Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Fast Convolution Quadrature for Wave Equation in Three Dimensions

(revised version: December 2013)

by

Lehel Banjai and Maryna Kachanovska

Preprint no.: 67 2012

Fast Convolution Quadrature for the Wave Equation in Three Dimensions ☆

L. Banjai[a,1], M. Kachanovska[b,*]

[a] *Department of Mathematics, Heriot-Watt University, Edinburgh, EH14 4AS, UK*
[b] *Max Planck Institute for Mathematics in the Sciences, Inselstr. 22, 04103 Leipzig, Germany.*

Abstract

This work addresses the numerical solution of time-domain boundary integral equations arising from acoustic and electromagnetic scattering in three dimensions. The semidiscretization of the time-domain boundary integral equations by Runge-Kutta convolution quadrature leads to a lower triangular Toeplitz system of size N. This system can be solved recursively in an almost linear time ($O(N \log^2 N)$), but requires the construction of $O(N)$ dense spatial discretizations of the single layer boundary operator for the Helmholtz equation. This work introduces an improvement of this algorithm that allows to solve the scattering problem in an almost linear time.

The new approach is based on two main ingredients: the near-field reuse and the application of data-sparse techniques. Exponential decay of Runge-Kutta convolution weights $w_n^h(d)$ outside of a neighborhood of $d \approx nh$ (where h is a time step) allows to avoid constructing the near-field (i.e. singular and near-singular integrals) for most of the discretizations of the single layer boundary operators (near-field reuse). The far-field of these matrices is compressed with the help of data-sparse techniques, namely, $\mathcal{H}$-matrices and the high-frequency fast multipole method. Numerical experiments indicate the efficiency of the proposed approach compared to the conventional Runge-Kutta convolution quadrature algorithm.

Keywords: wave scattering, time-domain boundary integral equations, Runge-Kutta convolution quadrature, $\mathcal{H}$-matrices, fast multipole method, boundary element method
2000 MSC: 65M38,35L05

1. Introduction

Many physical applications, e.g. transient acoustic or electromagnetic scattering, require the solution of the three-dimensional scalar or vector wave equation outside of a bounded obstacle. An elegant approach to treating problems that are posed in unbounded domains is offered by the use of boundary integral equations. However, compared to the field of elliptic problems, efficient

☆ A brief review of the main results of this paper has been published in the Proceedings of the 6th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012)

* Corresponding author. Tel: +49 (0) 341- 9959 - 774
Email addresses: l.banjai@hw.ac.uk (L. Banjai), kachanov@mis.mpg.de (M. Kachanovska)

[1] A major part of this work has been done during the stay of the first author at the Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

solvers for time domain boundary integral equations (TDBIE) are not that extensively developed. A comprehensive review of the available methods at the time of publishing is given in [1].

A rigorous theory of retarded potentials for the acoustic wave equation dates back to 1986 [2, 3]. The numerical solution of corresponding integral equations is traditionally performed using time-domain Galerkin methods [4, 5, 6, 7, 8], collocation methods [9, 10] or Laplace-domain approaches (see [1], as well as [11, 12] for the application of the method for problems of elasticity).

Galerkin methods [4] often require the underlying spatial quadrature to be evaluated with high accuracy and become extremely complicated in the case of curved boundary elements. This kind of issue was recently overcome with the help of specially designed time basis functions, see [8, 13, 14].

To achieve higher accuracies, marching-on-in-time (MOT) solvers (collocation in time) require that time steps are chosen very small, which can lead to instabilities on long time intervals. To solve this problem, in [15] a new procedure for the accurate MOT matrix element evaluation was suggested. The field of fast solvers that are based on MOT is relatively well-developed. Recent advances in this field include the seminal works [16, 17] on the plane-wave time-domain algorithm, [18, 19] on time-domain adaptive integral equation methods and [20, 21] on the nonuniform (Cartesian) grid time-domain algorithms.

An alternative approach to treating time-domain boundary integral equations was suggested by C. Lubich [22, 23, 24]. The convolution quadrature (CQ) method combines Laplace transform techniques and the usual time-stepping approach and results in a stable and efficient algorithm.

The history of the application of convolution quadrature to problems of wave propagation starts with the work [24] where multistep CQ was employed to discretize the indirect time-domain boundary integral formulation for the wave equation. In [25, 26, 27] convolution quadrature was applied to discretize the TDBIE arising from visco- and poroelasticity. For such applications convolution quadrature is of particular importance, since it requires only the Laplace transform of the fundamental solution to be explicitly known. In [28] convolution quadrature was applied to the time-domain boundary integral formulation of the wave equation with non-zero initial conditions.

In [29, 30, 31] a range of numerical experiments demonstrating properties of the convolution quadrature was conducted. The use of Runge-Kutta CQ allows to achieve arbitrary high convergence rates, see [32, 33]. The use of the method does not require the application of sophisticated spatial quadratures, and hence it can be straightforwardly applied when boundary elements are curvilinear. Compared to multistep convolution quadrature, Runge-Kutta convolution quadrature has low dissipation and dispersion, see [34, 35] for a quantifiable definition and analysis of these properties, as well as numerical experiments in [29]. Recent works [36, 37] provide the analysis of multistep convolution quadrature combined with the Galerkin discretization for the scattering by a sound-hard obstacle, as well as suggest the procedure of the reduced convolution weight computation. In [38] the convolution quadrature formulation for the Maxwell equations was studied analytically and numerically.

To our knowledge, despite many attractive features of CQ, there exist very few fast convolution quadrature methods. Particularly, the method of [39, 40, 41], namely sparse convolution quadrature, though offering a great improvement both in the asymptotic complexity and in constants in complexity estimates, does not allow to compute the solution in linear time. Another fast algorithm, directional FMM accelerated convolution quadrature of [42], requires the solution of many Helmholtz integral formulations with wavenumbers that have large real and small imaginary part, see also [29]. Currently there are, to our knowledge, no efficient preconditioners for this kind of problems. In [43] the authors developed a multistep CQ method based on the fast multipole accelerated BEM of [44, 45]. Though this algorithm performs better than most conventional convolution quadrature

methods for many practical problems, the total solution time still does not scale linearly.

In this paper we propose an improved algorithm for the solution of the time-domain boundary integral equations with retarded potentials based on the Runge-Kutta convolution quadrature discretization [46]. As a model problem we consider acoustic wave scattering in three dimensions. The use of the method results in the sparsity of convolution weights [47], a property that our approach heavily relies on. We show how the application of this property combined with data-sparse techniques allows to reduce storage and computational costs required by the conventional recursive algorithm of [23]. The sparsification of convolution weights has already been employed in [41, 40, 39], but our approach is different. We construct the algorithm based on the method of originally linear complexity rather than quadratic used in [41]. Due to the recursive nature of the algorithm we are able to employ fast techniques based on analytic expansions (namely, the high-frequency fast multipole method). We also use Runge-Kutta methods instead of linear multistep ones.

This work is organized as follows. In Section 2 we state the boundary integral formulation for the wave equation, apply Runge-Kutta convolution quadrature to it and discuss some properties of Runge-Kutta convolution weights. Next, we briefly describe the conventional recursive algorithm our approach is based on. This algorithm relies on the construction of many Galerkin discretizations of single layer boundary operators for the Helmholtz equation with decay. Hence, the use of data-sparse techniques ($\mathcal{H}$-matrices and high-frequency fast multipole method of [48, 49]) is required.

However, a simple application of these methods does not allow to avoid the evaluation of singular and near-singular Galerkin integrals (so called near-field), which in practice takes a significant part of the total computation time. To justify this statement, in Section 4 we review the main ideas of data-sparse techniques and present some related studies. In Section 5 we describe an approach that allows to evaluate only a small part of singular and near-singular integrals. In the final sections we present numerical experiments that confirm the efficiency of the suggested method and discuss open questions related to the problem.

2. Convolution Quadrature Discretization for Wave Scattering

Let $\Omega \subset \mathbb{R}^3$ be a bounded Lipschitz domain, Γ be its boundary and $\Omega^c = \mathbb{R}^3 \setminus \overline{\Omega}$ its complement. As a model problem we consider acoustic scattering:

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2}(t,x) - \Delta u(t,x) &= 0 \qquad \text{in} \quad [0,T] \times \Omega^c, \\ u(0,x) = \frac{\partial u}{\partial t}(0,x) &= 0, \qquad x \in \Omega^c, \\ u(t,x) &= g(t,x) \qquad \text{on} \quad [0,T] \times \Gamma. \end{aligned} \tag{1}$$

There exist several time-domain boundary integral formulations of this problem. For a concise introduction into the theory of TDBIE see [2, 3, 4, 50]. In this work we employ the indirect boundary integral formulation. The solution $u(t,x)$ can be represented as the single-layer potential of an

unknown density λ:

$$u(t,\tilde{x}) = (\mathcal{S}\lambda)(t,\tilde{x}) = \int_0^t \int_\Gamma \frac{\delta(t-\tau-\|\tilde{x}-y\|)}{4\pi\|\tilde{x}-y\|}\lambda(\tau,y)d\Gamma_y d\tau, \qquad (t,\tilde{x}) \in [0,T]\times\Omega^c,$$

where $\delta(t)$ is the Dirac δ-function. Since the single layer potential is continuous across Γ, letting in the above equation $\tilde{x} \to x \in \Gamma$ and using the boundary condition from (1), we obtain the boundary integral equation for the unknown λ, see also [2],

$$g(t,x) = (\mathcal{V}\lambda)(t,x) = \int_0^t \int_\Gamma \frac{\delta(t-\tau-\|x-y\|)}{4\pi\|x-y\|}\lambda(\tau,y)d\Gamma_y d\tau, \qquad (t,x) \in [0,T]\times\Gamma. \tag{2}$$

The density $\lambda(t,x)$ is causal, namely, $\lambda(t,x) = 0,\ t \le 0$. To discretize (2) in time, we apply convolution quadrature based on Runge-Kutta methods. Here we provide only a very brief description of this approach. More details can be found in [24, 46, 29].

Let an m-stage Runge-Kutta method be given by its Butcher tableau:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}.$$

The corresponding stability function is defined by $R(z) = 1 + zb^T(I - Az)^{-1}\mathbb{1}$, where $\mathbb{1} = (1,\ \dots\ ,1)^T$. For Runge-Kutta methods of order p it is an approximantion to the exponential of the same order:

$$R(z) = \mathrm{e}^z + C_{p+1}z^{p+1} + O(z^{p+2}), \qquad \text{for} \quad z \to 0,\ C_{p+1} \neq 0.$$

We will only use Runge-Kutta methods with a nonsingular matrix A satisfying the following assumptions.

Assumption 2.1. *(a) A-stability, namely* $|R(z)| \le 1$ *for all* z, *s.t.* $\operatorname{Re} z \le 0$.

(b) stiff accuracy, i.e. $R(\infty) = 0$.

(c) for all $y \in \mathbb{R} \setminus \{0\}$, $|R(iy)| < 1$.

(d) if p *is even,* $p \ge m$.

Assumptions (a), (b) and (c) are required by the theory of Runge-Kutta convolution quadrature, see [33], and the assumption (d) is a technical assumption needed for the proof of the sparsity of convolution weights in [47]. Typical examples of Runge-Kutta methods that satisfy these assumptions are Radau IIA and Lobatto IIIC.

The time interval $[0,\ T]$ is subdivided into $N+1$ time steps of size h. By g_n and $\boldsymbol{g}_n$ we denote functions

$$g_n(x) = g(nh,x), \qquad \boldsymbol{g}_n(x) = (g(nh+c_1h,x),\dots,g(nh+c_mh,x))^T$$

and define λ_n and $\boldsymbol{\lambda}_n$ similarly. The Runge-Kutta convolution quadrature discretization of (2) results in the following discrete convolution

$$\begin{aligned} \boldsymbol{g}_n(x) &= \sum_{k=0}^{n} W_{n-k}^h \boldsymbol{\lambda}_k, \\ g_{n+1}(x) &= b^T A^{-1} \sum_{i=0}^{n} \left(W_{n-i}^h \boldsymbol{\lambda}_i\right)(x), \qquad n = 0, \dots, N-1, \end{aligned} \tag{3}$$

where $W_k^h : \left(H^{-\frac{1}{2}}(\Gamma)\right)^m \to \left(H^{\frac{1}{2}}(\Gamma)\right)^m$, $k = 0, \dots, N-1$, are boundary integral operators. Namely,

$$\left(W_k^h \boldsymbol{\phi}\right)(x) = \int_\Gamma w_k^h(\|x-y\|)\boldsymbol{\phi}(y)d\Gamma_y,\ k = 0, \dots, N-1. \tag{4}$$

We will refer to the kernels $w_n^h : \mathbb{R}_{\geq 0} \to \mathbb{R}^{m\times m}$ as convolution weights. In the Runge-Kutta convolution quadrature theory they are defined as follows. The convolution equation (2) in the Laplace domain (denoting by $(\mathcal{L}f)(s)$ the Laplace transform of the causal function f) takes the form:

$$(\mathcal{L}g)(s,x) = \int_\Gamma \frac{\mathrm{e}^{-s\|x-y\|}}{4\pi\|x-y\|}(\mathcal{L}\lambda)(s,y)d\Gamma_y, \qquad x \in \Gamma,\ s \in \mathbb{C}. \tag{5}$$

Let also $\Delta : \mathbb{C} \to \mathbb{C}^{m\times m}$ be a matrix-valued function defined by the underlying Runge-Kutta method:

$$\Delta(\xi) = \left(A + \frac{\xi}{1-\xi}\mathbb{1}b^T\right)^{-1} = A^{-1} - \xi A^{-1}\mathbb{1}b^T A^{-1}.$$

The last equation can be obtained with the help of the Sherman-Woodbury-Morrison formula, see [29]. Convolution weights are then the coefficients of the expansion of the integral kernel of (5) with the Laplace variable s substituted by $\frac{\Delta(\xi)}{h}$:

$$\frac{\exp\left(-\frac{\Delta(\xi)}{h}\|x-y\|\right)}{4\pi\|x-y\|} = \sum_{n=0}^{\infty} w_n^h\left(\|x-y\|\right)\xi^n, \qquad \xi \in \mathbb{C} :\ |\xi| < 1. \tag{6}$$

The Laplace transform of the operator $\mathcal{V}$, see (2), is the single layer boundary integral operator of the Helmholtz equation:

$$\begin{aligned} V(s) &: H^{-\frac{1}{2}}(\Gamma) \to H^{\frac{1}{2}}(\Gamma), \\ \left(V(s)\lambda\right)(x) &= \int_\Gamma \frac{\mathrm{e}^{-s\|x-y\|}}{4\pi\|x-y\|}\lambda(y)d\Gamma_y,\ s \in \mathbb{C},\ \operatorname{Re} s > 0. \end{aligned}$$

Using (6) we obtain

$$\left(V\left(\frac{\Delta(\xi)}{h}\right)\phi\right)(x) = \int_\Gamma \frac{\exp\left(-\frac{\Delta(\xi)}{h}\|x-y\|\right)}{4\pi d}\phi(y)d\Gamma_y$$
$$= \sum_{n=0}^{\infty}\int_\Gamma w_n^h\left(\|x-y\|\right)\phi(y)d\Gamma_y\xi^n,$$

which, with the help of (4), lets us obtain the generating function for W_n^h:

$$V\left(\frac{\Delta(\xi)}{h}\right) = \sum_{n=0}^{\infty} W_n^h(V)\xi^n. \tag{7}$$

The existence of the expansions (6) and (7), as well as approximation properties of (3) are proved in [33].

The semidiscretized system (3) is lower triangular Toeplitz, similarly to the ones resulting from the discretization in time of the convolution equation (2) by most MOT and Galerkin methods [51]. Spatial Galerkin discretizations of corresponding boundary integral operators with, say, test and trial piecewise constant basis functions are sparse, due to the Huygens principle. This is also true in a sense for discretizations of boundary integral operators W_n^h (some of the entries of these discretizations are approximately equal to zero), under additional constraints on the Runge-Kutta method. In [41] it was proved that convolution weights of the second order backward difference method $\omega_n^h(d)$ decay away from $d \approx nh$, and in [47] the same property was deduced for convolution weights of Runge-Kutta methods satisfying Assumption 2.1. Here we present a simplified version of the result of [47].

