$ and the degree of smoothness s is not present for the 3d-periodic case.

Once the parameters for the 3d-periodic P^{2}NFFT are found, we can use the following heuristic to find a parameter set for 2d-, 1d-, and 0d-periodic boundary conditions. This heuristic was proposed in [3] and we only give a short summary of the approach. The key idea is to keep the NFFT parameters c_φ, $\boldsymbol{\sigma}$ from the 3d-periodic case and define the grid size $\hat{\boldsymbol{m}}$ in such a way that the number of grid points per volume in the extended box $\mathbf{H}\big[-1/2, 1/2\big)^3$ remains constant. More precisely, we set

$$\hat{m}_t = \begin{cases} 2\lceil k_\mathrm{c} \rceil & \text{if the } t\text{-th dimension is periodic,} \\ 4\sqrt{3-p}\lceil k_\mathrm{c} \rceil + P & \text{if the } t\text{-th dimension is non-periodic.} \end{cases} \tag{4.56}$$

Hereby, p denotes the number of periodic dimensions and the so-called transition grid size $P \in 2\mathbb{N}$ denotes a small number of extra grid points that are introduced within the support of the transitions T_{k_0,k_1}, T_{k_0}, T in (4.17),(4.20), and (4.23). Note that the additional factor $2\sqrt{3-p}$ in the non-periodic case results from the period of the regularizations (4.17), (4.20), and (4.23). Thereby, $\sqrt{3-p}$ reflects the scaling of a $(3-p)$-dimensional ball into a $(3-p)$-dimensional cube. Next, the extended box shape $\mathbf{H} = \operatorname{diag}(H_0, H_1, H_2)$ is given by

$$H_t = L\tfrac{\hat{m}_t}{2\lceil k_\mathrm{c} \rceil}. \tag{4.57}$$

Note that for a given transition grid size P the NFFT grid size $\hat{\boldsymbol{m}}$ and the extended box shape $\mathbf{H}$ can be easily computed. Our numerical test in Section 4.11.3 confirm the claim that P stays constant for $N \to \infty$ and, therefore, does not effect the complexity of the P^{2}NFFT.

The parameter selection can be summarized in the following easy steps:

1. Choose the short-range cutoff r_c.

2. Calculate the Ewald splitting parameter α and the NFFT grid size $\hat{\boldsymbol{m}}$ by (4.53)–(4.55).
3. Determine the NFFT oversampling factor $\boldsymbol{\sigma}$, and the window cutoff c_φ numerically for a small test system.
4. Set the degree of smoothness $s := 10$ and tune the regularization grid size P numerically for a small test system.
5. Adjust the grid size $\hat{\boldsymbol{m}}$ according to (4.56) and set the extended box shape $\mathbf{H}$ according to (4.57).
6. Apply the NFFT oversampling factor $\boldsymbol{\sigma}$ to the adjusted grid size $\hat{\boldsymbol{m}}$ to get the adjusted FFT grid size $\boldsymbol{M} = \boldsymbol{\sigma} \odot \hat{\boldsymbol{m}}$.

Once all parameters have been determined we can execute P^{2}NFFT for a small test system and record the runtime $t_N^{\mathsf{sr}} \sim r_\mathrm{c}^3 N$ of the short-range computations and the runtime $t_N^{\mathsf{lr}} \sim k_\mathrm{c}^3 \log k_\mathrm{c}^3$ of the long-range computations. With these two times we can easily extrapolate the value for r_c that minimizes the total runtime $t_N^{\mathsf{sr}} + t_N^{\mathsf{lr}}$ for any particle number N.

Remark 4.7. The asymptotic runtime $t_N^{\mathsf{lr}} \sim k_\mathrm{c}^3 \log k_\mathrm{c}^3$ of the long-range part is a rather crude approximation for small k_c. Indeed, the runtime of the long-range part accumulates from many steps that have different asymptotic runtime. The most time consuming parts are given by the matrix-decomposition (3.20) of the NFFT. These are the deconvolution step, the FFT, and the discrete convolution step. In order to get a good prediction, we must measure and extrapolate the runtime of these steps individually. This is possible in our implementation, since all these times are available by the PNFFT timer interface. □

4.9 Parallel P^{2}NFFT

Because of the highly modularized structure of the P^{2}NFFT framework it is easy to substitute every module with its parallel counterpart. Again we assume a process mesh $\mathcal{P}_{\boldsymbol{P}}$ of $P_0 \times P_1 \times P_2$ processes and a parallel block decomposition as needed for the PNFFT, see Section 3.3.1.

Our parallel P^{2}NFFT framework starts with a parallel forward sort that assures the following two conditions. First, we distribute the nodes $\boldsymbol{r}_j$ according to a block domain decomposition such that every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ holds the nodes $\boldsymbol{r}_j$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$. Second, every process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$ needs local copies of all the nodes that are involved in the calculation of the short-range Module 4.1. These are all the nodes $\boldsymbol{r}_i$ that fulfill $\|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\| \leq r_\mathrm{c}$ for any $\boldsymbol{n} \in \mathcal{S}_p$. We perform these two tasks at once using a fine-grained data distribution operation [71] that is implemented within the software library for parallel sorting algorithms [36]. After this parallel forward sort, we can simply apply the short-range interaction Module 4.1 locally on each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ as given in the parallel short-range interaction Module 4.5.

Module 4.5 Parallel short-range interactions for each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$

Input:
- local number of particles $|\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}|$
- primary box shape $\mathbf{L} \in \mathbb{R}^{3\times 3}$, $\det(\mathbf{L}) \neq 0$
- block of particle positions $\boldsymbol{r}_j \in \mathbf{L}^3$ and sources $q_j \in \mathbb{R}$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$
- number of periodic dimensions p
- short-range cutoff $r_\mathrm{c} > 0$
- short-range potential $\psi^{\mathsf{sr}} \colon \mathbb{R}^3 \to \mathbb{R}$

1: For all local particle positions $\boldsymbol{r}_j$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$, compute

$$\phi_j \leftarrow \sum_{i=1}^{N} q_i \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p: \\ \|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\|\le r_\mathrm{c}}}{}' \psi^{\mathsf{sr}}(\boldsymbol{r}_{ij}^{\boldsymbol{n}}), \qquad \boldsymbol{E}_j \leftarrow -\sum_{i=1}^{N} q_i \sum_{\substack{\boldsymbol{n}\in\mathcal{S}_p: \\ \|\boldsymbol{r}_{ij}^{\boldsymbol{n}}\|\le r_\mathrm{c}}}{}' \nabla\psi^{\mathsf{sr}}(\boldsymbol{r}_{ij}^{\boldsymbol{n}}),$$

by a linked cell algorithm; see [70, Chap. 8.4] or [63, Chap. 3].

Output:
- block of approximated short-range potential $\phi_j \approx \phi_{\mathcal{S}_p}^{\mathsf{sr}}(\boldsymbol{r}_j) \in \mathbb{R}$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$
- block of approximated short-range field $\boldsymbol{E}_j \approx \boldsymbol{E}_{\mathcal{S}_p}^{\mathsf{sr}}(\boldsymbol{r}_j) \in \mathbb{R}^3$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$

Arithmetic cost: $\mathcal{O}\left(|\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}|\right)$

Module 4.6 Parallel self interactions for each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$

Input:
- block of sources $q_j \in \mathbb{R}$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$
- self potential $\psi^{\mathsf{self}} \in \mathbb{R}$

1: For all local $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$ compute

$$\phi_j^{\mathsf{self}} \leftarrow \psi^{\mathsf{self}} q_j.$$

Output:
- block of self potential $\phi^{\mathsf{self}}(\boldsymbol{r}_j) \in \mathbb{R}$, $j \in \mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}$

Arithmetic cost: $\mathcal{O}\left(|\mathcal{N}_{\boldsymbol{P}}^{\boldsymbol{r}}|\right)$

The combination of parallel domain decomposition with the serial self interaction Module 4.2 is straightforward and directly leads to the parallel self interaction Module 4.6.

Now, we present a parallel counterpart of the serial long-range interaction Module 4.3. Therein, we introduced the extended box shape $\mathbf{H} \in \mathbb{R}^{3\times 3}$ in order to deal with long-range interactions subject to non-periodic boundary conditions. We assumed that the extended box is at least a factor $2\sqrt{3-p}$ larger than the primary box shape $\mathbf{L} \in \mathbb{R}^{3\times 3}$ in each non-periodic dimension. This leads to the problem, that the NFFT input nodes fulfill $\boldsymbol{x}_j := \mathbf{H}^{-1}\boldsymbol{r}_j \in \mathbf{H}^{-1}\mathbf{L}\big[-1/2, 1/2\big)^3 \subset \big[-1/2, 1/2\big)^p \times (3-p)^{-1/2}[-1/4, 1/4)^{3-p} \subset \big[-1/2, 1/2\big)^3$. This means, if our parallel NFFT worked on a simple block decomposition of the whole unit cube $\big[-1/2, 1/2\big)^3$, many processes would end up without any nodes. For example, in the 0d-periodic case more than seven eighth of the unit cube $\big[-1/2, 1/2\big)^3$ remain empty. Parallel decomposition of these empty regions would give raise to severe load balancing problems. Fortunately, we already developed an algorithmic adaption for this case in terms of the truncated unit cube $\mathbf{S}\big[-1/2, 1/2\big)^3$, see also (3.19) for the definition of the resulting pruned FFT. By the definition of $\mathbf{H}$ we get a diagonal matrix $\mathbf{S} := \mathbf{H}^{-1}\mathbf{L}$. More precisely, depending on the number p of dimensions with periodic boundary conditions we have

$$p=3\colon\ \mathbf{S} = \operatorname{diag}(1,1,1), \qquad p=2\colon\ \mathbf{S} = \operatorname{diag}\left(1,1,\frac{L_{22}}{H_{22}}\right),$$
$$p=1\colon\ \mathbf{S} = \operatorname{diag}\left(1,\frac{L_{11}}{H_{11}},\frac{L_{22}}{H_{22}}\right), \qquad p=0\colon\ \mathbf{S} = \operatorname{diag}\left(\frac{L_{00}}{H_{00}},\frac{L_{11}}{H_{11}},\frac{L_{22}}{H_{22}}\right).$$

In the following, the parallel NFFT frameworks operate on a three-dimensional block decomposition of the truncated unit cube $\mathbf{S}\big[-1/2, 1/2\big)^3$.

