Particle Simulation Based on Nonequispaced Fast Fourier Transforms

Michael Pippig and Daniel Potts

Chemnitz University of Technology
Department of Mathematics
09107 Chemnitz, Germany
E-mail: {*michael.pippig, daniel.potts*}*@mathematik.tu-chemnitz.de*

The fast calculation of long-range interactions is a demanding problem in particle simulation. The main focus of our approach is the decomposition of the problem in building blocks and present efficient numerical realizations for these blocks. For that reason we recapitulate the fast Fourier transform at nonequispaced nodes and the fast summation method. We describe the application of these algorithms to the evaluation of long-range potentials and compare our methods with the existing fast multipole method.

Keywords and Phrases. fast discrete summation, fast Fourier transform at nonequispaced nodes, NFFT, fast multipole method, FMM, Ewald method, FFT-accelerated Ewald sum, particle-particle particle-mesh (P^3M), particle-mesh Ewald (PME), smooth particle-mesh Ewald (SPME).

1 Introduction

An important concern of applied mathematics is the development of efficient algorithms for frequently recurring problems. On the other hand one should decompose the practical problems such that one can use the efficient algorithms and consequently incorporate advanced implementations.

The aim of this tutorial is to decompose problems from particle simulation into building blocks. These blocks are the fast Fourier transform (FFT), the fast Fourier transform at nonequispaced nodes (NFFT) and the fast summation method.

In Section 2 we summarize the main ideas of the NFFT. A severe shortcoming of traditional Fourier schemes in recent applications is the need for equispaced sampling. During the last two decades that problem has attracted much attention. The nonequispaced fast Fourier transform[19] overcomes these disadvantages while keeping the number of floating point operations at $N \log N$. The concept of NFFT is the trade of exactness for efficiency. Instead of precise computations (up to machine precision for actual implementations), the proposed methods guarantee a user specified target accuracy. An early review of several algorithms for nonuniform Fourier transforms[3,6] has been given by A. Ware[32]. A unified approach to fast algorithms for the NFFT has been obtained by G. Steidl in[30]. The main idea is the use of a window function which is well localized in space as well as in frequency domain. Then one is able to use an approximation by translates of the scaled window function and estimate the approximation error, see the tutorial paper[29]. It became clear, that this approach is related to the gridding algorithm, which was known in image processing context and astrophysics years ago. Similar methods where used in particle simulation. A widely used implementation is available as part of the NFFT package[19] and is based on the FFTW[13].

In Section 3 we summarize the main ideas of the fast summation method based on NFFT. This method can be interpreted as nonequispaced convolution. For equispaced nodes the discrete convolution and its fast computation is typically realized by FFT exploiting the basic property $\mathrm{e}^{2\pi \mathrm{i}(y-x)} = \mathrm{e}^{2\pi \mathrm{i} y}\mathrm{e}^{-2\pi \mathrm{i} x}$. Following these lines, we propose to compute the "convolution at nonequispaced nodes" by Fourier methods as well, more precisely by the NFFT. This new method includes convolutions, e.g., with kernels of the form $1/\|x\|_2$.

In Section 4 we describe how to use the building blocks from Section 2 and Section 3 for particle simulation. Here we focus on the Coulomb potential for open and periodic systems. We remark that some FFT-accelerated Ewald[17,15] methods contain similar steps as the fast summation based on NFFT.

Finally Section 5 contains various different numerical examples, where we compare our methods with the fast multipole method.

2 Nonequispaced Fourier transforms

This section summarizes the mathematical theory and ideas behind the NFFT based on[29,19–21]. Let the dimension $d \in \mathbb{N}$, the torus $\mathbb{T}^d := \mathbb{R}^d/\mathbb{Z}^d \sim [-\frac{1}{2}, \frac{1}{2})^d$ and the sampling set $\mathcal{X} := \{\mathbf{x}_j \in \mathbb{T}^d \colon j = 1, \ldots, M\}$ with $M \in \mathbb{N}$ be given. Furthermore, let the multi degree $\mathbf{N} = (N_0, N_1, \ldots, N_{d-1})^\top \in 2\mathbb{N}^d$ and the index set for possible frequencies $I_{\mathbf{N}} := \{-\frac{N_0}{2}, \ldots, \frac{N_0}{2} - 1\} \times \ldots \times \{-\frac{N_{d-1}}{2}, \ldots, \frac{N_{d-1}}{2} - 1\}$ be given. We define the space of d-variate trigonometric polynomials $T_{\mathbf{N}}$ of multi degree $\mathbf{N}$ by

$$T_{\mathbf{N}} := \operatorname{span}\left\{\mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot} \colon \mathbf{k} \in I_{\mathbf{N}}\right\}.$$

The dimension of this space and hence the total number of Fourier coefficients is $N_\pi = N_0 \cdot \ldots \cdot N_{d-1}$. Note, that we abbreviate the inner product between the frequency $\mathbf{k}$ and the time/spatial node $\mathbf{x}$ by $\mathbf{k}\mathbf{x} = \mathbf{k}^\top \mathbf{x} = k_0 x_0 + k_1 x_1 + \ldots + k_{d-1} x_{d-1}$. For clarity of presentation the multi index $\mathbf{k}$ addresses elements of vectors and matrices as well.

2.1 Nonequispaced discrete Fourier transform (NDFT)

For a finite number N_π of given Fourier coefficients $\hat{f}_{\mathbf{k}} \in \mathbb{C},\ \mathbf{k} \in I_{\mathbf{N}}$, one wants to evaluate the trigonometric polynomial

$$f(\mathbf{x}) := \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{f}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\mathbf{x}} \in T_{\mathbf{N}} \tag{2.1}$$

at given nonequispaced nodes $\mathbf{x}_j \in \mathbb{T}^d$, $j = 1, \ldots, M$. Thus, our concern is the computation of the matrix vector product

$$\mathbf{f} = \mathbf{A}\hat{\mathbf{f}} \tag{2.2}$$

where

$$\mathbf{f} := (f(\mathbf{x}_j))_{j=1,\ldots,M}, \quad \mathbf{A} := \left(\mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\mathbf{x}_j}\right)_{j=1,\ldots,M;\ \mathbf{k} \in I_{\mathbf{N}}}, \quad \hat{\mathbf{f}} := \left(\hat{f}_{\mathbf{k}}\right)_{\mathbf{k} \in I_{\mathbf{N}}}.$$

The straightforward algorithm for this matrix vector product, which is called NDFT, takes $\mathcal{O}(MN_\pi)$ arithmetical operations. A related matrix vector product is the adjoint NDFT

$$\hat{\mathbf{f}} = \mathbf{A}^{\mathsf{H}}\mathbf{f}, \qquad \hat{f}_{\mathbf{k}} = \sum_{j=1}^{M} f_j \mathrm{e}^{+2\pi \mathrm{i}\mathbf{k}\mathbf{x}_j},$$

where $\mathbf{A}^{\mathsf{H}} = \overline{\mathbf{A}}^{\top}$. Note furthermore, that the inversion formula $\mathbf{F}^{-1} = \mathbf{F}^{\mathsf{H}}$ for the (equispaced and normalized) Fourier matrix $\mathbf{F}$ does **not** hold in the general situation of arbitrary sampling nodes for the matrix $\mathbf{A}$.

2.2 Nonequispaced fast Fourier transform (NFFT)

The most successful approach for the fast computation of (2.2), cf.[6,3,30,29,11,14], is based on the usage of an oversampled FFT and a window function φ which is simultaneously localized in time/space and frequency. Basically, the scheme utilizes the convolution theorem in the following three informal steps:

1. deconvolve the trigonometric polynomial $f \in T_{\mathbf{N}}$ in (2.1) with a window function in frequency domain,
2. compute an oversampled FFT on the result of step 1.,
3. convolve the result of step 2. with the window function in time/spatial domain, i.e., evaluate this convolution at the nodes $\mathbf{x}_j$.

Throughout the rest of the paper we denote by $\sigma > 1$ the oversampling factor and by $n = \sigma N \in \mathbb{N}$ the FFT size. Furthermore, for $d > 1$ let $\boldsymbol{\sigma} = (\sigma_0, \ldots, \sigma_{d-1})^{\top} \in \mathbb{R}^d$, $\sigma_0, \ldots, \sigma_{d-1} > 1$, $\mathbf{n} = \boldsymbol{\sigma} \odot \mathbf{N}$, and $n_\pi = n_0 \cdot \ldots \cdot n_{d-1}$ denote the oversampling factor, the FFT size, and the total FFT size, respectively. For notational convenience we use the pointwise product $\boldsymbol{\sigma} \odot \mathbf{N} := (\sigma_0 N_0, \sigma_1 N_1, \ldots, \sigma_{d-1} N_{d-1},)^{\top}$ and the pointwise inverse $\mathbf{N}^{-1} := \left(\frac{1}{N_0}, \frac{1}{N_1}, \ldots, \frac{1}{N_{d-1}}\right)^{\top}$.

The window function

Starting with a window function $\varphi \in L_2(\mathbb{R})$, which is well localized in the time/spatial domain $\mathbb{R}$ and in the frequency domain $\mathbb{R}$, respectively, one assumes that its 1-periodic version $\tilde{\varphi}$, i.e.,

$$\tilde{\varphi}(x) := \sum_{r \in \mathbb{Z}} \varphi(x + r)$$

has an uniformly convergent Fourier series and is well localized in the time/spatial domain $\mathbb{T}$ and in the frequency domain $\mathbb{Z}$, respectively. Thus, the periodic window function $\tilde{\varphi}$ may be represented by its Fourier series

$$\tilde{\varphi}(x) = \sum_{k \in \mathbb{Z}} \hat{\varphi}(k) \mathrm{e}^{-2\pi \mathrm{i} k x}$$

with the Fourier coefficients

$$\hat{\varphi}(k) := \int_{\mathbb{T}} \tilde{\varphi}(x)\, \mathrm{e}^{+2\pi \mathrm{i} k x}\, \mathrm{d}x = \int_{\mathbb{R}} \varphi(x)\, \mathrm{e}^{+2\pi \mathrm{i} k x}\, \mathrm{d}x\,, \qquad k \in \mathbb{Z}.$$

We truncate this series at the FFT length n, which causes an aliasing error.

If φ is furthermore well localized in time/spatial domain $\mathbb{R}$, it can be truncated with truncation parameter $m \in \mathbb{N}$, $m \ll n$ and approximated by the function $\varphi \cdot \chi_{[-\frac{m}{n}, \frac{m}{n}]}$ which has compact support within the interval $[-\frac{m}{n}, \frac{m}{n}]$. Furthermore, the periodic window function can be approximated by the periodic version of the truncated window function. For $d > 1$, univariate window functions $\varphi_0, \ldots, \varphi_{d-1}$, and a node $\mathbf{x} = (x_0, \ldots, x_{d-1})^\top$ the multivariate window function is simply given by

$$\varphi(\mathbf{x}) := \varphi_0(x_0)\, \varphi_1(x_1) \ldots \varphi_{d-1}(x_{d-1})$$

and $\tilde{\varphi}(\mathbf{x}) = \sum_{\mathbf{r} \in \mathbb{Z}^d} \varphi(\mathbf{x} + \mathbf{r})$ again denotes the 1-periodic version; an immediate observation is

$$\hat{\varphi}(\mathbf{k}) := \int_{\mathbb{R}^d} \varphi(\mathbf{x})\, \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}}\, \mathrm{d}\mathbf{x} = \hat{\varphi}_0(k_0)\, \hat{\varphi}_1(k_1) \ldots \hat{\varphi}_{d-1}(k_{d-1})\,.$$

For a single truncation parameter $m \in \mathbb{N}$ the window function is truncated to the cube $\mathbf{n}^{-1} \odot [-m, m]^d$.