Proposition 2.2. *Let w_n^h, $n \geq 0$, be convolution weights of a Runge-Kutta method that satisfies Assumption 2.1. Then there exist constants G, G', C, $C' > 0$, $0 < \alpha, \beta < 1$ and $\bar{\delta} \in (0,\ 1)$, such that for $n \geq 1$, $h > 0$, and $0 < \delta < \bar{\delta}$ the following estimates hold:*

$$\|w_n^h(d)\| \leq \frac{G}{d}(1-\delta)^{n-\frac{d}{h}}(1+C\delta^\alpha)^{\frac{d}{h}} \qquad \text{for} \quad \frac{d}{h} \leq n,$$
$$\|w_n^h(d)\| \leq \frac{G'}{d}(1+\delta)^{n-\frac{d}{h}}(1+C'\delta^\beta)^{\frac{d}{h}} \qquad \text{for} \quad \frac{d}{h} > n;$$

The convolution weight $w_0^h(d)$ satisfies for all $h > 0$:

$$\|w_0^h(d)\| \leq \frac{\exp(-\mu\frac{d}{h})}{4\pi d},$$

with some $\mu > 0$.

All the constants depend only on the Runge-Kutta method and not on n, d or h.

Explicit expressions for α and β in the above proposition can be found in [47].

Assumption 2.1 is crucial for decay properties of the convolution weights. For example, the trapezoidal rule does not satisfy (b) in Assumption 2.1. Numerical experiments in [29] show that convolution weights $w_n^h(d)$ of the trapezoidal rule do not exhibit exponential decay as $\frac{d}{h} \leq n$.

The following proposition is a corollary of decay properties of convolution weights and is crucial for our algorithm. It shows that in a certain range of d the convolution sum (6) can be truncated to a finite sum with an arbitrary accuracy (c.f. [47]).

Proposition 2.3. *Let w_n^h, $n \geq 0$, be Runge-Kutta convolution weights, and let the Runge-Kutta method satisfy Assumption 2.1.*

Let $K > 0$ be fixed. There exist $\mu_1, \mu_2 > 0$ and $\mu_3 \in \mathbb{R}$ that do not depend on K, s.t. for all $\epsilon > 0$ and for all $L \in \mathbb{N}$ satisfying

$$L \geq \mu_1 \log \frac{1}{\epsilon} + \mu_2 K + \mu_3,$$

the following holds true for arbitrary $h > 0$.

1. *The convolution kernel $\mathcal{K}_d(\xi) = \frac{\exp\left(-\Delta(\xi)\frac{d}{h}\right)}{4\pi d}$ can be approximated with the accuracy $\epsilon > 0$ by the polynomial of the degree $L-1$:*

$$\left| \mathcal{K}_d(\xi) - \sum_{\ell=0}^{L-1} w_\ell^h(d) \xi^\ell \right| \leq \frac{\epsilon}{4\pi d}$$

   for all $\xi \in \mathbb{C} : \ |\xi| \leq 1$ and $0 \leq d \leq Kh$.

2. *Convolution weights can be approximated with the accuracy ϵ by the following L-term discrete Fourier transform of the convolution kernel:*

$$\left| w_n^h(d) - \frac{1}{L} \sum_{\ell=0}^{L-1} \mathcal{K}_d(\mathrm{e}^{i\ell\frac{2\pi}{L}}) \mathrm{e}^{-i\ell n \frac{2\pi}{L}} \right| \leq \frac{\epsilon}{4\pi d}$$

   for all $n < L$ and $0 \leq d \leq Kh$.

3. Conventional Recursive Algorithm for the Solution of a Lower Triangular Toeplitz System

Equations (3) for $n = 0, \ldots, N-1$ constitute the following lower triangular block Toeplitz system:

$$\begin{pmatrix} W_0^h & 0 & \cdots & 0 \\ W_1^h & W_0^h & \cdots & 0 \\ \vdots & & & \\ W_{N-1}^h & W_{N-2}^h & \cdots & W_0^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_0 \\ \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_{N-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{N-1} \end{pmatrix}. \tag{8}$$

In this section we describe the conventional recursive algorithm for the solution of this system.

3.1. Algorithm

In our description of the algorithm for the solution of a lower triangular Toeplitz system we closely follow [52].

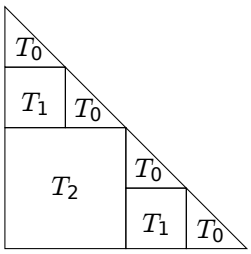

Figure 1: The structure of the matrix in the system (8).

The main idea of the recursive algorithm is to substitute the solution of the full system by the solution of identical small lower triangular systems and matrix-vector multiplications with Toeplitz matrices of different sizes. The structure of the matrix of the system (8) is shown in Figure 1. The identical subblocks are marked with the same letters. The algorithm proceeds recursively as follows. First, the lower triangular system T_0 (that has the same structure as the large system) is solved. Next, a matrix-vector multiplication of the Toeplitz matrix T_1 with the vector known after the solution of the system T_0 is performed; the result is substracted from the right-hand side. After that the lower triangular system T_0 is solved. Next, the matrix-vector multiplication with T_2 is performed, and the lower triangular system consisting of two blocks T_0 and one block T_1 needs to be solved.

Remark 1. The levels of the recursive algorithm are enumerated starting from the base case. For example, assuming that T_0 is solved directly, i.e. without the use of the recursive procedure, we will say that the matrix-vector products with T_1 are computed on the first stage or level of the algorithm and the matrix-vector products with T_2 on the second level.

The size of T_0 in the recursive algorithm has to be chosen constant, and hence large systems of size n have $O(\log n)$ such levels. Clearly, the matrix-vector multiplications with large blocks on higher levels are performed more rarely then matrix-vector multiplications with small blocks on lower levels.

Let us introduce the basic procedures of the algorithm.

`Solve` (n_0, n_1, $\boldsymbol{g}$, $\boldsymbol{\lambda}$) - solves recursively the following system of equations

$$\begin{pmatrix} W_0^h & 0 & \cdots & 0 \\ W_1^h & W_0^h & \cdots & 0 \\ \vdots & & & \\ W_{n_1}^h & W_{n_1-1}^h & \cdots & W_0^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_{n_0} \\ \boldsymbol{\lambda}_{n_0+1} \\ \vdots \\ \boldsymbol{\lambda}_{n_1+n_0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_{n_0} \\ \boldsymbol{g}_{n_0+1} \\ \vdots \\ \boldsymbol{g}_{n_1+n_0} \end{pmatrix}.$$

`Multiply` (k, n, p, l, $\boldsymbol{\lambda}$, $\boldsymbol{h}$) - performs the matrix-vector multiplication

$$\begin{pmatrix} \boldsymbol{h}_l \\ \boldsymbol{h}_{l+1} \\ \vdots \\ \boldsymbol{h}_{l+n-k} \end{pmatrix} = \begin{pmatrix} W_k^h & W_{k-1}^h & \cdots & W_1^h \\ W_{k+1}^h & W_k^h & \cdots & W_2^h \\ \vdots & & & \\ W_n^h & W_{n-1}^h & \cdots & W_{n-k+1}^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_p \\ \boldsymbol{\lambda}_{p+1} \\ \vdots \\ \boldsymbol{\lambda}_{p+k-1} \end{pmatrix}. \tag{9}$$

`SolveTri` (n_0, n_1, $\boldsymbol{g}$, $\boldsymbol{\lambda}$) - solves the following triangular system of equations directly

$$\begin{pmatrix} W_0^h & 0 & \cdots & 0 \\ W_1^h & W_0^h & \cdots & 0 \\ \cdots & & & \\ W_{n_1-n_0}^h & W_{n_1-n_0-1}^h & \cdots & W_0^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_{n_0} \\ \boldsymbol{\lambda}_{n_0+1} \\ \vdots \\ \boldsymbol{\lambda}_{n_1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_{n_0} \\ \boldsymbol{g}_{n_0+1} \\ \vdots \\ \boldsymbol{g}_{n_1} \end{pmatrix}.$$

Let us fix $J > 0$: every system of size smaller or equal to $J + 1$ is to be solved directly. By procedure `Solve`, larger systems are split in two and solved recursively, until their size reaches $J+1$ - in this case they are solved by `SolveTri`. A pseudocode of this procedure is given below.

Function `Solve` (n_0, n_1, $\boldsymbol{G}$, $\boldsymbol{\Lambda}$)
if ($n_1 - n_0 \leq J$) **then**
  `SolveTri`(n_0, n_1, $\boldsymbol{G}$, $\boldsymbol{\Lambda}$);
else
  $n_{\frac{1}{2}} = \left\lfloor \frac{n_0+n_1}{2} \right\rfloor$;
  `Solve`(n_0, $n_{\frac{1}{2}}$, $\boldsymbol{G}$, $\boldsymbol{\Lambda}$);
  `Multiply` ($n_{\frac{1}{2}} - n_0 + 1$, $n_1 - n_0$, n_0, $n_{\frac{1}{2}} + 1$, $\boldsymbol{\Lambda}$, $\boldsymbol{H}$);
  $\boldsymbol{G}|_{n_{\frac{1}{2}}+1,\ldots,n_1} = \boldsymbol{G}|_{n_{\frac{1}{2}}+1,\ldots,n_1} - \boldsymbol{H}|_{n_{\frac{1}{2}}+1,\ldots,n_1}$;
  `Solve`($n_{\frac{1}{2}} + 1$, n_1, $\boldsymbol{G}$, $\boldsymbol{\Lambda}$);
end if
endFunction

The question of the efficient solution of the small system (procedure `SolveTri`) was addressed in detail in [53]. Since improving this algorithm is not a subject of the present work, we omit its description and refer the reader to the aforementioned paper. For the details of the efficient application of data-sparse techniques, in particular $\mathcal{H}$-matrices, for the solution of the small system, as well as available preconditiners see [29].

Complexity of the full algorithm depends heavily on the complexity of matrix-vector product (9). We will show that for every such matrix in (9) of size n, the matrix-vector product can be performed in $O(n \log n)$ time, thus resulting in an almost linear $O(N \log^2 N)$ complexity of the entire algorithm.

3.2. Conventional Matrix-Vector Multiplication

In this section we briefly explain the algorithm for the fast matrix-vector multiplication

$$\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{n-\ell} \end{pmatrix} = \begin{pmatrix} W_\ell^h & W_{\ell-1}^h & \cdots & W_1^h \\ W_{\ell+1}^h & W_\ell^h & \cdots & W_2^h \\ \vdots & & & \\ W_n^h & W_{n-1}^h & \cdots & W_{n-\ell+1}^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_0 \\ \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_{\ell-1} \end{pmatrix} \tag{10}$$

following the description in [52, 23, 54, 55]. The above matrix is Toeplitz, and therefore can be extended to a twice larger circulant matrix that can be further diagonalized with the help of the discrete Fourier transform. The corresponding matrix-vector product can be rewritten for some $0 < \rho \leq 1$ (the use of this parameter will be explained later) as:

$$\boldsymbol{h}_\rho = \begin{pmatrix} W_0^h & \cdots & W_1^h\rho \\ W_1^h\rho & \cdots & W_2^h\rho^2 \\ \vdots & & \\ W_{\ell-1}^h\rho^{\ell-1} & \cdots & W_\ell^h\rho^\ell \\ W_\ell^h\rho^\ell & \cdots & W_{\ell+1}^h\rho^{\ell+1} \\ W_{\ell+1}^h\rho^{\ell+1} & \cdots & W_{\ell+2}^h\rho^{\ell+2} \\ \vdots & & \\ W_n^h\rho^n & \cdots & W_0^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_0 \\ \boldsymbol{\lambda}_1\rho \\ \vdots \\ \boldsymbol{\lambda}_{\ell-1}\rho^{\ell-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathcal{F}_{n+1}^{-1} D_{n+1,\rho} \mathcal{F}_{n+1} \boldsymbol{\lambda}_\rho, \tag{11}$$

where

$$\begin{aligned} &\boldsymbol{h}_\rho = [\boldsymbol{h}_0,\ \ldots,\ \boldsymbol{h}_n\rho^n]^T,\ \boldsymbol{h}_{\ell+j} = \boldsymbol{r}_j,\ j = 0, \ldots, n-\ell, \\ &\boldsymbol{\lambda}_\rho = \left[\boldsymbol{\lambda}_0,\ \boldsymbol{\lambda}_1\rho \ldots \boldsymbol{\lambda}_{\ell-1}\rho^{\ell-1},\ 0,\ \ldots\ , 0\right]^T, \end{aligned}$$

$\mathcal{F}_{n+1}$ is the discrete Fourier matrix of size $(n+1) \times (n+1)$ and $D_{n+1,\rho}$ is the diagonal matrix with entries $\hat{d}_{jj} = \sum_{k=0}^{n} W_k^h \rho^k \mathrm{e}^{-i\frac{2\pi}{n+1}kj},\ j = 0, \ldots, n$. Using (7), we obtain an alternative expression for $\hat{d}_{jj}$:

$$\hat{d}_{jj} = V\left(\frac{\Delta(\rho \mathrm{e}^{-i\frac{2\pi}{n+1}j})}{h}\right) - \sum_{k=n+1}^{\infty} W_k^h \rho^k \mathrm{e}^{-i\frac{2\pi}{n+1}kj}. \tag{12}$$

In [29, 55] it was shown that it is possible to choose the parameter ρ so that the second term in (12) can be discarded. Instead of the matrix $D_{n+1,\rho}$ we can then use the diagonal matrix $\mathcal{D}_{n+1,\rho} = \mathcal{D}_{n+1,\rho}(V)$ with elements

$$d_{jj} = V\left(\frac{\Delta(\rho \mathrm{e}^{-i\frac{2\pi}{n+1}j})}{h}\right), \qquad j = 0, \ldots, n. \tag{13}$$

The choice of the parameter ρ depends on the desired accuracy of the evaluation of (10). It can be shown that given $\epsilon_0 > 0$ the choice $\rho = \epsilon_0^{\frac{1}{2n+1}}$ is optimal and ensures that the matrix-vector product (10) is evaluated with the accuracy E_{MV} bounded by

$$E_{MV} \leq K\sqrt{\epsilon_0}, \tag{14}$$

for some $K > 0$ independent of ϵ_0. The best achievable accuracy is $K\sqrt{\text{eps}}$, where eps is the machine precision.

Hence, (11) can be rewritten as:

$$\boldsymbol{h}_\rho \approx \mathcal{F}_{n+1}^{-1}\mathcal{D}_{n+1,\rho}(V)\mathcal{F}_{n+1}\boldsymbol{\lambda}_\rho. \tag{15}$$

This allows to evaluate the matrix-vector product (10) in $O(n \log n)$ steps.

Remark 2. To compute (10) we have to construct $O(n)$ discretizations of the boundary integral operator $V(s)$. For the whole system of size N the total number of such discretizations scales as $O(N)$.

Remark 3. Since in the time domain all values are real, after the DFT because of the symmetry only a half of matrix-vector multiplications need to be performed, the other half are obtained by complex conjugation. Therefore, it is sufficient to construct discretizations of the boundary integral operators $V\left(\frac{\Delta\left(\rho \mathrm{e}^{-i\frac{2\pi}{n+1}j}\right)}{h}\right)$ for $j = 0, \ldots, \left\lfloor \frac{n+1}{2} \right\rfloor$, and compute matrix-vector products only with these matrices.

4. Data-Sparse Techniques

For the space discretization we employ a Galerkin method. Let $(\phi_j(x))_{j=1}^M$ span a finite dimensional subspace of $H^{-\frac{1}{2}}(\Gamma)$. Since $V(s) : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$, we will use this as both the test and trial space. We assume that the boundary Γ is subdivided into $\tilde{M}$ panels π_i. Let the meshwidth be equal to Δx. We assume that $\tilde{M} = O\left(\frac{1}{(\Delta x)^2}\right)$. For further considerations it is important that each of the functions $(\phi_i(x))_{i=1}^M$ is supported only on a few boundary elements, i.e., for some constants $c,\ C > 0$,

$$\operatorname{diam}(\operatorname{supp}\phi_i) = C_i\Delta x, \qquad c < C_i < C,\ i = 1, \ldots, M.$$

For simplicity we assume that $(\phi_j(x))_{j=1}^M$ are piecewise-constant basis functions and $\operatorname{supp}(\phi_j) = \pi_j,\ j = 1, \ldots, M = \tilde{M}$. All the arguments of this section can be trivially extended to a more general case.