At first, we compute the structure factors

$$\hat{S}_{\boldsymbol{k}} = \sum_{s\in\mathcal{P}_{\mathbf{P}}} \sum_{j\in\mathcal{N}^s_{\boldsymbol{P}}} q_j \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}, \quad \boldsymbol{k}\in\mathcal{I}^r_{\hat{\boldsymbol{m}},\boldsymbol{P}},$$

by the PNFFT$^{\mathsf{H}}$ Framework 3.2 with a diagonal node scaling matrix $\mathbf{S} = \mathbf{H}^{-1}\mathbf{L}$. Thereby, the formal order of summation was chosen in order to reflect the parallel data decomposition. The convolution in frequency domain is a simple point-wise multiplication and is performed locally for each process $\boldsymbol{r}\in\mathcal{P}_{\mathbf{P}}$, i.e.,

$$\hat{a}_{\boldsymbol{k}} := \hat{S}_{\boldsymbol{k}}\hat{R}_{\boldsymbol{k}}, \quad \boldsymbol{k}\in\mathcal{I}^r_{\hat{\boldsymbol{m}},\boldsymbol{P}}.$$

Hereby, the Fourier coefficients $\hat{R}_{\boldsymbol{k}}$ of the regularization R have been precomputed corresponding to the number p of periodic dimensions, see Sections 4.4.2–4.4.5 for

Module 4.7 Parallel long-range interactions for each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$

Input:

- primary box shape $\mathbf{L} \in \mathbb{R}^{3\times 3}$, $\det(\mathbf{L}) \neq 0$
- block of particle positions $\boldsymbol{r}_j \in \mathbf{L}\big[-1/2, 1/2\big)^3$ and sources $q_j \in \mathbb{R}$, $j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}$
- mesh size $\hat{\boldsymbol{m}} \in 2\mathbb{N}^3$
- block of Fourier coefficients $\hat{R}_{\boldsymbol{k}} \in \mathbb{C}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}$
- extended box shape $\mathbf{H} \in \mathbb{R}^{3\times 3}$, $\det(\mathbf{H}) \neq 0$

1: For all particle positions $\boldsymbol{r}_j$ compute the rescaled coordinates

$$\boldsymbol{x}_j \leftarrow \mathbf{H}^{-1}\boldsymbol{r}_j \in \mathbf{H}^{-1}\mathbf{L}\big[-1/2, 1/2\big)^3.$$

2: For all mesh points $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}$ compute

$$\hat{S}_{\boldsymbol{k}} \leftarrow \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} \sum_{i\in\mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}} q_i \mathrm{e}^{+2\pi \mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_i}$$

by a three-dimensional PNFFT$^{\mathsf{H}}$ (Framework 3.2) with node scaling matrix $\mathbf{S} = \mathbf{H}^{-1}\mathbf{L}$.

3: For all mesh points $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}$ compute the point-wise products

$$\hat{a}_{\boldsymbol{k}} \leftarrow \hat{R}_{\boldsymbol{k}}\hat{S}_{\boldsymbol{k}}.$$

4: For all rescaled particle positions $\boldsymbol{x}_j$, compute

$$\phi_j \leftarrow \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k}\in\mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}} \hat{a}_{\boldsymbol{k}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j}, \qquad \boldsymbol{E}_j \leftarrow -\mathbf{H}^{-\mathsf{T}} \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k}\in\mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}} \hat{a}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}^{\mathsf{T}}\boldsymbol{x}_j},$$

by a three-dimensional PNFFT (Framework 3.1) with node scaling matrix $\mathbf{S} = \mathbf{H}^{-1}\mathbf{L}$.

Output:

- block of approximated long-range potential $\phi_j \approx \phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \in \mathbb{R}$, $j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}$
- block of approximated long-range field $\boldsymbol{E}_j \approx \boldsymbol{E}^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \in \mathbb{R}^3$, $j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}$

Arithmetic cost: $\mathcal{O}\left(|\mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}| + |\mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}| \log |\mathcal{I}_{\hat{\boldsymbol{m}}}|\right)$

Framework 4.8 Parallel P^{2}NFFT for each process $\boldsymbol{r} \in \mathcal{P}_{\mathbf{P}}$

Input:

- short-range potential $\psi^{\mathsf{sr}}\colon \mathbb{R}^3 \to \mathbb{R}$
- self potential $\psi^{\mathsf{self}} \in \mathbb{R}$
- precomputed Fourier coefficients $\hat{R}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}$

1: Assign the local nodes $\boldsymbol{r}_j$, $j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}$, and the associated neighboring particles within the short-range cutoff radius r_{c} by a parallel forward sort.

2: Apply the parallel short-range interaction Module 4.5.

3: Apply the parallel long-range interaction Module 4.7.

4: Apply the parallel self interaction Module 4.6.

5: Restore the initial particle distribution by a parallel backward sort.

Output:

- block of approximated potential $\phi_{\mathcal{S}_p}(\boldsymbol{r}_j)$, $j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}$
- block of approximated field $\boldsymbol{E}_{\mathcal{S}_p}(\boldsymbol{r}_j)$, $j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}$

details. Afterward, the approximated long-range potential and field is computed by the PNFFT Framework 3.1,

$$\phi^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \approx \sum_{\boldsymbol{s}\in\mathcal{P}_{\mathbf{P}}} \sum_{\boldsymbol{k}\in\mathcal{I}^{\boldsymbol{s}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}} \hat{a}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}_j}\,, \quad j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}\,,$$

$$\boldsymbol{E}^{\mathsf{lr}}_{\mathcal{S}_p}(\boldsymbol{r}_j) \approx -\mathbf{H}^{-\mathsf{T}} \sum_{\boldsymbol{s}\in\mathcal{P}_{\mathbf{P}}} \sum_{\boldsymbol{k}\in\mathcal{I}^{\boldsymbol{s}}_{\hat{\boldsymbol{m}},\boldsymbol{P}}} \hat{a}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}_j}\,, \quad j \in \mathcal{N}^{\boldsymbol{r}}_{\boldsymbol{P}}\,.$$

In summary we obtain the parallel long-range interaction Module 4.7.

Finally, we use the fine-grained data distribution operation from the parallel sorting library to revert the particle redistribution that was caused by the parallel forward sort. Altogether, we obtain the parallel P^{2}NFFT Framework 4.8 for the massively parallel computation of the P^{2}NFFT.

4.10 The parallel P^{2}NFFT software library

We implemented the parallel P^{2}NFFT Framework 4.8 as part of a publicly available software library called ScaFaCoS (Scalable Fast Coulomb Solvers). The development code is distributed under a GNU General Public License and is freely available at [10]. In the following, we highlight selected features of our parallel P^{2}NFFT implementation in order to give an impression of its flexibility.

- Installing the library is easy. It is based on the common sequence of configure, make, and make install.
- P^{2}NFFT supports any kind of mixed-periodic boundary conditions.
- P^{2}NFFT can handle arbitrary non-orthogonal grids in the periodic dimensions.
- P^{2}NFFT supports interlacing, i$\boldsymbol{k}$-differentiation and ad-differentiation.
- P^{2}NFFT includes the NFFT-based fast summation method for 0d-periodic boundary conditions.
- P^{2}NFFT can be run with all the window functions that are available in PNFFT.
- We have a well working heuristic for parameter selection.
- P^{2}NFFT is build on top of the hardware adaptive libraries FFTW, PFFT and PNFFT.
- P^{2}NFFT is part of the ScaFaCoS library that offers a common interface to various fast Coulomb solvers. This makes it easy to benchmark against alternative methods and to compute reference data.

4.11 Numerical results

In the following, we provide selected results from the numerical tests of P^{2}NFFT published in [5, 1, 3].

4.11.1 Description of test systems

Our numerical tests have been performed with three different particle systems.

1. *Random distribution:* This test system consists of $N = 1000$ uniformly randomly distributed particles $\boldsymbol{r}_j \in [0,1]^3$ located in a cubic box of edge length $L = 1$. The charges are chosen oppositely $q_j = (-1)^j$ such that $\sum_{j=1}^{N} q_j = 0$ and $Q = \sum_{j=1}^{N} q_j^2 = N$. Larger test systems with similar distribution are generated in the steps $N = 1000 + 11000n, n = 0, \ldots, 9$. Thereby, we increase the box size at the rate $L = \sqrt[3]{N}$, i.e., we keep the particle density $\rho = NL^3$ at a constant level. A similar test system was proposed in [93, Sect. IV.A.] with a constant box size $L = 1$. However, using constant L only implies a rescaling of the real space cutoff r_c and the Ewald splitting parameter α at rate $\sqrt[3]{N}$ in order to get equivalent numerical results. In order to get comparable results we set the real space cutoff $r_{\text{cut}} = 0.62$ as given in [93, Sect. IV.A.] for this test case.
2. *Cloud wall distribution:* The cloud-wall model system, shown in Figure 4.8, consists of 300 particles, which represent two oppositely charged walls centered in a cubic box together with a diffuse cloud of charges. This ensures a strong long-range contribution in the potential. The periodic box was replicated 7 and 15 times in every direction of space to yield cubic boxes filled with 102 900 and

