VERLAG DER TECHNISCHEN UNIVERSITÄT GRAZ

Raphael Watschinger

Fast space-time boundary element methods for the heat equation

CES 44

MONOGRAPHIC SERIES TU GRAZ
COMPUTATION IN ENGINEERING AND SCIENCE

Raphael Watschinger

Fast space-time boundary element methods for the heat equation

Monographic Series TU Graz

Computation in Engineering and Science

Series Editors

Monographic Series TU Graz

Computation in Engineering and Science **Volume 44**

Raphael Watschinger

Fast space-time boundary element methods for the heat equation

This work is based on the dissertation *"Fast space-time boundary element methods for the heat equation"*, presented at Graz University of Technology, Institute of Applied Mathematics in 2022.

Supervision / Assessment:

Günther Of (Graz University of Technology)

Johannes Tausch (Southern Methodist University, Dallas)

Cover	Verlag der Technischen Universität Graz
Cover photo	Vier-Spezies-Rechenmaschine by courtesy of the Gottfried Wilhelm Leibniz Bibliothek Niedersächsische Landesbibliothek Hannover
Print	Buchschmiede (Dataform Media GmbH, Vienna)

2023 Verlag der Technischen Universität Graz
www.tugraz-verlag.at

Print
ISBN 978-3-85125-949-0

E-Book
ISBN 978-3-85125-950-6
DOI 10.3217/978-3-85125-949-0

Abstract

In this thesis we consider space-time boundary element methods for the solution of the heat equation. Since the system matrices in boundary element methods are generally dense, fast methods and related matrix compression algorithms are required to solve large systems in a reasonable amount of time. The main focus in this work lies on the extension of such a fast method, the parabolic fast multipole method (pFMM), to be able to exploit the frequently mentioned advantages of space-time methods: the possibility for a parallelization in space and time and space-time adaptivity.

In classical time stepping methods for the solution of time-dependent partial differential equations the occurring systems of linear equations are solved sequentially in time and, therefore, only a parallelization with respect to the spatial variables is possible. In space-time methods the whole space-time system can be solved at once, which enables an additional parallelization in time. Such a parallelization in space and time of a fast boundary element method for the solution of the heat equation is considered for the first time in this work. We provide an implementation of the parallel method for the execution on modern computer clusters and demonstrate a high parallel scalability in numerical experiments.

The other substantial modifications of the pFMM in this work improve the performance of the method when locally refined meshes are used for the discretization of the corresponding boundary integral equations. In particular, we introduce new FMM operations for tensor product meshes with adaptively chosen time step sizes and a new method for the compression of the temporal nearfield, which is especially relevant for meshes with adaptively refined spatial elements. In several numerical experiments we demonstrate the corresponding improvements.

In addition to the mentioned improvements of the parabolic multipole method, we consider an integration by parts formula for the hypersingular boundary integral operator of the heat equation in the theoretical part of this thesis. While such a formula is available in the literature, it contains a problematic integral term with a non-integrable kernel function. Furthermore, a proof of the formula is missing. We close these gaps by providing a rigorous proof of the integration by parts formula and an alternative, well-defined representation of the problematic integral term.

Zusammenfassung

In dieser Arbeit werden Raum-Zeit-Randelementmethoden zur Lösung der Wärmeleitgleichung betrachtet. Da die Matrizen der Gleichungssysteme bei Randelementmethoden im Allgemeinen vollbesetzt sind, werden schnelle Methoden, beziehungsweise spezielle Matrixkompressionsalgorithmen benötigt, um größere Gleichungssysteme in angemessener Zeit lösen zu können. Der Schwerpunkt dieser Arbeit liegt auf der Weiterentwicklung einer solchen schnellen Methode, der parabolischen Multipolmethode (pFMM), um die häufig genannten Vorteile von Raum-Zeit-Methoden ausnutzen zu können: Die gute Parallelisierbarkeit und lokale Adaptierbarkeit in Raum und Zeit.

Bei klassischen Zeitschrittverfahren zur Lösung zeitabhängiger partieller Differentialgleichung werden die entsprechenden Gleichungssysteme schrittweise in der Zeit gelöst, weshalb nur eine Parallelisierung bezüglich der örtlichen Freiheitsgrade möglich ist. Bei Raum-Zeit-Methoden kann hingegen das gesamte Raum-Zeit-Gleichungssystem auf einmal betrachtet und gelöst werden, was eine zusätzliche Parallelisierung in der Zeit ermöglicht. Eine solche Raum-Zeit-Parallelisierung für schnelle Randelementmethoden zur Lösung der Wärmeleitgleichung wird in dieser Arbeit erstmals betrachtet und für die Ausführung auf modernen Supercomputern implementiert. In verschiedenen numerischen Experimenten wird eine hohe parallele Skalierbarkeit der Methode gezeigt.

Die weiteren Anpassungen der pFMM, die in dieser Arbeit präsentiert werden, sorgen für eine höhere Effizienz der Methode, wenn lokal verfeinerte Netze zur Diskretisierung der entsprechenden Randintegralgleichungen verwendet werden. Konkret werden neue Multipoloperationen für Tensorproduktnetze mit adaptiv gewählten Zeitschrittweiten eingeführt, und eine neue Methode zur Kompression des zeitlichen Nahfelds, die insbesondere bei Netzen mit adaptiv verfeinerten Ortselementen von Bedeutung ist. In zahlreichen numerischen Experimenten werden die positiven Effekte dieser Anpassungen demonstriert.

Zusätzlich zu den angeführten Verbesserungen der parabolischen Multipolmethode, wird im theoretischen Teil dieser Arbeit eine partielle Integrationsformel für den hypersingulären Randintegraloperator der Wärmeleitgleichung betrachtet. In der Fachliteratur ist eine solche Formel zwar bekannt, allerdings enthält diese einen problematischen Integralterm mit einer nicht integrierbaren Kernfunktion und es fehlt ein Beweis für ihre Gültigkeit. Diese Lücken werden in dieser Arbeit geschlossen, indem ein rigoroser Beweis der partiellen Integrationsformel und eine alternative, wohldefinierte Darstellung des problematischen Integralterms präsentiert werden.

Contents

1 Introduction

The heat equation is a time-dependent partial differential equation (PDE) that models diffusion processes like the propagation of heat in a medium. The high relevance of such processes in natural and applied sciences and the fact that the heat equation is one of the simplest parabolic PDEs explain why it is widely studied in the mathematical literature. Often, initial boundary value problems are considered: For a given domain $\Omega \subset \mathbb{R}^3$, a finite time $T > 0$, a heat capacity constant $\alpha > 0$, and an initial datum u_0 one has to find a function u that satisfies the heat equation

$$\frac{\partial}{\partial t} u(\boldsymbol{x}, t) - \alpha \Delta u(\boldsymbol{x}, t) = 0 \qquad \text{for all } (\boldsymbol{x}, t) \text{ in } Q := \Omega \times (0, T), \tag{1.1}$$

the initial condition

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x} \text{ in } \Omega, \tag{1.2}$$

and additional boundary conditions, for example, a Dirichlet boundary condition

$$u(\boldsymbol{x}, t) = g(\boldsymbol{x}, t) \qquad \text{on } \Sigma := \partial\Omega \times (0, T), \tag{1.3}$$

where the values of u on the boundary $\partial\Omega$ of Ω are prescribed by a function g for all times $t \in (0, T)$, or a Neumann boundary condition

$$\boldsymbol{n}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}, t) = q(\boldsymbol{x}, t) \qquad \text{on } \Sigma, \tag{1.4}$$

where the normal derivative $\boldsymbol{n} \cdot \nabla u$ on the boundary $\partial\Omega$ is prescribed by a function q for all times $t \in (0, T)$. A variety of methods has been developed for the solution of such initial boundary value problems. In this thesis we focus on boundary element methods.

When it comes to the solution of boundary value problems of elliptic PDEs, boundary element methods (BEM) are well established; see e.g. the textbooks [63, 66]. If the fundamental solution of a PDE is known, the solutions of corresponding boundary value problems can be represented by boundary and volume potentials involving unknown density functions on the boundary of the considered domains which can be determined by solving corresponding boundary integral equations. Boundary element methods deal with the discretization and numerical solution of such boundary integral equations using a decomposition of the boundary into so-called boundary elements. Initial boundary value problems of the heat equation — and many other

time-dependent PDEs — allow for a similar treatment. In the corresponding space-time boundary integral equations, the unknown quantity is a density function defined on the lateral boundary $\Sigma = \partial\Omega \times (0,T)$ of the space-time cylinder $Q = \Omega \times (0,T)$. Such boundary integral equations are discretized in space-time boundary element methods by using a decomposition of Σ into space-time elements, e.g., elements of a tensor-product mesh obtained by combining a triangulation of $\partial\Omega$ with a partition of the time interval $(0,T)$. In the resulting systems of linear equations, a discretization of the unknown density function can be computed at once instead of sequentially in time as in time stepping methods. While this means that the linear systems can become quite large, it also brings certain benefits. Firstly, a parallelization of the solution process with respect to space and time is possible and not only with respect to space as in time stepping methods. This helps to improve the parallel scalability of the solution methods to exploit the full computing power of modern computer clusters. Secondly, the approach enables space-time adaptivity: The considered boundary meshes can be refined locally with respect to space and time to resolve certain local features or singularities of the desired solutions well while keeping the global number of boundary elements relatively low.

In this thesis we consider Galerkin space-time boundary element methods for the heat equation. Such Galerkin methods have been investigated in many publications, see e.g. [8, 24, 28, 31, 53, 57, 60]. In particular, a proper mathematical framework is well established and many important properties of the occurring boundary integral operators have already been proven. Two particularly important results which were first proven in [8, 24] are the ellipticity of the single layer operator and the hypersingular boundary integral operator in certain anisotropic Sobolev spaces. These ellipticity results can be used to show the unique solvability of related boundary integral equations and corresponding linear systems in BEM. A comprehensive summary of these and many other known results is given in [28].

A major difficulty when working with the hypersingular boundary integral operator is that it does not have an integral representation in a classical sense, since the related integral kernel has a strong singularity. In elliptic PDEs this problem is often overcome by considering the bilinear form associated with the hypersingular operator and using a special kind of integration by parts formula to obtain a representation that involves only weakly singular integrals; see e.g. [55] for the Laplace equation, [47, 56] for the Helmholtz equation, [56] for the time-harmonic Maxwell equations and [36, 41] for linear elastostatics. A similar integration by parts formula for the heat equation in two spatial dimensions was provided in [24] together with an outline of its proof. For the heat equation in three spatial dimensions as we consider it in this work, an integration by parts formula of the hypersingular operator is provided, for example, in [27, 51, 53], but without a proof. Moreover, the integration by parts formula presented in those works includes an integral over the time derivative of the heat kernel, i.e. the fundamental solution of the heat equation, which is not

Lebesgue integrable on the respective integration domain. To overcome these issues we provide a general integration by parts formula for the hypersingular operator of the heat equation in three spatial dimensions together with a rigorous proof in this thesis. Furthermore, we explain how to correctly represent the problematic integral term and how to evaluate it for a certain class of functions including those which are typically used for the discretization in BEM. This is one of the main theoretical results in this thesis and is published in [77].

The other main results in this thesis are achieved in the field of fast boundary element methods. Fast boundary element methods deal with the compression of BEM system matrices. A compression of these matrices is necessary since they are generally dense due to the non-local nature of the related boundary integral operators. In particular, the costs for assembling, storing, and applying such a matrix scale like the product of its numbers of rows and columns, which significantly limits the size of the problems that can be considered in a standard BEM. A variety of fast methods has been developed in the last decades to overcome this problem for BEM and related particle simulations. Some examples for the heat equation are algorithms based on Fourier series expansions and fast Fourier transforms [32, 33], algorithms based on the use of wavelet bases [16, 17, 60], a fast sparse grid method [37], and the parabolic fast multipole method (pFMM) [52, 53, 67, 68]. In the pFMM the matrix compression is achieved by using a clustering strategy to partition the matrix, efficiently approximating many of the resulting blocks by using a separable approximation of the heat kernel, and discarding others by exploiting the exponential decay of the heat kernel in space. In addition, the causality of the operators is used in the pFMM to solve systems of linear equations sequentially in time in a forward-sweeping manner.

In this thesis we consider the pFMM and present several substantial improvements. One of the main results is the parallelization of the method with respect to space and time for the execution on modern computer clusters with distributed memory architectures. In the literature one can find many publications that deal with the parallelization of fast methods for purely spatial problems, see e.g. [2, 4, 5, 25, 43, 79] for FMMs related to particle simulations, or e.g. [1, 15] for the solution of spatial boundary integral equations. In the context of the heat equation, a parallelization of a standard BEM with respect to space and time was described in [29], but a space-time parallelization of a fast method has not yet been considered. To enable such a parallelization, we reorganize the operations in the pFMM to obtain an FMM that realizes the full matrix application at once. This FMM is parallelized in shared and distributed memory by using a special task based execution scheme that exploits the temporal structure of the FMM for the distribution of the workload and the reduction of the number of communication events between the involved computing nodes. We present the corresponding algorithm in detail in this thesis and investigate its performance in several numerical experiments. The original results which we discuss are published in [76].

The other contributions in this thesis are related to improvements in the performance of the FMM for more general meshes. The pFMM has originally been described for tensor product meshes with uniform time steps. While an application for non-uniform time steps is possible, the performance of the pFMM can suffer significantly for meshes that are adaptive in time. To overcome this deficiency, we introduce new FMM operations in the spirit of adaptive fast multipole methods of purely spatial problems [21, 22, 54]. We present an analysis of the underlying temporally one-sided kernel expansions, a complete description of the resulting time-adaptive FMM, and numerical experiments to demonstrate the improvements. The corresponding results have been accepted for publication in [78].

The need for an additional temporal nearfield compression in the pFMM for meshes with fine spatial resolutions but rather coarse time steps was already discussed in [52]. In that work, a corresponding compression method was presented and several numerical examples were provided to demonstrate its effectiveness. Since the compression method in [52] is quite difficult to implement and heavily exploits the prismatic structure of the considered space-time boundary elements, which poses a restriction on the types of meshes for which it can be applied, we present an alternative method in this thesis. Our nearfield compression scheme makes use of the partially pivoted adaptive cross approximation (ACA), a matrix compression algorithm described, for example, in [10, 13]. We motivate why the ACA is applicable in our setting, show how to incorporate it in the FMM to compress the temporal nearfield, and present an additional recompression strategy with a novel recompression criterion to further improve the achieved compression. In several numerical experiments we investigate the effects of this nearfield compression scheme on the performance of the FMM.

The rest of this thesis is structured into seven main chapters and an additional concluding chapter. Each of the seven chapters starts with an introduction that includes an overview of the respective contents. Therefore, we only give a short summary of the overall structure of the thesis here. In Chapter 2 we repeat several basic results from the literature that are used in the rest of the thesis. Chapter 3 contains a short overview of boundary integral equations and boundary element methods for the heat equation and introduces the boundary integral operators and BEM matrices considered in the later chapters. Chapter 4 is devoted to the integration by parts formula for the hypersingular operator. In all the following chapters we focus on fast boundary element methods. In Chapter 5 we introduce a space-time FMM based on the pFMM which serves as a basis for all later considerations. In Chapter 6 we present our space-time parallelization approach for the FMM in shared and distributed memory. The extension of the FMM for temporally adaptive meshes is discussed in Chapter 7 and the novel nearfield compression scheme in Chapter 8. In Chapter 9 we conclude the thesis with a summary of our results and an outlook.

Contributions by coauthors

Several results in this thesis are based on joint work with the other members of the team in the Austrian-Czech research project BESTHEA, namely Günther Of, Michal Merta, and Jan Zapletal. The C++ code in [49] was written in equal shares by Michal Merta, Jan Zapletal, and me. Jan Zapletal is the main author of the research paper [81]. The quadrature formulas described in that paper are used in this thesis and are shortly mentioned in Chapter 3. Michal Merta and I are the main authors of the research paper [76] and contributed similarly to the development of the parallelization approach in that paper. The other new contributions in this thesis, in particular the results in Chapters 4, 7 and 8, were authored by me under the supervision of Günther Of.

2 Preliminaries

In this chapter we introduce basic definitions and notations and repeat results from the literature which are used throughout this work. We start with a short overview of basic functions spaces and Lipschitz domains in Sections 2.1 and 2.2. In Section 2.3 we recall standard anisotropic Sobolev spaces and related trace operators for the treatment of initial boundary value problems of the heat equation and related boundary integral equations. Standard discretization spaces and related meshes for these boundary integral equations are covered in Section 2.4.

2.1 Basic function spaces

Throughout this section, let $A \subset \mathbb{R}^d$ be open. The set of all continuous functions on A is denoted by $C(A)$ and the set of k times continuously differentiable functions by $C^k(A)$, where $k \in \mathbb{N} \cup \{\infty\}$. Here and in the rest of this work all functions are assumed to be real-valued, if not stated otherwise. A function f is contained in $C^k(\overline{A})$ if there exists an open set $B \supset \overline{A}$ and a function $\widetilde{f} \in C^k(B)$ such that $\widetilde{f}|_A = f$, i.e. f is the restriction of $\widetilde{f}$ to A. Functions in $C^\infty(A)$ with compact support are denoted by $C_c^\infty(A)$.

For any $p \in [1, \infty]$, $L^p(A)$ denotes the standard Lebesgue space containing equivalence classes of p-integrable functions on A with the usual norm. When referring to functions in these spaces, we always mean the related equivalence classes. For a given Banach space X the space $L^p(0,T;X)$ is the Bochner space containing equivalence classes of Bochner measurable functions $f : (0,T) \to X$ whose norm

$$\|f\|_{L^p(0,T;X)} = \left(\int_0^T \|f(t)\|_X^p \mathrm{d}t \right)^{1/p}$$

is bounded; see e.g. [26] for more details.

To simplify the notation we use boldface letters to denote spaces of functions with values in $\mathbb{R}^3$ whose components are in the respective function space. For example, $\boldsymbol{L}^2(A)$ is used instead of $(L^2(A))^3$.

2.2 Lipschitz domains and their smooth approximation

In the literature several different but equivalent definitions of a Lipschitz domain can be found. In this work we use the following definition similar to [19, Definition 2.1].

DEFINITION 2.1 (Lipschitz domain). *Let $d \in \mathbb{N}$ and $d > 1$. A set $Z_j \subset \mathbb{R}^d$ is called an* open coordinate cylinder *with radius $r_j > 0$ and height $h_j > 0$ if there exists a rectangular coordinate system of $\mathbb{R}^d$ obtained from the standard Cartesian coordinate system by rotation and translation with corresponding coordinates $\boldsymbol{x}^{(j)} \in \mathbb{R}^{d-1}$ and $s^{(j)} \in \mathbb{R}$ such that*

$$Z_j = \{(\boldsymbol{x}^{(j)}, s^{(j)}) \in \mathbb{R}^d : |\boldsymbol{x}^{(j)}| < r_j, \quad |s^{(j)}| < h_j\}.$$

A bounded, connected, open subset $\Omega \subset \mathbb{R}^d$ is called a Lipschitz domain, *if there exists a finite family $\{Z_j\}_j$ of open coordinate cylinders covering $\Gamma := \partial\Omega$ and for each coordinate cylinder Z_j there exists a Lipschitz continuous function $\eta_j : \mathbb{R}^{d-1} \to \mathbb{R}$, i.e. $|\eta_j(\boldsymbol{x}) - \eta_j(\boldsymbol{y})| \leq L_j |\boldsymbol{x} - \boldsymbol{y}|$ with $L_j \in \mathbb{R}$, such that $|\eta_j(\boldsymbol{x})| < h_j$ and*

$$\begin{aligned} \Omega \cap Z_j &= \{(\boldsymbol{x}^{(j)}, s^{(j)}) \in Z_j : s^{(j)} > \eta_j(\boldsymbol{x}^{(j)})\}, \\ \Gamma \cap Z_j &= \{(\boldsymbol{x}^{(j)}, \eta_j(\boldsymbol{x}^{(j)})) : \boldsymbol{x}^{(j)} \in \mathbb{R}^{d-1}\} \cap Z_j, \end{aligned} \tag{2.1}$$

where $(\boldsymbol{x}^{(j)}, s^{(j)})$ denote the coordinates associated with Z_j and h_j its height.

REMARK 2.2. *For each Lipschitz domain Ω one can find a family of coordinate cylinders $\{Z_j\}_j$ covering Γ as in Definition 2.1 such that for all j the dilated cylinder*

$$3Z_j := \{(\boldsymbol{x}^{(j)}, s^{(j)}) \in \mathbb{R}^d : |\boldsymbol{x}^{(j)}| < 3r_j, |s^{(j)}| < 3h_j\}$$

satisfies (2.1) too; see [19, Definition 2.1] and [73, Section 0.2]. Furthermore, one can choose the cylinders to be congruent, i.e. to have the same radii and heights.

We can define a C^∞ domain in the same way as we defined a Lipschitz domain, by requiring that all of the functions η_j in Definition 2.1 are in $C^\infty(\mathbb{R}^{d-1})$. The following theorem states that every Lipschitz domain Ω can be approximated from the inside by a sequence $\{\Omega_m\}_m$ of C^∞ domains. It is proven in [72, Theorem A.1].

THEOREM 2.3 ([19, Lemma 2.2], [72, Theorem A.1], [73, Theorem 1.12]). *Let $d > 1$ and $\Omega \subset \mathbb{R}^d$ be a Lipschitz domain whose boundary is denoted by Γ. Then there exist sequences of C^∞ domains $\{\Omega_m\}_m$, homeomorphisms $\{\boldsymbol{\Lambda}_m\}_m$ and functions $\{\omega_m\}_m$ that satisfy:*

(i) $\overline{\Omega}_m \subset \Omega$ and the homeomorphisms $\boldsymbol{\Lambda}_m : \Gamma \to \Gamma_m := \partial\Omega_m$ satisfy

$$\lim_{m\to\infty} (\sup\{|\boldsymbol{x} - \boldsymbol{\Lambda}_m(\boldsymbol{x})| : \boldsymbol{x} \in \Gamma\}) = 0. \tag{2.2}$$

In addition, $\boldsymbol{\Lambda}_m(\boldsymbol{x})$ approaches $\boldsymbol{x}$ non-tangentially.

(ii) There exists a finite family of coordinate cylinders $\{Z_j\}_j$ *that cover* Γ *as in Remark 2.2 and associated Lipschitz functions* $\{\eta_j\}_j$ *such that for each* j *and* m *there exists a function* $\eta_j^{(m)} \in C^\infty(\mathbb{R}^{d-1})$ *that represents* Γ_m *in* $3Z_j$*, i.e.*

$$\Gamma_m \cap 3Z_j = \{(\boldsymbol{x}^{(j)}, \eta_j^{(m)}(\boldsymbol{x}^{(j)})) : \boldsymbol{x}^{(j)} \in \mathbb{R}^{d-1}\} \cap 3Z_j.$$

Furthermore, $\eta_j^{(m)} \to \eta_j$ *uniformly and* $\nabla\eta_j^{(m)} \to \nabla\eta_j$ *pointwise almost everywhere as* m *tends to infinity, and* $\|\eta_j^{(m)}\|_{L^\infty(\mathbb{R}^{d-1})} \leq \|\eta_j\|_{L^\infty(\mathbb{R}^{d-1})}$ *for all* m*.*

(iii) The normal vectors $\boldsymbol{n}_m$ *on* Γ_m *converge pointwise almost everywhere to the normal vector* $\boldsymbol{n}$ *on* Γ*, in the sense that* $\boldsymbol{n}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x})) \to \boldsymbol{n}(\boldsymbol{x})$ *for almost all* $\boldsymbol{x} \in \Gamma$ *as* m *tends to infinity.*

(iv) The functions $\omega_m : \Gamma \to \mathbb{R}_{>0}$ *are such that*

$$\int_E \omega_m(\boldsymbol{x}) \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} = \int_{\boldsymbol{\Lambda}_m(E)} \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}$$

for all measurable sets $E \subset \Gamma$*, where* $\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}$ *and* $\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}$ *denote the surface measures on* Γ *and* Γ_m*, respectively. Furthermore, there exists a constant* $c > 0$ *such that* $c \leq \omega_m \leq c^{-1}$ *for all* m *and* $\omega_m \to 1$ *pointwise almost everywhere as* m *tends to infinity.*

REMARK 2.4. *Item* (i) *of Theorem 2.3 states that* $\boldsymbol{\Lambda}_m(\boldsymbol{x})$ *approaches* $\boldsymbol{x}$ *non-tangentially. Roughly speaking this means that the points* $\boldsymbol{\Lambda}_m(\boldsymbol{x})$ *all lie in a cone centered at* $\boldsymbol{x}$*. A rigorous definition is given in [19, 73]. Some additional properties of the approximating domains* $\{\Omega_m\}_m$ *can be found in the referenced works.*

2.3 Anisotropic Sobolev spaces and trace operators

From this point onwards, let $\Omega \subset \mathbb{R}^3$ be a bounded Lipschitz domain as in Definition 2.1 with boundary Γ, let $T > 0$ be finite and $Q := \Omega \times (0, T)$ be the space-time cylinder with lateral boundary $\Sigma := \Gamma \times (0, T)$. In this work we consider initial boundary value problems for the heat equation, i.e. we want to find a function u that satisfies the heat equation (1.1), an initial condition (1.2), and a Dirichlet boundary condition (1.3) or a Neumann boundary condition (1.4). In the following, suitable anisotropic Sobolev spaces and trace operators are introduced, which allow to treat these initial boundary problems in a well-posed manner. Note that throughout this work differential operators like $\Delta = \sum_{j=1}^3 \partial_{x_j}$ and $\nabla = (\partial_{x_1}, \partial_{x_2}, \partial_{x_3})^\top$ act only with respect to the spatial variables $\{x_j\}_{j=1}^3$, not the temporal variable t.

To start off, we briefly recall the definitions of relevant Sobolev spaces on the time interval $(0,T)$, the spatial domain Ω and its boundary Γ. For a better overview see [27, Section 2] and [80, Sections 2.2–2.5]. On $(0,T)$ one defines the Sobolev space $H^1(0,T)$ by

$$H^1(0,T) = \{v \in L^2(0,T) : v' \in L^2(0,T)\}, \quad \|v\|^2_{H^1(0,T)} := \|v\|^2_{L^2(0,T)} + \|v'\|^2_{L^2(0,T)},$$

where v' denotes the weak derivative of v. The fractional order Sobolev spaces $H^s(0,T)$ for a given order s with $0 < s < 1$ can be defined by

$$H^s(0,T) := \Big\{v \in L^2(0,T) : |v|_{H^s(0,T)} < \infty\Big\},$$

where the seminorm and norm on $H^s(0,T)$ are given by

$$|v|_{H^s(0,T)} := \left(\int_0^T \int_0^T \frac{|v(t)-v(\tau)|^2}{|t-\tau|^{1+2s}}\,\mathrm{d}\tau\,\mathrm{d}t\right)^{1/2},$$
$$\|v\|_{H^s(0,T)} := \Big(\|v\|^2_{L^2(0,T)} + |v|^2_{H^s(0,T)}\Big)^{1/2}.$$

We are particularly interested in the case $s = 1/2$ and the subspaces

$$H^{1/2}_{0,}(0,T) := \Big\{v \in H^{1/2}(0,T) : |v|_{H^{1/2}_{0,}(0,T)} < \infty\Big\},$$
$$H^{1/2}_{,0}(0,T) := \Big\{v \in H^{1/2}(0,T) : |v|_{H^{1/2}_{,0}(0,T)} < \infty\Big\},$$

with

$$|v|_{H^{1/2}_{0,}(0,T)} := \left(\int_0^T \frac{v(t)^2}{t}\,\mathrm{d}t\right)^{1/2}, \quad \|v\|_{H^{1/2}_{0,}(0,T)} := \Big(\|v\|^2_{H^{1/2}(0,T)} + |v|^2_{H^{1/2}_{0,}(0,T)}\Big)^{1/2},$$
$$|v|_{H^{1/2}_{,0}(0,T)} := \left(\int_0^T \frac{v(t)^2}{T-t}\,\mathrm{d}t\right)^{1/2}, \quad \|v\|_{H^{1/2}_{,0}(0,T)} := \Big(\|v\|^2_{H^{1/2}(0,T)} + |v|^2_{H^{1/2}_{,0}(0,T)}\Big)^{1/2}.$$

$H^{1/2}_{0,}(0,T)$ can be understood as the subspace of $H^{1/2}(0,T)$ whose functions satisfy a homogeneous initial condition in a weak sense. Indeed, the extension by zero of a function in $H^{1/2}_{0,}(0,T)$ lies in $H^{1/2}(-\infty,T)$; see Proposition 5.2 (ii) in [44, Chapter 3]. Furthermore, it is shown in [44, Chapter 1, Remark 11.5] that $H^{1/2}_{0,}(0,T)$ coincides with the interpolation space $[L^2(0,T), H^1_{0,}(0,T)]_{1/2}$, where

$$H^1_{0,}(0,T) := \{v \in H^1(0,T) : v(0) = 0\}.$$

Similarly, $H^{1/2}_{,0}(0,T)$ corresponds to the subspace of $H^{1/2}(0,T)$ with a weak homogeneous end condition. Note that, in general, an element in $H^{1/2}(0,T)$ does not have a

continuous representative, so one cannot simply enforce homogeneous initial or end conditions in a strong sense.

When it comes to the spatial domain, we consider the classical Sobolev space $H^1(\Omega)$ with the norm

$$\|u\|_{H^1(\Omega)} := \left(\int_\Omega u(\boldsymbol{x})^2 + |\nabla u(\boldsymbol{x})|^2 \,\mathrm{d}\boldsymbol{x} \right)^{1/2}$$

and its subspace

$$H_0^1(\Omega) = \overline{C_c^\infty(\Omega)},$$

where the closure is taken with respect to the norm $\|\cdot\|_{H^1(\Omega)}$. On the surface Γ of Ω we consider the Sobolev space $H^{1/2}(\Gamma)$ with the Sobolev Slobodeckij norm

$$\|\varphi\|_{H^{1/2}(\Gamma)} := \left(\|\varphi\|_{L^2(\Gamma)}^2 + \int_\Gamma \int_\Gamma \frac{|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})|^2}{|\boldsymbol{x} - \boldsymbol{y}|^3} \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \right)^{1/2}$$

and its dual space $H^{-1/2}(\Gamma)$.

In the space-time domain Q we are particularly interested in the anisotropic Sobolev space

$$H^{1,1/2}(Q) := L^2(0,T;H^1(\Omega)) \cap H^{1/2}(0,T;L^2(\Omega))$$

with the norm

$$\|u\|_{H^{1,1/2}(Q)} := \left(\int_0^T \|u(\cdot,t)\|_{H^1(\Omega)}^2 \,\mathrm{d}t + \int_\Omega |u(\boldsymbol{x},\cdot)|_{H^{1/2}(0,T)}^2 \,\mathrm{d}\boldsymbol{x} \right)^{1/2}$$

and its subspaces

$$H_{;0,}^{1,1/2}(Q) := L^2(0,T;H^1(\Omega)) \cap H_{0,}^{1/2}(0,T;L^2(\Omega)),$$
$$H_{;,0}^{1,1/2}(Q) := L^2(0,T;H^1(\Omega)) \cap H_{,0}^{1/2}(0,T;L^2(\Omega)),$$

where

$$\|u\|_{H_{;0,}^{1,1/2}(Q)} := \left(\|u\|_{H^{1,1/2}(Q)}^2 + \int_\Omega |u(\boldsymbol{x},\cdot)|_{H_{0,}^{1/2}(0,T)}^2 \,\mathrm{d}\boldsymbol{x} \right)^{1/2}$$

and $\|\cdot\|_{H_{;,0}^{1,1/2}(Q)}$ is defined analogously. The spaces $H_{;0,}^{1,1/2}(Q)$ and $H_{;,0}^{1,1/2}(Q)$ can be understood as the subspaces of $H^{1,1/2}(Q)$ with weak homogeneous initial and end conditions, respectively. The notation here corresponds to the one in [28]. In [24], $\widetilde{H}^{1,1/2}(Q)$ and $\overset{(T)}{H}{}^{1,1/2}(Q)$ denote the spaces equivalent to $H_{;0,}^{1,1/2}(Q)$ and $H_{;,0}^{1,1/2}(Q)$, respectively. Another subspace of $H^{1,1/2}(Q)$ which is of interest in this work is the space

$$H_{0;}^{1,1/2}(Q) = L^2(0,T;H_0^1(\Omega)) \cap H^{1/2}(0,T;L^2(\Omega)).$$

There hold the following density results.

PROPOSITION 2.5. *Let the spaces* $C_c^\infty(\overline{\Omega}\times(0,T])$, $C_c^\infty(\overline{\Omega}\times[0,T))$ *and* $C_c^\infty(\overline{\Omega}\times(0,T))$ *be defined by*

$$C_c^\infty(\overline{\Omega}\times(0,T]) := \{u : u = \widetilde{u}|_Q, \widetilde{u} \in C_c^\infty(\mathbb{R}^3\times(0,\infty)\}, \tag{2.3}$$

$$C_c^\infty(\overline{\Omega}\times[0,T)) := \{u : u = \widetilde{u}|_Q, \widetilde{u} \in C_c^\infty(\mathbb{R}^3\times(-\infty,T)\}, \tag{2.4}$$

$$C_c^\infty(\overline{\Omega}\times(0,T)) := \{u : u = \widetilde{u}|_Q, \widetilde{u} \in C_c^\infty(\mathbb{R}^3\times(0,T))\}. \tag{2.5}$$

The following inclusions are dense:

$$C_c^\infty(\overline{\Omega}\times(0,T)) \subset X \text{ where } X \in \{H^{1,1/2}(Q), H^{1,1/2}_{;0,}(Q), H^{1,1/2}_{;,0}(Q)\}, \tag{2.6a}$$

$$C_c^\infty(\overline{\Omega}\times(0,T]) \subset H^{1,1/2}_{;0,}(Q), \tag{2.6b}$$

$$C_c^\infty(\overline{\Omega}\times[0,T)) \subset H^{1,1/2}_{;,0}(Q), \tag{2.6c}$$

$$C_c^\infty(Q) \subset H^{1,1/2}_{0;}(Q). \tag{2.6d}$$

Sketch of the proof. In the proof of Lemma 2.22 in [24] it is mentioned that the inclusion (2.6b) is dense, and that this follows from the density of $C^\infty(\overline{\Omega})$ in $H^1(\Omega)$ and the density of $C_c^\infty((0,T]) := \{f : f = \widetilde{f}|_{(0,T)}, \widetilde{f} \in C_c^\infty(0,\infty)\}$ in $H^{1/2}_{0,}(0,T)$ by tensor product arguments. The same tensor product arguments can be used to show the remaining density results. For this purpose we use that $C_c^\infty(0,T)$ is dense in $H^{1/2}(0,T)$ — see e.g. [34, Theorem 1.4.2.4] — and also in $H^{1/2}_{0,}(0,T)$ and $H^{1/2}_{,0}(0,T)$ — see [80, Theorem 2.2.2]. Furthermore, $C_c^\infty([0,T)) := \{f : f = \widetilde{f}|_{(0,T)}, \widetilde{f} \in C_c^\infty(-\infty,T)\}$ is dense in $H^{1/2}_{,0}(0,T)$ and $C_c^\infty(\Omega)$ in $H^1_0(\Omega)$. □

On the lateral space-time boundary $\Sigma = \Gamma\times(0,T)$ we focus on the anisotropic Sobolev space

$$H^{1/2,1/4}(\Sigma) := L^2(0,T;H^{1/2}(\Gamma)) \cap H^{1/4}(0,T;L^2(\Gamma))$$

with the norm

$$\|\varphi\|_{H^{1/2,1/4}(\Sigma)} := \left(\int_0^T \|\varphi(\cdot,t)\|^2_{H^{1/2}(\Gamma)}\,\mathrm{d}t + \int_\Gamma |\varphi(\boldsymbol{x},\cdot)|^2_{H^{1/4}(0,T)}\,\mathrm{d}\boldsymbol{x}\right)^{1/2}$$

and its dual

$$H^{-1/2,-1/4}(\Sigma) = (H^{1/2,1/4}(\Sigma))'.$$

The duality product on $H^{-1/2,-1/4}(\Sigma)\times H^{1/2,1/4}(\Sigma)$ is denoted by $\langle\cdot,\cdot\rangle_\Sigma$ and understood as the continuous extension of the L^2 inner product

$$\langle\psi,\varphi\rangle_{L^2(\Sigma)} = \int_0^T\int_\Gamma \psi(\boldsymbol{x},t)\varphi(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t$$

from $L^2(\Sigma)\times H^{1/2,1/4}(\Sigma)$ to $H^{-1/2,-1/4}(\Sigma)\times H^{1/2,1/4}(\Sigma)$.

The following theorem establishes trace operators and corresponding right inverse operators between $H^{1,1/2}(Q)$ and its subspaces and $H^{1/2,1/4}(\Sigma)$.

THEOREM 2.6 ([45, Chapter 4, Theorem 2.1], [24, Lemma 2.4]). *There exists a unique continuous operator* $\gamma_{0,\Sigma}^{\mathrm{int}} : H_{;0,}^{1,1/2}(Q) \to H^{1/2,1/4}(\Sigma)$ *such that* $\gamma_{0,\Sigma}^{\mathrm{int}} u = u|_\Sigma$ *holds for all* $u \in C_c^\infty(\overline{\Omega} \times (0,T))$. *This operator is surjective.*

REMARK 2.7. *The trace operator* $\gamma_{0,\Sigma}^{\mathrm{int}}$ *can also be considered as a surjective operator from* $H^{1,1/2}(Q)$ *or* $H_{;0}^{1,1/2}(Q)$ *to* $H^{1/2,1/4}(\Sigma)$. *In a slight abuse of notation, we denote all three operators by* $\gamma_{0,\Sigma}^{\mathrm{int}}$ *in this work. Note that the target space is the same for* $H_{;0,}^{1,1/2}(Q)$, $H_{;0}^{1,1/2}(Q)$ *and* $H^{1,1/2}(Q)$, *i.e. it does not depend on the homogeneous initial or end condition. In fact, initial or end conditions cannot be observed in* $H^{1/4}(0,T)$. *This is related to the fact that the interpolation spaces* $H_{0,}^s(0,T) := [L^2(0,T), H_{0,}^1(0,T)]_s$ *coincide with* $H^s(0,T)$ *for all* s *with* $0 < s < 1/2$; *see [42, Remark 2.5 (1)].*

COROLLARY 2.8. *There exist continuous operators*

$$\begin{aligned} E_\Sigma &: H^{1/2,1/4}(\Sigma) \to H^{1,1/2}(Q), \\ E_{\Sigma;0,} &: H^{1/2,1/4}(\Sigma) \to H_{;0,}^{1,1/2}(Q), \\ E_{\Sigma;,0} &: H^{1/2,1/4}(\Sigma) \to H_{;,0}^{1,1/2}(Q), \end{aligned}$$

which are right inverses of the operator $\gamma_{0,\Sigma}^{\mathrm{int}}$ *on the respective space on* Q.

For the definition of the Neumann trace operator we introduce the space

$$H_{;0,}^{1,1/2}(Q, \partial_t - \alpha\Delta) = \left\{ u \in H_{;0,}^{1,1/2}(Q) : \frac{\partial}{\partial t} u - \alpha\Delta u \in L^2(Q) \right\}$$

with the norm

$$\|u\|_{H_{;0,}^{1,1/2}(Q,\partial_t-\alpha\Delta)} := \left(\|u\|^2_{H_{;0,}^{1,1/2}(Q)} + \left\| \frac{\partial}{\partial t} u - \alpha\Delta \right\|^2_{L^2(Q)} \right)^{1/2}$$

as in [24, Equation (2.29)] and [28, page 5]. For a function $u \in C^2(\overline{Q})$ the Neumann trace is given by $\boldsymbol{n} \cdot \nabla u$ and due to Green's first identity it satisfies

$$\begin{aligned} &\alpha \int_0^T \int_\Gamma (\boldsymbol{n} \cdot \nabla u)\, v \,\mathrm{d}s_{\boldsymbol{x}} \,\mathrm{d}t \\ &\quad = -\int_0^T \int_\Omega \left(\frac{\partial}{\partial t} u - \alpha\Delta u \right) v \,\mathrm{d}\boldsymbol{x} \,\mathrm{d}t + \alpha \int_0^T \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x} \,\mathrm{d}t + \int_0^T \int_\Omega \frac{\partial}{\partial t} uv \,\mathrm{d}\boldsymbol{x} \,\mathrm{d}t \end{aligned}$$

for all $v \in C^1(\overline{Q})$. This identity is used to define the Neumann trace operator for arbitrary $u \in H_{;0,}^{1,1/2}(Q, \partial_t - \alpha\Delta)$ in Proposition 2.10. For this purpose, we need to show that the right-hand side is well-defined and continuous for $u \in H_{;0,}^{1,1/2}(Q, \partial_t - \alpha\Delta)$ and $v \in H_{;,0}^{1,1/2}(Q)$. This is clear for the first two integrals and for the third one it is established in the following proposition.

PROPOSITION 2.9 ([24, cf. Lemma 2.6]). *The bilinear form*

$$d(u,v) := \int_0^T \int_\Omega \frac{\partial}{\partial t} u(\boldsymbol{x},t) v(\boldsymbol{x},t) \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \tag{2.7}$$

can be continuously extended from the product space $C_c^\infty(\overline{\Omega}\times(0,T])\times C_c^\infty(\overline{\Omega}\times[0,T))$ *to* $H^{1,1/2}_{;0,}(Q)\times H^{1,1/2}_{;,0}(Q)$ *and there holds*

$$|d(u,v)| \le c_d \|u\|_{H^{1,1/2}_{;0,}(Q)} \|v\|_{H^{1,1/2}_{;0,}(Q)}, \tag{2.8}$$

where the constant c_d *does not depend on* Ω.

Sketch of the proof. It suffices to show the estimate in (2.8) for $u \in C_c^\infty(\overline{\Omega}\times(0,T])$ and $v \in C_c^\infty(\overline{\Omega}\times[0,T))$, since these spaces are dense in $H^{1,1/2}_{;0,}(Q)$ and $H^{1,1/2}_{;,0}(Q)$, respectively; see Proposition 2.5.

We can extend $u \in C_c^\infty(\overline{\Omega}\times(0,T])$ to a function $\widetilde{u} \in H^{1/2}(\mathbb{R};L^2(\Omega))$ by setting $\widetilde{u}(t,\cdot)=0$ for all $t<0$ and then setting $\widetilde{u}(\cdot,t)=\widetilde{u}(\cdot,2T-t)$ for all $t>T$. For this extension one can show that

$$\|\widetilde{u}\|_{H^{1/2}(\mathbb{R};L^2(\Omega))} \le c(T)\|u\|_{H^{1,1/2}_{;0,}(Q)},$$

where the constant $c(T)$ depends only on T. Similarly, we can extend a function $v \in C_c^\infty(\overline{\Omega}\times[0,T))$ to $\widetilde{v} \in H^{1/2}(\mathbb{R};L^2(\Omega))$ such that

$$\|\widetilde{v}\|_{H^{1/2}(\mathbb{R};L^2(\Omega))} \le c(T)\|v\|_{H^{1,1/2}_{;,0}(Q)}.$$

Since u and v are compactly supported in $\overline{\Omega}\times(0,T]$ and $\overline{\Omega}\times[0,T)$, respectively, there holds

$$\begin{aligned} d(u,v) &= \int_0^T \int_\Omega \frac{\partial}{\partial t} u(\boldsymbol{x},t) v(\boldsymbol{x},t) \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t = \int_{\mathbb{R}} \int_\Omega \frac{\partial}{\partial t} \widetilde{u}(\boldsymbol{x},t) \widetilde{v}(\boldsymbol{x},t) \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\ &\le c\|\widetilde{u}\|_{H^{1/2}(\mathbb{R};L^2(\Omega))} \|\widetilde{v}\|_{H^{1/2}(\mathbb{R};L^2(\Omega))} \le c_d(T)\|u\|_{H^{1,1/2}_{;0,}(Q)} \|v\|_{H^{1,1/2}_{;,0}(Q)}. \end{aligned} \tag{2.9}$$

The first of these estimates can be shown by switching to the Fourier domain in time using Plancherel's theorem, where the estimate follows easily when considering the equivalent norm in $H^{1/2}(\mathbb{R};L^2(\Omega))$ defined via Fourier transforms. The second estimate is a consequence of the two estimates above. □

PROPOSITION 2.10 ([24, cf. Lemma 2.16]). *The map*

$$\gamma_{1,\Sigma}^{\mathrm{int}} : \; H_{;0,}^{1,1/2}(Q,\partial_t-\alpha\Delta) \to H^{-1/2,-1/4}(\Sigma)$$

defined by

$$\begin{aligned}&\langle\gamma_{1,\Sigma}^{\mathrm{int}}u,\psi\rangle_\Sigma \qquad (2.10)\\ &:= \frac{1}{\alpha}\Big(-\int_0^T\int_\Omega \Big(\Big(\frac{\partial}{\partial t}-\alpha\Delta\Big)u\; E_{\Sigma;,0}\psi - \alpha\nabla u\cdot\nabla(E_{\Sigma;,0}\psi)\Big)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t + d(u,E_{\Sigma;,0}\psi)\Big)\end{aligned}$$

for all $u \in H_{;0,}^{1,1/2}(Q,\partial_t-\alpha\Delta)$ *and* $\psi \in H^{1/2,1/4}(\Sigma)$ *is well-defined and continuous. In particular, it does not depend on the choice of the extension* $E_{\Sigma;,0}$ *from* $H^{1/2,1/4}(\Sigma)$ *to* $H_{;,0}^{1,1/2}(Q)$. *Furthermore, for* $u \in C^2(\overline{Q})$ *there holds* $\gamma_{1,\Sigma}^{\mathrm{int}}u = \boldsymbol{n}\cdot\nabla u|_\Sigma$.

REMARK 2.11. *By considering a suitable exterior domain* $\Omega^{\mathrm{ext}} \subset \mathbb{R}^3\setminus\overline{\Omega}$ *and the related space-time cylinder* $Q^{\mathrm{ext}} = \Omega^{\mathrm{ext}}\times(0,T)$ *we can define exterior trace operators* $\gamma_{0,\Sigma}^{\mathrm{ext}} : H^{1,1/2}(Q^{\mathrm{ext}}) \to H^{1/2,1/4}(\Sigma)$ *and* $\gamma_{1,\Sigma}^{\mathrm{ext}} : H^{1,1/2}(Q^{\mathrm{ext}},\partial_t-\alpha\Delta) \to H^{-1/2,-1/4}(\Sigma)$; *see e.g. [24, p. 515f.]. These exterior trace operators will be used for the definition of the double layer and adjoint time-reversed double layer operator in Section 3.1.*

2.4 Piecewise polynomial approximation spaces

In this section we introduce finite-dimensional spaces for the discretization of the anisotropic Sobolev spaces $H^{1/2,1/4}(\Sigma)$ and $H^{-1/2,-1/4}(\Sigma)$ in Section 2.3 and the space $L^2(\Omega)$. We assume that Ω is a polyhedral Lipschitz domain.

On $\Sigma = \Gamma\times(0,T)$ we consider piecewise polynomial tensor product spaces defined on corresponding tensor product meshes, which is the standard discretization strategy; see for example [57, Section 6] or [24, Section 5]. A tensor product mesh of Σ is constructed by combining the elements of a spatial surface mesh Γ_h with the time intervals of a partition $\mathcal{I}_{h_t}$ of $(0,T)$. We assume that $\Gamma_h = \{\gamma_{j_{\boldsymbol{x}}}\}_{j_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}}$ is a triangular surface mesh with plane triangles $\gamma_{j_{\boldsymbol{x}}}$ and require that Γ_h is admissible, i.e. the intersection $\overline{\gamma_{j_{\boldsymbol{x}}}}\cap\overline{\gamma_{k_{\boldsymbol{x}}}}$ of two triangles with $j_{\boldsymbol{x}} \neq k_{\boldsymbol{x}}$ is either empty, or a shared vertex, or a shared edge. The local mesh size $h_{\boldsymbol{x},j_{\boldsymbol{x}}}$ of a triangle $\gamma_{j_{\boldsymbol{x}}} \in \Gamma_h$ is defined by

$$h_{\boldsymbol{x},j_{\boldsymbol{x}}} := |\gamma_{j_{\boldsymbol{x}}}|^{(1/2)} = \Big(\int_{\gamma_{j_{\boldsymbol{x}}}} 1\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\Big)^{1/2} \qquad (2.11)$$

and the global mesh size $h_{\boldsymbol{x}}$ of Γ_h by

$$h_{\boldsymbol{x}} := \max_{j_{\boldsymbol{x}}=1,\dots,E_{\boldsymbol{x}}} h_{\boldsymbol{x},j_{\boldsymbol{x}}}. \qquad (2.12)$$

A partition $\mathcal{I}_{h_t}$ of the time interval $(0,T)$ is formed by a sequence of grid points $\{t_{j_t}\}_{j_t=0}^{E_t}$ satisfying $0 = t_0 < t_1 < \ldots < t_{E_t} = T$. The size h_{t,j_t} of the corresponding subintervals (t_{j_t-1}, t_{j_t}) of $(0,T)$ is given by

$$h_{t,j_t} = t_{j_t} - t_{j_t-1} \tag{2.13}$$

and the global mesh size h_t of $\mathcal{I}_{h_t}$ by

$$h_t = \max_{j_t=1,\ldots,E_t} h_{t,j_t}. \tag{2.14}$$

Note that in this work we will not only consider uniform partitions of $(0,T)$ where $h_{t,j_t} = h_t$ for all $j_t \in \{1,\ldots,E_t\}$, but also non-uniform partitions where the sizes h_{t,j_t} may differ. By combining a temporal partition $\mathcal{I}_{h_t}$ with a surface mesh Γ_h we obtain the tensor product mesh

$$\begin{aligned}\Sigma_h &:= \Gamma_h \otimes \mathcal{I}_{h_t} \\ &= \{\sigma_{j_t,j_{\boldsymbol{x}}} = \gamma_{j_{\boldsymbol{x}}} \times (t_{j_t-1}, t_{j_t}) : j_t \in \{1,\ldots,E_t\}, j_{\boldsymbol{x}} \in \{1,\ldots,E_{\boldsymbol{x}}\}\}.\end{aligned} \tag{2.15}$$

We define the space of all piecewise constant functions on the partition $\mathcal{I}_{h_t}$ by $S^0_{h_t}(\mathcal{I}_{h_t})$. A basis of $S^0_{h_t}(\mathcal{I}_{h_t})$ is given by $\{\varphi^0_{t,j_t}\}_{j_t=1}^{E_t}$, where $\varphi^0_{t,j_t}(t) = 1$ for $t \in (t_{j_t-1}, t_{j_t})$ and zero otherwise. Likewise, we define the space of all piecewise constant functions on the mesh Γ_h by $S^0_{h_{\boldsymbol{x}}}(\Gamma_h)$ with the basis $\{\varphi^0_{\boldsymbol{x},j_{\boldsymbol{x}}}\}_{j_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}}$, where $\varphi^0_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \in \gamma_{j_{\boldsymbol{x}}}$ and zero otherwise. This allows us to define the tensor product space $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ by

$$\begin{aligned}S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h) &:= S^0_{h_{\boldsymbol{x}}}(\Gamma_h) \otimes S^0_{h_t}(\mathcal{I}_{h_t}) \\ &= \operatorname{span}\{\varphi^0_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t} : j_t \in \{1,\ldots,E_t\}, j_{\boldsymbol{x}} \in \{1,\ldots E_{\boldsymbol{x}}\}\}.\end{aligned} \tag{2.16}$$

A function $f_h \in S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ can be written as a linear combination of the basis functions $\{\varphi^0_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}\}_{j_t,j_{\boldsymbol{x}}}$, i.e.

$$f_h = \sum_{j_t=1}^{E_t}\sum_{j_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}} f_{j_t,j_{\boldsymbol{x}}}\varphi^0_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}. \tag{2.17}$$

In particular, f_h is uniquely defined by the vector $\boldsymbol{f} \in \mathbb{R}^{E_tE_{\boldsymbol{x}}}$ of coefficients $f_{j_t,j_{\boldsymbol{x}}}$. Here and in the rest of the work we use pairs of indices $(j_t, j_{\boldsymbol{x}})$ to denote the entries of such coefficient vectors $\boldsymbol{f}$ and sort them in a "time major" order, i.e. $\boldsymbol{f}^\top = (\boldsymbol{f}_1^\top, \ldots, \boldsymbol{f}_{E_t}^\top)$, where $\boldsymbol{f}_{j_t} \in \mathbb{R}^{E_{\boldsymbol{x}}}$ contains the entries $f_{j_t,j_{\boldsymbol{x}}}$ for a given time-index j_t.