Under the above assumptions, the Galerkin discretization of the operator $V(s)$ is an $M \times M$ matrix $\boldsymbol{V}(s)$ with elements

$$(\boldsymbol{V}(s))_{ij} = \iint\limits_{\Gamma\times\Gamma} \frac{\mathrm{e}^{-s\|x-y\|}}{4\pi\|x-y\|}\phi_i(x)\phi_j(y)d\Gamma_x d\Gamma_y, \qquad i,j = 1, \ldots, M.$$

Remark 4. We assume that the Dirichlet data in (1) is temporally and spatially (approximately) bandlimited to a frequency f_m. For many applications the case $f_m \gg 1$ is of importance. Then the values of N, M are chosen so that $M = O(N^2)$. The primary reason for such a choice is the sampling condition: the meshwidth has to be chosen as $\Delta x \approx C_1 f_m^{-1}$ and the time step $h \approx C_2 f_m^{-1}$, for some constants $C_1, C_2 > 0$.

In [34, 35] the authors analyzed dissipation and dispersion errors associated with the time discretization, for both multistep and Runge-Kutta convolution quadrature. The results of these studies can be summarized as follows:

- if the direct integral formulation is used and the domain is convex, the time step has to be chosen as $h \approx C_2 f_m^{-1}$;
- otherwise $h \approx C_2 f_m^{-1-\frac{1}{p}}$, where p is the classical order of the Runge-Kutta method (or the order of the multistep method).

The errors due to the spatial discretization (also for the Maxwell equations) were analyzed in [24, 53, 38, 56]. In this work we assume $\Delta x \approx h$, similarly to MOT methods. We did not encounter significant pollution effects with such a choice, at least for the range of discretizations of interest.

To perform the matrix-vector multiplication (10) one has to construct $O(n)$ Galerkin discretizations of the Helmholtz single-layer boundary operator $V(s)$. This leads to the following difficulties:

- the resulting matrices are densely populated. Moreover, a substantial part of the corresponding frequencies s (the eigenvalues of $\frac{\Delta(\xi)}{h}$ with ξ lying on a circle of a radius $\rho < 1$, see (13)) are located close to the imaginary axis (see Figure 2). This is typical for convolution quadrature based on Runge-Kutta methods of high order, see [29, Lemma 3.1] and [47, Lemma 3.1].
- the construction of Galerkin discretizations requires the evaluation of many weakly-singular and nearly-singular integrals

$$\int_{\pi_i}\int_{\pi_j} \frac{\mathrm{e}^{-s\|x-y\|}}{4\pi\|x-y\|} d\Gamma_y d\Gamma_x, \qquad \mathrm{dist}(\pi_i,\ \pi_j) < C\Delta x,\ C > 0. \tag{16}$$

  A commonly employed technique for the numerical integration is the coordinate transformation [57, 58], which requires that the number of quadrature points to evaluate one such integral increases as $O(\log^k M)$, $k \leq 4$. The power k depends on the distance between the panels.

The first difficulty can be overcome by the use of data-sparse techniques, such as $\mathcal{H}$-matrices and fast multipole methods. The application of these methods allows to assemble, store and perform the matrix-vector multiplication with matrices $\boldsymbol{V}(s)$ in almost linear time, i.e. $O(M \log^\alpha M)$, $\alpha \geq 0$.

To deal with the second problem, we modify the procedure of the matrix-vector multiplication described in Section 3.2, so that only a small number of singular and near-singular integrals is computed. Before presenting the numerical evidence of the improvement given by the use of the suggested procedure (as well as the description of the procedure itself), we review recent studies which demonstrate that as long as data-sparse techniques are employed, the evaluation of singular integrals requires a significant part of the total matrix assembly time.

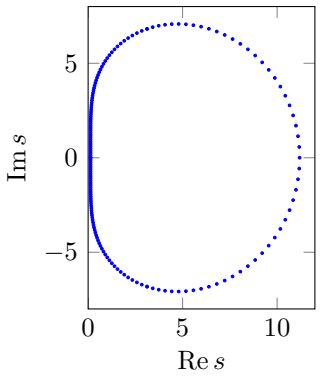

Figure 2: Frequencies s for which we need to construct discretizations of single-layer potentials $\boldsymbol{V}(\frac{s}{h})$; they are computed as eigenvalues of $\Delta(\xi)$. Here we use values $|\xi| = 10^{-\frac{6}{105}}$ and $h = 1$ as an example.

This section is organized as follows. First, we recall basic notions related to data-sparse techniques. The second part of the section provides some important details on the application of data-sparse techniques to the algorithm of Section 3.2. The final part is dedicated to the motivation of the procedure of the near-field reuse described in Section 5.

4.1. Data-Sparse Techniques: Main Ideas and Definitions

Below we provide some auxiliary definitions due to [59, 60] crucial for the understanding of the class of data-sparse techniques we rely on.

First, let us introduce a hierarchical partitioning of the boundary of the domain Γ.

Definition 1. Given a constant C, a tree $\mathcal{T}_\mathcal{I}$ is called a cluster tree corresponding to an index set $\mathcal{I}$ if $\mathcal{T}_\mathcal{I}$ is a binary labeled tree with the following properties:

- the label $\hat{\tau}$ of a vertex τ of $\mathcal{T}_\mathcal{I}$ is a subset of $\mathcal{I}$;
- the label of the root of the tree is $\mathcal{I}$;
- the label of a vertex τ is a disjoint union of labels of its sons;

The set of the leaves of the cluster tree $\mathcal{T}_\mathcal{I}$ is denoted by $\mathcal{L}(\mathcal{T}_\mathcal{I})$. The structure of the cluster tree introduces a hierarchical subdivision of Γ into sets of panels.

The bounding box of a cluster τ is the (axis-parallel) box containing the set Ω_τ; the center of the box we denote by c_τ and its diameter by d_τ. Next, let us introduce a special relation on the pairs of clusters, or the admissibility condition. In the $\mathcal{H}$-matrix theory, this relation is chosen so that the matrix blocks corresponding to the pairs of admissible clusters can be approximated by low-rank matrices.

Definition 2. We will call a pair of clusters (τ, σ) admissible if for some fixed $\eta > 1$ the following holds true:

$$\|c_\tau - c_\sigma\| \geq \frac{\eta}{2}(d_\tau + d_\sigma).$$

Now let us introduce the concept of the admissible block-cluster tree. We adopt here a slightly modified definition, similar to the one used in the high-frequency fast multipole method [49]. In the $\mathcal{H}$-matrix theory it corresponds to the level-consistent admissible block-cluster tree.

Definition 3. Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree. We will call an admissible block-cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ a subtree of a labeled tree $\mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{I}}$ that satisfies the following conditions:

- the root of the tree is $(\mathrm{root}(\mathcal{T}_{\mathcal{I}}), \mathrm{root}(\mathcal{T}_{\mathcal{I}}))$.
- the son clusters of each block-cluster $b = (\tau, \sigma)$ are defined by

$$\mathrm{sons}(b) = \begin{cases} \{(\tau', \sigma'),\ \tau' \in \mathrm{sons}(\tau),\ \sigma' \in \mathrm{sons}(\sigma)\}, & \mathrm{sons}(\tau) \neq \emptyset \text{ and } \mathrm{sons}(\sigma) \neq \emptyset, \\ \emptyset, & \text{else;} \end{cases}$$

- a block-cluster (τ, σ) is a leaf if and only if one of the following holds true:
  1. (τ, σ) is admissible;
  2. (τ, σ) is not admissible, and $\tau \in \mathcal{L}(\mathcal{T}_{\mathcal{I}})$ or $\sigma \in \mathcal{L}(\mathcal{T}_{\mathcal{I}})$;

Thus, all the leaves of the admissible block-cluster tree can be split into two sets, namely $\mathcal{L}_+(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$ of admissible block-clusters and $\mathcal{L}_-(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$ of non-admissible block-clusters. The first set is called the **far-field**, while the second one is referred to as the **near-field**.

Further, we assume that the cluster tree is constructed so that $\mathrm{depth}(\mathcal{T}_{\mathcal{I}}) = O(\log \#\mathcal{I})$. Let $n_{\min} \in \mathbb{N}_+$ be s.t. for all $(\tau, \sigma) \in \mathcal{L}_-(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$:

$$\#\tau \leq n_{\min} \ \text{ and } \ \#\sigma \leq n_{\min}.$$

Additionally, let the diameter of clusters that have non-admissible neighbours be bounded by $C\Delta x$, where Δx is the meshwidth and C is a constant that does not depend on the meshwidth.

From the above considerations it follows that the far-field consists of distant pairs of clusters, while the near-field contains pairs (τ, σ) s.t. $\tau = \sigma$ or $\mathrm{dist}(\tau, \sigma)$ is of $O(\Delta x)$. Let us denote by $\boldsymbol{V}(s)|_{\hat{\tau}\times\hat{v}}$ the following matrix block:

$$(\boldsymbol{V}(s)|_{\hat{\tau}\times\hat{v}})_{k_i\ell_j} = (\boldsymbol{V}(s))_{ij}, \qquad k_i \in \{1, \ldots, n\},\ \ell_j \in \{1, \ldots, m\},\ i \in \hat{\tau},\ j \in \hat{v}.$$

By definition the near-field contains blocks $(\boldsymbol{V}(s))_{\tau\times\tau}$, $\tau \in \mathcal{L}(\mathcal{T}_{\mathcal{I}})$, i.e. those blocks that contain weakly singular integrals of the type (16).

The main idea behind data-sparse techniques lies in the efficient compression of the matrix blocks corresponding to the far-field. Namely, the fast multipole method allows to diagonalize these blocks using computationally inexpensive multipole-to-multipole (local-to-local) transformations, see also [61, 49]. Such transformations typically come from analytic expansions of the fundamental solution of the Helmholtz equation. The $\mathcal{H}$-matrix approximations are based on the similar idea: the far-field blocks are approximated by low-rank matrices. Such approximations can be done with the help of tensor product interpolation [62], however, this requires the a priori knowledge about the ranks of the matrices. For an alternative approach (adaptive cross-approximation, ACA) that relies solely on the evaluation of the elements of a BEM matrix see [63] and [64].

4.2. Remarks on the Application of Data-Sparse Techniques

In order to approximate the matrices

$$(\boldsymbol{V}(s))_{ij} = \int_\Gamma \int_\Gamma \frac{\mathrm{e}^{-s\|x-y\|}}{4\pi\|x-y\|}\phi_i(x)\phi_j(y)d\Gamma_x d\Gamma_y, \qquad i,j = 1,\ldots,M,$$

for different values of $s \in \mathbb{C}$, $\operatorname{Re} s \geq 0$, we employ compression based on $\mathcal{H}$-matrix techniques [59, 60] and the multilevel fast multipole method, namely its high-frequency version (HF FMM) described in [49] and based on expansions of [48], of the complexity $O(M \log^2 M)$ (depending on the implementation, the complexity $O(M \log M)$ is achievable as well).

Whilst the HF FMM allows to approximate the matrix $\boldsymbol{V}(s)$ with an almost linear complexity in the high-frequency regime, i.e. $|s| = O\left(\sqrt{M}\right)$, $\mathcal{H}$-matrices perform worse, with the complexity $O(M^{\frac{3}{2}} \log^k M)$, $k \geq 0$, see [65, Section 3.4.5.2]. However, the complexity of $\mathcal{H}$-matrix approximations scales almost linearly ($O(M \log^k M)$) if the wavenumber $s \in \mathbb{C}$ satisfies $\frac{|\operatorname{Im} s|}{|\operatorname{Re} s|} < c$, for some $c > 0$. This was demonstrated in [65, p.114, Theorem 3.18, p.157] based on the results from [66], as well as in [29]. Additionally, an $\mathcal{H}$-matrix approximation is of almost linear complexity in the low-frequency regime, see [65, Section 3.4.5.2].

To avoid the low-frequency breakdown of the fast multipole method (see [67, 68]) we employ the decomposition introduced in [69]:

$$\boldsymbol{V}(s) = \mathcal{H} + \mathcal{H}^2. \tag{17}$$

Specifically, the matrix $\boldsymbol{V}(s)$ is represented as a sum of two matrices, with an $\mathcal{H}$-matrix-part approximating the near-field and small block-clusters that are subject to the low-frequency breakdown of the fast multipole method, and an $\mathcal{H}^2$-matrix-part which approximates the rest and is constructed using expansions coming from the high-frequency fast multipole method, see [69, 70]. For more details on $\mathcal{H}^2$-matrices we refer to [71, 60, 72]. The advantages and disadvantages of both techniques have been analyzed in [73]. The results of this study suggest that the fast multipole assembling time is small compared to that of $\mathcal{H}$-matrices. On the other hand, matrix-vector multiplication performed by the fast multipole method, though is proved to be of asymptotic complexity $O(M \log^2 M)$, is in practice considerably slower than that for $\mathcal{H}$-matrices, even for large problems (about 10^5 unknowns).

This disadvantage of the HF FMM does not play a significant role in our algorithm. Many matrix-vector multiplications are required only at early stages of the recursive algorithm (see Remark 1), but in this case, as shown in [29], wavenumbers are either small or have large decaying part. Hence, $\boldsymbol{V}(s)$ involved in such matrix-vector products can be efficiently approximated with the help of $\mathcal{H}$-matrix techniques. The rest of the matrices needed for later stages of the recursive algorithm are involved only in several matrix-vector multiplications, which makes a mixed $\mathcal{H} + \mathcal{H}^2$-approximation (17) advantageous in this case. However, in the case when the frequency s in $\boldsymbol{V}(s)$ has a moderate or large non-zero real part, use of $\mathcal{H}$-matrices pays off in terms of storage costs and matrix-vector multiplication times. One of our observations was that for moderate accuracies (i.e. the accuracy of the matrix approximation $\epsilon \geq 10^{-7}$) it is better to use a pure $\mathcal{H}$-matrix based approximation for $\boldsymbol{V}(s)$ if $\operatorname{Re} s \geq |\operatorname{Im} s|$ (the case of prevailing decay). A detailed analysis of the applicability of $\mathcal{H}$-matrices and HF FMM to Helmholtz equation with decay can be found in the technical report [74].

Both techniques allow to compress matrix blocks corresponding to admissible clusters (the far-field), while the rest, namely the near-field, has to be stored as dense matrices. This part is computationally demanding: for example, to represent the BEM discretized Helmholtz single-layer potential matrix of the size $M \times M$ as an $\mathcal{H}$-matrix, one has to compute $O(M)$ 4-dimensional weakly singular and nearly singular integrals of the form $\int\limits_{\pi_i}\int\limits_{\pi_j} \frac{\mathrm{e}^{-s\|x-y\|}}{\|x-y\|}\phi_i(x)\phi_j(y)d\Gamma_x d\Gamma_y$.

4.3. Motivation for the Approach

The evaluation of the near-field integrals is commonly done with the help of coordinate transformation techniques, see [58, 75, 76, 77, 57]. Given the kernel $k(x, y)$ of a single layer boundary operator, for the evaluation of

$$\iint\limits_{\pi_i\times\pi_j} k(x,y)\phi_i(x)\phi_j(y)d\Gamma_x d\Gamma_y, \qquad \operatorname{supp}\phi_i = \pi_i,\ \operatorname{supp}\phi_j = \pi_j,$$

with the accuracy sufficient to preserve the stability (and convergence) of the Galerkin method, it is required that the number of quadrature points scales as $O(\log^3 M)$ (or even as $O\left(\log^4 M\right)$ for the Helmholtz single-layer boundary operator in the high-frequency regime, as observed in [78, p. 86]) if $\operatorname{dist}(\pi_i, \pi_j) = 0$, $O(\log^4 M)$ if $\operatorname{dist}(\pi_i, \pi_j) = O(\Delta x)$ (nearly singular integrals) and $O(1)$ if $\operatorname{dist}(\pi_i, \pi_j)$ is of order $O(1)$. Thus the computation of the near-field (singular and nearly singular integrals) of one matrix is of $O(M \log^4 M)$ complexity. Within the recursive convolution quadrature algorithm $O(N)$ such matrices need to be assembled, hence resulting in the total complexity $O\left(NM \log^4 M\right)$.