We follow the general approach of[30,29] and approximate the complex exponentials in the trigonometric polynomial (2.1) by

$$\mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{x}} \approx \frac{1}{n_\pi \hat{\varphi}(\mathbf{k})} \sum_{\mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x})} \tilde{\varphi}\left(\mathbf{x} - \mathbf{n}^{-1} \odot \mathbf{l}\right) \mathrm{e}^{-2\pi \mathrm{i} \left(\mathbf{n}^{-1} \odot \mathbf{l}\right) \mathbf{k}} \tag{2.3}$$

where the set

$$I_{\mathbf{n},m}(\mathbf{x}) := \{\mathbf{l} \in I_{\mathbf{n}} \colon \mathbf{n} \odot \mathbf{x} - m\mathbf{1} \leq \mathbf{l} \leq \mathbf{n} \odot \mathbf{x} + m\mathbf{1}\}$$

collects these indices where the window function is mostly concentrated (the inequalities have to be fulfilled modulo $\mathbf{n}$ and for each component). After changing the order of summation in (2.1) we obtain for $\mathbf{x}_j \in \mathbb{T}^d$, $j = 1, \ldots, M$, the approximation

$$f(\mathbf{x}_j) \approx \sum_{\mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x}_j)} \left(\sum_{\mathbf{k} \in I_{\mathbf{N}}} \frac{\hat{f}_{\mathbf{k}}}{n_\pi \hat{\varphi}(\mathbf{k})} \mathrm{e}^{-2\pi \mathrm{i} \left(\mathbf{n}^{-1} \odot \mathbf{l}\right) \mathbf{k}} \right) \tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right)\,.$$

This causes a truncation and an aliasing error, see[29,26] for details. As can be readily seen, after an initial deconvolution step, the expression in brackets can be computed via a d-variate FFT of total size n_π. The final step consists of the evaluation of sums having at most $(2m+1)^d$ terms where the window function is sampled only in the neighborhood of the node $\mathbf{x}_j$.

The algorithm and its matrix notation

The proposed scheme reads in matrix vector notation as

$$\mathbf{A}\hat{\mathbf{f}} \approx \mathbf{B}\mathbf{F}\mathbf{D}\hat{\mathbf{f}},$$

where $\mathbf{B}$ denotes the real $M \times n_\pi$ sparse matrix

$$\mathbf{B} := \left(\tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right) \cdot \chi_{I_{\mathbf{n},m}(\mathbf{x}_j)}(\mathbf{l})\right)_{j=1,\ldots,M;\, \mathbf{l} \in I_{\mathbf{n}}}, \tag{2.4}$$

where $\mathbf{F}$ is the d-variate Fourier matrix of size $n_\pi \times n_\pi$,

$$\mathbf{F} := \left(\mathrm{e}^{-2\pi \mathrm{i}\left(\mathbf{n}^{-1} \odot \mathbf{l}\right)\mathbf{k}}\right)_{\mathbf{l} \in I_{\mathbf{n}};\, \mathbf{k} \in I_{\mathbf{n}}}, \tag{2.5}$$

and where $\mathbf{D}$ is the real $n_\pi \times N_\pi$ 'diagonal' matrix

$$\mathbf{D} := \bigotimes_{t=0}^{d-1} \left(\mathbf{O}_t \mid \operatorname{diag}\left(\frac{1}{n_t \hat{\varphi}_t(k_t)}\right)_{k_t \in I_{N_t}} \mid \mathbf{O}_t\right)^\top \tag{2.6}$$

with zero matrices $\mathbf{O}_t$ of size $N_t \times \frac{n_t - N_t}{2}$. Obviously, the approximate matrix splitting can be applied to the adjoint matrix as $\mathbf{A}^H \approx \mathbf{D}^\top \mathbf{F}^H \mathbf{B}^\top$, where the multiplication with the sparse matrix $\mathbf{B}^\top$ is implemented in a 'transposed' way, summation as outer loop and only using the index sets $I_{\mathbf{n},m}(\mathbf{x}_j)$.

Algorithm 1 NFFT

Input: $d \in \mathbb{N}$ dimension,
$M \in \mathbb{N}$ number of nodes, nodes $\mathbf{x}_j \in [-\frac{1}{2}, \frac{1}{2})^d,\ j = 1, \ldots, M$,
$\mathbf{N} \in 2\mathbb{N}^d$ multi degree, Fourier coefficients $\hat{f}_{\mathbf{k}} \in \mathbb{C},\ \mathbf{k} \in I_{\mathbf{N}}$,
$\boldsymbol{\sigma} \in \mathbb{R}^d$ oversampling factor with $\sigma_t > 1$, $t = 0, \ldots, d-1$,
$m \in \mathbb{N}$ window size of $\tilde{\varphi}$.

1. For $\mathbf{k} \in I_{\mathbf{N}}$ compute

$$\hat{g}_{\mathbf{k}} := \frac{\hat{f}_{\mathbf{k}}}{n_\pi \hat{\varphi}(\mathbf{k})}.$$

2. For $\mathbf{l} \in I_{\mathbf{n}}$ with $\mathbf{n} = \boldsymbol{\sigma} \odot \mathbf{N}$ compute by d-variate (forward) FFT

$$g_{\mathbf{l}} := \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{g}_{\mathbf{k}}\, \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\left(\mathbf{n}^{-1} \odot \mathbf{l}\right)}.$$

3. For $j = 1, \ldots, M$ compute

$$s_j := \sum_{\mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x}_j)} g_{\mathbf{l}}\, \tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right).$$

Output: approximate values $s_j \approx f(\mathbf{x}_j)$, $j = 1, \ldots, M$.

Evaluation techniques for window functions

To keep the aliasing error and the truncation error small, several univariate functions φ with good localization in time and frequency domain were proposed. For an oversampling factor $\sigma > 1$, a degree $N \in 2\mathbb{N}$, the FFT length $n = \sigma N$, and a cut-off parameter $m \in \mathbb{N}$, the following window functions are considered:

Algorithm 2 NFFT$^{\mathsf{H}}$

Input: $d \in \mathbb{N}$ dimension,
$M \in \mathbb{N}$ number of nodes,
nodes $\mathbf{x}_j \in [-\frac{1}{2}, \frac{1}{2})^d$, coefficients $f_j \in \mathbb{C}$, $j = 1, \dots, M$,
$\mathbf{N} \in 2\mathbb{N}^d$ multi degree,
$\boldsymbol{\sigma} \in \mathbb{R}^d$ oversampling factor with $\sigma_t > 1$, $t = 0, \dots, d-1$,
$m \in \mathbb{N}$ window size of $\tilde{\varphi}$.

1. For $\mathbf{l} \in I_{\mathbf{n}}$ with $\mathbf{n} = \boldsymbol{\sigma} \odot \mathbf{N}$ compute

$$g_{\mathbf{l}} := \sum_{\substack{j=1 \\ \mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x}_j)}}^{M} f_j \, \tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right).$$

2. For $\mathbf{k} \in I_{\mathbf{n}}$ compute by d-variate (backward) FFT

$$\hat{g}_{\mathbf{k}} := \sum_{\mathbf{l} \in I_{\mathbf{n}}} g_{\mathbf{l}} \, \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k}\left(\mathbf{n}^{-1} \odot \mathbf{l}\right)}.$$

3. For $\mathbf{k} \in I_{\mathbf{N}}$ compute

$$\hat{s}_{\mathbf{k}} := \frac{\hat{g}_{\mathbf{k}}}{n_\pi \hat{\varphi}(\mathbf{k})}.$$

Output: approximate values $\hat{s}_{\mathbf{k}} \approx \hat{f}_{\mathbf{k}}$, $\mathbf{k} \in I_{\mathbf{N}}$.

1. for a shape parameter $b = \frac{2\sigma}{2\sigma-1}\frac{m}{\pi}$ the dilated *Gaussian window*[6,30,5]

$$\begin{aligned} \varphi(x) &= (\pi b)^{-1/2} \, \mathrm{e}^{-(nx)^2/b}, \\ \hat{\varphi}(k) &= \frac{1}{n} \, \mathrm{e}^{-(\frac{\pi k}{n})^2 b}, \end{aligned} \tag{2.7}$$

2. for M_{2m} denoting the centered cardinal B-Spline of order $2m$ the dilated *B-Spline window*[3,30]

$$\begin{aligned} \varphi(x) &= M_{2m}(nx), \\ \hat{\varphi}(k) &= \frac{1}{n} \begin{cases} 1 & \text{for } k = 0, \\ \operatorname{sinc}^{2m}\left(\frac{k\pi}{n}\right) & \text{otherwise}, \end{cases} \end{aligned} \tag{2.8}$$

3. the dilated *Sinc window*[26]

$$\begin{aligned} \varphi(x) &= \frac{N\,(2\sigma-1)}{2m} \operatorname{sinc}^{2m}\left(\frac{\pi N x\,(2\sigma-1)}{2m}\right), \\ \hat{\varphi}(k) &= M_{2m}\left(\frac{2mk}{(2\sigma-1)\,N}\right), \end{aligned} \tag{2.9}$$

with $\operatorname{sinc}(x) := \sin(x) \,/\, x$ for $x \neq 0$ and $\operatorname{sinc}(0) := 1$

4. and for a shape parameter $b = \pi(2 - \frac{1}{\sigma})$ the dilated *Kaiser-Bessel window*[27]

$$\varphi(x) = \frac{1}{\pi} \begin{cases} \dfrac{\sinh(b\sqrt{m^2 - n^2x^2})}{\sqrt{m^2 - n^2x^2}} & \text{for } |x| \leq \frac{m}{n}, \\ \dfrac{\sin(b\sqrt{n^2x^2 - m^2})}{\sqrt{n^2x^2 - m^2}} & \text{otherwise,} \end{cases} \tag{2.10}$$

$$\hat{\varphi}(k) = \begin{cases} \frac{1}{n} I_0\left(m\sqrt{b^2 - (2\pi k/n)^2}\right) & \text{for } k = -n\left(1 - \frac{1}{2\sigma}\right), \ldots, n\left(1 - \frac{1}{2\sigma}\right), \\ 0 & \text{otherwise,} \end{cases}$$

where I_0 denotes the modified zero-order Bessel function.

Note, that the latter two have compact support in frequency domain while the second one has compact support in time domain. Further references on the usage of (generalized) Kaiser-Bessel window functions include[18,23], where some authors prefer to interchange the role of time and frequency domain. For these univariate window functions φ, the approximation error of Algorithm 1 obeys

$$|f(x_j) - s_j| \leq C_{\sigma,m} \|\hat{\mathbf{f}}\|_1 ,$$

where

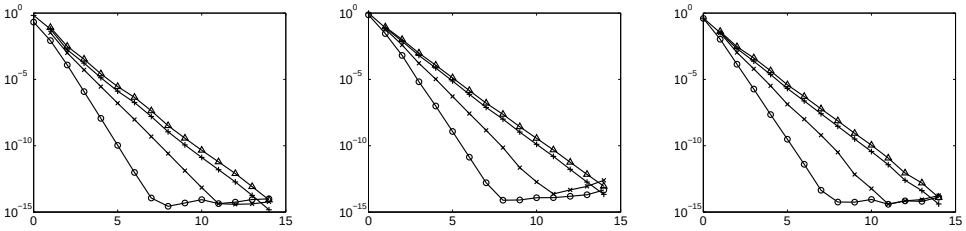
$$C_{\sigma,m} := \begin{cases} 4\,\mathrm{e}^{-m\pi(1-1/(2\sigma-1))} & \text{for (2.7), cf.[30],} \\ 4\left(\frac{1}{2\sigma-1}\right)^{2m} & \text{for (2.8), cf.[30],} \\ \frac{1}{m-1}\left(\frac{2}{\sigma^{2m}} + \left(\frac{\sigma}{2\sigma-1}\right)^{2m}\right) & \text{for (2.9), cf.[26],} \\ 4\pi\left(\sqrt{m}+m\right)\sqrt[4]{1-\frac{1}{\sigma}}\,\mathrm{e}^{-2\pi m\sqrt{1-1/\sigma}} & \text{for (2.10), cf.[26].} \end{cases}$$

Thus, for fixed $\sigma > 1$, the approximation error introduced by the NFFT decays exponentially with the number m of terms in sum (2.3). Using the tensor product approach, the error estimates above have been generalized for the multivariate setting in[7,5].