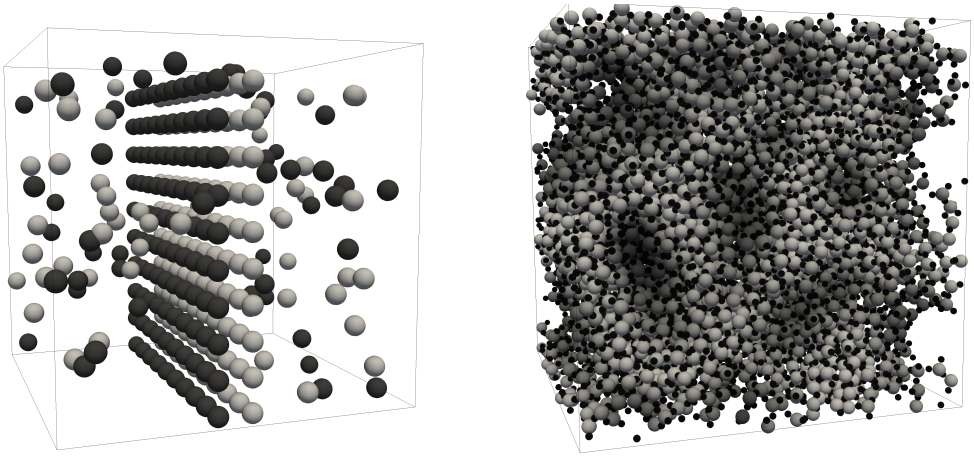

Figure 4.8: Left: The cloud-wall system (300 charges) consists of two oppositely charged walls in the center of the box and a surrounding diffuse cloud. The system was artificially created to contain a strong long-range field component.
Right: A Silica melt (12 960 charges) is sufficiently homogeneous while retaining a significant long-range contribution.

1 012 500 particles, respectively. Since these test cases represent periodically replicated systems, high precision reference value for potentials and forces have to be computed only for the smallest test case with $N = 300$ particles. Then, the reference values for larger systems are obtained by periodic replication.

3. *Silica melt:* This test system consists of a cubic box filled with 12 960 particles of a silica melt shown in Figure 4.8. It was taken from an MD simulation of a melting silica crystal using the BKS force field [25]. The overall charge neutral system consists of positively and negatively charged ions which are sufficiently homogeneously distributed, while the electrostatic potential still has a significant long-range contribution. For the scaling and benchmark runs the original silica melt system was replicated 4, 9, and 20 times in every direction of space to yield cubic boxes filled with 829 440, 9 447 840, and 103 680 000 particles, respectively.

4.11.2 Parameter selection and accuracy of 3d-periodic P^{2}NFFT

In this section we compute the parameters and achieved rms errors of 3d-periodic P^{2}NFFT for an prescribed rms potential error bound $\Delta\phi = 10^{-9}$. We use the randomly distributed particle system with a short-range cutoff $r_c = 0.62$. In Table 4.2 we give numerical evidence that the parameter tuning presented in Section 4.8.6 works very well for the random particle distribution defined in Section 4.11.1. Note that the achieved error is even better than the prescribed error bound of $\Delta\phi = 10^{-9}$.

N	L	$\hat{\boldsymbol{m}}$	$\boldsymbol{M}$	$\Delta\phi$	$\Delta\boldsymbol{E}$
1000	1.0	(26,26,26)	(32,32,32)	4.63e-10	3.48e-08
12000	2.29	(54,54,54)	(68,68,68)	4.81e-10	3.55e-08
23000	2.84	(66,66,66)	(82,82,82)	4.87e-10	3.56e-08
34000	3.24	(74,74,74)	(92,92,92)	4.92e-10	3.57e-08
45000	3.56	(82,82,82)	(102,102,102)	4.88e-10	3.55e-08
56000	3.83	(88,88,88)	(110,110,110)	4.89e-10	3.55e-08
67000	4.06	(92,92,92)	(114,114,114)	5.02e-10	3.57e-08
78000	4.27	(98,98,98)	(122,122,122)	4.91e-10	3.89e-08
89000	4.46	(102,102,102)	(126,126,126)	4.98e-10	4.31e-08
100000	4.64	(106,106,106)	(132,132,132)	4.94e-10	4.88e-08

Table 4.2: List of P^{2}NFFT parameters and achieved accuracies for the computation of Coulomb potentials ϕ and fields $\boldsymbol{E}$ between N randomly distributed particles in a cubic box of size $L \times L \times L$ subject to 3d-periodic boundary conditions. Thereby, the parameters were tuned to absolute root mean square potential error $\Delta\phi = 10^{-9}$ and are given as real space cutoff $r_c = 0.62$, Ewald splitting parameter $\alpha = 7.489225$, B-spline window function, and B-spline order 14 ($c_\varphi = 7$). The FFT grid size $\boldsymbol{M}$ and the order of the B-Spline have been tuned only for the smallest test case $N = 1000$. For the larger test cases we set $\boldsymbol{M} \approx 1.23\hat{\boldsymbol{m}}$ in correspondence to the tuned value.

This is due to the fact that the error estimates [85] have been derived for a radial cutoff scheme in Fourier space. In contrast, our FFT based fast solver uses a cubic cutoff scheme and, therefore, takes into account slightly more Fourier coefficients than necessary.

4.11.3 Parameter selection and accuracy for mixed periodicity

We repeated the tests from the previous Section with 2d-periodic boundary conditions and summarized the results in Table 4.3. Note that a similar test was given in [93, Figure 11] for the 2d-periodic spectral Ewald method. However, our results improve the findings of [93] in several ways. First, we provide numerical evidence that the error bound holds for all sets of parameters, while [93] only recorded the runtime of their method with some theoretically chosen parameters. Second, our grid size in the non-periodic dimension becomes smaller in comparison to [93] for large N. This is due to the fact that the transition grid size P does not increase with the number of particles N. However, our grid size in the periodic dimension is slightly large due to oversampling. Note that [93] used an oversampling factor of 4 in the non-periodic dimension. The window cutoff c_φ is comparable between the two

N	L	H	$\hat{\boldsymbol{m}}$	$\boldsymbol{M}$	$\Delta\phi$	$\Delta\boldsymbol{E}$
1000	1.0	3.0	(26,26,78)	(32,32,96)	8.59e-10	6.05e-08
12000	2.29	5.68	(54,54,134)	(68,68,166)	4.82e-10	3.45e-08
23000	2.84	6.81	(66,66,158)	(82,82,196)	5.06e-10	3.62e-08
34000	3.24	7.62	(74,74,174)	(92,92,216)	5.14e-10	3.63e-08
45000	3.56	8.15	(82,82,188)	(102,102,232)	5.22e-10	3.69e-08
56000	3.83	8.7	(88,88,200)	(110,110,246)	5.11e-10	3.59e-08
67000	4.06	9.27	(92,92,210)	(114,114,260)	5.11e-10	3.57e-08
78000	4.27	9.5	(98,98,218)	(122,122,270)	5.19e-10	3.95e-08
89000	4.46	9.98	(102,102,228)	(126,126,282)	5.13e-10	4.33e-08
100000	4.64	10.2	(106,106,234)	(132,132,288)	5.23e-10	4.92e-08

Table 4.3: List of P^{2}NFFT parameters and achieved accuracies for the computation of Coulomb potentials ϕ and fields $\boldsymbol{E}$ between N randomly distributed particles in a cubic box of size $L \times L \times L$ subject to 2d-periodic boundary conditions. Thereby, the parameters were tuned to absolute root mean square potential error $\Delta\phi = 10^{-9}$ and are given as short-range cutoff $r_c = 0.62$, Ewald splitting parameter $\alpha = 7.489225$, extended box size $L \times L \times H$, FFT grid size $\boldsymbol{M} \approx 1.23\hat{\boldsymbol{m}}$, B-Spline window function, and B-spline order 14 ($c_\varphi = 7$), degree of smoothness $s = 10$, and transition grid size $P = 32$. The FFT grid size $\boldsymbol{M}$, the order of the B-Spline and the transition grid size P have been tuned only for the smallest test case $N = 1000$.

methods, which shows that the spectral convergence of the Gaussian window does not outperform the B-Spline window. Last but not least, we were able to generate similar results for 1d-periodic boundary conditions presented in Table 4.4. To the best of our knowledge, this is the first time that a fast FFT based method is able to reach such high accuracy with 1d-periodic boundary conditions.

4.11.4 Strong scaling of 0d-periodic P^{2}NFFT on JUGENE

The following strong scaling tests have been performed with the two-point Taylor regularization presented in Section 4.7.1. Note that the parameter selection approach from Section 4.8.6 does not apply for this regularization. Instead, all parameters have been tuned manually such that the error bounds are fulfilled at minimal runtime. The tests presented in this Section have been performed on JUGENE – a Blue Gene/P architecture that has been explained in detail at the beginning of Section 2.10.1. Thereby, we use a relative error that is defined as follows. Let $\phi_{i,\text{approx}}$ denote the potentials that are calculated by the P^{2}NFFT Framework 4.8 and $\phi_{i,\text{exact}}$ the potentials that are calculated by any high accuracy reference method. For small

N	L	H	$\hat{\boldsymbol{m}}$	$\boldsymbol{M}$	$\Delta\phi$	$\Delta\boldsymbol{E}$
1000	1.0	4.15	(26,108,108)	(32,134,134)	8.41e-10	5.94e-08
12000	2.29	7.8	(54,184,184)	(68,228,228)	6.11e-10	4.08e-08
23000	2.84	9.39	(66,218,218)	(82,270,270)	5.72e-10	3.81e-08
34000	3.24	10.6	(74,242,242)	(92,298,298)	5.95e-10	3.85e-08
45000	3.56	11.3	(82,260,260)	(102,320,320)	6.48e-10	4.17e-08
56000	3.83	12.0	(88,276,276)	(110,340,340)	6.59e-10	4.23e-08
67000	4.06	12.8	(92,290,290)	(114,358,358)	6.52e-10	4.17e-08
78000	4.27	13.3	(98,304,304)	(122,374,374)	6.30e-10	4.34e-08
89000	4.46	13.8	(102,316,316)	(126,390,390)	6.24e-10	4.72e-08
100000	4.64	14.3	(106,326,326)	(132,402,402)	6.32e-10	5.26e-08

Table 4.4: List of P^{2}NFFT parameters and achieved accuracies for the computation of Coulomb potentials ϕ and fields $\boldsymbol{E}$ between N randomly distributed particles in a cubic box of size $L \times L \times L$ subject to 1d-periodic boundary conditions. Thereby, the parameters were tuned to absolute root mean square potential error $\Delta\phi = 10^{-9}$ and are given as short-range cutoff $r_\mathrm{c} = 0.62$, Ewald splitting parameter $\alpha = 7.489225$, extended box size $L \times H \times H$, FFT grid size $\boldsymbol{M} \approx 1.23\hat{\boldsymbol{m}}$, B-Spline window function, B-spline order 14 ($c_\varphi = 7$), degree of smoothness $s = 10$, and transition grid size $P = 44$. The FFT grid size $\boldsymbol{M}$, the order of the B-Spline and the transition grid size P have been tuned only for the smallest test case $N = 1000$.