On Γ_h we define in addition $S^1_{h_{\boldsymbol{x}}}(\Gamma_h)$ as the space of all piecewise linear, globally continuous functions. Let the set of all vertices of the mesh Γ_h be given by $\{\boldsymbol{x}_{j_{\boldsymbol{x}}}\}_{j_{\boldsymbol{x}}=1}^{N_{\boldsymbol{x}}}$. The nodal basis functions of $S^1_{h_{\boldsymbol{x}}}(\Gamma_h)$ are given by $\{\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\}_{j_{\boldsymbol{x}}=1}^{N_{\boldsymbol{x}}}$, where $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}$ is uniquely

defined by setting $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x}_{k_{\boldsymbol{x}}}) = \delta_{j_{\boldsymbol{x}}k_{\boldsymbol{x}}}$. Here, $\delta_{j_{\boldsymbol{x}}k_{\boldsymbol{x}}}$ denotes the Kronecker delta. By combining $S^1_{h_{\boldsymbol{x}}}(\Gamma_h)$ with $S^0_{h_t}(\mathcal{I}_{h_t})$ we get the tensor product space

$$\begin{aligned} S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h) &:= S^1_{h_{\boldsymbol{x}}}(\Gamma_h) \otimes S^0_{h_t}(\mathcal{I}_{h_t}) \\ &= \operatorname{span}\{\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t} : j_t \in \{1,\dots,E_t\}, j_{\boldsymbol{x}} \in \{1,\dots N_{\boldsymbol{x}}\}\}. \end{aligned} \tag{2.18}$$

A function $g_h \in S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ is uniquely defined by the vector $\boldsymbol{g} \in \mathbb{R}^{E_t N_{\boldsymbol{x}}}$ of coefficients $g_{j_t,j_{\boldsymbol{x}}}$ in the representation

$$g_h = \sum_{j_t=1}^{E_t}\sum_{j_{\boldsymbol{x}}=1}^{N_{\boldsymbol{x}}} g_{j_t,j_{\boldsymbol{x}}}\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}.$$

The spaces $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma)$ and $S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma)$ will be used for the approximation of functions in $H^{-1/2,-1/4}(\Sigma)$ and $H^{1/2,1/4}(\Sigma)$, respectively.

For the discretization of functions $v \in L^2(\Omega)$ we consider an admissible tetrahedral mesh $\Omega_h = \{T_{\Omega,k_{\boldsymbol{x}}}\}_{k_{\boldsymbol{x}}=1}^{E_\Omega}$ and define the space of all piecewise linear, globally continuous functions on Ω_h by $S^1_{h_{\boldsymbol{x}}}(\Omega_h)$. Let $\{\boldsymbol{x}_{\Omega,k_{\boldsymbol{x}}}\}_{k_{\boldsymbol{x}}=1}^{N_\Omega}$ denote the vertices of the mesh Ω_h. Then the nodal basis of $S^1_{h_{\boldsymbol{x}}}(\Omega_h)$ is given by $\{\varphi^1_{\Omega,k_{\boldsymbol{x}}}\}_{k_{\boldsymbol{x}}=1}^{N_\Omega}$, where $\varphi^1_{\Omega,k_{\boldsymbol{x}}}$ is defined by $\varphi^1_{\Omega,k_{\boldsymbol{x}}}(\boldsymbol{x}_{\Omega,j_{\boldsymbol{x}}}) = \delta_{k_{\boldsymbol{x}}j_{\boldsymbol{x}}}$. A function $v_h \in S^1_{h_{\boldsymbol{x}}}(\Omega_h)$ can be expressed in terms of this nodal basis by $v_h = \sum_{k_{\boldsymbol{x}}=1}^{N_\Omega} v_{k_{\boldsymbol{x}}}\varphi^1_{\Omega,k_{\boldsymbol{x}}}$, where $v_{k_{\boldsymbol{x}}}$ are the entries of the corresponding vector $\boldsymbol{v} \in \mathbb{R}^{N_\Omega}$.

We conclude the section by introducing L^2 projection operators for the discrete spaces on Σ and Ω introduced above. The L^2 projection $Q^1_\Omega v \in S^1_{h_{\boldsymbol{x}}}(\Omega_h)$ of a function $v \in L^2(\Omega)$ is defined as the unique solution of

$$\langle Q^1_\Omega v, \Phi_h\rangle_{L^2(\Omega)} = \langle v, \Phi_h\rangle_{L^2(\Omega)} \quad \text{for all } \Phi_h \in S^1_{h_{\boldsymbol{x}}}(\Omega_h).$$

On Σ we consider the operators $Q^{p_{\boldsymbol{x}}\otimes p_t}_\Sigma : L^2(\Sigma) \to S^{p_{\boldsymbol{x}}\otimes p_t}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ where $p_{\boldsymbol{x}} \in \{0,1\}$ and $p_t = 0$. The projection $Q^{p_{\boldsymbol{x}}\otimes p_t}_\Sigma u$ of $u \in L^2(\Sigma)$ is defined as the unique function in $S^{p_{\boldsymbol{x}}\otimes p_t}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ that satisfies

$$\langle Q^{p_{\boldsymbol{x}}\otimes p_t}_\Sigma u, \varphi_h\rangle_\Sigma = \langle u, \varphi_h\rangle_\Sigma \quad \text{for all } \varphi_h \in S^{p_{\boldsymbol{x}}\otimes p_t}_{h_{\boldsymbol{x}},h_t}(\Sigma_h). \tag{2.19}$$

In [27, Section 6.3.2], the interested reader can find several approximation properties of the L^2 projection operators $Q^{p_{\boldsymbol{x}}\otimes p_t}_\Sigma$.

3 Boundary integral equations and boundary element methods

In this chapter we give a short overview of boundary integral operators for the heat equation, boundary integral equations for the solution of related initial boundary value problems, and corresponding boundary element methods for their approximate solution. The description here is close to the one in [28], where an interested reader can find additional details. Most of the discussed results stem from the seminal works [8, 24, 57].

3.1 The boundary integral operators

Let u be a solution to the homogeneous heat equation (1.1) that satisfies an initial condition (1.2) for a given function u_0 on Ω. Such a function u can be expressed by the representation formula

$$u = \widetilde{V}(\alpha\gamma_{1,\Sigma}^{\mathrm{int}}u) - W\gamma_{0,\Sigma}^{\mathrm{int}}u + \widetilde{M_0}u_0, \tag{3.1}$$

see e.g. [28, Section 4.1]. Here, $\widetilde{V}$ is the single layer potential operator, W the double layer potential operator and $\widetilde{M_0}$ the initial potential operator. For densities g and w in $L^1(\Sigma)$ the potentials $\widetilde{V}w$ and Wg have the integral representations

$$\widetilde{V}w(\boldsymbol{x},t) = \int_0^t \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)w(\boldsymbol{y},\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau, \tag{3.2}$$

$$Wg(\boldsymbol{x},t) = \int_0^t \int_\Gamma \alpha\frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{y}}}G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)g(\boldsymbol{y},\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau \tag{3.3}$$

for all $\boldsymbol{x} \in \mathbb{R}^3 \setminus \Gamma$ and $t > 0$, where G_α is the fundamental solution of the heat equation defined by

$$G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau) = \begin{cases} \frac{1}{(4\pi\alpha(t-\tau))^{3/2}}\exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4\alpha(t-\tau)}\right) & \text{if } t-\tau > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

see e.g. [28, Sections 4.4 and 4.6]. More general definitions of the operators $\widetilde{V}$ and W are given in [24, Section 3]. There it is also shown that $\widetilde{V} : H^{-1/2,-1/4}(\Sigma) \to H^{1,1/2}_{;0,}(Q)$

and $W : H^{1/2,1/4}(\Sigma) \to H^{1,1/2}_{;0,}(Q)$ are continuous and that the corresponding potentials satisfy the homogeneous heat equation, i.e. $(\partial_t - \alpha\Delta)\Phi = 0$ in Q for $\Phi = \widetilde{V}w$ and $\Phi = Wu$, respectively.

The initial potential $\widetilde{M}_0 u_0$ of a function $u_0 \in L^2(\Omega)$ admits the representation

$$\widetilde{M}_0 u_0(\boldsymbol{x}, t) = \int_\Omega G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t) u_0(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^3 \text{ and } t > 0. \tag{3.5}$$

The operator $\widetilde{M}_0$ is continuous as a map from $L^2(\Omega)$ to $\mathcal{V}(Q)$; see e.g. [28, Section 4.2]. The space $\mathcal{V}(Q)$ is defined by

$$\mathcal{V}(Q) := L^2(0, T; H^1(\Omega)) \cap H^1(0, T; H^{-1}(\Omega)), \tag{3.6}$$

and equipped with the usual graph norm; see [28, Section 3.3]. It is a dense subspace of $H^{1,1/2}(Q)$, and in addition there holds $\mathcal{V}(Q) \subset C([0, T]; L^2(\Omega))$; see [24, p. 502f.]. Hence, functions $u \in \mathcal{V}(Q)$ have a representative which is continuous in time, which makes point evaluations in time and, in particular, initial conditions well defined. In [28, Section 4.2] it is also proven that $\widetilde{M}_0 u_0(\boldsymbol{x}, 0) = u_0(\boldsymbol{x})$ almost everywhere in Ω and $(\partial_t - \alpha\Delta)\widetilde{M}_0 u_0 = 0$ in Q.

Note that the single layer potential operator $\widetilde{V}$ in (3.2) and the double layer potential operator W in (3.3) are causal. This means that for any $t > 0$ the values of densities g and w in the interval (t, T) do not influence the functions Vw and Wg in $(0, t)$. From a physical perspective, this is to be expected. Mathematically, this property is related to the fact that the heat kernel $G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau)$ in (3.4) vanishes for $t < \tau$. Due to this fact, we will also denote the heat kernel as causal in this work.

By applying the trace operators $\gamma^{\mathrm{int}}_{0,\Sigma}$ and $\alpha\gamma^{\mathrm{int}}_{1,\Sigma}$ to the representation formula (3.1) we obtain two boundary integral equations, which we will later use for the explicit solution of concrete initial boundary value problems. Applying the trace operator $\gamma^{\mathrm{int}}_{0,\Sigma}$ yields the first boundary integral equation

$$\gamma^{\mathrm{int}}_{0,\Sigma} u = V(\alpha\gamma^{\mathrm{int}}_{1,\Sigma} u) + \Big(\frac{1}{2}\mathrm{Id} - K\Big)\gamma^{\mathrm{int}}_{0,\Sigma} u + M_0 u_0, \tag{3.7}$$

where $V := \gamma^{\mathrm{int}}_{0,\Sigma}\widetilde{V}$ is the single layer operator, $K := \frac{1}{2}(\gamma^{\mathrm{int}}_{0,\Sigma} + \gamma^{\mathrm{ext}}_{0,\Sigma})W$ the double layer operator and $M_0 = \gamma^{\mathrm{int}}_{0,\Sigma}\widetilde{M}_0$. Recall that $\gamma^{\mathrm{ext}}_{0,\Sigma}$ is a suitable exterior trace operator; see Remark 2.11. The operators V, K and M_0 admit the integral representations

$$Vw(\boldsymbol{x}, t) = \int_0^t \int_\Gamma G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau) w(\boldsymbol{y}, \tau) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\tau \qquad \text{for all } (\boldsymbol{x}, t) \in \Sigma, \tag{3.8}$$

$$Kg(\boldsymbol{x}, t) = \int_0^t \int_\Gamma \alpha \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{y}}} G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau) g(\boldsymbol{y}, \tau) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\tau \quad \text{for all } (\boldsymbol{x}, t) \in \Sigma, \tag{3.9}$$

$$M_0 u_0(\boldsymbol{x}, t) = \int_\Omega G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t) u_0(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} \qquad \text{for all } (\boldsymbol{x}, t) \in \Sigma, \tag{3.10}$$

for $w,\ g \in L^\infty(\Sigma)$ and $u_0 \in L^2(\Omega)$, where additional regularity assumptions on the spatial boundary Γ are needed for the representation in (3.9); see for example [63, Sections 3.3.2 and 3.3.3] for similar results on the corresponding boundary integral operators V and K for elliptic PDEs.

By applying the scaled Neumann trace operator $\alpha\gamma_{1,\Sigma}^{\mathrm{int}}$ to the representation formula (3.1) we obtain the second boundary integral equation

$$\alpha\gamma_{1,\Sigma}^{\mathrm{int}} u = \Big(\frac{1}{2}\mathrm{Id} + K_T'\Big)(\alpha\gamma_{1,\Sigma}^{\mathrm{int}} u) + D\gamma_{0,\Sigma}^{\mathrm{int}} u + M_1 u_0, \tag{3.11}$$

with the adjoint time-reversed double layer operator $K_T' := \frac{1}{2}(\alpha\gamma_{1,\Sigma}^{\mathrm{int}} + \alpha\gamma_{1,\Sigma}^{\mathrm{ext}})\widetilde{V}$, the hypersingular operator $D := -\alpha\gamma_{1,\Sigma}^{\mathrm{int}}(W)$ and the operator $M_1 := \alpha\gamma_{1,\Sigma}^{\mathrm{int}}\widetilde{M}_0$. The operator K_T' is indeed the adjoint of the time-reversed double layer operator $K_T := \Theta_T \circ K$, where $\Theta_T f(\boldsymbol{x}, t) := f(\boldsymbol{x}, T - t)$. For $w \in L^\infty(\Sigma)$ and a sufficiently regular boundary Γ there holds

$$K_T' w = \int_0^t \int_\Gamma \alpha \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{x}}} G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau) w(\boldsymbol{y}, \tau)\, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\, \mathrm{d}\tau \qquad \text{for all } (\boldsymbol{x}, t) \in \Sigma, \tag{3.12}$$

see [63, Section 3.3.3] for a similar result for elliptic PDEs. The integral representation

$$M_1 u_0 = \int_\Omega \alpha \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{x}}} G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t) u_0(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y} \qquad \text{for all } (\boldsymbol{x}, t) \in \Sigma \tag{3.13}$$

is valid for all $u_0 \in L^2(\Omega)$. The hypersingular operator D does not admit a similar integral representation. This is related to the fact that the kernel

$$((\boldsymbol{x}, t), (\boldsymbol{y}, \tau)) \mapsto \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{x}}} \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{y}}} G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau) \tag{3.14}$$

has a strong singularity on the diagonal $(\boldsymbol{x}, t) = (\boldsymbol{y}, \tau)$. When we consider the bilinear form associated with D, we can use a so-called integration by parts formula to overcome this problem. This integration by parts formula will be discussed in detail in Chapter 4.

From the continuity of the potential operators $\widetilde{V}$, W and $\widetilde{M}_0$ and the continuity of the trace operators $\gamma_{0,\Sigma}^{\mathrm{int}}$ and $\gamma_{1,\Sigma}^{\mathrm{int}}$ we conclude that the operators

$$\begin{aligned}
&V : H^{-1/2,-1/4}(\Sigma) \to H^{1/2,1/4}(\Sigma), && K_T' : H^{-1/2,-1/4}(\Sigma) \to H^{-1/2,-1/4}(\Sigma),\\
&K : H^{1/2,1/4}(\Sigma) \to H^{1/2,1/4}(\Sigma), && D : H^{1/2,1/4}(\Sigma) \to H^{-1/2,-1/4}(\Sigma),\\
&M_0 : L^2(\Omega) \to H^{1/2,1/4}(\Sigma), && M_1 : L^2(\Omega) \to H^{-1/2,-1/4}(\Sigma)
\end{aligned}$$

are continuous. Furthermore one can show that V and D are positive definite isomorphisms, which we express in the following theorems.

THEOREM 3.1 ([24, Corollary 3.13 (a)] and [57, Theorem 4.3]). *The single layer operator V is an isomorphism between $H^{-1/2,-1/4}(\Sigma)$ and $H^{1/2,1/4}(\Sigma)$ and there exists a constant $c_1^V > 0$ such that*

$$\langle Vq, q\rangle_\Sigma \geq c_1^V \|q\|^2_{H^{-1/2,-1/4}(\Sigma)} \qquad \text{for all } q \in H^{-1/2,-1/4}(\Sigma). \tag{3.15}$$

THEOREM 3.2 ([24, Corollary 3.13 (b)]). *The hypersingular operator D is an isomorphism between $H^{1/2,1/4}(\Sigma)$ and $H^{-1/2,-1/4}(\Sigma)$ and there exists a constant $c_1^D > 0$ such that*

$$\langle D\varphi, \varphi\rangle_\Sigma \geq c_1^D \|\varphi\|^2_{H^{1/2,1/4}(\Sigma)} \qquad \text{for all } \varphi \in H^{1/2,1/4}(\Sigma). \tag{3.16}$$

3.2 Solving initial BVPs by boundary integral equations

The integral operators introduced in Section 3.1 can be used to construct solutions to initial boundary value problems of the heat equation. Let us first focus on the initial Dirichlet boundary value problem (1.1)–(1.3), where we follow the lines of [28, Section 5.3] and assume $g \in H^{1/2,1/4}(\Sigma)$ and $u_0 \in L^2(\Omega)$. If we want to use the representation formula (3.1) to construct the solution u, we have to determine the unknown Neumann datum $\gamma_{1,\Sigma}^{\text{int}} u$. According to the first boundary integral equation (3.7) there holds

$$Vq = \Big(\frac{1}{2}\mathrm{Id} + K\Big)g - M_0 u_0$$

for $q := \alpha\gamma_{1,\Sigma}^{\text{int}} u$. The corresponding variational problem is to find $q \in H^{-1/2,-1/4}(\Sigma)$ such that

$$\langle Vq, \psi\rangle_\Sigma = \Big\langle \Big(\frac{1}{2}\mathrm{Id} + K\Big)g - M_0 u_0, \psi\Big\rangle_\Sigma \quad \text{for all } \psi \in H^{-1/2,-1/4}(\Sigma). \tag{3.17}$$

Due to the continuity and linearity of the boundary integral operators, Theorem 3.1, and the Lemma of Lax–Milgram, this variational problem admits a unique solution q that depends continuously on g and u_0.

Another option is to use the following indirect approach. We know that for any density $w \in H^{-1/2,-1/4}(\Sigma)$ the function

$$u := \widetilde{V} w + \widetilde{M_0} u_0 \tag{3.18}$$

is a solution of the homogeneous heat equation (1.1) that satisfies the initial condition (1.2). By applying the trace operator $\gamma_{0,\Sigma}^{\text{int}}$ to u we see that the Dirichlet condition (1.3) is satisfied if $w \in H^{-1/2,-1/4}(\Sigma)$ solves the variational problem

$$\langle Vw, \psi\rangle_\Sigma = \langle g - M_0 u_0, \psi\rangle_\Sigma \quad \text{for all } \psi \in H^{-1/2,-1/4}(\Sigma). \tag{3.19}$$

The existence of a unique density w with this property follows as before.

The initial Neumann boundary value problem (1.1)–(1.2) with (1.4) and the Neumann datum $q \in H^{-1/2,-1/4}(\Sigma)$ can be treated in a similar way; see [27, Section 5.3]. The direct approach uses the representation formula (3.1) to construct the solution u. The unknown Dirichlet datum $g := \gamma_{0,\Sigma}^{\mathrm{int}} u$ can be determined by using the second boundary integral equation (3.11), i.e.

$$Dg = \Big(\frac{1}{2}\mathrm{Id} - K_T'\Big)(\alpha q) - M_1 u_0.$$

The equivalent variational problem is to find $g \in H^{1/2,1/4}(\Sigma)$ such that

$$\langle Dg, \varphi\rangle_\Sigma = \Big\langle \Big(\frac{1}{2}\mathrm{Id} - K_T'\Big)(\alpha q) - M_1 u_0, \varphi\Big\rangle_\Sigma \quad \text{for all } \varphi \in H^{1/2,1/4}. \tag{3.20}$$

Due to the continuity and linearity of the boundary integral operators, Theorem 3.2, and the Lemma of Lax–Milgram, it follows that there exists a unique solution g to this problem which depends continuously on q and u_0.

Alternatively, we can use the following indirect approach, where we set

$$u := -Wf + \widetilde{M}_0 u_0 \tag{3.21}$$

and require that the density $f \in H^{1/2,1/4}(\Sigma)$ satisfies

$$\langle Df, \varphi\rangle_\Sigma = \langle \alpha q - M_1 u_0, \varphi\rangle_\Sigma \quad \text{for all } \varphi \in H^{1/2,1/4}(\Sigma). \tag{3.22}$$

A unique density f with these properties exists and the resulting function u is the solution to the considered initial Neumann boundary value problem.

3.3 Boundary element methods

In this section we describe how to discretize the variational formulations (3.17) and (3.20) by a Galerkin–Bubnov method to obtain approximate solutions to the initial Dirichlet and Neumann boundary value problems of the heat equation, respectively. The variational formulations from the indirect approaches can be handled analogously.

Let $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ be a space-time tensor product mesh of Σ, Ω_h be a tetrahedral mesh of Ω, and $S^{p_x \otimes p_t}_{h_x,h_t}(\Sigma_h)$ and $S^1_{h_x}(\Omega_h)$ be piecewise polynomial spaces on Σ_h and Ω_h as in Section 2.4. A Galerkin–Bubnov discretization of the variational problem (3.17) is to find $q_h \in S^{0\otimes 0}_{h_x,h_t}$ such that

$$\langle Vq_h, \psi_h\rangle_\Sigma = \Big\langle \Big(\frac{1}{2}\mathrm{Id} + K\Big) g_h - M_0 u_{0,h}, \psi_h\Big\rangle_\Sigma \quad \text{for all } \psi_h \in S^{0\otimes 0}_{h_x,h_t}(\Sigma_h). \tag{3.23}$$

Note that we have replaced g and u_0 on the right-hand side of (3.17) by the corresponding approximations $g_h = Q_\Sigma^{1\otimes 0} g \in S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ and $u_{0,h} = Q^1_\Omega u_0 \in S^1_{h_x}(\Omega_h)$ with the L^2 projection operators $Q_\Sigma^{1\otimes 0}$ and Q^1_Ω from Section 2.4. The variational formulation (3.23) has a unique solution due to Theorem 3.1. It can be rewritten as the system of linear equations

$$\mathsf{V}_h \boldsymbol{q} = \Big(\frac{1}{2}\mathsf{M}_h + \mathsf{K}_h\Big)\boldsymbol{g} - \mathsf{M}_h^0 \boldsymbol{u}^0. \tag{3.24}$$

Here $\boldsymbol{q}$, $\boldsymbol{g}$ and $\boldsymbol{u}^0$ denote the coefficient vectors of the discrete functions q_h, g_h and $u_{0,h}$, respectively. As stated in Section 2.4 we use pairs of indices $(j_t, j_{\boldsymbol{x}})$ to address the entries of $\boldsymbol{q}$ and $\boldsymbol{g}$ and assume that these vectors are sorted in a "time-major" order. The matrices V_h, K_h and M_h^0 are the BEM matrices of the operators V, K and M_0, respectively, and M_h is the mass matrix. Their entries are obtained by evaluating the corresponding bilinear forms $\langle V\cdot,\cdot\rangle$, etc., for pairs of basis functions. The entries of V_h for row indices $(k_t, k_{\boldsymbol{x}})$ and column indices $(j_t, j_{\boldsymbol{x}})$ with k_t, $j_t \in \{1,\dots,E_t\}$ and $k_{\boldsymbol{x}}$, $j_{\boldsymbol{x}} \in \{1,\dots,E_{\boldsymbol{x}}\}$ are given by

$$\begin{aligned}&\mathsf{V}_h[(k_t-1)E_{\boldsymbol{x}} + k_{\boldsymbol{x}}, (j_t-1)E_{\boldsymbol{x}} + j_{\boldsymbol{x}}] \\ &\qquad = \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.\end{aligned} \tag{3.25}$$

For the assembly of K_h and the sparse mass matrix M_h we need to evaluate

$$\begin{aligned}&\mathsf{K}_h[(k_t-1)E_{\boldsymbol{x}} + k_{\boldsymbol{x}}, (j_t-1)N_{\boldsymbol{x}} + j_{\boldsymbol{x}}] \\ &\qquad = \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_\Gamma \alpha \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{y}}} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t,\end{aligned} \tag{3.26}$$

$$\mathsf{M}_h[(k_t-1)E_{\boldsymbol{x}} + k_{\boldsymbol{x}}, (j_t-1)N_{\boldsymbol{x}} + j_{\boldsymbol{x}}] = \begin{cases} h_{t,k_t} \int_{\gamma_{k_{\boldsymbol{x}}}} \varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x})\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} & \text{if } j_t = k_t, \\ 0 & \text{otherwise} \end{cases} \tag{3.27}$$

for all k_t, $j_t \in \{1,\dots,E_t\}$, $k_{\boldsymbol{x}} \in \{1,\dots,E_{\boldsymbol{x}}\}$ and $j_{\boldsymbol{x}} \in \{1,\dots,N_{\boldsymbol{x}}\}$, with h_{t,k_t} given in (2.13). Finally, the entries of M_h^0 are given by

$$\mathsf{M}_h^0[(k_t-1)E_{\boldsymbol{x}} + k_{\boldsymbol{x}}, j_{\boldsymbol{x}}] = \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} \int_\Omega G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t)\varphi^1_{\Omega,j_{\boldsymbol{x}}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \tag{3.28}$$

for all $k_t \in \{1,\dots,E_t\}$, $k_{\boldsymbol{x}} \in \{1,\dots,E_{\boldsymbol{x}}\}$ and $j_{\boldsymbol{x}} \in \{1,\dots,N_\Omega\}$.

The matrix V_h has the lower triangular block structure

$$\begin{pmatrix} \mathsf{V}_h^{1,1} & 0 & \dots & 0 \\ \mathsf{V}_h^{2,1} & \mathsf{V}_h^{2,2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{V}_h^{E_t,1} & \mathsf{V}_h^{E_t,2} & \dots & \mathsf{V}_h^{E_t,E_t} \end{pmatrix}, \tag{3.29}$$

where the blocks $\mathsf{V}_h^{k_t,j_t} \in \mathbb{R}^{E_{\boldsymbol{x}} \times E_{\boldsymbol{x}}}$ contain all entries of V_h corresponding to a temporal row index k_t and a temporal column index j_t. The blocks $\mathsf{V}_h^{k_t,j_t}$ with $j_t > k_t$ above the diagonal are zero due to the causality of the operator V in (3.8). If the time intervals in the partition $\mathcal{I}_{h_t}$ are uniform, the matrix V_h is even a lower triangular block Toeplitz matrix, i.e. $\mathsf{V}_h^{k_t,j_t} = \mathsf{V}_h^{m_t,\ell_t}$ if $k_t - j_t = m_t - \ell_t$. The matrix K_h has the same block structure as V_h with blocks $\mathsf{K}_h^{k_t,j_t} \in \mathbb{R}^{E_{\boldsymbol{x}} \times N_{\boldsymbol{x}}}$. The mass matrix M_h is block-diagonal and the diagonal blocks $\mathsf{M}_h^{k_t,k_t} \in \mathbb{R}^{E_{\boldsymbol{x}} \times N_{\boldsymbol{x}}}$ are sparse. The structure of the matrix M_h^0 is different. One can subdivide its rows according to the temporal row indices k_t, but not its columns. In particular, all entries of M_h^0 are nonzero.

The numerical evaluation of the integrals (3.25) and (3.26) is discussed, for example, in [57, Section 6.2], [51, Appendix B], and [81, Sections 3.1 and 3.2]. The usual strategy is to combine an analytic integration in the temporal variables with numerical quadrature formulae in space. In [81], the authors took particular care to give a complete and error-free description of this process. The temporal mesh in that work is assumed to be uniform but the generalization for the case of non-uniform time steps is straightforward. The computation of the entries of M_h^0 in (3.28) was not considered in [81] or the other mentioned works. An analytic integration in time can also be done in this case. The remaining spatial integrals can be handled by numerical quadrature formulae on pairs of tetrahedra $\tau_{k_{\boldsymbol{x}}} \in \Omega_h$ and triangles $\gamma_{j_{\boldsymbol{x}}} \in \Gamma_h$. The only difficulty arises for the time-index $k_t = 1$ because in that case the resulting integrals are singular if the intersection of $\overline{\tau_{k_{\boldsymbol{x}}}}$ and $\overline{\gamma_{j_{\boldsymbol{x}}}}$ is not empty. In [31, Section A.1.2] it is shown how to handle such integrals in the spatially two-dimensional case. For the examples in this work, we used a heuristic subdivision strategy instead, which worked sufficiently well.

When the solution $\boldsymbol{q}$ to the system (3.24) has been computed, we can use the corresponding function q_h together with the approximations g_h of the Dirichlet datum g and the approximation $u_{0,h}$ of the initial datum u_0 to construct

$$\widetilde{u}_H = \widetilde{V} q_h - W g_h + \widetilde{M}_0 u_{0,h}, \tag{3.30}$$

which approximates the solution u to the initial Dirichlet boundary value problem (1.1)–(1.3). The evaluation of $\widetilde{V} q_h(\boldsymbol{x}, t)$ and $W g_h(\boldsymbol{x}, t)$ for $\boldsymbol{x} \in \Omega$ and $t \in (0, T)$ is discussed in [81, Section 3.5]. The initial potential $\widetilde{M}_0 u_{0,h}$ can be evaluated by standard quadrature schemes since the related integrals are not singular.

In the discrete version of the variational problem (3.20) we search for $g_h \in S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ such that

$$\langle D g_h, \varphi_h \rangle_\Sigma = \left\langle \left(\frac{1}{2}\mathrm{Id} - K_T'\right)(\alpha q_h) - M_1 u_{0,h}, \varphi_h \right\rangle_\Sigma \quad \text{for all } \varphi_h \in S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}, \tag{3.31}$$

where $u_{0,h} = Q^1_\Omega u_0 \in S^1_{h_x}(\Omega_h)$ and $q_h = Q^{0\otimes 0}_\Sigma q \in S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}$. Note that the Neumann datum q has to be sufficiently regular to define q_h in this manner, which we assume

here. The discrete variational problem (3.31) admits a unique solution g_h, which is a consequence of Theorem 3.2. We can use it to construct an approximate solution $\widetilde{u}_H$ to the initial Neumann boundary value problem (1.1)–(1.2) with (1.4) by using (3.30) as above.

The system of linear equations equivalent to (3.31) reads

$$\mathsf{D}_h \boldsymbol{g} = \Big(\frac{1}{2}\mathsf{M}_h^{\top_{\boldsymbol{x}}} - \mathsf{K}_h^{\top_{\boldsymbol{x}}}\Big)\boldsymbol{q} - \mathsf{M}_h^1 \boldsymbol{u}^0. \tag{3.32}$$

As before, $\boldsymbol{g}$, $\boldsymbol{q}$ and $\boldsymbol{u}^0$ denote the vector of coefficients of g_h, q_h and $u_{0,h}$, respectively. The matrices $\mathsf{M}_h^{\top_{\boldsymbol{x}}}$ and $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ are obtained by a blockwise transposition from the matrices M_h and K_h in (3.24), which we indicate by the superscript $\top_{\boldsymbol{x}}$. E.g., there holds

$$\mathsf{K}_h^{\top_{\boldsymbol{x}}} = \begin{pmatrix} (\mathsf{K}_h^{1,1})^\top & 0 & \dots & 0 \\ (\mathsf{K}_h^{2,1})^\top & (\mathsf{K}_h^{2,2})^\top & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ (\mathsf{K}_h^{E_t,1})^\top & (\mathsf{K}_h^{E_t,2})^\top & \dots & (\mathsf{K}_h^{E_t,E_t})^\top \end{pmatrix}. \tag{3.33}$$

The matrix D_h has the same block structure as the matrix V_h in (3.29) with blocks $\mathsf{D}_h^{k_t,j_t} \in \mathbb{R}^{N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}}$, due to the causality of the operator D. The entries of D_h are computed using the integration by parts formula in Theorem 4.8 together with (4.59). Further details about this computation are given at the end of Section 4.2.2.

The entries of the matrix M_h^1 are given by

$$\begin{aligned} &\mathsf{M}_h^1[(k_t-1)N_{\boldsymbol{x}} + k_{\boldsymbol{x}}, j_{\boldsymbol{x}}] \\ &\qquad = \int_{t_{k_t-1}}^{t_{k_t}} \int_\Gamma \int_\Omega \alpha \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{x}}} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t) \varphi_{\Omega,j_{\boldsymbol{x}}}^1(\boldsymbol{y}) \varphi_{\boldsymbol{x},k_{\boldsymbol{x}}}^1(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{y}\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \end{aligned} \tag{3.34}$$

for all $k_t \in \{1, \dots, E_t\}$, $k_{\boldsymbol{x}} \in \{1, \dots, N_{\boldsymbol{x}}\}$ and $j_{\boldsymbol{x}} \in \{1, \dots, N_\Omega\}$. The computation of these entries can be handled similarly to the computation of the entries of M_h^0, which we described above.

For a priori error estimates related to the discretization (3.23) of the variational problem (3.17) corresponding to the Dirichlet problem and the discretization (3.31) of the variational problem (3.20) corresponding to the Neumann problem we refer to the literature. A detailed discussion of error estimates for a Dirichlet problem is already given in [57, Section 7] and results for both problems in [24, Section 5]. A comprehensive overview can be found in [27, Sections 7.1 and 7.2], where also meshes without tensor product structure are considered. In [53, Section 4] error estimates for a mixed boundary value problem including prescribed Dirichlet, Neumann, and Robin boundary values are given. Note that in [53] and [57] also the error related to

the approximation of the right-hand side is investigated for the respective problems. Of further interest is [23], where an alternative estimate for the approximation error for the Dirichlet problem in the energy norm $\|\cdot\|_{H^{-1/2,-1/4}(\Sigma)}$ is given; see Theorem 3.3. This result shows, in particular, how to adapt the time step sizes to the spatial mesh widths when constructing a sequence of tensor product meshes to obtain better convergence rates in the energy norm than those predicted in the previously mentioned works.

4 An integration by parts formula for the hypersingular operator

The hypersingular operator D was introduced in Section 3.1 as $D = -\alpha\gamma_{1,\Sigma}^{\text{int}} W$ with the double layer potential operator W from (3.3). We have already pointed out that D does not admit an integral representation like the single layer operator V in (3.8) or the double layer operator K in (3.9) due to the strong singularity of the related kernel function (3.14). However, an alternative representation is available for the bilinear form $\langle D\cdot, \cdot\rangle_\Sigma$ on $H^{1/2,1/4}(\Sigma) \times H^{1/2,1/4}(\Sigma)$, which eventually allows for an evaluation by means of weakly singular integrals. This representation is known as integration by parts formula and is covered in detail in this chapter and the corresponding publication [77].

The contents of this chapter are structured as follows. In Section 4.1 we present auxiliary definitions and results which are required in this chapter. A general version of the integration by parts formula for the bilinear form $\langle D\cdot, \cdot\rangle_\Sigma$ is discussed in Section 4.2 with a detailed proof in Section 4.2.1. The integration by parts formula includes a bilinear form b, which cannot be evaluated directly for non-smooth functions in its general form. In Section 4.2.2 we provide an alternative representation for this bilinear form which is valid for functions in the typical tensor product discretization spaces.

4.1 Auxiliary definitions and results

In the following subsections we discuss a few additional results that are needed later in this chapter. We start by citing several definitions and results from distribution theory following the lines of [48] and [71], which will be needed for the proof of the integration by parts formula in Section 4.2. This formula includes surface curls of functions in $H^{1/2,1/4}(\Sigma)$, which we define in Section 4.1.2. In Section 4.1.3 we consider selected additional aspects of the heat equation, which we will need in Section 4.2.2.

4.1.1 Selected results from distribution theory

The distributions on an open set $A \subset \mathbb{R}^d$ are denoted by $\mathcal{D}'(A)$. These are the linear functionals on $C_c^\infty(A)$ which are sequentially continuous with the usual notion of

convergence in $C_c^\infty(A)$; see e.g. [48, page 65]. In the same manner, the set of all linear, sequentially continuous functionals on $C^\infty(A)$ is denoted by $\mathcal{E}'(A)$. We use the notation $u[w]$ for the application of u in $\mathcal{D}'(A)$ or $\mathcal{E}'(A)$ to a function w in $C_c^\infty(A)$ or $C^\infty(A)$, respectively.

Let $L^1_{\mathrm{loc}}(A)$ be the set of all measurable functions $u : A \to \mathbb{R}$ such that $\|u\|_{L^1(K)} < \infty$ for all compact subsets K of A. For each $u \in L^1_{\mathrm{loc}}(A)$ we can define a distribution by setting

$$u[w] := \int_A u\, w \,\mathrm{d}\boldsymbol{x}.$$

A distribution that can be represented by a function in $L^1_{\mathrm{loc}}(A)$ in this way is often called regular.

For a multi-index $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ the derivative $D^{\boldsymbol{\alpha}} = \prod_{j=1}^d \partial_{x_j}^{\alpha_j}$ of a distribution $S \in \mathcal{D}'(A)$ is defined by

$$D^{\boldsymbol{\alpha}} S[w] = (-1)^{|\boldsymbol{\alpha}|} S[D^{\boldsymbol{\alpha}} w]$$

for all $w \in C_c^\infty(A)$, where $|\boldsymbol{\alpha}| := \sum_j |\alpha_j|$. The derivative $D^{\boldsymbol{\alpha}} S$ is itself a distribution.

The restriction $S|_B$ of a distribution $S \in \mathcal{D}'(A)$ to an open subset B of A is defined by setting $S|_B[w] = S[\widetilde{w}]$ for all $w \in C_c^\infty(B)$, where $\widetilde{w}$ denotes the extension of w to A by zero. There holds $S|_B \in \mathcal{D}'(B)$. The support $\operatorname{supp}(S)$ of a distribution $S \in \mathcal{D}'(A)$ is defined as the largest relatively closed subset F of A such that $S|_{A\setminus F} = 0$. In [71, Theorem 24.2] it is shown that

$$\mathcal{E}'(A) = \{S \in \mathcal{D}'(A) : \operatorname{supp}(S) \text{ is a compact subset of } A\}.$$

In the following the focus lies on distributions on the whole space $\mathbb{R}^d$. The convolution of a distribution $S \in \mathcal{D}'(\mathbb{R}^d)$ and a test function $u \in C_c^\infty(\mathbb{R}^d)$ is defined by

$$S * u : \ \boldsymbol{x} \mapsto S_{\boldsymbol{y}}[u(\boldsymbol{x} - \cdot_{\boldsymbol{y}})],$$

where the index $\boldsymbol{y}$ added to S indicates that S acts with respect to this variable. It can be shown that this is a function in $C^\infty(\mathbb{R}^d)$ or even in $C_c^\infty(\mathbb{R}^d)$ if $\operatorname{supp}(S)$ is compact; see e.g. [71, Theorem 27.3]. For $S \in \mathcal{D}'(\mathbb{R}^d)$, $R \in \mathcal{E}'(\mathbb{R}^d)$ and $u \in C_c^\infty(\mathbb{R}^d)$ we can also consider the functions

$$S_{\boldsymbol{y}}[u(\cdot_{\boldsymbol{x}} + \cdot_{\boldsymbol{y}})] : \ \boldsymbol{x} \mapsto S_{\boldsymbol{y}}[u(\boldsymbol{x} + \cdot_{\boldsymbol{y}})]$$

and $R_{\boldsymbol{y}}[u(\cdot_{\boldsymbol{x}} + \cdot_{\boldsymbol{y}})]$, which are in $C^\infty(\mathbb{R}^d)$ and $C_c^\infty(\mathbb{R}^d)$, respectively. This follows directly from the alternative representation $S_{\boldsymbol{y}}[u(\boldsymbol{x} + \cdot_{\boldsymbol{y}})] = (S * \check{u})(-x)$, where $\check{u}(\boldsymbol{x}) := u(-\boldsymbol{x})$. In particular, we can define the convolution of $S \in \mathcal{D}'(\mathbb{R}^d)$ and $R \in \mathcal{E}'(\mathbb{R}^d)$ as an element in $\mathcal{D}'(\mathbb{R}^d)$ via

$$(S * R)[u] := S_{\boldsymbol{x}}[R_{\boldsymbol{y}}[u(\cdot_{\boldsymbol{x}} + \cdot_{\boldsymbol{y}})]] \qquad \text{for all } u \in C_c^\infty(\mathbb{R}^d) \tag{4.1}$$

and similarly $(R * S) \in \mathcal{D}'(\mathbb{R}^d)$ by

$$(R * S)[u] := R_{\boldsymbol{x}}[S_{\boldsymbol{y}}[u(\cdot_{\boldsymbol{x}} + \cdot_{\boldsymbol{y}})]] \qquad \text{for all } u \in C_c^\infty(\mathbb{R}^d).$$

Let us collect some properties of the convolution of distributions defined in this way.

PROPOSITION 4.1 ([71, Theorem 27.4, Propositions 27.3 and 27.5]). *Let $S \in \mathcal{D}'(\mathbb{R}^d)$, $R \in \mathcal{E}'(\mathbb{R}^d)$ and $\boldsymbol{\alpha} \in \mathbb{N}_0^d$. Then*

$$\begin{aligned} S * R &= R * S, \\ D^{\boldsymbol{\alpha}}(S * R) &= (D^{\boldsymbol{\alpha}} S) * R = S * (D^{\boldsymbol{\alpha}} R), \\ \delta_{\mathbf{0}} * S &= S, \end{aligned}$$

where $\delta_{\mathbf{0}}$ is the delta distribution defined by $\delta_{\mathbf{0}}[w] = w(0)$ for all $w \in C^\infty(\mathbb{R}^d)$.

The distributions that play an important role in this chapter are those generated by the adjoint operators $(\gamma_{0,\Sigma}^{\mathrm{int}})'$ and $(\gamma_{1,\Sigma}^{\mathrm{int}})'$ of the trace operators in Section 2.3. If we interpret the Dirichlet trace operator $\gamma_{0,\Sigma}^{\mathrm{int}}$ as a continuous operator from $C^\infty(\mathbb{R}^3 \times \mathbb{R})$ to $H^{1/2,1/4}(\Sigma)$, its adjoint $(\gamma_{0,\Sigma}^{\mathrm{int}})' : H^{-1/2,-1/4}(\Sigma) \to \mathcal{E}'(\mathbb{R}^3 \times \mathbb{R})$ is given by

$$(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi)[w] := \langle \varphi, \gamma_{0,\Sigma}^{\mathrm{int}} w \rangle_\Sigma \tag{4.2}$$

for $\varphi \in H^{-1/2,-1/4}(\Sigma)$ and $w \in C^\infty(\mathbb{R}^3 \times \mathbb{R})$. Similarly, we consider the Neumann trace operator $\gamma_{1,\Sigma}^{\mathrm{int}}$ as a continuous operator from $C^\infty(\mathbb{R}^3 \times \mathbb{R})$ to $H^{-1/2,-1/4}(\Sigma)$ and end up with its adjoint $(\gamma_{1,\Sigma}^{\mathrm{int}})' : H^{1/2,1/4}(\Sigma) \to \mathcal{E}'(\mathbb{R}^3 \times \mathbb{R})$ defined by

$$(\gamma_{1,\Sigma}^{\mathrm{int}})'(\psi)[w] := \langle \gamma_{1,\Sigma}^{\mathrm{int}} w, \psi \rangle_\Sigma \tag{4.3}$$

for $\psi \in H^{1/2,1/4}(\Sigma)$ and $w \in C^\infty(\mathbb{R}^3 \times \mathbb{R})$.

4.1.2 The surface curl in $H^{1/2,1/4}(\Sigma)$

For the integration by parts formula of the hypersingular operator D we need to define the surface curl of a function in $H^{1/2,1/4}(\Sigma)$. In this work we use a weak variational definition. This is inspired by the definitions of the purely spatial tangential trace and surface curl in [64]; see the definitions of γ_T and $\nabla_\Gamma^\perp$ in Sections 16.2 and 16.10, respectively.

DEFINITION 4.2. *The* surface curl $\mathbf{curl}_\Sigma \varphi \in \boldsymbol{H}^{-1/2,-1/4}(\Sigma)$ *of* $\varphi \in H^{1/2,1/4}(\Sigma)$ *is defined by*

$$\langle \mathbf{curl}_\Sigma \varphi, \boldsymbol{\psi} \rangle_\Sigma = \langle \nabla E_\Sigma \varphi, \mathbf{curl}(E_\Sigma \boldsymbol{\psi}) \rangle_{\boldsymbol{L}^2(Q)} \quad \text{for all } \boldsymbol{\psi} \in \boldsymbol{H}^{1/2,1/4}(\Sigma), \tag{4.4}$$

where E_Σ is the continuous right inverse of $\gamma_{0,\Sigma}^{\mathrm{int}}$ from Corollary 2.8 and its application to a vector-valued function is understood componentwise.

PROPOSITION 4.3. *The operator* $\mathbf{curl}_\Sigma : H^{1/2,1/4}(\Sigma) \to \boldsymbol{H}^{-1/2,-1/4}(\Sigma)$ *is well-defined and continuous. In particular,* (4.4) *is independent of the extension* E_Σ. *If* $\widetilde{\varphi} \in C^2(\overline{Q})$ *and* $\varphi = \widetilde{\varphi}|_\Sigma$, *then there holds*

$$\mathbf{curl}_\Sigma \varphi = \nabla \widetilde{\varphi} \times \boldsymbol{n}. \tag{4.5}$$

Proof. We start by showing that the definition in (4.4) is independent of the extension $E_\Sigma \boldsymbol{\psi}$ of the test function $\boldsymbol{\psi} \in \boldsymbol{H}^{1/2,1/4}(\Sigma)$. For this purpose, let $\widetilde{\boldsymbol{\psi}}_1$ and $\widetilde{\boldsymbol{\psi}}_2$ in $\boldsymbol{H}^{1,1/2}(Q)$ denote two extensions of $\boldsymbol{\psi}$ to Q. Then, the difference $\widetilde{\boldsymbol{\psi}}_1 - \widetilde{\boldsymbol{\psi}}_2$ is in $\boldsymbol{H}^{1,1/2}_{0;}(Q)$ and, therefore,

$$\langle \nabla u, \mathbf{curl}\, \widetilde{\boldsymbol{\psi}}_1 - \mathbf{curl}\, \widetilde{\boldsymbol{\psi}}_2 \rangle_{\boldsymbol{L}^2(Q)} = 0 \tag{4.6}$$

for all $u \in H^{1,1/2}(Q)$. Indeed, integration by parts yields

$$\langle \nabla u, \mathbf{curl}\, \boldsymbol{w} \rangle_{\boldsymbol{L}^2(Q)} = \langle u, \boldsymbol{n} \cdot \mathbf{curl}\, \boldsymbol{w} \rangle_\Sigma - \langle u, \operatorname{div}(\mathbf{curl}\, \boldsymbol{w}) \rangle_{L^2(Q)} = 0$$

for $\boldsymbol{w} \in \boldsymbol{C}^\infty_c(Q)$, and thus (4.6) follows from the density of $\boldsymbol{C}^\infty_c(Q)$ in $\boldsymbol{H}^{1,1/2}_{0;}(Q)$; see Proposition 2.5. This proves that (4.4) is independent of the extension $E_\Sigma \boldsymbol{\psi}$ of $\boldsymbol{\psi}$.

Similarly, we conclude that (4.4) is independent of the extension $E_\Sigma \varphi$ of φ by using that

$$\langle \nabla u, \mathbf{curl}\, \boldsymbol{w} \rangle_{\boldsymbol{L}^2(Q)} = \langle \nabla u \times \boldsymbol{n}, \boldsymbol{w} \rangle_\Sigma + \langle \mathbf{curl}(\nabla u), \boldsymbol{w} \rangle_{\boldsymbol{L}^2(Q)} = 0$$

for all $u \in C^\infty_c(Q)$ and $\boldsymbol{w} \in \boldsymbol{H}^{1,1/2}(Q)$.

To see that $\mathbf{curl}_\Sigma \varphi \in \boldsymbol{H}^{-1/2,-1/4}(\Sigma)$ and that $\mathbf{curl}_\Sigma$ is continuous as a mapping from $H^{1/2,1/4}(\Sigma)$ to $\boldsymbol{H}^{-1/2,-1/4}(\Sigma)$ we estimate

$$\begin{aligned} |\langle \mathbf{curl}_\Sigma \varphi, \boldsymbol{\psi} \rangle_\Sigma| &= |\langle \nabla E_\Sigma \varphi, \mathbf{curl}(E_\Sigma \boldsymbol{\psi}) \rangle_{\boldsymbol{L}^2(Q)}| \le \|\nabla E_\Sigma \varphi\|_{\boldsymbol{L}^2(Q)} \|\mathbf{curl}(E_\Sigma \boldsymbol{\psi})\|_{\boldsymbol{L}^2(Q)} \\ &\le c\, \|E_\Sigma \varphi\|_{H^{1,1/2}(Q)} \|E_\Sigma \boldsymbol{\psi}\|_{\boldsymbol{H}^{1,1/2}(Q)} \le c\, c_{\mathrm{IT}}^2 \|\varphi\|_{H^{1/2,1/4}(\Sigma)} \|\boldsymbol{\psi}\|_{\boldsymbol{H}^{1/2,1/4}(\Sigma)}, \end{aligned}$$

where c_{IT} denotes the boundedness constant of the extension operators E_Σ.

The only thing left to show is (4.5) for $\widetilde{\varphi} \in C^2(\overline{Q})$ and $\varphi = \widetilde{\varphi}|_\Sigma$. Since the definition of $\mathbf{curl}_\Sigma$ in (4.4) is independent of the extension of φ we can use the particular extension $\widetilde{\varphi}$ to get

$$\begin{aligned} \langle \mathbf{curl}_\Sigma \varphi, \boldsymbol{\psi} \rangle_\Sigma &= \langle \nabla \widetilde{\varphi}, \mathbf{curl}(E_\Sigma \boldsymbol{\psi}) \rangle_{\boldsymbol{L}^2(Q)} = \langle \nabla \widetilde{\varphi} \times \boldsymbol{n}, \boldsymbol{\psi} \rangle_\Sigma + \langle \mathbf{curl}(\nabla \widetilde{\varphi}), E_\Sigma \boldsymbol{\psi} \rangle_{\boldsymbol{L}^2(Q)} \\ &= \langle \nabla \widetilde{\varphi} \times \boldsymbol{n}, \boldsymbol{\psi} \rangle_\Sigma \end{aligned}$$

for all $\boldsymbol{\psi} \in \boldsymbol{H}^{1/2,1/4}(\Sigma)$, where we used integration by parts in the second step. This means that $\mathbf{curl}_\Sigma \varphi = \nabla \widetilde{\varphi} \times \boldsymbol{n}$ in $\boldsymbol{H}^{-1/2,-1/4}(\Sigma)$. □

4.1.3 The heat equation – selected results

In Section 3.2 we have seen how boundary integral equations can be used for the solution of initial boundary value problems for the transient heat equation. Here we present additional solvability results for an initial Dirichlet problem with homogeneous initial conditions. In addition, we discuss the classical parabolic maximum principle and a generalization of it, which we will need in Section 4.2.2.

The first theorem which we state is a classical existence and uniqueness result that can be found in a more general form in [7, Theorem 6.2.8].