In summary, the whole algorithm for the fast approximate computation of (2.1) consists of the computation of N_π multiplications, the computation of a d-variate FFT of total size n_π and the sparse summations. Therefore it requires $\mathcal{O}(n_\pi + n_\pi \log n_\pi + (2m+1)^d M) = \mathcal{O}(n_\pi \log n_\pi + m^d M)$ arithmetic operations.

In[21] we suggest different methods for the compressed storage and application of matrix $\mathbf{B}$, which are all available within our NFFT library[19] by choosing particular flags in a simple way during the initialization phase. These methods include fully precomputed window function, tensor product based precomputation, linear interpolation from a lookup table, fast Gaussian gridding (see also[14]) and no precomputation of the window function. The choice of precomputation does not yield a different asymptotic performance but rather yields a lower constant in the amount of computation.

Finally we remark, that similar approximations of the exponentials as in (2.3) are used in various particle methods, see [4, Formula (24)], [8, Formula (3.5)] or [15, Formula (7.57)]. However, mainly splines are used as window functions. We note that, e.g., Kaiser-Bessel functions lead to better results, while using the same window size m, see Figure 2.1



(a) $d = 1$, $N = 2^{12}$. (b) $d = 2$, $N = 2^6$. (c) $d = 3$, $N = 2^4$.

Figure 2.1. Error E_∞ for increasing cut-off parameter $m = 0, \ldots, 14$ and $d = 1, 2, 3$. In each case, the degree N was chosen to be equal along each dimension such that $|I_{\mathbf{N}}| = 2^{12}$. We fixed the oversampling factor $\sigma = 2$ and $M = 10000$. Shown are results for the Kaiser-Bessel ($\circ$), the Sinc ($\times$), the B-spline ($+$), and the Gaussian window function ($\triangle$).

and[20], where the accuracy is measured by

$$E_\infty := \max_{1 \le j \le M} |f_j - s_j| / \sum_{\mathbf{k} \in I_{\mathbf{N}}} |\hat{f}_{\mathbf{k}}| \, .$$

For further NFFT approaches see [20, Appendix D].

3 Fast Summation Algorithms

We are interested in the fast evaluation of sums

$$h(\mathbf{y}) := \sum_{l=1}^{L} \alpha_l \mathcal{K}(\mathbf{y} - \mathbf{x}_l) = \sum_{l=1}^{L} \alpha_l K(\|\mathbf{y} - \mathbf{x}_l\|_2), \tag{3.1}$$

at M different target nodes $\mathbf{y}_j$, $j = 1, \ldots, M$, where $\|\mathbf{x}\|_2 := (x_0^2 + \ldots + x_{d-1}^2)^{1/2}$ denotes the Euclidean norm. This approach was suggested in[27,28,9]. The original idea for our algorithm came from the consideration of (3.1) for equispaced source nodes $\mathbf{x}_l$ and target nodes $\mathbf{y}_j$. In this case we have simply to compute the multiplication of a vector with a Toeplitz matrix or a block–Toeplitz–Toeplitz–block matrix in the multivariate setting. The standard algorithm to do this uses the FFT, see [27, Section 2]. Here we propose the summation algorithm based on NFFT for arbitrary distributed source and target nodes.

The kernel function K is in general a non-periodic function, while the use of Fourier methods requires to replace K by a periodic version. Without loss of generality we may assume that the source and target nodes are scaled, such that $\|\mathbf{x}_l\|_2, \|\mathbf{y}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2}$ and consequently $\|\mathbf{y}_j - \mathbf{x}_l\|_2 < \frac{1}{2} - \varepsilon_{\mathrm{B}}$. The parameter $\varepsilon_{\mathrm{B}} > 0$, which we specify later, guarantees that K has to be evaluated only at nodes in the ball with radius $\frac{1}{2} - \varepsilon_{\mathrm{B}}$. This simplifies the later consideration of a 1-periodic version of K.

We deal with kernels $\mathcal{K}$ which are C^∞ except for the origin, where $\mathcal{K}$ or its derivatives may have singularities. Examples of such kernels are

$$\|\mathbf{x}\|_2^2 \log \|\mathbf{x}\|_2, \; \log \|\mathbf{x}\|_2 \quad \text{and} \quad \frac{1}{\|\mathbf{x}\|_2^\beta} \quad (\beta \in \mathbb{N}).$$

For the sake of simplicity we define $\mathcal{K}(\mathbf{0}) = 0$ whenever a singularity appears at the origin. This enables us to evaluate (3.1) at source nodes which coincide with a target node. Of course, our algorithm can be modified for other kernels frequently used in the approximation by radial basis functions, e.g., the Gaussian[22] or the (inverse) multiquadric[10] $(x^2 + c^2)^{\pm 1/2}$. Our algorithm, in particular our regularization procedure, is simply structured, can easily be adapted to different kernels K and requires $\mathcal{O}(L \log \sqrt[d]{L} + M)$ or $\mathcal{O}(M \log \sqrt[d]{M} + L)$ arithmetic operations for uniformly distributed source nodes $\mathbf{x}_l$ or target nodes $\mathbf{y}_j$, respectively.

Beyond a special treatment of $\mathcal{K}$ near the boundary, we have to be concerned about the singularity of $\mathcal{K}$ at the origin. We regularize $\mathcal{K}$ near $\mathbf{0}$ and near the boundary $\{\mathbf{x} \in \mathbb{T}^d : \|\mathbf{x}\|_2 = \frac{1}{2}\}$ as follows

$$\mathcal{K}_{\mathrm{R}}(\mathbf{x}) := \begin{cases} T_{\mathrm{I}}(\|\mathbf{x}\|_2) & \text{if} \quad \|\mathbf{x}\|_2 \le \varepsilon_{\mathrm{I}}, \\ T_{\mathrm{B}}(\|\mathbf{x}\|_2) & \text{if} \quad \frac{1}{2} - \varepsilon_{\mathrm{B}} < \|\mathbf{x}\|_2 < \frac{1}{2}, \\ T_{\mathrm{B}}(\frac{1}{2}) & \text{if} \quad \frac{1}{2} \le \|\mathbf{x}\|_2, \\ K(\|\mathbf{x}\|_2) & \text{otherwise}, \end{cases} \tag{3.2}$$

where $0 < \varepsilon_{\mathrm{I}} < \frac{1}{2} - \varepsilon_{\mathrm{B}} < \frac{1}{2}$. The functions T_{I} and T_{B} will be chosen such that $\mathcal{K}_{\mathrm{R}}$ is in the Sobolev space $H^p(\mathbb{T}^d)$ for an appropriate parameter $p \in \mathbb{N}$. Several regularizations of $\mathcal{K}$ are possible, e.g., by algebraic polynomials, splines or trigonometric polynomials, see[10].

Here we focus our attention on two point Taylor interpolation, i.e., we are interested in the unique polynomial of degree $2p-1$ which satisfies the interpolation conditions

$$P^{(j)}(m-r) = a_j, \quad P^{(j)}(m+r) = b_j, \quad j = 0, \dots, p-1 \tag{3.3}$$

at the endpoints of an interval $[m-r, m+r], r > 0$.

Lemma 3.1. *(see [9, Proposition 2.2]) For given a_j, b_j $(j = 0, \dots, p-1)$ there exists a unique polynomial P of degree $2p-1$ which satisfies* (3.3)*. With $y := \frac{z-m}{r}$ this polynomial can be written as*

$$\begin{aligned} P(z) = \frac{1}{2^p} \sum_{j=0}^{p-1} \sum_{k=0}^{p-1-j} \binom{p-1+k}{k} \frac{r^j}{j! 2^k} \\ \Big((1+y)^{j+k}(1-y)^p a_j + (1-y)^{j+k}(1+y)^p (-1)^j b_j \Big). \end{aligned} \tag{3.4}$$

Lemma 3.2. *The derivative of the polynomial P from* (3.4) *is given by*

$$\begin{aligned} P'(z) = & \frac{1}{2^p} \sum_{j=0}^{p-1} \sum_{\substack{k=0 \\ j+k \neq 0}}^{p-1-j} \binom{p-1+k}{k} \frac{r^{j-1}}{j! 2^k} \\ & \Big((1+y)^{j+k-1}(1-y)^{p-1} \big[(j+k)(1-y) - p(1+y) \big] a_j \\ & + (1-y)^{j+k-1}(1+y)^{p-1} \big[(j+k)(1+y) - p(1-y) \big] (-1)^{j-1} b_j \Big). \end{aligned} \tag{3.5}$$

We exploit (3.4) with $a_j = K^{(j)}(\varepsilon_{\mathrm{I}}), b_j = (-1)^j a_j, j = 0, \dots, p-1, m = 0, r = 2\varepsilon_{\mathrm{I}}$ to obtain T_{I} and with $a_j = K^{(j)}(\frac{1}{2} - \varepsilon_{\mathrm{B}}), b_j = \delta_{0,j} K(\frac{1}{2}), j = 0, \dots, p-1, m = \frac{1-\varepsilon_{\mathrm{B}}}{2}, r = \varepsilon_{\mathrm{B}}$ to obtain T_{B}.

Next we approximate the smooth function $\mathcal{K}_{\mathrm{R}}$ by the Fourier series

$$\mathcal{K}_{\mathrm{R}} \approx \mathcal{K}_{\mathrm{RF}}(\mathbf{x}) := \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\mathbf{x}}, \tag{3.6}$$

where

$$\hat{b}_{\mathbf{k}} := \frac{1}{N_\pi} \sum_{\mathbf{l}\in I_{\mathbf{N}}} \mathcal{K}_{\mathrm{R}}(\mathbf{N}^{-1} \odot \mathbf{l}) \mathrm{e}^{+2\pi \mathrm{i}(\mathbf{N}^{-1}\odot \mathbf{l})\mathbf{k}}, \quad \mathbf{k} \in I_{\mathbf{N}} \,. \tag{3.7}$$

Then our original kernel splits as

$$\mathcal{K} = (\mathcal{K} - \mathcal{K}_{\mathrm{R}}) + (\mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}) + \mathcal{K}_{\mathrm{RF}} = \mathcal{K}_{\mathrm{NE}} + \mathcal{K}_{\mathrm{ER}} + \mathcal{K}_{\mathrm{RF}}, \tag{3.8}$$

where $\mathcal{K}_{\mathrm{NE}} := \mathcal{K} - \mathcal{K}_{\mathrm{R}}$ and $\mathcal{K}_{\mathrm{ER}} := \mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}$. Since $\mathcal{K}_{\mathrm{R}}$ is smooth, the approximation error $\mathcal{K}_{\mathrm{ER}}$ of its Fourier approximation $\mathcal{K}_{\mathrm{RF}}$ should be small. We neglect this error and approximate h in (3.1) by

$$h(\mathbf{y}) \approx \tilde{h}(\mathbf{y}) := h_{\mathrm{NE}}(\mathbf{y}) + h_{\mathrm{RF}}(\mathbf{y}),$$

where

$$h_{\mathrm{NE}}(\mathbf{y}) := \sum_{l=1}^{L} \alpha_l \mathcal{K}_{\mathrm{NE}}(\mathbf{y} - \mathbf{x}_l)\,, \tag{3.9}$$

$$h_{\mathrm{RF}}(\mathbf{y}) := \sum_{l=1}^{L} \alpha_l \mathcal{K}_{\mathrm{RF}}(\mathbf{y} - \mathbf{x}_l)\,. \tag{3.10}$$

Instead of h we evaluate $\tilde{h}$ at the target nodes $\mathbf{y}_j$. If either the source nodes $\mathbf{x}_k$ or the target nodes $\mathbf{y}_j$ are "sufficiently uniformly distributed" this can indeed be done in a fast way, namely:

Near field computation (3.9)

By definition (3.2), the function $\mathcal{K}_{\mathrm{NE}}$ has a small support contained in the ball of radius ε_{I} around $\mathbf{0}$ and in the neighborhood of the boundary. The boundary is not interesting for us since $\|\mathbf{y}_j - \mathbf{x}_l\|_2 \leq 1/2 - \varepsilon_{\mathrm{B}}$. To achieve the desired complexity of our algorithm we suppose that either the L source nodes $\mathbf{x}_l$ or the M target nodes $\mathbf{y}_j$ are "sufficiently uniformly distributed", i.e., we suppose that there exists a small constant $\nu \in \mathbb{N}$ such that every ball of radius ε_{I} contains at most ν of the nodes $\mathbf{x}_l$ or of the nodes $\mathbf{y}_j$, respectively. This implies that ε_{I} depends linearly on $1/\sqrt[d]{L}$, respectively $1/\sqrt[d]{M}$. In the following, we restrict our attention to the case

$$\varepsilon_{\mathrm{I}} \sim \sqrt[d]{\frac{\nu}{L}}.$$

Then for fixed $\mathbf{y}_j$ the sum (3.9) contains not more than ν terms so that its evaluation at M target nodes $\mathbf{y}_j$ requires only $\mathcal{O}(\nu M)$ arithmetic operations.