The question of the efficient evaluation of singular and nearly singular integrals was addressed in recent works [79, 80, 81]. Particularly, in [79] such integrals were represented as functions of multiple parameters and efficiently computed using interpolation and tensor decomposition techniques. In [78] the effect of the application of such techniques on the total $\mathcal{H}$-matrix assembly time was numerically studied. For the Laplace single layer boundary operator on various geometries it was demonstrated that that the 50%-70% reduction of the time required for the evaluation of the near-field results in the 10%-20% reduction of the total $\mathcal{H}$-matrix assembly time. Given the bound r on the ranks of an $\mathcal{H}$-matrix, the rest of the time is spent for the evaluation of $O(rM \log M)$ far-field integrals within the ACA+ procedure of the $\mathcal{H}$-matrix construction (see [63, 82]). If the evaluation of the far-field is done in a more efficient manner, the gain can be significantly larger. And this is the case for the fast multipole methods.

The precomputation time (i.e. time needed for the construction of the translation operators) for the HF FMM scales as $O(M \log M)$ (assuming $M = O(|\kappa|^2)$ for the wavenumber κ) and the constants involved are significantly smaller than that for the $\mathcal{H}$-matrix assembly. This can be seen in the experiments of [73] for the Burton-Miller formulation for the acoustic problem in a half-space. In this work the performance of the HF FMM (correspondingly modified for the efficient application with the half-space mirror techniques, see [83]) was compared to the performance of $\mathcal{H}$-matrices. The HF FMM precomputation times were reported to be in practice 9-20 times smaller than that for the $\mathcal{H}$-matrix construction.

In [84, Tables 3.2-3.3] the time to compute the near-field for the HF FMM accelerated Burton-Miller formulation was compared to the time needed to construct the corresponding HF FMM translation matrices. The results show that for BEM discretizations with $10^3 - 10^5$ triangular boundary elements the computation of the near-field is typically an order of magnitude slower than the assembly of translation matrices.

The actual constants depend much on the implementation and the desired accuracy. Nevertheless, for large problems we should be able to see the improvement if we skip constructing the near-field.

Since in the course of the recursive algorithm described in Section 3 the matrix-vector multiplication with the same matrix block is performed multiple times, it makes sense to precompute the corresponding discretizations of boundary integral operators and keep them in memory, rather than recompute them every time the matrix-vector multiplication is needed. For the matrices that are approximated with the help of the fast multipole method the near-field and translation operators (multipole-to-multipole and multipole-to-local) can be stored. If only a small part of matrices has the near-field, the storage costs needed for HF FMM approximated matrices can be affected as well. The storage for the translation matrices of the FMM scales as $S_{ff}(s) = O(|\kappa|^2 \log M) = O(M \log M)$, while for the near-field $S_{nf}(s) = O(M)$. Clearly, as $M \to +\infty$, S_{nf} is smaller than S_{ff} (though only by a logarithmic term). However, numerical experiments in Section 6 show that constants in S_{ff} are so small that even for rather large M, $S_{nf} > S_{ff}$, and hence the near-field reuse results in the improvement of storage costs.

The presence of decay, i.e. in the case when in $\boldsymbol{V}(s)$ $s = s_1 + is_2$, $s_1 > 0$, facilitates the reduction of storage costs. If s_1 is large enough, the far-field integrals

$$\iint\limits_{\pi_i \times \pi_j} \frac{\mathrm{e}^{-s\|x-y\|}}{4\pi\|x-y\|} \phi_i(x)\phi_j(y) d\Gamma_x d\Gamma_y \approx 0, \qquad \operatorname{dist}(\pi_i,\ \pi_j) > C\Delta x,\ C > 0.$$

For such discretizations $\boldsymbol{V}(s)$ the far-field part $S_{ff} \approx 0$, see also Figure 2.

5. Fast Runge-Kutta Convolution Quadrature

Our algorithm improves the matrix-vector multiplication (3.2) procedure of the algorithm of Section 3.1. We start this section with the introduction of auxiliary relations on the leaves of a block-cluster tree.

5.1. Near-Field Reuse

Before describing our strategy for dealing with the near-field, we introduce two auxiliary relations defined on leaves of a block-cluster tree, namely the near-field d-admissibility and the far-field d-admissibility.

Definition 4. Given $d > 0$, we will call a leaf (τ, σ) near-field d-admissible if $\|c_\tau - c_\sigma\| < d - \frac{d_\tau}{2} - \frac{d_\sigma}{2}$.

Definition 5. Given $D > 0$, a leaf (τ, σ) is far-field D-admissible if $\|c_\tau - c_\sigma\| < D + \frac{d_\tau}{2} + \frac{d_\sigma}{2}$.

Remark 5. The following properties hold:

1. If (τ, σ) is near-field d-admissible then
   $(\forall x \in \Omega_\tau)(\forall y \in \Omega_\sigma),\ \|x - y\| < d.$
2. If (τ, σ) is not far-field D-admissible then
   $(\forall x \in \Omega_\tau)(\forall y \in \Omega_\sigma),\ \|x - y\| > D.$

We will denote the set of near-field d-admissible leaves of a block-cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ by $\mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$ and the set of far-field D-admissible leaves by $\mathcal{L}_D^+(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$.

Remark 6. The following observation is crucial for our algorithm. Recall that $\mathcal{L}_-(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$ is defined as the set of all non-admissible block-clusters of the block-cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$. Then it is possible to choose d so that

$$\mathcal{L}_-(\mathcal{T}_{\mathcal{I}\times\mathcal{I}}) \subset \mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}}). \tag{18}$$

This follows from the definition of the admissibility condition (Definition 2). Namely, given $\eta > 1$, non-admissible leaves (τ, σ) satisfy

$$\|c_\tau - c_\sigma\| < \frac{\eta}{2}(d_\tau + d_\sigma),$$

where $c_\tau,\ c_\sigma$ are the centers of bounding boxes of $\tau,\ \sigma$ and $d_\tau,\ d_\sigma$ are their diameters. Let $\gamma \geq \frac{\eta+1}{2}$ be fixed. Then the choice

$$d = \gamma \sup_{(\tau,\sigma)\in\mathcal{L}_-} (d_\tau + d_\sigma) \tag{19}$$

ensures that (18) holds true.

Now we have all the ingredients needed to describe fast Runge-Kutta convolution quadrature. Consider the matrix-vector product (10), namely

$$\begin{pmatrix} \boldsymbol{h}_0 \\ \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_{n-\ell} \end{pmatrix} = \begin{pmatrix} W_\ell^h & W_{\ell-1}^h & \cdots & W_1^h \\ W_{\ell+1}^h & W_\ell^h & \cdots & W_2^h \\ \vdots & & & \\ W_n^h & W_{n-1}^h & \cdots & W_{n-\ell+1}^h \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_0 \\ \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_{\ell-1} \end{pmatrix}. \tag{20}$$

After the discretization in space with the help of the Galerkin method (with trial and test basis functions $(\phi_j(x))_{j=1}^M$), the above system of equations can be rewritten as:

$$\begin{pmatrix} \boldsymbol{h}_0^j \\ \boldsymbol{h}_1^j \\ \vdots \\ \boldsymbol{h}_{n-\ell}^j \end{pmatrix} = \iint\limits_{\Gamma\times\Gamma} \sum_{k=1}^{M} T^{\ell,n}(\|x-y\|) \begin{pmatrix} \boldsymbol{\lambda}_0^k \\ \boldsymbol{\lambda}_1^k \\ \vdots \\ \boldsymbol{\lambda}_{\ell-1}^k \end{pmatrix} \phi_k(y)\phi_j(x) d\Gamma_x d\Gamma_y, \qquad j = 1, \ldots, M, \tag{21}$$

where

$$\boldsymbol{h}_k^j = \int\limits_\Gamma \boldsymbol{h}_k(x)\phi_j(x)d\Gamma_x, \qquad k = 0, \ldots, n-\ell,\ j = 1, \ldots, M,$$

$$\boldsymbol{\lambda}_k^j = \int\limits_\Gamma \boldsymbol{\lambda}_k(x)\phi_j(x)d\Gamma_x, \qquad k = 0, \ldots, \ell-1,\ j = 1, \ldots, M,$$

and $T^{\ell,n}$ is the kernel function

$$T^{\ell,n}(\|x-y\|) = \begin{pmatrix} w_\ell^h(\|x-y\|) & \cdots & w_1^h(\|x-y\|) \\ w_{\ell+1}^h(\|x-y\|) & \cdots & w_2^h(\|x-y\|) \\ \vdots & & \\ w_n^h(\|x-y\|) & \cdots & w_{n-\ell+1}^h(\|x-y\|) \end{pmatrix}. \tag{22}$$

Let d be chosen as in (19). The double integral in (21) can be split into a sum of two double integrals: one over the leaves of the block-cluster tree belonging to the set $\mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$ and the other being the remainder. Namely,

$$\iint\limits_{\Gamma\times\Gamma} T^{\ell,n}(\|x-y\|)\phi_j(x)\phi_k(y)d\Gamma_x d\Gamma_y = \boldsymbol{N}_{jk} + \boldsymbol{F}_{jk}, \tag{23}$$

$$\boldsymbol{N}_{jk} = \iint\limits_{\Omega_\sigma\times\Omega_\tau,\ (\sigma,\tau)\in\mathcal{L}_d} T^{\ell,n}(\|x-y\|)\phi_j(x)\phi_k(y)d\Gamma_x d\Gamma_y, \tag{24}$$

$$\boldsymbol{F}_{jk} = \iint\limits_{\Omega_\sigma\times\Omega_\tau,\ (\sigma,\tau)\in\mathcal{L}_F} T^{\ell,n}(\|x-y\|)\phi_j(x)\phi_k(y)d\Gamma_x d\Gamma_y, \quad j,k=1,\dots,M,$$

where $\mathcal{L}_d = \mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$, $\mathcal{L}_F = \mathcal{L}(\mathcal{T}_{\mathcal{I}\times\mathcal{I}}) \setminus \mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$. In this case $\boldsymbol{\mathcal{F}} = (\boldsymbol{F}_{jk})_{k,j=1}^M$ does not contain the near-field, since all non-admissible block-clusters belong to $\boldsymbol{\mathcal{N}} = (\boldsymbol{N}_{jk})_{k,j=1}^M$. Since w_j^h are matrix-valued functions, $\boldsymbol{N}_{jk}$ and $\boldsymbol{F}_{jk}$ are tensors.

First, we demonstrate why such a splitting may improve storage and computational costs.

The bounds in Proposition 2.2 show that, for any given $\epsilon > 0$, there exists L, such that

$$\left\|w_j^h(\tilde{d})\right\| < \frac{\epsilon}{4\pi\tilde{d}}, \qquad \text{for all} \quad j \geq L \quad \text{and} \quad \tilde{d} < d. \tag{25}$$

Recall that d is defined by (19). Let $\boldsymbol{\Omega}_d = \bigcup\limits_{(\sigma,\tau)\in\mathcal{L}_d} \Omega_\sigma \times \Omega_\tau$. We assume w.l.o.g. that $L < \min(\ell, n-\ell+1)$. Then some of the elements of the tensor $\boldsymbol{\mathcal{N}}$ are approximately equal to zero. Let us show this. For $k \geq L$,

$$\begin{aligned}\left|\iint\limits_{\boldsymbol{\Omega}_d} w_k^h(\|x-y\|)\phi_j(x)\phi_i(y)d\Gamma_x d\Gamma_y\right| &< \epsilon\left|\iint\limits_{\boldsymbol{\Omega}_d} \frac{|\phi_j(x)\phi_i(y)|}{4\pi\|x-y\|}d\Gamma_x d\Gamma_y\right| \\ &\leq \epsilon \iint\limits_{\Gamma\times\Gamma} \frac{|\phi_j(x)\phi_i(y)|}{4\pi\|x-y\|}d\Gamma_x d\Gamma_y,\ i,j=1,\dots M.\end{aligned}$$

Recall that the single layer boundary operator for the Laplacian is continuous from $L_2(\Gamma) \to L_2(\Gamma)$, see e.g. [58, 41, 53]. Hence, for some $C > 0$ that depends only on Γ it holds:

$$\left| \iint\limits_{\boldsymbol{\Omega}_d} w_k^h(\|x-y\|)\phi_j(x)\phi_i(y)d\Gamma_x d\Gamma_y \right| \leq C\epsilon \|\phi_i\|_{L_2(\Gamma)} \|\phi_j\|_{L_2(\Gamma)}$$

$$\leq C'\epsilon \left(\Delta x\right)^2, \qquad C' > 0, \tag{26}$$

where the last expression follows from the fact that basis functions are supported on a constant number of mesh elements, and hence $\|\phi_i\|_{L_2(\Gamma)} = O\left(\Delta x\right),\ i = 1, \ldots, M$. Then ϵ, and hence L, can always be chosen so that up to a desired precision $\boldsymbol{\mathcal{N}}$ can be rewritten as

$$\boldsymbol{N}_{jk} \approx \iint\limits_{\boldsymbol{\Omega}_d} T_L^{\ell,n}(\|x-y\|)\phi_k(y)\phi_j(x)d\Gamma_x d\Gamma_y, \qquad k, j = 1, \ldots, M, \tag{27}$$

where

$$T_L^{\ell,n}(\|x-y\|) = \begin{pmatrix} 0 & \cdots & w_{L-1}^h(\|x-y\|) & \cdots & w_2^h(\|x-y\|) & w_1^h(\|x-y\|) \\ 0 & \cdots & 0 & \cdots & w_3^h(\|x-y\|) & w_2^h(\|x-y\|) \\ \vdots & & & & & \\ 0 & \cdots & 0 & \cdots & w_{L-1}^h(\|x-y\|) & w_{L-2}^h(\|x-y\|) \\ 0 & \cdots & 0 & \cdots & 0 & w_{L-1}^h(\|x-y\|) \\ \vdots & & & & & \\ 0 & \cdots & 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{28}$$

Hence, to approximate completely the near-field part of the matrix of the system (20), only $O(L)$ Galerkin matrices

$$\left(\widetilde{\boldsymbol{W}_\nu^h}\right)_{kj} = \iint\limits_{\boldsymbol{\Omega}_d} w_\nu^h(\|x-y\|)\phi_k(y)\phi_j(x)d\Gamma_x d\Gamma_y, \qquad k, j = 1, \ldots, M,\ \nu = 1, \ldots, L-1,$$

need to be constructed. In practice we do not assemble these matrices, but rather evaluate the matrix-vector product with $\boldsymbol{\mathcal{N}}$ with the help of either of the two procedures we present below. Before describing these procedures, we would like to show that $L = O\left(\log \frac{1}{\epsilon}\right)$ and does not depend on the size of the system (20).

Recall that the diameter of a cluster τ that has a non-admissible neighbour is $d_\tau = C_\tau \Delta x,\ C_\tau > 0$, where Δx is the meshwidth (this is by construction of the admissible block-cluster tree, see also Section 4.1). Additionally, for all such clusters τ, the constant C_τ is bounded from above and below by constants independent of Δx, see Section 4. Hence, by (19), there exists $\gamma' > 0$, s.t. $d = \gamma' \Delta x$. Since $\Delta x \approx Ch$, for some $C > 0$, see Remark 4, $d = \tilde{\gamma} h,\ \tilde{\gamma} > 0$. Importantly, there exist $\tilde{\gamma}_0, \tilde{\gamma}_1 > 0$, s.t. $\tilde{\gamma}_0 \leq \tilde{\gamma} \leq \tilde{\gamma}_1$, with $\tilde{\gamma}_0, \tilde{\gamma}_1$ independent of h and Δx.