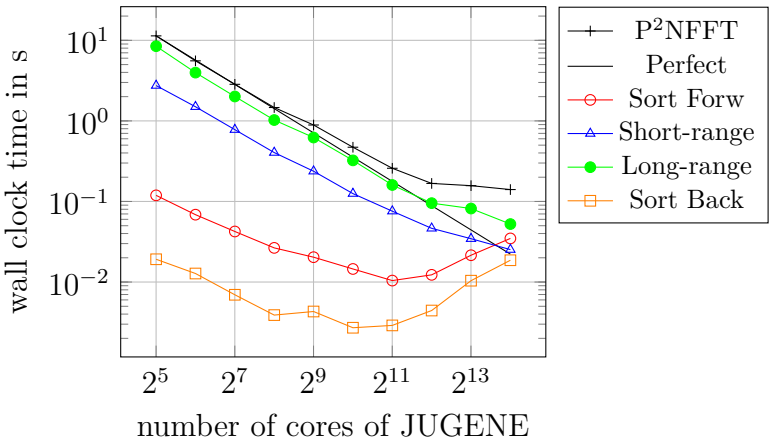

Figure 4.9: Runtime of the parallel P²NFFT Framework 4.8 for a silica melt with $N = 829\,440$ particles, relative rms potential error $\Delta\phi_{\text{rel}} = 10^{-5}$, short-range cutoff $r_c = 6.82$, box length $L = 248.24$, and extended box length $H = 873.58$, pruned FFT input size $\hat{\boldsymbol{m}} = (512, 512, 512)^{\mathsf{T}}$, oversampled FFT size $\boldsymbol{M} = (576, 576, 576)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (174, 174, 174)^{\mathsf{T}}$, and window cutoff parameter $c_\varphi = 4$. Times have been recorded on JUGENE up to 16 384 cores.

numbers of particles we chose the direct summation as reference method, while a fast method with higher accuracy was used to calculate the reference potentials for large systems. The relative rms potential error is given by

$$\Delta\phi_{\text{rel}} := \Big(\sum_{i=1}^{N} |\phi_{i,\text{approx}} - \phi_{i,\text{exact}}|^2 \Big)^{1/2} \Big(\sum_{i=1}^{N} |\phi_{i,\text{exact}}|^2 \Big)^{-1/2}. \tag{4.58}$$

For the following runtime measurements we manually tuned the parameters of the 0d-periodic P²NFFT in order to hold $\Delta\phi_{\text{rel}} < 10^{-5}$.

In Figure 4.9 we show the wall clock time measurements of the parallel P²NFFT Framework 4.8 for the silica melt test case with 829 440 particles on JUGENE using up to 16 384 cores. For comparison purposes we show the perfect strong scaling time (perfect) of the first recorded time. In addition, we add the wall clock time of the most time consuming parts of Framework 4.8. These are the forward sorting step 1 (Sort Forw), the short-range computation step 2 (Short-range), the long-range computation step 3 (Long-range), and the backward sorting step 5 (Sort Back). Note that the individual wall clock time measurements of the PNFFT$^{\mathsf{H}}$ step 2 and the PNFFT step 4 are given in Figure 3.7. We stress that the pruned FFT output size $\boldsymbol{m} = (174, 174, 174)^{\mathsf{T}}$ is significantly smaller than the oversampled FFT size $\boldsymbol{M} = (576, 576, 576)^{\mathsf{T}}$ for this test case, i.e., only 2.8% of the oversampled FFT output is necessary to compute the convolution steps of the PNFFT and its adjoint. This problem arises from the fact that the extended box length H is much larger

than the primary box length L and our parallel long-range interaction Module 4.7 scales all nodes as $\boldsymbol{x}_j = H^{-1}\boldsymbol{r}_j$. However, our algorithm takes care of this fact since we apply the parallel pruned NFFT Frameworks 3.1 and 3.2.

Although the near field and far field computations show good strong scaling, we observe that the sorting steps are the most limiting factor for strong scaling of the P^{2}NFFT algorithm. Remember, that the parallel sorting assures the block domain decomposition and generates the ghost cell particles that are necessary for the near field computations. Since the parallel sorting algorithm calls `MPI_Alltoallv` we see the typical linear increase of sorting time from 2048 to 16 384 processes. However, we stress that using 16 384 cores for this test case implies a very small number of 51 local nodes for every process. Therefore, generating the ghost particles that are necessary for the near field computation is rather costly in comparison to the actual near field computations itself.

Note that typical molecular dynamics software packages already have a high performance implementation of the near field part and use a block wise domain decomposition that perfectly fits the three-dimensional data decomposition of our P^{2}NFFT Framework 4.8. In this case we could redirect the short-range interaction Module 4.5 and self interaction Module 4.6 to the molecular dynamics software package and omit the whole sorting steps 1 and 5 of Framework 4.8. Then, we are left with the standalone long-range interaction Module 4.7. The corresponding runtime behavior of the long-range part is dominated by the parallel NFFTs, which were analyzed in detail in Figures 3.7, 3.8, and 3.9.

In Figure 4.10 we show the wall clock time measurements of the P^{2}NFFT Framework 4.8 for the silica melt test case with 9 447 840 particles on JUGENE using up to 65 536 cores. Furthermore, in Figure 4.11 we see the wall clock time measurements of P^{2}NFFT for the silica melt test case with 103 680 000 particles on JUGENE using up to 65 536 cores. A comparison of Figures 4.10 and 4.11 yields a better scaling of the larger test case since the quota of sorting decreases. In both test cases the wall clock time of the sorting steps increases almost linearly with the number of processes and prevents a good scaling of the overall runtime. However, we stress again that the number of local particles is rather small in comparison to the number of used processes in these test cases and typical molecular dynamics software may omit the sorting by using an appropriate domain decomposition.

4.11.5 Comparison of P^{2}NFFT to other fast Coulomb solvers

The ScaFaCoS library [10] is a collection of fast Coulomb solvers that can be accessed via a common interface. These solvers cover a broad range of different algorithmic approaches for computing the Coulomb potential (4.1) and field (4.2). All of these solvers are fast in the sense that the asymptotic runtime increases with the number of particles N as $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$. Without going too much into details we give a brief overview of the supported fast methods. For more details we refer to the

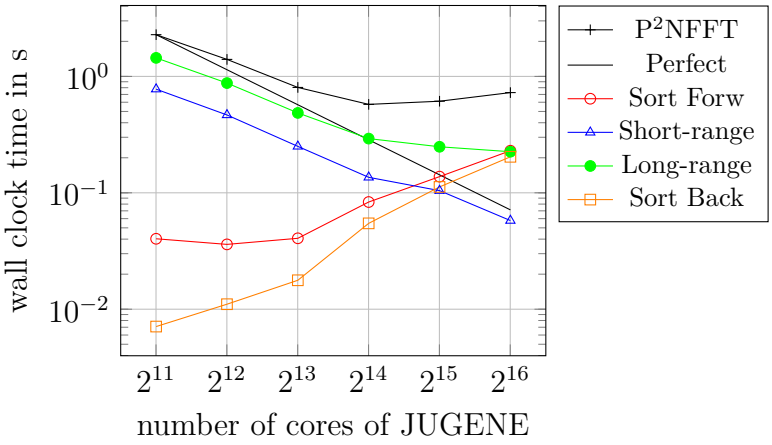

Figure 4.10: Runtime of the parallel P²NFFT Framework 4.8 for a silica melt with $N = 9\,447\,840$ particles, relative rms potential error $\Delta\phi_{\text{rel}} = 10^{-5}$, short-range cutoff $r_c = 7.62$, box length $L = 558.44$, and extended box length $H = 1949.73$, pruned FFT input size $\hat{\boldsymbol{m}} = (1024, 1024, 1024)^{\mathsf{T}}$, oversampled FFT size $\boldsymbol{M} = (1152, 1152, 1152)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (340, 340, 340)^{\mathsf{T}}$, and window cutoff parameter $c_\varphi = 4$. Times have been recorded on JUGENE up to 65 536 cores.