We remark that also $\mathcal{O}(\log \sqrt[d]{M})$, respectively $\mathcal{O}(\log \sqrt[d]{L})$ nodes instead of $\mathcal{O}(1)$ nodes per ball will keep a complexity of $\mathcal{O}(M \log \sqrt[d]{M} + L \log \sqrt[d]{L})$ of our whole algorithm.

Far field summation (3.10) **by NFFT$^{\intercal}$/NFFT**

Substituting (3.6) for $\mathcal{K}_{\mathrm{RF}}$ we obtain

$$h_{\mathrm{RF}}(\mathbf{y}_j) \;=\; \sum_{l=1}^{L} \alpha_l \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}(\mathbf{y}_j-\mathbf{x}_l)} \;=\; \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \left(\sum_{l=1}^{L} \alpha_l \mathrm{e}^{+2\pi \mathrm{i}\mathbf{k}\mathbf{x}_l} \right) \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\mathbf{y}_j}.$$

The expression in the inner brackets can be computed by a d-variate NFFT$^{\intercal}$ of total size N_π. This is followed by N_π multiplications with $\hat{b}_{\mathbf{k}}$ and completed by a d-variate NFFT of total size N_π to compute the outer sum with the complex exponentials. If m is the cut-off parameter and $\sigma = (2)_{t=0,\ldots,d-1}$ the oversampling factor of the NFFT$^{\intercal}$/NFFT, then the proposed evaluation of h_{RF} at the nodes $\mathbf{y}_j$, $j = 1,\ldots,M$ requires $\mathcal{O}(m^d(L+M) + \sigma_\pi N_\pi \log(\sigma_\pi N_\pi))$ arithmetic operations. The relation between M, L and $\mathbf{N}$ is determined by the approximation error of the algorithm and is discussed in detail in[27,28,10].

Note, we can avoid the error of $\mathcal{K}_{\mathrm{ER}}$ in the near field in (3.8) by splitting kernel function $\mathcal{K}$ as

$$\mathcal{K} = (\mathcal{K} - \mathcal{K}_{\mathrm{RF}}) \;+\; \mathcal{K}_{\mathrm{RF}} = \mathcal{K}_{\mathrm{NE}} \;+\; \mathcal{K}_{\mathrm{ER}} \;+\; \mathcal{K}_{\mathrm{RF}}, \tag{3.11}$$

and setting $\mathcal{K}_{\mathrm{NE}} := (\mathcal{K} - \mathcal{K}_{\mathrm{RF}})\, \chi_{\{\|\cdot\|\le\varepsilon_I\}}$, $\mathcal{K}_{\mathrm{ER}} := \mathcal{K} - \mathcal{K}_{\mathrm{RF}} - \mathcal{K}_{\mathrm{NE}}$. The first splitting (3.8) is preferable, if we are able to evaluate the near field regularization T_I in a fast way. If T_I can not be computed in a fast way but the Fourier coefficients $\hat{b}_{\mathbf{k}}$ are given, the splitting (3.11) should be used.

The algorithm and its matrix notation

The proposed scheme reads in matrix vector notation as

$$\mathbf{K}\alpha \approx \mathbf{B}_{\mathbf{y}}\mathbf{F}\mathbf{D}\mathbf{D}_{\mathbf{b}}\mathbf{D}^{\top}\mathbf{F}^{\mathrm{H}}\mathbf{B}_{\mathbf{x}}^{\top}\alpha + \mathbf{K}_{\mathrm{NE}}\alpha\,, \tag{3.12}$$

where $\mathbf{B}_{\mathbf{x}}$ denotes the real $L \times n_\pi$ sparse matrix depending on the source nodes $\mathbf{x}_l$, $l = 1,\ldots,L$ as given in (2.4)

$$\mathbf{B}_{\mathbf{x}} := \left(\tilde{\varphi}\left(\mathbf{x}_l - \mathbf{n}^{-1} \odot \mathbf{l}\right) \cdot \chi_{I_{\mathbf{n},m}(\mathbf{x}_l)}(\mathbf{l})\right)_{l=1,\ldots,L;\, \mathbf{l}\in I_{\mathbf{n}}},$$

$\mathbf{B}_{\mathbf{y}}$ denotes the real $M \times n_\pi$ sparse matrix depending on the target nodes $\mathbf{y}_j$, $j = 1,\ldots,M$ as given in (2.4)

$$\mathbf{B}_{\mathbf{y}} := \left(\tilde{\varphi}\left(\mathbf{y}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right) \cdot \chi_{I_{\mathbf{n},m}(\mathbf{y}_j)}(\mathbf{l})\right)_{j=1,\ldots,M;\, \mathbf{l}\in I_{\mathbf{n}}},$$

$\mathbf{F}$ is the d-variate Fourier matrix of size $n_\pi \times n_\pi$ given in (2.5), $\mathbf{D}$ is the real $n_\pi \times N_\pi$ 'diagonal' matrix given in (2.6), which contains the Fourier coefficients of the window function, $\mathbf{D}_{\mathbf{b}} = \mathrm{diag}(\hat{b}_{\mathbf{k}})_{\mathbf{k}\in I_{\mathbf{N}}}$ contains the Fourier coefficients of the regularized kernel $\mathcal{K}_{\mathrm{R}}$ given in (3.7). Furthermore, $\mathbf{K}_{\mathrm{NE}}$ contains the near field correction (3.9). From the representation (3.12) we see, that $\mathbf{F}\mathbf{D}\mathbf{F}^{\mathrm{H}}$ with a diagonal matrix $\mathbf{D}$ is well known as fast realization of a convolution on a mesh. The matrix $\mathbf{B}^{\top}$ is used to smear the charge density onto a grid, then one can use a convolution on a mesh and finally a back-interpolation via $\mathbf{B}$.

In summary we obtain Algorithm 3 for the fast evaluation of (3.1).

Algorithm 3 Fast sum

Input: $d \in \mathbb{N}$ dimension,
$p \in \mathbb{N}$ smoothness of regularized kernel K_{R},
$\mathbf{N} \in 2\mathbb{N}^d$ multi degree,
$\varepsilon_{\mathrm{I}} > 0$ nearfield size,
$\varepsilon_{\mathrm{B}} > 0$ boundary size,
$L \in \mathbb{N}$ number of source nodes,
$M \in \mathbb{N}$ number of target nodes,
source nodes $\mathbf{x}_l \in [-\frac{1}{4} + \frac{\varepsilon_{\mathrm{B}}}{2}, \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2})^d$, $l = 1, \ldots, L$,
target nodes $\mathbf{y}_j \in [-\frac{1}{4} + \frac{\varepsilon_{\mathrm{B}}}{2}, \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2})^d$, $j = 1, \ldots, M$,
coefficients $\alpha_l \in \mathbb{C},\ l = 1, \ldots, L$.

Precomputation:

i) Computation of $(\hat{b}_{\mathbf{k}})_{\mathbf{k} \in I_{\mathbf{N}}}$ by (3.7) and (3.2).

ii) Computation of $K_{\mathrm{NE}}(\mathbf{y}_j - \mathbf{x}_l)$ for all $j = 1, \ldots, M$ and $l \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)$, where $I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j) := \{l \in \{1, \ldots, L\} \colon \|\mathbf{y}_j - \mathbf{x}_l\|_2 < \varepsilon_{\mathrm{I}}\}$.

1. For $\mathbf{k} \in I_{\mathbf{N}}$ compute by Algorithm 2 the d-variate NFFT$^{\mathsf{H}}$
$$\hat{a}_{\mathbf{k}} := \sum_{l=1}^{L} \alpha_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l}.$$
2. For $\mathbf{k} \in I_{\mathbf{N}}$ compute the products
$$\hat{d}_{\mathbf{k}} := \hat{a}_{\mathbf{k}} \hat{b}_{\mathbf{k}}.$$
3. For $j = 1, \ldots, M$ compute by Algorithm 1 the d-variate NFFT
$$h_{\mathrm{RF}}(\mathbf{y}_j) := \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{d}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{y}_j}.$$
4. For $j = 1, \ldots, M$ compute the near field sums
$$h_{\mathrm{NE}}(\mathbf{y}_j) = \sum_{l \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)} \alpha_l \mathcal{K}_{\mathrm{NE}}(\mathbf{y}_j - \mathbf{x}_l).$$
5. For $j = 1, \ldots, M$ compute the near field corrections
$$\tilde{h}(\mathbf{y}_j) = h_{\mathrm{NE}}(\mathbf{y}_j) + h_{\mathrm{RF}}(\mathbf{y}_j).$$

Output: approximate values $\tilde{h}(\mathbf{y}_j) \approx h(\mathbf{y}_j),\ j = 1, \ldots, M$.

Generalization to cuboidal domains

Whenever the restrictions $\|\mathbf{x}_l\|_2, \|\mathbf{y}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2}$ are not fulfilled we must scale the nodes. We now explain the resulting changes to Algorithm 3. In order to handle node distributions with unequal dimensional extend we introduce $\mathbf{s} = (s_0, s_1, \ldots, s_{d-1})^{\top} \in \mathbb{N}^d$

as a component wise scaling such that $\|\mathbf{s}^{-1} \odot \mathbf{x}_l\|_2, \|\mathbf{s}^{-1} \odot \mathbf{y}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_\mathrm{B}}{2}$ and therefore $\|\mathbf{s}^{-1} \odot (\mathbf{y}_j - \mathbf{x}_l)\| < \frac{1}{2} - \varepsilon_\mathrm{B}$ for all $l = 1, \ldots, L$ and $j = 1, \ldots, M$. By the substitution $\mathbf{z} := \mathbf{s}^{-1} \odot \mathbf{x}$ we obtain $\mathbf{x} = \mathbf{s} \odot \mathbf{z}$ and therefore $\mathcal{K}^\mathbf{s}(\mathbf{z}) := \mathcal{K}(\mathbf{s} \odot \mathbf{z}) = \mathcal{K}(\mathbf{x})$.