THEOREM 4.4. *Let $g \in C(\overline{\Sigma})$ such that $g(\boldsymbol{x}, 0) = 0$ for all $\boldsymbol{x} \in \Gamma$. Then the initial Dirichlet boundary value problem (1.1)–(1.3) with initial datum $u_0 = 0$ admits a unique classical solution $u \in C(\overline{Q}) \cap C^\infty(Q)$.*

In Section 3.2 we have already seen that we can construct the solution of an initial Dirichlet boundary value problem with Dirichlet datum $g \in H^{1/2,1/4}(\Sigma)$ by means of integral operators. Theorem 4.5 shows that this is even possible for $g \in L^2(\Sigma)$ when interpreting boundary values in terms of non-tangential limits. The non-tangential limit of a function u in Q at $(\boldsymbol{x}, t) \in \Sigma$ is defined by

$$\lim_{\gamma(\boldsymbol{x},t) \ni (\boldsymbol{y},\tau) \to (\boldsymbol{x},t)} u(\boldsymbol{y}, \tau), \tag{4.7}$$

where $\gamma(\boldsymbol{x}, t) \subset Q$ is the so-called parabolic non-tangential approach region of a point $(\boldsymbol{x}, t) \in \Sigma$. Since a rigorous definition of this region $\gamma(\boldsymbol{x}, t)$ is slightly technical and we do not use it extensively here, we refer to [19, Section 1] for it.

THEOREM 4.5 (cf. [19, Theorems 8.1 and 8.3]). *The operator $(-1/2\,\mathrm{Id} + K)$ as a map from $L^2(\Sigma)$ to $L^2(\Sigma)$ is an isomorphism. In particular, for any $g \in L^2(\Sigma)$ the function*

$$u = W\Big(\Big(-\frac{1}{2}\mathrm{Id} + K\Big)^{-1} g\Big) \tag{4.8}$$

is well-defined. Furthermore, it is the unique function that satisfies:

(i) $u \in C^\infty(Q)$ *and* $(\partial/\partial t - \alpha\Delta)u = 0$.

(ii) $u \in C(\Omega \times [0, T))$ *and* $u(\cdot, 0) = 0$.

(iii) The non-tangential maximal function $N(u)$ of u is in $L^2(\Sigma)$, where

$$N(u)(\boldsymbol{x}, t) := \sup\{|u(\boldsymbol{y}, \tau)| : (\boldsymbol{y}, \tau) \in \gamma(\boldsymbol{x}, t)\} \quad \text{for all } (\boldsymbol{x}, t) \in \Sigma, \tag{4.9}$$

and $\gamma(\boldsymbol{x}, t) \subset Q$ is the parabolic non-tangential region of $(\boldsymbol{x}, t)$ as in (4.7).

(iv) $u = g$ on Σ in the sense of non-tangential limits almost everywhere.

The next result is known as the parabolic maximum principle; see e.g. Theorem 4 in Section 2.3.3 of [30].

PROPOSITION 4.6. *Let $u \in C(\overline{Q}) \cap C^2(Q)$ satisfy (1.1). Then*

$$\max_{(\boldsymbol{x},t)\in\overline{Q}} u(\boldsymbol{x},t) = \max_{(\boldsymbol{x},t)\in(\overline{\Sigma}\cup(\overline{\Omega}\times\{0\}))} u(\boldsymbol{x},t)$$

and the same holds if the maximum is replaced by the minimum on both sides.

We conclude this section with Theorem 4.7, which is a generalization of the maximum principle in Proposition 4.6 and will be required in the proof of Theorem 4.17.

THEOREM 4.7 (Extended parabolic maximum principle). *Let $g \in L^\infty(\Sigma)$ and u be the solution (4.8) to the initial Dirichlet boundary value problem (1.1)–(1.3) with the initial datum $u_0 = 0$. Then there holds*

$$\sup\{|u(\boldsymbol{x},t)| : (\boldsymbol{x},t) \in Q\} \le \|g\|_{L^\infty(\Sigma)}. \tag{4.10}$$

Proof. The theorem is proven in three steps. First we approximate the boundary datum g in $L^2(\Sigma)$ by a sequence $\{g_n\}_n$ in $C(\overline{\Sigma})$ such that $g_n(\boldsymbol{x},0) = 0$ for all $\boldsymbol{x} \in \Gamma$ and $\|g_n\|_{L^\infty(\Sigma)} \le \|g\|_{L^\infty(\Sigma)}$. Secondly, we construct solutions u_n of the homogeneous heat equation by Theorems 4.4 and 4.5 such that $u_n = g_n$ on Σ. Finally, we show assertion (4.10) by contradiction using Proposition 4.6 for u_n and a continuity argument.

We start with the construction of the sequence $\{g_n\}_n$. Due to the density of $C(\overline{\Sigma})$ in $L^2(\Sigma)$ we find a sequence $\{f_n\}_n$ in $C(\overline{\Sigma})$ such that $f_n \to g$ in $L^2(\Sigma)$ as $n \to \infty$. Let us first define $\widetilde{g}_n$ by

$$\widetilde{g}_n(\boldsymbol{x},t) := \begin{cases} f_n(\boldsymbol{x},t) & \text{if } (\boldsymbol{x},t) \text{ is such that } |f_n(\boldsymbol{x},t)| \le \|g\|_{L^\infty(\Sigma)}, \\ \operatorname{sign}(f_n(\boldsymbol{x},t))\|g\|_{L^\infty(\Sigma)} & \text{otherwise.} \end{cases}$$

One can show that $\widetilde{g}_n \in C(\overline{\Sigma})$ and $\|\widetilde{g}_n\|_{L^\infty(\Sigma)} \le \|g\|_{L^\infty(\Sigma)}$ for all $n \in \mathbb{N}$. In addition, $\widetilde{g}_n \to g$ in $L^2(\Sigma)$ as $n \to \infty$, which follows from the estimate

$$|\widetilde{g}_n(\boldsymbol{x},t) - g(\boldsymbol{x},t)| \le |f_n(\boldsymbol{x},t) - g(\boldsymbol{x},t)|$$

for all $n \in \mathbb{N}$ and almost all $(\boldsymbol{x},t) \in \Sigma$. Since $\widetilde{g}_n(\boldsymbol{x},0) = 0$ might be violated for some $\boldsymbol{x} \in \Gamma$ we set

$$g_n(\boldsymbol{x},t) := \begin{cases} n\,t\,\widetilde{g}_n(\boldsymbol{x},t) & \text{if } 0 \le t \le \frac{1}{n}, \\ \widetilde{g}_n(\boldsymbol{x},t) & \text{otherwise.} \end{cases}$$

By construction, there holds $g_n \in C(\overline{\Sigma})$, $g_n(\cdot,0) = 0$ on Γ and $\|g_n\|_{L^\infty(\Sigma)} \le \|g\|_{L^\infty(\Sigma)}$ for all $n \in \mathbb{N}$. In addition, $g_n - \widetilde{g}_n \to 0$ in $L^2(\Sigma)$ as $n \to \infty$, which implies the convergence of g_n to g in $L^2(\Sigma)$. Therefore, $\{g_n\}_n$ is a sequence in $C(\overline{\Sigma})$ with the desired properties.

By Theorem 4.4, we can find for each $n \in \mathbb{N}$ a unique $u_n \in C^\infty(Q) \cap C(\overline{Q})$ which solves the heat equation (1.1) and satisfies $u_n(\boldsymbol{x}, 0) = 0$ for all $\boldsymbol{x} \in \Omega$ as well as $u_n = g_n$ on Σ. Hence, u_n satisfies the properties *(i)*, *(ii)* and *(iv)* of Theorem 4.5 and we need to show only *(iii)* to obtain the representation

$$u_n = W\Big(\Big(-\frac{1}{2}\mathrm{Id} + K\Big)^{-1} g_n\Big). \tag{4.11}$$

Due to the definition of $N(u_n)$ in (4.9) and the inclusion $\gamma(\boldsymbol{x}, t) \subset Q$ there holds

$$N(u_n) \leq \sup\{|u_n(\boldsymbol{x}, t)| : (\boldsymbol{x}, t) \in Q\}.$$

From the classical parabolic maximum principle in Proposition 4.6 we can further estimate

$$\sup\{|u_n(\boldsymbol{x}, t)| : (\boldsymbol{x}, t) \in Q\} \leq \|g_n\|_{L^\infty(\Sigma)} \leq \|g\|_{L^\infty(\Sigma)}. \tag{4.12}$$

Therefore, $N(u_n) \leq \|g\|_{L^\infty(\Sigma)}$ on Σ and, in particular, $N(u_n) \in L^2(\Sigma)$, which is property *(iii)* in Theorem 4.5.

With the representation (4.11) we can show that $u_n \to u$ locally in Q in L^2 as $n \to \infty$. In fact, let $Q_\varepsilon := \{(\boldsymbol{x}, t) \in Q : \operatorname{dist}(\boldsymbol{x}, \Gamma) > \varepsilon\}$. Then the convergence of u_n to u in $L^2(Q_\varepsilon)$ follows immediately from $g_n \to g$ in $L^2(\Sigma)$, because the operator $(-\frac{1}{2}\mathrm{Id}+K)$ is an isomorphism in $L^2(\Sigma)$ as stated in Theorem 4.5, and because $W : L^2(\Sigma) \to L^2(Q_\varepsilon)$ is continuous, which is easy to see.

Suppose now that (4.10) does not hold true. Then there exists a point $(\boldsymbol{x}_0, t_0) \in Q$ such that $|u(\boldsymbol{x}_0, t_0)| > \|g\|_{L^\infty(\Sigma)}$. Since u is continuous in Q due to Theorem 4.5 *(ii)*, we can find some $\varepsilon > 0$, $\delta > 0$ and an open set $A \subset Q_\varepsilon$ with measure $|A| > 0$ such that $(\boldsymbol{x}_0, t_0) \in A$ and $|u(\boldsymbol{x}, t)| > \|g\|_{L^\infty(\Sigma)} + \delta$ for all $(\boldsymbol{x}, t) \in A$. Together with (4.12) it follows that

$$\iint_{Q_\varepsilon} |u_n(\boldsymbol{x}, t) - u(\boldsymbol{x}, t)|^2 \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \geq \iint_A |u_n(\boldsymbol{x}, t) - u(\boldsymbol{x}, t)|^2 \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t > \delta^2 |A|$$

for all $n \in \mathbb{N}$, which is a contradiction to $u_n \to u$ in $L^2(Q_\varepsilon)$. Therefore, (4.10) is satisfied. □

4.2 A general integration by parts formula for the hypersingular operator

In Theorem 4.8 we finally present a general form of the integration by parts formula for the bilinear form $\langle D\cdot, \cdot\rangle_\Sigma$ of the hypersingular operator. In the rest of the section we prove this theorem and discuss the formula in detail.

THEOREM 4.8. *Let $\Omega \subset \mathbb{R}^3$ be a bounded Lipschitz domain with boundary Γ and let $\Sigma = \Gamma \times (0,T)$. For $\varphi, \psi \in H^{1/2,1/4}(\Sigma)$ there holds the integration by parts formula*

$$\langle D\varphi, \psi\rangle_\Sigma = \alpha^2 \langle \mathbf{curl}_\Sigma\, \psi, V(\mathbf{curl}_\Sigma\, \varphi)\rangle_\Sigma + \alpha\, b(\varphi,\psi). \tag{4.13}$$

Here, the single layer boundary integral operator V is applied componentwise to $\mathbf{curl}_\Sigma\, \varphi$ and the bilinear form $b(\cdot,\cdot) : H^{1/2,1/4}(\Sigma) \times H^{1/2,1/4}(\Sigma) \to \mathbb{R}$ is defined by

$$b(\varphi,\psi) := \left(\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'\Big(V(\varphi \boldsymbol{n}) \cdot \boldsymbol{n}\Big)\right)[\widetilde{\psi}] := -\left\langle V(\varphi \boldsymbol{n}), \frac{\partial}{\partial t}\psi \boldsymbol{n}\right\rangle_\Sigma, \tag{4.14}$$

for $\varphi \in H^{1/2,1/4}(\Sigma)$, $\psi \in \gamma_{0,\Sigma}^{\mathrm{int}}(C_c^\infty(\mathbb{R}^3 \times (0,T)))$ and $\widetilde{\psi} \in C_c^\infty(\mathbb{R}^3 \times (0,T))$ such that $\psi = \widetilde{\psi}|_\Sigma$, and as its continuous extension for general $\psi \in H^{1/2,1/4}(\Sigma)$.

We will give a rigorous proof of this theorem in Section 4.2.1. A corresponding result for the 2D case has been given in [24, Theorem 6.1], including an outline of the proof. In that paper the bilinear form b is represented by $b(\varphi,\psi) = \langle \partial/\partial t\, V(\varphi \boldsymbol{n}), \psi \boldsymbol{n}\rangle_\Sigma$ and it is stated in the proof that it has to be interpreted in the sense of a continuous extension. For the 3D case a similar statement can be found in [27, Section 4.7] and [51, Section 3.1.3], but no proof is given. In addition, the latter authors formulate the result in a less rigorous way and do not clarify how the second term on the right-hand side of (4.13), which they represent as

$$-\alpha \int_0^T \int_\Gamma \psi(\boldsymbol{x},t)\boldsymbol{n}(\boldsymbol{x}) \cdot \int_0^t \int_\Gamma \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\, \mathrm{d}\tau\, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\, \mathrm{d}t, \tag{4.15}$$

has to be understood for general φ and ψ. In fact, we cannot interpret it as a standard Lebesgue integral, as we show next.

PROPOSITION 4.9. *The function*

$$((\boldsymbol{x},t),(\boldsymbol{y},\tau)) \mapsto \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau) \tag{4.16}$$

is not Lebesgue integrable on $\Sigma \times \Sigma$.

The following proof is valid for arbitrary Lipschitz boundaries Γ. For smooth boundaries Γ a more elegant proof is available, which is given in [77, see Proposition 5.1].

Proof. We start by computing the derivative

$$\frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau) = \left[\frac{6\alpha(t-\tau) - |\boldsymbol{x}-\boldsymbol{y}|^2}{(4\alpha)^{5/2}\pi^{3/2}(t-\tau)^{7/2}}\right] \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4\alpha(t-\tau)}\right) \qquad \text{for } t > \tau. \tag{4.17}$$

For fixed t, $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfying $6\alpha t > |\boldsymbol{x}-\boldsymbol{y}|^2$ and $\boldsymbol{x} \neq \boldsymbol{y}$, we observe that $\tau \mapsto \partial_\tau G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)$ is positive for $\tau < \tau^* := t - |\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha$ and negative for $\tau > \tau^*$. Thus, we get

$$\begin{aligned}\int_0^t \left|\frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\right| \mathrm{d}\tau &= \int_0^{\tau^*} \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\,\mathrm{d}\tau - \int_{\tau^*}^t \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\,\mathrm{d}\tau \\ &= 2G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau^*) - G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t),\end{aligned}$$

where $t - \tau^* = |\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha$ is independent of t and

$$G_\alpha\left(\boldsymbol{x}-\boldsymbol{y}, \frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{6\alpha}\right) = \frac{(6)^{3/2}\exp(-3/2)}{(4\pi)^{3/2}|\boldsymbol{x}-\boldsymbol{y}|^3}. \tag{4.18}$$

If instead $t \leq |\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha$, there holds

$$\int_0^t \left|\frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\right| \mathrm{d}\tau = G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t).$$

To simplify the arguments let us assume for a moment that $T > |\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha$ for all points $\boldsymbol{x}$ and $\boldsymbol{y}$ on Γ. Then we immediately see that

$$\begin{aligned}\int_0^T \int_0^t \left|\frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\right| \mathrm{d}\tau\,\mathrm{d}t &= \int_0^{|\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t)\,\mathrm{d}t \\ &\quad - \int_{|\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha}^T G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t)\,\mathrm{d}t + 2\left(T - \frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{6\alpha}\right) G_\alpha\left(\boldsymbol{x}-\boldsymbol{y}, \frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{6\alpha}\right)\end{aligned} \tag{4.19}$$

for all $\boldsymbol{x}$ and $\boldsymbol{y}$ in Γ with $\boldsymbol{x} \neq \boldsymbol{y}$. It remains to integrate (4.19) over $\Gamma \times \Gamma$. The first two terms are integrable on $\Gamma \times \Gamma$, which is a consequence of the estimate [59, cf. Chapter 13 §3]

$$\begin{aligned}|G_\alpha(\boldsymbol{r}, t)| &= \frac{1}{(4\pi\alpha t)^{3/2}} \exp\left(\frac{-|\boldsymbol{r}|^2}{4\alpha t}\right) = \left(\frac{|\boldsymbol{r}|^2}{4\alpha t}\right)^{3/4} \exp\left(\frac{-|\boldsymbol{r}|^2}{4\alpha t}\right) \frac{1}{\pi^{3/2}(4\alpha t)^{3/4}|\boldsymbol{r}|^{3/2}} \\ &\leq \left(\frac{3}{4}\right)^{3/4} \exp\left(-\frac{3}{4}\right) \frac{1}{\pi^{3/2}(4\alpha t)^{3/4}|\boldsymbol{r}|^{3/2}} = c(\alpha)\frac{1}{t^{3/4}}\frac{1}{|\boldsymbol{r}|^{3/2}}\end{aligned} \tag{4.20}$$

for $t > 0$, where we used that $q^m \exp(-q) \leq m^m \exp(-m)$ for all $m, q \geq 0$. The integral over the last term on the right-hand side of (4.19), however, is unbounded due to the strong singularity of (4.18). Therefore, (4.16) is not Lebesgue integrable on $\Sigma \times \Sigma$. In the general case, the estimate $T > |\boldsymbol{x}-\boldsymbol{y}|^2/6\alpha$ still holds on local parts of Γ which is enough to show the assertion. □

In this work, the problematic integral term (4.15) is replaced by $\alpha\, b(\varphi, \psi)$ in the integration by parts formula. Since we define b as the continuous extension of (4.14) it is a priori not clear, how to evaluate it for nonsmooth ψ. The reason is that neither $V(\varphi \boldsymbol{n})$ nor ψ does admit a weak derivative with respect to time in general, which is why the second term in (4.14) has to be understood in the stated distributional sense for smooth ψ as above. In Section 4.2.2 we present an alternative representation of b which is valid for functions with a certain tensor product structure and overcomes this deficiency.

4.2.1 A proof of the general integration by parts formula

The proof of Theorem 4.8 is split into three main steps, to each of which we dedicate a separate paragraph. In the first paragraph we derive a distributional representation of $\alpha \nabla W\varphi$. The steps in the second and third paragraphs are based on the ideas given in [24, Proof of Theorem 6.1]. We show an integration by parts formula on an auxiliary boundary inside of the space-time domain Q in the second paragraph, using the representation of $\alpha \nabla W\varphi$ from the first one. In the third paragraph we construct a sequence of auxiliary boundaries Σ_m inside of Q which approximate Σ and show that the integration by parts formula on Σ is obtained from the formulas on Σ_m in the limit as m tends to infinity. The actual proof of Theorem 4.8 is given at the end of the third paragraph.

Throughout this section we use the general definitions of the single layer potential operator

$$\widetilde{V} q = G_\alpha * \left((\gamma_{0,\Sigma}^{\mathrm{int}})' q \right) \tag{4.21}$$

for $q \in H^{-1/2,-1/4}(\Sigma)$ and the double layer potential operator

$$W\varphi = G_\alpha * \left((\alpha \gamma_{1,\Sigma}^{\mathrm{int}})' \varphi \right) \tag{4.22}$$

for $\varphi \in H^{1/2,1/4}(\Sigma)$, cf. [24, Section 3]. Here, $(\gamma_{0,\Sigma}^{\mathrm{int}})'$ and $(\gamma_{1,\Sigma}^{\mathrm{int}})'$ are the adjoint trace operators which we have introduced at the end of Section 4.1.1 and the convolution with the fundamental solution $(\boldsymbol{r}, t) \mapsto G_\alpha(\boldsymbol{r}, t)$ in (3.4) is understood as a convolution of distributions.

A distributional representation of the gradient of $W\varphi$

The hypersingular operator D is defined as the Neumann trace $\gamma_{1,\Sigma}^{\mathrm{int}}$ of the double layer potential operator W. Since $\gamma_{1,\Sigma}^{\mathrm{int}} = \boldsymbol{n} \cdot \nabla$ holds for smooth functions — see Proposition 2.10 — we are interested in the evaluation of $\nabla W\varphi$ for $\varphi \in H^{1/2,1/4}(\Sigma)$. In the following proposition we compute this gradient in a distributional sense. This approach is motivated by the proof of the integration by parts formula for elliptic operators in [63, Section 3.3.4].

PROPOSITION 4.10. *Let $\varphi \in H^{1/2,1/4}(\Sigma)$ and $u = W\varphi$ be defined by (4.22). Then there holds*

$$\alpha \nabla u = -G_\alpha * \left(\alpha \, \mathbf{curl}\, \mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}) + \frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}) \right) + (\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}) \quad (4.23)$$

in the distributional sense on $\mathbb{R}^3 \times \mathbb{R}$, where the convolution is understood componentwise. In particular, inside of Q there holds

$$\alpha \nabla u = -\left(\alpha \, \mathbf{curl}\, \mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})\right) * G_\alpha - \left(\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}) \right) * G_\alpha \quad (4.24)$$

in the classical sense, and the terms on the right-hand side can be understood as functions in $C^\infty(Q)$.

Proof. Let $\boldsymbol{v} \in \boldsymbol{C}_c^\infty(\mathbb{R}^3 \times \mathbb{R})$. Applying the distributional gradient ∇u to $\boldsymbol{v}$ yields

$$\begin{aligned} \alpha \nabla u\,[\boldsymbol{v}] &= -\alpha u[\operatorname{div} \boldsymbol{v}] = -\alpha \Big(G_\alpha * ((\alpha \gamma_{1,\Sigma}^{\mathrm{int}})'\varphi) \Big)[\operatorname{div} \boldsymbol{v}] \\ &= -\alpha (G_\alpha)_{(\boldsymbol{x},t)} \left[((\alpha \gamma_{1,\Sigma}^{\mathrm{int}})'\varphi)_{(\boldsymbol{y},\tau)} [\operatorname{div} \boldsymbol{v}(\cdot_{\boldsymbol{x}} + \cdot_{\boldsymbol{y}}, \cdot_t + \cdot_\tau)] \right], \end{aligned} \quad (4.25)$$

where we used the definition of the convolution of distributions in (4.1). With the definition of $(\gamma_{1,\Sigma}^{\mathrm{int}})'$ in (4.3) and $(\gamma_{0,\Sigma}^{\mathrm{int}})'$ in (4.2) it follows that

$$\begin{aligned} &((\alpha \gamma_{1,\Sigma}^{\mathrm{int}})'\varphi)_{(\boldsymbol{y},\tau)} [\operatorname{div} \boldsymbol{v}(\boldsymbol{x} + \cdot_{\boldsymbol{y}}, t + \cdot_\tau)] \\ &\qquad = \langle (\alpha \gamma_{1,\Sigma}^{\mathrm{int}})_{\boldsymbol{y},\tau} (\operatorname{div} \boldsymbol{v}(\boldsymbol{x} + \cdot_{\boldsymbol{y}}, t + \cdot_\tau)), \varphi \rangle_\Sigma \\ &\qquad = \int_0^T \int_\Gamma \varphi(\boldsymbol{y}, \tau)\, \alpha \boldsymbol{n}(\boldsymbol{y}) \cdot \nabla \operatorname{div} \boldsymbol{v}(\boldsymbol{x} + \boldsymbol{y}, t + \tau) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\tau \\ &\qquad = ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}))_{(\boldsymbol{y},\tau)} [\nabla \operatorname{div} \boldsymbol{v}(\boldsymbol{x} + \cdot_{\boldsymbol{y}}, t + \cdot_\tau)] \end{aligned}$$

for each $(\boldsymbol{x}, t) \in \mathbb{R}^3 \times \mathbb{R}$. By inserting this into (4.25) and using the identity $\nabla \operatorname{div} \boldsymbol{v} = \mathbf{curl}\, \mathbf{curl}\, \boldsymbol{v} + \Delta \boldsymbol{v}$, where Δ is applied componentwise to $\boldsymbol{v}$, we get

$$\begin{aligned} \alpha \nabla u\,[\boldsymbol{v}] &= -\alpha (G_\alpha)_{(\boldsymbol{x},t)} \left[((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}))_{(\boldsymbol{y},\tau)} [(\mathbf{curl}\, \mathbf{curl} + \Delta) \boldsymbol{v}(\cdot_{\boldsymbol{x}} + \cdot_{\boldsymbol{y}}, \cdot_t + \cdot_\tau)] \right] \\ &= -\alpha \Big(G_\alpha * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})) \Big)[\mathbf{curl}\, \mathbf{curl}\, \boldsymbol{v} + \Delta \boldsymbol{v}] \\ &= -\alpha \, \mathbf{curl}\, \mathbf{curl} \Big(G_\alpha * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})) \Big)[\boldsymbol{v}] - \alpha \Delta \Big(G_\alpha * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})) \Big)[\boldsymbol{v}]. \end{aligned}$$

Here and in the following the convolution of G_α and a vector-valued distribution like $(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})$ is understood componentwise. By differentiation rules for convolutions of distributions, which follow from the one stated in Proposition 4.1, it follows

$$\begin{aligned} -\alpha \, \mathbf{curl}\, \mathbf{curl} \Big(G_\alpha * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})) \Big)[\boldsymbol{v}] &= -\Big(G_\alpha * \Big(\alpha \, \mathbf{curl}\, \mathbf{curl} (\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n}) \Big) \Big)[\boldsymbol{v}], \\ -\alpha \Delta \Big(G_\alpha * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})) \Big)[\boldsymbol{v}] &= \Big((-\alpha \Delta G_\alpha) * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi \boldsymbol{n})) \Big)[\boldsymbol{v}]. \end{aligned}$$

Since G_α is the fundamental solution of the heat equation with heat capacity constant α, there holds $\partial_t G_\alpha - \alpha\Delta G_\alpha = \delta_{\mathbf{0}}$, where $\delta_{\mathbf{0}}$ denotes the delta distribution concentrated at $\mathbf{0}$. As a consequence, we can rewrite the second equation as

$$\begin{aligned}\Big((-\alpha\Delta G_\alpha) * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n}))\Big)[\boldsymbol{v}] &= \Big(\Big(-\frac{\partial}{\partial t}G_\alpha + \delta_{\mathbf{0}}\Big) * ((\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n}))\Big)[\boldsymbol{v}] \\ &= -\Big(G_\alpha * \Big(\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big)\Big)[\boldsymbol{v}] + (\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})[\boldsymbol{v}].\end{aligned}$$

Collecting all results, we see that

$$\begin{aligned}\alpha\nabla u\,[\boldsymbol{v}] = &-\Big(G_\alpha * \Big(\alpha\,\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n}) + \frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big)\Big)[\boldsymbol{v}] \\ &+ (\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})[\boldsymbol{v}].\end{aligned}$$

Since $\boldsymbol{v} \in C_c^\infty(\mathbb{R}^3 \times \mathbb{R})$ was arbitrary, we conclude that (4.23) holds.

Let us now interpret both sides of (4.23) as elements in $\mathcal{D}'(Q)$ by restricting the test functions to $\boldsymbol{C}_c^\infty(Q)$. Since the support of $(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})$ is a subset of $\overline{\Sigma}$ there holds

$$\alpha\nabla u\,[\boldsymbol{v}] = -\Big(G_\alpha * \Big(\alpha\,\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n}) + \frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big)\Big)[\boldsymbol{v}]$$

for all $\boldsymbol{v} \in \boldsymbol{C}_c^\infty(Q)$. The left-hand side can be interpreted as a regular distribution on Q induced by $\alpha\nabla u \in \boldsymbol{L}^2(Q)$, because $u \in H^{1,1/2}_{;0,}(Q)$. The first term on the right-hand side can be rewritten as

$$G_\alpha * \Big(\alpha\,\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big) = \Big(\alpha\,\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big) * G_\alpha.$$

Since the distribution $\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})$ has support in $\overline{\Sigma}$ and $(\boldsymbol{r},t) \mapsto G_\alpha(\boldsymbol{r},t)$ is in $C^\infty((\mathbb{R}^3 \times \mathbb{R})\backslash\{\mathbf{0}\})$, one can show that $(\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})) * G_\alpha$ is a regular distribution on Q and can be interpreted as a function in $\boldsymbol{C}^\infty(Q)$. The same arguments apply to

$$G_\alpha * \Big(\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big) = \Big(\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big) * G_\alpha. \qquad \square$$

For later reference we define $\boldsymbol{g}_1, \boldsymbol{g}_2 \in \boldsymbol{C}^\infty(Q)$ in agreement with Proposition 4.10 by

$$\boldsymbol{g}_1 := -\Big(\alpha\,\mathbf{curl}\,\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big) * G_\alpha, \tag{4.26}$$

$$\boldsymbol{g}_2 := -\Big(\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\alpha\varphi\boldsymbol{n})\Big) * G_\alpha. \tag{4.27}$$

Equation (4.24) can then be written in the short form

$$\alpha\nabla u = \boldsymbol{g}_1 + \boldsymbol{g}_2 \qquad \text{in } Q. \tag{4.28}$$

An integration by parts formula inside the space-time domain Q

The next major step in the proof of Theorem 4.8 is to show a result equivalent to the integration by parts formula (4.13) on an auxiliary boundary Σ_m inside of Q. This is formulated in Proposition 4.11. The approach is motivated by the smoothness of $u = W\varphi$ or rather $\alpha\nabla u$ inside of Q, which permits us to represent normal derivatives of u on Σ_m in a classical way. Furthermore, it allows us to interpret derivatives appearing in duality products $\langle\cdot,\cdot\rangle_{\Sigma_m}$ in a distributional sense, and thus to integrate by parts.

PROPOSITION 4.11. *Let Ω_m be a Lipschitz domain satisfying $\overline{\Omega}_m \subset \Omega$ with outward normal vector $\boldsymbol{n}_m$ and boundary Γ_m. Let $Q_m := \Omega_m \times (0,T)$ and $\Sigma_m := \Gamma_m \times (0,T)$. Let $\varphi \in H^{1/2,1/4}(\Sigma)$, $u = W\varphi$ and $\psi_m \in H^{1/2,1/4}(\Sigma_m)$. Then*

$$\begin{aligned} -\alpha \left\langle \gamma_{1,\Sigma_m}^{\mathrm{int}} u, \psi_m \right\rangle_{\Sigma_m} = {} & \alpha^2 \langle \mathbf{curl}_{\Sigma_m} \psi_m, \gamma_{0,\Sigma_m}^{\mathrm{int}} \widetilde{V}(\mathbf{curl}_\Sigma \varphi) \rangle_{\Sigma_m} \\ & + \alpha \left\langle \boldsymbol{n}_m \cdot \gamma_{0,\Sigma_m}^{\mathrm{int}} \left(\frac{\partial}{\partial t} \widetilde{V}(\varphi \boldsymbol{n}) \right), \psi_m \right\rangle_{\Sigma_m}. \end{aligned} \tag{4.29}$$

Before we prove Proposition 4.11, we consider two lemmata. Keeping in mind the decomposition (4.28), we focus first on $\boldsymbol{g}_1$ in (4.26). In Lemma 4.13 we show that it is related to the first term on the right-hand side of (4.29). For this purpose, we have to draw a connection between $\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n})$ and the surface curl of φ defined in (4.4). This is done in Lemma 4.12. A similar result for purely spatial curls is proven in [63, cf. Lemma 3.3.21].

LEMMA 4.12. *Let $\varphi \in H^{1/2,1/4}(\Sigma)$. Then*

$$\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n}) = (\gamma_{0,\Sigma}^{\mathrm{int}})'(\mathbf{curl}_\Sigma \varphi) \tag{4.30}$$

in $\mathcal{D}'(\mathbb{R}^3 \times \mathbb{R})$.

Proof. For all $\boldsymbol{w} \in \boldsymbol{C}_c^\infty(\mathbb{R}^3 \times \mathbb{R})$ there holds

$$\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n})[\boldsymbol{w}] = (\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n})[\mathbf{curl}\,\boldsymbol{w}] = \int_0^T \int_\Gamma \varphi(\boldsymbol{x},t)\boldsymbol{n}(\boldsymbol{x}) \cdot \mathbf{curl}\,\boldsymbol{w}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.$$

Let us first assume that $\varphi \in \gamma_{0,\Sigma}^{\mathrm{int}}(C_c^\infty(\overline{\Omega}\times(0,T)))$, i.e. there exists $\widetilde{\varphi} \in C_c^\infty(\overline{\Omega}\times(0,T))$ such that $\varphi = \widetilde{\varphi}|_\Sigma$. Then we can rewrite

$$\varphi(\boldsymbol{x},t)\,\mathbf{curl}\,\boldsymbol{w}(\boldsymbol{x},t) = \mathbf{curl}(\widetilde{\varphi}\boldsymbol{w})(\boldsymbol{x},t) - \nabla\widetilde{\varphi}(\boldsymbol{x},t) \times \boldsymbol{w}(\boldsymbol{x},t)$$

for all $\boldsymbol{x} \in \Gamma$ and $t \in (0,T)$. By inserting this into the previous equation we get

$$\int_0^T \int_\Gamma \boldsymbol{n}(\boldsymbol{x}) \cdot \mathbf{curl}(\widetilde{\varphi}\boldsymbol{w})(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t - \int_0^T \int_\Gamma \boldsymbol{n}(\boldsymbol{x}) \cdot (\nabla\widetilde{\varphi}(\boldsymbol{x},t) \times \boldsymbol{w}(\boldsymbol{x},t))\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.$$

The first integral vanishes, which follows by applying the divergence theorem and using the identity $\operatorname{div}\mathbf{curl}(\widetilde{\varphi}\boldsymbol{w}) = 0$. The integrand of the second integral is

$$\boldsymbol{n}(\boldsymbol{x}) \cdot (\nabla\widetilde{\varphi}(\boldsymbol{x},t) \times \boldsymbol{w}(\boldsymbol{x},t)) = -\boldsymbol{w}(\boldsymbol{x},t) \cdot (\nabla\widetilde{\varphi}(\boldsymbol{x},t) \times \boldsymbol{n}(\boldsymbol{x})) = -\boldsymbol{w}(\boldsymbol{x},t) \cdot (\mathbf{curl}_\Sigma\,\varphi),$$

where we used (4.5) in the last step. Hence,

$$\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n})[\boldsymbol{w}] = \int_0^T \int_\Gamma \boldsymbol{w}(\boldsymbol{x},t) \cdot (\mathbf{curl}_\Sigma\,\varphi)(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t = (\gamma_{0,\Sigma}^{\mathrm{int}})'(\mathbf{curl}_\Sigma\,\varphi)[\boldsymbol{w}],$$

and thus (4.30) holds for all $\varphi \in \gamma_{0,\Sigma}^{\mathrm{int}}(C_c^\infty(\overline{\Omega} \times (0,T)))$.

Let us now consider a general $\varphi \in H^{1/2,1/4}(\Sigma)$. The space $\gamma_{0,\Sigma}^{\mathrm{int}}(C_c^\infty(\overline{\Omega} \times (0,T)))$ is dense in $H^{1/2,1/4}(\Sigma)$, which follows from Proposition 2.5. Therefore, we can find a sequence $\{\varphi_k\}_k$ in $\gamma_{0,\Sigma}^{\mathrm{int}}(C_c^\infty(\overline{\Omega} \times (0,T)))$ such that $\varphi_k \to \varphi$ in $H^{1/2,1/4}(\Sigma)$ as $k \to \infty$. It follows that

$$\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi_k\boldsymbol{n}) \to \mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n})$$

in the distributional sense as $k \to \infty$. In fact, for all $\boldsymbol{w} \in \boldsymbol{C}_c^\infty(\mathbb{R}^3 \times \mathbb{R})$ there holds

$$\begin{aligned}\lim_{k\to\infty} \mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi_k\boldsymbol{n})[\boldsymbol{w}] &= \lim_{k\to\infty} \int_0^T \int_\Gamma \varphi_k(\boldsymbol{x},t)\boldsymbol{n}(\boldsymbol{x}) \cdot \mathbf{curl}\,\boldsymbol{w}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \\ &= \lim_{k\to\infty} \langle \varphi_k, \boldsymbol{n} \cdot \mathbf{curl}\,\boldsymbol{w}\rangle_\Sigma = \langle \varphi, \boldsymbol{n} \cdot \mathbf{curl}\,\boldsymbol{w}\rangle_\Sigma \\ &= \mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n})[\boldsymbol{w}].\end{aligned}$$

At the same time

$$\begin{aligned}\lim_{k\to\infty} \mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi_k\boldsymbol{n})[\boldsymbol{w}] &= \lim_{k\to\infty} (\gamma_{0,\Sigma}^{\mathrm{int}})'(\mathbf{curl}_\Sigma\,\varphi_k)[\boldsymbol{w}] = \lim_{k\to\infty} \langle \mathbf{curl}_\Sigma\,\varphi_k, \gamma_{0,\Sigma}^{\mathrm{int}}\boldsymbol{w}\rangle_\Sigma \\ &= \langle \mathbf{curl}_\Sigma\,\varphi, \gamma_{0,\Sigma}^{\mathrm{int}}\boldsymbol{w}\rangle_\Sigma = (\gamma_{0,\Sigma}^{\mathrm{int}})'(\mathbf{curl}_\Sigma\,\varphi)[\boldsymbol{w}]\end{aligned}$$

for all $\boldsymbol{w} \in \boldsymbol{C}_c^\infty(\mathbb{R}^3 \times \mathbb{R})$ due to the continuity of $\mathbf{curl}_\Sigma$ in $H^{1/2,1/4}(\Sigma)$. Hence,

$$\mathbf{curl}(\gamma_{0,\Sigma}^{\mathrm{int}})'(\varphi\boldsymbol{n}) = (\gamma_{0,\Sigma}^{\mathrm{int}})'(\mathbf{curl}_\Sigma\,\varphi)$$

holds for general $\varphi \in H^{1/2,1/4}(\Sigma)$. □

LEMMA 4.13. *Let Σ_m be given as in Proposition 4.11 and $\boldsymbol{g}_1$ be defined by (4.26). Let $\varphi \in H^{1/2,1/4}(\Sigma)$ and $\psi_m \in H^{1/2,1/4}(\Sigma_m)$. Then*

$$-\langle \boldsymbol{n}_m \cdot \gamma_{0,\Sigma_m}^{\mathrm{int}}\boldsymbol{g}_1, \psi_m\rangle_{\Sigma_m} = \alpha^2 \langle \mathbf{curl}_{\Sigma_m}\,\psi_m, \gamma_{0,\Sigma_m}^{\mathrm{int}}\widetilde{V}(\mathbf{curl}_\Sigma\,\varphi)\rangle_{\Sigma_m}. \tag{4.31}$$

Proof. Since $\boldsymbol{g}_1 \in \boldsymbol{C}^\infty(Q)$ we can interpret

$$\begin{aligned} -\langle \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m} \boldsymbol{g}_1, \psi_m \rangle_{\Sigma_m} &= -\int_0^T \int_{\Gamma_m} \psi_m(\boldsymbol{x},t) \boldsymbol{n}_m(\boldsymbol{x}) \cdot \boldsymbol{g}_1(\boldsymbol{x},t) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \, \mathrm{d}t \\ &= -(\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\psi_m \boldsymbol{n}_m)[\boldsymbol{g}_1], \end{aligned}$$

with $(\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\psi_m \boldsymbol{n}_m) \in \mathcal{E}'(Q)$. By (4.26) we have

$$\boldsymbol{g}_1 = -\left(\alpha \, \mathbf{curl}\, \mathbf{curl}(\gamma^{\mathrm{int}}_{0,\Sigma})'(\alpha\varphi\boldsymbol{n})\right) * G_\alpha = -\alpha^2 \, \mathbf{curl}\left(\left(\mathbf{curl}(\gamma^{\mathrm{int}}_{0,\Sigma})'(\varphi\boldsymbol{n})\right) * G_\alpha\right)$$

and we see that

$$\begin{aligned} -(\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\psi_m \boldsymbol{n}_m)[\boldsymbol{g}_1] &= \alpha^2 (\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\psi_m \boldsymbol{n}_m) \left[\mathbf{curl}\left(\left(\mathbf{curl}(\gamma^{\mathrm{int}}_{0,\Sigma})'(\varphi\boldsymbol{n})\right) * G_\alpha\right)\right] \\ &= \alpha^2 \, \mathbf{curl} (\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\psi_m \boldsymbol{n}_m) \left[\left(\mathbf{curl}(\gamma^{\mathrm{int}}_{0,\Sigma})'(\varphi\boldsymbol{n})\right) * G_\alpha\right], \end{aligned}$$

i.e. we can integrate by parts in a distributional sense in the duality pairing of $\mathcal{E}'(Q)$ and $C^\infty(Q)$. Identity (4.30), which obviously still holds if Σ is replaced by Σ_m, combined with the previous equations yields

$$\begin{aligned} -\langle \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m} \boldsymbol{g}_1, \psi_m \rangle_{\Sigma_m} &= \alpha^2 (\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\mathbf{curl}_{\Sigma_m} \psi_m) \left[\left((\gamma^{\mathrm{int}}_{0,\Sigma})'(\mathbf{curl}_\Sigma \varphi)\right) * G_\alpha\right] \\ &= \alpha^2 (\gamma^{\mathrm{int}}_{0,\Sigma_m})'(\mathbf{curl}_{\Sigma_m} \psi_m) \left[\widetilde{V}(\mathbf{curl}_\Sigma \varphi)\right] \\ &= \alpha^2 \langle \mathbf{curl}_{\Sigma_m} \psi_m, \gamma^{\mathrm{int}}_{0,\Sigma_m} \widetilde{V}(\mathbf{curl}_\Sigma \varphi) \rangle_{\Sigma_m}, \end{aligned}$$

where we used the definition of $\widetilde{V}$ in (4.21) in the second line and understand its application here in a componentwise way. □

Proof of Proposition 4.11. We have seen in (4.28) that the scaled gradient $\alpha\nabla u$ can be written as the sum of the functions $\boldsymbol{g}_1, \boldsymbol{g}_2 \in C^\infty(Q)$ given in (4.26) and (4.27). As a consequence, the scaled Neumann trace $\alpha\gamma^{\mathrm{int}}_{1,\Sigma_m} u$ on the boundary Σ_m inside of Q is simply given by

$$\alpha\gamma^{\mathrm{int}}_{1,\Sigma_m} u = \alpha \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m}(\nabla u) = \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m} \boldsymbol{g}_1 + \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m} \boldsymbol{g}_2$$

and, hence,

$$-\alpha \, \langle \gamma^{\mathrm{int}}_{1,\Sigma_m} u, \psi_m \rangle_{\Sigma_m} = -\langle \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m} \boldsymbol{g}_1, \psi_m \rangle_{\Sigma_m} - \langle \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m} \boldsymbol{g}_2, \psi_m \rangle_{\Sigma_m}$$

for all $\psi_m \in H^{1/2,1/4}(\Sigma_m)$. By Equation (4.31) the first term on the right-hand side of this equation coincides with the first term on the right-hand side of (4.29). Since

$$\boldsymbol{g}_2 = -\left(\frac{\partial}{\partial t}(\gamma^{\mathrm{int}}_{0,\Sigma})'(\alpha\varphi\boldsymbol{n})\right) * G_\alpha = -\alpha \frac{\partial}{\partial t}\left(((\gamma^{\mathrm{int}}_{0,\Sigma})'(\varphi\boldsymbol{n})) * G_\alpha\right) = -\alpha \frac{\partial}{\partial t} \widetilde{V}(\varphi\boldsymbol{n}),$$

where $\widetilde{V}$ from (4.21) is applied componentwise again, the equality of the other terms follows immediately. □

The integration by parts formula (4.13) **as the limit case of** (4.29)

Proposition 4.11 provides us with an integration by parts formula (4.29) on artificial boundaries Σ_m inside of Q. The final step in the proof of Theorem 4.8 is to deduce the actual integration by parts formula (4.13) therefrom. For this purpose, we consider a sequence $\{\Omega_m\}_m$ of smooth domains approximating Ω as established by Theorem 2.3 and denote $Q_m := \Omega_m \times (0,T)$ and $\Sigma_m := \Gamma_m \times (0,T)$ as before. We will refer to $\{\Sigma_m\}_m$ in the following as a *smooth approximating sequence* of Σ. Let $\varphi, \psi \in H^{1/2,1/4}(\Sigma)$ and define $\psi_m \in H^{1/2,1/4}(\Sigma_m)$ by

$$\psi_m := \gamma^{\mathrm{int}}_{0,\Sigma_m}(E_{\Sigma;,0}\psi). \tag{4.32}$$

Then, the left-hand side and the first term on the right-hand side of (4.29) converge to the respective terms of (4.13) in the limit as m tends to infinity. This is the content of the next two lemmata. For the convergence of the remaining term, which is handled in Lemma 4.16, additional assumptions on ψ are required.

LEMMA 4.14. *Let $\{\Sigma_m\}_m$ be a smooth approximating sequence of Σ as introduced at the beginning of the current paragraph. Let $\varphi, \psi \in H^{1/2,1/4}(\Sigma)$, $u = W\varphi$, and ψ_m be defined by* (4.32)*. Then*

$$\lim_{m\to\infty} -\alpha \, \langle \gamma^{\mathrm{int}}_{1,\Sigma_m} u, \psi_m \rangle_{\Sigma_m} = \langle D\varphi, \psi \rangle_\Sigma. \tag{4.33}$$

Proof. By the definition of $\gamma^{\mathrm{int}}_{1,\Sigma}$ via Green's first identity in (2.10) there holds

$$\langle D\varphi, \psi \rangle_\Sigma = -\alpha \langle \gamma^{\mathrm{int}}_{1,\Sigma} u, \psi \rangle_\Sigma = -\alpha \left(\int_0^T \int_\Omega \nabla u \cdot \nabla (E_{\Sigma;,0}\psi) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t + d(u, E_{\Sigma;,0}\psi) \right),$$

where we use that $(\partial_t - \alpha\Delta)u = 0$, since $u = W\varphi$. The Neumann trace on Σ_m is defined analogously, and thus

$$-\alpha \, \langle \gamma^{\mathrm{int}}_{1,\Sigma_m} u, \psi_m \rangle_{\Sigma_m} = -\alpha \left(\int_0^T \int_{\Omega_m} \nabla u \cdot \nabla (E_{\Sigma;,0}\psi) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t + d_{Q_m}(u, E_{\Sigma;,0}\psi) \right),$$

where the bilinear form $d_{Q_m} : H^{1,1/2}_{;0,}(Q_m) \times H^{1,1/2}_{;,0}(Q_m) \to \mathbb{R}$ is defined in the same way as the bilinear form d in (2.7) for Ω_m instead of Ω, and we use that $E_{\Sigma;,0}\psi$ is an extension of ψ_m into $H^{1,1/2}_{;,0}(Q_m)$. Here and in the following we identify u and $E_{\Sigma;,0}\psi$ with the restrictions $u|_{Q_m}$ and $(E_{\Sigma;,0}\psi)|_{Q_m}$, respectively, when operating on Q_m to simplify the notation. By subtracting the second equation from the first one we can estimate

$$\begin{aligned} &\left| \langle D\varphi, \psi \rangle_\Sigma + \alpha \, \langle \gamma^{\mathrm{int}}_{1,\Sigma_m} u, \psi_m \rangle_{\Sigma_m} \right| \\ &\qquad \le \alpha \left| \int_0^T \int_{\Omega \setminus \Omega_m} \nabla u \cdot \nabla (E_{\Sigma;,0}\psi) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t \right| + \alpha \left| d(u, E_{\Sigma;,0}\psi) - d_{Q_m}(u, E_{\Sigma;,0}\psi) \right|. \end{aligned} \tag{4.34}$$

The first term on the right-hand side converges to 0 as $m \to \infty$. This follows from the Cauchy–Schwarz inequality, which we can use since ∇u, $\nabla E_{\Sigma;,0}\psi \in \boldsymbol{L}^2(Q)$ because u, $E_{\Sigma;,0}\psi \in H^{1,1/2}(Q)$, and $|\Omega \backslash \Omega_m| \to 0$.

It is slightly more difficult to see that the second term of the right-hand side in (4.34) converges to zero. The problem is that d and d_{Q_m} are defined only as continuous extensions of (2.7) for general functions in $H^{1,1/2}_{;0,}(Q)$ and $H^{1,1/2}_{;,0}(Q)$. Therefore, let $\{u_n\}_n$ be a sequence of functions in $C_c^\infty(\overline{\Omega} \times (0,T])$ converging to u in $H^{1,1/2}_{;0,}(Q)$ and $\{v_n\}_n$ be a sequence in $C_c^\infty(\overline{\Omega} \times [0,T))$ converging to $v := E_{\Sigma;,0}\psi$ in $H^{1,1/2}_{;,0}(Q)$. The restrictions $u_n|_{Q_m} \in C_c^\infty(\overline{\Omega}_m \times (0,T])$ and $v_n|_{\Omega_m} \in C_c^\infty(\overline{\Omega}_m \times [0,T))$ of these functions converge to $u|_{Q_m}$ in $H^{1,1/2}_{;0,}(Q_m)$ and $v|_{Q_m}$ in $H^{1,1/2}_{;,0}(Q_m)$, respectively. In particular,

$$\lim_{n\to\infty} d(u_n, v_n) = d(u,v), \qquad \lim_{n\to\infty} d_{Q_m}(u_n, v_n) = d_{Q_m}(u,v),$$

due to the continuity of d and d_{Q_m} stated in Proposition 2.9. This motivates us to estimate

$$\begin{aligned} |d(u,v) - d_{Q_m}(u,v)| \le{}& |d(u,v) - d(u_n,v_n)| + |d(u_n,v_n) - d_{Q_m}(u_n,v_n)| \\ &+ |d_{Q_m}(u_n,v_n) - d_{Q_m}(u,v)| \end{aligned} \tag{4.35}$$

and to show that the right-hand side can be bounded by an arbitrarily small ε for a suitably chosen n and sufficiently large m.