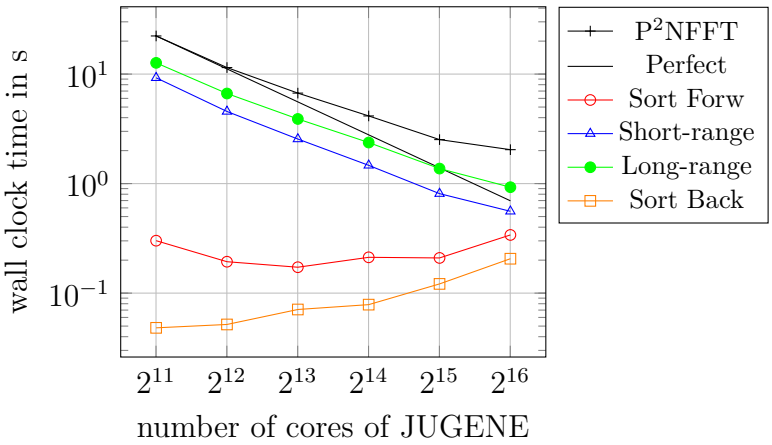

Figure 4.11: Runtime of the parallel P²NFFT Framework 4.8 for a silica melt with $N = 103\,680\,000$ particles, relative rms potential error $\Delta\phi_{\text{rel}} = 10^{-5}$, short-range cutoff $r_c = 8.43$, box length $L = 1241.2$, and extended box length $H = 4316.5$, pruned FFT input size $\hat{\boldsymbol{m}} = (2048, 2048, 2048)^{\mathsf{T}}$, oversampled FFT size $\boldsymbol{M} = (2304, 2304, 2304)^{\mathsf{T}}$, pruned FFT output size $\boldsymbol{m} = (674, 674, 674)^{\mathsf{T}}$, and window cutoff parameter $c_\varphi = 4$. Times have been recorded on JUGENE up to 65 536 cores.

nice overview given in [1]. Beside P^{2}NFFT the following methods are part of the ScaFaCoS project:

1. *P^{3}M:* The Particle-Particle–Particle-Mesh method [70] is essentially a special case of P^{2}NFFT as pointed out in Section 4.6. However, ScaFaCoS also includes a standalone implementation of the P^{3}M that originates from the ESPResSo [13, 21] software package. The parameter tuning is based on error estimates presented in [38, 39]. Also interlacing is supported [101].
2. *PP^{3}MG:* The Parallel Particle-Particle–Particle-Multigrid method [28] uses a splitting of the Coulomb interactions into a short-range and long-range part via B-Splines. The short-range part part is computed directly, while fast multigrid solvers [125] are applied for the computation of the long-range part. This results in an overall complexity of $\mathcal{O}(N)$.
3. *VMG:* Versatile Multigrid is another multigrid based solver that yields an asymptotic complexity of $\mathcal{O}(N)$.
4. *FMM:* The fast multipole method [62] replaces the individual interactions between particles with interactions between particles and particle groups via multipole expansions. This results in $\mathcal{O}(N)$ complexity. The FMM algorithm presented in [35, 81, 80] and its massively parallel implementation [79, 82] are part of the ScaFaCoS project. Note that this implementation of the FMM has been extended to periodic boundary conditions pursuant to [86].
5. *MEMD:* The method called Maxwell Equations Molecular Dynamics has been proposed in [95] and adapted for MD simulations in [105]. The implementation within ScaFaCoS was provided by the authors of [50]. This method results in $\mathcal{O}(N)$ complexity and provides the possibility for spatially varying dielectric properties in the system.
6. *PEPC:* The Pretty Efficient Parallel Coulomb Solver [132, 78] is based on the Barnes-Hut algorithm [24]. The recursive tree structure of the algorithm results in $\mathcal{O}(N \log N)$ complexity. Note that this method is not included in the following numerical comparison.

An extensive comparison of these methods in terms of numerical complexity, efficiency and parallel scalability has been published in [1] for 3d-periodic boundary conditions. Thereby, the relative rms potential error given in (4.58) was taken as common measure of accuracy. We want to highlight two results of the comparison given in Figures 4.12 and 4.13.

In Figure 4.12 we see a comparison of the required runtime in order to achieve a certain relative rms potential error bound. Note that most of the methods got into problems for accuracies better than 10^{-5}. The two noteworthy exceptions are FMM and P^{2}NFFT. Moreover, P^{2}NFFT outperforms all the other methods in terms of serial runtime. Note that the very different trend of P^{3}M originates from restrictions of the automatic P^{3}M parameter tuning that does not tune the short-range cutoff r_c for optimal runtime. Otherwise, the curves of P^{2}NFFT and P^{3}M are assumed to look similar.

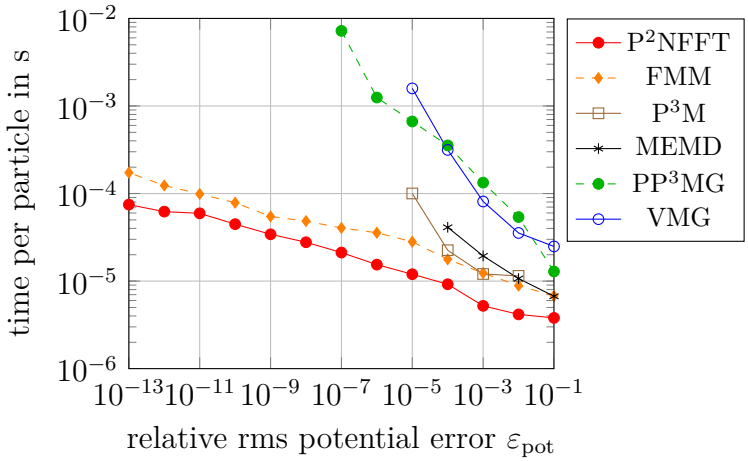

Figure 4.12: Required wall clock time per particle versus the relative rms potential error $\Delta\phi_{\text{rel}}$. The runs were performed for a cloud wall system with 102 900 charges on a single core of JUROPA.

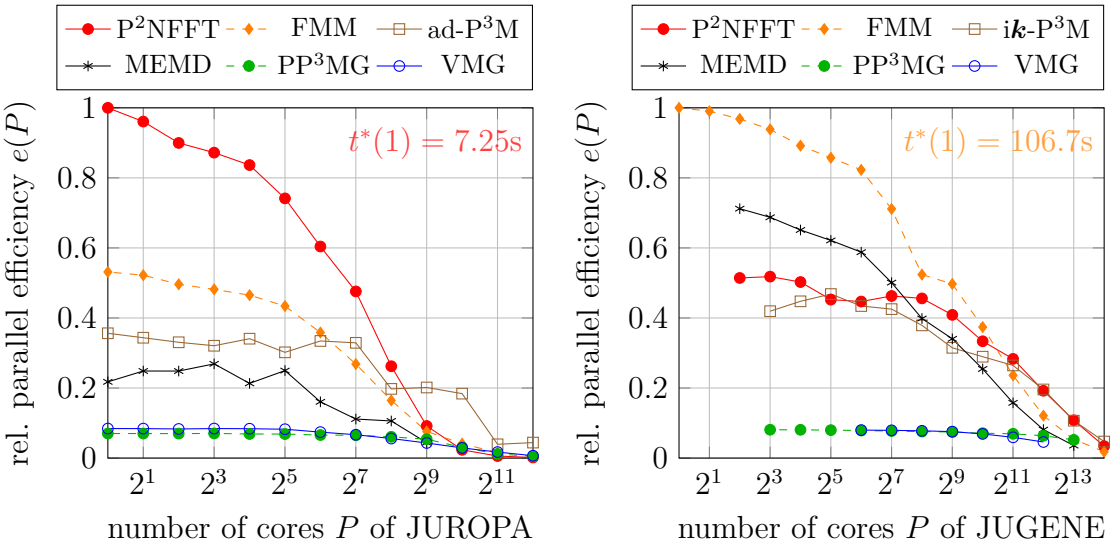

Figure 4.13: Relative parallel efficiency (4.59) for computing the cloud wall test case with 1 012 500 charges versus the number of cores on JUROPA (left, max. 4096 cores) and JUGENE (right, max. 16 384 cores). The relative rms potential error was tuned to $\Delta\phi_{\text{rel}} < 10^{-3}$ for all methods. Note that P³M with analytic differentiation was used on the left, while i$\boldsymbol{k}$-differentiation was used on the right. In contrast, P²NFFT uses analytic differentiation for all tests. The best runtime $t^*(1)$ with a single process have been achieved by P²NFFT on JUROPA and FMM on JUGENE.

Figure 4.13 summarizes the parallel scalability of the fast Coulomb solvers on two different hardware architectures, see Section 2.10.1 for the description of JUROPA and JUGENE. Let $t(P)$ denote the runtime dependent on the number of cores P. A good resolution of the wide plot range in Figure 4.13 is achieved by plotting the relative parallel efficiency

$$e(P) := \frac{t^*(1)}{t(P)} \frac{1}{P}, \tag{4.59}$$

where $t^*(1)$ denotes the runtime of the fastest method with one process.

On the JUROPA architecture, P^{2}NFFT and P^{3}M outperform the other methods. Both methods were used without interlacing and with analytic differentiation. Note that the significant difference in performance and scaling seen in Figure 4.13 stems solely from the different choice in the algorithms' specific parameters. In these simulations, P^{3}M uses an automatic parameter tuning method, whereas P^{2}NFFT timings were done with manually tuned parameters. The automatic tuning results in a larger short-range cutoff and, therefore, does not achieve the best single core performance. In contrast, P^{2}NFFT was optimized for best single core performance only and no further optimization of the short-range cutoff was done for higher core counts. Hence, the P^{2}NFFT has more work to do in Fourier space via the fast Fourier transforms, which is advantageous on small numbers of cores. On a large number of cores, this situation is reversed and P^{3}M as well as also other methods perform better for the same reason: More Fourier space calculations demand more global communication whereas in the case of P^{3}M more interactions are calculated in real space with a near field solver requiring less global communication. However, since P^{2}NFFT can use the same parameter set as P^{3}M, it is possible to switch parameters at the point of crossover. In this sense, the Fourier based solvers are the fastest ones over the full range of measurements.