Since $\mathcal{K}^\mathbf{s}$ is not radial symmetric anymore, we define the regularized kernel function $\mathcal{K}^\mathbf{s}_\mathrm{R}$ in a slightly different way

$$\mathcal{K}^\mathbf{s}_\mathrm{R}(\mathbf{z}) := \begin{cases} T_\mathrm{I}(\|\mathbf{s} \odot \mathbf{z}\|_2) & \text{if} \quad \|\mathbf{s} \odot \mathbf{z}\|_2 \leq \varepsilon_\mathrm{I}, \\ T^\mathbf{z}_\mathrm{B}(\|\mathbf{z}\|_2) & \text{if} \quad \frac{1}{2} - \varepsilon_\mathrm{B} < \|\mathbf{z}\| < \frac{1}{2}, \\ T^\mathbf{z}_\mathrm{B}(\frac{1}{2}) & \text{if} \quad \frac{1}{2} \leq \|\mathbf{z}\|, \\ K(\|\mathbf{s} \odot \mathbf{z}\|_2) & \text{otherwise.} \end{cases}$$

While the definition of T_I remains the same as in the non-scaled case, we again exploit (3.4) to obtain $T^\mathbf{z}_\mathrm{B}$ with the altered parameters $a_j = K^{(j)}((\frac{1}{2} - \varepsilon_\mathrm{B})\frac{\|\mathbf{s}\odot\mathbf{z}\|_2}{\|\mathbf{z}\|_2}), b_j = \delta_{0,j} K(\frac{1}{2} s_{\max}), j = 0, \ldots, p-1, m = \frac{1}{2} - \frac{\varepsilon_\mathrm{B}}{2}, r = \varepsilon_\mathrm{B}$, where $s_{\max} := \max\{s_t \colon t = 0, \ldots, d-1\}$. Note that the interpolation polynomial $T^\mathbf{z}_\mathrm{B}$ may change for every node, since the coefficients a_j now depend on $\mathbf{z}$. We only need to evaluate $T^\mathbf{z}_\mathrm{B}$ during a precomputation step to obtain the Fourier coefficients $\hat{b}_\mathbf{k}$ of the regularized kernel function $\mathcal{K}_\mathrm{RF}$, therefore the dependency of the interpolation polynomial on $\mathbf{z}$ has no impact on the complexity of our fastsum algorithm. In order to assure differentiability of $\mathcal{K}_\mathrm{RF}$ at the border $\{\mathbf{x} \in \mathbb{T}^d : \|\mathbf{x}\|_2 \geq \frac{1}{2}\}$, we set $T^\mathbf{z}_\mathrm{B}(\frac{1}{2}) = b_0$ constant for all $\mathbf{z}$.

Let $s_\pi := s_0 \cdot \ldots \cdot s_{d-1}$. We approximate the smooth function $\mathcal{K}^\mathbf{s}_\mathrm{R}$ by the Fourier series

$$\mathcal{K}^\mathbf{s}_\mathrm{RF}(\mathbf{x}) := \sum_{\mathbf{k} \in I_{\mathbf{s}\odot\mathbf{N}}} \hat{b}_\mathbf{k} \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k}\mathbf{x}},$$

where the Fourier coefficients $\hat{b}_\mathbf{k}$ are given by the d-variate discrete Fourier transform

$$\hat{b}_\mathbf{k} = \frac{1}{s_\pi N_\pi} \sum_{\mathbf{l} \in I_{\mathbf{s}\odot\mathbf{N}}} \mathcal{K}^\mathbf{s}_\mathrm{R}(\mathbf{s}^{-1} \odot \mathbf{N}^{-1} \odot \mathbf{l}) \mathrm{e}^{+2\pi \mathrm{i} (\mathbf{s}^{-1} \odot \mathbf{N}^{-1} \odot \mathbf{l}) \mathbf{k}}, \quad \mathbf{k} \in I_{\mathbf{s}\odot\mathbf{N}}.$$

The remaining part of the fast summation algorithm can be done analogously to the non-scaled case with the scaled NFFT size $\mathbf{s} \odot \mathbf{N}$ instead of $\mathbf{N}$.

4 Application to particle simulation for the Coulomb potential

In this part we apply our fast summation algorithm to the long-range potential $1/r$. The fundamental idea is to split the long-range part of the potential into a smooth long-range part and a singular short range part. We will use our splitting (3.9) and (3.10). There exists a variety of methods which use similar splittings such as the P^3M method[17]. In the following we will discuss open and periodic systems.

4.1 Open systems

In this part we apply our fast summation algorithm to a system of M charged particles located at source nodes $\mathbf{x}_l \in \mathbb{R}^3$ with charge $q_l \in \mathbb{R}$. We are interested in the evaluation of the electrostatic potential ϕ at target node $\mathbf{y} \in \mathbb{R}^3$,

$$\phi(\mathbf{y}) := \sum_{l=1}^{M} q_l \mathcal{K}(\mathbf{y} - \mathbf{x}_l), \tag{4.1}$$

and the force $\mathbf{F}$ acting at particle $\mathbf{y} \in \mathbb{R}^3$ with charge $q(\mathbf{y}) \in \mathbb{R}$,

$$\mathbf{F}(\mathbf{y}) := -q(\mathbf{y})\nabla\phi(\mathbf{y}) = -q(\mathbf{y}) \sum_{l=1}^{M} q_l \mathcal{K}'(\mathbf{y} - \mathbf{x}_l), \tag{4.2}$$

where the kernel functions $\mathcal{K}$ and $\mathcal{K}'$ corresponding to the Coulomb potential are given by

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \|\mathbf{x}\|_2 = 0, \\ 1/\|\mathbf{x}\|_2 & \text{otherwise,} \end{cases} \quad \text{and} \quad \mathcal{K}'(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \|\mathbf{x}\|_2 = 0, \\ \mathbf{x}/\|\mathbf{x}\|_2^3 & \text{otherwise.} \end{cases} \tag{4.3}$$

For equal sets of source and target nodes the sum (4.1) turns into

$$\phi(\mathbf{x}_j) = \sum_{\substack{l=1 \\ l \neq j}}^{M} \frac{q_l}{\|\mathbf{x}_j - \mathbf{x}_l\|_2}, \quad j = 1, \ldots, M,$$

and (4.2) becomes

$$\mathbf{F}(\mathbf{x}_j) = -q_j \sum_{\substack{l=1 \\ l \neq j}}^{M} q_l \frac{\mathbf{x}_j - \mathbf{x}_l}{\|\mathbf{x}_j - \mathbf{x}_l\|_2^3}, \quad j = 1, \ldots, M.$$

Furthermore we are interested in the computation of the total electrostatic potential energy

$$U := \frac{1}{2} \sum_{j=1}^{M} q_j \phi(\mathbf{x}_j),$$

which can be evaluated straightforward after the computation of the potentials $\phi(\mathbf{x}_j), j = 1, \ldots, M$. To get the potentials $\phi(\mathbf{x}_j)$ we apply Algorithm 3 on (4.1) by choosing equal sets of source and target nodes and $\alpha_l = q_l$, $l = 1, \ldots, M$. For the near field corrections of this algorithm we require repeated evaluations of the near field interpolation polynomial T_{I}. Instead of using (3.4) it is more efficient to calculate the coefficients once explicitly for smoothness $p = 2, \ldots, 12$ and evaluate the polynomial with a Horner scheme. E.g., for smoothness $p = 5$ we get the following explicit representation

$$T_{\mathrm{I}}(\|\mathbf{x}\|_2) = \frac{315}{128\varepsilon_{\mathrm{I}}} + \left(\frac{-105}{32\varepsilon_{\mathrm{I}}^3} + \left(\frac{189}{64\varepsilon_{\mathrm{I}}^5} + \left(\frac{-45}{32\varepsilon_{\mathrm{I}}^7} + \frac{35}{128\varepsilon_{\mathrm{I}}^9} \|\mathbf{x}\|_2^2 \right) \|\mathbf{x}\|_2^2 \right) \|\mathbf{x}\|_2^2 \right) \|\mathbf{x}\|_2^2.$$

For the fast calculation of the forces $\mathbf{F}(\mathbf{y})$ we follow the main steps of Algorithm 3. First we observe that Algorithm 3 decomposes potential $\phi(\mathbf{y})$ into

$$\phi(\mathbf{y}) \approx \phi_{\mathrm{NE}}(\mathbf{y}) + \phi_{\mathrm{RF}}(\mathbf{y}), \tag{4.4}$$

where the near field part ϕ_{NE} is given by

$$\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1 \\ \|\mathbf{y}-\mathbf{x}_l\|_2 < \varepsilon_{\mathrm{I}}}}^{M} q_l \left(\mathcal{K}(\mathbf{y} - \mathbf{x}_l) - T_{\mathrm{I}}(\|\mathbf{y} - \mathbf{x}_l\|_2) \right)$$

and the far field part ϕ_{RF} reads as

$$\phi_{\mathrm{RF}}(\mathbf{y}) = \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l} \right) \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{y}}.$$

Combination of (4.2) and (4.4) yields

$$\mathbf{F}(\mathbf{y}) = -q(\mathbf{y})\nabla\phi(\mathbf{y}) \approx -q(\mathbf{y})(\nabla\phi_{\mathrm{NE}}(\mathbf{y}) + \nabla\phi_{\mathrm{RF}}(\mathbf{y})).$$

Taking into account that

$$\nabla\phi_{\mathrm{RF}}(\mathbf{y}) = -2\pi\mathrm{i}\sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}}\mathbf{k}\left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi\mathrm{i}\mathbf{k}\mathbf{x}_l}\right)\mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}\mathbf{y}},$$

we are able to compute the inner sum of the far field part $\nabla\phi_{\mathrm{RF}}$ with only one NFFT$^{\mathsf{H}}$ and the outer vector sum with three NFFTs, one for each component. This is a remarkable improvement to the algorithm proposed in[25] where four NFFT$^{\mathsf{H}}$s and four NFFTs are needed to calculate the far field part of the forces. The gradient of the near field part reads as

$$\nabla\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1\\ \|\mathbf{y}-\mathbf{x}_l\|_2<\varepsilon_{\mathrm{I}}}}^{M} q_l\left(\mathcal{K}'(\mathbf{y}-\mathbf{x}_l) - \nabla T_{\mathrm{I}}(\|\mathbf{y}-\mathbf{x}_l\|_2)\right).$$

This vector sum can be evaluated straightforward. One way to obtain the gradient of T_{I} is to use (3.5) and the chain rule with $z = \|\mathbf{x}\|_2$. We obtain for $\|\mathbf{x}\| \neq 0$

$$\begin{aligned}\nabla T_{\mathrm{I}}(\|\mathbf{x}\|_2) =& \frac{1}{2^p}\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\sum_{j=0}^{p-1}\sum_{\substack{k=0\\ j+k\neq 0}}^{p-1-j}\binom{p-1+k}{k}\frac{r^{j-1}}{j!2^k}\\ &\Big((1+y)^{j+k-1}(1-y)^{p-1}\big[(j+k)(1-y)-p(1+y)\big]a_j\\ &+(1-y)^{j+k-1}(1+y)^{p-1}\big[(j+k)(1+y)-p(1-y)\big](-1)^{j-1}b_j\Big),\end{aligned}$$

where $y = \frac{\|\mathbf{x}\|_2 - m}{r}$ and $\nabla T_{\mathrm{I}}(0) = \mathbf{0}$. Alternatively we represent the gradient of T_{I}, e.g., for smoothness $p = 5$, by

$$\nabla T_{\mathrm{I}}(\|\mathbf{x}\|_2) = \left(\frac{-105}{16\varepsilon_{\mathrm{I}}^3} + \left(\frac{189}{16\varepsilon_{\mathrm{I}}^5} + \left(\frac{-1080}{128\varepsilon_{\mathrm{I}}^7} + \frac{35}{16\varepsilon_{\mathrm{I}}^9}\|\mathbf{x}\|_2^2\right)\|\mathbf{x}\|_2^2\right)\|\mathbf{x}\|_2^2\right)\|\mathbf{x}\|_2^2\right)\mathbf{x}.$$

In summary we can apply Algorithm 3 with the matrix representation given in (3.12).