Let $\varepsilon > 0$ be fixed. The last summand in (4.35) can be estimated by

$$\begin{aligned} |d_{Q_m}(u_n,v_n) - d_{Q_m}(u,v)| &\le |d_{Q_m}(u_n,v_n) - d_{Q_m}(u,v_n)| + |d_{Q_m}(u,v_n) - d_{Q_m}(u,v)| \\ &\le c_d \|u-u_n\|_{H^{1,1/2}_{;0,}(Q_m)} \|v_n\|_{H^{1,1/2}_{;,0}(Q_m)} + c_d \|u\|_{H^{1,1/2}_{;0,}(Q_m)} \|v-v_n\|_{H^{1,1/2}_{;,0}(Q_m)} \\ &\le c_d \Big(\sup_{n\in\mathbb{N}} \Big\{ \|v_n\|_{H^{1,1/2}_{;,0}(Q)} \Big\} + \|u\|_{H^{1,1/2}_{;0,}(Q)} \Big) \Big(\|u-u_n\|_{H^{1,1/2}_{;0,}(Q)} + \|v-v_n\|_{H^{1,1/2}_{;,0}(Q)} \Big). \end{aligned}$$

Here we used that the Sobolev norms of functions restricted to Q_m can be estimated by the respective norms on Q and that the bilinear forms d_{Q_m} can be bounded by a constant c_d independent of the domains Ω_m, as stated in Proposition 2.9. Due to the convergence of u_n to u and v_n to v there exists an $n(\varepsilon)$ independent of m such that

$$\max\Big\{ \|u-u_n\|_{H^{1,1/2}_{;0,}(Q)}, \|v-v_n\|_{H^{1,1/2}_{;,0}(Q)} \Big\} < \frac{\varepsilon}{6c_d} \Big(\sup_{n\in\mathbb{N}} \Big\{ \|v_n\|_{H^{1,1/2}_{;,0}(Q)} \Big\} + \|u\|_{H^{1,1/2}_{;0,}(Q)} \Big)^{-1}$$

for all $n > n(\varepsilon)$. For all such n we conclude that

$$|d_{Q_m}(u_n,v_n) - d_{Q_m}(u,v)| < \frac{\varepsilon}{3}.$$

By repeating the same arguments for the first term in (4.35) it follows that

$$|d(u,v) - d(u_n,v_n)| < \frac{\varepsilon}{3}$$

for all $n > n(\varepsilon)$, with the same $n(\varepsilon)$ as before. The second summand in (4.35) is given by

$$|d(u_n,v_n) - d_{Q_m}(u_n,v_n)| = \left| \int_0^T \int_{\Omega\setminus\Omega_m} \frac{\partial u_n}{\partial t}(\boldsymbol{x},t) v_n(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \right|.$$

Note that we can use the explicit representation (2.7) of d and d_{Q_m} since u_n and v_n are smooth. The convergence of $|\Omega\setminus\Omega_m|$ to zero as m tends to infinity allows us to find an $m(\varepsilon,n)$ such that

$$|d(u_n,v_n) - d_{Q_m}(u_n,v_n)| < \frac{\varepsilon}{3}$$

for all $m > m(\varepsilon,n)$. Hence, for a fixed $n > n(\varepsilon)$ and all $m > m(\varepsilon,n)$ we can bound the right-hand side of (4.35) by ε. Therefore, both terms on the right-hand side of (4.34) converge to zero as $m \to \infty$. □

LEMMA 4.15. *Let $\{\Sigma_m\}_m$ be a smooth approximating sequence of Σ as introduced at the beginning of the current paragraph. Let $\varphi, \psi \in H^{1/2,1/4}(\Sigma)$ and ψ_m be defined by* (4.32)*. Then*

$$\lim_{m\to\infty} \langle \mathbf{curl}_{\Sigma_m} \psi_m, \gamma_{0,\Sigma_m}^{\mathrm{int}} \widetilde{V}(\mathbf{curl}_\Sigma\, \varphi)\rangle_{\Sigma_m} = \langle \mathbf{curl}_\Sigma\, \psi, V(\mathbf{curl}_\Sigma\, \varphi)\rangle_\Sigma. \tag{4.36}$$

Proof. By the definition of the surface curl in (4.4) there holds

$$\begin{aligned}\langle \mathbf{curl}_\Sigma\, \psi, V(\mathbf{curl}_\Sigma\, \varphi)\rangle_\Sigma &= \langle \nabla E_{\Sigma;,0}\psi, \mathbf{curl}(\widetilde{V}(\mathbf{curl}_\Sigma\, \varphi))\rangle_{\boldsymbol{L}^2(Q)} \\ &= \int_0^T \int_\Omega (\nabla E_{\Sigma;,0}\psi)(\boldsymbol{x},t) \cdot \mathbf{curl}(\widetilde{V}(\mathbf{curl}_\Sigma\, \varphi))(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t,\end{aligned}$$

where we used that $\widetilde{V}(\mathbf{curl}_\Sigma\, \varphi)$ is an extension of $V(\mathbf{curl}_\Sigma\, \varphi)$ and that the definition in (4.4) is independent of the extensions of the function ψ and the test function; see Proposition 4.3. Similarly, it follows that

$$\begin{aligned}&\langle \mathbf{curl}_{\Sigma_m} \psi_m, \gamma_{0,\Sigma_m}^{\mathrm{int}} \widetilde{V}(\,\mathbf{curl}_\Sigma\, \varphi)\rangle_{\Sigma_m} \\ &\qquad = \int_0^T \int_{\Omega_m} (\nabla E_{\Sigma;,0}\psi)(\boldsymbol{x},t) \cdot \mathbf{curl}(\widetilde{V}(\mathbf{curl}_\Sigma\, \varphi))(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t,\end{aligned}$$

by using in addition, that $(E_{\Sigma;,0}\psi)|_{Q_m}$ is an extension of ψ_m due to its definition in (4.32). Therefore,

$$\begin{aligned}&\left| \langle \mathbf{curl}_\Sigma\, \psi, V(\mathbf{curl}_\Sigma\, \varphi)\rangle_\Sigma - \langle \mathbf{curl}_{\Sigma_m} \psi_m, \gamma_{0,\Sigma_m}^{\mathrm{int}} \widetilde{V}(\mathbf{curl}_\Sigma\, \varphi)\rangle_{\Sigma_m} \right| \\ &\qquad = \left| \int_0^T \int_{\Omega\setminus\Omega_m} (\nabla E_{\Sigma;,0}\psi)(\boldsymbol{x},t) \cdot \mathbf{curl}(\widetilde{V}(\mathbf{curl}_\Sigma\, \varphi))(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \right|.\end{aligned}$$

Note that $(\nabla E_{\Sigma;,0}\psi)$ and $\mathbf{curl}(\widetilde{V}(\mathbf{curl}_\Sigma\,\varphi))$ are in $\boldsymbol{L}^2(Q)$, because $\widetilde{V}(\mathbf{curl}_\Sigma\,\varphi)$ is in $\boldsymbol{H}^{1,1/2}_{;0,}(Q)$. Hence, the right-hand side in the last equation converges to zero as $m\to\infty$ because $|\Omega\backslash\Omega_m|\to 0$. □

To show a similar convergence result for the second term of the right-hand side in (4.29) we need to require the test function ψ to be more regular. The following result holds.

LEMMA 4.16. *Let $\psi\in\gamma^{\rm int}_{0,\Sigma}(C_c^\infty(\overline{\Omega}\times(0,T)))$ and $\widetilde{\psi}\in C_c^\infty(\mathbb{R}^3\times(0,T))$ be such that $\psi=\widetilde{\psi}|_\Sigma$. Let $\varphi\in H^{1/2,1/4}(\Sigma)$. Let $\{\Sigma_m\}_m$ be a smooth approximating sequence of Σ as introduced at the beginning of the current paragraph and $\psi_m:=\widetilde{\psi}|_{\Sigma_m}$. Then*

$$\lim_{m\to\infty}\Big\langle \boldsymbol{n}_m\cdot\gamma^{\rm int}_{0,\Sigma_m}\Big(\frac{\partial}{\partial t}\widetilde{V}(\varphi\boldsymbol{n})\Big),\psi_m\Big\rangle_{\Sigma_m}=b(\varphi,\psi), \tag{4.37}$$

with the bilinear form b defined in (4.14).

Proof. We start by showing the identity

$$\Big\langle \boldsymbol{n}_m\cdot\gamma^{\rm int}_{0,\Sigma_m}\Big(\frac{\partial}{\partial t}\widetilde{V}(\varphi\boldsymbol{n})\Big),\psi_m\Big\rangle_{\Sigma_m}=\frac{\partial}{\partial t}(\gamma^{\rm int}_{0,\Sigma_m})'\Big(\gamma^{\rm int}_{0,\Sigma_m}(\widetilde{V}(\varphi\boldsymbol{n}))\cdot\boldsymbol{n}_m\Big)[\widetilde{\psi}], \tag{4.38}$$

where the right-hand side is understood as the application of a distribution on $\mathbb{R}^3\times(0,T)$ to the test function $\widetilde{\psi}\in C_c^\infty(\mathbb{R}^3\times(0,T))$. For the duality product on the left-hand side there holds

$$\begin{aligned}\Big\langle \boldsymbol{n}_m\cdot\gamma^{\rm int}_{0,\Sigma_m}\Big(\frac{\partial}{\partial t}\widetilde{V}(\varphi\boldsymbol{n})\Big),\psi_m\Big\rangle_{\Sigma_m}&=\int_0^T\int_{\Gamma_m}\frac{\partial}{\partial t}(\widetilde{V}(\varphi\boldsymbol{n}))(\boldsymbol{x},t)\cdot\boldsymbol{n}_m(\boldsymbol{x})\widetilde{\psi}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\\&=-\int_0^T\int_{\Gamma_m}\widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t)\cdot\boldsymbol{n}_m(\boldsymbol{x})\frac{\partial\widetilde{\psi}}{\partial t}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t,\end{aligned}$$

where we used classical integration by parts in the second step, which is possible since $\widetilde{V}(\varphi\boldsymbol{n})\in C^\infty(Q)$, $\widetilde{\psi}\in C_c^\infty(\mathbb{R}^3\times(0,T))$ and $\Sigma_m\subset Q$. The right-hand side of this equation corresponds to the duality product

$$-(\gamma^{\rm int}_{0,\Sigma_m})'\Big(\gamma^{\rm int}_{0,\Sigma_m}(\widetilde{V}(\varphi\boldsymbol{n}))\cdot\boldsymbol{n}_m\Big)\Big[\frac{\partial\widetilde{\psi}}{\partial t}\Big]=\frac{\partial}{\partial t}(\gamma^{\rm int}_{0,\Sigma_m})'\Big(\gamma^{\rm int}_{0,\Sigma_m}(\widetilde{V}(\varphi\boldsymbol{n}))\cdot\boldsymbol{n}_m\Big)[\widetilde{\psi}]$$

on $\mathcal{D}'(\mathbb{R}^3\times(0,T))\times C_c^\infty(\mathbb{R}^3\times(0,T))$, where the last equality is just the definition of the distributional time derivative. Therefore, (4.38) holds true.

Due to (4.38) the convergence in (4.37) follows if we can show that

$$\frac{\partial}{\partial t}(\gamma^{\rm int}_{0,\Sigma_m})'\Big(\gamma^{\rm int}_{0,\Sigma_m}(\widetilde{V}(\varphi\boldsymbol{n}))\cdot\boldsymbol{n}_m\Big)\to\frac{\partial}{\partial t}(\gamma^{\rm int}_{0,\Sigma})'\Big(V(\varphi\boldsymbol{n})\cdot\boldsymbol{n}\Big)\quad\text{in }\mathcal{D}'(\mathbb{R}^3\times(0,T)). \tag{4.39}$$

For this purpose, let $w \in C_c^\infty(\mathbb{R}^3 \times (0,T))$. Then

$$\begin{aligned}\Big|\frac{\partial}{\partial t}(\gamma_{0,\Sigma}^{\mathrm{int}})'\Big(V(\varphi\boldsymbol{n})\cdot\boldsymbol{n}\Big)[w] - \frac{\partial}{\partial t}(\gamma_{0,\Sigma_m}^{\mathrm{int}})'\Big(\gamma_{0,\Sigma_m}^{\mathrm{int}}(\widetilde{V}(\varphi\boldsymbol{n}))\cdot\boldsymbol{n}_m\Big)[w]\Big| \\ = \Big|\int_0^T\int_\Gamma V(\varphi\boldsymbol{n})(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x})\frac{\partial w}{\partial t}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \\ - \int_0^T\int_{\Gamma_m}\widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t)\cdot\boldsymbol{n}_m(\boldsymbol{x})\frac{\partial w}{\partial t}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\Big|.\end{aligned}$$

Since the product of $\partial w/\partial t$ and $\widetilde{V}(\varphi\boldsymbol{n})$ is in $L^2(0,T;\boldsymbol{H}^1(\Omega))$ we can apply the divergence theorem to both surface integrals in the previous equation and get

$$\begin{aligned}&\Big|\int_0^T\int_\Omega\Big(\operatorname{div}(\widetilde{V}(\varphi\boldsymbol{n}))(\boldsymbol{x},t)\frac{\partial w}{\partial t}(\boldsymbol{x},t) + \widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t)\cdot\nabla\Big(\frac{\partial w}{\partial t}\Big)(\boldsymbol{x},t)\Big)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\ &\quad - \int_0^T\int_{\Omega_m}\Big(\operatorname{div}(\widetilde{V}(\varphi\boldsymbol{n}))(\boldsymbol{x},t)\frac{\partial w}{\partial t}(\boldsymbol{x},t) + \widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t)\cdot\nabla\Big(\frac{\partial w}{\partial t}\Big)(\boldsymbol{x},t)\Big)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t\Big| \\ &\le \Big|\int_0^T\int_{\Omega\setminus\Omega_m}\operatorname{div}(\widetilde{V}(\varphi\boldsymbol{n}))(\boldsymbol{x},t)\frac{\partial w}{\partial t}(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t\Big| \\ &\qquad + \Big|\int_0^T\int_{\Omega\setminus\Omega_m}\widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t)\cdot\nabla\Big(\frac{\partial w}{\partial t}\Big)(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t\Big| \\ &\le \|\operatorname{div}(\widetilde{V}(\varphi\boldsymbol{n}))\|_{L^2(Q\setminus Q_m)}\Big\|\frac{\partial w}{\partial t}\Big\|_{L^2(Q\setminus Q_m)} + \|\widetilde{V}(\varphi\boldsymbol{n})\|_{\boldsymbol{L}^2(Q\setminus Q_m)}\Big\|\nabla\Big(\frac{\partial w}{\partial t}\Big)\Big\|_{\boldsymbol{L}^2(Q\setminus Q_m)}.\end{aligned}$$

The norms in the last line are bounded for all m and converge to 0 as $m \to \infty$, since $|Q\setminus Q_m| \to 0$. Therefore, we have established the convergence in (4.39) and thus in (4.37). □

Finally, we are ready to give a proof of Theorem 4.8 by collecting all the results.

Proof of Theorem 4.8. Let $\{\Sigma_m\}_m$ be a smooth approximating sequence of Σ as introduced at the beginning of the current paragraph. As in Lemma 4.16 we assume first that $\psi \in \gamma_{0,\Sigma}^{\mathrm{int}}(C_c^\infty(\overline{\Omega}\times(0,T)))$, i.e. $\psi = \widetilde{\psi}|_\Sigma$ for some $\widetilde{\psi} \in C_c^\infty(\mathbb{R}^3\times(0,T))$, and denote $\psi_m := \widetilde{\psi}|_{\Sigma_m}$. On each boundary Σ_m the integration by parts formula (4.29) from Proposition 4.11 is valid for φ and ψ_m. By applying Lemmata 4.14–4.16 we see that the limit of (4.29) as $m \to \infty$ is given by

$$\langle D\varphi,\psi\rangle_\Sigma = \alpha^2\langle\mathbf{curl}_\Sigma\,\psi, V(\mathbf{curl}_\Sigma\,\varphi)\rangle_\Sigma + \alpha\,b(\varphi,\psi),$$

which is the desired integration by parts formula (4.13) on Σ.

It remains to show that (4.13) holds also for general $\psi \in H^{1/2,1/4}(\Sigma)$ in the appropriate sense. By reordering the terms in the integration by parts formula we get

$$\alpha\, b(\varphi,\psi) = \langle D\varphi,\psi\rangle_\Sigma - \alpha^2 \langle \mathbf{curl}_\Sigma\, \psi, V(\mathbf{curl}_\Sigma\, \varphi)\rangle_\Sigma$$

for $\psi \in \gamma^{\mathrm{int}}_{0,\Sigma}(C_c^\infty(\overline{\Omega}\times(0,T)))$. The right-hand side of this equation is continuous as a bilinear form on $H^{1/2,1/4}(\Sigma)\times H^{1/2,1/4}(\Sigma)$. This follows from the continuity of the operators $D : H^{1/2,1/4}(\Sigma) \to H^{-1/2,-1/4}(\Sigma)$, $\mathbf{curl}_\Sigma : H^{1/2,1/4}(\Sigma) \to \boldsymbol{H}^{-1/2,-1/4}(\Sigma)$ and $V : \boldsymbol{H}^{-1/2,-1/4}(\Sigma) \to \boldsymbol{H}^{1/2,1/4}(\Sigma)$. Hence, also the left-hand side, i.e. the bilinear form

$$\begin{aligned} b(\cdot,\cdot) : H^{1/2,1/4}(\Sigma)\times \gamma^{\mathrm{int}}_{0,\Sigma}(C_c^\infty(\overline{\Omega}\times(0,T))) &\to \mathbb{R}\\ (\varphi,\psi) &\mapsto b(\varphi,\psi) = \Big\langle V(\varphi \boldsymbol{n}), \frac{\partial}{\partial t}\psi \boldsymbol{n}\Big\rangle_\Sigma \end{aligned}$$

is continuous with respect to the norm in $H^{1/2,1/4}(\Sigma)\times H^{1/2,1/4}(\Sigma)$ and it admits a unique, continuous extension to this space due to the density of $\gamma^{\mathrm{int}}_{0,\Sigma}(C_c^\infty(\overline{\Omega}\times(0,T)))$ in $H^{1/2,1/4}(\Sigma)$. In particular, the integration by parts formula (4.13) holds for general $\varphi,\psi \in H^{1/2,1/4}(\Sigma)$ by continuity, if we identify b with its continuous extension. □

4.2.2 An integral representation of the bilinear form b

In this section we focus on special situations in which we can express the bilinear form $b(\cdot,\cdot)$ on the right-hand side of the general integration by parts formula (4.13) in terms of weakly singular integrals. In Proposition 4.9 we have seen that (4.15) is not an appropriate representation of b, since the corresponding integral kernel is not Lebesgue integrable. Nonetheless, it was used in applications where it yielded satisfactory results; see e.g. [51, Example 3.2]. Let us further comment on this.

Let Σ_h be a space-time tensor product mesh of Σ and $S^{1\otimes 0}_{h_x,h_t}(\Sigma_h)$ be defined as in Section 2.4. Let $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}$ be one of the basis functions in $S^{1\otimes 0}_{h_x,h_t}(\Sigma_h)$ that vanishes outside of the time interval (t_{j_t-1},t_{j_t}). By a very formal computation we can evaluate $\alpha\, b(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t},\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t})$ via (4.15) as follows:

$$\begin{aligned}
&\alpha\, b(\varphi^1_{\boldsymbol{x},j_x}\varphi^0_{t,j_t},\varphi^1_{\boldsymbol{x},j_x}\varphi^0_{t,j_t})\\
&\overset{\text{form.}}{=} -\alpha \int_{t_{j_t-1}}^{t_{j_t}} \int_\Gamma \varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x}) \cdot \int_{t_{j_t-1}}^{t}\int_\Gamma \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y},t-\tau)\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\boldsymbol{n}(\boldsymbol{y})\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\mathrm{d}\tau\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\mathrm{d}t\\
&\overset{\text{form.}}{=} -\alpha \int_\Gamma\int_\Gamma \varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x})\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{y})\int_{t_{j_t-1}}^{t_{j_t}}\int_{t_{j_t-1}}^{t}\frac{\partial G_\alpha}{\partial\tau}(\boldsymbol{x}-\boldsymbol{y},t-\tau)\mathrm{d}\tau\mathrm{d}t\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\\
&\overset{\text{form.}}{=} \alpha \int_\Gamma\int_\Gamma \varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x})\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{y})\int_{t_{j_t-1}}^{t_{j_t}} G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-t_{j_t-1})\mathrm{d}t\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}, \quad (4.40)
\end{aligned}$$

where we used that $G_\alpha(\boldsymbol{x}-\boldsymbol{y},0)$ is zero if $\boldsymbol{x}-\boldsymbol{y}\neq\boldsymbol{0}$ for the last step. The remaining expression is then unproblematic since the kernel $((\boldsymbol{x},t),\boldsymbol{y})\mapsto G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-t_{j-1})$ is Lebesgue integrable on $(\Gamma\times(t_{j_t-1},t_{j_t}))\times\Gamma$. However, the steps taken to get this expression cannot be easily justified. Indeed, we cannot apply Fubini's theorem to change the order of integration, since the kernel (4.16) is not Lebesgue integrable on the integration domain as we have seen in Proposition 4.9, and we cannot ignore the singularities of $(\boldsymbol{x},\boldsymbol{y})\mapsto G_\alpha(\boldsymbol{x}-\boldsymbol{y},0)$ at $\boldsymbol{x}=\boldsymbol{y}$. Nonetheless, numerical results indicate that the result in (4.40) is valid.

In Theorem 4.17 we provide a representation of $\alpha\, b(\cdot,\cdot)$ that is similar to (4.15) but overcomes the problem of the locally non-integrable kernel. This formula is only valid for functions φ in suitable tensor product spaces on Σ. To define such a space we need to consider a partition $\mathcal{I}_{h_t}$ of the time interval $(0,T)$ as in Section 2.4 with subintervals (t_{j_t-1},t_{j_t}) for $j_t\in\{1,\dots,E_t\}$. We introduce

$$\begin{aligned} C^1_{\mathrm{pw}}(\mathcal{I}_{h_t}) := \Big\{ f \in & L^\infty(0,T) : f|_{(t_{j_t-1},t_{j_t})} \in C^1(t_{j_t-1},t_{j_t}), \\ & (f|_{(t_{j_t-1},t_{j_t})})' \in L^\infty(t_{j_t-1},t_{j_t}) \text{ for all } j_t\in\{1,\dots,E_t\}\Big\} \end{aligned} \tag{4.41}$$

and in addition the tensor product space $(L^\infty(\Gamma)\cap H^{1/2}(\Gamma))\otimes C^1_{\mathrm{pw}}(\mathcal{I}_{h_t})$, which is a subspace of $H^{1/2,1/4}(\Sigma)$.

THEOREM 4.17. *Let b be the bilinear form defined in Theorem 4.8. Let $\mathcal{I}_{h_t}$ be a partition of the time interval $(0,T)$ into subintervals $\{(t_{j_t-1},t_{j_t})\}_{j_t=1}^{E_t}$. For functions $\varphi\in(L^\infty(\Gamma)\cap H^{1/2}(\Gamma))\otimes C^1_{\mathrm{pw}}(\mathcal{I}_{h_t})$ and $\psi\in H^{1/2,1/4}(\Sigma)\cap L^\infty(\Sigma)$ there holds*

$$\begin{aligned} b(\varphi,\psi) = \sum_{j_t=1}^{E_t} \int_{t_{j_t-1}}^{t_{j_t}} \int_\Gamma \psi(\boldsymbol{x},t)\boldsymbol{n}(\boldsymbol{x}) \\ \cdot\Bigg[-\int_0^{t_{j_t-1}}\int_\Gamma \frac{\partial G_\alpha}{\partial\tau}(\boldsymbol{x}-\boldsymbol{y},t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau \\ + \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-t_{j_t-1})\varphi(\boldsymbol{y},t_{j_t-1}+)\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \\ + \int_{t_{j_t-1}}^t\int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)\frac{\partial\varphi}{\partial\tau}(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\Bigg]\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t, \end{aligned} \tag{4.42}$$

where $\varphi(\cdot,t_{j_t-1}+)$ denotes the right limit of φ with respect to time in t_{j_t-1}. In particular, all occurring integrands are Lebesgue integrable on the respective integration domains.

To prove Theorem 4.17 we follow a strategy that is similar to the one in the proof of Theorem 4.8. First, we show a corresponding result on a sequence of auxiliary boundaries $\{\Sigma_m\}_m$ in Lemma 4.18 and then we show that (4.42) follows in the limit as m tends to infinity.

LEMMA 4.18. *Let Ω_m be a Lipschitz domain with boundary Γ_m such that $\overline{\Omega}_m \subset \Omega$. Let $\Sigma_m = \Gamma_m \times (0,T)$, $\varphi \in (L^\infty(\Gamma) \cap H^{1/2}(\Gamma)) \otimes C^1_{\mathrm{pw}}(\mathcal{I}_{h_t})$ and $\psi_m \in H^{1/2,1/4}(\Sigma_m)$. Then*

$$\begin{aligned}
&\left\langle \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m}\left(\frac{\partial}{\partial t}\widetilde{V}(\varphi \boldsymbol{n})\right), \psi_m \right\rangle_{\Sigma_m} \qquad (4.43)\\
&= \sum_{j_t=1}^{E_t} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\Gamma_m} \psi_m(\boldsymbol{x},t)\boldsymbol{n}_m(\boldsymbol{x}) \cdot \Bigg[-\int_0^{t_{j_t-1}} \int_\Gamma \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau \\
&\qquad + \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-t_{j_t-1})\varphi(\boldsymbol{y}, t_{j_t-1}+)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \\
&\qquad + \int_{t_{j_t-1}}^{t} \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\frac{\partial \varphi}{\partial \tau}(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau \Bigg] \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \,\mathrm{d}t.
\end{aligned}$$

Proof. The left-hand side of (4.43) admits the integral representation

$$\begin{aligned}
&\left\langle \boldsymbol{n}_m \cdot \gamma^{\mathrm{int}}_{0,\Sigma_m}\left(\frac{\partial}{\partial t}\widetilde{V}(\varphi \boldsymbol{n})\right), \psi_m \right\rangle_{\Sigma_m} \\
&\qquad = \sum_{j_t=1}^{E_t} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\Gamma_m} \psi_m(\boldsymbol{x},t)\boldsymbol{n}_m(\boldsymbol{x}) \cdot \frac{\partial}{\partial t}\widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \,\mathrm{d}t.
\end{aligned} \qquad (4.44)$$

Note that we subdivided the integral over $(0,T)$ into a sum of integrals over the subintervals $\{(t_{j_t-1}, t_{j_t})\}_{j_t=1}^{E_t}$ of the partition $\mathcal{I}_{h_t}$. For an index $j_t \in \{1,\dots,E_t\}$, a time $t \in (t_{j_t-1}, t_{j_t})$, and a point $\boldsymbol{x} \in \Gamma_m \subset \Omega$ there holds

$$\begin{aligned}
\frac{\partial}{\partial t}\widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t) &= \frac{\partial}{\partial t}\left(\int_0^t \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau\right) \\
&= \int_0^t \int_\Gamma \frac{\partial G_\alpha}{\partial t}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau, \qquad (4.45)
\end{aligned}$$

where we used the smoothness of G_α away from zero to apply the Leibniz integral rule and that $G_\alpha(\boldsymbol{x}-\boldsymbol{y},0) = 0$ for all $\boldsymbol{y} \in \Gamma$. Next, we replace $\partial_t G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)$ by $-\partial_\tau G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)$ and split the temporal integral in (4.45) into integrals over $(0, t_{j_t-1})$ and (t_{j_t-1}, t). For the latter we apply the standard integration by parts formula to get

$$\begin{aligned}
&-\int_{t_{j_t-1}}^t \int_\Gamma \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau \\
&\qquad = \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-t_{j_t-1})\varphi(\boldsymbol{y}, t_{j_t-1}+)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_y \\
&\qquad\quad + \int_{t_{j_t-1}}^t \int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\frac{\partial \varphi}{\partial \tau}(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau.
\end{aligned}$$

Again, we used that $G_\alpha(\boldsymbol{x}-\boldsymbol{y},0) = 0$ for all $\boldsymbol{y} \in \Gamma$. Inserting the resulting representation of $\partial_t \widetilde{V}(\varphi\boldsymbol{n})(\boldsymbol{x},t)$ into (4.44) yields (4.43). □

Proof of Theorem 4.17. Let $\{\Omega_m\}_m$, $\{\mathbf{\Lambda}_m\}_m$ and $\{\omega_m\}_m$ be given as in Theorem 2.3, $\Gamma_m := \partial\Omega_m$ and $\Sigma_m := \Gamma_m \times (0,T)$. For $\psi \in H^{1/2,1/4}(\Sigma) \cap L^\infty(\Sigma)$ let $\widetilde{\psi}$ be the unique solution of the initial boundary value problem (1.1)–(1.3) with Dirichlet datum ψ and initial datum $u_0 = 0$. Note that $\widetilde{\psi}$ admits the representation

$$\widetilde{\psi} = W((-1/2\,\mathrm{Id} + K)^{-1}\psi), \tag{4.46}$$

cf. Theorem 4.5. In particular, $\widetilde{\psi} \in C^\infty(Q)$, and thus we can define $\psi_m := \widetilde{\psi}|_{\Sigma_m}$ in the classical sense. The idea of the proof is to show that for this choice of Σ_m and ψ_m the terms on the right-hand side of (4.43) converge to the respective terms of (4.42) in the limit $m \to \infty$.

We start with the first term on the right-hand side of (4.43). By transforming the integral over Γ_m into an integral over Γ we get

$$\begin{aligned}
&\sum_{j_t=1}^{E_t} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\Gamma_m} \int_0^{t_{j_t-1}} \int_\Gamma \psi_m(\boldsymbol{x},t)\boldsymbol{n}_m(\boldsymbol{x}) \cdot \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \\
&\quad = \sum_{j_t=1}^{E_t} \int_{t_{j_t-1}}^{t_{j_t}} \int_\Gamma \int_0^{t_{j_t-1}} \int_\Gamma \boldsymbol{h}_m(\mathbf{\Lambda}_m(\boldsymbol{x}), \boldsymbol{y}, t, \tau) \cdot \boldsymbol{f}(\boldsymbol{y},\tau)\omega_m(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t,
\end{aligned} \tag{4.47}$$

where we introduced the functions

$$\boldsymbol{h}_m(\boldsymbol{x},\boldsymbol{y},t,\tau) := \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\psi_m(\boldsymbol{x},t)\boldsymbol{n}_m(\boldsymbol{x}), \tag{4.48}$$

$$\boldsymbol{f}(\boldsymbol{y},\tau) := \varphi(\boldsymbol{y},\tau)\boldsymbol{n}(\boldsymbol{y}) \tag{4.49}$$

and used the homeomorphism $\mathbf{\Lambda}_m : \Gamma \to \Gamma_m$ and *(iv)* of Theorem 2.3. Let $\{Z_j\}_{j=1}^J$ be a finite family of coordinate cylinders covering Γ as in *(ii)* of Theorem 2.3; see also Definition 2.1. We split the outer spatial integrals in (4.47) into segments

$$U_j := \Gamma \cap Z_j$$

related to these coordinate cylinders. For this purpose, let $\{\Phi_j\}_{j=1}^J$ be a smooth partition of unity on Γ subordinate to $\{Z_j\}_{j=1}^J$, i.e. $\Phi_j \in C^\infty(\mathbb{R}^3)$, $\operatorname{supp}(\Phi_j) \subset Z_j$, $0 \le \Phi_j \le 1$ and $\sum_j \Phi_j(\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in \Gamma$. Then (4.47) is further equal to

$$\begin{aligned}
\sum_{j_t=1}^{E_t} \sum_{j=1}^{J} \int_{t_{j_t-1}}^{t_{j_t}} \int_{U_j} \int_0^{t_{j_t-1}} \int_\Gamma \Phi_j(\boldsymbol{x}) \boldsymbol{h}_m(\mathbf{\Lambda}_m(\boldsymbol{x}), \boldsymbol{y}, t, \tau) \\
\cdot \boldsymbol{f}(\boldsymbol{y},\tau)\omega_m(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.
\end{aligned} \tag{4.50}$$

In particular, the convergence of the first term in (4.43) to the first term in (4.42) follows if we can show that

$$\lim_{m\to\infty}\left(\int_{t_{j_t-1}}^{t_{j_t}}\int_{U_j}\int_0^{t_{j_t-1}}\int_\Gamma \Phi_j(\boldsymbol{x})\boldsymbol{h}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}),\boldsymbol{y},t,\tau)\cdot\boldsymbol{f}(\boldsymbol{y},\tau)\omega_m(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\right)$$
$$=\int_{t_{j_t-1}}^{t_{j_t}}\int_{U_j}\int_0^{t_{j_t-1}}\int_\Gamma \Phi_j(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t,\tau)\cdot\boldsymbol{f}(\boldsymbol{y},\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \tag{4.51}$$

for all $j_t\in\{1,\dots,E_t\}$ and $j\in\{1,\dots,J\}$, where

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t,\tau):=\frac{\partial G_\alpha}{\partial\tau}(\boldsymbol{x}-\boldsymbol{y},t-\tau)\psi(\boldsymbol{x},t)\boldsymbol{n}(\boldsymbol{x}).$$

For this purpose, we split the inner integral of $\boldsymbol{y}$ over Γ into an integral over

$$V_j:=3Z_j\cap\Gamma$$

and one over the remainder $\Gamma\backslash V_j$ and show the convergence of both parts using the classical dominated convergence theorem.

First, we consider

$$\int_{t_{j_t-1}}^{t_{j_t}}\int_{U_j}\int_0^{t_{j_t-1}}\int_{\Gamma\backslash V_j} \Phi_j(\boldsymbol{x})\boldsymbol{h}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}),\boldsymbol{y},t,\tau)\cdot\boldsymbol{f}(\boldsymbol{y},\tau)\omega_m(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \tag{4.52}$$

and observe the pointwise convergence of $\omega_m(\boldsymbol{x})\boldsymbol{h}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}),\boldsymbol{y},t,\tau)$ to $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t,\tau)$ almost everywhere in the integration domain as m tends to infinity. In fact, $\omega_m\to 1$ pointwise almost everywhere on Γ by Theorem 2.3 *(iv)*. For the respective convergence of $\boldsymbol{h}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}),\boldsymbol{y},t,\tau)$ in (4.48) to $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t,\tau)$ we show the convergence of the individual terms. By *(iii)* of Theorem 2.3 we get that $\boldsymbol{n}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}))\to\boldsymbol{n}(\boldsymbol{x})$ for almost all $\boldsymbol{x}\in\Gamma$. Furthermore $\boldsymbol{\Lambda}_m(\boldsymbol{x})\to\boldsymbol{x}$ even uniformly on Γ, by *(i)*. Thus, we obtain the convergence of $\partial G_\alpha/\partial\tau(\boldsymbol{\Lambda}_m(\boldsymbol{x})-\boldsymbol{y},t-\tau)$ to $\partial G_\alpha/\partial\tau(\boldsymbol{x}-\boldsymbol{y},t-\tau)$ for all $\boldsymbol{x},\boldsymbol{y}\in\Gamma$ and $\tau<t$. Finally, we know that $\psi_m=\widetilde{\psi}|_{\Sigma_m}$ and that $\widetilde{\psi}$ given by (4.46) attains the Dirichlet boundary values ψ on Σ in the sense of non-tangential limits almost everywhere; see Theorem 4.5. Hence, $\psi_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}),t)\to\psi(\boldsymbol{x},t)$ since $\boldsymbol{\Lambda}_m(\boldsymbol{x})$ approaches $\boldsymbol{x}$ non-tangentially by point *(i)* of Theorem 2.3.

It remains to find an integrable function that dominates the sequence of integrands. Obviously there holds $\Phi_j\le 1$ on U_j by construction and $|\boldsymbol{f}(\boldsymbol{y},\tau)|\le\|\varphi\|_{L^\infty(\Sigma)}$ almost everywhere on Σ. By Theorem 2.3 *(iv)* we get in addition, that $\omega_m(\boldsymbol{x})\le c^{-1}$ for a constant $c>0$ independent of m and all $\boldsymbol{x}\in\Gamma$. Furthermore, there holds $|\psi_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}))\boldsymbol{n}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}))|\le\|\psi\|_{L^\infty(\Sigma)}$ for almost all $\boldsymbol{x}\in\Gamma$ by the extended maximum principle in Theorem 4.7. The only term left to bound is $\partial G_\alpha/\partial\tau(\boldsymbol{\Lambda}_m(\boldsymbol{x})-\boldsymbol{y},t-\tau)$. Let r and h be the radius and the height of the congruent cylinders Z_j, respectively,

and let $\delta := \min(r, h)$. Due to (2.2) we can assume without loss of generality that $|\boldsymbol{x} - \boldsymbol{\Lambda}_m(\boldsymbol{x})| < \delta$ for all $\boldsymbol{x} \in \Gamma$ and all m. Then we immediately get $\boldsymbol{\Lambda}_m(\boldsymbol{x}) \in 2Z_j$ for all $\boldsymbol{x} \in U_j = \Gamma \cap Z_j$, and thus $|\boldsymbol{\Lambda}_m(\boldsymbol{x}) - \boldsymbol{y}| > \delta$ for $\boldsymbol{x} \in U_j$ and $\boldsymbol{y} \in \Gamma \backslash V_j = \Gamma \backslash 3Z_j$. This allows us to bound

$$\left| \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{\Lambda}_m(\boldsymbol{x}) - \boldsymbol{y}, t - \tau) \right| \leq \left\| \frac{\partial G_\alpha}{\partial t} \right\|_{L^\infty((\overline{B_R(\mathbf{0})} \backslash B_\delta(\mathbf{0})) \times [0,T])}$$

for all $\boldsymbol{x} \in U_j$, $\boldsymbol{y} \in \Gamma \backslash V_j$, and $0 < \tau < t < T$, where $B_R(\mathbf{0})$ and $B_\delta(\mathbf{0})$ are balls centered around the origin with radii R and δ, respectively, and R is so large that $\overline{\Omega} \subset B_{R/2}(\mathbf{0})$. The right-hand side in this estimate is bounded since the function $(\boldsymbol{r}, t) \mapsto \partial G_\alpha / \partial t(\boldsymbol{r}, t)$ is in $C^\infty(\mathbb{R}^3 \backslash \{\mathbf{0}\} \times [0, T])$. Altogether we have found the desired dominating function, as

$$\begin{aligned} &|\Phi_j(\boldsymbol{x}) \boldsymbol{h}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}), \boldsymbol{y}, t, \tau) \cdot \boldsymbol{f}(\boldsymbol{y}, \tau) \omega_m(\boldsymbol{x})| \\ &\qquad \leq \|\varphi\|_{L^\infty(\Sigma)} \|\psi\|_{L^\infty(\Sigma)} \left\| \frac{\partial G_\alpha}{\partial t} \right\|_{L^\infty((\overline{B_R(\mathbf{0})} \backslash B_\delta(\mathbf{0})) \times [0,T])} \end{aligned}$$

almost everywhere in the integration domain of (4.52). Therefore, we have established the convergence of (4.52) to

$$\int_{t_{j_t-1}}^{t_{j_t}} \int_{U_j} \int_0^{t_{j_t-1}} \int_{\Gamma \backslash V_j} \Phi_j(\boldsymbol{x}) \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, t, \tau) \cdot \boldsymbol{f}(\boldsymbol{y}, \tau) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\tau \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \, \mathrm{d}t. \tag{4.53}$$

To show (4.51) we have to consider the remaining part

$$\int_{t_{j_t-1}}^{t_{j_t}} \int_{U_j} \int_0^{t_{j_t-1}} \int_{V_j} \Phi_j(\boldsymbol{x}) \boldsymbol{h}_m(\boldsymbol{\Lambda}_m(\boldsymbol{x}), \boldsymbol{y}, t, \tau) \cdot \boldsymbol{f}(\boldsymbol{y}, \tau) \omega_m(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\tau \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \, \mathrm{d}t. \tag{4.54}$$

We use the parametrization of the regions $\boldsymbol{\Lambda}_m(U_j)$ and V_j by the Lipschitz functions $\eta_j^{(m)}$ and η_j, respectively, established in Theorem 2.3 *(ii)*. This is possible, since $\boldsymbol{\Lambda}_m(U_j) \subset \Gamma_m \cap 2Z_j$ as seen before. For the sake of simplicity, we assume that the coordinates associated with the cylinder Z_j correspond to the original rectangular coordinates, i.e. we neglect additional translations and rotations. Then (4.54) can be transformed into

$$\begin{aligned} \int_{t_{j_t-1}}^{t_{j_t}} \int_{B_{2r}(\mathbf{0})} \int_0^{t_{j_t-1}} \int_{B_{3r}(\mathbf{0})} &\mathbb{1}_{\mathcal{U}_j^{(m)}}(\hat{\boldsymbol{x}}) \Phi_j(\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}))) \boldsymbol{h}_m(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}), \boldsymbol{\Gamma}(\hat{\boldsymbol{y}}), t, \tau) \\ &\cdot \boldsymbol{f}(\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}), \tau) g_m(\hat{\boldsymbol{x}}) g(\hat{\boldsymbol{y}}) \, \mathrm{d}\hat{\boldsymbol{x}} \, \mathrm{d}\tau \, \mathrm{d}\hat{\boldsymbol{y}} \, \mathrm{d}t. \end{aligned} \tag{4.55}$$

where $\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}) := (\hat{\boldsymbol{y}}, \eta_j(\hat{\boldsymbol{y}}))$, $\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}) := (\hat{\boldsymbol{x}}, \eta_j^{(m)}(\hat{\boldsymbol{x}}))$, r is still the radius of Z_j, $B_{3r}(\mathbf{0})$ the parameter region of V_j, $\mathcal{U}_j^{(m)} \subset B_{2r}(\mathbf{0})$ the parameter region of $\boldsymbol{\Lambda}_m(U_j)$, and $g_m(\hat{\boldsymbol{x}})$ and $g(\hat{\boldsymbol{y}})$ denote the surface elements given by

$$g_m(\hat{\boldsymbol{x}}) = \sqrt{1 + |\nabla \eta_j^{(m)}(\hat{\boldsymbol{x}})|^2}, \qquad g(\hat{\boldsymbol{y}}) = \sqrt{1 + |\nabla \eta_j(\hat{\boldsymbol{y}})|^2}.$$

We compute the limit of (4.55) for $m \to \infty$ using the dominated convergence theorem again. First we show that the integrand in (4.55) converges to

$$\mathbb{1}_{B_r(\mathbf{0})}(\hat{\boldsymbol{x}})\Phi_j(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}))\boldsymbol{h}(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}),\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),t,\tau)\cdot\boldsymbol{f}(\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),\tau)g(\hat{\boldsymbol{x}})g(\hat{\boldsymbol{y}}) \tag{4.56}$$

for almost all $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, t and τ in the integration domain as $m \to \infty$. In general, there holds $\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}) \neq \boldsymbol{\Lambda}_m(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}))$, so we cannot use the previous results about pointwise convergence to show this. Instead, we consider again all terms of the integrand in (4.55) depending on m separately to show pointwise convergence. Recall the definition of $\boldsymbol{h}_m$ in (4.48) for this purpose. Theorem 2.3 *(ii)* implies that $\eta_j^{(m)}$ converges uniformly to η_j. As a result, $\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}) \to \boldsymbol{\Gamma}(\hat{\boldsymbol{x}})$ for all $\hat{\boldsymbol{x}} \in B_{2r}(\mathbf{0})$, and thus $\partial G_\alpha/\partial\tau(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}),\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),t,\tau)$ converges to $\partial G_\alpha/\partial\tau(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}),\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),t,\tau)$ for all such $\hat{\boldsymbol{x}}$, and $\hat{\boldsymbol{y}}$, t and τ in the respective domains of integration. Since the convergence of $\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})$ to $\boldsymbol{\Gamma}(\hat{\boldsymbol{x}})$ is non-tangential it follows as before that $\psi_m(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}),t) \to \psi(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}),t)$ for almost all $\hat{\boldsymbol{x}} \in B_{2r}(\mathbf{0})$ and $t \in (t_{j_t-1}, t_{j_t})$. From Theorem 2.3 *(ii)* we know in addition that $\nabla\eta_j^{(m)}$ converges pointwise almost everywhere to $\nabla\eta_j$ in $\mathbb{R}^2$. This implies $g_m(\hat{\boldsymbol{x}}) \to g(\hat{\boldsymbol{x}})$ for almost all $\hat{\boldsymbol{x}}$ and furthermore the convergence of $\boldsymbol{n}_m(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}))$ to $\boldsymbol{n}(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}))$ for almost all $\hat{\boldsymbol{x}} \in B_{2r}(\mathbf{0})$, due to the representations

$$\boldsymbol{n}_m(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})) = \frac{1}{g_m(\hat{\boldsymbol{x}})}(\nabla\eta_j^{(m)}(\hat{\boldsymbol{x}})^\top, -1)^\top, \qquad \boldsymbol{n}(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}})) = \frac{1}{g(\hat{\boldsymbol{y}})}(\nabla\eta_j(\hat{\boldsymbol{x}})^\top, -1)^\top.$$

Next we show the convergence of $\mathbb{1}_{\mathcal{U}_j^{(m)}}$ to $\mathbb{1}_{B_r(\mathbf{0})}$ pointwise almost everywhere in the ball $B_{2r}(\mathbf{0})$. From (2.2) it follows that for all $\hat{\boldsymbol{x}} \in B_{2r}(\mathbf{0})$ and sufficiently small $\varepsilon > 0$ there exists an $m(\varepsilon)$ such that for all $m > m(\varepsilon)$

$$\boldsymbol{\Lambda}_m^{-1}(\Gamma_m \cap Z_{\varepsilon/2}(\hat{\boldsymbol{x}})) \subset (\Gamma \cap Z_\varepsilon(\hat{\boldsymbol{x}})), \tag{4.57}$$

where $Z_\rho(\hat{\boldsymbol{x}}) := \{(\hat{\boldsymbol{\xi}}, \hat{s}) \in \mathbb{R}^3 : |\hat{\boldsymbol{\xi}} - \hat{\boldsymbol{x}}| < \rho, \hat{s} < h_j\}$ is the cylinder with center $(\hat{\boldsymbol{x}}, 0)$, radius $\rho > 0$ and the same height h_j as Z_j. If $\hat{\boldsymbol{x}} \in B_r(\mathbf{0})$ and ε is so small that $(\Gamma \cap Z_\varepsilon(\hat{\boldsymbol{x}})) \subset U_j$, (4.57) implies that $\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}) \in \boldsymbol{\Lambda}_m(U_j)$ for all $m > m(\varepsilon)$. This means that $\hat{\boldsymbol{x}} \in \mathcal{U}_j^{(m)}$, and thus $\mathbb{1}_{\mathcal{U}_j^{(m)}}(\hat{\boldsymbol{x}}) = 1 = \mathbb{1}_{B_r(\mathbf{0})}(\hat{\boldsymbol{x}})$ for all $m > m(\varepsilon)$. Likewise, if $\hat{\boldsymbol{x}} \in B_{2r}(\mathbf{0})\backslash\overline{B_r(\mathbf{0})}$ and ε is so small that $(\Gamma \cap Z_\varepsilon(\hat{\boldsymbol{x}})) \cap U_j = \emptyset$, we get $\mathbb{1}_{\mathcal{U}_j^{(m)}}(\hat{\boldsymbol{x}}) = 0 = \mathbb{1}_{B_r(\mathbf{0})}(\hat{\boldsymbol{x}})$ for all $m > m(\varepsilon)$. Together this proves $\mathbb{1}_{\mathcal{U}_j^{(m)}} \to \mathbb{1}_{B_r(\mathbf{0})}$ almost everywhere in $B_{2r}(\mathbf{0})$.

Finally we have to show that $\Phi_j(\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})))$ converges to $\Phi_j(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}))$ in $B_{2r}(\mathbf{0})$. Since Φ_j is continuous it suffices to show the convergence of its arguments. This follows again from (4.57). Indeed, for each $\varepsilon > 0$ we know by (4.57) that for all $m > m(\varepsilon)$ there exists a $\hat{\boldsymbol{y}}_m$ such that $\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})) = \boldsymbol{\Gamma}(\hat{\boldsymbol{y}}_m)$ and $|\hat{\boldsymbol{y}}_m - \hat{\boldsymbol{x}}| < \varepsilon$. As a consequence there holds

$$|\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})) - \boldsymbol{\Gamma}(\hat{\boldsymbol{x}})| = |\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}_m) - \boldsymbol{\Gamma}(\hat{\boldsymbol{x}})| = |(\hat{\boldsymbol{y}}_m, \eta_j(\hat{\boldsymbol{y}}_m)) - (\hat{\boldsymbol{x}}, \eta_j(\hat{\boldsymbol{x}}))| \leq \varepsilon\sqrt{1 + L_j^2},$$

where L_j is the Lipschitz constant of η_j. This yields the desired convergence. In particular, we have shown that the integrand in (4.55) converges pointwise almost everywhere to (4.56).

For the application of the dominated convergence theorem we bound the integrands in (4.55) uniformly by using the estimates

$$\begin{aligned}
|\mathbb{1}_{\mathcal{U}_j^{(m)}}(\hat{\boldsymbol{x}})\Phi_j(\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})))| &\leq 1,\\
|\varphi(\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),\tau)\boldsymbol{n}(\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}))| &\leq \|\varphi\|_{L^\infty(\Sigma)},\\
|\psi_m(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}),t)\boldsymbol{n}_m(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}))| &\leq \|\psi\|_{L^\infty(\Sigma)},\\
|g(\hat{\boldsymbol{y}})| &\leq \sqrt{1+\|\nabla\eta_j\|_{L^\infty(\mathbb{R}^2)}},\\
|g_m(\hat{\boldsymbol{x}})| &\leq \sqrt{1+\|\nabla\eta_j^{(m)}\|_{L^\infty(\mathbb{R}^2)}} \leq \sqrt{1+\|\nabla\eta_j\|_{L^\infty(\mathbb{R}^2)}}
\end{aligned}$$

for almost all $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, t and τ in the integration domain. The first estimate is clear by definition. The second one holds true due to the assumption that $\varphi \in L^\infty(\Sigma)$. The third estimate is again a consequence of the parabolic maximum principle in Theorem 4.7. The last two estimates follow from the definition of the surface elements and the fact that $\|\nabla\eta_j^{(m)}\|_{L^\infty(\mathbb{R}^2)} \leq \|\nabla\eta_j\|_{L^\infty(\mathbb{R}^2)}$; see Theorem 2.3 *(ii)*. Therefore, the product of all these functions is bounded by a constant C independent of m. The remaining term, i.e. the derivative of the heat kernel which we computed in (4.17), can be estimated by

$$\begin{aligned}
\left|\frac{\partial G_\alpha}{\partial\tau}(\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}}),\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),t,\tau)\right| &\leq \frac{c(\alpha)}{(t-\tau)^{7/4}|\boldsymbol{\Gamma}_m(\hat{\boldsymbol{x}})-\boldsymbol{\Gamma}(\hat{\boldsymbol{y}})|^{3/2}}\\
&\leq \frac{c(\alpha)}{(t-\tau)^{7/4}|\hat{\boldsymbol{x}}-\hat{\boldsymbol{y}}|^{3/2}} \qquad (4.58)
\end{aligned}$$

almost everywhere in the integration domain. This estimate follows similarly as (4.20) by considering the two terms in the numerator of (4.17) separately. Since the function on the right-hand side of (4.58) is integrable on the considered integration domain $B_{2r}(\mathbf{0}) \times (t_{j_t-1}, t_{j_t}) \times B_{3r}(\mathbf{0}) \times (0, t_{j_t-1})$ we have found a function dominating the sequence of integrands.

By the dominated convergence theorem we get that (4.55) converges to

$$\begin{aligned}
&\int_{t_{j_t-1}}^{t_{j_t}}\int_{B_r(\mathbf{0})}\int_0^{t_{j_t-1}}\int_{B_{3r}(\mathbf{0})} \Phi_j(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}))\boldsymbol{h}(\boldsymbol{\Gamma}(\hat{\boldsymbol{x}}),\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),t,\tau)\cdot\boldsymbol{f}(\boldsymbol{\Gamma}(\hat{\boldsymbol{y}}),\tau)g(\hat{\boldsymbol{x}})g(\hat{\boldsymbol{y}})\,\mathrm{d}\hat{\boldsymbol{x}}\,\mathrm{d}\tau\,\mathrm{d}\hat{\boldsymbol{y}}\,\mathrm{d}t\\
&\quad=\int_{t_{j_t-1}}^{t_{j_t}}\int_{U_j}\int_0^{t_{j_t-1}}\int_{V_j}\Phi_j(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t,\tau)\cdot\boldsymbol{f}(\boldsymbol{y},\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.
\end{aligned}$$

Together with the convergence of (4.52) to (4.53) we conclude (4.51).