The estimate on L can be obtained from Proposition 2.2, choosing a priori $L \geq \tilde{\gamma}_1 \geq \frac{d}{h}$. Namely, there exist constants $\delta, G, A, \beta > 0$, s.t.

$$\left\| w_k^h(d') \right\| \leq \frac{G}{d'}(1-\delta)^{k-\frac{d'}{h}}(1+A\delta^\beta)^{\frac{d'}{h}}, \quad \text{for all } d' < d, \text{ and } k \geq \frac{d}{h}.$$

Then L can be determined as the smallest integer satisfying

$$\frac{G}{d}(1-\delta)^{L-\frac{d'}{h}}(1+A\delta^\beta)^{\frac{d'}{h}} < \frac{\epsilon}{4\pi d}, \tag{29}$$

for which it is sufficient that

$$G(1-\delta)^{L-\tilde{\gamma}_1}(1+A\delta^\beta)^{\tilde{\gamma}_1} < \frac{\epsilon}{4\pi}.$$

From this it follows that for a fixed accuracy ϵ, $L = O\left(\log \frac{1}{\epsilon}\right)$, where the hidden constant depends on $\tilde{\gamma}_1$.

Therefore, to approximate the full near-field of the system (20) only $O\left(\log \frac{1}{\epsilon}\right)$ matrices need to be constructed, **independently** of the size of this system.

Remark 7. We do not address here the question how $\epsilon > 0$ has to be chosen to preserve the stability and convergence of the method. A full analysis would require the combination of the estimates of [53] and [41]. In particular, in [41] it is shown that the convergence of the sparse BDF2 convolution quadrature is preserved if the convolution weights are cut off with the accuracy ϵ satisfying $\log \frac{1}{\epsilon} = O(\log M) = O(\log N)$. We expect similar estimates to hold for our case as well, since all the errors are linear, and bounds for the errors and operator norms depend on $h, \Delta x$ as powers (positive or negative) of $h, \Delta x$, c.f. [53, 32].

Remark 8. In practice we fix L a priori (we discuss how we choose it later) and choose d as

$$d = \sup\left\{ \tilde{d} : \left\| w_j^h(d') \right\| < \frac{\epsilon}{4\pi d'}, \text{ for all } j \geq L, \ d' \in [0, \tilde{d}] \right\}.$$

This can be rewritten as

$$d = \sup\left\{ \tilde{d} : \left\| d' w_j^h(d') \right\| < \frac{\epsilon}{4\pi}, \text{ for all } j \geq L, \ d' \in [0, \tilde{d}] \right\},$$

By definition of convolution weights (6) $d' w_j^h(d')$, $j \geq 0$, depends only on the ratio $\frac{d'}{h}$ and j, hence such choice ensures that $d = \tilde{\gamma} h$ for some $\tilde{\gamma} > 0$ independent of h.

In this case L has to be chosen large enough for (18) to hold true, i.e. so that d satisfies (19) for some $\gamma \geq \frac{\eta+1}{2}$. This is possible due to the bounds of Proposition 2.2 and computations similar to (29).

Remark 9. Increasing the value of d allows to reuse a part of the far-field as well.

Next the question of the efficient evaluation of a matrix vector product with the system (27) is addressed. We suggest the use of either of the following two methods.

5.1.1. Near-Field Matrix-Vector Multiplication with Diagonalization

The main idea of this approach is that the matrix (28) in (27) can be represented in the form of a Toeplitz matrix, and hence easily diagonalized. Using (26), the matrix (27) can be rewritten in the form

$$\boldsymbol{N}_{jk} \approx \iint\limits_{\boldsymbol{\Omega}_d} \begin{pmatrix} 0 & \cdots & w^h_{L-1}(\|x-y\|) & \cdots & w^h_1(\|x-y\|) \\ 0 & \cdots & w^h_L(\|x-y\|) & \cdots & w^h_2(\|x-y\|) \\ \vdots & & & & \\ 0 & \cdots & w^h_{2L-3}(\|x-y\|) & \cdots & w^h_{L-1}(\|x-y\|) \\ \vdots & & & & \\ 0 & \cdots & 0 & \cdots & 0 \end{pmatrix} \phi_k(y)\phi_j(x) d\Gamma_x d\Gamma_y, \tag{30}$$

for all $k, j = 1, \ldots, M$. This matrix has the same structure as (20). The algorithm for the efficient evaluation of matrix-vector products involving such matrices is based on the embedding of the matrix into a circulant matrix, scaling it with a properly chosen parameter ρ and diagonalizing the resulting matrix with the help of the discrete Fourier transform, see Section 3.2. As $\|x-y\| < d$, for all $(x, y) \in \boldsymbol{\Omega}_d$, see Remark 5, we can make use of Proposition 2.3 to see that the second term in (12) is small even for $\rho = 1$. Hence in this case there is no need for the scaling parameter and no restriction in accuracy, see (14).

This strategy allows to perform the matrix vector multiplication with $\boldsymbol{\mathcal{N}}$ in $O(L \log L)$ steps. It requires assembling the Galerkin matrices $\boldsymbol{\mathcal{M}}^k$ defined as

$$\boldsymbol{\mathcal{M}}^k_{ij} = \iint\limits_{\boldsymbol{\Omega}_d} \frac{\exp\left(-\Delta\left(\mathrm{e}^{-i\frac{2\pi}{2L-2}k}\right)\frac{\|x-y\|}{h}\right)}{4\pi\|x-y\|} \phi_i(x)\phi_j(y) d\Gamma_x d\Gamma_y, \tag{31}$$
$$i, j = 1 \ldots M,\ k = 0, \ldots, L-1,$$

where we also used the fact that half of these matrices are complex conjugates of the rest, see Remark 3. Importantly, these matrices need to be constructed once and can then be reused for all the matrix-vector multiplications of type (20).

Remark 10. Such approximation can be done with arbitrary accuracy, by adjusting the value of $\epsilon > 0$ in (25). For $\|x-y\| < d$, convolution weights $w^h_n(\|x-y\|)$ decay exponentially with $n \geq L$, see Proposition 2.2. Therefore, choosing ϵ being equal to the desired accuracy allows to approximate

$$\frac{\exp\left(-\frac{\Delta(\xi)}{h}\|x-y\|\right)}{4\pi\|x-y\|} = \sum_{k=0}^{\infty} w^h_k(\|x-y\|)\xi^k, \qquad |\xi| \leq 1,$$

by L terms of the above series with an accuracy close to $\frac{\epsilon}{4\pi\|x-y\|}$. This shows some redundancy of the representation (30).

5.1.2. Direct Near-Field Matrix-Vector Multiplication

Remark 10 shows that the representation (30), though allowing to evaluate the matrix-vector product with $\boldsymbol{\mathcal{N}}$ efficiently, may be redundant, in the sense that it requires constructing more

matrices than needed. Alternatively, one could perform a direct matrix-vector multiplication with the matrix (27) of size L in quadratic time. Since $L = O(\log N)$ (see Remark 7), computing this matrix-vector product with a complexity of $O(L^2)$ may increase the time of the solution of the system of equations, but, as it is shown in this section, allows to decrease storage costs as well as the time needed to construct the matrices.

Matrices $\boldsymbol{\mathcal{M}}^k,\ k = 0, \ldots, L-1$, in (31) contain only the near-field and (possibly) a part of the far-field. Therefore, if they are approximated with the help of $\mathcal{H}$-matrix techniques, the time for the computation of corresponding matrix-vector products is linear in their size and in practice is often insignificant.

Let

$$\begin{pmatrix} \boldsymbol{v}_0^i \\ \boldsymbol{v}_1^i \\ \vdots \\ \boldsymbol{v}_{n-\ell}^i \end{pmatrix} = \sum_{j=1}^{M} \boldsymbol{N}_{ij} \begin{pmatrix} \boldsymbol{\lambda}_0^j \\ \boldsymbol{\lambda}_1^j \\ \vdots \\ \boldsymbol{\lambda}_{\ell-1}^j \end{pmatrix}, \qquad i = 1, \ldots, M.$$

This matrix-vector multiplication can be alternatively written as, see (27),

$$\begin{aligned} \boldsymbol{v}_j &= \sum_{k=1}^{L-1-j} \iint\limits_{\boldsymbol{\Omega_d}} w_{k+j}^h(\|x-y\|)\boldsymbol{\lambda}_{\ell-k} d\Gamma_x d\Gamma_y,\ j = 0, \ldots, L-2, \\ \boldsymbol{v}_k &= 0, \qquad k = L-1, \ldots, n-\ell. \end{aligned}$$

Using Proposition 2.3,

$$\boldsymbol{v}_j \approx \frac{1}{L} \sum_{\ell_1=0}^{L-1} \iint\limits_{\boldsymbol{\Omega_d}} \frac{\exp\left(-\Delta\left(\mathrm{e}^{-i\ell_1 \frac{2\pi}{L}}\right)\frac{\|x-y\|}{h}\right)}{4\pi\|x-y\|} \sum_{k=0}^{L-1-j} \mathrm{e}^{i\ell_1(k+j)} u_{\ell-k}, \tag{32}$$

for all $j = 0, \ldots, L-2$. The error of such an approximation of convolution weights $w_\nu^h(\|x-y\|),\ \nu = 1, \ldots, L-1$, is close to $\frac{\epsilon}{4\pi\|x-y\|}$, where ϵ is as in (25), see Remark 10. As before, the L_2-continuity of the single-layer boundary integral operator can be used, similarly to (26), to show how the respective errors can be controlled by a proper choice of $\epsilon > 0$, see also Remark 7.

From the above expression it follows that to perform the matrix-vector multiplication with $\boldsymbol{\mathcal{N}}$ it is sufficient to construct the near-field matrices $\widetilde{\boldsymbol{\mathcal{M}}}^k$:

$$\widetilde{\boldsymbol{\mathcal{M}}}_{ij}^k = \iint\limits_{\boldsymbol{\Omega}_d} \frac{\exp\left(-\Delta(e^{-i\frac{2\pi}{L}k})\frac{\|x-y\|}{h}\right)}{4\pi\|x-y\|} \phi_i(x)\phi_j(y) d\Gamma_x d\Gamma_y,\ k = 0, \ldots, \left\lfloor \frac{L}{2} \right\rfloor.$$

Note that the number of matrices $\widetilde{\boldsymbol{\mathcal{M}}}^k,\ k = 0, \ldots, \left\lfloor \frac{L}{2} \right\rfloor$, is twice smaller than the number of matrices $\boldsymbol{\mathcal{M}}^k,\ k = 0, \ldots, L-1$, see (31).

For most of the experiments we used this approach, since the time overhead due to additional matrix-vector multiplications was smaller than the time needed to construct additional matrices with

the near-field (using the method of Section 5.1). We explicitly remark when we use the approach from the previous section. A heuristic to choose between the two should be based on the number of the matrices reused and the number of time steps. The larger the number of matrices with the near-field is and the larger the number of time steps is, the likelier it is that the algorithm with the diagonalization will perform better. The precise limiting size of the time interval, as well as the critical number of matrices with the near-field that would require the use of the algorithm with the diagonalization has to be determined based on extensive numerical experiments.

5.1.3. Matrix-Vector Multiplication with the Far-Field Matrices

The matrix-vector multiplication, see (23),

$$\sum_{k=1}^{M} \boldsymbol{F}_{jk} \begin{pmatrix} \boldsymbol{\lambda}_0^k \\ \boldsymbol{\lambda}_1^k \\ \dots \\ \boldsymbol{\lambda}_{\ell-1}^k \end{pmatrix}, \qquad j = 1, \dots, M, \tag{33}$$

can be performed as described in Section 3.2, with the help of scaling and FFT. We have to assemble the Galerkin matrices

$$\boldsymbol{L}_{ij}^k = \iint\limits_{\substack{\Omega_\sigma \times \Omega_\tau, \\ (\sigma,\tau) \in \mathcal{L}(\mathcal{T}_{\mathcal{I}\times\mathcal{I}}) \setminus \mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})}} \frac{\exp\left(-\Delta\left(\rho \mathrm{e}^{-i\frac{2\pi}{n+1}k}\right)\frac{\|x-y\|}{h}\right)}{4\pi\|x-y\|}\phi_i(x)\phi_j(y)d\Gamma_x d\Gamma_y,$$

$$i,j = 1,\dots,M,\ k = 0,\dots,\left\lfloor \frac{n+1}{2} \right\rfloor,$$

where, given $\epsilon_0 > 0$, $\rho = \epsilon_0^{\frac{1}{2n+1}}$. Importantly, the near-field does not appear in this computation.

Some additional improvement in storage costs and computational complexity for these matrices can be achieved if one notices that the matrix-vector multiplication (33) involves only convolution weights with indices up to n. The bounds stated in Proposition 2.2 imply that for all $\epsilon > 0$ there exists $D_n > 0$ such that for all $m \le n$ and for all $D > D_n$

$$\left\| w_m^h(D) \right\| \le \frac{\epsilon}{4\pi D}.$$

That is why one can construct the matrices $\boldsymbol{L}^k$, $k = 0, \dots, \lfloor \frac{n+1}{2} \rfloor$ only on the far-field D_n-admissible block-clusters $(\sigma,\tau) \in \mathcal{L}^+_{D_n,F} = \mathcal{L}^+_{D_n}(\mathcal{T}_{\mathcal{I}\times\mathcal{I}}) \setminus \mathcal{L}_d(\mathcal{T}_{\mathcal{I}\times\mathcal{I}})$, see also Remark 5:

$$\boldsymbol{L}_{ij}^k = \iint\limits_{\substack{\Omega_\sigma \times \Omega_\tau, \\ (\sigma,\tau) \in \mathcal{L}^+_{D_n,F}}} \frac{\exp\left(-\Delta(\rho e^{i\frac{2\pi}{n+1}k})\frac{\|x-y\|}{h}\right)}{4\pi\|x-y\|}\phi_i(x)\phi_j(y)d\Gamma_x d\Gamma_y, \tag{34}$$

$$i,j = 1,\dots,M.$$

5.2. Fast CQ Algorithm and Its Complexity

Compared to the conventional recursive algorithm, see Section 3, in the fast CQ algorithm the multiplication with Toeplitz matrix blocks is replaced by one of the improved procedures of Sections 5.1.1 and 5.1.2.

Let us discuss the complexity of this algorithm. Compared to the conventional recursive algorithm, see Section 3, the new algorithm performs an extra block matrix-vector multiplication with the near-field matrices at each Toeplitz matrix-vector multiplication stage. The computational complexity of each of the matrix-vector multiplication with the near-field matrix block is either $O(ML\log L) = O(M\log N\log\log N)$ (if the near-field matrix-vector multiplication with the diagonalization is used) or $O(L^2M) = O(M\log^2 N)$. In total, there are $O(N)$ matrix blocks (21), hence the full complexity of the near-field related matrix-vector multiplications is $O(NM\log^2 N)$.

The number of matrix-vector multiplications with the far-field matrices scales as $O(N\log N)$, while each matrix-vector multiplication requires about $O(M\log M)$ (or $O(M\log^2 M)$, depending on the implementation of the HF FMM) operations. Therefore, the total complexity of the algorithm is $O(NM\log N\log^2 M + NM\log^2 N)$, or $O(NM\log^3 M)$. The memory costs for the near-field matrices scale almost linearly, namely $O(M\log N)$, while for the rest of the matrices as $O(NM\log M)$. The construction times for $\mathcal{H}$-matrices scale as $O(N\mathcal{T}_q M\log M)$, where $\mathcal{T}_q$ is the complexity of the evaluation of the integrals in BEM. Since for the evaluation of the BEM integrals we use the coordinate transformation technique described in detail in [58], T_q scales not worse than $O(\log^4 M)$. The construction times for $\mathcal{H}^2$-matrices scale as $O(NM\log M)$.

Let us additionally remark that the hidden constants in these complexity estimates depend on the accuracy of the matrix approximations.

Thus, the computational complexity of the `Solve` procedure is not worse than $O(NM\log^3 M)$ and the time to construct the matrices $O(NM\log^k M)$, for $k \geq 1$. The storage costs scale as $O(NM\log M)$.

Remark 11. In [53] a rather restrictive condition on the accuracy ϵ of the separable expansions and $\mathcal{H}$-matrix approximation was imposed, suggesting that it has to be proportional to h^α, $\alpha \geq 1$. However, as noted in the same work, this is not too prohibitive when applied to the HF FMM for the Helmholtz kernel and leads to logarithmic ($\log^2 \frac{1}{h} = \log^2 M$) increase of the complexity. Same holds true for $\mathcal{H}$-matrices. In our algorithm they are applied to approximate the discretizations of $\mathcal{V}(s)$ with s either being small or $\left|\frac{\operatorname{Im} s}{\operatorname{Re} s}\right| < C$, for some $C > 0$. In both cases the $\mathcal{H}$-matrix complexity depends on the desired accuracy ϵ as $\log^k \frac{1}{\epsilon}$, for $k \geq 1$.