On the JUGENE architecture FMM is the method with highest single core performance. In addition, due to its small memory footprint FMM was the only method that was able to run the test case on a single node. This time P^{2}NFFT used the same automatic parameter tuning as P^{3}M and, therefore, did not optimize the short-range cutoff r_c for minimal runtime. For large core counts P^{2}NFFT and P^{3}M show better scaling and outperform the FMM. Note that P^{3}M was invoked with i$\boldsymbol{k}$-differentiation, while P^{2}NFFT used analytic differentiation. Both methods did not use interlacing.

List of publications

Peer-reviewed articles

[1] A. Arnold, F. Fahrenberger, C. Holm, O. Lenz, M. Bolten, H. Dachsel, R. Halver, I. Kabadshow, F. Gähler, F. Heber, J. Iseringhausen, M. Hofmann, M. Pippig, D. Potts, and G. Sutmann: *Comparison of scalable fast methods for long-range interactions.* Phys. Rev. E, 88:063308, 2013.

[2] P. García-Risueño, J. Alberdi-Rodriguez, M.J.T. Oliveira, X. Andrade, M. Pippig, J. Muguerza, A. Arruabarrena, and A. Rubio: *A survey of the parallel performance and accuracy of poisson solvers for electronic structure calculations.* J. Comput. Chem., 35(6):427–444, 2014.

[3] F. Nestler, M. Pippig, and D. Potts: *Fast Ewald summation based on NFFT with mixed periodicity.* J. Comput. Phys., 285:280 – 315, 2015.

[4] M. Pippig: *PFFT - An extension of FFTW to massively parallel architectures.* SIAM J. Sci. Comput., 35:C213 – C236, 2013.

[5] M. Pippig and D. Potts: *Parallel three-dimensional nonequispaced fast Fourier transforms and their application to particle simulation.* SIAM J. Sci. Comput., 35:C411 – C437, 2013.

Proceedings

[6] F. Nestler, M. Pippig, and D. Potts: *NFFT based fast Ewald summation for various types of periodic boundary conditions.* In G. Sutmann, J. Grotendorst, G. Gompper, and D. Marx (eds.): *Computational Trends in Solvation and Transport in Liquids*, vol. 28 of *IAS-Series*, pp. 575 – 598, Jülich, 2015. Forschungszentrum Jülich GmbH.

[7] M. Pippig: *Scaling parallel fast Fourier transform on BlueGene/P.* In B. Mohr and W. Frings (eds.): *Jülich BlueGene/P Scaling Workshop 2010*, pp. 27 – 30, Jülich, 2010. Forschungszentrum Jülich. Technical Report.

[8] M. Pippig: *An efficient and flexible parallel FFT implementation based on FFTW.* In C. Bischof, H.G. Hegering, W.E. Nagel, and G. Wittum (eds.): *Competence in High Performance Computing 2010*, pp. 125–134. Springer Berlin Heidelberg, 2012.

[9] M. Pippig and D. Potts: *Particle simulation based on nonequispaced fast Fourier transforms.* In G. Sutmann, P. Gibbon, and T. Lippert (eds.): *Fast Methods for Long-Range Interactions in Complex Systems*, vol. 6 of *IAS-Series*, pp. 131 – 158, Jülich, 2011. Forschungszentrum Jülich GmbH.

Software libraries

[10] A. Arnold, M. Bolten, H. Dachsel, F. Fahrenberger, F. Gähler, R. Halver, F. Heber, M. Hofmann, J. Iseringhausen, I. Kabadshow, O. Lenz, and M. Pippig: *ScaFaCoS - Scalable Fast Coloumb Solvers*. `http://www.scafacos.de`.

[11] M. Pippig: *PFFT - Parallel FFT software library*. `https://github.com/mpip/pfft`.

[12] M. Pippig: *PNFFT - Parallel Nonequispaced FFT software library*. `https://github.com/mpip/pnfft`.

Bibliography

[13] *ESPResSo – Extensible Simulation Package for Research on Soft Matter.* `http://espressomd.org`.

[14] *JuGene: Jülich Blue Gene/P.* `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUGENE/JUGENE_node.html`.

[15] *JuQeen: Jülich Blue Gene/Q.* `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/JUQUEEN_node.html`.

[16] *JuRoPA: Jülich Research on Petaflop Architectures.* `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUROPA/JUROPA_node.html`.

[17] R.P. Agarwal and P.J.Y. Wong: *Error inequalities in polynomial interpolation and their applications*, vol. 262 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1993.

[18] A. Arnold and C. Holm: *MMM2D: A fast and accurate summation method for electrostatic interactions in 2D slab geometries.* Comput. Phys. Commun., 148:327 – 348, 2002.

[19] A. Arnold and C. Holm: *MMM1D: A method for calculating electrostatic interactions in one-dimensional periodic geometries.* J. Chem. Phys., 123:144103, 2005.

[20] A. Arnold, J. de Joannis, and C. Holm: *Electrostatics in periodic slab geometries. I.* J. Chem. Phys., 117:2496, 2002.

[21] A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Roehm, P. Košovan, and C. Holm: *ESPResSo 3.1: Molecular dynamics software for coarse-grained models.* In M. Griebel and M.A. Schweitzer (eds.): *Meshfree Methods for Partial Differential Equations VI*, vol. 89 of *Lecture Notes in Computational Science and Engineering*, pp. 1 – 23. Springer Berlin Heidelberg, 2013, ISBN 978-3-642-32978-4.

[22] O. Ayala and L.P. Wang: *Parallel implementation and scalability analysis of 3d fast Fourier transform using 2d domain decomposition.* Parallel Comput., 39:58 – 77, 2013.

[23] V. Ballenegger, J.J. Cerdà, and C. Holm: *How to convert SPME to P3M: Influence functions and error estimates.* J. Chem. Theory Comput., 8(3):936–947, 2012.

[24] J. Barnes and P. Hut: *A hierarchical $\mathcal{O}(n \log n)$ force-calculation algorithm.* Nature, 324(6096):446 – 449, 1986.

[25] B.W.H. van Beest and G.J. Kramer: *Force fields for silicas and aluminophosphates based on ab initio calculations.* Phys. Rev. Lett., 64(16):1955–1958, 1990.

[26] G. Beylkin: *On the fast Fourier transform of functions with singularities.* Appl. Comput. Harmon. Anal., 2:363 – 381, 1995.

[27] L.I. Bluestein: *A linear filtering approach to the computation of the discrete Fourier transform.* IEEE Trans AU, 18(4):451–455, 1970.

[28] M. Bolten: *Multigrid methods for structured grids and their application in particle simulation.* PhD thesis, Bergische Universität Wuppertal, Wuppertal, 2008.

[29] A. Bródka: *Ewald summation method with electrostatic layer correction for interactions of point dipoles in slab geometry.* Chem. Phys. Lett., 400:62 – 67, 2004.

[30] A. Bródka and P. Sliwinski: *Three-dimensional Ewald method with correction term for a system periodic in one direction.* J. Chem. Phys., 120:5518 – 5523, 2004.

[31] D.S. Cerutti, R.E. Duke, T.a. Darden, and T.P. Lybrand: *Staggered Mesh Ewald: An extension of the Smooth Particle-Mesh Ewald method adding great versatility.* J. Chem. Theory Comput., 5(9):2322, 2009.

[32] J.W. Cooley and J.W. Tukey: *An algorithm for machine calculation of complex Fourier series.* Math. Comput., 19:297 – 301, 1965.

[33] R.M. Corless, G.H. Gonnet, D.E.G. Hare, D.J. Jeffrey, and D.E. Knuth: *On the LambertW function.* Adv. Comput. Math., 5(1):329 – 359, 1996.

[34] K. Czechowski, C. Battaglino, C. McClanahan, K. Iyer, P.K. Yeung, and R. Vuduc: *On the communication complexity of 3d FFTs and its implications for exascale.* In *Proceedings of the 26th ACM International Conference on Supercomputing*, ICS '12, pp. 205 – 214, New York, NY, USA, 2012. ACM.

[35] H. Dachsel: *An error-controlled Fast Multipole Method.* J. Chem. Phys., 132(11):119901, 2010.

[36] H. Dachsel, M. Hofmann, and G. Rünger: *Library support for parallel sorting in scientific computations.* In *Proc. of the 13th International Euro-Par Conference*, vol. 4641 of *LNCS*, pp. 695–704. Springer, 2007.

[37] T. Darden, D. York, and L. Pedersen: *Particle mesh Ewald: An $n\log(n)$ method for Ewald sums in large systems.* J. Chem. Phys., 98:10089–10092, 1993.

[38] M. Deserno and C. Holm: *How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines.* J. Chem. Phys., 109:7678 – 7693, 1998.

[39] M. Deserno and C. Holm: *How to mesh up Ewald sums. II. An accurate error estimate for the Particle-Particle-Particle-Mesh algorithm.* J. Chem. Phys., 109:7694 – 7701, 1998.

[40] H.Q. Ding, R.D. Ferraro, and D.B. Gennery: *A portable 3d FFT package for distributed-memory parallel architectures.* In *Proceedings of the 7th SIAM Conference on Parallel Processing*, pp. 70 – 71, Philadelphia, 1995. SIAM.

[41] A.J.W. Duijndam and M.A. Schonewille: *Nonuniform fast Fourier transform.* Geophysics, 64:539 – 551, 1999.

[42] A. Dutt and V. Rokhlin: *Fast Fourier transforms for nonequispaced data.* SIAM J. Sci. Stat. Comput., 14:1368 – 1393, 1993.

[43] T.V.T. Duy and T. Ozaki: *OpenFFT - Parallel 3d FFT software package.* `http://www.openmx-square.org/openfft`, 2013.

[44] T.V.T. Duy and T. Ozaki: *A decomposition method with minimum communication amount for parallelization of multi-dimensional FFTs.* Computer Physics Communications, 185(1):153 – 164, 2014.

[45] H. Eggers, T. Knopp, and D. Potts: *Field inhomogeneity correction based on gridding reconstruction.* IEEE Trans. Med. Imag., 26:374 – 384, 2007.