Recall that we wanted to show that all terms on the right-hand side of (4.43) converge to the respective terms of (4.42). Equation (4.51) implies the convergence of the first term. In particular, the integrand of the first term on the right-hand side of (4.42) is Lebesgue integrable in the corresponding integration domain, which is another consequence of the dominated convergence theorem. The remaining two terms can be handled analogously. For both terms one can transform the integrals of $\boldsymbol{x}$ over Γ_m into integrals over Γ as in (4.47) and split the integrals up as in (4.50). The individual parts can then be handled as before by splitting the inner integral of $\boldsymbol{y}$ over Γ into integrals over V_j and $\Gamma \backslash V_j$. For both one applies the dominated convergence theorem where one can use (4.20) to bound the heat kernel and in addition the estimates

$$\begin{aligned} |\varphi(\boldsymbol{y}, t_{j_t-1}+)| &\leq \|\varphi(\cdot, t_{j_t-1}+)\|_{L^\infty(\Gamma)} && \text{for all } \boldsymbol{y} \in \Gamma, \\ \left|\frac{\partial \varphi}{\partial \tau}(\boldsymbol{y}, \tau)\right| &\leq \left\|\frac{\partial \varphi}{\partial \tau}\right\|_{L^\infty(\Gamma \times (t_{j_t-1}, t_{j_t}))} && \text{for all } \boldsymbol{y} \in \Gamma \text{ and } \tau \in (t_{j_t-1}, t_{j_t}). \end{aligned}$$

These estimates are justified since $\varphi \in (L^\infty(\Gamma) \cap H^{1/2}(\Gamma)) \otimes C^1_{\mathrm{pw}}(\mathcal{I}_{h_t})$ with the space $C^1_{\mathrm{pw}}(\mathcal{I}_{h_t})$ defined in (4.41). □

The representation of the bilinear form b in (4.42) is tailored to boundary element methods with tensor product spaces used for discretization, like in Section 2.4. Indeed, the assumptions of Theorem 4.17 are satisfied for $\varphi_h, \psi_h \in S^{1\otimes 0}_{h_x,h_t}(\Sigma_h)$. Since the derivative $\partial_\tau \varphi_h(\boldsymbol{y}, \tau)$ is zero for such functions, we can simplify the representation of b in (4.42) even further and obtain

$$\begin{aligned} b(\varphi_h, \psi_h) = \sum_{j_t=1}^{E_t} \int_{t_{j_t-1}}^{t_{j_t}} \int_\Gamma \psi_h(\boldsymbol{x}, t) \boldsymbol{n}(\boldsymbol{x}) \\ \cdot \left[- \int_0^{t_{j_t-1}} \int_\Gamma \frac{\partial G_\alpha}{\partial \tau}(\boldsymbol{x} - \boldsymbol{y}, t-\tau) \varphi_h(\boldsymbol{y}, \tau) \boldsymbol{n}(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\tau \right. \\ \left. + \int_\Gamma G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - t_{j_t-1}) \varphi_h(\boldsymbol{y}, t_{j_t-1}+) \boldsymbol{n}(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \right] \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \, \mathrm{d}t. \end{aligned} \tag{4.59}$$

When we evaluate this bilinear form for $\varphi_h = \psi_h = \varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}} \varphi^0_{t,j_t}$, where $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}} \varphi^0_{t,j_t}$ is a basis function of $S^{1\otimes 0}_{h_x,h_t}(\Sigma_h)$, we obtain the same result as in (4.40). Hence, Theorem 4.17 justifies the calculations found in the literature.

4.2.3 Evaluating the BEM matrix of the hypersingular operator

With the integration by parts formula (4.13) and the representation of b in (4.59) we can finally describe the assembly of the matrix D_h in (3.32). The entries of this

matrix correspond to the values of the bilinear form $\langle D\cdot,\cdot\rangle_\Sigma$ evaluated for pairs of basis functions in $S^{1\otimes 0}_{h_x,h_t}(\Sigma_h)$. They are given by

$$\begin{aligned}&\mathsf{D}_h[(k_t-1)N_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)N_{\boldsymbol{x}}+j_{\boldsymbol{x}}]\\&\quad=\alpha^2\langle V\,\mathbf{curl}_\Sigma(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}),\mathbf{curl}_\Sigma(\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t})\rangle_\Sigma+\alpha b(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t},\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t})\end{aligned}\tag{4.60}$$

for all k_t, $j_t\in\{1,\dots,E_t\}$ and $k_{\boldsymbol{x}}$, $j_{\boldsymbol{x}}\in\{1,\dots,N_{\boldsymbol{x}}\}$, where we use the notation from Section 2.4. Let us take a closer look at the bilinear forms on the right-hand side.

The surface curls of functions in $S^{1\otimes 0}_{h_x,h_t}(\Sigma_h)$ are in $\boldsymbol{S}^{0\otimes 0}_{h_x,h_t}(\Sigma_h)$, i.e. their components are piecewise constant on the mesh Σ_h. In fact, there holds

$$\mathbf{curl}_\Sigma(\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t})=\mathbf{curl}_\Gamma(\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}})\varphi^0_{t,k_t},$$

where $\mathbf{curl}_\Gamma:H^{1/2}(\Gamma)\to\boldsymbol{H}^{-1/2}(\Gamma)$ can be defined similarly as in (4.4), or by $\mathbf{curl}_\Gamma\varphi=\nabla\widetilde{\varphi}\times\boldsymbol{n}$ for $\varphi\in H^{1/2}(\Gamma)$ and a suitably regular extension $\widetilde{\varphi}$ on $\overline{\Omega}$. Computing $\mathbf{curl}_\Gamma\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}$ and concluding that it is in $\boldsymbol{S}^0_{h_{\boldsymbol{x}}}(\Gamma_h)$ for a mesh Γ_h consisting of plane triangles is a simple exercise; see [66, Section 12.5, Exercise 12.5]. It follows that

$$\begin{aligned}&\langle V\,\mathbf{curl}_\Sigma(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}),\mathbf{curl}_\Sigma(\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t})\rangle_\Sigma\\&=\int_{t_{k_t-1}}^{t_{k_t}}\int_\Gamma\int_{t_{j_t-1}}^{t_{j_t}}\int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)\,\mathbf{curl}_\Gamma(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}})(\boldsymbol{y})\cdot\mathbf{curl}_\Gamma(\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}})(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\end{aligned}\tag{4.61}$$

and the entries of the single layer operator matrix V_h in (3.25) can be reused to evaluate this bilinear form. More details are given by the authors in [81, Section 3.3].

When evaluating the bilinear form b for basis functions in $S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}$ with (4.59) we distinguish three cases. There holds

$$\begin{aligned}&b(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t},\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t})\\&=\int_{t_{k_t-1}}^{t_{k_t}}\int_\Gamma\int_{t_{j_t-1}}^{t_{j_t}}\int_\Gamma\frac{\partial G_\alpha}{\partial\tau}(\boldsymbol{x}-\boldsymbol{y},t-\tau)\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\end{aligned}\tag{4.62}$$

for $j_{\boldsymbol{x}},k_{\boldsymbol{x}}\in\{1,\dots,N_{\boldsymbol{x}}\}$ and $j_t,k_t\in\{1,\dots,E_t\}$ with $j_t<k_t$. If $j_t>k_t$ the result is zero. Finally, if $j_t=k_t$,

$$\begin{aligned}&b(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t},\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,j_t})\\&\qquad=\int_{t_{j_t-1}}^{t_{j_t}}\int_\Gamma\int_\Gamma G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-t_{j_t-1})\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.\end{aligned}\tag{4.63}$$

To evaluate the remaining integrals we can use an analytic integration in time and numerical quadrature formulae in space. The details about this procedure can be found again in [81, Section 3.3].

REMARK 4.19. *For basis functions* $\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t}$ *and* $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t}$ *with* $j_t < k_t - 1$ *we can compute the corresponding entry of* D_h *alternatively by*

$$\begin{aligned}\mathsf{D}_h[(k_t-1)N_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)N_{\boldsymbol{x}}+j_{\boldsymbol{x}}] &= \langle D\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t},\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}\varphi^0_{t,k_t}\rangle_\Sigma \qquad (4.64)\\ &= -\int_{t_{k_t-1}}^{t_{k_t}}\int_\Gamma\int_{t_{j_t-1}}^{t_{j_t}}\int_\Gamma \alpha^2\frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{x}}}\frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{y}}}G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t.\end{aligned}$$

The integral is well-defined, since $((\boldsymbol{x},t),(\boldsymbol{y},\tau)) \mapsto \partial_{\boldsymbol{n}_{\boldsymbol{x}}}\partial_{\boldsymbol{n}_{\boldsymbol{y}}}G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)$ *is uniformly bounded for* $\boldsymbol{x},\boldsymbol{y}\in\Gamma$, $t\in(t_{k_t-1},t_{k_t})$, *and* $\tau\in(t_{j_t-1},t_{j_t})$ *with* $t_{j_t} < t_{k_t-1}$. *This alternative is particularly interesting for fast methods for the application of* D_h *like the one considered in the next chapters. Note that we still have to use the representation in* (4.60) *if* $j_t \in \{k_t-1,k_t\}$.

5 A space-time FMM for the heat equation

A major difficulty in applying BEM for the solution of initial boundary value problems of the transient heat equation is that the matrices occurring in the corresponding linear systems like (3.24) and (3.32) are dense. This is not a peculiarity of BEM for the heat equation, but of BEM for the solution of PDEs in general due to the non-local integrals that appear in the definition of typical boundary integral operators. The matrix $\mathsf{V}_h \in \mathbb{R}^{E_t E_{\boldsymbol{x}} \times E_t E_{\boldsymbol{x}}}$ related to the single layer boundary integral operator V of the heat equation, for example, has the lower triangular block structure (3.29), but the entries of all blocks $\mathsf{V}_h^{k_t,j_t}$ with $k_t \geq j_t$ are non-zero. Therefore, assembling, storing, or applying V_h, or solving systems of the form $\mathsf{V}_h \boldsymbol{q} = \boldsymbol{f}$ with a blockwise forward elimination scheme has a complexity of $\mathcal{O}((E_t E_{\boldsymbol{x}})^2)$ which is prohibitive for large examples.

To overcome this problem several fast and data-sparse methods have been developed to compress and efficiently handle matrices related to BEM in general and BEM for the heat equation in particular. Some examples for the heat equation are mentioned in the general introduction in Chapter 1. The fast method which we consider in this work is closely related to the parabolic fast multipole method (pFMM); see [52, 53, 67, 68]. In its original form the pFMM combines standard ideas and principles of fast multipole methods (FMM) — like partitioning the computational domain into a hierarchy of boxes and approximating interactions in well-separated boxes by using suitable truncated series expansions of the related kernel functions — with an efficient forward-sweeping execution scheme that can be used for a block forward elimination procedure to solve linear systems like (3.24) and (3.32). Since one of the aims of this work is to develop a fast method that allows for an efficient parallelization in space and time, we describe how to compress and apply whole BEM matrices like V_h at once, instead. The resulting method is called space-time FMM to distinguish it from the pFMM.

The description of the space-time FMM in this chapter is similar to the description in [78]. In Section 5.1 we describe the truncated series expansion of the heat kernel that is used for the FMM and analyze the corresponding approximation error. The space-time FMM itself is described in full detail in Section 5.2. While we focus on the single layer operator matrix V_h in this description, we also mention the necessary changes to obtain fast methods for the application of the BEM matrices K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h introduced in Section 3.3.

5.1 A separable approximation of the heat kernel

The FMM presented in this chapter is based on a suitable separable approximation of the heat kernel (3.4) that combines an interpolation in time with a truncated Chebyshev expansion in space. The expansion has been considered in the same form in [51, 68]. We start by fixing the notation and stating a few results from the literature.

The Chebyshev polynomial T_m of order $m \in \mathbb{N}_0$ on the interval $[-1,1]$ is given by $T_m(x) = \cos(m \arccos(x))$. The polynomial T_{m+1} has $m+1$ distinct roots on $[-1,1]$ which are known as the Chebyshev nodes of order $m+1$ and are denoted by $\{\xi_k^{(m)}\}_{k=0}^m$. Let $I = [a,b]$ be non-empty and φ_I be the affine map from $[-1,1]$ to I defined by $\varphi_I(x) := (a(1-x) + b(1+x))/2$. The transformed Chebyshev polynomials $T_{I,m}$ on the interval I are given by $T_{I,m} := T_m \circ \varphi_I^{-1}$ and the transformed Chebyshev nodes $\{\xi_{I,k}^{(m)}\}_{k=0}^m$ of degree $m+1$ by $\xi_{I,k}^{(m)} := \varphi_I(\xi_k^{(m)})$ for all $k \in \{0, \dots, m\}$.

Let $\mathcal{P}_m(I)$ be the space of all polynomials with degrees less than or equal to m on an interval $I = [a,b]$. When interpolating a function $f \in C(I)$ by polynomials in $\mathcal{P}_m(I)$ we use the transformed Chebyshev nodes as interpolation points and obtain

$$\mathfrak{I}_I^{(m)}[f] = \sum_{k=0}^{m} f(\xi_{I,k}^{(m)}) L_{I,k}^{(m)}, \tag{5.1}$$

where $\mathfrak{I}_I^{(m)} : C(I) \to \mathcal{P}_m(I)$ denotes the corresponding interpolation operator and $\{L_{I,k}^{(m)}\}_{k=0}^m$ are the Lagrange polynomials given by

$$L_{I,k}^{(m)}(t) := \prod_{\substack{j=0 \\ j \neq k}}^{m} \frac{t - \xi_{I,k}^{(m)}}{\xi_{I,j}^{(m)} - \xi_{I,k}^{(m)}}. \tag{5.2}$$

The norm of the operator $\mathfrak{I}_I^{(m)}$ is known as the Lebesgue constant and is given by

$$\Lambda_m := \sup_{u \in C(I) \setminus \{0\}} \frac{\|\mathfrak{I}_I^{(m)}[u]\|_{\infty,I}}{\|u\|_{\infty,I}}. \tag{5.3}$$

Note that Λ_m depends on the choice of the interpolation points on the interval I, but it does not depend on the particular interval I as long as the same interpolation points — up to an affine transformation — are used. For transformed Chebyshev nodes as interpolation points we can estimate [61, cf. Theorem 1.2]

$$\Lambda_m \leq \frac{2}{\pi} \log(m+1) + 1. \tag{5.4}$$

For two non-empty intervals I_1 and I_2 we define the two-sided interpolation operator $\mathfrak{I}^{(m)}_{I_1\times I_2} : C(I_1 \times I_2) \to \mathcal{P}_m(I_1) \otimes \mathcal{P}_m(I_2)$ by $\mathfrak{I}^{(m)}_{I_1\times I_2} := \mathfrak{I}^{(m)}_{I_1} \otimes \mathfrak{I}^{(m)}_{I_2}$. It acts on a function $f \in C(I_1 \times I_2)$ by

$$\mathfrak{I}^{(m)}_{I_1\times I_2}[f](t,\tau) = \sum_{j=0}^{m}\sum_{k=0}^{m} f(\xi^{(m)}_{I_1,j}, \xi^{(m)}_{I_2,k}) L^{(m)}_{I_1,j}(t) L^{(m)}_{I_2,k}(\tau).$$

We will use this interpolation operator to approximate the heat kernel (3.4) in the temporal variables t and τ.

For the approximation of the heat kernel (3.4) in the spatial variables $\boldsymbol{x}$, $\boldsymbol{y} \in \mathbb{R}^3$ we use a truncated Chebyshev expansion. The Chebyshev series of $f \in C([-1,1])$ is given by

$$\mathfrak{S}[f](x) = \sum_{j=0}^{\infty} c_j T_j(x) \quad \text{for all } x \in [-1,1], \quad c_j = \frac{\lambda_j}{\pi}\int_{-1}^{1} f(x) T_j(x) w(x)\,\mathrm{d}x, \tag{5.5}$$

where $w(x) := (1-x^2)^{-1/2}$ and $\lambda_j = 2 - \delta_{0j}$ with the Kronecker delta δ_{0j}. The partial sums of this series converge to f pointwise in $[-1,1]$ and also with respect to the norm in $L^2(-1,1)$; see e.g. [46, Chapter 5]. For an arbitrary interval I and functions in $C(I)$ an analogous series representation with transformed Chebyshev polynomials $T_{I,m}$ holds.

In higher dimensions we can consider related Chebyshev series by using tensor products of Chebyshev polynomials. For a box $[\boldsymbol{a},\boldsymbol{b}] := \prod_{j=1}^{3}[a_j,b_j] \subset \mathbb{R}^3$ we define the tensor product Chebyshev polynomials $T_{[\boldsymbol{a},\boldsymbol{b}],\boldsymbol{\kappa}}$ on $[\boldsymbol{a},\boldsymbol{b}]$ by

$$T_{[\boldsymbol{a},\boldsymbol{b}],\boldsymbol{\kappa}}(\boldsymbol{x}) := \prod_{j=1}^{3} T_{[a_j,b_j],\kappa_j}(x_j) \tag{5.6}$$

for all multi-indices $\boldsymbol{\kappa} \in \mathbb{N}_0^3$. Let $X_1 = [\boldsymbol{a},\boldsymbol{b}]$ and $X_2 = [\boldsymbol{c},\boldsymbol{d}]$ be two boxes in $\mathbb{R}^3$. A function $f \in C(X_1 \times X_2)$ can be represented by the Chebyshev series

$$\mathfrak{S}_{X_1\times X_2}[f](\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{\kappa}\in\mathbb{N}_0^3}\sum_{\boldsymbol{\nu}\in\mathbb{N}_0^3} f_{\boldsymbol{\kappa},\boldsymbol{\nu}} T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x}) T_{X_2,\boldsymbol{\kappa}}(\boldsymbol{y}), \tag{5.7}$$

$$f_{\boldsymbol{\kappa},\boldsymbol{\nu}} = \frac{\lambda_{\boldsymbol{\kappa}}\lambda_{\boldsymbol{\nu}}}{\pi^6}\int_{[-1,1]^3}\int_{[-1,1]^3} \hat{f}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) T_{[-1,1]^3,\boldsymbol{\kappa}}(\hat{\boldsymbol{y}}) T_{[-1,1]^3,\boldsymbol{\nu}}(\hat{\boldsymbol{x}}) w_{\mathrm{prod}}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})\,\mathrm{d}\hat{\boldsymbol{x}}\,\mathrm{d}\hat{\boldsymbol{y}}, \tag{5.8}$$

where $\lambda_{\boldsymbol{\kappa}} := \prod_{j=1}^{3}\lambda_{\kappa_j}$ and $w_{\mathrm{prod}}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) := \prod_{j=1}^{3}(w(\hat{x}_j)w(\hat{y}_j))$ for λ_j and w as in (5.5) and $\hat{f}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) := f(\boldsymbol{\varphi}_{X_1}(\hat{\boldsymbol{x}}),\boldsymbol{\varphi}_{X_2}(\hat{\boldsymbol{y}}))$ with the affine maps $\boldsymbol{\varphi}_{X_j} : [-1,1]^3 \to X_j$. An approximation of f is given by the truncated series

$$\mathfrak{S}^{(m)}_{X_1\times X_2}[f](\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{\kappa}:|\boldsymbol{\kappa}|\le m}\sum_{\boldsymbol{\nu}:|\boldsymbol{\kappa}+\boldsymbol{\nu}|\le m} f_{\boldsymbol{\kappa},\boldsymbol{\nu}} T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x}) T_{X_2,\boldsymbol{\kappa}}(\boldsymbol{y})$$

that includes only polynomials of total degree less than or equal to some $m \in \mathbb{N}_0$.

Let us now consider the heat kernel $((\boldsymbol{x},t),(\boldsymbol{y},\tau)) \mapsto G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)$ in (3.4) for $(\boldsymbol{x},t)$ and $(\boldsymbol{y},\tau)$ in axis-parallel, 4D boxes $Z_{\text{tar}} = X_1 \times I_1$ and $Z_{\text{src}} = X_2 \times I_2$, respectively. We focus on the case where the spatial parts X_1 and X_2 of the boxes are cubes with edge length $2\widetilde{h}_x$, i.e. $X_1 = [\boldsymbol{c}, \boldsymbol{c} + 2\widetilde{h}_x\mathbf{1}]$ and $X_2 = [\boldsymbol{d}, \boldsymbol{d} + 2\widetilde{h}_x\mathbf{1}]$ for some $\boldsymbol{c}$, $\boldsymbol{d} \in \mathbb{R}^3$. Furthermore, we assume that the time intervals $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ are such that $a_1 > b_2$ to ensure that the heat kernel is non-zero and bounded on $Z_{\text{tar}} \times Z_{\text{src}}$. By combining a two-sided interpolation in the temporal variables t and τ with a truncated Chebyshev expansion in the spatial variables $\boldsymbol{x}$ and $\boldsymbol{y}$ we can approximate the heat kernel G_α by

$$\begin{aligned}&\mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2}\mathfrak{I}^{(m_t)}_{I_1\times I_2}[G_\alpha](\boldsymbol{x},t,\boldsymbol{y},\tau)\\ &\qquad= \sum_{a,b=0}^{m_t} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu}\in\mathbb{N}_0^3:\\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|\le m_{\boldsymbol{x}}}} E^{a,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}\, T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x})\, T_{X_2,\boldsymbol{\kappa}}(\boldsymbol{y})\, L^{(m_t)}_{I_1,b}(t)\, L^{(m_t)}_{I_2,a}(\tau)\end{aligned} \tag{5.9}$$

for $(\boldsymbol{x},t) \in Z_{\text{tar}}$ and $(\boldsymbol{y},\tau) \in Z_{\text{src}}$. The coefficients $E^{a,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}$ correspond to the expansion coefficients in (5.8) when expanding the function $(\boldsymbol{x},\boldsymbol{y}) \mapsto G_\alpha(\boldsymbol{x}-\boldsymbol{y}, \xi^{(m_t)}_{I_1,b} - \xi^{(m_t)}_{I_2,a})$. They admit the product structure

$$E^{a,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}} := \frac{1}{(4\pi\alpha(\xi^{(m_t)}_{I_1,b} - \xi^{(m_t)}_{I_2,a}))^{3/2}} \prod_{j=1}^{3} E_{\kappa_j,\nu_j}(r_j, d_{a,b}), \text{ with} \tag{5.10a}$$

$$r_j := (c_j - d_j)/\widetilde{h}_x, \qquad d_{a,b} := 4\alpha(\xi^{(m_t)}_{I_1,b} - \xi^{(m_t)}_{I_2,a})/\widetilde{h}_x^2, \tag{5.10b}$$

$$E_{k,\ell}(r, d_{a,b}) := \frac{\lambda_k\lambda_l}{\pi^2} \int_{-1}^{1}\int_{-1}^{1} \exp\left(-\frac{|r+\hat{x}-\hat{y}|^2}{d_{a,b}}\right) T_\ell(\hat{x})T_k(\hat{y})w(\hat{x})w(\hat{y})\mathrm{d}\hat{x}\,\mathrm{d}\hat{y}, \tag{5.10c}$$

where λ_j and w are the same as in (5.5), cf. [68, Section 5.3, p. 209]. Note that there is not any closed formula known for the evaluation of the integrals in (5.10c). Therefore, we approximate them by a Gauß–Chebyshev quadrature rule in the application as proposed in [69, p. 3563] which yields

$$E_{k,\ell}(r, d_{a,b}) \approx \frac{\lambda_k\lambda_l}{(m_{\boldsymbol{x}}+1)^2} \sum_{n,m=0}^{m_{\boldsymbol{x}}} \exp\left(-\frac{|r + \xi^{(m_{\boldsymbol{x}})}_n - \xi^{(m_{\boldsymbol{x}})}_m|^2}{d_{a,b}}\right) T_\ell(\xi^{(m_{\boldsymbol{x}})}_n)T_k(\xi^{(m_{\boldsymbol{x}})}_m).$$

In the following sections we investigate the error that results when we approximate the heat kernel by (5.9).

5.1.1 Analysis of the interpolation error in time

The error that results when we interpolate the heat kernel (3.4) in two separated time intervals decays exponentially with increasing interpolation degree. This result

is stated, for example, in [67, Section 3.1] and is also proven in [51, p. 73]. In Theorem 5.1 we present the corresponding error estimate in a slightly more general form.

To simplify the notation we fix the heat capacity constant α and the spatial points $\boldsymbol{x}$ and $\boldsymbol{y}$, and set

$$g(t_1, t_2) := G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t_1 - t_2) = \frac{1}{(4\pi\alpha(t_1 - t_2))^{3/2}} \exp\left(-\frac{r^2}{4\alpha(t_1 - t_2)}\right) \tag{5.11}$$

for all $t_1 > t_2$ and $r^2 = |\boldsymbol{x} - \boldsymbol{y}|^2$. We interpolate g on suitable pairs of intervals $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ and use $|I_j| = b_j - a_j$ to denote their size and $\operatorname{dist}(I_1, I_2) := \min\{|t - \tau| : t \in I_1, \tau \in I_2\}$ for their distance.

THEOREM 5.1 (Two-sided interpolation error, cf. [51, p. 73]). *Let $\eta_2 \in \mathbb{R}_{>0}$ and $q_2 := 1 + 3/(2\,\eta_2)$. Let $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ be two non-empty intervals such that $a_1 > b_2$. Let the admissibility criterion*

$$\eta_2 \operatorname{dist}(I_1, I_2) \geq \max\{|I_1|, |I_2|\} \tag{5.12}$$

be satisfied. Then there exists a constant $c_2 > 0$ such that

$$\|g - \mathfrak{I}^{(m)}_{I_1 \times I_2}[g]\|_{\infty, I_1 \times I_2} \leq \frac{c_2}{(\alpha \operatorname{dist}(I_1, I_2))^{3/2}} q_2^{-(m+1)}. \tag{5.13}$$

The admissibility criterion (5.12) is a standard criterion used to identify pairs of sets on which so-called asymptotically smooth functions can be approximated well, cf. e.g. [35, Section 4.2.3]. In particular, interpolation error estimates like the one in Theorem 5.1 are well known for such functions. They can be defined as follows.

DEFINITION 5.2 (Asympotically smooth functions; see [35, Definition 4.14]). *Let $d \in \mathbb{N}$ and $X, Y \subset \mathbb{R}^d$. Let f be a function on $X \times Y$ such that f is arbitrarily often differentiable with respect to $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$ with $\boldsymbol{x} \neq \boldsymbol{y}$. Then f is called asymptotically smooth if there exists an $s \in \mathbb{R}$ and suitable constants C, ρ and γ such that*

$$|D^{\boldsymbol{\alpha}}_{\boldsymbol{x}} D^{\boldsymbol{\beta}}_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})| \leq h_{\mathrm{as}}(\boldsymbol{\alpha} + \boldsymbol{\beta})\, |\boldsymbol{x} - \boldsymbol{y}|^{-|\boldsymbol{\alpha}| - |\boldsymbol{\beta}| - s} \quad \text{with } h_{\mathrm{as}}(\boldsymbol{\nu}) := C \boldsymbol{\nu}! |\boldsymbol{\nu}|^\rho \gamma^{|\boldsymbol{\nu}|} \tag{5.14}$$

for all $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$ with $\boldsymbol{x} \neq \boldsymbol{y}$ and all multi-indices $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathbb{N}_0^d$ such that $\boldsymbol{\alpha} + \boldsymbol{\beta} \neq \boldsymbol{0}$.

The proof of Theorem 5.1 relies on the fact that g in (5.11) is asymptotically smooth, which we show next.

PROPOSITION 5.3 (cf. [51, p. 73]). *For the function g defined in (5.11) there exists a constant c_{as} such that*

$$|\partial_{t_1}^k \partial_{t_2}^\ell g(t_1,t_2)| \leq \frac{c_{\text{as}}\,(k+\ell)!(k+\ell+1)^{3/2}}{\alpha^{3/2}(t_1-t_2)^{k+\ell+3/2}} \quad \forall\, k,\ell \in \mathbb{N}_0 \text{ and all } t_1 > t_2. \tag{5.15}$$

In particular, g is asymptotically smooth on intervals I_1 and I_2 as in Theorem 5.1.

Proof. The proof follows the lines of [51, proof of Lemma 4.1], where an estimate analogous to (5.15) is shown for a similar function. Since g depends only on the difference of the variables t_1 and t_2 it suffices to bound $|\partial_{t_1}^n g(t_1,0)|$ for all $t_1 > 0$ to show (5.15). For $n = 0$ the estimate in (5.15) is clearly satisfied for all constants $c_{\text{as}} \geq 1/(4\pi)^{3/2}$. For the case $n \geq 1$ we consider the function $z \mapsto \widetilde{g}(z) := g(z,0)$, which is holomorphic on $\mathbb{C} \setminus \mathbb{R}_{\leq 0}$. Hence, we can use Cauchy's integral formula to get

$$\frac{\mathrm{d}^n}{\mathrm{d}z^n}\widetilde{g}(t_1) = \frac{n!}{2\pi i}\int_{\partial B(t_1,\delta)} \frac{\widetilde{g}(\zeta)}{(\zeta - t_1)^{(n+1)}}\mathrm{d}\zeta$$

for all $t_1 \in \mathbb{R}_{>0}$ and $\delta < t_1$, where $B(t_1,\delta) \subset \mathbb{C}$ is the ball with center t_1 and radius δ. For all $\zeta \in \partial B(t_1,\delta)$ there holds $|\zeta| \geq \Re(\zeta) > (t_1 - \delta)$ and $\Re(-r^2/(4\alpha\zeta)) < 0$, so it follows that

$$|\widetilde{g}(\zeta)| = \frac{1}{(4\pi\alpha)^{3/2}}\left|\frac{1}{\zeta^{3/2}}\exp\left(-\frac{r^2}{4\alpha\zeta}\right)\right| \leq \frac{1}{(4\pi\alpha)^{3/2}}\frac{1}{(t_1-\delta)^{3/2}}.$$

Therefore, we can estimate

$$\begin{aligned}\left|\frac{\mathrm{d}^n}{\mathrm{d}z^n}\widetilde{g}(t_1)\right| &\leq \frac{n!}{2\pi}\int_{\partial B(t_1,\delta)} \frac{|\widetilde{g}(\zeta)|}{|\zeta - t_1|^{(n+1)}}\mathrm{d}\zeta \\ &\leq \frac{n!}{2\pi}\int_{\partial B(t_1,\delta)} \frac{1}{(4\pi\alpha(t_1-\delta))^{3/2}\delta^{(n+1)}}\mathrm{d}\zeta = \frac{n!}{(4\pi\alpha(t_1-\delta))^{3/2}\delta^n}.\end{aligned}$$

The optimal radius δ^* to minimize this bound is $\delta^* = 2nt_1/(2n+3)$. By inserting δ^* in the previous estimate we get

$$\left|\frac{\mathrm{d}^n}{\mathrm{d}z^n}\widetilde{g}(t_1)\right| \leq \frac{n!}{(4\pi\alpha)^{3/2}}\left(\frac{2n+3}{3}\right)^{3/2}\left(1+\frac{3}{2n}\right)^n \frac{1}{t_1^{n+3/2}} \leq \frac{e^{3/2}}{(4\pi)^{3/2}}\frac{n!(n+1)^{3/2}}{\alpha^{3/2}t_1^{n+3/2}}.$$

This yields (5.15) for all $n \geq 1$ with $c_{\text{as}} = e^{3/2}/(4\pi)^{3/2}$. The asymptotical smoothness of g on $I_1 \times I_2$ with I_1 and I_2 as in Theorem 5.1 follows directly from (5.15). □

Proof of Theorem 5.1. According to [14, Corollary 4.21] it suffices to show that there exist $C_g \in \mathbb{R}_{\geq 0}$, $\gamma_g \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{N}$ such that

$$\|\partial_{t_k}^n g\|_{\infty, I_1\times I_2} \leq \frac{C_g}{\gamma_g^n}\frac{(n+\sigma-1)!}{(\sigma-1)!} \qquad \forall n \in \mathbb{N}_0 \text{ and } k \in \{1,2\} \tag{5.16}$$

to conclude that

$$\begin{aligned}&\|g-\mathfrak{I}^{(m)}_{I_1\times I_2}[g]\|_{\infty,I_1\times I_2} \\ &\quad\le 4eC_g(\Lambda_m+1)^2(m+2)^\sigma\left(1+\frac{\max\{|I_1|,|I_2|\}}{\gamma_g}\right)\varrho\left(\frac{2\gamma_g}{\max\{|I_1|,|I_2|\}}\right)^{-(m+1)},\end{aligned} \tag{5.17}$$

where $\varrho(r) := r+\sqrt{1+r^2}$ and Λ_m is the Lebesgue constant defined in (5.3). From the estimate (5.15) and the admissibility criterion (5.12) it follows that

$$\begin{aligned}\|\partial_{t_k}^n g\|_{\infty,I_1\times I_2} &\le \max_{(t_1,t_2)\in I_1\times I_2}\frac{c_{\mathrm{as}}\,n!(n+1)^{3/2}}{\alpha^{3/2}(t_1-t_2)^{n+3/2}} = \frac{c_{\mathrm{as}}\,n!(n+1)^{3/2}}{\alpha^{3/2}\operatorname{dist}(I_1,I_2)^{n+3/2}} \\ &\le \frac{c_{\mathrm{as}}\,n!\,c_\beta\beta^n}{\alpha^{3/2}\operatorname{dist}(I_1,I_2)^{3/2}}\left(\frac{\eta_2}{\max\{|I_1|,|I_2|\}}\right)^n,\end{aligned}$$

where $\beta>1$ is arbitrary and the constant c_β is chosen such that $(n+1)^{3/2}\le c_\beta\beta^n$ for all $n\ge 0$. Thus, (5.16) is satisfied for the choice $\sigma=1$, $\gamma_g=\max\{|I_1|,|I_2|\}/(\eta_2\,\beta)$ and $C_g = c_{\mathrm{as}}c_\beta/(\alpha\operatorname{dist}(I_1,I_2))^{3/2}$. Since $c_\beta\to\infty$ as $\beta\to 1$ we fix $\beta=4/3$ and a proper constant c_β. Inserting these choices of σ, γ_g and C_g into (5.17) yields

$$\|g-\mathfrak{I}^{(m)}_{I_1\times I_2}[g]\|_{\infty,I_1\times I_2}\le\frac{4ec_{\mathrm{as}}c_\beta(\Lambda_m+1)^2(m+2)}{(\alpha\operatorname{dist}(I_1,I_2))^{3/2}}\left(1+\frac{4}{3}\eta_2\right)\varrho\left(\frac{3}{2\eta_2}\right)^{-(m+1)}.$$

Similarly as in [14, Remark 4.23], we want to simplify this estimate. For this purpose, we set $q_2=1+3/(2\,\eta_2)$, and $\mu_2=\varrho\big(3/(2\,\eta_2)\big)/q_2$. Since $\varrho(r)>1+r$ for all $r\in\mathbb{R}$ we see that $\mu_2>1$. Thus, we can rewrite the right-hand side of the above estimate as

$$\frac{4ec_{\mathrm{as}}c_\beta(\Lambda_m+1)^2(m+2)(1+4\,\eta_2/3)}{\mu_2^{m+1}}\frac{1}{(\alpha\operatorname{dist}(I_1,I_2))^{3/2}}q_2^{-(m+1)}.$$

The first fraction is a zero sequence in m and can thus be bounded by a constant c_2. In fact, its numerator grows like $\log(m+1)^2(m+2)$ due to (5.4), while its denominator grows exponentially with increasing m. Therefore, we end up with the estimate in (5.13). □

Remark 5.4. *The result in Theorem 5.1 holds for general intervals $I_1=[a_1,b_1]$ and $I_2=[a_2,b_2]$ with $a_1>b_2$ as long as they are separated. However, the convergence rate q_2 deteriorates as the constant η_2 in the admissibility criterion (5.12) increases. The smallest value η_2 for two given intervals I_1 and I_2 for which (5.12) holds is $\eta_2=\max\{|I_1|,|I_2\}/\operatorname{dist}(I_1,I_2)$. This ratio will always be reasonably bounded in the applications in this chapter.*

5.1.2 Analysis of the approximation error in space

The approximation error of the truncated Chebyshev expansion of the heat kernel for a fixed temporal setting has been investigated in [51, 68]. Here we collect and slightly improve those results. We start by citing an estimate of the absolute value of the expansion coefficients defined in (5.10c).

LEMMA 5.5 (cf. [68, Lemma 1]). *For the coefficients $E_{k,\ell}(r,\delta)$ in (5.10c) there holds*

$$|E_{k,\ell}(r,\delta)| \leq \frac{\lambda_k \lambda_\ell}{a^{k+\ell}} \exp\left(\frac{1}{\delta}\left(a - \frac{1}{a}\right)^2\right), \tag{5.18}$$

where $a \in \mathbb{R}_{>0}$ can be chosen arbitrarily.

REMARK 5.6. *The estimate in (5.18) is slightly sharper than the one in the referenced paper where an additional multiplicative factor 4 is included in the argument of the exponential function. This factor can be dropped in the last step of the proof in [68]; see [39, Satz 4.13].*

THEOREM 5.7 (Chebyshev series truncation error, cf. [68, p. 209]). *Let $\widetilde{c}_{\mathrm{st}} \in \mathbb{R}_{>0}$. Let X_1 and X_2 be two cubes with edge length $2\widetilde{h}_x$. Let $t, \tau \in \mathbb{R}$ such that $t > \tau$ and*

$$\frac{4\alpha(t-\tau)}{\widetilde{h}_x^2} \geq \widetilde{c}_{\mathrm{st}}. \tag{5.19}$$

Let the function $\sigma : \mathbb{R}_{>0} \to \mathbb{R}$ be defined by

$$\sigma(s) := \frac{1}{2}\left(\ln\left(s + \sqrt{1+s^2}\right) - \frac{\sqrt{s^2+1}-1}{s}\right). \tag{5.20}$$

Then there exists a constant $c_{\boldsymbol{x}} > 0$ such that

$$\begin{aligned} &\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1 \times X_2})[G_\alpha(\cdot_{\boldsymbol{x}} - \cdot_{\boldsymbol{y}}, t-\tau)]\|_{\infty, X_1 \times X_2} \\ &\qquad \leq c_{\boldsymbol{x}} \frac{(m_{\boldsymbol{x}}+2)^5}{(\alpha(t-\tau))^{3/2}} \exp\Big(-(m_{\boldsymbol{x}}+1)\,\sigma\Big((m_{\boldsymbol{x}}+1)\widetilde{c}_{\mathrm{st}}/12\Big)\Big). \end{aligned} \tag{5.21}$$

Proof. The proof is based on the proof of [51, Corollary 4.1] and the idea to minimize the right-hand side of (5.18) in [69, p. 3551], where the function κ corresponds to σ here. To estimate the error we represent it as the remainder of the Chebyshev series, namely

$$(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1 \times X_2})[G_\alpha(\cdot_{\boldsymbol{x}} - \cdot_{\boldsymbol{y}}, t-\tau)](\boldsymbol{x}, \boldsymbol{y}) = \sum_{n=m_{\boldsymbol{x}}+1}^{\infty} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu} \in \mathbb{N}_0^3: \\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|=n}} E^-_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t,\tau) T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x}) T_{X_2,\boldsymbol{\kappa}}(\boldsymbol{y}),$$

where $E^-_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t,\tau) = (4\pi\alpha(t-\tau))^{-3/2}\prod_{j=1}^3 E_{\kappa_j,\nu_j}(r_j,\delta)$ with $\delta = 4\alpha(t-\tau)/\widetilde{h}_x^2$ and r_j and $E_{\kappa_j,\nu_j}(r_j,\delta)$ from (5.10). The polynomials $T_{X_1,\boldsymbol{\nu}}$ and $T_{X_2,\boldsymbol{\kappa}}$ are tensor products of transformed Chebyshev polynomials as defined in (5.6). Thus, there holds $|T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x})| \leq 1$ for all $\boldsymbol{x} \in X_1$ and all $\boldsymbol{\nu} \in \mathbb{N}_0^3$ and the same for $T_{X_2,\boldsymbol{\kappa}}$ in X_2. Hence, we can estimate

$$\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})[G_\alpha(\cdot_{\boldsymbol{x}} - \cdot_{\boldsymbol{y}}, t-\tau)]\|_{\infty,X_1\times X_2} \leq \sum_{n=m_{\boldsymbol{x}}+1}^{\infty} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu}\in\mathbb{N}_0^3:\\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|=n}} |E^-_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t,\tau)|. \tag{5.22}$$

For a given n and $\boldsymbol{\kappa}$, $\boldsymbol{\nu} \in \mathbb{N}_0^3$ such that $|\boldsymbol{\kappa}+\boldsymbol{\nu}| = n$ the estimate in (5.18) yields

$$\begin{aligned}(4\pi\alpha(t-\tau))^{3/2}|E^-_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t,\tau)| &\leq \exp\left(\frac{1}{\delta}\left(a_n - \frac{1}{a_n}\right)^2\right)\prod_{j=1}^3 \frac{\lambda_{\kappa_j}\lambda_{\nu_j}}{a_n^{\kappa_j+\nu_j}}\\ &\leq \frac{64}{a_n^n}\exp\left(\frac{3}{\delta}\left(a_n - \frac{1}{a_n}\right)^2\right),\end{aligned}$$

where $a_n \in \mathbb{R}_{>0}$ can be chosen arbitrarily. To find the optimal a_n we minimize the function f_n on the right-hand side, which can be rewritten in the form

$$f_n(a_n) = 64\exp\left(-n\ln(a_n) + \frac{3}{\delta}\left(a_n - \frac{1}{a_n}\right)^2\right).$$

From the necessary condition $f_n'(a_n^*) = 0$ we obtain $a_n^* = \sqrt{n\delta/12 + \sqrt{1+(n\delta/12)^2}}$, which is indeed a minimizer of f_n in $\mathbb{R}_{>0}$ since f_n becomes unbounded for a_n tending to zero or infinity. Therefore, the minimum of f_n is given by

$$f_n(a_n^*) = 64\exp\left(-n\left(\ln(a_n^*) - \frac{12}{4n\delta}\left(a_n^* - \frac{1}{a_n^*}\right)^2\right)\right) = 64\exp\big(-n\sigma\big(n\delta/12\big)\big)$$

with the function σ from (5.20) and we obtain

$$|E^-_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t,\tau)| \leq \frac{64}{(4\pi\alpha(t-\tau))^{3/2}}\exp\big(-n\sigma\big(n\delta/12\big)\big).$$

By using this estimate and $\#\{(\boldsymbol{\kappa},\boldsymbol{\nu}) \in \mathbb{N}_0^3\times\mathbb{N}_0^3 : |\boldsymbol{\kappa}+\boldsymbol{\nu}| = n\} = \binom{n+5}{5}$ we can further estimate the approximation error in (5.22) by

$$\begin{aligned}&\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})[G_\alpha(\cdot_{\boldsymbol{x}} - \cdot_{\boldsymbol{y}}, t-\tau)]\|_{\infty,X_1\times X_2} \\ &\qquad\leq \frac{64}{(4\pi\alpha(t-\tau))^{3/2}}\sum_{n=m_{\boldsymbol{x}}+1}^{\infty}\binom{n+5}{5}\exp\big(-n\sigma\big(n\delta/12\big)\big).\end{aligned} \tag{5.23}$$

The function σ is monotonically increasing. In fact, one can show that $\sigma'(s) > 0$ for all $s > 0$. Hence, we can use (5.19), which corresponds to the estimate $\delta = 4\alpha(t-\tau)/\widetilde{h}_x^2 \geq \widetilde{c}_{\mathrm{st}}$, to get for all $n > m_{\boldsymbol{x}}$

$$\exp\big(-n\sigma\big(n\delta/12\big)\big) \leq \exp\Big(-n\sigma\Big((m_{\boldsymbol{x}}+1)\widetilde{c}_{\mathrm{st}}/12\Big)\Big) = q^n,$$

where $q := \exp\big(-\sigma\big((m_{\boldsymbol{x}}+1)\widetilde{c}_{\mathrm{st}}/12\big)\big)$. In particular, the series on the right-hand side of (5.23) is bounded by $\sum_{n=m_{\boldsymbol{x}}+1}^{\infty}\binom{n+5}{5}q^n$. There holds

$$\begin{aligned}\sum_{n=m_{\boldsymbol{x}}+1}^{\infty}\binom{n+5}{5}q^n &= q^{m_{\boldsymbol{x}}+1}\sum_{n=0}^{\infty}\binom{n+5+m_{\boldsymbol{x}}+1}{5}q^n \\ &\leq q^{m_{\boldsymbol{x}}+1}(m_{\boldsymbol{x}}+2)^5\sum_{n=0}^{\infty}\binom{n+5}{5}q^n = q^{m_{\boldsymbol{x}}+1}\frac{(m_{\boldsymbol{x}}+2)^5}{(1-q)^6}, \qquad (5.24)\end{aligned}$$

where we used the estimate $(n+k+m_{\boldsymbol{x}}+1) \leq (n+k)(m_{\boldsymbol{x}}+2)$ for all $n \geq 0$ and $k \geq 1$. Note that the last identity in (5.24) follows immediately by identifying $5!(1-q)^{-6}$ as the fifth derivative of the geometric series $\sum_{n=0}^{\infty} q^n$. Since q depends on $m_{\boldsymbol{x}}$ but converges to 0 as $m_{\boldsymbol{x}} \to \infty$, we can bound the term $(1-q)^{-6}$ by a constant c for all $m_{\boldsymbol{x}} \geq 0$. Together with (5.23) and (5.24) this yields

$$\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})[G_\alpha(\cdot_{\boldsymbol{x}} - \cdot_{\boldsymbol{y}}, t-\tau)]\|_{\infty, X_1\times X_2} \leq \frac{64}{(4\pi\alpha(t-\tau))^{3/2}}c\,(m_{\boldsymbol{x}}+2)^5 q^{m_{\boldsymbol{x}}+1},$$

which is (5.21) with $c_{\boldsymbol{x}} = 64c/(4\pi)^{3/2}$. □

Remark 5.8. *The behavior of the function σ in (5.20) determines the approximation error in (5.21). In the proof of Theorem 5.7 we have already pointed out that σ is monotonically increasing. Furthermore there holds $\sigma(s) \sim s/4$ for $s \to 0$ and $\sigma(s) \sim \ln(2s)/2$ for $s \to \infty$; see [69, p. 3551]. Thus, the convergence in (5.21) is super-exponential in $m_{\boldsymbol{x}}$. Note that the approximation quality depends on the constant $\widetilde{c}_{\mathrm{st}}$ in (5.19), which appears in the argument of σ. For small values of $\widetilde{c}_{\mathrm{st}}$ the estimate suffers. Thus, we have to ensure that (5.19) holds for a reasonably large constant $\widetilde{c}_{\mathrm{st}}$ in the later application. This means that we have to adapt the size of the cubes X_1 and X_2 to the temporal configuration.*

5.1.3 The space-time approximation error

By combining the results from the previous two sections we can estimate the approximation error of the expansion (5.9) in $Z_{\mathrm{tar}} \times Z_{\mathrm{src}}$.

THEOREM 5.9 (Full space-time expansion error, [51, cf. Lemma 7.4]). *Let $\widetilde{c}_{\mathrm{st}} \in \mathbb{R}_{>0}$, $\eta_2 \in \mathbb{R}_{>0}$ and $q_2 := 1 + 3/(2\,\eta_2)$. Let Λ_{m_t} be the Lebesgue constant defined in (5.3) and σ be the function in (5.20). Let $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ be two non-empty intervals with $a_1 > b_2$ that satisfy the admissibility criterion (5.12). Let X_1 and X_2 be two cubes in $\mathbb{R}^3$ with edge length $2\widetilde{h}_x$ such that*

$$\frac{4\alpha\,\mathrm{dist}(I_1, I_2)}{\widetilde{h}_x^2} \geq \widetilde{c}_{\mathrm{st}}. \tag{5.25}$$

Let $Z_{\mathrm{tar}} = X_1 \times I_1$ and $Z_{\mathrm{src}} = X_2 \times I_2$. Then there exist constants c_2, $c_{\boldsymbol{x}} \in \mathbb{R}_{>0}$ such that

$$\begin{aligned}\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2}\mathfrak{I}^{(m_t)}_{I_1\times I_2})[G_\alpha]\|_{\infty,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}} &\leq \frac{1}{(\alpha\,\mathrm{dist}(I_1,I_2))^{3/2}} \\ &\times \Big(c_2 q_2^{-(m_t+1)} + c_{\boldsymbol{x}}\Lambda_{m_t}^2 (m_{\boldsymbol{x}}+2)^5 \exp\Big(-(m_{\boldsymbol{x}}+1)\sigma\Big((m_{\boldsymbol{x}}+1)\widetilde{c}_{\mathrm{st}}/12\Big)\Big)\Big).\end{aligned} \tag{5.26}$$

Proof. Throughout this proof we use $\|\cdot\|_\infty$ to denote $\|\cdot\|_{\infty,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ to shorten the notation. We start to estimate the approximation error by adding and subtracting $\mathfrak{I}^{(m_t)}_{I_1\times I_2}[G_\alpha]$ and using the triangle inequality to get

$$\begin{aligned}&\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2}\mathfrak{I}^{(m_t)}_{I_1\times I_2})[G_\alpha]\|_\infty \\ &\qquad \leq \|(\mathrm{Id} - \mathfrak{I}^{(m_t)}_{I_1\times I_2})[G_\alpha]\|_\infty + \|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})\mathfrak{I}^{(m_t)}_{I_1\times I_2}[G_\alpha]\|_\infty.\end{aligned}$$

The first term can be further estimated by using Theorem 5.1. In fact, the function g defined in (5.11) which is considered in that theorem is just the heat kernel for fixed spatial points $\boldsymbol{x}$ and $\boldsymbol{y}$ and the estimate does not depend on these points. Thus,

$$\|(\mathrm{Id} - \mathfrak{I}^{(m_t)}_{I_1\times I_2})[G_\alpha]\|_\infty \leq \frac{c_2}{(\alpha\,\mathrm{dist}(I_1,I_2))^{3/2}} q_2^{-(m_t+1)}$$

with c_2 from Theorem 5.1. For the second term we observe that

$$\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})\mathfrak{I}^{(m_t)}_{I_1\times I_2}[G_\alpha]\|_\infty \leq \sup_{f\in C(Z_{\mathrm{tar}}\times Z_{\mathrm{src}})\setminus\{0\}} \frac{\|\mathfrak{I}^{(m_t)}_{I_1\times I_2}[f]\|_\infty}{\|f\|_\infty}\,\|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})[G_\alpha]\|_\infty,$$

where we used that the operators $\mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2}$ and $\mathfrak{I}^{(m_t)}_{I_1\times I_2}$ commute. The operator norm of $\mathfrak{I}^{(m_t)}_{I_1\times I_2}$ is bounded by $\Lambda^2_{m_t}$, which follows from (5.3). For the other term we use assumption (5.25) and apply Theorem 5.7 to get

$$\begin{aligned}&\|(\mathrm{Id}-\mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1\times X_2})[G_\alpha]\|_\infty \\ &\qquad \leq \sup_{(t,\tau)\in I_1\times I_2}\left\{c_{\boldsymbol{x}}\frac{(m_{\boldsymbol{x}}+2)^5}{(\alpha(t-\tau))^{3/2}}\exp\Big(-(m_{\boldsymbol{x}}+1)\,\sigma\Big((m_{\boldsymbol{x}}+1)\widetilde{c}_{\mathrm{st}}/12\Big)\Big)\right\} \\ &\qquad \leq c_{\boldsymbol{x}}\frac{(m_{\boldsymbol{x}}+2)^5}{(\alpha\,\mathrm{dist}(I_1,I_2))^{3/2}}\exp\Big(-(m_{\boldsymbol{x}}+1)\,\sigma\Big((m_{\boldsymbol{x}}+1)\widetilde{c}_{\mathrm{st}}/12\Big)\Big).\end{aligned}$$

Combining all these estimates completes the proof. □

Theorem 5.9 shows that we can control the approximation error of the expansion (5.9) if we can control the respective errors in time and space separately. The spatial sizes of the boxes Z_{tar} and Z_{src} have to be adapted to the distance of their temporal components according to the criterion (5.25).