6. Numerical Experiments

In this section we present the results of the numerical experiments for the solution of the problem of wave scattering by a sound-soft obstacle. In particular, we solve the boundary integral equation (2), namely

$$g(t,x) = \int_0^t \int_\Gamma \frac{\delta(t-\tau-\|x-y\|)}{4\pi\|x-y\|}\lambda(\tau,y)d\Gamma_y, \qquad x \in \Gamma, \tag{35}$$

on the interval $[0,\ T]\,,\ T > 0$. Knowing $\lambda(t, y)$, we compute the (scattered) field outside of the domain Ω using the indirect boundary integral formulation:

$$u(t,x) = \int_0^t \int_\Gamma \frac{\delta(t-\tau-\|x-y\|)}{4\pi\|x-y\|}\lambda(\tau,y)d\Gamma_y, \qquad x \in \Omega^c,\ t \in [0,\ T]\,.$$

We consider several different domains, including a unit sphere, an elongated domain similar to the NASA almond, see [85], and a trapping domain. We demonstrate almost linear complexity of fast Runge-Kutta convolution quadrature, as well as show that it indeed outperforms conventional Runge-Kutta CQ, especially for large problems.

In all the computations of this section the Helmholtz single layer boundary operators are discretized by the Galerkin method with piecewise constant test and trial basis functions. The matrices are approximated with the accuracy $\epsilon_0 = 1e-6$, unless stated otherwise. For all the experiments the 3-stage Radau IIA method of order 5 is used.

To cut off the convolution weights, we fix $L > 0$, $\epsilon = 5e-4$ and choose the parameter d, see (25) and Remark 8, as

$$d = \sup\left\{\tilde{d} : \left\|w_j^h\left(d'\right)\right\| < \frac{\epsilon}{4\pi d'},\ \text{for all } j \geq L,\ d' \in [0,\tilde{d}]\right\}. \tag{36}$$

We use this accuracy setup for all the experiments.

For long-time computations we employ the procedure described in [29] that allows to reduce the amount of matrices to be assembled. Let us briefly describe the main idea of this method. Let the diameter of the domain be equal to D. Given $\epsilon > 0$, there exists N_D, s.t. for all $n > N_D$ and for all $\tilde{d} \leq D$, $\left\|w_n^h\left(\tilde{d}\right)\right\| < \frac{\epsilon}{4\pi\tilde{d}}$. Then Toeplitz matrix blocks of size $N_T > N_D$, see (10), can be substituted by Toeplitz blocks of size N_D. This allows to significantly reduce the number of matrices that need to be constructed. For our accuracy setting the choice $\epsilon = 5e-4$ was always sufficient.

All the experiments of this section were performed with the help of $\mathcal{H}$LIBpro [86] on three clusters of the Max Planck Institute, each having 8x Dual Core AMD Opteron 8220 CPUs with 2.8 GHz and 256 GB RAM. The computation time we show is the total CPU time (excluding the time needed for the communications between CPUs), i.e. CPU time needed to solve the scattering problem on one CPU. It includes the time of construction of all the matrices for the recursive CQ algorithm and the time for the actual solution of the lower triangular Toeplitz system.

As discussed before, we assemble the matrices once and store them on a disk. For the discretizations that are approximated with the help of the fast multipole method we keep all translation operators. We report storage costs, i.e. the disk space needed to keep the precomputed matrices.

Additionally, we introduce the following notation:

- $\mathcal{H}$ denotes the approach that uses $\mathcal{H}$-matrices only and requires the construction of the near-field for all the matrices (the conventional RK CQ algorithm);
- $\mathcal{H}^{sp}$ is the approach based on $\mathcal{H}$-matrices with the near-field reuse;
- $\mathcal{H}^2$ is the algorithm that uses the fast multipole method but does not reuse the near-field;
- $\mathcal{H}^{2,sp}$ is fast Runge-Kutta convolution quadrature based on the near-field reuse and the HF FMM.

6.1. Experiments with a Sphere

In this section we consider scattering by the unit sphere. In the first part we demonstrate that the approach with the near-field reuse allows to obtain the solution with the accuracy not worse than the accuracy of the conventional Runge-Kutta convolution quadrature method. In the second part we consider the scattering of wide-band incident waves and show the efficiency of the new algorithm.

6.1.1. Correctness of the Approach

As the first example, we consider the scattering problem for the unit sphere on the time interval $[0,\ 25]$ for which the explicit solution is known. We choose the Dirichlet data as

$$g(t,x) = g(t) = \mathrm{e}^{-\frac{(t-3)^2}{0.4^2}} \cos 3t, \qquad t \geq 0. \tag{37}$$

Importantly, $|g(0)| < 10^{-24}$. The solution to (35) with such incident wave does not depend on spatial variables, see [13]:

$$\lambda(t) = 2 \sum_{k=0}^{\lfloor \frac{T}{2} \rfloor} g'(t-2k).$$

We fix the time step $h = 0.125$ and choose the spatial discretization with $M = 16200$ triangles. The results of the computation with the conventional Runge-Kutta CQ based on $\mathcal{H}$-matrices ($\mathcal{H}$) and with fast Runge-Kutta CQ ($\mathcal{H}^{2,sp}$) are shown in Figure 3.

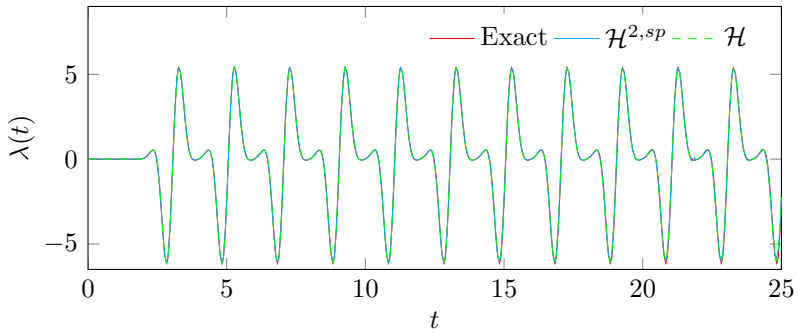

Figure 3: The solution to the problem (37) at one of the points on the sphere. We plot the solution obtained at internal stages of Runge-Kutta convolution quadrature as well.

Let $\tilde{\lambda}_k(x),\ k = 0, \ldots, N$, be the boundary density at the time step $t = kh$ obtained numerically. We measure the relative error of the solution by

$$\epsilon_{\mathrm{rel}} = \frac{\left(h \sum\limits_{k=0}^{N} \|\tilde{\lambda}_k(x) - \lambda(kh,x)\|^2_{H^{-\frac{1}{2}}(\Gamma)} \right)^{\frac{1}{2}}}{\left(h \sum\limits_{k=0}^{N} \|\lambda(kh,x)\|^2_{H^{-\frac{1}{2}}(\Gamma)} \right)^{\frac{1}{2}}}. \tag{38}$$

To compute $\|.\|_{H^{-\frac{1}{2}}(\Gamma)}$, we make use of the results of [2, Proposition 3]. For a given $s \in \mathbb{R}_{>0}$ and $\phi \in H^{-\frac{1}{2}}(\Gamma)$, the equivalent norm in $H^{-\frac{1}{2}}(\Gamma)$ is given by

$$\|\phi\|^2 := \langle \mathcal{V}(s)\phi, \phi \rangle, \tag{39}$$

where $\langle ., . \rangle$ is the sesquilinear duality pairing that extends the inner product on Γ, i.e. $\langle \phi, \psi \rangle = \int_\Gamma \phi(x)\overline{\psi(x)} d\Gamma_x$. For the current experiment the estimate on the norm (39) was found using an $\mathcal{H}$-matrix approximation of $\boldsymbol{V}(s)$ for $s = 20$. The relative error of the solution obtained with the help of fast Runge-Kutta CQ does not exceed $5.31 \cdot 10^{-4}$, and for the solution obtained with the help of $\mathcal{H}$-matrices $\epsilon_{\text{rel}} \approx 5.24 \cdot 10^{-4}$. This shows that the error that stems from the near-field reuse is negligible compared to the error coming from matrix approximations and the discretization.

6.1.2. Scattering of a Wide-Band Signal

Let us consider scattering of the following incident wave:

$$u^{inc}(t, x) = -0.33 \sum_{i=1}^{3} \mathrm{e}^{-\frac{(t-\alpha_i \cdot x - 6\sigma - 1)^2}{\sigma^2}}, \tag{40}$$

with parameters $\alpha_1 = (-1,\ 0,\ 0)\,,\ \alpha_2 = (0,\ -1,\ 0)\,,\ \alpha_3 = (0,\ 0,\ -1)$. The Dirichlet data is given by $g(t, x) = -u^{inc}(t, x)$ and almost vanishes in $t = 0$:

$$|g(0, x)| < 10^{-15}, \qquad x \in \Gamma.$$

In order to resolve the solution for smaller σ, time and spatial discretizations have to be refined, preserving relations $\frac{h}{\sigma} \approx \text{const}$, $\frac{\Delta x}{h} \approx \text{const}$.

At each step of the experiment $k = 1, \ldots, 8$, $\sigma = \sigma_k$ is reduced by a factor $\sqrt{2}$, and the number of time steps N_k on the interval $[0,\ 12.5]$ is increased by the same factor; for the spatial discretization $M_k \approx 2M_{k-1}$. To check the validity of the result obtained for a certain value of σ, we perform the experiment on a finer mesh and compare the scattered field outside of the domain computed on the coarse and fine meshes. The largest $\sigma_{max} = 0.8$, the smallest $\sigma_{min} = 0.07$.

For the first four experiments (discretizations with $N \leq 70$ and $M \leq 8192$) the use of the fast Runge-Kutta convolution quadrature algorithm does not give significant gains compared to the conventional Runge-Kutta algorithm, whereas for the four largest problems, as Table 1 shows, the new algorithm is up to 2.4 times faster than conventional $\mathcal{H}$- or $\mathcal{H}^2$-matrix based approaches. The storage costs are reduced more than 3 times compared to the purely $\mathcal{H}$-matrix based CQ algorithm.

The new algorithm requires more time to solve the system of equations after all the matrices were constructed, which can be attributed to the use of the high-frequency fast multipole method, see the related discussion in Section 4.2. However, it reduces the matrix assembly time drastically compared to the $\mathcal{H}$-matrix based approach.

Figure 4 demonstrates almost linear complexity of the fast Runge-Kutta convolution quadrature algorithm. The time of the matrix construction and memory costs increase sublinearly for the above range of problems. The reason for this is that the assembly (and storage) costs of the full near-field of all the matrices are in this case significantly larger than that required for the far-field. Hence, if only a small part of the near-field is constructed, the total complexity is improved. One can notice that the computation time is close to $O(M \log MN \log N)$ (compared to the estimate

σ	0.2		0.14		0.1		0.07	
h	0.125		0.09		0.0625		0.045	
M	16200		32768		65448		129970	
N	100		139		200		278	
L	14		15		16		16	
$\mathcal{H}$	33.2G	6.2 (0.6)h	118.6G	25.9 (1.6)h	-		-	
$\mathcal{H}^{sp}$	22G	3.9 (0.9)h	78.9G	19.4 (4.2)h	-		-	
$\mathcal{H}^2$	19.7G	6.4 (2)h	57.6G	20.5 (5.9)h	117.3G	97.8 (30.7)h	-	
$\mathcal{H}^{2,sp}$	12.5G	3.9 (1.5)h	34.7G	12.5 (4.2)h	56.8G	40.1 (16)h	145.1G	116.9 (48.6)h

Table 1: Storage costs (in GB) and total CPU time (in hours) for different discretizations and techniques for the problem with the right-hand side defined by (40), time interval [0, 12.5]. In parentheses we show the CPU time needed to solve the system of equations after all the matrices were constructed. Here h is the time step, N is the number of time steps, M is size of the spatial discretization and L is as in (36).

$O(M \log^2 MN \log N)$, see Section 5.2). The reasons for this are explained in the report [74]. In a nutshell, an additional logarithmic factor comes from the HF FMM multipole-to-multipole and local-to-local transforms. However, they are performed only once per cluster, and hence for smaller problems their complexity appears to be negligible compared to the complexity of multipole-to-local transforms.

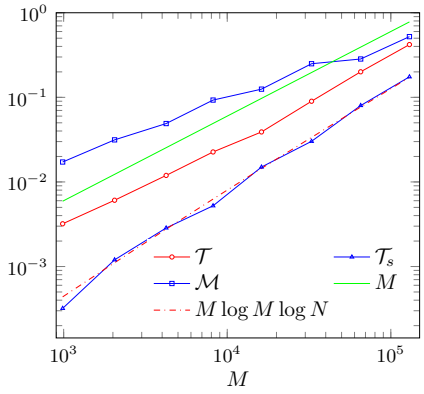

Figure 4: The dependence of the total CPU time per time step $\mathcal{T}$, the CPU time per time step without the time needed for the matrix assembly $\mathcal{T}_s$ and the memory per time step $\mathcal{M}$ on the spatial discretization size M. The time is measured in hours and the memory in GB.

The solutions to the problem for different σ computed outside of the domain, at the point

(2.5, 0, 0), are depicted in Figure 5. Here we depict as well the solutions obtained at internal stages of the Runge-Kutta method. Like in the previous section, the near-field reuse allows to obtain the solution with the same accuracy as the conventional $\mathcal{H}$-matrix based Runge-Kutta CQ algorithm.

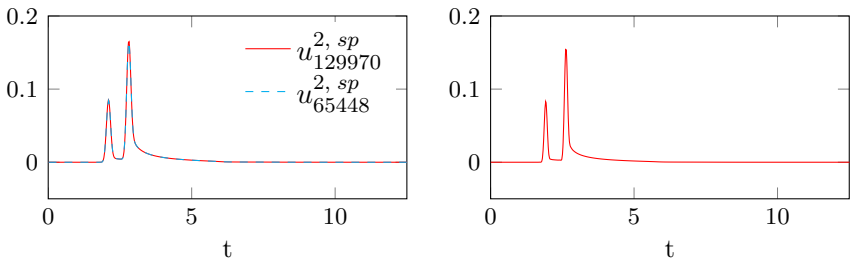

Figure 5: In the left picture we depict the scattered field at the point (2.5, 0, 0) obtained on the discretizations of the domain with 65448 and 129970 triangles, $\sigma = 0.1$: on this scale the solutions are practically indistinguishable. In the right picture the solution computed for $\sigma = 0.07$ is shown. We plot here the solution obtained at internal stages of the Runge-Kutta method as well.

The benefit of the suggested technique applied to scattering by a unit sphere is not as significant for smaller discretizations as for larger ones. Nevertheless, we can see a significant benefit from the use of the algorithm already for problems with 4.5 million unknowns. In further sections we show how the efficiency of the improved recursive algorithm depends on the domain and the problem size.

6.2. Experiments with an Elongated Domain

To demonstrate the efficiency of the algorithm, we perform a set of tests for the domain depicted in Figure 6. The domain and the mesh for it were generated with the help of Gmsh [87]. The length of this domain is 2.5, width 1 and height 0.32.

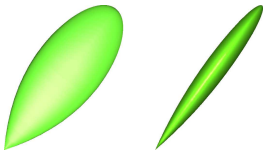

Figure 6: The domain that we use in experiments. The domain is oriented parallel to x-axis; the incoming wave first hits the tip of the domain.

The incident wave used in the experiments is the plane-wave modulated by a Gaussian:

$$u^{inc}(t,x) = -\cos(\omega(t-\alpha\cdot x-6\sigma-A))\mathrm{e}^{-\frac{(x-\alpha\cdot x-6\sigma-A)^2}{\sigma^2}}, \tag{41}$$

with parameters $\alpha = (-1,\,0,\,0)$, $A = 1.45$, $\sigma = \frac{6}{\omega}$. The Dirichlet data is given by $g(t,x) = -u^{inc}(t,x)$ and satisfies, for all σ we considered,

$$|u^{inc}(0,x)| < 10^{-15}, \qquad x \in \Gamma.$$

As previously, in order to resolve the right-hand side for higher frequencies, time and spatial discretizations have to be refined, preserving relations $h\omega \approx \text{const}$, $\frac{\Delta x}{h} \approx \text{const}$.