[46] B. Elbel and G. Steidl: *Fast Fourier transform for nonequispaced data.* In C.K. Chui and L.L. Schumaker (eds.): *Approximation Theory IX*, pp. 39 – 46, Nashville, 1998. Vanderbilt University Press.

[47] M. Eleftheriou, J.E. Moreira, B.G. Fitch, and R.S. Germain: *A volumetric FFT for BlueGene/L.* In T.M. Pinkston and V.K. Prasanna (eds.): *HiPC*, vol. 2913 of *Lecture Notes in Computer Science*, pp. 194 – 203, Berlin, 2003. Springer.

[48] U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee, and L.G. Pedersen: *A smooth particle mesh Ewald method.* J. Chem. Phys., 103:8577 – 8593, 1995.

[49] P.P. Ewald: *Die Berechnung optischer und elektrostatischer Gitterpotentiale.* Ann. Phys., 369:253–287, 1921.

[50] F. Fahrenberger, Z. Xu, and C. Holm: *Simulation of electric double layers around charged colloids in aqueous solution of variable permittivity.* J. Chem. Phys., 141(6), 2014.

[51] B. Fang, Y. Deng, and G. Martyna: *Performance of the 3d FFT on the 6d network torus QCDOC parallel supercomputer.* Comp. Phys. Comm., 176:531 – 538, 2007.

[52] M. Fenn and G. Steidl: *Fast NFFT based summation of radial functions.* Sampl. Theory Signal Image Process., 3:1 – 28, 2004.

[53] J.A. Fessler and B.P. Sutton: *Nonuniform fast Fourier transforms using min-max interpolation.* IEEE Trans. Signal Process., 51:560 – 574, 2003.

[54] S. Filippone: *The IBM parallel engineering and scientific subroutine library.* In J. Dongarra, K. Madsen, and J. Wasniewski (eds.): *PARA*, vol. 1041 of *Lecture Notes in Computer Science*, pp. 199 – 206. Springer, 1995.

[55] D. Fincham: *Optimisation of the Ewald sum for large systems.* Mol. Simul., 13:1 – 9, 1994.

[56] K. Fourmont: *Non equispaced fast Fourier transforms with applications to tomography.* J. Fourier Anal. Appl., 9:431 – 450, 2003.

[57] M. Frigo and S.G. Johnson: *The design and implementation of FFTW3.* Proc. IEEE, 93:216 – 231, 2005.

[58] M. Frigo and S.G. Johnson: *FFTW, C subroutine library.* `http://www.fftw.org`, 2009.

[59] M. Frigo, C.E. Leiserson, H. Prokop, and S. Ramachandran: *Cache-oblivious algorithms.* In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 285 – 297, Washington, DC, 1999. IEEE Computer Society.

[60] A. Gholami and G. Biros: *AccFFT - Parallel FFT subroutine library.* `http://accfft.org`.

[61] L. Greengard and J.Y. Lee: *Accelerating the nonuniform fast Fourier transform.* SIAM Rev., 46:443 – 454, 2004.

[62] L. Greengard and V. Rokhlin: *A fast algorithm for particle simulations.* J. Comput. Phys., 73:325 – 348, 1987.

[63] M. Griebel, S. Knapek, and G. Zumbusch: *Numerical simulation in molecular dynamics*, vol. 5 of *Texts in Computational Science and Engineering.* Springer, Berlin, 2007.

[64] W. Gropp, E. Lusk, and R. Thakur: *Using MPI-2: Advanced Features of the Message-Passing Interface.* MIT Press, Cambridge, MA, USA, 1999.

[65] A. Grzybowski, E. Gwóźdź, and A. Bródka: *Ewald summation of electrostatic interactions in molecular dynamics of a three-dimensional system with periodicity in two directions.* Phys. Rev. B, 61:6706–6712, 2000.

[66] A. Gupta and V. Kumar: *The scalability of FFT on parallel computers.* IEEE Trans. Parallel Distributed Systems, 4:922 – 932, 1993.

[67] F.E. Harris: *Incomplete Bessel, generalized incomplete gamma, or leaky aquifer functions.* J. Comput. Appl. Math., 215:260 – 269, 2008.

[68] F. Hedman and A. Laaksonen: *Ewald summation based on nonuniform fast Fourier transform.* Chem. Phys. Lett., 425:142 – 147, 2006.

[69] M.T. Heideman, D.H. Johnson, and C.S. Burrus: *Gauss and the history of the fast Fourier transform.* Arch. Hist. Exact Sci., 34:265 – 277, 1985.

[70] R.W. Hockney and J.W. Eastwood: *Computer simulation using particles.* Taylor & Francis, Inc., Bristol, PA, USA, 1988.

[71] M. Hofmann and G. Rünger: *Fine-grained Data Distribution Operations for Particle Codes.* In M. Ropo, J. Westerholm, and J. Dongarra (eds.): *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 16th European PVM/MPI Users Group Meeting*, vol. 5759 of *LNCS*, pp. 54–63. Springer, 2009.

[72] P.H. Hünenberger: *Lattice-sum methods for computing electrostatic interactions in molecular simulations.* In *Simulation and theory of electrostatic interactions in solution*, vol. 17, pp. 17 – 83. ASCE, 1999.

[73] P.H. Hünenberger and J.A. McCammon: *Ewald artifacts in computer simulations of ionic solvation and ion–ion interaction: A continuum electrostatics study.* The Journal of Chemical Physics, 110(4):1856, 1999.

[74] D. Huybrechs: *On the Fourier extension of nonperiodic functions.* SIAM J. Numer. Anal., 47(6):4326 – 4355, 2010.

[75] Intel Corporation: *Intel math kernel library.* `http://software.intel.com/en-us/intel-mkl/`.

[76] J.I. Jackson, C.H. Meyer, D.G. Nishimura, and A. Macovski: *Selection of a convolution function for Fourier inversion using gridding.* IEEE Trans. Med. Imag., 10:473 – 478, 1991.

[77] H. Jagode: *Fourier transforms for the BlueGene/L communication network.* Master's thesis, The University of Edinburgh, 2006.

[78] F.J. Jülich Supercomputing Centre: *PEPC – The Pretty Efficient Coulomb Solver.* `http://www.fz-juelich.de/ias/jsc/pepc`.

[79] I. Kabadshow: *The Fast Multipole Method - Alternative gradient algorithm and parallelization.* Diplom (univ.), Technische Universität Chemnitz, Jülich, 2006.

[80] I. Kabadshow: *Periodic Boundary Conditions and the Error-Controlled Fast Multipole Method.* PhD thesis, Bergische Universität Wuppertal, Jülich, 2012.

[81] I. Kabadshow and H. Dachsel: *The error-controlled Fast Multipole Method for open and periodic boundary conditions.* In G. Sutmann, P. Gibbon, and T. Lippert (eds.): *Fast Methods for Long-Range Interactions in Complex Systems*, IAS-Series, pp. 85 – 113, Jülich, 2011. Forschungszentrum Jülich.

[82] I. Kabadshow and B. Lang: *Latency-optimized parallelization of the FMM near-field computations.* In *Computational Science - ICCS 2007*, vol. 4487 of *Lecture Notes in Computer Science*, pp. 716 – 722. Springer, 2007.

[83] J. Keiner, S. Kunis, and D. Potts: *NFFT 3.0, C subroutine library.* `http://www.tu-chemnitz.de/~potts/nfft`.

[84] J. Keiner, S. Kunis, and D. Potts: *Using NFFT3 - a software library for various nonequispaced fast Fourier transforms.* ACM Trans. Math. Software, 36:Article 19, 1 – 30, 2009.

[85] J. Kolafa and J.W. Perram: *Cutoff errors in the Ewald summation formulae for point charge systems.* Mol. Simul., 9(5):351 – 368, 1992.

[86] K.N. Kudin and G.E. Scuseria: *Revisiting infinite lattice sums with the periodic Fast Multipole Method.* J. Chem. Phys., 121(7):2886 – 2890, 2004.

[87] S. Kunis and S. Kunis: *The nonequispaced FFT on graphics processing units.* PAMM, Proc. Appl. Math. Mech., 12, 2012.

[88] S. Kunis and D. Potts: *Stability results for scattered data interpolation by trigonometric polynomials.* SIAM J. Sci. Comput., 29:1403 – 1419, 2007.

[89] S.W. de Leeuw, J.W. Perram, and E.R. Smith: *Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants.* Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 373:27 – 56, 1980.

[90] N. Li: *2DECOMP&FFT - Parallel FFT subroutine library.* `http://www.2decomp.org`.

[91] N. Li and S. Laizet: *2DECOMP & FFT - A highly scalable 2d decomposition library and FFT interface.* In *Cray User Group 2010 conference*, pp. 1 – 13, Edinburgh, Scotland, 2010.

[92] D. Lindbo and A.K. Tornberg: *Spectral accuracy in fast Ewald-based methods for particle simulations.* J. Comput. Phys., 230:8744 – 8761, 2011.

[93] D. Lindbo and A.K. Tornberg: *Fast and spectrally accurate Ewald summation for 2-periodic electrostatic systems.* J. Chem. Phys., 136:164111, 2012.

[94] M. Lyon: *Approximation error in regularized SVD-based Fourier continuations.* Appl. Numer. Math., 62(12):1790 – 1803, 2012.

[95] A.C. Maggs and V. Rosseto: *Local simulation algorithms for Coulombic interactions.* Phys. Rev. Lett., 88:196402, 2002.

[96] S. Matej, J.A. Fessler, and I.G. Kazantsev: *Iterative tomographic image reconstruction using Fourier-based forward and back-projectors.* IEEE Trans. Med. Imag., 23:401 – 412, 2004.

[97] P. Minary, J.A. Morrone, D.A. Yarne, M.E. Tuckerman, and G.J. Martyna: *Long range interactions on wires: a reciprocal space based formalism.* J. Chem. Phys., 121:11949 – 11956, 2004.