4.2 Periodic systems

In this section we present a straightforward method, that accelerates the traditional Ewald summation technique by NFFT. This combination was first presented in[16] and is very similar to the FFT-accelerated Ewald sum, namely, the so-called particle-particle particle-mesh (P^{3}M), particle-mesh Ewald (PME) and smooth particle-mesh Ewald (SPME), see[4]. Additionally we will see, that the accelerated Ewald summation can be reinterpreted into a method very similar to our fastsum Algorithm 3.

We consider a system of charged particles coupled via the Coulomb potential, a cubic simulation box with edge length B, containing M charged particles, each with a charge $q_l \in \mathbb{R}$, located at $\mathbf{x}_l \in B\mathbb{T}^3$. Periodic boundary conditions in a system without cut-off is

represented by replicating the simulation box in all directions of space. The electrostatic potential ϕ at $\mathbf{y} \in B\mathbb{T}^3$, can be written as a lattice sum, see [12, Chapter 12] and[31],

$$\phi(\mathbf{y}) := \sum_{l=1}^{M} q_l \tilde{\mathcal{K}}(\mathbf{y} - \mathbf{x}_l), \tag{4.5}$$

and the force $\mathbf{F}$ at particle $\mathbf{y} \in B\mathbb{T}^3$ with charge $q(\mathbf{y}) \in \mathbb{R}$ is given by

$$\mathbf{F}(\mathbf{y}) := -q(\mathbf{y})\nabla\phi(\mathbf{y}) = -q(\mathbf{y}) \sum_{l=1}^{M} q_l \tilde{\mathcal{K}}'(\mathbf{y} - \mathbf{x}_l), \tag{4.6}$$

where the kernel functions

$$\tilde{\mathcal{K}}(\mathbf{x}) = \sum_{\mathbf{r}\in\mathbb{Z}^3} \mathcal{K}(\mathbf{x} + \mathbf{r}B) \quad \text{and} \quad \tilde{\mathcal{K}}'(\mathbf{x}) = \sum_{\mathbf{r}\in\mathbb{Z}^3} \mathcal{K}'(\mathbf{x} + \mathbf{r}B)$$

are periodizations of the kernel functions (4.3) corresponding to the Coulomb potential. For equal sets of source and target nodes the evaluation of the potential (4.5) and the force (4.6) reads as

$$\phi(\mathbf{x}_j) = \sum_{\mathbf{r}\in\mathbb{Z}^3} \sum_{\substack{l=1 \\ l\neq j \text{ for } \mathbf{r}=\mathbf{0}}}^{M} \frac{q_l}{\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2}, \quad j = 1, \ldots, M \tag{4.7}$$

and

$$\mathbf{F}(\mathbf{x}_j) = -q_j \sum_{\mathbf{r}\in\mathbb{Z}^3} \sum_{\substack{l=1 \\ l\neq j \text{ for } \mathbf{r}=\mathbf{0}}}^{M} q_l \frac{\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B}{\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2^3}, \quad j = 1, \ldots, M,$$

respectively. Furthermore we are interested in the computation of the total electrostatic potential energy

$$U := \frac{1}{2} \sum_{j=1}^{M} q_j \phi(\mathbf{x}_j),$$

which can be evaluated straightforward after the computation of the potentials $\phi(\mathbf{x}_j)$, $j = 1, \ldots, M$ like in the non-periodic case. The well-known Ewald formula for the computation of (4.7) splits the electrostatic potential ϕ into three parts

$$\phi = \phi^{\text{real}} + \phi^{\text{reci}} + \phi^{\text{self}}, \tag{4.8}$$

where the contribution from real space ϕ^{real}, reciprocal space ϕ^{reci} and the self-energy ϕ^{self} are given by

$$\begin{aligned}
\phi^{\text{real}}(\mathbf{x}_j) &= \sum_{\mathbf{r}\in\mathbb{Z}^3} \sum_{\substack{l=1 \\ l\neq j \text{ for } \mathbf{r}=\mathbf{0}}}^{M} q_l \frac{\operatorname{erfc}(\alpha\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2)}{\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2}, \\
\phi^{\text{reci}}(\mathbf{x}_j) &= \frac{1}{\pi B} \sum_{\mathbf{k}\in\mathbb{Z}^3\setminus\{\mathbf{0}\}} \frac{\mathrm{e}^{-\pi^2\|\mathbf{k}\|_2^2/(\alpha B)^2}}{\|\mathbf{k}\|_2^2} \sum_{l=1}^{M} q_l \mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}(\mathbf{x}_j - \mathbf{x}_l)/B}, \\
\phi^{\text{self}}(\mathbf{x}_j) &= -2q_j \frac{\alpha}{\sqrt{\pi}}.
\end{aligned} \tag{4.9}$$

Thereby, the complementary error function is defined by $\operatorname{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty \mathrm{e}^{-t^2} \mathrm{d}t$. Choosing optimal parameters, Ewald summation scales as $\mathcal{O}(M^{3/2})$. In order to overcome this increase in time we apply the NFFT for the calculation of the reciprocal-space potential ϕ^{reci} and we obtain a method similar as our fast summation method. To this end, we compute the Fourier transformed charge density

$$S(\mathbf{k}) = \sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l / B}$$

by NFFT$^{\mathrm{H}}$ and after truncation of the sum (4.9) we obtain by NFFT

$$\phi^{\mathrm{reci}}(\mathbf{x}_j) \approx \frac{1}{\pi B} \sum_{\mathbf{k} \in I_{\mathbf{N}} \setminus \{\mathbf{0}\}} \frac{\mathrm{e}^{-\pi^2 \|\mathbf{k}\|_2^2 / (\alpha B)^2}}{\|\mathbf{k}\|_2^2} S(\mathbf{k}) \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{x}_j / B}.$$

We now reinterpret the approximation steps of Ewald summation in order to use the steps of our Algorithm 3. Following (3.2) we define the slightly different regularization $\mathcal{K}_{\mathrm{R}}$ of the kernel function $\mathcal{K}$ by

$$\mathcal{K}_{\mathrm{R}}(\mathbf{x}) := T_{\mathrm{I}}(\|\mathbf{x}\|_2) \approx \begin{cases} T_{\mathrm{I}}(\|\mathbf{x}\|_2) \text{ if } \quad \|\mathbf{x}\|_2 \leq \varepsilon_{\mathrm{I}}, \\ K(\|\mathbf{x}\|_2) \text{ otherwise,} \end{cases}$$

where the near field regularization T_{I} is given by

$$T_{\mathrm{I}}(z) = \begin{cases} 2\alpha/\sqrt{\pi} & \text{if} \quad z = 0, \\ \operatorname{erf}(\alpha z)/z \text{ otherwise.} \end{cases}$$

Here the error function is defined by $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z \mathrm{e}^{-t^2} \mathrm{d}t$ and the parameter α must be chosen large enough to ensure $T_{\mathrm{I}}(\mathbf{x}) \approx \mathcal{K}(\mathbf{x})$ for all $\|\mathbf{x}\|_2 \geq \varepsilon_{\mathrm{I}}$. Since the periodic regularization $\tilde{\mathcal{K}}_{\mathrm{R}}(\mathbf{x}) := \sum_{\mathbf{r} \in \mathbb{Z}^3} \mathcal{K}_{\mathrm{R}}(\mathbf{x} + \mathbf{r})$ is smooth at the domain border, we do not need a regularization T_{B}. Instead of (4.8) we use (3.8) to split the electrostatic potential ϕ from (4.5) into the two parts

$$\phi(\mathbf{y}) \approx \phi_{\mathrm{NE}}(\mathbf{y}) + \phi_{\mathrm{RF}}(\mathbf{y}),$$

where the near field part ϕ_{NE} is given by

$$\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1 \\ \|\mathbf{y} - \mathbf{x}_l\|_2 < \varepsilon_{\mathrm{I}}}}^{M} q_l \left(\tilde{\mathcal{K}}(\mathbf{y} - \mathbf{x}_l) - T_{\mathrm{I}}(\|\mathbf{y} - \mathbf{x}_l\|_2) \right)$$

and the far field part ϕ_{RF} reads as

$$\phi_{\mathrm{RF}}(\mathbf{y}) = \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l / B} \right) \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{y} / B}.$$

Since the Fourier coefficients $\hat{b}_{\mathbf{k}}$ of $\tilde{\mathcal{K}}_{\mathrm{R}}$ can be calculated analytically,

$$\hat{b}_{\mathbf{k}} = \begin{cases} 0 & \text{if} \quad \|\mathbf{k}\|_2 = 0, \\ \frac{1}{\pi B} \frac{\mathrm{e}^{-\pi^2 \|\mathbf{k}\|_2^2 / (\alpha B)^2}}{\|\mathbf{k}\|_2^2} & \text{otherwise,} \end{cases}$$

we do not need to compute an inverse discrete Fourier transformation of $\mathcal{K}_{\mathrm{R}}$ in the pre-computation step as in (3.7). Analogously to Algorithm 3 the far field part ϕ_{RF} can be evaluated by one NFFT$^{\intercal}$ and one NFFT, while the near field part only includes nearest neighbors of every target node. Therefore we get an algorithm with the same complexity as our fastsum algorithm for non-periodic systems. For equal sets of source and target nodes the resulting algorithm coincides with the NFFT-accelerated Ewald summation.

As in the non-periodic case we get

$$\mathbf{F}(\mathbf{y}) = -q(\mathbf{y})\nabla\phi(\mathbf{y}) \approx -q(\mathbf{y})(\nabla\phi_{\mathrm{NE}}(\mathbf{y}) + \nabla\phi_{\mathrm{RF}}(\mathbf{y})),$$

where the far field part $\nabla\phi_{\mathrm{RF}}(\mathbf{y})$ can be computed by one NFFT$^{\intercal}$ and three NFFTs. The near field part $\nabla\phi_{\mathrm{NE}}(\mathbf{y})$ reads as

$$\nabla\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1 \\ \|\mathbf{y}-\mathbf{x}_l\|_2<\varepsilon_{\mathrm{I}}}}^{M} q_l \left(\mathcal{K}'(\mathbf{y}-\mathbf{x}_l) - \nabla T_{\mathrm{I}}(\|\mathbf{y}-\mathbf{x}_l\|_2)\right).$$

This vector sum can be evaluated straightforward. Since

$$\frac{\mathrm{d}}{\mathrm{d}z}\frac{\mathrm{erf}(\alpha z)}{z} = \frac{1}{z}\left(\frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2 z^2} - \frac{\mathrm{erf}(\alpha z)}{z}\right)$$

the gradient of T_{I} is given by

$$\nabla T_{\mathrm{I}}(\|\mathbf{x}\|_2) = \begin{cases} 0 & \text{if} \quad \|\mathbf{x}\|_2 = 0, \\ \frac{\mathbf{x}}{\|\mathbf{x}\|_2^2}\left(\frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2\|\mathbf{x}\|_2^2} - \frac{\mathrm{erf}(\alpha\|\mathbf{x}\|_2)}{\|\mathbf{x}\|_2}\right) & \text{otherwise.} \end{cases}$$

For equal sets of source and target nodes we get

$$\mathbf{F}(\mathbf{x}_j) \approx \mathbf{F}_{\mathrm{NE}}(\mathbf{x}_j) + \mathbf{F}_{\mathrm{RF}}(\mathbf{x}_j),$$

where

$$\begin{aligned}\mathbf{F}_{\mathrm{NE}}(\mathbf{x}_j) =& -q_j\nabla\phi_{\mathrm{NE}}(\mathbf{x}_j) \\ =& -q_j\sum_{\substack{l=1 \\ l\neq j}}^{M} q_l \frac{\mathbf{x}_j-\mathbf{x}_l}{\|\mathbf{x}_j-\mathbf{x}_l\|_2^2}\left(\frac{\mathrm{erfc}(\alpha\|\mathbf{x}_j-\mathbf{x}_l\|_2)}{\|\mathbf{x}_j-\mathbf{x}_l\|_2} + \frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2\|\mathbf{x}_j-\mathbf{x}_l\|_2^2}\right),\end{aligned}$$

$$\begin{aligned}\mathbf{F}_{\mathrm{RF}}(\mathbf{x}_j) =& -q_j\nabla\phi_{\mathrm{RF}}(\mathbf{x}_j) \\ =& \frac{2\mathrm{i}q_j}{B^2}\sum_{\mathbf{k}\in I_{\mathbf{N}}\setminus\{\mathbf{0}\}} \frac{\mathrm{e}^{-\pi^2\|\mathbf{k}\|_2^2/(\alpha B)^2}}{\|\mathbf{k}\|_2^2}\mathbf{k}S(\mathbf{k})\mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}\mathbf{x}_j/B}.\end{aligned}$$

This splitting coincides with the well known results of Ewald summation.