REMARK 5.10. *In [51, 67, 68, 76] a criterion different from* (5.25) *is used to determine the proper spatial size of the boxes for the expansion* (5.9)*. There it is required that there exists a constant* c_{st} *such that*

$$\frac{\widetilde{h}_x^2}{4\alpha\,\widetilde{h}_t} \leq c_{\text{st}} \tag{5.27}$$

for each space-time box, where $\widetilde{h}_x$ *denotes the half length of the edges of its spatial component and* $\widetilde{h}_t$ *the half length of its temporal interval. However, criterion* (5.25) *follows from* (5.27) *for two boxes* $Z_{\text{tar}} = X_1 \times I_1$ *and* $Z_{\text{src}} = X_2 \times I_2$*, if the admissibility criterion* (5.12) *holds in addition. Indeed,*

$$\frac{4\alpha\,\operatorname{dist}(I_1, I_2)}{\widetilde{h}_x^2} \geq \frac{4\alpha \max\{|I_1|, |I_2|\}}{\widetilde{h}_x^2 \eta_2} \geq \frac{2}{c_{\text{st}}\eta_2}, \tag{5.28}$$

which is (5.25) *with* $\widetilde{c}_{st} = 2(c_{\text{st}}\eta_2)^{-1}$*. Therefore, we use the space-time configuration criterion* (5.27) *instead of* (5.25) *in the rest of this work to allow for easier comparison with previous articles.*

5.2 Description of the space-time FMM

In this section we describe the space-time FMM for the fast computation of matrix-vector products for BEM matrices of the heat equation like V_h. As stated at the beginning of this chapter, the space-time FMM here is closely related to the pFMM in [52, 67, 68], but realizes the full matrix-vector product at once instead of in a forward-sweeping manner. The description in this section is similar to those in [76] and [78]. We consider a space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ of the lateral space-time boundary Σ of a space-time cylinder Q as in Section 2.4 and the BEM matrices from Section 3.3. As a first step, we create a hierarchy of space-time boxes that subdivides the elements in Σ_h; see Section 5.2.1. This hierarchy, which is often denoted as a box cluster tree, will allow us to find a suitable partition of a BEM matrix into blocks that can be approximated using the expansion in Section 5.1, and a small remainder of non-approximable blocks. For the partitioning process we construct suitable operation lists in Section 5.2.2. While these lists are independent of the considered BEM matrices V_h, K_h, $\mathsf{K}_h^{\top_x}$, or D_h, the related approximation of admissible blocks is not. In Section 5.2.3 we describe the approximation of these

blocks and the corresponding FMM operations. Additional nested FMM operations which are needed for higher efficiency are discussed in Section 5.2.4 together with the resulting space-time FMM.

5.2.1 A 4D space-time box cluster tree

In this section we construct a hierarchy of 4D boxes to partition a given space-time tensor product mesh Σ_h. The construction is based on a recursive subdivision of an initial box $Z^{(0)} \subset \mathbb{R}^4$ that contains the whole mesh. The resulting structure is denoted as a box cluster tree $\mathcal{T}_\Sigma$. Similar trees have already been considered in [51, 67, 68]. In those works separate trees for the spatial mesh Γ_h and the temporal partition $\mathcal{I}_{h_t}$ were constructed first and then combined to obtain a space-time box cluster tree. The direct subdivision of 4D boxes, which we consider here, is a more general approach applicable also to space-time meshes without a strict tensor product structure.

Note that from here on we consider half-open boxes $Z = (\boldsymbol{a}, \boldsymbol{a} + 2\widetilde{h}_{\boldsymbol{x}}\mathbf{1}] \times (c, d] \subset \mathbb{R}^4$ since they allow for a simple, non-overlapping partition into smaller half-open boxes. The spatial parts of boxes Z will be cubes throughout this work. By $\widetilde{h}_{\boldsymbol{x}}(Z)$ we denote the half edge length of the spatial part of Z and by $\widetilde{h}_t(Z)$ the half length of the temporal part of Z. We implicitly assign a space-time boundary element $\sigma_{j_t, j_{\boldsymbol{x}}} = \gamma_{j_{\boldsymbol{x}}} \times (t_{j_t - 1}, t_{j_t})$ of the given space-time tensor product mesh Σ_h to a box Z if the geometrical center of $\sigma_{j_t, j_{\boldsymbol{x}}}$ is contained in Z. By $\hat{Z}$ we denote the set of all indices $(j_t, j_{\boldsymbol{x}})$ corresponding to elements $\sigma_{j_t, j_{\boldsymbol{x}}} \in \Sigma_h$ that are assigned to Z. The cardinality of $\hat{Z}$ is denoted by $\#\hat{Z}$.

Algorithm 5.1 describes the recursive construction of a space-time box cluster tree $\mathcal{T}_\Sigma$ for a mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$. In the algorithm it is assumed that an initial box $Z^{(0)} = (\boldsymbol{a}, \boldsymbol{a} + 2\widetilde{h}_x^{(0)}\mathbf{1}] \times (0, T]$ is already given. The spatial part of this box can be constructed, for example, by determining the minimal axis-parallel bounding box containing Γ_h and extending it to a half-open cube that contains Γ_h. The box $Z^{(0)}$ is the root of the tree $\mathcal{T}_\Sigma$. It is subdivided into smaller boxes which are added to $\mathcal{T}_\Sigma$ as children of $Z^{(0)}$ and further subdivided recursively. If an empty box Z_k is obtained in the recursive subdivision process, i.e. a box such that $\#\hat{Z}_k = 0$, it is not added to $\mathcal{T}_\Sigma$ and not further subdivided. The subdivision process is also stopped for a box Z if the number $\#\hat{Z}$ of space-time elements assigned to Z is lower than a certain number $n_{\max}$. This number $n_{\max}$ has to be provided as an input argument for the algorithm as well as the constant $c_{\mathrm{st}} > 0$ for the space-time configuration criterion (5.27). The concrete subdivision of a box $Z = (\boldsymbol{a}, \boldsymbol{b}] \times (c, d]$ in lines 7 and 9 is executed as follows:

(i) Purely temporal subdivision (line 7): We split the time interval $(c, d]$ into $(c, \widetilde{c}]$ and $(\widetilde{c}, d]$, where $\widetilde{c}$ is the grid point t_j of the time partition $\mathcal{I}_{h_t}$ closest to the

midpoint $(c+d)/2$. Then we subdivide Z into the two boxes $Z_1 = (\boldsymbol{a}, \boldsymbol{b}] \times (c, \widetilde{c}]$ and $Z_2 = (\boldsymbol{a}, \boldsymbol{b}] \times (\widetilde{c}, d]$.

(ii) Space-time subdivision (line 9): We subdivide the time interval $(c, d]$ as in (i). The spatial box $(\boldsymbol{a}, \boldsymbol{b}]$ is uniformly split into 8 boxes $(\boldsymbol{a}, \widetilde{\boldsymbol{a}}], \ldots, (\widetilde{\boldsymbol{a}}, \boldsymbol{b}]$, where $\widetilde{\boldsymbol{a}} = 1/2\,(\boldsymbol{a}+\boldsymbol{b})$ is the center of $(\boldsymbol{a}, \boldsymbol{b}]$. By considering all possible combinations of the resulting time intervals and spatial boxes we get a subdivision of Z into 16 space-time boxes.

Space-time boxes are only subdivided if it is *feasible*. We say that a temporal subdivision of $Z = X \times I$ is feasible, if the interval I contains more than one time step of the partition $\mathcal{I}_{h_t}$. A spatial subdivision of Z is feasible, if $\max_{(j_t, j_{\boldsymbol{x}}) \in \hat{Z}} \operatorname{diam}(\gamma_{j_{\boldsymbol{x}}}) \leq \widetilde{h}_{\boldsymbol{x}}(Z)$, where $\operatorname{diam}(\gamma_{j_{\boldsymbol{x}}}) := \max\{|\boldsymbol{x} - \boldsymbol{y}| : \boldsymbol{x}, \boldsymbol{y} \in \gamma_{j_{\boldsymbol{x}}}\}$ is the diameter of the spatial element $\gamma_{j_{\boldsymbol{x}}}$. This means that we do not refine a box in space if it contains a space-time element whose spatial size is larger than the spatial half length of the box itself. A space-time subdivision of $Z = X \times I$ is feasible if the temporal subdivision and the spatial subdivision of Z are feasible.

Algorithm 5.1 Construction of a 4D space-time box cluster tree $\mathcal{T}_\Sigma$ for Σ_h.

Require: Let a space-time tensor product mesh Σ_h be given.
Let $n_{\max}$ be a bound for the number of elements in a leaf box.
Let $c_{\mathrm{st}} > 0$ for the space-time configuration criterion (5.27) be given.
Let $Z^{(0)} = (\boldsymbol{a}, \boldsymbol{a} + 2\widetilde{h}_x^{(0)}\mathbf{1}] \times (0, T] \subset \mathbb{R}^4$ be given such that Σ_h is contained in $Z^{(0)}$ and $\widetilde{h}_x^{(0)}$ and $\widetilde{h}_t(Z^{(0)}) = T/2$ satisfy (5.27).

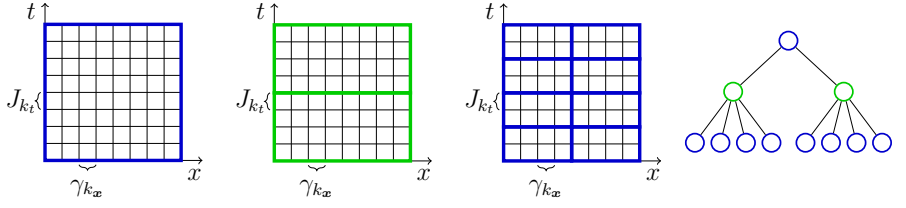
1: Construct an empty tree $\mathcal{T}_\Sigma$ and add $Z^{(0)}$ as its root.
2: Call SubdivideCluster($Z^{(0)}$, $\mathcal{T}_\Sigma$)
3: **function** SubdivideCluster(Z, $\mathcal{T}_\Sigma$)
4: **if** $\#\hat{Z} \geq n_{\max}$
5: Let $\ell_t = \ell_t(Z)$ be the temporal level of Z, $\widetilde{h}_t^{(\ell_t+1)} = 2^{-\ell_t - 1}\widetilde{h}_t(Z^{(0)})$
6: **if** $\widetilde{h}_t^{(\ell_t+1)}$ and $\widetilde{h}_x(Z)$ satisfy (5.27) **and** a temporal subdivision is feasible
7: Temporal subdivision of Z into $n_{\mathrm{C}} = 2$ children $\{Z_k\}_{k=1}^{n_{\mathrm{C}}}$.
8: **else if** a space-time subdivision is feasible
9: Space-time subdivision of Z into $n_{\mathrm{C}} = 16$ children $\{Z_k\}_{k=1}^{n_{\mathrm{C}}}$.
10: **else**
11: **return**. *// No subdivision is feasible. Stop the recursion.*
12: **for** $k = 1, \ldots, n_{\mathrm{C}}$
13: **if** $\#\hat{Z}_k \neq 0$
14: Add Z_k to $\mathcal{T}_\Sigma$ as a child of Z and call SubdivideCluster(Z_k, $\mathcal{T}_\Sigma$).

In the rest of this work we will use the terms box and cluster interchangeably when talking about boxes in $\mathcal{T}_\Sigma$. In addition, we use standard terms of graph theory

related to rooted trees: We denote the *parent* of a box $Z \in \mathcal{T}_\Sigma$ by $\mathrm{par}(Z)$, the set of all its *children* by $\mathrm{child}(Z)$ and its *level* by $\ell(Z)$, which is the length of the path between Z and the root of $\mathcal{T}_\Sigma$ in this tree. The largest level in the tree $\mathcal{T}_\Sigma$ is denoted by $\mathrm{depth}(\mathcal{T}_\Sigma)$. In addition, we define the *temporal level* $\ell_t(Z)$ of a box Z in $\mathcal{T}_\Sigma$ by setting $\ell_t(Z^{(0)}) := 0$ for the root $Z^{(0)}$ and defining $\ell_t(Z)$ for all other $Z \in \mathcal{T}_\Sigma$ as the number of performed temporal subdivisions starting from $Z^{(0)}$ to obtain Z. This temporal level is used in line 5 of Algorithm 5.1. Similarly, we define the *spatial level* $\ell_{\boldsymbol{x}}(Z)$ of a box Z in $\mathcal{T}_\Sigma$ by setting $\ell_{\boldsymbol{x}}(Z^{(0)}) := 0$ and defining $\ell_{\boldsymbol{x}}(Z)$ for all other $Z \in \mathcal{T}_\Sigma$ as the number of performed spatial subdivisions starting from $Z^{(0)}$ to obtain Z.

A simple illustration of the recursive subdivision process in Algorithm 5.1 is given in Figure 5.1. In general, we switch between purely temporal and space-time subdivisions of boxes in the tree construction due to criterion (5.27) — at least after some initial, purely temporal subdivisions which might be sufficient to guarantee (5.27) for the first tree levels. Note that we assume that the criterion (5.27) is already satisfied for $Z^{(0)}$, or rather $\widetilde{h}_{\boldsymbol{x}}(Z^{(0)})$ and $\widetilde{h}_t(Z^{(0)})$. If this is not the case, we can subdivide $Z^{(0)}$ in space as often as necessary to obtain boxes $\{Z_k\}_{k=1}^{N_{\mathrm{init}}}$ for which (5.27) is satisfied. The tree construction can then be continued by adding the boxes Z_k that satisfy $\#\hat{Z}_k > 0$ as children of $Z^{(0)}$ to $\mathcal{T}_\Sigma$ and calling the function SubdivideCluster in line 3 of Algorithm 5.1 for them.

The described splitting in time yields the following property: If a space-time element $\sigma_{j_t,j_{\boldsymbol{x}}} = \gamma_{j_{\boldsymbol{x}}} \times (t_{j_t-1}, t_{j_t})$ is assigned to a box $Z = X \times I$, the temporal part (t_{j_t-1}, t_{j_t})



a) Mesh in a box $Z^{(0)}$. b) Boxes at level 1. c) Boxes at level 2. d) Resulting tree $\mathcal{T}_\Sigma$.

Figure 5.1: Construction of a space-time cluster tree in 2D. Instead of 4D space-time elements σ we consider 2D rectangular elements consisting of temporal intervals $\{J_{k_t}\}_{k_t=1}^{E_t}$ and spatial 1D elements $\{\gamma_{k_{\boldsymbol{x}}}\}_{k_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}}$. First, a box $Z^{(0)}$ at level zero is constructed that contains all these elements; see a). This box is refined recursively as described in Algorithm 5.1. A purely temporal refinement leads to the two boxes at level one in b). Refining these boxes in space and time results in eight boxes at level two; see c). By connecting all child boxes with their respective parent box we obtain the space-time tree in d).

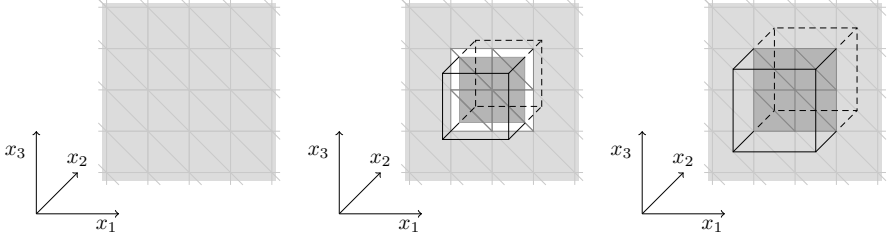
is fully contained in the temporal interval I of Z. We want to ensure in addition that the inclusion $\gamma_{j_{\boldsymbol{x}}} \subset X$ holds for all space-time elements $\sigma_{j_t,j_{\boldsymbol{x}}}$ assigned to Z. If this is not the case, we pad the boxes in $\mathcal{T}_\Sigma$ appropriately in a post-processing step, i.e. we extend the spatial size of a box Z such that $\sigma_{j_t,j_{\boldsymbol{x}}} \subset \overline{Z_{\text{pad}}}$ for the extended box Z_{pad} and all $\sigma_{j_t,j_{\boldsymbol{x}}}$ with $(j_t, j_{\boldsymbol{x}}) \in \hat{Z}$. For a box $Z = X \times I$ with $X = (\boldsymbol{a}, \boldsymbol{a} + 2\widetilde{h}_{\boldsymbol{x}}\mathbf{1}]$ we set

$$\widetilde{h}_{\text{pad},\boldsymbol{x}}(Z) = \max\left\{\min_{\boldsymbol{x}\in X} \|\boldsymbol{x} - \boldsymbol{y}\|_\infty : \boldsymbol{y} \in \gamma_{j_{\boldsymbol{x}}}, (j_t, j_{\boldsymbol{x}}) \in \hat{Z}\right\},$$

where $\|\cdot\|_\infty$ denotes the maximum norm in $\mathbb{R}^3$. The box with spatial padding $\widetilde{h}_{\text{pad},\boldsymbol{x}}(Z)$ is given by $\big(\boldsymbol{a} - \widetilde{h}_{\text{pad},\boldsymbol{x}}(Z)\mathbf{1},\ \boldsymbol{a} + (2\widetilde{h}_{\boldsymbol{x}} + \widetilde{h}_{\text{pad},\boldsymbol{x}}(Z))\mathbf{1}\big] \times I$, and all $\sigma_{j_t,j_{\boldsymbol{x}}}$ with $(j_t, j_{\boldsymbol{x}}) \in \hat{Z}$ are fully contained in the closure of this extended box. The corresponding padding of a box $X \in \mathbb{R}^3$ is illustrated in Figure 5.2. Note that we might lose the spatial uniformity of the boxes in $\mathcal{T}_\Sigma$ if we pad all boxes Z by an individual amount $\widetilde{h}_{\text{pad},\boldsymbol{x}}(Z)$. Furthermore, the relation $Z_{\text{c}} \subset Z$ for $Z_{\text{c}} \in \text{child}(Z)$ might not be satisfied anymore for the padded boxes. Since we want to exploit these properties in the FMM, we pad all boxes at the same level ℓ in $\mathcal{T}_\Sigma$ by the same amount $\widetilde{h}^\ell_{\text{pad},\boldsymbol{x}}$ which we define by

$$\begin{aligned} \widetilde{h}^\ell_{\text{pad},\boldsymbol{x}} &:= \max\{\widetilde{h}^{k,\text{loc}}_{\text{pad},\boldsymbol{x}} : k \in \{\ell, \dots, \text{depth}(\mathcal{T}_\Sigma)\}\}, \\ \widetilde{h}^{\ell,\text{loc}}_{\text{pad},\boldsymbol{x}} &:= \max\{\widetilde{h}_{\text{pad},\boldsymbol{x}}(Z) : Z \in \mathcal{T}_\Sigma, \ell(Z) = \ell\}. \end{aligned} \tag{5.29}$$

Note that the space-time configuration criterion (5.27) with the constant c_{st} provided in Algorithm 5.1 might be violated for the lengths $\widetilde{h}_{\boldsymbol{x}}(Z_{\text{pad}})$ and $\widetilde{h}_t(Z_{\text{pad}})$ of the padded version Z_{pad} of a box $Z \in \mathcal{T}_\Sigma$, because $\widetilde{h}_{\boldsymbol{x}}(Z)$ is increased. The criterion might



a) Part of a mesh Γ_h. b) Box X without padding. c) Padded box X_{pad}.

Figure 5.2: Illustration of the padding of a spatial box X. In a) we see a planar part of a spatial surface mesh Γ_h. The spatial box X in b) intersects this mesh. The parts of the triangles which are contained in X are colored in a darker shade of gray while the protruding parts are colored in white. By padding X we obtain the larger box X_{pad} in c) which fully contains all triangles.

even be violated for an non-padded box Z, since we ensure only that (5.27) holds for the quantity $\widetilde{h}_t^{(\ell_t(Z))} = 2^{-\ell_t(Z)}\widetilde{h}_t(Z^{(0)})$ instead of the actual temporal half length $\widetilde{h}_t(Z)$ of Z in line 6 of Algorithm 5.1. However, we assume in the following that $\widetilde{h}_t(Z)$ differs only slightly from $\widetilde{h}_t^{(\ell_t(Z))}$ and that the amount of padding is reasonably small related to the original spatial size of Z. Then (5.27) is satisfied for a constant c_{st} slightly larger than the original one. Note that the assumption on the spatial padding is justified since we stop the subdivision process in Algorithm 5.1 earlier if a spatial subdivision of a box is not feasible.

In the following sections we identify a box Z with its padded version Z_{pad} to simplify the notation. Furthermore, we use the following naming convention: The spatial part of a box Z is denoted by X and the temporal part by I. If an index is used to specify Z, the spatial and temporal components are labeled with the same index. For example, $Z_{\mathrm{src}} = X_{\mathrm{src}} \times I_{\mathrm{src}}$ and $Z_{\mathrm{tar}} = X_{\mathrm{tar}} \times I_{\mathrm{tar}}$.

5.2.2 Matrix partitioning using operation lists

Let $\mathcal{T}_\Sigma$ be a space-time box cluster tree constructed by Algorithm 5.1. The boxes in $\mathcal{T}_\Sigma$ can be used to partition BEM matrices for the heat equation. For this purpose, we introduce suitable operation lists in this section. Similar lists have already been considered, for example, in [51, 68, 76], but the description here has a more algorithmic nature. The operation lists of clusters in $\mathcal{T}_\Sigma$ will be constructed by a recursive procedure based on purely geometrical criteria related to the approximation of the heat kernel (3.4) in Section 5.1. In particular, they can be used to partition any of the matrices V_h, K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ or D_h from Section 3.3. We start by describing the desired partition for the matrix V_h whose entries are given in (3.25). This will reveal how to construct the operation lists of clusters in $\mathcal{T}_\Sigma$. The corresponding partition of the other BEM matrices is described at the end of the section.

For two boxes Z_{tar} and Z_{src} in $\mathcal{T}_\Sigma$ let $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ be the block of V_h formed by the rows corresponding to indices $(j_t, j_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{tar}}$ and the columns corresponding to indices $(k_t, k_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{src}}$. Let $Z^{(0)}$ be the root of $\mathcal{T}_\Sigma$. The block $\mathsf{V}_h|_{\hat{Z}^{(0)} \times \hat{Z}^{(0)}}$ corresponds to the full matrix V_h. We can subdivide it into n_{c}^2 blocks $\mathsf{V}_h|_{\hat{Z}_i^{(1)} \times \hat{Z}_j^{(1)}}$, where $\{Z_j^{(1)}\}_{j=1}^{n_{\mathrm{c}}}$ are the children of $Z^{(0)}$ in $\mathcal{T}_\Sigma$. For the FMM we recursively subdivide the matrix into blocks in this manner. The subdivision is stopped for a block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ if Z_{tar} and Z_{src} allow for the approximation (5.9) of the heat kernel (3.4) due to a proper temporal separation. Such a block is called *admissible*. We will see in Section 5.2.3 how to approximate admissible blocks of V_h efficiently. All other blocks are called inadmissible and are further subdivided. This is possible as long as Z_{tar} and Z_{src} have children in the tree $\mathcal{T}_\Sigma$.

The operation lists mentioned at the beginning of the section are used to identify and keep track of the blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ in the final partition of V_h. We introduce interaction lists and nearfield lists for clusters Z_{tar} in $\mathcal{T}_\Sigma$. The *interaction list* $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ is filled with clusters Z_{src} such that the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ is admissible, but the corresponding block of the parents of Z_{tar} and Z_{src} is inadmissible. The subscript M2L of this list is related to the associated FMM operation, which we introduce in Section 5.2.3. The nearfield list or simply the *nearfield* $\mathcal{N}(Z_{\mathrm{tar}})$ of Z_{tar} contains clusters Z_{src} corresponding to blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ which are inadmissible and cannot be further subdivided to obtain admissible subblocks. For a given cluster Z_{tar} one or even both of these lists might be empty.

Before we can describe the construction of the nearfield lists and interaction lists of clusters in $\mathcal{T}_\Sigma$, we have to specify how to identify admissible blocks. In Theorem 5.9 and Remark 5.10 we have shown that the expansion (5.9) of the heat kernel (3.4) is well-suited for Z_{tar} and Z_{src} if the temporal intervals I_{tar} and I_{src} satisfy the standard admissibility criterion (5.12) for a constant $\eta_2 > 0$ and if the space-time configuration criterion (5.27) is satisfied for a constant $c_{\mathrm{st}} > 0$. Criterion (5.12) can be checked explicitly during the recursive construction of the operation lists, while criterion (5.27) is always satisfied due to the construction of the boxes of the tree $\mathcal{T}_\Sigma$ in Algorithm 5.1.

The partitioning of BEM matrices like V_h and the construction of the related operation lists is further influenced by the causality and the exponential decay of the heat kernel G_α in (3.4). Due to the causality, $G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau)$ vanishes for all $(\boldsymbol{x}, t) \in Z_{\mathrm{tar}}$ and $(\boldsymbol{y}, \tau) \in Z_{\mathrm{src}}$ with $I_{\mathrm{tar}} = (a_1, b_1]$ and $I_{\mathrm{src}} = (a_2, b_2]$ if $a_2 \geq b_1$. We say that I_{src} is *causally relevant* for I_{tar} in the contrary case, i.e. if $a_2 < b_1$. If I_{src} is not causally relevant for I_{tar}, all entries of the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ are zero and we can ignore it. This is related to the lower triangular block structure of V_h; see (3.29). In particular, the nearfield and interaction list of a cluster Z_{tar} will be filled only with clusters Z_{src} whose temporal component I_{src} is causally relevant for I_{tar}.

The exponential decay of the heat kernel allows to neglect interactions between clusters that are sufficiently separated in space. We identify such pairs of clusters as follows. Let Z_{tar} be a cluster in $\mathcal{T}_\Sigma$ with spatial level $\ell_{\boldsymbol{x}} = \ell_{\boldsymbol{x}}(Z)$. Due to the uniform subdivision strategy employed in the construction of $\mathcal{T}_\Sigma$, X_{tar} is one of $8^{\ell_{\boldsymbol{x}}}$ possible 3D boxes in a regular grid $\mathcal{G}^{\ell_{\boldsymbol{x}}}$. We can label the boxes in this grid with multi-indices in $\{0, \dots, 2^{\ell_{\boldsymbol{x}}} - 1\}^3$ in a regular way (ascending componentwise) and use these indices to measure the distance of boxes. For two spatial boxes X and Y in $\mathcal{G}^{\ell_{\boldsymbol{x}}}$ with corresponding multi-indices $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ we define the *grid distance* of X and Y by

$$\operatorname{griddist}(X, Y) := \max_{j=1,\dots,3} \{|\xi_j - \zeta_j|\}. \tag{5.30}$$

For a fixed truncation parameter n_{tr} we define the *interaction area* $\mathcal{I}_{\mathcal{A}}(X)$ of X in $\mathcal{G}^{\ell_{\boldsymbol{x}}}$ by

$$\mathcal{I}_{\mathcal{A}}(X) := \{Y \in \mathcal{G}^{\ell_{\boldsymbol{x}}} : \operatorname{griddist}(X, Y) \leq n_{\mathrm{tr}}\}. \tag{5.31}$$

Blocks of BEM matrices like $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ that are related to Z_{tar} and Z_{src} can be neglected if $X_{\mathrm{src}} \notin \mathcal{I}_{\mathcal{A}}(X_{\mathrm{tar}})$. Therefore, we do not have to include such clusters Z_{src} in the operation lists of Z_{tar}. From [68, Section 5.4] it follows that a single, properly chosen truncation parameter n_{tr} can be used for the definition of the interaction area and the related truncation for various pairs of clusters Z_{tar} and Z_{src}. A suitable requirement is that the ratio $\widetilde{h}_{\boldsymbol{x}}^2 / \max\{|I_{\mathrm{tar}}|, |I_{\mathrm{src}}|\}$ of these clusters is similar, where $\widetilde{h}_{\boldsymbol{x}} = \widetilde{h}_{\boldsymbol{x}}(Z_{\mathrm{tar}}) = \widetilde{h}_{\boldsymbol{x}}(Z_{\mathrm{src}})$. This will be the case in the construction of the operation lists for clusters Z_{src} and Z_{tar} in the part of $\mathcal{T}_\Sigma$ where we alternate between purely temporal and space-time subdivisions. Note that in the remaining, initial part of the tree no spatial truncation is possible since all clusters are subdivided only in time and, therefore, share the same spatial component.

Finally, we are ready to describe the construction of the operation lists of clusters in a tree $\mathcal{T}_\Sigma$. It is based on the recursive routine DetermineOperationLists in Algorithm 5.2 that acts on pairs of clusters, and is first called for $(Z^{(0)}, Z^{(0)})$ where $Z^{(0)}$ denotes the root of $\mathcal{T}_\Sigma$. In the following we call pairs of clusters $(Z_{\mathrm{src}}, Z_{\mathrm{tar}})$ blocks due to their connection to blocks of BEM matrices. In the routine DetermineOperationLists we first check if $X_{\mathrm{src}} \in \mathcal{I}_{\mathcal{A}}(X_{\mathrm{tar}})$ and if I_{src} is causally relevant for I_{tar} (line 3). If this is not the case we can ignore the current block as seen before. Otherwise, we check whether the admissibility criterion (5.12) is satisfied (line 4) and add Z_{src} to the interaction list $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ if this is the case. In the contrary case, we want to subdivide the block. If neither Z_{src} nor Z_{tar} is a leaf in $\mathcal{T}_\Sigma$, we can subdivide the block by calling the routine DetermineOperationLists in line 9 for all pairs of children of Z_{src} and Z_{tar}. If either Z_{src} or Z_{tar} is a leaf, further subdivisions are not possible or would not yield admissible subblocks anymore, so we add Z_{src} to the

Algorithm 5.2 Recursive construction of the operation lists in $\mathcal{T}_\Sigma$.

Require: Let $\mathcal{T}_\Sigma$ be a space-time cluster tree with root $Z^{(0)}$.
Fix a constant $\eta_2 \in \mathbb{R}_{>0}$ for criterion (5.12) and n_{tr} for the definition of (5.31).

1: Call DetermineOperationLists($Z^{(0)}$, $Z^{(0)}$).
2: **function** DetermineOperationLists(Z_{src}, Z_{tar})
3: **if** $X_{\mathrm{src}} \in \mathcal{I}_{\mathcal{A}}(X_{\mathrm{tar}})$ **and** I_{src} is causally relevant for I_{tar}
4: **if** I_{src} and I_{tar} satisfy the admissibility criterion (5.12)
5: Add Z_{src} to $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$.
6: **else**
7: **if** Z_{tar} and Z_{src} are not leaves in $\mathcal{T}_\Sigma$
8: **for** all $(Z_{\mathrm{src,c}}, Z_{\mathrm{tar,c}})$ with $Z_{\mathrm{src,c}} \in \mathrm{child}(Z_{\mathrm{src}})$, $Z_{\mathrm{tar,c}} \in \mathrm{child}(Z_{\mathrm{tar}})$
9: Call DetermineOperationLists($Z_{\mathrm{src,c}}$, $Z_{\mathrm{tar,c}}$).
10: **else**
11: Add Z_{src} to $\mathcal{N}(Z_{\mathrm{tar}})$.

nearfield $\mathcal{N}(Z_{\mathrm{tar}})$ (line 11). Note that by construction the levels $\ell(Z_{\mathrm{tar}})$ and $\ell(Z_{\mathrm{src}})$ coincide whenever $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ or $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$.

REMARK 5.11 (Subdividing inadmissible blocks). *If Z_{src} is a cluster in the nearfield $\mathcal{N}(Z_{\mathrm{tar}})$ of a cluster Z_{tar}, one of the two clusters is a leaf by construction. Let us assume that Z_{src} is a leaf but Z_{tar} is not. If the cluster Z_{tar} contains many space-time elements, the inadmissible block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ has many rows. In particular, the inadmissible blocks of V_h might vary greatly in size. In Chapter 6 we will talk about the parallelization of the FMM presented in the current sections, where such varying block sizes are undesirable since they make a balanced distribution of work more difficult. However, we can split $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ into smaller blocks by further subdividing Z_{tar} and thereby the rows of the block. In this way, we obtain the set of matrix blocks $\{\mathsf{V}_h|_{\hat{Z}_{\mathrm{d}} \times \hat{Z}_{\mathrm{src}}}\}_{\mathrm{d}}$, where $\{Z_{\mathrm{d}}\}_{\mathrm{d}}$ denotes the set of all descendants of Z_{tar} in $\mathcal{T}_\Sigma$ which are leaves. While these blocks are still inadmissible, their sizes are uniformly bounded in general.*

To adapt the nearfield lists according to the described subdivision of the inadmissible block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$, we remove Z_{src} from the nearfield $\mathcal{N}(Z_{\mathrm{tar}})$ and add it to the nearfield $\mathcal{N}(Z_{\mathrm{d}})$ of each leaf descendant Z_{d} of Z_{tar}. If, instead, Z_{tar} is a leaf while $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$ is not, we can similarly remove Z_{src} from $\mathcal{N}(Z_{\mathrm{tar}})$ and add all leaf descendants of Z_{src} to $\mathcal{N}(Z_{\mathrm{tar}})$. If all nearfield lists of clusters in $\mathcal{T}_\Sigma$ are treated in this way, the resulting nearfield lists of non-leaf clusters in $\mathcal{T}_\Sigma$ are empty, and the nearfield lists of leaf clusters contain only leaf clusters. When we talk about parallelization in later sections we will assume that this is the case. For the further description of the FMM it does not make a difference.

The partitioning of the matrices K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h with the operation lists constructed by Algorithm 5.2 is slightly more complicated. Let us first consider the matrix K_h, whose entries are given in (3.26). Each column index $(j_t, j_{\boldsymbol{x}})$ is related to a basis function $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}\varphi^0_{t,j_t} \in S^{1\otimes 0}_{h_{\boldsymbol{x}},h_t}$. Recall that $\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}$ is a piecewise linear nodal basis function on Γ_h that satisfies

$$\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{x}_{k_{\boldsymbol{x}}}) = \delta_{j_{\boldsymbol{x}}k_{\boldsymbol{x}}}$$

for the vertices $\{\boldsymbol{x}_{k_{\boldsymbol{x}}}\}_{k_{\boldsymbol{x}}=1}^{N_{\boldsymbol{x}}}$ of Γ_h. Hence, we have to assign such "nodal" indices $(j_t, j_{\boldsymbol{x}})$ to clusters in $\mathcal{T}_\Sigma$ to partition the columns of K_h with respect to these clusters. For this purpose we define the nodal index set $\check{Z}$ of a cluster Z by

$$\check{Z} := \{(j_t, j_{\boldsymbol{x}}) : \exists (j_t, n_{\boldsymbol{x}}) \in \hat{Z} \text{ such that } \boldsymbol{x}_{j_{\boldsymbol{x}}} \in \overline{\gamma_{n_{\boldsymbol{x}}}}\}. \tag{5.32}$$

In other words: For each space-time element $\sigma_{j_t,n_{\boldsymbol{x}}}$ assigned to Z we consider the vertices $\{\boldsymbol{x}_{j_{\boldsymbol{x}}}\}_{j_{\boldsymbol{x}}}$ of the related spatial triangle $\gamma_{n_{\boldsymbol{x}}}$ and assign the corresponding nodal index pairs $\{(j_t, j_{\boldsymbol{x}})\}_{j_{\boldsymbol{x}}}$ to Z. While each space-time element $\sigma_{j_t,n_{\boldsymbol{x}}}$ is assigned to at most one cluster $Z^{(\ell)}$ at level ℓ of the tree $\mathcal{T}_\Sigma$, a nodal index pair $(j_t, j_{\boldsymbol{x}})$ can be

assigned to several clusters at a single level ℓ. We account for this by defining the block $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ of K_h by

$$\begin{aligned}&\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}[(k_t-1)E_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)N_{\boldsymbol{x}}+j_{\boldsymbol{x}}] \\ &\quad=\int_{t_{k_t-1}}^{t_{k_t}}\int_{\gamma_{k_{\boldsymbol{x}}}}\int_{t_{j_t-1}}^{t_{j_t}}\int_{\mathcal{S}_\Gamma(Z_{\mathrm{src}})}\alpha\frac{\partial}{\partial\boldsymbol{n}_{\boldsymbol{y}}}G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\end{aligned}\tag{5.33}$$

for all $(k_t,k_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{tar}}$ and all $(j_t,j_{\boldsymbol{x}})\in\check{Z}_{\mathrm{src}}$, where $\mathcal{S}_\Gamma(Z_{\mathrm{src}})$ is the subset of the spatial surface Γ given by

$$\mathcal{S}_\Gamma(Z_{\mathrm{src}}):=\bigcup_{(j_t,n_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{src}}}\gamma_{n_{\boldsymbol{x}}}.\tag{5.34}$$

Note that here we use global indices instead of local indices to access blocks of BEM matrices like $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ in (5.33), since it simplifies the description. The definition of blocks of K_h in (5.33) leads to an overlapping partition of the matrix, where an entry $\mathsf{K}_h[(k_t-1)E_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)N_{\boldsymbol{x}}+j_{\boldsymbol{x}}]$ is distributed among blocks $\{\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_n}\}_n$ with $(k_t,k_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{tar}}$ and $(j_t,j_{\boldsymbol{x}})\in\bigcap_n\check{Z}_n$. In the later application, we apply a block like $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ to the local part $\boldsymbol{g}|_{\check{Z}_{\mathrm{src}}}$ of a vector $\boldsymbol{g}\in\mathbb{R}^{E_tN_{\boldsymbol{x}}}$ that contains the full entries of $\boldsymbol{g}$ corresponding to indices in $\check{Z}_{\mathrm{src}}$.

The matrix $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ which discretizes the adjoint time-reversed double layer operator K_T' is obtained from the matrix K_h by a suitable blockwise transposition; see (3.33). Since this transposition is not very intuitive to handle when we work with blocks related to clusters Z_{tar} and Z_{src}, we define the block $\mathsf{K}_h^{\top_{\boldsymbol{x}}}|_{\check{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ directly by

$$\begin{aligned}&\mathsf{K}_h^{\top_{\boldsymbol{x}}}|_{\check{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}[(k_t-1)N_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)E_{\boldsymbol{x}}+j_{\boldsymbol{x}}] \\ &\quad=\int_{t_{k_t-1}}^{t_{k_t}}\int_{\mathcal{S}_\Gamma(Z_{\mathrm{tar}})}\int_{t_{j_t-1}}^{t_{j_t}}\int_{\gamma_{j_{\boldsymbol{x}}}}\alpha\frac{\partial}{\partial\boldsymbol{n}_{\boldsymbol{x}}}G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\end{aligned}\tag{5.35}$$

for all $(k_t,k_{\boldsymbol{x}})\in\check{Z}_{\mathrm{tar}}$ and all $(j_t,j_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{src}}$.

The matrix D_h can be partitioned similarly into blocks $\mathsf{D}_h|_{\check{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$, but we distinguish two cases. If $Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$, we can use the representation in (4.64) and define

$$\begin{aligned}&\mathsf{D}_h|_{\check{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}[(k_t-1)N_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)N_{\boldsymbol{x}}+j_{\boldsymbol{x}}] \\ &\quad=-\int_{t_{k_t-1}}^{t_{k_t}}\int_{\mathcal{S}_\Gamma(Z_{\mathrm{tar}})}\int_{t_{j_t-1}}^{t_{j_t}}\int_{\mathcal{S}_\Gamma(Z_{\mathrm{src}})}\alpha\frac{\partial}{\partial\boldsymbol{n}_{\boldsymbol{x}}}\frac{\partial}{\partial\boldsymbol{n}_{\boldsymbol{y}}}G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau) \\ &\qquad\qquad\times\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\mathrm{d}\tau\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\mathrm{d}t\end{aligned}\tag{5.36}$$

for all $(k_t,k_{\boldsymbol{x}})\in\check{Z}_{\mathrm{tar}}$ and all $(j_t,j_{\boldsymbol{x}})\in\check{Z}_{\mathrm{src}}$. This is possible, since I_{src} satisfies the admissibility criterion (5.12) in this case and, therefore, there holds $j_t<k_t-1$ for all such indices. If instead $Z_{\mathrm{src}}\in\mathcal{N}(Z_{\mathrm{tar}})$, we use the representation (5.36) for the entries

of the block $\mathsf{D}_h|_{\check{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ for pairs of indices $(k_t, k_{\boldsymbol{x}}) \in \check{Z}_{\mathrm{tar}}$ and $(j_t, j_{\boldsymbol{x}}) \in \check{Z}_{\mathrm{src}}$ with $j_t < k_t - 1$. For all other entries we use the representation based on the integration by parts formula in (4.60) and treat the spatial integrals in both bilinear forms in the same way as those in (5.36) to distribute entries corresponding to shared nodal indices among the involved blocks. Note that using these two representations together is unproblematic, as long as we do not mix the representations for a single matrix entry corresponding to two or more blocks. This is guaranteed, since we use (4.64) whenever $j_t < k_t - 1$ holds for the involved temporal indices and (4.60) otherwise.

REMARK 5.12 (Important terms). *The terms collected in the following list play an important role in the rest of the thesis, which is why we repeat them here:*

- *Admissible block: A block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ is called admissible in this work if the corresponding clusters Z_{tar} and Z_{src} allow for the approximation (5.9) of the heat kernel (3.4). Admissible blocks of K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h are defined analogously. A block that is not admissible is called inadmissible.*
- *Interaction list: The interaction list of a cluster Z_{tar} is denoted by $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$. It is defined by the construction in Algorithm 5.2. A cluster Z_{src} is contained in $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ if the block of a BEM matrix corresponding to Z_{tar} and Z_{src} is admissible, but the block corresponding to their parent clusters is not.*
- *Nearfield list (or simply nearfield): The nearfield of a cluster Z_{tar} is denoted by $\mathcal{N}(Z_{\mathrm{tar}})$ and defined by the construction in Algorithm 5.2. A cluster Z_{src} is contained in $\mathcal{N}(Z_{\mathrm{tar}})$ if the block of a BEM matrix corresponding to Z_{tar} and Z_{src} is inadmissible, and either Z_{tar} or Z_{src} cannot be further subdivided.*

5.2.3 Approximation of admissible matrix blocks

In this section we describe how to efficiently approximate admissible blocks of the BEM matrices V_h, K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h that are obtained from the partition constructed in Section 5.2.2. The approximation of blocks is based on the approximation of the heat kernel in (5.9) in Section 5.1.

Let $\mathcal{T}_\Sigma$ be a space-time box cluster tree constructed by Algorithm 5.1 and the operation lists of clusters in $\mathcal{T}_\Sigma$ be constructed by Algorithm 5.2. We focus on the matrix V_h first. For Z_{tar} and Z_{src} in $\mathcal{T}_\Sigma$ we consider the matrix-vector product $\boldsymbol{f}^{\mathrm{loc}} = \mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$, where $\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ is the local part of a vector $\boldsymbol{q} \in \mathbb{R}^{E_t E_{\boldsymbol{x}}}$. There holds

$$f^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} = \sum_{(j_t,j_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{src}}} q_{j_t,j_{\boldsymbol{x}}} \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \quad (5.37)$$

for all indices $(k_t, k_{\boldsymbol{x}}) \in \hat{Z}_{\text{tar}}$, where we used the representation of the entries of V_h in (3.25). If the block $\mathsf{V}_h|_{\hat{Z}_{\text{tar}} \times \hat{Z}_{\text{src}}}$ is admissible, i.e. Z_{src} is contained in the interaction list $\mathcal{I}_{\text{M2L}}(Z_{\text{tar}})$, we can approximate (5.37) in an efficient way by replacing the heat kernel with its approximation in (5.9). The resulting vector $\widetilde{\boldsymbol{f}}^{\text{loc}}$ can be computed in three steps:

S2M (source to moment): We compute the *moments* $\boldsymbol{\mu}(Z_{\text{src}})$ by

$$\mu_{a,\boldsymbol{\kappa}}(Z_{\text{src}}) = \sum_{(j_t, j_{\boldsymbol{x}}) \in \hat{Z}_{\text{src}}} q_{j_t, j_{\boldsymbol{x}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} T_{X_{\text{src}},\boldsymbol{\kappa}}(\boldsymbol{y}) L^{(m_t)}_{I_{\text{src}},a}(\tau) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \,\mathrm{d}\tau \tag{5.38}$$

for all $a \in \{0, ..., m_t\}$ and all multi-indices $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \leq m_{\boldsymbol{x}}$.

M2L (moment to local): We compute the *local contributions* $\boldsymbol{\lambda}(Z_{\text{tar}}, Z_{\text{src}})$ by

$$\lambda_{b,\boldsymbol{\nu}}(Z_{\text{tar}}, Z_{\text{src}}) = \sum_{a=0}^{m_t} \sum_{\boldsymbol{\kappa} \in \mathbb{N}_0^3 : |\boldsymbol{\kappa}+\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}} E^{a,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}} \, \mu_{a,\boldsymbol{\kappa}}(Z_{\text{src}}) \tag{5.39}$$

for all $b \in \{0, ..., m_t\}$ and all multi-indices $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}$, where $E^{a,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}$ are the coefficients in (5.10).

L2T (local to target): For all $(k_t, k_{\boldsymbol{x}})$ in $\hat{Z}_{\text{tar}}$ we evaluate

$$\widetilde{f}^{\text{loc}}_{k_t,k_{\boldsymbol{x}}} = \sum_{b=0}^{m_t} \sum_{|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}} \lambda_{b,\boldsymbol{\nu}}(Z_{\text{tar}}, Z_{\text{src}}) \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} T_{X_{\text{tar}},\boldsymbol{\nu}}(\boldsymbol{x}) L^{(m_t)}_{I_{\text{tar}},b}(t) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \,\mathrm{d}t. \tag{5.40}$$

The integrals of polynomials in the S2M and L2T operations can be computed by numerical quadrature formulae. For the integrals of the Lagrange polynomials $L^{(m_t)}_{I,a}$ over a time interval (t_{j_t-1}, t_{j_t}) we use a Gauß–Legendre quadrature formula with $\lceil (m_t+1)/2 \rceil$ quadrature points, which is exact in this case. For the integrals of the Chebyshev polynomials over a triangle $\gamma_{j_{\boldsymbol{x}}}$ we use quadrature rules defined on triangles. In practice, a 7-point formula proves to be accurate enough.

For the execution of an S2M operation $\mathcal{O}(\hat{Z}_{\text{src}}\binom{m_{\boldsymbol{x}}+3}{3} + n_t(\hat{Z}_{\text{src}})\binom{m_{\boldsymbol{x}}+3}{3}(m_t+1))$ arithmetic operations are required, where $n_t(\hat{Z}_{\text{src}})$ denotes the number of all distinct time-indices in $\hat{Z}_{\text{src}}$ or, in other words, the number of time intervals (t_{j_t-1}, t_{j_t}) in I_{src}. This can be seen by rearranging the operation in (5.38) as

$$\mu_{a,\boldsymbol{\kappa}}(Z_{\text{src}}) = \sum_{j_t} \int_{t_{j_t-1}}^{t_{j_t}} L^{(m_t)}_{I_{\text{src}},a}(\tau) \,\mathrm{d}\tau \sum_{j_{\boldsymbol{x}}} q_{j_t,j_{\boldsymbol{x}}} \int_{\gamma_{j_{\boldsymbol{x}}}} T_{X_{\text{src}},\boldsymbol{\kappa}}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}.$$

Similarly, an L2T operation requires $\mathcal{O}(\hat{Z}_{\text{tar}}\binom{m_{\boldsymbol{x}}+3}{3} + n_t(\hat{Z}_{\text{tar}})\binom{m_{\boldsymbol{x}}+3}{3}(m_t+1))$ arithmetic operations. A naive realization of the M2L operation in (5.39) would require $\mathcal{O}(\binom{m_{\boldsymbol{x}}+3}{3}^2(m_t+1)^2)$ arithmetic operations. In [69, Section 4.3] an efficient alternative is proposed which requires only $\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)^2)$ operations. We use this alternative in our implementation.

Admissible blocks $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}|_{\check{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ and $\mathsf{D}_h|_{\check{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ of the BEM matrices K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h are also approximated by replacing the corresponding kernel functions with suitable approximations. The related matrix-vector multiplications can then again be executed by appropriate S2M, M2L, and L2T operations. We will see that the S2M and L2T operations depend on the considered matrix, while the M2L operations are the same for all matrices. In the following, we describe these operations in more detail. As before, we will use $\boldsymbol{\mu}(Z_{\mathrm{src}})$ and $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ to denote the moments and local contributions corresponding to clusters Z_{src} and Z_{tar}, and $\widetilde{\boldsymbol{f}}^{\mathrm{loc}}$ to denote the result of the approximate matrix-vector multiplication for a given block of a BEM matrix. While all these objects depend on the considered BEM matrix and are not necessarily the same as those in (5.38), (5.39) and (5.40), we use the same names to keep the notation simple. In the following paragraphs and sections we will always focus on a single BEM matrix, so there is no risk of confusion.

The kernel function $((\boldsymbol{x},t),(\boldsymbol{y},\tau)) \mapsto \partial_{\boldsymbol{n_y}} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)$ corresponding to the matrix K_h can be approximated by the normal derivative $\partial_{\boldsymbol{n_y}}$ of (5.9), i.e.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{n_y}} &\mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_{\mathrm{tar}}\times X_{\mathrm{src}}} \mathfrak{I}^{(m_t)}_{I_{\mathrm{tar}}\times I_{\mathrm{src}}}[G_\alpha](\boldsymbol{x},t,\boldsymbol{y},\tau) \\ &= \sum_{a,b=0}^{m_t} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu}\in\mathbb{N}_0^3:\\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|\le m_{\boldsymbol{x}}}} E^{a,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}\, T_{X_{\mathrm{tar}},\boldsymbol{\nu}}(\boldsymbol{x})\, \frac{\partial}{\partial \boldsymbol{n_y}} T_{X_{\mathrm{src}},\boldsymbol{\kappa}}(\boldsymbol{y})\, L^{(m_t)}_{I_{\mathrm{tar}},b}(t)\, L^{(m_t)}_{I_{\mathrm{src}},a}(\tau)\end{aligned}$$

for all $(\boldsymbol{x},t) \in Z_{\mathrm{tar}} \cap \Sigma$ and $(\boldsymbol{y},\tau) \in Z_{\mathrm{src}} \cap \Sigma$ with $Z_{\mathrm{src}} \in \mathcal{I}_{M2L}(Z_{\mathrm{tar}})$. The S2M operation for the related, approximate matrix-vector product $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}\boldsymbol{g}|_{\check{Z}_{\mathrm{src}}}$ with $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ defined in (5.33) is to compute the moments $\mu_{a,\boldsymbol{\kappa}}(Z_{\mathrm{src}})$ for all indices $a \in \{0,\dots,m_t\}$ and all multi-indices $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \le m_{\boldsymbol{x}}$ by

$$\mu_{a,\boldsymbol{\kappa}}(Z_{\mathrm{src}}) = \sum_{(j_t,j_{\boldsymbol{x}})\in\check{Z}_{\mathrm{src}}} g_{j_t,j_{\boldsymbol{x}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{S_\Gamma(Z_{\mathrm{src}})} \alpha \frac{\partial}{\partial \boldsymbol{n_y}} T_{X_{\mathrm{src}},\boldsymbol{\kappa}}(\boldsymbol{y}) \varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y}) L^{(m_t)}_{I_{\mathrm{src}},a}(\tau)\,\mathrm{d}\boldsymbol{s_y}\,\mathrm{d}\tau, \quad (5.41)$$

where $S_\Gamma(Z_{\mathrm{src}})$ is defined in (5.34). The subsequent M2L and L2T operations are the same as in the case of the single layer operator matrix V_h, i.e. (5.39) and (5.40).

For an admissible block $\mathsf{K}_h^{\top_{\boldsymbol{x}}}|_{\check{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ the S2M and M2L operations for the approximation of $\mathsf{K}_h^{\top_{\boldsymbol{x}}}|_{\check{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ are the same as in (5.38) and (5.39), while the L2T operation for the evaluation of the local contributions $\boldsymbol{\lambda}(Z_{\mathrm{src}}, Z_{\mathrm{tar}})$ is given by

$$\widetilde{f}^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} = \sum_{b=0}^{m_t} \sum_{|\boldsymbol{\nu}|\le m_{\boldsymbol{x}}} \lambda_{b,\boldsymbol{\nu}}(Z_{\mathrm{tar}}, Z_{\mathrm{src}}) \int_{t_{k_t-1}}^{t_{k_t}} \int_{S_\Gamma(Z_{\mathrm{tar}})} \alpha \frac{\partial}{\partial \boldsymbol{n_x}} T_{X_{\mathrm{tar}},\boldsymbol{\nu}}(\boldsymbol{x}) \varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x}) L^{(m_t)}_{I_{\mathrm{tar}},b}(t)\,\mathrm{d}\boldsymbol{s_x}\,\mathrm{d}t \quad (5.42)$$

for all $(k_t,k_{\boldsymbol{x}}) \in \check{Z}_{\mathrm{tar}}$.