[98] P. Minary and M.E. Tuckerman: *A reciprocal space based method for treating long range interactions in ab initio and force-field-based calculations in clusters.* J. Chem. Phys., 110:2810 – 2821, 1999.

[99] P. Minary, M.E. Tuckerman, K.A. Pihakari, and G.J. Martyna: *A new reciprocal space based treatment of long range interactions on surfaces.* J. Chem. Phys., 116:5351 – 5362, 2002.

[100] MPI Forum: *MPI: A message-passing interface standard. Version 2.2.* `http://www.mpi-forum.org`, 2009.

[101] A. Neelov and C. Holm: *Interlaced P3M algorithm with analytical and ik-differentiation.* J. Chem. Phys., 132(23):234103, 2010.

[102] F. Nestler: *Automated parameter tuning based on RMS errors for nonequispaced FFTs.* Preprint 2015-01, Faculty of Mathematics, Technische Universität Chemnitz, 2015.

[103] F. Nestler: *Parameter tuning for the NFFT based fast Ewald summation.* Preprint 2015-05, Faculty of Mathematics, Technische Universität Chemnitz, 2015.

[104] M. Nikolić, A. Jović, J. Jakić, V. Slavnić, and A. Balaž: *An analysis of FFTW and FFTE performance.* In M. Dulea, A. Karaivanova, A. Oulas, I. Liabotis, D. Stojiljkovic, and O. Prnjat (eds.): *High-Performance Computing Infrastructure for South East Europe's Research Communities*, vol. 2 of *Modeling and Optimization in Science and Technologies*, pp. 163 – 170. Springer International Publishing, 2014.

[105] I. Pasichnyk and B. Dünweg: *Coulomb interactions via local dynamics: A molecular-dynamics algorithm.* J. Phys.: Condens. Mat., 16:3999 – 4020, 2004.

[106] D. Pekurovsky: *P3DFFT - parallel FFT subroutine library.* `http://code.google.com/p/p3dfft`.

[107] D. Pekurovsky: *P3DFFT: A framework for parallel computations of Fourier transforms in three dimensions.* SIAM J. Sci. Comput., 34:C192 – C209, 2012.

[108] J.W. Perram, H.G. Petersen, and S.W. De Leeuw: *An algorithm for the simulation of condensed matter which grows as the 3/2 power of the number of particles.* Molecular Phys., 65:875 – 893, 1988.

[109] S.J. Plimpton: *Parallel FFT subroutine library.* `http://www.sandia.gov/~sjplimp/docs/fft/README.html`.

[110] S.J. Plimpton, R. Pollock, and M. Stevens: *Particle-Mesh Ewald and rRESPA for parallel molecular dynamics simulations.* In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing (Minneapolis, 1997)*, Philadelphia, 1997. SIAM.

[111] M. Porto: *Ewald summation of electrostatic interactions of systems with finite extent in two of three dimensions.* J. Phys. A, 33:6211 – 6218, 2000.

[112] D. Potts and G. Steidl: *Fast summation at nonequispaced knots by NFFTs.* SIAM J. Sci. Comput., 24:2013 – 2037, 2003.

[113] D. Potts, G. Steidl, and A. Nieslony: *Fast convolution with radial kernels at nonequispaced knots.* Numer. Math., 98:329 – 351, 2004.

[114] D. Potts, G. Steidl, and M. Tasche: *Fast Fourier transforms for nonequispaced data: A tutorial.* In J.J. Benedetto and P.J.S.G. Ferreira (eds.): *Modern Sampling Theory: Mathematics and Applications*, pp. 247 – 270, Boston, MA, USA, 2001. Birkhäuser.

[115] J. Poulson, L. Demanet, N. Maxwell, and L. Ying: *A parallel butterfly algorithm.* SIAM J. Sci. Comput., 36(1):C49 – C65, 2014.

[116] C. Rader: *Discrete Fourier transforms when the number of data samples is prime.* Proceedings of the IEEE, 56(6):1107–1108, 1968.

[117] C. Sagui and T.A. Darden: *P3M and PME: A comparison of the two methods.* Simulation and theory of electrostatic interactions in solution, 104(1):104 – 113, 1999.

[118] Y. Shan, J.L. Klepeis, M.P. Eastwood, R.O. Dror, and D.E. Shaw: *Gaussian split Ewald: A fast Ewald mesh method for molecular simulation.* The Journal of chemical physics, 122(5):54101, 2005.

[119] S. Song and J.K. Hollingsworth: *Designing and auto-tuning parallel 3-d FFT for computation-communication overlap.* In *Proceedings of the 19th ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '14*, pp. 181 – 192, New York, New York, USA, 2014. ACM Press, ISBN 9781450326568.

[120] W.H. Steeb: *Kronecker Product of Matrices and Applications.* BI-Wissenschaftsverlag, Mannheim; Wien; Zürich, ISBN 3-411-14811-X.

[121] G. Steidl: *A note on fast Fourier transforms for nonequispaced grids.* Adv. Comput. Math., 9:337 – 353, 1998.

[122] J. Stoer and R. Bulirsch: *Introduction to Numerical Analysis.* Springer, Berlin, 2nd ed., 1996.

[123] D. Takahashi: *FFTE - a fast Fourier transform package.* `http://www.ffte.jp`, 2000.

[124] D. Takahashi: *An implementation of parallel 3-d FFT with 2-d decomposition on a massively parallel cluster of multi-core processors.* In R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski (eds.): *Parallel Processing and Applied Mathematics*, vol. 6067 of *Lecture Notes in Computer Science*, pp. 606 – 614. Springer, 2010.

[125] U. Trottenberg, C.W. Oosterlee, and A. Schuller: *Multigrid.* Academic Press, Inc., Orlando, FL, USA, 2000, ISBN 0-12-701070-X.

[126] M.E. Tuckerman, P. Minary, K. Pihakari, and G.J. Martyna: *A new reciprocal space based method for treating long range interactions in ab initio and force-field based calculations for surfaces, wires, and clusters.* In *Computational methods for macromolecules: challenges and applications (New York, 2000)*, vol. 24 of *Lect. Notes Comput. Sci. Eng.*, pp. 381 – 410. Springer, Berlin, 2002.

[127] C.F. Van Loan: *Computational Frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, PA, USA, 1992.

[128] T. Volkmer: *OpenMP parallelization in the NFFT software library.* Preprint 2012-07, Faculty of Mathematics, Technische Universität Chemnitz, 2012.

[129] Y.l. Wang, F. Hedman, M. Porcu, F. Mocci, and A. Laaksonen: *Non-uniform FFT and its applications in particle simulations.* Appl. Math., 05(03):520–541, 2014.

[130] A.F. Ware: *Fast approximate Fourier transforms for irregularly spaced data.* SIAM Rev., 40:838 – 856, 1998.

[131] A.H. Widmann and D.B. Adolf: *A comparison of Ewald summation techniques for planar surfaces.* Comput. Phys. Commun., 107:167 – 186, 1997.

[132] M. Winkel, R. Speck, H. Hübner, L. Arnold, R. Krause, and P. Gibbon: *A massively parallel, multi-disciplinary Barnes–Hut tree code for extreme-scale N-body simulations.* Comput. Phys. Commun., 183(4):880 – 889, 2012.

[133] I.C. Yeh and M.L. Berkowitz: *Ewald summation for systems with slab geometry.* J. Chem. Phys., 111(7):3155 – 3162, 1999.

[134] Y. Zhang, J. Liu, E. Kultursay, M. Kandemir, N. Pitsianis, and X. Sun: *Scalable parallelization strategies to accelerate NuFFT data translation on multi-cores.* In *Proceedings of the 16th International Euro-Par Conference on Parallel Processing: Part II*, Euro-Par'10, pp. 125 – 136, Berlin, Heidelberg, 2010. Springer-Verlag.

List of algorithms

Nomenclature

DFT ..... Discrete Fourier Transform, page 21
DFT$^{\mathsf{H}}$ ..... Adjoint Discrete Fourier Transform, page 21
FFT ..... Fast Fourier Transform, page 17
FFT ..... Fast Fourier Transform, page 21
FFT$^{\mathsf{H}}$ ..... Adjoint Fast Fourier Transform, page 21
NDFT ..... Nonequispaced Discrete Fourier Transform, page 56
NDFT$^{\mathsf{H}}$ ..... Adjoint Nonequispaced Discrete Fourier Transform, page 57
NFFT ..... Nonequispaced Fast Fourier Transform, page 57
NFFT$^{\mathsf{H}}$ ..... Adjoint Nonequispaced Fast Fourier Transform, page 57
ad-NFFT ..... gradient NFFT with analytic derivative, page 60
i$\boldsymbol{k}$-NFFT ..... gradient NFFT with derivative in Fourier space, page 60
PFFT ..... Parallel FFT, page 36
PFFT$^{\mathsf{H}}$ ..... Parallel FFT$^{\mathsf{H}}$, page 36
PNDFT$^{\mathsf{H}}$ ..... Parallel NDFT$^{\mathsf{H}}$, page 81
PNDFT ..... Parallel NDFT, page 81
PNFFT ..... Parallel NFFT, page 74
PNFFT$^{\mathsf{H}}$ ..... Parallel NFFT$^{\mathsf{H}}$, page 77
P^{2}NFFT ..... Particle-Particle–NFFT, page 109
ad-P^{2}NFFT ... P^{2}NFFT with derivative in Fourier space, page 110
i$\boldsymbol{k}$-P^{2}NFFT .... P^{2}NFFT with analytic derivative, page 110
P^{3}M ..... Particle-Particle–Particle-Mesh, page 109
ad-P^{3}M ..... P^{3}M with derivative in Fourier space, page 110
i$\boldsymbol{k}$-P^{3}M ..... P^{3}M with analytic derivative, page 110