At each step of the experiment $k = 1, ..., 4$, we double ω_k, i.e. $\omega_k = 2\omega_{k-1}$, as well as increase the number of time steps N_k on the interval $[0,\ 6.4]$ twice. For the spatial discretization $M_k \approx 4M_{k-1}$. To check the validity of the results, we perform the experiment for every frequency ω_k, $k = 1, ..., 4$, on the finer mesh and compare the obtained solutions. The accuracy of the solution for the largest frequency, namely $\omega = 48$, is compared to the solution obtained on the time-space mesh with 92 million unknowns.

We increase the number of matrices to be reused for larger problems in order to improve the performance of the algorithm: it makes sense to reuse also a part of the far-field as the problem size increases, see Remark 9. For the two largest problems we employ the approach for the near-field reuse with the diagonalization, while for the smaller problems the direct approach is used (see Section 5.1.2).

ω	6		12		24		48		48	
h	0.12		0.06		0.03		0.015		0.01	
M	1134		4096		16072		64230		144092	
N	54		107		214		427		640	
L	21		24		24		26		37	
$\mathcal{H}$	0.95G	0.38 (0.01)h	11.5G	3 (0.08)h	159.7G	43.9 (1)h	-		-	
$\mathcal{H}^{sp}$	0.42G	0.13 (0.02)h	4.7G	1.1 (0.3)h	113.4G	32.2 (2.4)h	-		-	
$\mathcal{H}^2$	1.15G	0.28 (0.03)h	9.8G	2.5 (0.5)h	71.7G	23.6 (6.4)h	-		-	
$\mathcal{H}^{2,sp}$	0.42G	0.12 (0.03)h	4.2G	1.2 (0.4)h	30.9G	12.4 (5)h	169G	135.8 (47.2)h	414.3G	371.2 (158)h

Table 2: Storage costs stated in GB and total CPU times in hours for different discretizations and techniques for the problem with the right-hand side defined by (41), time interval $[0,\ 6.4]$. In parentheses we show the CPU time needed to solve the system of equations after all the matrices were constructed. Here h is the time step, N is the number of time steps, M is size of the spatial discretization and L is as in (36).

Storage costs and computation times for the solution of the problem with different approaches are shown in the Table 2. Numerical experiments show that the algorithm based on $\mathcal{H}^2$-matrices with the near-field reuse is more than 3 times faster than the conventional $\mathcal{H}$-matrix based method and allows to reduce storage costs 2-5 times. In the conventional $\mathcal{H}$-matrix based approach the matrix assembly time is significantly larger than the actual system solution time, and the use of the fast Runge-Kutta CQ algorithm allows to reduce this time significantly.

In Figures 7, 8 we plot the solutions outside of the domain, at the distance 1 from the tip of the domain (at the point $x_0 = (2.5,\ 0,\ 0)$). We show as well the error

$$e_n = e(nh) = |\tilde{u}_n^N(x_0) - \tilde{u}_n^{2N}(x_0)|, \qquad n = 1, \ldots, N. \tag{42}$$

Here $\tilde{u}_n^N(x_0)$ is the scattered field at the point x_0 obtained on the discretization with N time steps and

M spatial degrees of freedom. The quantity $\tilde{u}_n^{2N}(x_0)$ is the scattered field at the point x_0 computed with the help of fast Runge-Kutta CQ on the finer spatial discretization (with approximately $4M$ degrees of freedom) and $2N$ time steps.

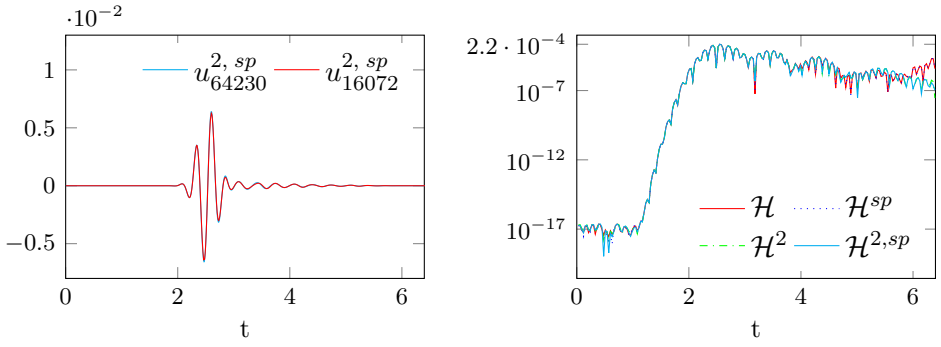

Figure 7: In the left plot we depict the solution computed for $\omega = 24$ (also at internal stages), while in the right plot we show the errors (42) for the same solution obtained with the help of different techniques. The errors that stem from the near-field reuse are negligible compared to the matrix approximation and discretization errors.

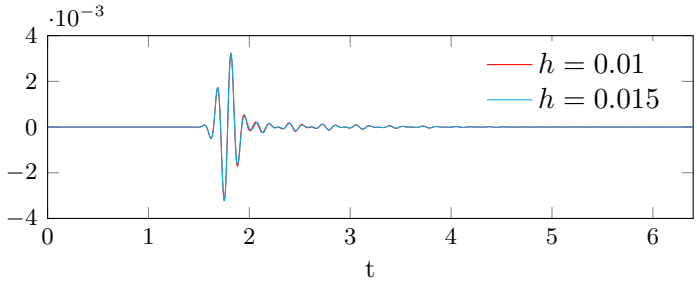

Figure 8: The solution for $\omega = 48.0$ at the point (2.5, 0, 0) obtained on two different discretizations, with $h = 0.015$ and $h = 0.01$ (we plot as well the solution computed on internal stages). Both solutions are in a good agreement.

The above results show that already for problems with 60000 unknowns the use of the FMM-based approach with the near-field reuse allows to obtain noticeable performance gain without deterioration of accuracy compared to the conventional $\mathcal{H}$-matrix- and FMM-based CQ algorithms.

6.3. Experiments with a Trapping Domain

In [88] it is shown that for a class of 2-dimensional domains, that contain an elliptic cavity, the condition number of the combined field integral formulation for the exterior Helmholtz problem grows exponentially with the frequency. Hence, for larger frequencies, the scattering problem seems to be better suited for the solution in the time domain. We consider the 3-dimensional domain of

the diameter 2.0 formed by rotating the 2D trapping domain in question; this domain is depicted in Figure 9. The domain and mesh for it were generated with the help of Gmsh [87].

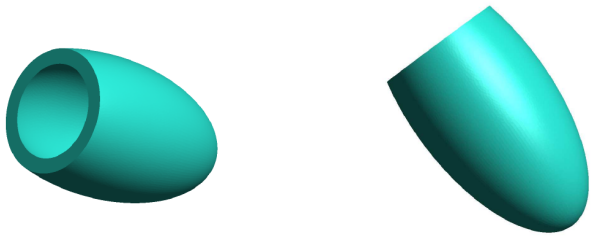

Figure 9: The trapping domain that we use in experiments. The domain is oriented parallel to x-axis; the incoming wave first hits the cavity.

We solve the scattering problem with the incident wave

$$u^{inc}(t,x) = -\exp\left(-\omega^2(t - x\cdot\alpha - 6\sigma - A)^2\right), \tag{43}$$

where ω varies from $\frac{11}{3} \approx 3.667$ to 28 and A is chosen so that in $t = 0$, for all $x \in \Gamma$ and all ω in the above range, $\left|u^{inc}(0,x)\right|$ does not exceed $2\cdot 10^{-10}$. For the three smallest experiments $A = 0.5$ and for the two largest experiments $A = 0.588$. As before, to control the error, we compare the solution in a point outside of the domain to the solution in the same point computed on a finer discretization, see (42).

The vector $\alpha = (1,\ 0,\ 0)$ is oriented along the axis of the rotation of the domain. The results of the numerical experiments are shown in Table 3. For all the experiments we employ the algorithm of the near-field reuse with the diagonalization, see Section 5.1.1.

Results in Table 3 show that storage costs for $\mathcal{H}$-matrix based techniques grow prohibitively large even for quite small problems, hence we do not construct $\mathcal{H}$-matrix based approximations for problems with $M \geq 39612$. For the trapping domain the $\mathcal{H}$-matrix based algorithm with the near-field reuse is twice faster than the conventional $\mathcal{H}$-matrix based approach already when dealing with problems with 94080 unknowns. However, both methods have prohibitively high storage requirements. For smaller problems, the FMM-based algorithm with the near-field reuse is slower than the algorithm with the near-field reuse that uses $\mathcal{H}$-matrices only, but is less memory-consuming. The use of the fast multipole method with the near-field reuse for larger discretizations allows to reduce storage costs about 3.5 times compared to $\mathcal{H}$-matrix based approaches.

The FMM-based algorithm with the near-field reuse is twice faster than the conventional FMM-based CQ algorithm for the problem with 46 million unknowns, while being only 1.5 times more efficient for smaller problems. Moreover, in this case the near-field reuse allows to reduce the system solution time after the matrices have been constructed. This can be explained as follows. With the choice of the parameter L as in this section also a part of the far-field is reused, and hence fewer multipole-to-local translations have to be done when computing the FMM accelerated matrix-vector product. Due to the use of the parameter d as in (36), the smallest distance between the admissible leaves of the 'far-field' block-cluster tree (i.e. $\mathcal{L}^+_{D_n,F}$ in (34)) is larger than the smallest distance between admissible leaves of the full block-cluster tree (since small close clusters are contained in the 'near-field' block-cluster tree). In this case the length of the corresponding multipole expansions for

ω	11/3		29/3		41/3		20		28	
h	0.075		0.028		0.02		0.014		0.01	
M	1344		9588		21900		39612		89202	
N	70		188		263		375		525	
L	26		36		41		41		43	
$\mathcal{H}$	1.6G	0.8 (0.02)h	78G	19.7 (0.4)h		-		-		-
$\mathcal{H}^{sp}$	1.2G	0.37 (0.03)h	56.6G	8.7 (0.7)h	230.2G	33.6 (4.2)h		-		-
$\mathcal{H}^2$	2.9G	1.1 (0.26)h	40.6G	21.8 (3.7)h	99.2G	59.7 (19)h	277G	161.8 (57.5)h	608G	745 (315)h
$\mathcal{H}^{2,sp}$	1.7G	0.7 (0.1)h	28.1G	15.1 (2.3)h	66.2G	41 (10.7)h	161.2G	105 (37)h	352G	373 (156)h

Table 3: Storage costs stated in GB and total CPU times in hours for different discretizations and techniques for the problem with the right-hand side defined by (43), time interval [0, 5.25]. In parentheses we show the CPU time needed to solve the system of equations after all the matrices were constructed. Here h is the time step, N is the number of time steps, M is size of the spatial discretization and L is as in (36).

leaves can be reduced, resulting in the improved FMM complexity (see also the forthcoming report [74]).

Remark 12. In a nutshell, the length of the multipole expansion for a cluster at the level ℓ of the cluster tree with a bounding box of diameter d can be determined by examining the convergence of the Gegenbauer's series. Namely, it is sufficient to find n s.t. $|j_n(isd)h_n(isc)| < \epsilon$, where c is the distance between the centers of the bounding boxes of two closest admissible clusters at the level ℓ (see e.g. [68] or [49] where a similar criterion is used). For small clusters the actual values produced by such criterion may be large ($n \gg |s|d$) due to the superexponential growth of spherical Hankel functions of a complex argument, c.f. [49]. However in our case c is increased, due to the reasons explained above.

This effect is enhanced by the fact that the length of the multipole expansion has to be chosen as $O\left(n^2\right)$.

This, combined with the fact that the computational complexity of the method for the near-field reuse with the diagonalization is quite low, see Section 5.2, results in improved computational times for the solution of the Toeplitz system of equations.

In Figures 10 and 11 the scattered field computed in the point (0, 0, 0) located inside the cavity is shown. These plots demonstrate that the wave is trapped inside the cavity for a long time.

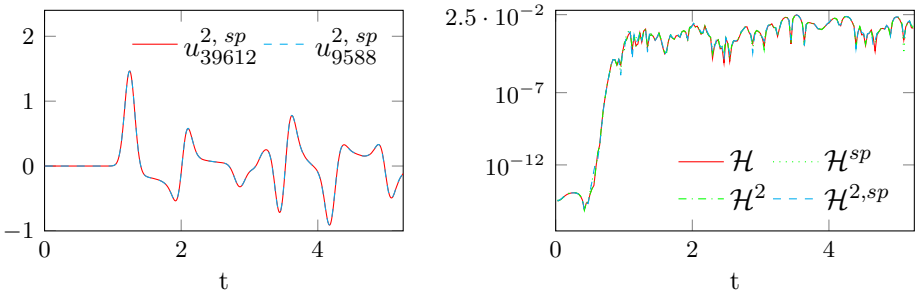

Figure 10: In the left plot we depict the scattered field computed for the incident wave with $\omega = \frac{29}{3}$ inside the cavity in the point $(0, 0, 0)$ (also computed at internal stages of the Runge-Kutta method), while in the right plot we show the errors of the solution obtained with different techniques measured at the same point (see also formula (42)). We can see that the errors coming from the near-field reuse are negligible compared to the discretization and matrix approximation errors.

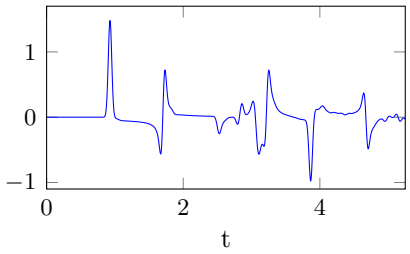

Figure 11: The scattered field for $\omega = 28$ at the point $(0,\ 0,\ 0)$ (computed also at internal stages of the Runge-Kutta method).

7. Conclusions

In this work we presented an algorithm of almost linear complexity for the solution of the scattering problem in three dimensions. The method uses the sparsity of Runge-Kutta convolution weights and allows to construct only a small number of matrices to approximate the near-field. For approximating the far-field data-sparse techniques, specifically, the multi-level high-frequency fast multipole method and $\mathcal{H}$-matrices, are employed.

Compared to the $\mathcal{H}$-matrix based convolution quadrature, the $\mathcal{H}$-matrix based algorithm with the near-field reuse allows to solve small scattering problems 1.5-2 times faster. For larger problems the high-frequency fast multipole based approach with the near-field reuse performs better, being 2-3 times faster and requiring 2-5 times less disk space. The performance of the algorithm was checked on problems having from 25000 to 92 million unknowns. In general, the gain from the use of the suggested approach depends on the problem size and on the geometry of a domain. Importantly, the suggested algorithm is easily parallellizable, though an optimally load-balanced approach would require the parallelization of an $\mathcal{H}$-matrix construction and HF FMM matrix-vector multiplication.

The near-field reuse approach relies only on the decay properties of convolution weights and hence can be extended to solve problems other than acoustic scattering, e.g. Maxwell equations, see [55], though the theoretical justification for these cases may be required. More work is needed for optimizing the construction of matrix approximations. Since the assembly of Galerkin matrices for various frequencies can be treated as a multiparametric problem, tensor decomposition methods can be used to improve it, see, e.g. [78]. The difficulty in the application of such techniques is the non-analyticity of high-frequency fast multipole operators in the frequency, which possibly can be overcome by the use of other fast multipole methods, e.g. [89]. The design of faster techniques for the matrix-vector multiplications involving Helmholtz potentials would significantly improve the presented approach.

More work should be done on the construction of the convolution quadrature based method that would take into account an a priori information about the solution and geometric properties of the domain, similarly to existing fast MOT methods.

References

[1] M. Costabel, Time-dependent problems with the boundary integral equation method, in: E. Stein, R. de Borst, T. J. Hughes. (Eds.), Encyclopedia of Computational Mechanics, John Wiley & Sons, Ltd., 2004.

[2] A. Bamberger, T. Ha-Duong, Formulation variationnelle espace-temps pour le calcul par potentiel retardé de la diffraction d'une onde acoustique, Mathematical Methods in the Applied Sciences 8 (1) (1986) 405–435.