Finally, we consider an admissible block $\mathsf{D}_h|_{\check{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}$ of D_h as defined in (5.36). By applying the normal derivatives $-\partial_{\boldsymbol{n}_{\boldsymbol{x}}}$ and $\partial_{\boldsymbol{n}_{\boldsymbol{y}}}$ to the approximation of the heat kernel in (5.9), we get an approximation of the kernel function in (5.36). The corresponding approximation of the matrix-vector product $\mathsf{D}_h|_{\check{Z}_{\mathrm{tar}}\times\check{Z}_{\mathrm{src}}}\boldsymbol{g}|_{\check{Z}_{\mathrm{src}}}$ consists of the S2M operation in (5.41), the M2L operation in (5.39) and the L2T operation in (5.42) up to a minus sign, i.e.

$$\widetilde{f}^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} = -\sum_{b=0}^{m_t}\sum_{|\boldsymbol{\nu}|\leq m_{\boldsymbol{x}}}\lambda_{b,\boldsymbol{\nu}}(Z_{\mathrm{tar}},Z_{\mathrm{src}})\int_{t_{k_t-1}}^{t_{k_t}}\int_{S_\Gamma(Z_{\mathrm{tar}})}\alpha\frac{\partial}{\partial\boldsymbol{n}_{\boldsymbol{x}}}T_{X_{\mathrm{tar}},\boldsymbol{\nu}}(\boldsymbol{x})\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})L_{I_{\mathrm{tar}},b}(t)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \tag{5.43}$$

for all $(k_t,k_{\boldsymbol{x}})\in\check{Z}_{\mathrm{tar}}$.

Remark 5.13. *An alternative strategy is to use the integration by parts formula representation* (4.60) *for the entries of all blocks of* D_h. *In this case, one has to handle the two bilinear forms* (4.61) *and* (4.62) *separately when approximating an admissible block. Since the functions* $\boldsymbol{y}\mapsto\mathbf{curl}_\Gamma(\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}})(\boldsymbol{y})$ *and* $\boldsymbol{x}\mapsto\mathbf{curl}_\Gamma(\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}})(\boldsymbol{x})$ *in* (4.61) *and the functions* $\boldsymbol{y}\mapsto\varphi^1_{\boldsymbol{x},j_{\boldsymbol{x}}}(\boldsymbol{y})\boldsymbol{n}(\boldsymbol{y})$ *and* $\boldsymbol{x}\mapsto\varphi^1_{\boldsymbol{x},k_{\boldsymbol{x}}}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})$ *in* (4.62) *are vector-valued, one has to execute three groups of S2M, M2L and L2T operations for the first bilinear form — one for each component of the vector-valued functions — and additional three groups for the second bilinear form. Due to the computational overhead of this approach, we use the integration by parts formula representation* (4.60) *only for values of* D_h *in inadmissible blocks as discussed at the end of Section 5.2.2.*

5.2.4 Nested FMM operations and the space-time FMM

In this section we present a space-time FMM for the computation of the matrix-vector product $\mathsf{V}_h\boldsymbol{q}$ for a given vector $\boldsymbol{q}$. We focus on the matrix V_h, but the BEM matrices K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h can be treated similarly as we will see in Remark 5.14 at the end of this section.

The FMM uses the partition of V_h into admissible and inadmissible blocks via the operation lists constructed in Algorithm 5.2 for a blockwise application. Inadmissible blocks are applied directly to the corresponding part of the vector $\boldsymbol{q}$, while admissible blocks can be approximated and applied as described in Section 5.2.3. In the following, we describe how the application of admissible blocks is handled even more efficiently in the FMM, before presenting the full method in Algorithm 5.3.

Recall that an admissible block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of V_h is applied to a local part of a vector $\boldsymbol{q}$ by executing the S2M operation (5.38), followed by the M2L operation (5.39) and the L2T operation (5.40). An important observation is that the computation of the moments $\boldsymbol{\mu}(Z_{\mathrm{src}})$ in the S2M operation (5.38) is independent of the target cluster Z_{tar}. Due to this property, we have to compute the moments $\boldsymbol{\mu}(Z_{\mathrm{src}})$ only once and can

reuse them for the approximate evaluation of all admissible blocks of V_h involving Z_{src} and other target clusters. Likewise, the L2T operation used to evaluate the local contributions $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ in (5.40) is independent of the source cluster Z_{src}. We can exploit this by adding up the local contributions $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ of the cluster Z_{tar} and all source clusters $Z_{\mathrm{src}} \in \mathcal{I}_{M2L}(Z_{\mathrm{src}})$ to get a vector of local contributions $\widetilde{\boldsymbol{\lambda}}(Z_{\mathrm{tar}})$ that can be evaluated by a single L2T operation. Note that the same holds for all the other S2M and L2T operations in Section 5.2.3 corresponding to other BEM matrices.

The computation of moments and evaluation of local contributions for large clusters in $\mathcal{T}_\Sigma$ can be further improved. In fact, it is possible to compute the moments $\boldsymbol{\mu}(Z_{\mathrm{src}})$ of a non-leaf cluster Z_{src} in $\mathcal{T}_\Sigma$ from the moments of its children. Furthermore, we can transform the local contributions $\widetilde{\boldsymbol{\lambda}}(Z_{\mathrm{tar}})$ of a non-leaf cluster Z_{tar} into local contributions of its children, add them to the local contributions of these children, and evaluate them in a single effort. This nested computation of moments and local contributions is a standard approach in FMM algorithms, and was used already in [51, 67, 68]. The related operations are based on suitable changes of bases of the temporal Lagrange polynomials and spatial Chebyshev polynomials, which we describe next.

Let I and I_{c} be two intervals with $I_{\mathrm{c}} \subset I$, and $\{L_{I,b}^{(m_t)}\}_{b=0}^{m_t}$ and $\{L_{I_{\mathrm{c}},a}^{(m_t)}\}_{a=0}^{m_t}$ be the Lagrange polynomials of degree m_t on I and I_{c}, respectively, as defined in (5.2). The restriction $L_{I,b}^{(m_t)}|_{I_{\mathrm{c}}}$ of $L_{I,b}^{(m_t)}$ to I_{c} is a polynomial of degree m_t on I_{c} for all $b \in \{0, \dots, m_t\}$. Hence, we can express it in terms of the basis $\{L_{I_{\mathrm{c}},a}^{(m_t)}\}_{a=0}^{m_t}$ by

$$L_{I,b}^{(m_t)}|_{I_{\mathrm{c}}} = \sum_{a=0}^{m_t} q_{a,b}^{(t)}(I_{\mathrm{c}}, I) L_{I_{\mathrm{c}},a}^{(m_t)}, \quad q_{a,b}^{(t)}(I_{\mathrm{c}}, I) := L_{I,b}^{(m_t)}(\xi_{I_{\mathrm{c}},a}^{(m_t)}) \text{ for } a \in \{0, \dots, m_t\}, \tag{5.44}$$

where $\{\xi_{I_{\mathrm{c}},a}^{(m_t)}\}_{a=0}^{m_t}$ are the Chebyshev nodes of order $m_t + 1$ on I_{c}. This representation follows immediately from the identity $L_{I,b}^{(m_t)}|_{I_{\mathrm{c}}} = \mathfrak{I}_{I_{\mathrm{c}}}^{(m_t)}\big[L_{I,b}^{(m_t)}|_{I_{\mathrm{c}}}\big]$, which holds since the interpolation operator $\mathfrak{I}_{I_{\mathrm{c}}}^{(m_t)}$ is a projection on $C(I_{\mathrm{c}})$ whose image is the space of polynomials $\mathcal{P}_{m_t}(I_{\mathrm{c}})$.

We can also consider Chebyshev polynomials $T_{I,b}$ and $T_{I_{\mathrm{c}},a}$ for $a, b \in \{0, \dots, m_{\boldsymbol{x}}\}$ on the intervals I and I_{c}. Since $T_{I,b}|_{I_{\mathrm{c}}}$ is a polynomial of degree b on I_{c}, it can be represented as a linear combination of the polynomials $\{T_{I_{\mathrm{c}},a}\}_{a=0}^{b}$ by

$$T_{I,b}|_{I_{\mathrm{c}}} = \sum_{a=0}^{b} q_{a,b}^{(x)}(I_{\mathrm{c}}, I) T_{I_{\mathrm{c}},a}. \tag{5.45}$$

To compute the coefficients $q_{a,b}^{(x)}(I_{\mathrm{c}}, I)$ we compare this representation with the Chebyshev series expansion in (5.5), or rather its equivalent on I_{c}, and conclude

$$q_{a,b}^{(x)}(I_{\mathrm{c}}, I) = \frac{\lambda_a}{\pi} \int_{-1}^{1} T_{I,b}(\varphi_{I_{\mathrm{c}}}(x)) T_a(x) w(x) \, \mathrm{d}x,$$

where λ_a and w are the same quantities as in (5.5), and φ_{I_c} is the affine map from $[-1,1]$ to I_c. By evaluating these integrals for all $a,b \in \{0,\dots,m_{\boldsymbol{x}}\}$ with a Gauß–Chebyshev quadrature rule with $m_{\boldsymbol{x}}+1$ points we get

$$q_{a,b}^{(x)}(I_c,I) = \frac{\lambda_a}{m_{\boldsymbol{x}}+1} \sum_{n=0}^{m_{\boldsymbol{x}}} T_{I,b}(\xi_{I_c,n}^{(m_{\boldsymbol{x}})}) T_a(\xi_n^{(m_{\boldsymbol{x}})}). \tag{5.46}$$

Since the quadrature rule is exact for polynomials in $\mathcal{P}_{2m_{\boldsymbol{x}}+1}(-1,1)$ and a and b are bounded by $m_{\boldsymbol{x}}$ we do not introduce an approximation error here. For tensor products of Chebyshev polynomials we get a similar representation. Let $X = (\boldsymbol{a},\boldsymbol{b}] \subset \mathbb{R}^3$ and $X_c = (\boldsymbol{c},\boldsymbol{d}] \subset \mathbb{R}^3$ be two boxes with $X_c \subset X$. Then there holds

$$T_{X,\boldsymbol{\nu}}|_{X_c} = \sum_{\boldsymbol{\kappa}\in\mathbb{N}_0^3:\boldsymbol{\kappa}\le\boldsymbol{\nu}} q_{\boldsymbol{\kappa},\boldsymbol{\nu}}^{(\boldsymbol{x})}(X_c,X) T_{X_c,\boldsymbol{\kappa}}, \quad q_{\boldsymbol{\kappa},\boldsymbol{\nu}}^{(\boldsymbol{x})}(X_c,X) := \prod_{k=1}^{3} q_{\kappa_k,\nu_k}^{(x)}((c_j,d_j],(a_j,b_j]) \tag{5.47}$$

for all $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \le m_{\boldsymbol{x}}$, where the inequality $\boldsymbol{\kappa} \le \boldsymbol{\nu}$ is understood componentwise and we use the coefficients from (5.46).

The identities (5.44) and (5.47) allow us to compute moments and local contributions of clusters in $\mathcal{T}_\Sigma$ in a nested way. The corresponding operations are known as moment to moment (M2M) and local to local (L2L) operations. As in [68, Sections 5.5.2 and 5.5.3] we distinguish two cases for each of them.

Temporal M2M: Let $Z = X \times I$ in $\mathcal{T}_\Sigma$ be a cluster whose children were constructed by a purely temporal subdivision in Algorithm 5.1. For the moments $\boldsymbol{\mu}(Z)$ of Z as defined in (5.38) there holds

$$\begin{aligned}
\mu_{a,\boldsymbol{\nu}}(Z) &= \sum_{\substack{Z_c\in\operatorname{child}(Z)\\ Z_c=X\times I_c}} \sum_{(j_t,j_{\boldsymbol{x}})\in\hat{Z}_c} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} T_{X,\boldsymbol{\nu}}(\boldsymbol{y}) L_{I,a}^{(m_t)}(\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau \\
&= \sum_{\substack{Z_c\in\operatorname{child}(Z)\\ Z_c=X\times I_c}} \sum_{a_c=0}^{m_t} q_{a_c,a}^{(t)}(I_c,I) \sum_{(j_t,j_{\boldsymbol{x}})\in\hat{Z}_c} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} T_{X,\boldsymbol{\nu}}(\boldsymbol{y}) L_{I_c,a_c}^{(m_t)}(\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau \\
&= \sum_{\substack{Z_c\in\operatorname{child}(Z)\\ Z_c=X\times I_c}} \sum_{a_c=0}^{m_t} q_{a_c,a}^{(t)}(I_c,I) \mu_{a_c,\boldsymbol{\nu}}(Z_c)
\end{aligned}$$

for all $a \in \{0,\dots,m_t\}$ and $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \le m_{\boldsymbol{x}}$. The equality in the first line is clear since the index set $\hat{Z}$ of Z is equal to the union of the index sets of all the children of Z. The equality in the second line follows by using the identity (5.44) for $L_{I,a}$ on the intervals $(t_{j_t-1},t_{j_t}) \subset I_c$ and changing the order of summation and

integration. By identifying the moments $\boldsymbol{\mu}(Z_\mathrm{c})$ of a child cluster Z_c we obtain the last line. We conclude that $\boldsymbol{\mu}(Z)$ can be computed by

$$\boldsymbol{\mu}(Z) = \sum_{\substack{Z_\mathrm{c}\in\mathrm{child}(Z)\\ Z_\mathrm{c}=X\times I_\mathrm{c}}} \boldsymbol{\mu}(Z, Z_\mathrm{c}),$$

where the moments $\boldsymbol{\mu}(Z, Z_\mathrm{c})$ are obtained by transforming the moments $\boldsymbol{\mu}(Z_\mathrm{c})$ of a child cluster $Z_\mathrm{c} = X \times I_\mathrm{c}$ into moments of Z with the temporal M2M operation

$$\mu_{a,\boldsymbol{\nu}}(Z, Z_\mathrm{c}) = \sum_{a_\mathrm{c}=0}^{m_t} q^{(t)}_{a_\mathrm{c},a}(I_\mathrm{c}, I)\, \mu_{a_\mathrm{c},\boldsymbol{\nu}}(Z_\mathrm{c}) \tag{5.48}$$

for all $a \in \{0, \dots, m_t\}$ and $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}$. Recall that $q^{(t)}_{a_\mathrm{c},a}(I_\mathrm{c}, I)$ are the coefficients from (5.44).

Space-time M2M: Let $Z = X \times I$ be a cluster whose children were constructed by a space-time subdivision in Algorithm 5.1. Its moments $\boldsymbol{\mu}(Z)$ can be computed by

$$\boldsymbol{\mu}(Z) = \sum_{\substack{Z_\mathrm{c}\in\mathrm{child}(Z)\\ Z_\mathrm{c}=X_\mathrm{c}\times I_\mathrm{c}}} \boldsymbol{\mu}(Z, Z_\mathrm{c}),$$

where the moments $\boldsymbol{\mu}(Z, Z_\mathrm{c})$ of Z related to a child $Z_\mathrm{c} = X_\mathrm{c} \times I_\mathrm{c}$ are computed by the space-time M2M operation

$$\mu_{a,\boldsymbol{\nu}}(Z, Z_\mathrm{c}) = \sum_{a_\mathrm{c}=0}^{m_t} \sum_{\boldsymbol{\kappa}:\boldsymbol{\kappa}\leq\boldsymbol{\nu}} q^{(t)}_{a_\mathrm{c},a}(I_\mathrm{c}, I)\, q^{(\boldsymbol{x})}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(X_\mathrm{c}, X)\, \mu_{a_\mathrm{c},\boldsymbol{\kappa}}(Z_\mathrm{c}) \tag{5.49}$$

for all $a \in \{0, \dots, m_t\}$ and $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}$. The coefficients $q^{(t)}_{a_\mathrm{c},a}(I_\mathrm{c}, I)$ and $q^{(\boldsymbol{x})}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(X_\mathrm{c}, X)$ are given in (5.44) and (5.47), respectively. This operation can be derived analogously as the one in (5.48) by using the representation of polynomials $T_{X,\boldsymbol{\nu}}|_{X_\mathrm{c}}$ in (5.47) in addition.

Temporal L2L: Let $Z = X \times I$ be in $\mathcal{T}_\Sigma$ and $Z_\mathrm{c} = X \times I_\mathrm{c}$ with $I_\mathrm{c} \subset I$ be a purely temporally subdivided child of Z. We consider the evaluation of the local contributions $\boldsymbol{\lambda}(Z)$ as in (5.40), but only for indices $(k_t, k_{\boldsymbol{x}})$ in $\hat{Z}_\mathrm{c}$ instead of $\hat{Z}$. This can be reformulated by

$$\begin{aligned}
\widetilde{f}^{\,\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} &= \sum_{b=0}^{m_t} \sum_{|\boldsymbol{\nu}|\leq m_{\boldsymbol{x}}} \lambda_{b,\boldsymbol{\nu}}(Z) \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} T_{X,\boldsymbol{\nu}}(\boldsymbol{x}) L^{(m_t)}_{I,b}(t)\, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\, \mathrm{d}t \\
&= \sum_{b=0}^{m_t} \sum_{|\boldsymbol{\nu}|\leq m_{\boldsymbol{x}}} \lambda_{b,\boldsymbol{\nu}}(Z) \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} T_{X,\boldsymbol{\nu}}(\boldsymbol{x}) \sum_{b_\mathrm{c}=0}^{m_t} q^{(t)}_{b_\mathrm{c},b}(I_\mathrm{c}, I) L^{(m_t)}_{I_\mathrm{c},b_\mathrm{c}}(t)\, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\, \mathrm{d}t \\
&= \sum_{b_\mathrm{c}=0}^{m_t} \sum_{|\boldsymbol{\nu}|\leq m_{\boldsymbol{x}}} \left(\sum_{b=0}^{m_t} q^{(t)}_{b_\mathrm{c},b}(I_\mathrm{c}, I) \lambda_{b,\boldsymbol{\nu}}(Z) \right) \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} T_{X,\boldsymbol{\nu}}(\boldsymbol{x}) L^{(m_t)}_{I_\mathrm{c},b_\mathrm{c}}(t)\, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\, \mathrm{d}t,
\end{aligned}$$

where we used (5.44) to obtain the identity in the second line. The expression in the last line corresponds to an L2T operation for the cluster Z_c and the local contributions $\boldsymbol{\lambda}(Z_c, Z)$ obtained by the temporal L2L operation

$$\lambda_{b_c,\boldsymbol{\nu}}(Z_c, Z) := \sum_{b=0}^{m_t} q^{(t)}_{b_c,b}(I_c, I)\lambda_{b,\boldsymbol{\nu}}(Z) \tag{5.50}$$

for all $b_c \in \{0, \dots, m_t\}$ and $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}$. We can add $\boldsymbol{\lambda}(Z_c, Z)$ and the local contributions $\widetilde{\boldsymbol{\lambda}}(Z_c)$ of Z_c that are obtained by M2L operations to get

$$\boldsymbol{\lambda}(Z_c) = \widetilde{\boldsymbol{\lambda}}(Z_c) + \boldsymbol{\lambda}(Z_c, Z).$$

In particular, we can evaluate all these local contributions of Z_c at once.

Space-time L2L: Let $Z = X \times I$ be in $\mathcal{T}_\Sigma$ and $Z_c = X_c \times I_c$ with $X_c \subset X$ and $I_c \subset I$ be a child of Z obtained by a space-time subdivision. The local contributions $\boldsymbol{\lambda}(Z)$ can be transformed into local contributions $\boldsymbol{\lambda}(Z_c, Z)$ of Z_c by the space-time L2L operation

$$\lambda_{b_c,\boldsymbol{\kappa}}(Z_c, Z) := \sum_{b=0}^{m_t} q^{(t)}_{b_c,b}(I_c, I) \sum_{\boldsymbol{\nu}:\boldsymbol{\nu}\geq\boldsymbol{\kappa}} q^{(\boldsymbol{x})}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(X_c, X)\lambda_{b,\boldsymbol{\nu}}(Z) \tag{5.51}$$

for all $b_c \in \{0, \dots, m_t\}$ and $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \leq m_{\boldsymbol{x}}$. This operation can be derived in the same way as the one in (5.50) by using the representation (5.47) in addition. As in the case of the temporal L2L operation, we can add up $\boldsymbol{\lambda}(Z_c, Z)$ and the local contributions $\widetilde{\boldsymbol{\lambda}}(Z_c)$ and evaluate them together.

A nested computation of moments and local contributions is not only possible for the matrix V_h but also for the other BEM matrices. In fact, the corresponding M2M and L2L operations are even the same regardless of the considered matrix. Let us compare the S2M operations (5.38) and (5.41), for example. The main difference — apart from the different basis function — is that the normal derivative $\partial_{\boldsymbol{n}} T_{X_{\mathrm{src}},\boldsymbol{\kappa}}$ of the Chebyshev polynomial $T_{X_{\mathrm{src}},\boldsymbol{\kappa}}$ appears in the integral in (5.41) instead of $T_{X_{\mathrm{src}},\boldsymbol{\kappa}}$. From (5.47) we obtain that

$$\frac{\partial}{\partial \boldsymbol{n}} T_{X,\boldsymbol{\nu}}|_{X_c} = \sum_{\boldsymbol{\kappa}:\boldsymbol{\kappa}\leq\boldsymbol{\nu}} q^{(\boldsymbol{x})}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(X_c, X) \frac{\partial}{\partial \boldsymbol{n}} T_{X_c,\boldsymbol{\kappa}}.$$

By using this relation and repeating the above steps to derive the M2M operations we end up with the same temporal and space-time M2M operations as in (5.48) and (5.49). The same holds for the different types of L2T operations.

In Algorithm 5.3 we finally present the space-time FMM for the single layer operator matrix V_h. The computation is split into four phases. The first phase is the forward transformation (lines 2–11) where all the moments of clusters in $\mathcal{T}_\Sigma$ are determined. The moments of leaf clusters are computed by S2M operations, while suitable M2M

operations are used for the computation of the moments of non-leaf clusters. After this phase all moments are available, so all M2L operations can be executed in the second phase, the multiplication phase (lines 12–15). The third phase is the backward transformation (lines 16–23) where the resulting local contributions are passed from the clusters in $\mathcal{T}_\Sigma$ to their children by suitable L2L operations. The local contributions of leaf clusters are finally evaluated by an L2T operation. In the last phase, the nearfield evaluation, the inadmissible blocks are applied directly (lines 24–26).

In [67, Section 5.4] and [51, Section 4.5.3] it is stated that the runtime complexity of the pFMM for the application of V_h is $\mathcal{O}(m_{\boldsymbol{x}}^4 m_t^2 E_t E_{\boldsymbol{x}})$. Recall that the pFMM is closely related to the space-time FMM which we consider in this work. In fact, the FMM operations executed in both methods are the same, but in the pFMM they are executed in a forward-sweeping manner. Thus, our space-time FMM has the same runtime complexity $\mathcal{O}(m_{\boldsymbol{x}}^4 m_t^2 E_t E_{\boldsymbol{x}})$, while a direct application of the matrix V_h has the complexity $\mathcal{O}((E_t E_{\boldsymbol{x}})^2)$. Note that the complexity estimate for the FMM is only valid if the considered space-time tensor product mesh is globally quasi-uniform in space and time and its temporal and spatial mesh sizes h_t and $h_{\boldsymbol{x}}$ satisfy the relation $h_t \sim h_{\boldsymbol{x}}^2$. In [52] an additional nearfield compression scheme was introduced for meshes where $h_{\boldsymbol{x}}$ is too fine compared to h_t. We will further comment on this topic in Chapter 8, where we present an alternative nearfield compression scheme.

REMARK 5.14. *The FMM for the matrices K_h, $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ and D_h is obtained from Algorithm 5.3 by replacing the S2M and L2T operations in lines 5 and 23, respectively, by the appropriate operations from Section 5.2.3 and the nearfield operations in line 26 accordingly. Due to this similarity, we focus on the single layer matrix V_h and the corresponding FMM in the following chapters.*

The space-time FMM in Algorithm 5.3 can be seen as a collection of operations in the space-time cluster tree $\mathcal{T}_\Sigma$. The S2M and L2T operations (lines 5 and 23, respectively) act on leaf clusters in $\mathcal{T}_\Sigma$. The M2M operations (lines 9 and 11), and L2L operations (lines 19 and 21) are operations between clusters and their children, i.e. they take place between clusters in $\mathcal{T}_\Sigma$ that are connected by an edge. The remaining M2L operations (line 15) and nearfield operations (line 26) act on pairs of clusters in $\mathcal{T}_\Sigma$ that are identified by the corresponding operation lists. In the next chapter we show that the FMM can even be interpreted as a collection of operations in a purely temporal tree structure. Based on this interpretation, we develop a parallel version of the FMM in shared and distributed memory.

Algorithm 5.3 The space-time FMM for the approximate evaluation of $\boldsymbol{f} = \mathsf{V}_h \boldsymbol{q}$.

Require: Let a space-time box cluster tree $\mathcal{T}_\Sigma$ as in Algorithm 5.1 be given.
Let the operation lists be constructed by Algorithm 5.2.
Let the expansion degrees m_t and $m_{\boldsymbol{x}}$ be given.

Initialize $\boldsymbol{f} = \boldsymbol{0}$
// Forward transformation
for all $Z \in \mathcal{T}_\Sigma$
  Initialize the moments by setting $\boldsymbol{\mu}(Z) = \boldsymbol{0}$
for all leaves $Z \in \mathcal{T}_\Sigma$
  S2M: Compute $\boldsymbol{\mu}(Z)$ by (5.38).
for all levels $\ell = \operatorname{depth}(\mathcal{T}_\Sigma), \ldots, 1$
  for all $Z \in \mathcal{T}_\Sigma$ with $\ell(Z) == \ell$
    if Z results from $\operatorname{par}(Z)$ by a temporal subdivision:
      Temporal M2M: Add the result $\boldsymbol{\mu}(\operatorname{par}(Z), Z)$ from (5.48) to $\boldsymbol{\mu}(\operatorname{par}(Z))$.
    else
      Space-time M2M:
      Add the result $\boldsymbol{\mu}(\operatorname{par}(Z), Z)$ from (5.49) to $\boldsymbol{\mu}(\operatorname{par}(Z))$.
// Multiplication phase
for all boxes $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma$
  Initialize the local contributions by setting $\boldsymbol{\lambda}(Z_{\mathrm{tar}}) = \boldsymbol{0}$.
  for all boxes $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$
    M2L: Add the result $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ from (5.39) to $\boldsymbol{\lambda}(Z_{\mathrm{tar}})$.
// Backward transformation
for all levels $\ell = 1, \ldots, \operatorname{depth}(\mathcal{T}_\Sigma)$
  for all boxes $Z \in \mathcal{T}_\Sigma$ with $\ell(Z) == \ell$
    if Z results from $\operatorname{par}(Z)$ by a temporal subdivision:
      Temporal L2L: Add the result $\boldsymbol{\lambda}(Z, \operatorname{par}(Z))$ from (5.50) to $\boldsymbol{\lambda}(Z)$.
    else
      Space-time L2L: Add the result $\boldsymbol{\lambda}(Z, \operatorname{par}(Z))$ from (5.51) to $\boldsymbol{\lambda}(Z)$.
for all leaves $Z \in \mathcal{T}_\Sigma$
  L2T: Evaluate $\boldsymbol{\lambda}(Z)$ by (5.40) and add the result to $\boldsymbol{f}|_{\hat{Z}}$.
// Nearfield evaluation
for all $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma$
  for all $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$
    Nearfield operation: Add the product $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}} \boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.

6 A task based parallelization of the space-time FMM

An advantage of space-time methods over classical time stepping methods for the solution of time dependent partial differential equations is that they allow for a parallelization in space and time. Such a parallelization has been considered for standard space-time boundary element methods for the heat equation in [29], but to the best of our knowledge, the parallelization of a related fast method like the space-time FMM presented in Chapter 5 has not been considered yet. However, a lot of attention has been devoted to the parallelization of FMM algorithms for purely spatial problems; see e.g. the non-exhaustive list in the general introduction in Chapter 1. The task based parallelization of the space-time FMM for the heat equation, which we discuss in this chapter, can be best compared with the task based parallelization in [4, 5], but it is tailored to exploit the specific temporal structure of the space-time FMM.

The temporal structure of the space-time FMM allows us to group FMM operations in the original space-time box cluster tree $\mathcal{T}_\Sigma$ to obtain a temporal version of the FMM with operations in a suitable time cluster tree $\mathcal{T}_I$. We define tasks corresponding to the operations in $\mathcal{T}_I$ and use a task scheduler to execute them in parallel based on individual dependencies. For a distributed memory parallelization the clusters in the tree $\mathcal{T}_I$ and the corresponding tasks are distributed among the available processes. The inter-process communication required for this kind of parallelization can be handled in the temporal tree in an asynchronous way, which allows to overlap communication and computation. By defining the tasks as well as the communication in the time tree $\mathcal{T}_I$ instead of the original space-time tree $\mathcal{T}_\Sigma$, the resulting dependencies between tasks and communication patterns between processes are simple and can be handled efficiently. To increase the parallel performance we subdivide tasks in $\mathcal{T}_I$ into smaller, concurrently executable tasks during their execution.

In the following sections we present the described parallelization strategy which was originally published in [76] in more detail. We focus on the FMM for the matrix V_h, but the FMM for the matrices K_h, $\mathsf{K}_h^{\top_x}$ and D_h can be parallelized in the same way. For our description and implementation we use basic concepts of high performance computing like multithreading for shared memory parallelization and the parallel execution of programs by multiple processes for distributed memory parallelization.

We assume that the reader is familiar with these concepts and has a basic understanding of the standard programming interfaces OpenMP and MPI which we use. A good overview of these topics and high performance computing in general is given in [70].

The rest of this chapter is structured as follows. In Section 6.1 we present a temporal version of the space-time FMM on which the parallelization is based. The parallelization is first described for shared memory systems in Section 6.2. In Section 6.3 we discuss how to extend this parallel algorithm to distributed memory systems. The related data and workload distribution is the topic of Section 6.4. The chapter is concluded by Section 6.5, where we present numerical experiments in which we investigate the parallel performance of the introduced algorithms in shared and distributed memory. The corresponding C++ implementation is publicly available in the library besthea [49].

6.1 A temporal version of the space-time FMM

In [68, Section 5.6] it was pointed out that the general structure of the considered parabolic FMM for the heat equation corresponds to the one of a one-dimensional FMM in the time domain. The same holds for the space-time FMM in this work due to its close relation to the original parabolic FMM. We emphasize this here by working with temporal projections to introduce a temporal version of the space-time FMM in Algorithm 5.3. This version forms the basis for the parallelization considered in the following sections.

The temporal projection $\Pi_t : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ is defined by $\Pi_t(\boldsymbol{x}, t) = t$ for all $\boldsymbol{x} \in \mathbb{R}^3$ and $t \in \mathbb{R}$. For a set $A \subset \mathbb{R}^3 \times \mathbb{R}$ we define the image of A under Π_t as

$$\Pi_t[A] = \{\Pi_t(a) : a \in A\}. \tag{6.1}$$

In particular, there holds $\Pi_t[Z] = I$ for a space-time box $Z = X \times I$.

With the projection operator Π_t we can define the temporal projection of a space-time box cluster tree $\mathcal{T}_\Sigma$.

Definition 6.1 (Temporal projection of $\mathcal{T}_\Sigma$). *Let $\mathcal{T}_\Sigma$ be a space-time box cluster tree constructed by Algorithm 5.1. The temporal projection $\mathcal{T}_I := \Pi_t[\mathcal{T}_\Sigma]$ of $\mathcal{T}_\Sigma$ is a graph that is defined as follows:*

- *The vertices of $\mathcal{T}_I$ are the distinct intervals in $\{I : \exists Z \in \mathcal{T}_\Sigma$ with $\Pi_t[Z] = I\}$.*
- *Two intervals I_1 and I_2 in $\mathcal{T}_I$ are connected by an edge in $\mathcal{T}_I$ if there exist clusters Z_1 and Z_2 in $\mathcal{T}_\Sigma$ such that $\Pi_t[Z_1] = I_1$, $\Pi_t[Z_2] = I_2$, and there is an edge between Z_1 and Z_2 in $\mathcal{T}_\Sigma$.*

In the following, we consider a fixed space-time box cluster tree $\mathcal{T}_\Sigma$ constructed by Algorithm 5.1 and denote its temporal projection by $\mathcal{T}_I$. The vertices of $\mathcal{T}_I$ are referred to as intervals, clusters, or time clusters. We use the same standard terms of graph theory for clusters in $\mathcal{T}_I$ as we do for clusters in $\mathcal{T}_\Sigma$. We define the root of $\mathcal{T}_I$ by $I^{(0)} = \Pi_t[Z^{(0)}]$, where $Z^{(0)}$ is the root of $\mathcal{T}_\Sigma$. Then $\mathcal{T}_I$ is a binary tree that is full but not necessarily perfect. This follows immediately from the definition of $\mathcal{T}_I$ and the subdivision strategy in Algorithm 5.1. Furthermore there holds

- If $I \in \mathcal{T}_I$ and $Z \in \mathcal{T}_\Sigma$ with $\Pi_t[Z] = I$, the level $\ell(I)$ of I in $\mathcal{T}_I$ corresponds to the temporal level $\ell_t(Z)$ of Z in $\mathcal{T}_\Sigma$.
- $I_\mathrm{c} \in \mathrm{child}(I)$ if and only if there exist clusters $Z, Z_\mathrm{c} \in \mathcal{T}_\Sigma$ with $\Pi_t[Z] = I$, $\Pi_t[Z] = I_\mathrm{c}$ and $Z_\mathrm{c} \in \mathrm{child}(Z)$.

A cluster $Z \in \mathcal{T}_\Sigma$ is associated with $I \in \mathcal{T}_I$ if $\Pi_t[Z] = I$. The set of all clusters in $\mathcal{T}_\Sigma$ associated with $I \in \mathcal{T}_I$ is defined by

$$\mathcal{Z}_\mathrm{assoc}(I) = \{Z \in \mathcal{T}_\Sigma : \Pi_t[Z] = I\}. \tag{6.2}$$

Operation lists for time clusters I_tar in $\mathcal{T}_I$ are defined via projections of the operation lists of clusters Z_tar in $\mathcal{T}_\Sigma$, which are constructed in Algorithm 5.2 and refined according to Remark 5.11. The interaction list of I_tar is defined by

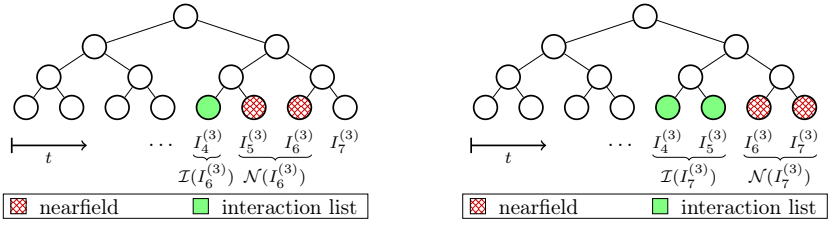
$$\begin{aligned}\mathcal{I}_\mathrm{M2L}(I_\mathrm{tar}) := \{I_\mathrm{src} \in \mathcal{T}_I : \exists Z_\mathrm{tar} \in \mathcal{T}_\Sigma \text{ and } Z_\mathrm{src} \in \mathcal{I}_\mathrm{M2L}(Z_\mathrm{tar}) \\ \text{with } \Pi_t[Z_\mathrm{src}] = I_\mathrm{src} \text{ and } \Pi_t[Z_\mathrm{tar}] = I_\mathrm{tar}\}\end{aligned} \tag{6.3}$$

and its nearfield by

$$\begin{aligned}\mathcal{N}(I_\mathrm{tar}) := \{I_\mathrm{src} \in \mathcal{T}_I : \exists Z_\mathrm{tar} \in \mathcal{T}_\Sigma \text{ and } Z_\mathrm{src} \in \mathcal{N}(Z_\mathrm{tar}) \\ \text{with } \Pi_t[Z_\mathrm{src}] = I_\mathrm{src} \text{ and } \Pi_t[Z_\mathrm{tar}] = I_\mathrm{tar}\}.\end{aligned} \tag{6.4}$$

Note that both operation lists contain only causally relevant clusters I_src. Furthermore, I_src is contained in $\mathcal{I}_\mathrm{M2L}(I_\mathrm{tar})$ only if I_src and I_tar satisfy the admissibility criterion (5.12) and $\mathrm{par}(I_\mathrm{src})$ and $\mathrm{par}(I_\mathrm{tar})$ do not. Likewise, I_src is contained in $\mathcal{N}(I_\mathrm{tar})$ only if (5.12) is violated for I_src and I_tar. As in the space-time setting, the levels $\ell(I_\mathrm{tar})$ and $\ell(I_\mathrm{src})$ coincide if I_src is in $\mathcal{I}_\mathrm{M2L}(I_\mathrm{tar})$. The nearfield list $\mathcal{N}(I_\mathrm{tar})$ of a cluster I_tar can also contain clusters I_src with levels $\ell(I_\mathrm{src}) < \ell(I_\mathrm{tar})$, even if I_src is not a leaf in $\mathcal{T}_I$. This is possible, if $\mathcal{Z}_\mathrm{assoc}(I_\mathrm{src})$ contains a leaf Z_src in $\mathcal{T}_\Sigma$ which is in the nearfield of a cluster $Z_\mathrm{tar} \in \mathcal{Z}_\mathrm{assoc}(I_\mathrm{tar})$. An illustration of nearfield and interaction lists of time clusters in a tree $\mathcal{T}_I$ is given in Figure 6.1.

We want to use the time clusters in $\mathcal{T}_I$ and the related operation lists to define a temporal version of the FMM in Algorithm 5.3. That algorithm consists of a collection of operations in the space-time box cluster tree $\mathcal{T}_\Sigma$. In the following we define corresponding operations in the temporal tree $\mathcal{T}_I$ by grouping FMM operations in $\mathcal{T}_\Sigma$ based on the lists of associated clusters $\mathcal{Z}_\mathrm{assoc}(I)$ for $I \in \mathcal{T}_I$. The sets of moments and local contributions of clusters in $\mathcal{T}_I$ are defined by a grouping as well.



a) Interaction and nearfield lists of $I_6^{(3)}$. b) Interaction and nearfield lists of $I_7^{(3)}$.

Figure 6.1: Illustration of the interaction lists (6.3) and nearfields (6.4) of clusters in the temporal projection $\mathcal{T}_I$ of a suitable tree $\mathcal{T}_\Sigma$. The time clusters/intervals are sorted in ascending order on each level in the figure. The depiction of the interaction lists is representative of trees with uniformly sized intervals on each level and the constant $\eta_2 = 1$ used for the admissibility criterion (5.12). The depicted nearfield lists in $\mathcal{T}_I$ can contain additional clusters in general. For example, the nearfield list $\mathcal{N}(I_6^{(3)})$ of the cluster $I_6^{(3)}$ in a) might also contain $\mathrm{par}(I_5^{(3)})$ in addition to $I_5^{(3)}$ if $\mathcal{Z}_{\mathrm{assoc}}(\mathrm{par}(I_5^{(3)}))$ contains leaf clusters in $\mathcal{T}_\Sigma$.

DEFINITION 6.2. *The moment set $\underline{\boldsymbol{\mu}}(I)$ of a cluster I in $\mathcal{T}_I$ is defined as the set of the moments of its associated space-time clusters, i.e.*

$$\underline{\boldsymbol{\mu}}(I) := \{\boldsymbol{\mu}(Z) : Z \in \mathcal{Z}_{\mathrm{assoc}}(I)\}. \tag{6.5}$$

The local contribution set $\underline{\boldsymbol{\lambda}}(I)$ is defined similarly by

$$\underline{\boldsymbol{\lambda}}(I) := \{\boldsymbol{\lambda}(Z) : Z \in \mathcal{Z}_{\mathrm{assoc}}(I)\}. \tag{6.6}$$

For each FMM operation in $\mathcal{T}_\Sigma$ we define corresponding operations in $\mathcal{T}_I$:

(i) S2M operations for I: For each leaf cluster $Z \in \mathcal{Z}_{\mathrm{assoc}}(I)$ compute the corresponding moments in $\underline{\boldsymbol{\mu}}(I)$ by the S2M operation (5.38).

(ii) M2M operations between I and $\mathrm{par}(I)$: For each $Z \in \mathcal{Z}_{\mathrm{assoc}}(I)$ compute the moments $\boldsymbol{\mu}(\mathrm{par}(Z), Z)$ by an M2M operation (either (5.48) *or* (5.49)*, depending on the configuration of $\mathrm{par}(Z)$ and Z) and add them to the corresponding moments in $\underline{\boldsymbol{\mu}}(\mathrm{par}(I))$.*

(iii) M2L operations between I_{tar} and $I_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$: For each $Z_{\mathrm{tar}} \in \mathcal{Z}_{\mathrm{assoc}}(I_{\mathrm{tar}})$ and each $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ with $\Pi_t[Z_{\mathrm{src}}] = I_{\mathrm{src}}$ compute $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ by the M2L operation (5.39) *and add the result to the corresponding local contributions in $\underline{\boldsymbol{\lambda}}(I_{\mathrm{tar}})$.*

(iv) L2L operations between $\mathrm{par}(I)$ *and* I*: For each* $Z \in \mathcal{Z}_{\mathrm{assoc}}(I)$ *compute the local contributions* $\boldsymbol{\lambda}(Z, \mathrm{par}(Z))$ *by an L2L operation (either* (5.50) *or* (5.51)*, depending on the configuration of* $\mathrm{par}(Z)$ *and* Z*) and add them to the corresponding local contributions in* $\underline{\boldsymbol{\lambda}}(I)$*.*

(v) L2T operations for I*: For each leaf cluster* Z *in* $\mathcal{Z}_{\mathrm{assoc}}(I)$ *evaluate the corresponding local contributions in* $\underline{\boldsymbol{\lambda}}(I)$ *by the L2T operation* (5.40)*.*

(vi) Nearfield operations between I_{tar} *and* $I_{\mathrm{src}} \in \mathcal{N}(I_{\mathrm{tar}})$*: For each* $Z_{\mathrm{tar}} \in \mathcal{Z}_{\mathrm{assoc}}(I_{\mathrm{tar}})$ *and each* $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$ *with* $\Pi_t[Z_{\mathrm{src}}] = I_{\mathrm{src}}$ *execute the corresponding nearfield operation; see line 26 of Algorithm 5.3.*

Algorithm 6.1 The temporal version of the space-time FMM in Algorithm 5.3.

Require: Let all requirements from Algorithm 5.3 be met.
Let $\mathcal{T}_I$ be the temporal projection of $\mathcal{T}_\Sigma$.
1: Initialize $\boldsymbol{f} = \mathbf{0}$.
// Forward transformation
2: **for** all $I \in \mathcal{T}_I$
3: Initialize all moments in $\underline{\boldsymbol{\mu}}(I)$ by zeros.
4: **for** all levels $\ell = \mathrm{depth}(\mathcal{T}_I), \ldots, 1$
5: **for** all $I \in \mathcal{T}_I$ with $\ell(I) == \ell$
6: **if** $\mathcal{Z}_{\mathrm{assoc}}(I)$ contains leaves in $\mathcal{T}_\Sigma$
7: Execute S2M operations for I (Definition 6.2 (i)).
8: Execute M2M operations for I and $\mathrm{par}(I)$ (Definition 6.2 (ii)).
// Multiplication phase
9: **for** all $I_{\mathrm{tar}} \in \mathcal{T}_I$
10: Initialize the local contributions in $\underline{\boldsymbol{\lambda}}(I)$ by zeros.
11: **for** all $I_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$
12: Execute M2L operations for I_{tar} and I_{src} (Definition 6.2 (iii)).
// Backward transformation
13: **for** all levels $\ell = 1, \ldots, \mathrm{depth}(\mathcal{T}_\Sigma)$
14: **for** all $I \in \mathcal{T}_I$ with $\ell(I) == \ell$
15: Execute L2L operations for $\mathrm{par}(I)$ and I (Definition 6.2 (iv)).
16: **if** $\mathcal{Z}_{\mathrm{assoc}}(I)$ contains leaves in $\mathcal{T}_\Sigma$
17: Execute L2T operations for I (Definition 6.2 (v)).
// Nearfield evaluation
18: **for** all $I_{\mathrm{tar}} \in \mathcal{T}_I$
19: **for** all $I_{\mathrm{src}} \in \mathcal{N}(I_{\mathrm{tar}})$
20: Execute nearfield operations for I_{tar} and I_{src} (Definition 6.2 (vi)).

REMARK 6.3. *S2M and L2T operations have to be executed for all leaves in the temporal tree $\mathcal{T}_I$, but also for some non-leaf clusters. In fact, even for a non-leaf cluster I the set of associated space-time clusters $\mathcal{Z}_{\text{assoc}}(I)$ might contain leaves in $\mathcal{T}_\Sigma$ for which S2M and L2T operations have to be executed.*

In Algorithm 6.1 we present the temporal version of the space-time FMM from Algorithm 5.3. The two algorithms are equivalent, which is clear by construction. The new temporal version allows us to view the FMM as a collection of operations in the temporal tree $\mathcal{T}_I$. This view reveals the temporal structure of the FMM, which we exploit in the parallelization described in the subsequent sections.

6.2 A task based shared memory parallelization

In this section we restructure the temporal version of the space-time FMM presented in Algorithm 6.1 to obtain a task based execution scheme where groups of FMM operations are executed based on individual dependencies. We discuss a shared memory parallelization of this approach. The extension to obtain a distributed memory parallelization is discussed in Section 6.3.

In Algorithm 5.3 and its temporal version Algorithm 6.1 we separate the FMM operations into four distinct phases. This ordering of operations allows for a concise and comprehensible presentation of the algorithms. In addition, it ensures that the FMM operations are processed in a correct order when they are executed by a single thread. However, this strict separation of phases might be suboptimal when it comes to parallelization. Let us, for example, consider the forward transformation phase of the space-time FMM in Algorithm 5.3. Here we compute the moments of all clusters in the space-time tree $\mathcal{T}_\Sigma$. After the computation of the moments of all leaf clusters, we proceed with the computation of the remaining moments by M2M operations in a level-wise manner starting with the largest level $\ell = \text{depth}(\mathcal{T}_\Sigma)$. The number of clusters per level decreases with decreasing levels and thereby the number of operations per level decreases. As a consequence, it may not be possible to distribute the operations of the forward transformation phase equally among the available parallel threads on all levels of the tree $\mathcal{T}_\Sigma$ which may cause undesired idle times during their execution.

By breaking the strict separation of phases in the FMM we can obtain more flexibility in its parallel execution. For example, we could execute some of the M2M operations of the forward transformation phase in parallel with some of the M2L operations of the multiplication phase, for which the required moments are already available. To make this possible, we consider Algorithm 6.1 and decompose it into groups of tasks that can be executed based on individual dependencies. These dependencies can be

determined by taking a closer look at the FMM operations in the temporal tree $\mathcal{T}_I$ in Algorithm 6.1, which are given in Definition 6.2. We can subdivide them into 3 classes based on their input data:

(i) Operations that require parts of the source vector $\boldsymbol{q}$: These are the S2M operations and the nearfield operations in Definition 6.2 (i) and (vi), respectively. We assume that the source vector $\boldsymbol{q}$ is always accessible without any restrictions.

(ii) Operations that require the set of moments $\underline{\boldsymbol{\mu}}(I)$ of a cluster $I \in \mathcal{T}_I$ to be computed: These are the M2M operations between I and $\mathrm{par}(I)$, and the M2L operations between $I_{\mathrm{tar}} \in \mathcal{T}_I$ and $I \in \mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$ in Definition 6.2 (ii) and (iii), respectively.

(iii) Operations that require the set of local contributions $\underline{\boldsymbol{\lambda}}(I)$ of $I \in \mathcal{T}_I$ to be computed: These are the L2L operations between I and $I_{\mathrm{c}} \in \mathrm{child}(I)$, and the L2T operations for I in Definition 6.2 (iv) and (v), respectively.

In principle, an FMM operation can be executed once its input data are available. Hence, we can execute the FMM operations in Algorithm 6.1 in a more flexible order, as long as we respect the above data dependencies. For this purpose, we introduce so-called *FMM tasks* for clusters $I \in \mathcal{T}_I$. By grouping some of the FMM operations we obtain the following four kinds of FMM tasks:

- The M-list task for $I \in \mathcal{T}_I$ which comprises the S2M operations for I and M2M operations between I and $\mathrm{par}(I)$.
- The M2L-list task for $I \in \mathcal{T}_I$ which includes the M2L operations between I and all $I_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(I)$, and additionally the L2T operations for I, if the list of associated clusters $\mathcal{Z}_{\mathrm{assoc}}(I)$ contains leaf clusters.
- The L-list task for $I \in \mathcal{T}_I$ which includes the L2L operations between $\mathrm{par}(I)$ and I, and additionally the L2T operations for I, if the list of associated clusters $\mathcal{Z}_{\mathrm{assoc}}(I)$ contains leaf clusters.
- The N-list task for $I \in \mathcal{T}_I$ which comprises the nearfield operations between I and all $I_{\mathrm{src}} \in \mathcal{N}(I)$.

As the names of the tasks suggest we collect them in four different lists. We will use a task scheduler to manage their execution in the later algorithm.

Note that the L2T operations for I are included in the M2L-list task as well as in the L-list task for I, but may be executed only once the local contributions $\underline{\boldsymbol{\lambda}}(I)$ are fully available. Recall that these local contributions are computed in parts by the M2L operations between I and $I_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(I)$ in the M2L-list task for I and in parts by the L2L operations between $\mathrm{par}(I)$ and I in the L-list task for I. These L2L and M2L operations are independent of each other. Hence, we can process the M2L- and

L-list tasks for I in an arbitrary order and execute the L2T operations in the task that is executed last.

Let us now describe the task based execution model for the FMM in Algorithm 6.1. Before we present the full algorithm, we discuss the data dependencies of the above FMM tasks, the construction of the corresponding task lists, the basic task scheduling procedure, and the desired parallel execution scheme.

Data dependencies of the tasks and the construction of the task lists. The data dependencies of the different FMM tasks introduced above correspond to the dependencies of the associated FMM operations. We say that a task A depends on some data B if A can be executed only if B is fully available. The FMM tasks have the following dependencies:

- The M-list task for I depends on the moments $\underline{\boldsymbol{\mu}}(I)$. If I does not have any children, these moments are computed by the S2M operations in the M-list task of I, so this task does not have any dependencies. Otherwise, the moments are computed in parts by the M-list tasks of the children of I.
- The M2L-list task for I depends on the moments $\underline{\boldsymbol{\mu}}(I_{\mathrm{src}})$ of all time clusters I_{src} in the interaction list $\mathcal{I}_{\mathrm{M2L}}(I)$.
- The L-list task for I depends on the local contributions $\underline{\boldsymbol{\lambda}}(\mathrm{par}(I))$ of its parent.
- The N-list task for I does not have any data dependencies, since we assume that the source vector $\boldsymbol{q}$ is always fully available.

To keep track of all the tasks which need to be executed in the FMM we construct four task lists, namely the *M-list*, *M2L-list*, *L-list*, and *N-list*. These lists are filled with the corresponding FMM tasks introduced above. The individual tasks in each list are uniquely identified by the corresponding clusters in $\mathcal{T}_I$ on which they act, so we can interpret them as lists of clusters as well. A cluster $I \in \mathcal{T}_I$ is added to a list if the corresponding task has to be executed for I.

A cluster I is added to the M2L-list only if the interaction list $\mathcal{I}_{\mathrm{M2L}}(I)$ is not empty, and to the N-list only if the nearfield list $\mathcal{N}(I)$ is not empty. Also M-list and L-list tasks do not necessarily have to be executed for all clusters in $\mathcal{T}_I$. In fact, we need to execute an M-list task for a cluster only if its moments are needed in the FMM. We can determine these clusters, and simultaneously fill the M-list by a recursive traversal of the tree $\mathcal{T}_I$ starting at its root. During the recursion we add a cluster I to the M-list if I is in the interaction list $\mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$ of another cluster $I_{\mathrm{tar}} \in \mathcal{T}_I$, or if $\mathrm{par}(I)$ is in the M-list. Likewise, we need to execute an L-list task for a cluster only if its local contributions are needed in the FMM. We fill the L-list by a recursive tree traversal, where we add a cluster I to the L-list if the interaction list $\mathcal{I}_{\mathrm{M2L}}(\mathrm{par}(I))$ of $\mathrm{par}(I)$ is not empty or if $\mathrm{par}(I)$ is in the L-list.