In summary we can apply Algorithm 3 with the matrix representation given in (3.12).

5 Numerical Results

5.1 Generation of pseudo random sampling sets

To guarantee a minimum distance between all nodes we used Hammersley and Halton pseudo random node distributions[33]. Let $p \in \mathbb{N}$ prime. Every non-negative integer $j \in$

$\mathbb{N} \cup \{0\}$ can be uniquely represented as

$$j = a_0 + a_1 p + a_2 p^2 + \ldots + a_r p^r, \quad a_i \in \{0, \ldots, p-1\}, \; i = 0, \ldots, r, \quad r \in \mathbb{N}.$$

We define

$$\Phi_p(j) := \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \ldots + \frac{a_r}{p^{r+1}}.$$

For d given primes $p_1 < p_2 < \ldots < p_d$ the d-variate Hammersley distribution is given by the following M nodes

$$\left(\frac{j}{M}, \Phi_{p_1}(j), \Phi_{p_2}(j), \ldots, \Phi_{p_{d-1}}(j) \right)^\top, \quad j = 0, \ldots, M-1$$

and the d-variate Halton distribution consists of the M nodes

$$\left(\Phi_{p_1}(j), \Phi_{p_2}(j), \ldots, \Phi_{p_d}(j) \right)^\top, \quad j = 0, \ldots, M-1.$$

Note that it is possible to increase a given set of Halton distributed nodes, while the number of Hammersley distributed nodes must be fixed because of the dependency of the first component of every node on M. Both distributions were implemented in[2]. The software package called *HAMMERSLEY* contains algorithms to generate Hammersley and Halton distributions on the square $[0,1]^2$ and the cube $[0,1]^3$. Furthermore it includes routines for mapping the square $[0,1]^2$ to the sphere $\{\mathbf{x} \in \mathbb{R}^3 \colon \|\mathbf{x}\|_2 = 1\}$ and mapping the cube $[0,1]^3$ to the ball $\{\mathbf{x} \in \mathbb{R}^3 \colon \|\mathbf{x}\|_2 \leq 1\}$.

5.2 Error definitions

We performed the computations of the total energy U, the potentials $\phi(\mathbf{x}_j)$ and the forces $\mathbf{F}(\mathbf{x}_j)$ for different node distributions with the direct algorithm, the fast multipole method from FCS software library[1] and the NFFT based algorithms. In this section we use the names of the applied methods to label numerical results, e.g., $U^{\mathrm{direct}}, U^{\mathrm{FMM}}$ and U^{NFFT} holds the results of the energy computations based on the three algorithms, respectively. We used the following error measurements to compare the results produced by the three methods.

The relative errors of the total potential energy with respect to the applied method read as

$$E_U^{\mathrm{FMM}} := \frac{U^{\mathrm{FMM}}}{U^{\mathrm{direct}}}, \quad E_U^{\mathrm{NFFT}} := \frac{U^{\mathrm{NFFT}}}{U^{\mathrm{direct}}}.$$

Let $\phi = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_M))^\top$. The relative errors of the potential with respect to the applied method are given by

$$E_\phi^{\mathrm{FMM}} := \frac{\|\phi^{\mathrm{direct}} - \phi^{\mathrm{FMM}}\|_2}{\|\phi^{\mathrm{direct}}\|_2}, \quad E_\phi^{\mathrm{NFFT}} := \frac{\|\phi^{\mathrm{direct}} - \phi^{\mathrm{NFFT}}\|_2}{\|\phi^{\mathrm{direct}}\|_2}.$$

For $\mathbf{F}(\mathbf{x}) = (F_0(\mathbf{x}), F_1(\mathbf{x}), F_2(\mathbf{x}))^\top$, and $\mathbf{F}_t := (F_t(\mathbf{x}_0), \ldots, F_t(\mathbf{x}_M))^\top$, $t = 0, 1, 2$, we define the average relative errors of the forces with respect to the applied method via

$$E_{\mathbf{F}}^{\mathrm{FMM}} := \frac{1}{3} \sum_{t=0}^{2} \frac{\|\mathbf{F}_t^{\mathrm{direct}} - \mathbf{F}_t^{\mathrm{FMM}}\|_1}{\|\mathbf{F}_t^{\mathrm{direct}}\|_1}, \quad E_{\mathbf{F}}^{\mathrm{NFFT}} := \frac{1}{3} \sum_{t=0}^{2} \frac{\|\mathbf{F}_t^{\mathrm{direct}} - \mathbf{F}_t^{\mathrm{NFFT}}\|_1}{\|\mathbf{F}_t^{\mathrm{direct}}\|_1}.$$

5.3 Test cases

We computed all test cases on the nowadays retired JUMP cluster at Research Center Jülich. Each of its 41 nodes got 32 Power4+ processors at 1.7 GHz and 128 GB memory. Our test runs were performed on one processor of one completely allocated node. The FCS software library[1] (timestamp 16.08.2007) and the NFFT library were compiled with IBM's *xlf* and *xlc* compilers at optimization level *-O5* and with the flag *-q64* to support the 64 bit architecture of JUMP.

We performed the NFFT based fastsum algorithm with equal oversampling factors $\sigma_t = 2, t = 0, 1, 2$, the truncation parameter $m = 2$, choose the Kaiser-Bessel window function, the regularization parameter $p = 5$ and set the initialization flags *PRE_PHI_HUT*, *PRE_LIN_PSI*, *FFT_OUT_OF_PLACE*, *FFTW_MEASURE* and *FFTW_DESTROY_INPUT* to obtain the following results.

The upper bound of the relative error of the total energy U^{FMM} was set to 0.001 for the FMM based calculations. For the NFFT based computations we tried to adjust the parameters to obtain the same upper bound on U^{NFFT}. All measured times include the computation of the energy, the potentials and the forces.

Hammersley distribution within a cube

For this test case M nodes satisfy a Hammersley distribution with parameters $p_1 = 2, p_2 = 3$ on $[0, 1]^3$. The randomly chosen charges $q_l \in \{-1, 1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1, 0, 1\}.$$

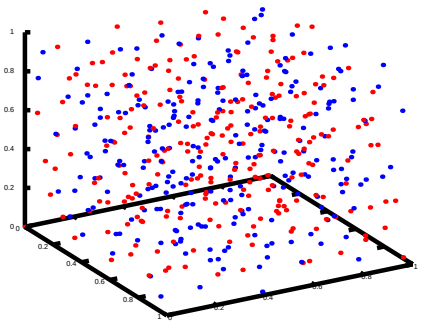


Figure 5.1. Hammersley distribution within a cube for 500 nodes.

Hammersley distribution within a ball

For this test case a Hammersley distribution with parameters $p_1 = 2, p_2 = 3$ on $[0, 1]^3$ is mapped to the ball

$$(x - 0.5)^2 + (y - 0.5)^2 + (z - 0.5)^2 \leq (0.5)^2.$$

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3/32	0.01	0.01	0.11
5 000	$(32,32,32)^\top$	3/32	0.86	0.25	0.38
50 000	$(64,64,64)^\top$	3/64	87.09*	3.00	4.63

Table 5.1. Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution in a cube. Times with $*$ are estimated.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	1.544e-15	3.094e-04	1.603e-15	8.086e-04	1.689e-15	1.917e-03
5 000	1.668e-07	9.205e-05	2.571e-06	5.626e-04	7.270e-06	9.513e-04
50 000	2.388e-08	3.006e-04	3.222e-07	5.454e-04	8.060e-07	1.240e-03

Table 5.2. Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution in a cube.

The randomly chosen charges $q_l \in \{-1, 1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1, 0, 1\}\,.$$

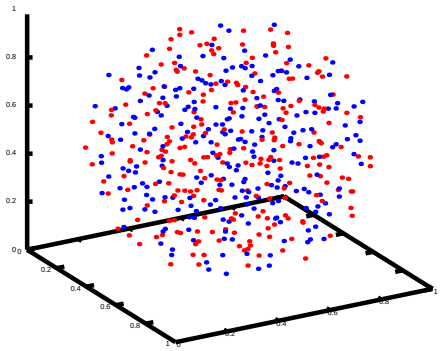


Figure 5.2. Hammersley distribution within a ball for 500 nodes.

Cylindrical Halton distribution

For this test case a Halton distribution with parameters $p_1 = 2, p_2 = 3, p_3 = 5$ on the cylinder

$$0 \le x_0 \le 10, \quad (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \le 1$$

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3.5/32	0.01	0.01	0.11
5 000	$(32,32,32)^\top$	3.5/32	0.86	0.24	0.34
50 000	$(64,64,64)^\top$	3.5/64	87.06	3.47	4.20
500 000	$(128,128,128)^\top$	3.5/128	8903.23	46.77	59.48

Table 5.3. Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution in a ball.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	9.677e-16	5.584e-05	1.751e-15	4.189e-04	2.328e-15	1.092e-03
5 000	1.339e-06	4.903e-05	5.471e-06	3.190e-04	1.454e-05	6.404e-04
50 000	1.113e-07	4.087e-04	2.579e-07	2.507e-04	7.650e-07	7.416e-04
500 000	1.075e-09	3.726e-04	7.366e-08	2.962e-04	2.195e-07	7.613e-04

Table 5.4. Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution in a ball.

is scaled into the cube $[0,1]^3$. The randomly chosen charges $q_l \in \{-1,1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1,0,1\}\,.$$

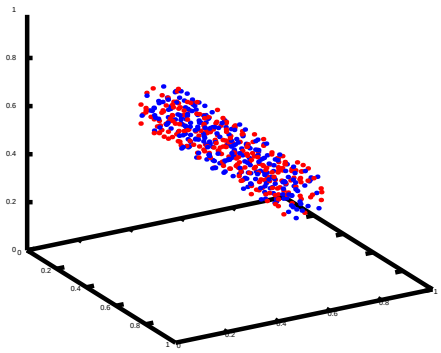


Figure 5.3. Cylindrical Halton distribution for 500 nodes.

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3/32	0.01	0.04	0.12
5 000	$(32,32,32)^\top$	3/32	0.86	0.25	0.60
50 000	$(256,64,64)^\top$	3.5/64	87.19	3.80	9.94

Table 5.5. Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Halton distribution in a cylinder.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	1.401e-06	1.201e-04	1.295e-05	4.374e-04	1.535e-05	3.973e-04
5 000	2.938e-07	2.161e-04	2.030e-06	3.914e-04	4.497e-06	2.491e-04
50 000	6.025e-08	1.065e-04	2.663e-08	3.011e-04	7.379e-08	1.302e-03

Table 5.6. Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Halton distribution in a cylinder.

Spherical Hammersley distribution

For this test case a two-dimensional Hammersley distribution on the square $[0,1]^2$ with parameter $p_1 = 2$ is mapped to the sphere

$$(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2 = (0.5)^2$$

The randomly chosen charges $q_l \in \{-1,1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1,0,1\}\,.$$

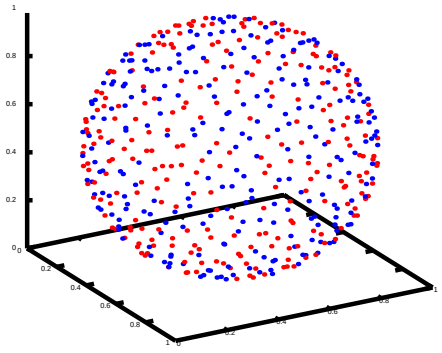


Figure 5.4. Spherical Hammersley distribution for 500 nodes.