[3] A. Bamberger, T. Ha-Duong, Formulation variationnelle pour le calcul de la diffraction d'une onde acoustique par une surface rigide, Mathematical Methods in the Applied Sciences 8 (1) (1986) 598–608.

[4] T. Ha-Duong, On retarded potential boundary integral equations and their discretization, in: P. Davies, D. Duncan, P. Martin, B. Rynne (Eds.), Topics in Computational Wave Propagation. Direct and Inverse Problems, Springer: Berlin, 2003, pp. 301–336.

[5] T. Ha-Duong, B. Ludwig, I. Terrasse, A Galerkin BEM for transient acoustic scattering by an absorbing obstacle, Internat. J. Numer. Methods Engrg. 57 (13) (2003) 1845–1882.

[6] A. Aimi, M. Diligenti, C. Guardasoni, I. Mazzieri, S. Panizzi, An energy approach to space-time Galerkin BEM for wave propagation problems, Internat. J. Numer. Methods Engrg. 80 (9) (2009) 1196–1240.

[7] A. Aimi, M. Diligenti, A. Frangi, C. Guardasoni, Neumann exterior wave propagation problems: computational aspects of 3D energetic Galerkin BEM, Computational Mechanics 51 (2013) 475–493.

[8] S. Sauter, A. Veit, A Galerkin method for retarded boundary integral equations with smooth and compactly supported temporal basis functions, Numer. Math. 123 (1) (2013) 145–176.

[9] P. J. Davies, D. B. Duncan, Stability and convergence of collocation schemes for retarded potential integral equations, SIAM J. Numer. Anal. 42 (3) (2004) 1167–1188.

[10] W. J. Mansur, A time-stepping technique to solve wave propagation problems using the boundary element method, Ph.D. thesis, University of Southampton (1983).

[11] T. Cruse, A direct formulation and numerical solution of the general transient elastodynamic problem ii., J. Math. Anal. Appl. 22 (1968) 341–355.

[12] T. A. Cruse, F. J. Rizzo, A direct formulation and numerical solution of the general transient elastodynamic problem i., J. Math. Anal. Appl. 22 (1968) 244–259.

[13] S. Sauter, A. Veit, Retarded boundary integral equations on the sphere: exact and numerical solution, IMA Journal of Numerical Analysis.

[14] B. Khoromskij, S. Sauter, A. Veit, Fast quadrature techniques for retarded potentials based on TT/QTT tensor approximation, Computational Methods in Applied Mathematics 11 (3) (2011) 342–362.

[15] Y. Shi, M.-Y. Xia, R.-S. Chen, E. Michielssen, M. Lu, Stable electric field TDIE solvers via quasi-exact evaluation of MOT matrix elements, IEEE Trans. Antennas and Propagation 59 (2) (2011) 574–585.

[16] A. A. Ergin, B.Shanker, E. Michielssen, Fast evaluation of three-dimensional transient wave fields using diagonal translation operators, Journal of Computational Physics 146 (1) (1998) 157 – 180.

[17] D. Jiao, A. A. Ergin, S. Balasubramaniam, E. Michielssen, J.-M. Jin, A fast higher-order time-domain finite element-boundary integral method for 3-D electromagnetic scattering analysis, IEEE Trans. Antennas and Propagation 50 (9) (2002) 1192–1202.

[18] A. Yilmaz, J.-M. Jin, E. Michielssen, Time domain adaptive integral method for surface integral equations, Antennas and Propagation, IEEE Transactions on 52 (10) (Oct.) 2692–2708.

[19] H. Bagci, A. Yilmaz, J.-M. Jin, E. Michielssen, Time Domain Adaptive Integral Method for Surface Integral Equations, Vol. 59 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2008.

[20] A. Boag, V. Lomakin, E. Michielssen, Nonuniform grid time domain (NGTD) algorithm for fast evaluation of transient wave fields, IEEE Trans. Antennas and Propagation 54 (7) (2006) 1943–1951.

[21] J. Meng, A. Boag, V. Lomakin, E. Michielssen, A multilevel Cartesian non-uniform grid time domain algorithm, J. Comput. Phys. 229 (22) (2010) 8430–8444.

[22] C. Lubich, Convolution quadrature and discretized operational calculus I, Numerische Mathematik 52 (2) (1988) 129–145.

[23] C. Lubich, Convolution quadrature and discretized operational calculus II, Numerische Mathematik 52 (4) (1988) 413–425.

[24] C. Lubich, On the multistep time discretization of linear initial-boundary value problems and their boundary integral equations, Numerische Mathematik 67 (1994) 365–389.

[25] M. Schanz, H. Antes, A new visco- and elastodynamic time domain: boundary element formulation, Comput. Mech. 20 (5) (1997) 452–459.

[26] M. Schanz, A boundary element formulation in time domain for viscoelastic solids, Comm. Numer. Methods Engrg. 15 (11) (1999) 799–809.

[27] L. Kielhorn, M. Schanz, Convolution quadrature method-based symmetric Galerkin boundary element method for 3-D elastodynamics, Internat. J. Numer. Methods Engrg. 76 (11) (2008) 1724–1746.

[28] S. Falletta, G. Monegato, L. Scuderi, A space-time BIE method for nonhomogeneous exterior wave equation problems. The Dirichlet case, IMA J. Numer. Anal. 32 (1) (2012) 202–226.

[29] L. Banjai, Multistep and multistage convolution quadrature for the wave equation: Algorithms and experiments, SIAM Journal on Scientific Computing 32 (5) (2010) 2964–2994.

[30] X. Wang, D. S. Weile, Implicit Runge-Kutta methods for the discretization of time domain integral equations, IEEE Trans. Antennas and Propagation 59 (12) (2011) 4651–4663.

[31] X. Wang, R. A. Wildman, D. S. Weile, P. Monk, A finite difference delay modeling approach to the discretization of the time domain integral equations of electromagnetics, IEEE Trans. Antennas and Propagation 56 (8, part 1) (2008) 2442–2452.

[32] L. Banjai, C. Lubich, An error analysis of Runge-Kutta convolution quadrature, BIT Numerical Mathematics 51 (2011) 483–496.

[33] L. Banjai, C. Lubich, J. Melenk, Runge-Kutta convolution quadrature for operators arising in wave propagation, Numer. Math. 119 (1) (2011) 1–20.

[34] L. Banjai, Time-domain dirichlet-to-neumann map and its discretization, IMA Journal of Numerical Analysis.

[35] Q. Chen, P. Monk, X. Wang, D. Weile, Analysis of convolution quadrature applied to the time-domain electric field integral equation, Commun. Comput. Phys. 11 (2) (2012) 383–399.

[36] D. J. Chappell, A convolution quadrature Galerkin boundary element method for the exterior Neumann problem of the wave equation, Math. Methods Appl. Sci. 32 (12) (2009) 1585–1608.

[37] D. J. Chappell, Convolution quadrature Galerkin boundary element method for the wave equation with reduced quadrature weight computation, IMA J. Numer. Anal. 31 (2) (2011) 640–666.

[38] J. Ballani, L. Banjai, S. Sauter, A. Veit, Numerical solution of exterior Maxwell problems by Galerkin BEM and Runge-Kutta convolution quadrature, Numer. Math. 123 (4) (2013) 643–670.

[39] W. Kress, S. Sauter, Numerical treatment of retarded boundary integral equations by sparse panel clustering, IMA J. Numer. Anal. 28 (1) (2008) 162–185.

[40] W. Hackbusch, W. Kress, S. A. Sauter, Sparse convolution quadrature for time domain boundary integral formulations of the wave equation by cutoff and panel-clustering 29 (2007) 113–134.

[41] W. Hackbusch, W. Kress, S. A. Sauter, Sparse convolution quadrature for time domain boundary integral formulations of the wave equation, IMA J. Numer. Anal. 29 (1) (2009) 158–179.

[42] M. Messner, Fast Boundary Element Methods in Acoustics, Verlag der Technischen Universitaet Graz, 2012.

[43] T. Saitoh, S. Hirose, Parallelized fast multipole BEM based on the convolution quadrature method for 3-D wave propagation problems in time-domain, IOP Conference Series: Materials Science and Engineering 10 (1).

[44] K. Yoshida, Applications of fast multipole method to boundary integral equation method, Ph.D. thesis, Kyoto University, Japan (2001).

[45] M. A. Epton, B. Dembart, Multipole translation theory for the three-dimensional Laplace and Helmholtz equations, SIAM J. Sci. Comput. 16 (4) (1995) 865–897.

[46] C. Lubich, A. Ostermann, Runge-Kutta methods for parabolic equations and convolution quadrature, Mathematics of Computation 60 (201) (1993) 105–131.

[47] L. Banjai, M. Kachanovska, Sparsity of Runge-Kutta convolution weights for three-dimensional wave equation.
URL `http://www.mis.mpg.de/publications/preprints/2012/prepr2012-72.html`

[48] V. Rokhlin, Diagonal forms of translation operators for the Helmholtz equation in three dimensions, Applied and Computational Harmonic Analysis 1 (1) (1993) 82 – 93.

[49] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, J. Zhao, A wideband fast multipole method for the Helmholtz equation in three dimensions, J. Comput. Phys. 216 (1) (2006) 300–325.

[50] F.-J. Sayas, Retarded potentials and time domain boundary integral equations: a road-map, the lecture notes for the workshop on the Theoretical and numerical aspects of inverse problems and scattering theory, La Coruna, Spain, July 4-8, 2011 (March 19 2013).

[51] W. C. Chew, J.-M. Jin, E. Michielssen, J. Song (Eds.), Fast and Efficient Algorithms in Computational Electromagnetics, Artech House, 2001.

[52] E. Hairer, C. Lubich, M. Schlichte, Fast numerical solution of nonlinear Volterra convolution equations, SIAM J. Sci. Statist. Comput. 6 (3) (1985) 532–541.

[53] L. Banjai, S. Sauter, Rapid solution of the wave equation in unbounded domains, SIAM J. Numerical Analysis 47 (1) (2008) 227–249.

[54] P. Henrici, Fast Fourier methods in computational complex analysis, SIAM Review 21 (4) (1979) 481–527.

[55] L. Banjai, M. Schanz, Wave propagation problems treated with convolution quadrature and BEM, Lecture Notes in Applied and Computational Mechanics 63 (2012) 145–184.

[56] L. Banjai, A. Laliena, F.-J. Sayas, Fully discrete Kirchhoff formulas with CQ-BEM, Preprint, http://arxiv.org/abs/1301.0267.

[57] W. Hackbusch, S. A. Sauter, On numerical cubatures of nearly singular surface integrals arising in BEM collocation, Computing 52 (2) (1994) 139–159.

[58] S. A. Sauter, C. Schwab, Boundary element methods, Vol. 39 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2011, translated and expanded from the 2004 German original.

[59] L. Grasedyck, W. Hackbusch, Construction and arithmetics of $\mathcal{H}$-matrices, Computing 70 (4) (2003) 295–334.

[60] W. Hackbusch, Hierarchische Matrizen: Algorithmen und Analysis, Springer-Verlag Berlin Heidelberg, 2009.

[61] R. Coifman, V. Rokhlin, S. Wandzura, The fast multipole method for the wave equation: a pedestrian prescription, Antennas and Propagation Magazine, IEEE 35 (3) (1993) 7 –12.

[62] S. Börm, L. Grasedyck, Low-rank approximation of integral operators by interpolation, Computing 72 (3-4) (2004) 325–332.

[63] M. Bebendorf, Approximation of boundary element matrices, Numer. Math. 86 (4) (2000) 565–589.

[64] S. Börm, L. Grasedyck, Hybrid cross approximation of integral operators, Numer. Math. 101 (2) (2005) 221–249.

[65] M. Bebendorf, Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems, Vol. 63 of Lecture Notes in Computational Science and Engineering (LNCSE), Springer-Verlag, 2008, iSBN 978-3-540-77146-3.

[66] J. Breuer, Schnelle randelementmethoden zur simulation von elektrischen wirbelstromfeldern sowie ihrer wärmeproduktion und kühlung, Ph.D. thesis, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart (2005).

[67] E. Darve, The fast multipole method I: error analysis and asymptotic complexity, SIAM Journal on Numerical Analysis 38 (1) (2000) 98–128.

[68] E. Darve, The fast multipole method: numerical implementation, Journal of Computational Physics 160 (1) (2000) 195 – 240.

[69] L. Banjai, W. Hackbusch, Hierarchical matrix techniques for low- and high-frequency Helmholtz problems, IMA Journal of Numerical Analysis 28 (1) (2008) 46–79.

[70] S. Amini, A. Profit, Multi-level fast multipole solution of the scattering problem, Engineering Analysis with Boundary Elements 27 (5) (2003) 547 – 564.

[71] S. Börm, W. Hackbusch, Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices, Computing 69 (1) (2002) 1–35.

[72] S. Börm, Efficient numerical methods for non-local operators, Vol. 14 of EMS Tracts in Mathematics, European Mathematical Society (EMS), Zürich, 2010.

[73] D. Brunner, M. Junge, P. Rapp, M. Bebendorf, L. Gaul, Comparison of the fast multipole method with hierarchical matrices for the Helmholtz-BEM, CMES: Computer Modeling in Engineering & Sciences 58 (2) (2010) 131–160.

[74] M. Kachanovska, $\mathcal{H}$-matrices and the high-frequency fast multipole method for the Helmholtz equation with decay, Tech. rep., Max Planck Institute for Mathematics in the Sciences, in preparation (2013).

[75] S. Erichsen, S. A. Sauter, Efficient automatic quadrature in 3-d Galerkin BEM, Comput. Methods Appl. Mech. Engrg. 157 (3-4) (1998) 215–224, seventh Conference on Numerical Methods and Computational Mechanics in Science and Engineering (NMCM 96) (Miskolc).

[76] S. A. Sauter, Cubature techniques for 3-D Galerkin BEM, in: Boundary elements: implementation and analysis of advanced algorithms (Kiel, 1996), Vol. 54 of Notes Numer. Fluid Mech., Vieweg, Braunschweig, 1996, pp. 29–44.

[77] S. A. Sauter, A. Krapp, On the effect of numerical integration in the Galerkin boundary element method, Numer. Math. 74 (3) (1996) 337–359.

[78] J. Ballani, Fast evaluation of near-field boundary integrals using tensor approximations, Dissertation, Universitt Leipzig (2012).

[79] J. Ballani, Fast evaluation of singular BEM integrals based on tensor approximations, Numerische Mathematik 121 (3) (2012) 433–460.

[80] P. Meszmer, Hierarchical quadrature for multidimensional singular integrals, J. Numer. Math. 18 (2) (2010) 91–117.

[81] P. Meszmer, J. Ballani, Tensor structured evaluation of singular volume integrals, Preprint MPI Leipzig.

[82] L. Grasedyck, Adaptive recompression of $\mathcal{H}$-matrices for BEM, Computing 74 (3) (2005) 205–223.

[83] D. Brunner, G. Of, M. Junge, O. Steinbach, L. Gaul, A fast BE-FE coupling scheme for partly immersed bodies, Internat. J. Numer. Methods Engrg. 81 (1) (2010) 28–47.

[84] M. Fischer, The fast multipole boundary element method and its application to structure-acoustic field interaction, Ph.D. thesis, University of Stuttgart (2004).

[85] A. Woo, H. Wang, M. Schuh, M. Sanders, Em programmer's notebook-benchmark radar targets for the validation of computational electromagnetics programs, Antennas and Propagation Magazine, IEEE 35 (1) (1993) 84–89.

[86] R. Kriemann, HLIBpro user manual. Technical Report 9/2008, MPI for Mathematics in the Sciences, Leipzig (2008).

[87] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.

[88] T. Betcke, S. N. Chandler-Wilde, I. G. Graham, S. Langdon, M. Lindner, Condition number estimates for combined potential integral operators in acoustics and their boundary element discretisation, Numer. Methods Partial Differential Equations 27 (1) (2011) 31–69.

[89] B. Engquist, L. Ying, Fast directional multilevel algorithms for oscillatory kernels, SIAM J. Sci. Comput. 29 (4) (2007) 1710–1737 (electronic).