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3/32	0.01	0.01	0.11
5 000	$(32,32,32)^\top$	3/32	0.86	0.17	0.23
50 000	$(64,64,64)^\top$	3/64	87.13	2.58	3.41

Table 5.7. Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution on a sphere.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	0.000e+00	1.338e-04	1.740e-15	6.655e-04	2.983e-15	1.697e-03
5 000	9.167e-08	1.957e-04	1.202e-06	4.101e-04	1.478e-06	3.644e-04
50 000	2.460e-08	4.866e-04	5.522e-08	2.422e-04	5.200e-08	1.545e-04

Table 5.8. Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution on a sphere.

Two Hammersley distributed balls

For this test case $\frac{63}{64}M$ Hammersley distributed nodes on the cube $[0,1]^3$ with parameters $p_1 = 2, p_2 = 3$ are mapped to the ball $(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2 \leq (0.5)^2$. A second set of $\frac{1}{64}M$ Hammersley distributed nodes on the cube $[0,1]^3$ with parameters $p_1 = 2, p_2 = 3$ is mapped to a smaller ball such that the density of nodes is equal to the first. We set the distance between the balls to 10. Finally the whole set of nodes is scaled into the cube $[0,1]^3$. The random charges $q_l \in \{-1,1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1,0,1\}\,.$$

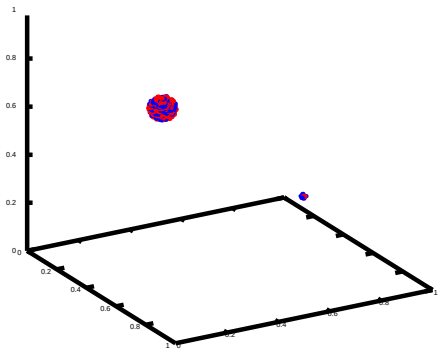


Figure 5.5. Hammersley distribution `hammersley_two_balls` with 500 nodes.

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	2/32	0.01	0.01	0.12
5 000	$(32,32,32)^\top$	2/32	0.86	0.26	1.15
50 000	$(384,48,48)^\top$	3.2/48	87.21	22.44	17.01

Table 5.9. Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution in two balls.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	6.111e-15	2.410e-04	9.988e-15	4.439e-04	1.794e-14	2.286e-04
5 000	1.188e-07	5.444e-04	8.945e-07	6.288e-04	8.189e-08	2.322e-04
50 000	5.916e-08	7.673e-04	1.115e-07	6.624e-04	7.228e-09	1.033e-03

Table 5.10. Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution in two balls.

NaCl grid structure

Let $M \in \mathbb{N} \setminus \{1\}$ be a cubic number of nodes. For $u, v, w \in \{0, \dots, \sqrt[3]{M} - 1\}$ we set

$$l = \left(\sqrt[3]{M}u + v\right)\sqrt[3]{M} + w + 1$$

and the nodes $\mathbf{x}_l$ and charges q_l are given by

$$\mathbf{x}_l = \frac{1}{\sqrt[3]{M} - 1}(u, v, w)^\top, \quad q_l = (-1)^{u+v+w+1}.$$

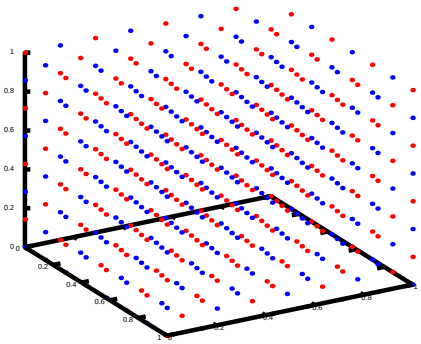


Figure 5.6. NaCl grid structure for 512 nodes.

$\mathbf{M}$	$\mathbf{N}$	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
512	$(32,32,32)^\top$	4/32	0.01	0.01	0.12
5 832	$(32,32,32)^\top$	2.5/32	1.17	0.13	0.33
50 653	$(64,64,64)^\top$	2.5/64	106.73*	2.41	3.21

Table 5.11. Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes are located like a NaCl grid. Times with $*$ are estimated.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
512	1.648e-15	2.316e-05	1.454e-15	2.892e-05	8.933e-15	2.799e-03
5 832	3.132e-07	4.491e-05	1.406e-05	5.430e-05	5.102e-04	9.325e-04
50 653	3.094e-07	5.422e-05	1.054e-05	7.071e-05	9.372e-04	1.451e-03

Table 5.12. Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes are located like a NaCl grid.

5.4 Conclusions

We computed the total energy, the potentials and the forces for several charged particle distributions with open boundary conditions. The runtimes of our NFFT based fast summation showed that it is able to compete with the highly optimized, kernel dependent FMM. We want to emphasize that the NFFT based algorithm is not restricted to the Coulomb potential since it can be easily adapted to various kernel functions. Testcases with equally distributed systems are the cubic Hammersley distribution, the Hammersley distribution within a ball and the NaCl grid structure. Especially for the last one we see in Table 5.12 that the errors of both algorithms are comparable. As examples for unequally distributed systems we chose the Halton distribution within a cylinder, the Hammersley distribution on a sphere and two Hammersley distributed balls. Although we did not optimize our algorithm for such inhomogeneous systems, the runtimes show an remarkable improvement in comparison to the direct algorithm. The generalization to cuboidal domains of the NFFT based fast summation was successfully tested for the cylindrical Halton distribution and the two Hammersley distributed balls, as one can see in Table 5.5 and Table 5.9.

6 Summary

We suggested fast summation algorithms of the form

$$\mathbf{K}\alpha \approx \mathbf{B_y F D D_b D}^\top \mathbf{F}^{\mathrm{H}} \mathbf{B}_\mathbf{x}^\top \alpha + \mathbf{K}_{\mathrm{NE}}\alpha$$

and applied this method to particle simulation for the Coulomb potential. This decomposition into separate building blocks simplifies the implementation. Note that particle-mesh methods are based on similar steps, see [15, Chapter 7]

- smear the charge density onto a grid, i.e., multiplication with $\mathbf{B}_\mathbf{x}^\top$, see step 1 of Algorithm 2
- Fourier transform the density, see step 2 of Algorithm 2 with FFT

- convolution in Fourier space, or solution of a differential equation in Fourier space
- Fourier transform back, see step 2 of Algorithm 1 with FFT
- Back-interpolation, or approximation, i.e., multiplication with $\mathbf{B_y}$, see step 3 of Algorithm 1
- Near field computation, i.e., multiplication with $\mathbf{K}_{\mathrm{NE}}$

A parallelization can be done step by step for the modules FFT, NFFT and finally the fast summation algorithm. A promising highly scalable FFT implementation has been tested up to 262144 cores of a BlueGene/P at Research Center Jülich[24].

Acknowledgments

The work was partly supported by the BMBF grant 01IH08001B. We are grateful to Toni Volkmer who produced the numerical results. Therefore we also thank the Jülich Supercomputing Center for providing the computational resources on JUMP. Furthermore, we thank Dr. Holger Dachsel and Dr. Ivo Kabadshow for their advise on the FCS software library. We remark that a new version of FCS is available for download.

References

1. *FCS software library*. `http://www.fz-juelich.de/jsc/fcs`.
2. *HAMMERSLEY. The Hammersley Quasirandom Sequence.* `http://people.scs.fsu.edu/~burkardt/cpp_src/hammersley/hammersley.html`.
3. G. Beylkin: *On the fast Fourier transform of functions with singularities*. Appl. Comput. Harmon. Anal., 2:363 – 381, 1995.
4. 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.
5. A.J.W. Duijndam and M.A. Schonewille: *Nonuniform fast Fourier transform*. Geophysics, 64:539 – 551, 1999.
6. A. Dutt and V. Rokhlin: *Fast Fourier transforms for nonequispaced data*. SIAM J. Sci. Stat. Comput., 14:1368 – 1393, 1993.
7. 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.
8. 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.
9. M. Fenn: *Fast Fourier transform at nonequispaced nodes and applications*. Dissertation, Universität Mannheim, 2006.
10. M. Fenn and G. Steidl: *Fast NFFT based summation of radial functions*. Sampl. Theory Signal Image Process., 3:1 – 28, 2004.
11. J.A. Fessler and B.P. Sutton: *Nonuniform fast Fourier transforms using min-max interpolation*. IEEE Trans. Signal Process., 51:560 – 574, 2003.
12. D. Frenkel and B. Smit: *Understanding molecular simulation: From algorithms to applications*. Academic Press, 2002.

13. M. Frigo and S.G. Johnson: *FFTW, C subroutine library*. `http://www.fftw.org`.
14. L. Greengard and J.Y. Lee: *Accelerating the nonuniform fast Fourier transform*. SIAM Rev., 46:443 – 454, 2004.
15. M. Griebel, S. Knapek, and G. Zumbusch: *Numerical simulation in molecular dynamics*, vol. 5 of *Texts in Computational Science and Engineering*. Springer, Berlin, 2007.
16. F. Hedman and A. Laaksonen: *Ewald summation based on nonuniform fast Fourier transform*. Chem. Phys.Lett., 425:142 – 147, 2006.
17. R.W. Hockney and J.W. Eastwood: *Computer simulation using particles*. Taylor & Francis, Inc., Bristol, PA, USA, 1988.
18. 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.
19. J. Keiner, S. Kunis, and D. Potts: *NFFT 3.0, C subroutine library*. `http://www.tu-chemnitz.de/~potts/nfft`.
20. 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.
21. S. Kunis and D. Potts: *Time and memory requirements of the nonequispaced FFT*. Sampl. Theory Signal Image Process., 7:77 – 100, 2008.
22. S. Kunis, D. Potts, and G. Steidl: *Fast Gauss transform with complex parameters using NFFTs*. J. Numer. Math., 14:295 – 303, 2006.
23. 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.
24. 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, May 2010. Forschungszentrum Jülich. Technical Report.
25. G. Pöplau, D. Potts, and U. van Rienen: *Calculation of 3D space-charge fields of bunches of charged particles by fast summation*. In A. Anile, G. Alì, and G. Mascaly (eds.): *Scientific Computing in Electrical Engineering*, pp. 241 – 246, Berlin Heidelberg, Germany, 2006. Springer.
26. D. Potts: *Schnelle Fourier-Transformationen für nichtäquidistante Daten und Anwendungen*. Habilitation, Universität zu Lübeck, `http://www.tu-chemnitz.de/~potts`, 2003.
27. D. Potts and G. Steidl: *Fast summation at nonequispaced knots by NFFTs*. SIAM J. Sci. Comput., 24:2013 – 2037, 2003.
28. D. Potts, G. Steidl, and A. Nieslony: *Fast convolution with radial kernels at nonequispaced knots*. Numer. Math., 98:329 – 351, 2004.
29. 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.
30. G. Steidl: *A note on fast Fourier transforms for nonequispaced grids*. Adv. Comput. Math., 9:337 – 353, 1998.
31. G. Sutmann: *Molecular dynamics - vision and reality*. In J. Grotendorst, S. Blügel,

and D. Marx (eds.): *Computational Nanoscience: Do It Yourself!*, vol. 31 of *NIC Series*, pp. 159 – 194, Jülich, Feb. 2006. Forschungszentrum Jülich, John von Neumann Institute for Computing, ISBN 3-00-017350-1. Lecture Notes.

32. A.F. Ware: *Fast approximate Fourier transforms for irregularly spaced data*. SIAM Rev., 40:838 – 856, 1998.
33. T.T. Wong, W.S. Luk, and P.A. Heng: *Sampling with Hammersley and Halton Points*. Journal of graphics tools, 2:9 – 24, 1997.