The basic task execution procedure. Once the task lists have been determined, we can start with the task based execution of the FMM operations. For this purpose, we need a task scheduler. The task scheduler has to check the data dependencies of the individual tasks and keep track of open tasks. In our implementation we use a simple custom scheduler for this purpose instead of an existing solution like the scheduler provided in OpenMP for the corresponding task constructs or a third party solution like, e.g., StarPU [9]. The primal motivation for our decision is that the task scheduler has to be suitable for the later distributed memory parallelization. In particular, it has to be able to handle communication and dependencies between different MPI processes. The OpenMP task scheduler (OpenMP API 5.0 specification [58]) is not suitable for this purpose, so we decided to use a custom solution to avoid dependencies on third party software.

Our custom scheduler iterates through the four lists of tasks. If it finds a task whose dependencies are satisfied, this task can be executed and removed from the list. When all lists are empty, all tasks and, therefore, all FMM operations from Algorithm 6.1 have been executed.

The parallel, multithreaded execution of tasks. The reason why we consider a new task based version of the FMM is to enable an efficient parallelization. Therefore, let us describe how the FMM tasks, i.e. the M-, M2L-, L-, and N-list tasks, can be executed in parallel. As described in the last paragraph, we use a custom task scheduler to keep track of the data dependencies of the FMM tasks and initiate FMM tasks whose dependencies are satisfied. For the parallel, multithreaded execution of these tasks we use the OpenMP `task` construct in our implementation. The following description is based on these OpenMP `tasks`, but other task based execution models could be used for the implementation as well. To distinguish the FMM tasks from OpenMP `tasks`, we use the term FMM task for the former and denote the latter in monospaced font from now on.

OpenMP `tasks` are generated with certain explicit constructs in a code, like the `task` and `taskloop` constructs which we encounter and describe in more detail later. We can imagine that `tasks` created by such constructs are added to a general pool of `tasks` and can be executed by all available threads concurrently. Various `tasks` of different kinds can be added to the pool of `tasks` at the same time which leads to a high flexibility of this parallelization approach. It is even possible to create `tasks` in a nested way, i.e. when executing a `task` a thread can generate new `tasks` which are added to the global pool of `tasks`. The actual task execution is managed by the task scheduler of the OpenMP runtime system that assigns tasks to the available threads.

In our task based version of the FMM we use OpenMP `tasks` as follows. Once our custom task scheduler detects a ready FMM tasks, i.e. one whose dependencies are satisfied, it creates an OpenMP `task` for its execution and continues looking for other

ready FMM tasks. This allows us to execute several ready FMM tasks in parallel. The FMM tasks can be quite large, so we want to further subdivide them. In fact, an FMM task for a time cluster $I \in \mathcal{T}_I$ consists of the related FMM operations for this time cluster which are groups of FMM operations for space-time clusters $Z \in \mathcal{Z}_{\text{assoc}}(I)$; see Definition 6.2. To execute these FMM operations for space-time clusters inside of an FMM task in parallel, we create additional OpenMP `tasks`. In Algorithm 6.2 we present a sketch of the resulting task based version of the FMM.

Algorithm 6.2 A parallel, task based version of the FMM in Algorithm 6.1.

```
1:  Initialize f = 0.
2:  for all I ∈ T_I
3:      Initialize μ(I) and λ(I) by zeros.
4:  Fill the M_list, M2L_list, L_list, and N_list with clusters in T_I.
5:  #pragma omp parallel
6:      #pragma omp single
7:          while the FMM task lists are not empty
8:              [I, list]
                    = FindNextReadyFMMTask(M_list, L_list, M2L_list)
9:              if list == M_list
10:                 Remove I from the M_list.
11:                 #pragma omp task      // + depend clause; see Remark 6.4.
12:                 MListTask(I)
13:             else if list == L_list
14:                 Remove I from the L_list.
15:                 #pragma omp task      // + depend clause; see Remark 6.4.
16:                 LListTask(I)
17:             else if list == M2L_list
18:                 Remove I from the M2L_list.
19:                 #pragma omp task      // + depend clause; see Remark 6.4.
20:                 M2LListTask(I)
21:             else      // No ready task in the M_list, L_list or M2L_list
22:                 if N_list is not empty
23:                     Choose I as the first cluster in the N_list.
24:                     Remove I from the N_list.
25:                     #pragma omp task
26:                     NListTask(I)
```

In the first lines of Algorithm 6.2 we initialize the sets of moments and local contributions of clusters in $\mathcal{T}_I$ and fill the FMM task lists as described above. In line 5 we use the OpenMP parallel pragma to create a parallel region for the execution of `tasks` using multiple threads. The remaining lines in the algorithm describe our custom

task scheduler. The scheduling is handled by a single thread, which is guaranteed by the pragma in line 6.

In the scheduling procedure we first search for a ready FMM task in the M-, L- or M2L-list. For this purpose, we use the routine FindNextReadyFMMTask in line 8 to loop through each of these lists consecutively at most once and check the dependencies of the contained FMM tasks. To be able to identify satisfied dependencies we have to keep track of the fully computed moments and local contributions in $\mathcal{T}_I$, e.g., by using some auxiliary variables which we do not specify here. As soon as a ready FMM task is detected in the routine FindNextReadyFMMTask, the corresponding time cluster I and the name of the list in which it was found are returned.

The pair $[I, \texttt{list}]$ is used to create a new OpenMP `task` for the execution of the related FMM task. Let us assume, for example, that we found a ready FMM task for a cluster I in the M-list. In this case, we jump to line 10, where I is removed from the M-list before initiating the corresponding task with the routine MListTask in line 12. Due to the OpenMP `task` pragma in line 11, this routine is not processed directly, but instead an OpenMP `task` is generated for its execution, which is added to the above mentioned pool of `tasks`. To be more precise, the scheduling thread could immediately execute the `task` or postpone its execution according to the OpenMP API 5.0 specification [58, Section 2.10]. In the latter case, any of the available threads may execute this `task`, while the scheduling thread can continue with the scheduling procedure. The routine MListTask is given in Algorithm 6.3 and discussed in more detail below. Tasks in the L-list and M2L-list are handled analogously. To avoid data races during the parallel execution of the created `tasks`, we introduce additional dependencies between them, which can be handled by the OpenMP task scheduler itself. This is discussed in more detail in Remark 6.4.

The tasks in the N-list are handled separately. Since they do not have any dependencies, they could be scheduled at any time. However, there are not any tasks which depend on N-list tasks, so executing N-list tasks does not have a high priority. Therefore, we consider the N-list only if no task was found in any of the other lists; see line 21. In this case, we schedule the first N-list task in the N-list in the same way as we scheduled the FMM tasks of the other lists.

The scheduling thread repeats the steps in lines 8–26 of Algorithm 6.1 and schedules new FMM tasks once they are ready. When all task lists are empty and all the scheduled tasks have been executed, the application of the FMM is complete and we obtain the same approximation $\boldsymbol{f}$ of the matrix-vector product $\mathsf{V}_h\boldsymbol{q}$ as in Algorithm 5.3.

Remark 6.4 (Additional dependencies between FMM tasks). *We have to ensure that the parallel execution of FMM tasks in Algorithm 6.2 works correctly without any data races between the different threads. Special care is necessary for M-list*

tasks of clusters I_1 and I_2 that share the same parent, i.e. $I_{\text{par}} = \text{par}(I_1) = \text{par}(I_2)$, because the results of the corresponding M2M operations are written to the same set of moments $\underline{\boldsymbol{\mu}}(I_{\text{par}})$. To avoid race conditions we can either use atomic operations for the writing access of these operations, or prohibit the simultaneous execution of such M-list tasks. In our implementation we choose the latter approach for which we use the `depend` *clause for OpenMP* `tasks`*; see [58, Section 2.17.11]. With this clause we introduce dependencies between* `tasks` *which ensure that they are executed in a sequential order. These additional dependencies are handled by the OpenMP task scheduler itself, in contrast to the data dependencies of FMM tasks that we introduced above and handle manually in our custom task scheduler. Recall that this was a design choice to enable the later distributed memory parallelization. Additional dependencies are also introduced to prohibit the simultaneous execution of the L-list and M2L-list tasks for a cluster I since the corresponding operations affect the same set of local contributions $\underline{\boldsymbol{\lambda}}(I)$.*

The routines for the execution of the FMM tasks in Algorithm 6.2 are specified in Algorithms 6.3 and 6.4. In these routines the actual FMM operations in the space-time tree $\mathcal{T}_\Sigma$ are handled. The routine MListTask in line 1 of Algorithm 6.3 is used to execute S2M and M2M operations for a time cluster I. The S2M operation for I has to be executed only if the set $\mathcal{Z}_{\text{assoc}}(I)$ of space-time clusters associated with I contains leaves in $\mathcal{T}_\Sigma$. In this case, the S2M operations of all associated leaves are executed in lines 4–5. To parallelize them, we use the `taskloop` construct of OpenMP in line 3; see [58, Section 2.10.2]. Due to this directive, the iterations in the subsequent for loop are subdivided into `tasks` that are added to the global pool of `tasks` and handled by the task scheduler of the OpenMP runtime system. A concurrent execution of the loop iterations is possible here without any restrictions, since in each iteration a separate vector of moments $\boldsymbol{\mu}(Z)$ is computed. The moments computed by the S2M operations are needed for the subsequent M2M operations, so we have to ensure that the main thread executing the routine MListTask waits for the completion of the S2M operations before it proceeds with the steps in lines 6–9. By default, the `taskloop` pragma in OpenMP introduces a so-called `taskgroup` which guarantees this; see [58, Section 2.17.6]. The following M2M operations for clusters $Z \in \mathcal{Z}_{\text{assoc}}(I)$ are grouped with respect to their parent clusters. This leads to the nested loops in lines 7 and 8. The outer loop is parallelized by the `taskloop` directive again. Due to the grouping of operations, the generated `tasks` can be executed concurrently without any restrictions.

The other FMM tasks are realized analogously. For the L2L operations for a cluster I in the routine LListTask of Algorithm 6.3 we use again a grouping of clusters $Z \in \mathcal{Z}_{\text{assoc}}(I)$ with respect to their parent clusters as we did for the M2M operations. The L2T operations are executed either in the L-list task or the M2L-list task once the local contributions $\underline{\boldsymbol{\lambda}}(I)$ are fully available, as we already mentioned above. Note

that we use atomic operations [58, see Section 2.17.7] when adding the results of the L2T operations to the output vector $\boldsymbol{f}$ to avoid race conditions when several threads try to access the same entries of $\boldsymbol{f}$ at the same time.

Algorithm 6.3 The M- and L-list tasks from Algorithm 6.2.

```
1:  function MListTask(I)
2:      if Z_assoc(I) contains leaves in T_Σ
            // Execution of the S2M operations for I:
3:          #pragma omp taskloop
4:          for all leaves Z ∈ Z_assoc(I)
5:              S2M: Compute μ(Z) in the set μ_(I) by (5.38).
        // Execution of the M2M operations between I and par(I):
6:      #pragma omp taskloop
7:      for all Z_par ∈ Z_assoc(par(I))
8:          for all Z ∈ child(Z_par) with Π_t[Z] = I
9:              M2M: Compute μ(Z_par, Z) (by (5.48) or (5.49) as appropriate).
                Add μ(Z_par, Z) to μ(Z_par) in μ_(par(I)).

10: function LListTask(I)
        // Execution of the L2L operations between par(I) and I:
11:     #pragma omp taskloop
12:     for all Z_par ∈ Z_assoc(par(I))
13:         for all Z ∈ child(Z_par) with Π_t[Z] = I
14:             L2L: Compute λ(Z, Z_par) (by (5.50) or (5.51) as appropriate).
                Add λ(Z, Z_par) to λ(Z) in λ_(I).
15:     if Z_assoc(I) contains leaves in T_Σ
16:         if λ_(I) is complete        // Only after potential M2L-list operations.
                // Execution of the L2T operations for I:
17:             #pragma omp taskloop
18:             for all leaves Z ∈ Z_assoc(I)
19:                 L2T: Evaluate λ(Z) in λ_(I) by (5.40).
                    Add the result to f|_Ẑ atomically.
```

The M2L operations of a cluster I in the routine M2LListTask of Algorithm 6.4 are handled by a nested loop; see lines 3–5. In the outer loop we iterate over all associated space-time clusters of I, while in the inner loop we iterate over their interaction lists. In this way, we can use the `taskloop` construct to subdivide the outer loop into groups of `tasks` that can be executed concurrently. The nearfield operations in the routine NListTask are handled analogously. When adding the results of the nearfield operations to the output vector $\boldsymbol{f}$ we use atomic operations again. Note that the assembly of the nearfield blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ used in line 15

involves the numerical evaluation of the integrals in (3.25) which is computationally expensive. Therefore, one should assemble and store these blocks in a preprocessing phase. A simple and efficient way of doing this is using two nested loops. In the outer loop we iterate over all clusters $Z_{\text{tar}} \in \mathcal{T}_\Sigma$ with non-empty nearfield lists. The nearfield blocks corresponding to Z_{tar} and all clusters $Z_{\text{src}} \in \mathcal{N}(Z_{\text{src}})$ are computed in the inner loop. The assembly can be parallelized by using the OpenMP `for` construct [58, see Section 2.9.2] for the outer loop with a dynamic scheduling with chunk size one. This way each available thread is assigned a single cluster Z_{tar}, assembles all nearfield blocks corresponding to Z_{tar} and requests a new cluster afterwards. A good workload balance is achieved with this parallelization strategy if the target clusters with non-empty nearfield lists are sorted according to the total sizes of their nearfield blocks before the parallel assembly.

Algorithm 6.4 The M2L- and N-list tasks from Algorithm 6.2.

function M2LListTask(I)
  // Execution of the M2L operations for I:
  `#pragma omp taskloop`
  for all $Z_{\text{tar}} \in \mathcal{Z}_{\text{assoc}}(I)$
    for all $Z_{\text{src}} \in \mathcal{I}_{\text{M2L}}(Z_{\text{tar}})$
      M2L: Compute $\boldsymbol{\lambda}(Z_{\text{tar}}, Z_{\text{src}})$ by (5.39) and add it to $\boldsymbol{\lambda}(Z_{\text{tar}})$ in $\underline{\boldsymbol{\lambda}}(I)$.
  if $\mathcal{Z}_{\text{assoc}}(I)$ contains leaves in $\mathcal{T}_\Sigma$
    if $\underline{\boldsymbol{\lambda}}(I)$ is complete *// Only after potential L-list operations.*
      // Execution of the L2T operations for I:
      `#pragma omp taskloop`
      for all leaves $Z \in \mathcal{Z}_{\text{assoc}}(I)$
        L2T: Evaluate $\boldsymbol{\lambda}(Z)$ in $\underline{\boldsymbol{\lambda}}(I)$ by (5.40).
        Add the result to $\boldsymbol{f}|_{\hat{Z}}$ atomically.

function NListTask(I)
  // Execution of the nearfield operations for I:
  `#pragma omp taskloop`
  for all $Z_{\text{tar}} \in \mathcal{Z}_{\text{assoc}}(I)$
    for all $Z_{\text{src}} \in \mathcal{N}(Z_{\text{tar}})$
      Nearfield operation: Add $\mathsf{V}_h|_{\hat{Z}_{\text{tar}} \times \hat{Z}_{\text{src}}} \boldsymbol{q}|_{\hat{Z}_{\text{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\text{src}}}$ atomically.

Remark 6.5. *The task based execution of the FMM in Algorithm 6.2 is based on the temporal version of the FMM in Algorithm 6.1 and uses two levels of tasks. FMM operations for time clusters in $\mathcal{T}_I$ are collected in suitable FMM tasks, which are initiated based on individual dependencies by a lightweight, custom task scheduler and executed using the OpenMP* `task` *construct. Once the FMM tasks in $\mathcal{T}_I$ are executed, they initiate the actual FMM operations in the underlying space-time tree $\mathcal{T}_\Sigma$, which form the second level of tasks. Alternatively one could also define tasks and*

corresponding dependencies directly in the space-time tree $\mathcal{T}_\Sigma$. Such an approach is described in [4] for an FMM related to the Laplace equation in $\mathbb{R}^3$. However, creating tasks for individual clusters in $\mathcal{T}_\Sigma$ leads to a very large number of tasks and dependencies between them, which cannot be handled efficiently anymore in general. Therefore, the authors in [4] proposed to group clusters into similarly sized blocks in a suitable manner and to create tasks for these blocks. In this way, a suitable task granularity can be obtained which allows for an efficient execution of tasks and an efficient management of dependencies between them. Our approach is similar to this block- and task based parallelization of the FMM. With the somewhat natural grouping of clusters with respect to their temporal projections we obtain "blocks" of different sizes but benefit from the simple temporal structure and dependencies between tasks. These two aspects are particularly important for the distributed memory parallelization which we describe in Section 6.3. Note that a distributed memory parallelization of the approach in [4] is also possible and is described in [5].

6.2.1 Additional aspects for a better parallel performance

In our C++ implementation in [49] we consider a few additional aspects to increase the performance of the task based parallel FMM in Algorithm 6.2. Our goal is to reduce the idle times of all threads involved in the parallel execution of `tasks` as much as possible. For this purpose, we influence the order in which FMM tasks are executed by introducing priorities. Furthermore, we bound the number of scheduled N-list tasks in certain situations and use the `taskyield` directive to include the scheduling thread in the computation. The details are given in the following paragraphs.

Assign priorities to FMM tasks. For the parallel performance of Algorithm 6.2 it is beneficial if there are always several FMM tasks scheduled for execution at the same time to keep all available threads busy. We can facilitate this by assigning priorities to FMM tasks to influence the order in which they are executed.

We assign the highest priorities to M-list tasks, the second highest to L-list tasks, the third highest to M2L-list tasks, and the lowest priorities to N-list tasks. This choice is motivated heuristically based on dependencies. FMM tasks computing moments or local contributions on which many other FMM tasks depend — directly or indirectly via chains of dependencies — should be executed as soon as possible. This is why we prioritize M-list tasks the most. L-list tasks are prioritized over M2L-list tasks since their dependencies are more restrictive than those of M2L-list tasks. In fact, the M2L-list task of a cluster I can be executed once the moments of the clusters in $\mathcal{I}_{\mathrm{M2L}}(I)$ are available while the L-list task of a typical cluster I is ready only once the M2L-list task and L-list task of its parent is completed. Hence, we can unlock more new L-list tasks, by focusing on the execution of L-list tasks once they are

ready. N-list tasks obtain the lowest priority since there are not any FMM tasks which depend on them.

We can also distinguish priorities between FMM tasks of the same kind. A natural choice is to use the levels of the corresponding clusters for such a distinction. M-list tasks for clusters with high levels should be executed before M-list tasks for clusters with low levels. For L-list tasks and M2L-list tasks the opposite is true. This choice of priorities corresponds to the natural order in which the corresponding operations are executed in the original Algorithm 6.1. When it comes to N-list tasks, it might be beneficial to execute small N-list tasks first. In fact, we would like to use N-list tasks only as buffers to keep all threads busy, as long as other FMM tasks are available. Large N-list tasks initiated at an early stage might hinder threads from executing other, more important FMM tasks.

We account for the priorities of the FMM tasks in our implementation in several ways. First, we sort the tasks in the individual FMM task lists according to their priorities. E.g., we sort the tasks/clusters in the M-list such that clusters with large levels come first. Secondly, we respect the priorities when searching for new ready tasks in line 8 of Algorithm 6.2 by traversing the M-list first, the L-list second and the M2L-list third. The N-list is only considered if there is not any ready task in any of the other lists. Finally, we use the `priority` clause of OpenMP `tasks` when we schedule the FMM tasks. This clause allows to specify priority values for `tasks`, which can be used by the OpenMP runtime system as hints to determine the execution order; see [58, Section 2.10.1, p. 138]. In our implementation we use different priority values for the different kinds of FMM tasks, but do not distinguish priorities between different tasks of the same kind.

Bound the number of scheduled N-list tasks. Since N-list tasks have the lowest priority we would like to prioritize other FMM tasks in the execution. This is why we consider the N-list in the scheduling process in Algorithm 6.2 only if we do not find a ready FMM task in any of the other lists. If the scheduling thread does not participate in the execution of tasks, it might however schedule many N-list tasks at a very early stage. In fact, once there are no ready M-, M2L-, and L-list tasks, it schedules the first N-list task before jumping to line 8 of Algorithm 6.2 to look again for ready tasks. Since the scheduling of tasks and searching for ready tasks is typically way faster than the execution of these tasks, the scheduling thread is likely to schedule many or even all N-list tasks at once in such a scenario. To avoid such a behavior, we introduced a bound for the number of N-list tasks that can be scheduled concurrently with other FMM tasks in our implementation.

Include the scheduling thread in the computation. The scheduling thread is trapped in the scheduling loop in lines 6–26 of Algorithm 6.2 until all FMM tasks have been scheduled. Hence, it does not participate in the execution of tasks most of the time in general. In [76] we propose to use the `taskyield` construct of OpenMP

to change this behavior. When a thread encounters this construct it can suspend the execution of the current task to work on a different one; see [58, Section 2.10.4]. Note that this is a non-binding request, so it is not guaranteed that the thread actually suspends the current task. In fact, we do not observe any changes when using `taskyield` in the scheduling routine with the `GCC` compiler (v9.3). In the case of the Intel Compiler (v19.1.1 and newer), however, the `taskyield` construct shows the desired effect. More details about the behavior of the `taskyield` construct for various compilers are given in [65].

6.3 A task based distributed memory parallelization

The task based, parallel version of the FMM in Section 6.2 can be extended to allow for a parallel execution on distributed memory systems. In this section we present such an extension, which was originally published in [76]. We start by describing the general idea.

For the data and workload distribution, the time clusters in the temporal projection tree $\mathcal{T}_I$ are distributed among all available processes. In the application of the FMM, each process executes a version of Algorithm 6.2 where it carries out tasks only for clusters that are assigned to it. Since a process might need data computed by other processes for some of its own tasks, communication between processes is necessary. We communicate sets of moments $\underline{\boldsymbol{\mu}}(I)$ and local contributions $\underline{\boldsymbol{\lambda}}(I)$ for clusters $I \in \mathcal{T}_I$ as a whole instead of single moments and local contributions for clusters $Z \in \mathcal{T}_\Sigma$. This reduces the number of communication events and leads to a simple communication pattern in the tree $\mathcal{T}_I$. To hide possible communication delays we overlap the computation and communication by using asynchronous, non-blocking communication routines in MPI.

A more detailed description of this distributed memory parallelization is given in the following subsections. In Section 6.3.1 we make a few assumptions about the data and work distribution for the parallel execution of the FMM, which we assume to be satisfied in all the following subsections. A strategy to obtain such a distribution is presented later in Section 6.4. In Section 6.3.2 we describe how each process determines the FMM tasks that it has to execute and how communication between processes is handled. The distributed, task based version of the FMM is finally presented in Section 6.3.3.

6.3.1 Assumptions about the data distribution

For the distributed, task based FMM we make a few assumptions about the data and work distribution that we assume to be satisfied in the subsequent Sections 6.3.2

and 6.3.3. These assumptions will be discussed in more detail in the following paragraphs and are stated in the following list:

(i) n_{proc} processes are available for the computation. They are identified by corresponding IDs in the set $\{0, \dots, n_{\text{proc}} - 1\}$.

(ii) The space-time tensor product mesh Σ_h is distributed suitably among all available processes.

(iii) An assignment of process IDs to clusters in $\mathcal{T}_I$ is given such that each cluster has exactly one process ID. A process is responsible for a time cluster $I \in \mathcal{T}_I$ if its ID is assigned to this cluster.

(iv) Only sets of moments and local contributions have to be communicated between processes during the parallel, distributed execution of the FMM. The processes have all the additional information that they need to carry out their own tasks and to handle the communication.

From the assignment of process IDs to clusters in the temporal tree $\mathcal{T}_I$ in (iii) we get an assignment of process IDs to clusters in the space-time cluster tree $\mathcal{T}_\Sigma$ as well. In fact, we can assign the process ID of a cluster I to all $Z \in \mathcal{Z}_{\text{assoc}}(I)$. In this way, a process that is responsible for a time cluster I is responsible for all space-time clusters associated with I.

In Assumption (iv) above we mentioned additional information that each process needs for its own tasks in the FMM. The required information includes local mesh information related to the distribution of the mesh Σ_h in Assumption (ii), and information about local parts of the space-time box cluster tree $\mathcal{T}_\Sigma$ and its temporal projection $\mathcal{T}_I$. We specify the required information in the following paragraphs. A strategy for process assignments as in Assumption (iii) and a description of how to provide each process with the required data are given later in Section 6.4.

Locally required mesh information. The individual processes do not need the information of the full space-time tensor product mesh Σ_h. The mesh is distributed among all available processes and each one stores only a local part related to some space-time clusters for which it is responsible. Recall that we assigned space-time elements σ_{j_t, j_x} of the mesh Σ_h to space-time clusters Z in the construction of the space-time box cluster tree $\mathcal{T}_\Sigma$ in Section 5.2.1. It is sufficient if a process p has the mesh information of all space-time elements assigned to the following clusters:

- All leaf clusters $Z \in \mathcal{T}_\Sigma$ for which process p is responsible.
- All clusters Z_{src} such that $Z_{\text{src}} \in \mathcal{N}(Z)$ for a cluster Z for which process p is responsible.

We assume that each process p stores this individual mesh information locally. It is needed for the S2M, L2T, and nearfield operations in the FMM related to clusters Z for which process p is responsible. For these operations a process p needs to access corresponding local parts of the source vector $\boldsymbol{q}$, which we assume to be stored locally as well. Likewise, we assume that each process p stores the local part of the result vector $\boldsymbol{f}$ that it computes in the FMM.

Locally required information about the time tree $\mathcal{T}_I$. Each process p needs only a local part of the full time tree $\mathcal{T}_I$. We call this part the *locally essential time tree* $\mathcal{T}_I^{\mathrm{LE},p}$ of p. The concept of locally essential trees has already been introduced in [75]. In our setting, the locally essential time tree $\mathcal{T}_I^{\mathrm{LE},p}$ is a subtree of $\mathcal{T}_I$ that contains all the clusters and process IDs which process p needs to determine and execute its tasks in the FMM, and the information for the related communication.

For a rigorous definition of the locally essential time tree $\mathcal{T}_I^{\mathrm{LE},p}$ of a process p we define local and essential clusters for p in Definition 6.6. An explanation as to why essential clusters are relevant for the FMM is given in the paragraphs after this definition.

DEFINITION 6.6. *A cluster $I \in \mathcal{T}_I$ is local (for process p) if process p is responsible for I. A cluster I is essential (for p) if it satisfies at least one of the following conditions:*

(i) I *is local.*

(ii) $I \in \mathcal{N}(I_{\mathrm{tar}})$ *for a local cluster* I_{tar}.

(iii) $I \in \mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$ *for a local cluster* I_{tar}.

(iv) There exists a local cluster I_{src} *such that* $I_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(I)$.

(v) $I = \mathrm{par}(I_{\mathrm{c}})$ *for a local cluster* I_{c}.

(vi) $I \in \mathrm{child}(I_{\mathrm{p}})$ *for a local cluster* I_{p}.

The essential clusters for a process p are related to local FMM tasks, i.e. the FMM tasks for local clusters in $\mathcal{T}_I$. A cluster I is defined to be essential either because it is local, or because its data is required for a local FMM task, or because the data which is computed in a local FMM task is needed for an FMM task of I. If a cluster I satisfies (ii) in Definition 6.7 it is obviously essential in this regard because its information is needed for the nearfield operations in the N-list task of a local cluster I_{tar}.

A cluster I satisfying (iii) in Definition 6.7 is essential since its moments $\underline{\boldsymbol{\mu}}(I)$ are needed for the M2L operations in the M2L-list task of the corresponding local cluster I_{tar}. If I is assigned to a different process q, this process q needs to send the moments $\underline{\boldsymbol{\mu}}(I)$ to process p once they are computed. Likewise, process p needs to

send the moments of a local cluster I_{src} to a process q if $I_{\text{src}} \in \mathcal{I}_{\text{M2L}}(I)$ and q is responsible for I. This is why clusters satisfying (iv) in Definition 6.7 are essential for process p.

The parent of a local cluster I_{c} is essential according to (v) in Definition 6.7. The reason is that the moments $\underline{\boldsymbol{\mu}}(\text{par}(I_{\text{c}}))$ are computed in parts by the M2M operations in the M-list task of I_{c}, and the local contributions $\underline{\boldsymbol{\lambda}}(\text{par}(I_{\text{c}}))$ are needed for the L2L operations in the L-list task of I_{c}. Communication is required if $\text{par}(I_{\text{c}})$ is assigned to a different process q. In this case, process p needs to send the parts of the moments of $\text{par}(I_{\text{c}})$ which it has computed in the M-list task of I_{c} to q, and process q needs to send the local contributions of $\text{par}(I_{\text{c}})$ to p for the L-list task of I_{c}. This explains why children of local clusters are essential as well; see Definition 6.7 (iv).

The locally essential time tree of a process p consists of all its essential clusters, some additional clusters which allow process p to identify all the FMM tasks that it needs to execute, and some auxiliary clusters to turn this set of clusters into a subtree of $\mathcal{T}_I$. More rigorously, we define the locally essential time tree as follows.

DEFINITION 6.7 (Locally essential time tree). *The locally essential time tree $\mathcal{T}_I^{\text{LE},p}$ of a process p is defined as the subtree of $\mathcal{T}_I$ that contains a cluster $I \in \mathcal{T}_I$ if it satisfies at least one of the following conditions:*

(a) I is essential.

(b) There exists a cluster $I_{\text{src}} \in \mathcal{T}_I$ such that $I_{\text{src}} \in \mathcal{I}_{\text{M2L}}(I)$ and I_{src} lies on the path between a local cluster I_{loc} and the root of $\mathcal{T}_I$.

(c) There exists a cluster $I_{\text{tar}} \in \mathcal{T}_I$ such that $I \in \mathcal{I}_{\text{M2L}}(I_{\text{tar}})$ and I_{tar} lies on the path between a local cluster I_{loc} and the root of $\mathcal{T}_I$.

(d) I lies on the path between the root of $\mathcal{T}_I$ and a cluster I_{inc} that satisfies at least one of the conditions (a)–(c).

The clusters satisfying (d) in Definition 6.7 are added to the locally essential time tree $\mathcal{T}_I^{\text{LE},p}$ of process p to obtain a subtree of $\mathcal{T}_I$. Organizing the essential clusters in a tree simplifies some preparatory steps in the FMM. For example, it is possible to determine the local FMM task lists for process p by a traversal of $\mathcal{T}_I^{\text{LE},p}$ similarly as described in Section 6.2. To be able to determine the M-list tasks and L-list tasks in general trees correctly, we add clusters satisfying (b) and (c) in Definition 6.7 to $\mathcal{T}_I^{\text{LE},p}$. An illustration of a locally essential time tree is given in Figure 6.2.

We assume that each process p has the following information about every time cluster I in its locally essential tree $\mathcal{T}_I^{\text{LE},p}$:

- the interval bounds of I,

- the process ID assigned to I,
- the locally essential part of the list $\mathcal{N}(I)$, i.e. all $I_{\text{src}} \in \mathcal{N}(I) \cap \mathcal{T}_I^{\text{LE},p}$,
- the locally essential part of the list $\mathcal{I}_{\text{M2L}}(I)$, i.e. all $I_{\text{src}} \in \mathcal{I}_{\text{M2L}}(I) \cap \mathcal{T}_I^{\text{LE},p}$.

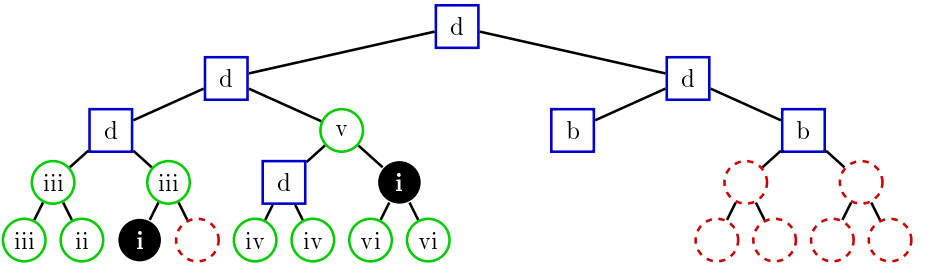


Figure 6.2: Illustration of a locally essential time tree $\mathcal{T}_I^{\text{LE},p}$ of a process p. In this example, process p is responsible for the clusters represented by the solid black circles labeled with i. The clusters represented by red dashed circles are part of the (non-uniform) tree $\mathcal{T}_I$ but are not contained in the locally essential tree $\mathcal{T}_I^{\text{LE},p}$. Green circles represent clusters which are in $\mathcal{T}_I^{\text{LE},p}$ because they are essential, i.e. they satisfy one of the conditions on the list in Definition 6.6. The label of such a cluster corresponds to a condition on that list that it satisfies. Clusters represented by blue squares are not essential, but satisfy one of the conditions (b), (c) or (d) in Definition 6.7 which is specified by the label again.

Locally required information about the space-time tree $\mathcal{T}_\Sigma$. Similarly as in the case of the time tree $\mathcal{T}_I$, each process needs only a local part of the space-time tree $\mathcal{T}_\Sigma$ for the distributed version of the task based FMM. It suffices that a process p knows all the space-time clusters which appear in FMM operations of local FMM tasks. Recall that local FMM tasks are FMM tasks of clusters I for which p is responsible. This motivates the following definition.

DEFINITION 6.8 (Locally essential space-time tree). *A space-time cluster $Z \in \mathcal{T}_\Sigma$ is essential (for the process p) if it satisfies at least one of the following conditions:*

(i) Z *is local, i.e. process p is responsible for Z.*

(ii) $Z \in \mathcal{N}(Z_{\text{tar}})$ *for a local cluster Z_{tar}.*

(iii) $Z \in \mathcal{I}_{\text{M2L}}(Z_{\text{tar}})$ *for a local cluster Z_{tar}.*

(iv) $Z = \text{par}(Z_{\text{c}})$ *for a local cluster Z_{c}.*

The locally essential space-time cluster tree $\mathcal{T}_\Sigma^{\text{LE},p}$ of process p is defined as the smallest subtree of $\mathcal{T}_\Sigma$ that contains the root $Z^{(0)}$ of $\mathcal{T}_\Sigma$ and all essential clusters.

When comparing the definitions of essential time clusters and essential space-time clusters in Definitions 6.6 and 6.8 we see that children of local time clusters are essential for process p, while children of local space-time clusters are not. Likewise, there is not any criterion equivalent to criterion (iv) from Definition 6.6 in Definition 6.8. The reason is that such clusters are not needed for local FMM tasks. Let us, for example, assume that I is a time cluster assigned to a process q, and $I_{\text{src}} \in \mathcal{I}_{\text{M2L}}(I)$ is assigned to a different process p. The M2L-list task for the cluster I is carried out by process q in this case, and it needs to receive the whole set of moments $\underline{\boldsymbol{\mu}}(I_{\text{src}})$ from process p to execute it. For the actual execution, process q needs to know the space-time clusters associated to I and I_{src}, i.e. $\mathcal{Z}_{\text{assoc}}(I)$ and $\mathcal{Z}_{\text{assoc}}(I_{\text{src}})$. Process p sends the whole set of moments $\underline{\boldsymbol{\mu}}(I_{\text{src}})$ corresponding to clusters in $\mathcal{Z}_{\text{assoc}}(I_{\text{src}})$ to process q once they are computed. For this purpose, it suffices that p knows that the time cluster I exists and is assigned to q. Process p does not need to know the actual space-time clusters $\mathcal{Z}_{\text{assoc}}(I)$, which is why they are not essential for p. Children of local space-time clusters are not essential for the same reason.

We assume that each process p has the geometrical information about every space-time cluster $Z = X \times I$ in its locally essential tree $\mathcal{T}_{\Sigma}^{\text{LE},p}$ and knows the process ID assigned to it. If Z is a local cluster in $\mathcal{T}_{\Sigma}^{\text{LE},p}$ also the nearfield $\mathcal{N}(Z)$ and interaction list $\mathcal{I}_{\text{M2L}}(Z)$ are assumed to be available.

In Section 6.4 we discuss how distributed meshes and locally essential time- and space-time trees can be constructed in practice. Before that, we continue with Section 6.3.2 in which we describe how to set up the FMM task lists in a distributed way and handle the communication between processes, before presenting the distributed, task based FMM algorithm in Section 6.3.3.

6.3.2 Distributed FMM task lists and inter-process communication

In the distributed version of the task based FMM each process needs to execute the FMM tasks corresponding to its local time clusters, i.e. those time clusters which are assigned to it. For some of these tasks, communication is necessary as we already mentioned in Section 6.3.1. In this section we describe how to construct the corresponding FMM task lists and how we handle the communication in practice using MPI.

The construction of the local FMM task lists. To organize its FMM tasks each process constructs a local version of the four FMM task lists introduced in Section 6.2. The M2L-list of a process p is filled with all local time clusters whose interaction lists are not empty. Similarly, the N-list of p is filled with all the local time clusters whose nearfield lists are not empty. To construct the M-list of p in a correct way, we traverse the locally essential time tree $\mathcal{T}_{I}^{\text{LE},p}$ and proceed in the same

way as described in Section 6.2. In a first step, we insert local and non-local clusters alike into the M-list. In a second step, all non-local clusters can be removed from this list to obtain the actual M-list of p. The same is possible for the L-list of p. Due to our definition of locally essential time trees, all local FMM tasks can be determined correctly in this way.

The inter-process communication in the FMM. The whole communication between processes in our parallel FMM is handled on the level of the time tree $\mathcal{T}_I$. All processes send and receive whole sets of moments $\underline{\boldsymbol{\mu}}(I)$ or local contributions $\underline{\boldsymbol{\lambda}}(I)$ of time clusters I whenever it is necessary. The situations in which communication is necessary have been discussed in the paragraphs following Definition 6.6. To summarize, there are two scenarios in which two processes p and q need to communicate: If they are responsible for two clusters that are connected by an edge in $\mathcal{T}_I$ — i.e. a cluster and one of its children — or two clusters where one is in the interaction list $\mathcal{I}_{\mathrm{M2L}}$ of the other. Hence, the communication pattern is simple and the number of overall communication events is small. In general, this is desirable as long as the workload of all processes is high enough so that they do not become idle while waiting for data.

Each process p can determine the data it needs to receive during the FMM procedure from its FMM task lists. To simplify the description we introduce three different lists for each process p to keep track of clusters for which data needs to be received and to identify the corresponding senders:

- The M-receive list: A pair $(I,\ q)$ is added to this list if I is a local cluster in the M-list of p, and a child I_{c} is assigned to process q.
- The M2L-receive list: A pair $(I_{\mathrm{src}},\ q)$ is added to this list if $I_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(I)$ for a local cluster I in the M2L-list of p, and q is responsible for I_{src}.
- The L-receive list: A pair $(I_{\mathrm{par}},\ q)$ is added to this list if I_{par} is the parent of a local cluster I in the L-list of p, and q is responsible for I_{par}.

Each process has to identify in addition the data it needs to send to other processes. To keep track of this information we create two sets of process IDs for each cluster I for which p is responsible:

- The M2L-recipient set of I: A process q is added to this set if there exists a cluster $I_{\mathrm{tar}} \in \mathcal{T}_I^{\mathrm{LE},p}$ such that $I \in \mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$ and q is responsible for I_{tar}.
- The L-recipient set of I: A process q is added to this set if it is responsible for a child I_{c} of I in $\mathcal{T}_I^{\mathrm{LE},p}$.

These lists can be filled by each process independently during a traversal of its locally essential time tree $\mathcal{T}_I^{\mathrm{LE},p}$. One could construct an additional M-recipient set of I and add a process q to this set if it is responsible for $\mathrm{par}(I)$. Since this information

is directly obtainable by checking the process ID of $\mathrm{par}(I)$, we do not create this additional list explicitly.

The actual communication during the FMM procedure can be handled asynchronously with non-blocking MPI send and receive operations. At the beginning of the FMM procedure each process p starts a non-blocking receive operation for each pair in the M-receive list, M2L-receive list, and L-receive list using the routine `MPI_Irecv`. For this purpose it has to allocate the memory for the expected incoming data, e.g., the moments in $\underline{\boldsymbol{\mu}}(I)$ for a pair $(I,\ q)$ in the M-receive list, and provide it to the routine `MPI_Irecv` together with some additional information like the ID of the sending process and a tag to identify the message. `MPI_Irecv` initiates the receive operation and returns immediately, so the thread that called it can proceed with other assignments. A request object provided by MPI is then used to check if data has actually been received. More details about `MPI_Irecv` and non-blocking communication in MPI can be found in [50, Section 3.7]. After starting all non-blocking receive operations, each process p can start working on its FMM tasks and check the MPI request objects regularly to update dependencies when data has actually been received.

Remark 6.9 (Processing received data). *When a process p has actually received a set of moments $\underline{\boldsymbol{\mu}}(I_{\mathrm{src}})$ for a pair $(I_{\mathrm{src}},\ q)$ in the M2L-receive list or a set of local contributions $\underline{\boldsymbol{\lambda}}(I_{\mathrm{p}})$ for a pair $(I_{\mathrm{p}},\ q)$ in the L-receive list, it has to update the dependencies of all its FMM tasks that require this data. Moments $\underline{\boldsymbol{\mu}}(I)$ received for a pair $(I,\ q)$ in the M-receive list, on the other hand, may not be complete yet. In fact, the received moments are those that process q computed by M2M operations between I and the children of I for which q is responsible, which may be only one of the two children. Hence, process p needs to add all parts of the moments $\underline{\boldsymbol{\mu}}(I)$ for a cluster I in the M-receive list and ensure that the moments are completely determined before it can update the dependencies of the related M-list task for I.*

The sending operations take place during the execution of the FMM tasks. A process p identifies the processes that need the data it has computed for a cluster I using the introduced M2L- or L-recipient sets of I or the process ID of $\mathrm{par}(I)$. Then it uses the non-blocking send operation `MPI_Isend` to initiate the sending process. In this routine the ID of the receiving process needs to be specified and, in addition, a suitable tag that allows the recipient to identify the message. Similarly as the routine `MPI_Irecv`, the routine `MPI_Isend` returns immediately. More details can be found again in [50, Section 3.7]. We specify the concrete sending operations in Section 6.3.3 when talking about the updated FMM task routines.

6.3.3 The distributed algorithm

In Algorithm 6.5 we present a sketch of the distributed, task based version of the FMM. Each process p handles its own tasks by using a local version of the task scheduler described in Section 6.2 and computes a local part $\boldsymbol{f}^{\mathrm{loc},p}$ of the global result vector $\boldsymbol{f}$. The local part $\boldsymbol{f}^{\mathrm{loc},p}$ includes all entries of $\boldsymbol{f}$ corresponding to local space-time elements, i.e. those associated to space-time leaf clusters in $\mathcal{T}_{\Sigma}^{\mathrm{LE},p}$ for which p is responsible. In particular, $\{\boldsymbol{f}^{\mathrm{loc},p}\}_{p=0}^{n_{\mathrm{proc}}}$ is a disjoint partition of the global vector $\boldsymbol{f}$ in our setting.

Algorithm 6.5 A distributed, task based version of the FMM in Algorithm 6.1.

Require: Let the assumptions (i)–(iv) at the beginning of Section 6.3 be satisfied.

// Each available process p executes the following steps:

1: Initialize $\boldsymbol{f}^{\mathrm{loc},p} = \boldsymbol{0}$.
2: **for** all local clusters $I \in \mathcal{T}_I^{\mathrm{LE},p}$
3: Initialize $\underline{\boldsymbol{\mu}}(I)$ and $\underline{\boldsymbol{\lambda}}(I)$ by zeros.
4: Fill the `M_list`, `M2L_list`, `L_list`, and `N_list` with local clusters in $\mathcal{T}_I^{\mathrm{LE},p}$.
5: `#pragma omp parallel`
6: `#pragma omp single`
7: StartMPIReceiveOperations( )
8: **while** the FMM task lists are not empty
9: CheckMPIForReceivedData( )
10: Execute the scheduling procedure in lines 8–26 of Algorithm 6.2.
// Use the routines of Algorithm 6.6 for the M-, L-, and M2L-list tasks.
11: **return** $\boldsymbol{f}^{\mathrm{loc},p}$ as part of the global result vector $\boldsymbol{f}$.

In the first steps of Algorithm 6.5 each process p initializes its local result vector and the sets of moments and local contributions of the time clusters assigned to it. Then it fills the FMM tasks lists with local clusters as described above and starts the scheduling procedure. This scheduling procedure is almost the same as the one in Algorithm 6.2. The only differences are the additional communication routines. The routine StartMPIReceiveOperations starts the non-blocking receive operations as we have discussed in the corresponding paragraph above. In each new iteration of the scheduling loop, the scheduling thread checks for received data with the routine CheckMPIForReceivedData and processes it according to Remark 6.9. In particular, it updates the dependencies of its FMM tasks. For the execution of these FMM tasks we use OpenMP `tasks` as in Algorithm 6.2, but the routines for the M-, L- and M2L-list tasks have to be adapted to incorporate the new sending routines.

In Algorithm 6.6 we present the updated M-, L- and M2L-list tasks. In the new M-list task in line 1, a process p sends the moments $\underline{\boldsymbol{\mu}}(I)$ of the considered cluster I to all processes in the M2L-recipient set of I after the completion of potential

S2M operations. Note that the recipient set might be empty in which case the moments $\underline{\boldsymbol{\mu}}(I)$ do not have to be sent to any other process. Since the non-blocking routine `MPI_Isend` is used for a potential send operation, the executing thread can directly continue with the following M2M operations in any case. Once these M2M operations are completed, process p needs to check whether it has to send its part of the moments $\underline{\boldsymbol{\mu}}(\mathrm{par}(I))$ of $\mathrm{par}(I)$ to another process q. This is the case, if q is responsible for $\mathrm{par}(I)$. If p is responsible only for I and not the other child I_{sib} of $\mathrm{par}(I)$ it can send the computed moments immediately to q, because they are "locally complete". Otherwise, p sends the moments to q in the M-list task of I or I_{sib} that is executed last. This is the meaning of the if clause in line 7. In this way, we avoid additional, unnecessary communication.

Algorithm 6.6 M-, L- and M2L-list tasks for distributed parallel execution.

```
   // All functions are assumed to be executed by a process p.
 1: function MListTask(I)
 2:     Execute potential S2M operations as in lines 2–5 of Algorithm 6.3.
 3:     for each process q in M2L_Recipient_Set(I)
 4:         Send the moments μ(I) to q using MPI_Isend.
 5:     Execute the M2M operations as in lines 6–9 of Algorithm 6.3.
 6:     if par(I) is assigned to a different process q.
 7:         if the computation of μ(par(I)) is locally complete
 8:             Send the moments μ(par(I)) to q using MPI_Isend.

 9: function LListTask(I)
10:     Execute the L2L operations as in lines 11–14 of Algorithm 6.3.
11:     if λ(I) is complete        // Only after potential M2L-list operations.
12:         if I is not a leaf in the locally essential time tree T_I^{EL,p}
13:             for each process q in L_Recipient_Set (I)
14:                 Send the local contributions λ(I) to q using MPI_Isend.
15:         if Z_assoc(I) contains leaves in T_Σ
16:             Execute the L2T operations as in lines 17–19 of Algorithm 6.3.

17: function M2LListTask(I)
18:     Execute the M2L operations as in lines 2–5 of Algorithm 6.4.
19:     if λ(I) is complete        // Only after potential L-list operations.
20:         Proceed as in lines 12–16 above.
```

The new L-list task is given in line 9 of Algorithm 6.6. The process p executes the L2L operations for the cluster I first. Like in the original version in Algorithm 6.3 it then checks if the local contributions are complete, i.e. if the M2L-list task of the cluster I is already completed. If this is the case, process p sends the local contributions $\underline{\boldsymbol{\lambda}}(I)$

to all processes in the L-recipient set of I, if I is not a leaf cluster. Note that this set is empty, if p is responsible for both children of I or if I does not have any children. Afterwards, process p can proceed with potential L2T operations like in the original L-list task. The same sending operation is added to the new M2L-list task for a cluster I in line 17. It is executed, if the M2L-list task for a cluster I is carried out after the L-list task for I.

The N-list tasks in the distributed, task based version of the FMM are the same as in Algorithm 6.4. Additional communication is not required since each process has all the information it needs to assemble and apply the nearfield blocks in all its N-list tasks. The assembly of the nearfield blocks should be handled in a preprocessing phase as described in Section 6.2. The same nested loop construct described in that section can be used for the assembly where each process needs to consider only clusters Z_{tar} assigned to it in the outer loop.

REMARK 6.10. *The additional implementation aspects which we discussed in Section 6.2.1 for the task based shared memory parallelization of the space-time FMM are also relevant for the distributed memory parallelization. Bounding the number of scheduled N-list tasks and prioritizing other FMM tasks is even more important here. In fact, a process p might need the data computed in an FMM task of another process q. If process q executes all N-list tasks first, process p might run out of tasks while waiting for the data of q, which would have a negative impact on the parallel performance. Hence, each process prioritizes FMM tasks that compute data needed by other processes in our implementation.*

6.4 A data and workload distribution strategy

Let $\Sigma = \Gamma \times (0,T)$ and $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ be a space-time tensor product mesh of Σ as in Section 2.4. We want to distribute the mesh Σ_h among a set of n_{proc} processes and create a corresponding space-time box cluster tree $\mathcal{T}_\Sigma$ and a related time tree $\mathcal{T}_I$ in a distributed way to enable the distributed memory parallelization of the FMM described in Section 6.3. Our data distribution relies on a subdivision of the time interval $(0,T)$ into a set of time slices which are groups of consecutive time steps in $\mathcal{I}_{h_t}$. We arrange these time slices in a coarse temporal cluster tree, assign process IDs to clusters in this tree and use this assignment of clusters to determine the data distribution. In this way, we can distribute the data and workload among all processes before the actual tree $\mathcal{T}_\Sigma$ and its temporal projection $\mathcal{T}_I$ are constructed.

Let a set of time slices related to the partition $\mathcal{I}_{h_t}$ of $(0,T)$ be given. We construct a binary tree for the time slices by a purely temporal version of Algorithm 5.1 in Section 5.2.1, where we subdivide the initial time cluster $(0,T)$ recursively until the resulting clusters contain only a single time slice. The resulting tree is denoted

by $\mathcal{T}_I^{\text{crs}}$. If the time slices are chosen suitably, the coarse tree $\mathcal{T}_I^{\text{crs}}$ is a subtree of the temporal projection $\mathcal{T}_I$ of the space-time box cluster tree $\mathcal{T}_\Sigma$ which we construct later. We assume that this is the case in the rest of this section.

For the data distribution we need to define interaction lists and nearfield lists for clusters in $\mathcal{T}_I^{\text{crs}}$. In Section 6.1 we defined such lists in a time tree $\mathcal{T}_I$ via projections of related lists in the corresponding space-time tree $\mathcal{T}_\Sigma$; see (6.3) and (6.4). Since we do not have a space-time tree corresponding to $\mathcal{T}_I^{\text{crs}}$ we determine the lists directly by using Algorithm 6.7. This is a temporal version of Algorithm 5.2, where we incorporate the ideas from Remark 5.11 to ensure that nearfield lists are non-empty only for leaf clusters in $\mathcal{T}_I^{\text{crs}}$ and contain only leaf clusters.

Algorithm 6.7 Recursive construction of the operation lists in $\mathcal{T}_I^{\text{crs}}$.

Require: Let $I^{(0)}$ be the root of $\mathcal{T}_I^{\text{crs}}$.
Let $\eta_2 \in \mathbb{R}_{>0}$ be the same constant for (5.12) as in Algorithm 5.2.

Call DetermineTemporalOperationLists($I^{(0)}$, $I^{(0)}$).

function DetermineTemporalOperationLists(I_{src}, I_{tar})
  if I_{src} is causally relevant for I_{tar}
    if I_{src} and I_{tar} satisfy the admissibility criterion (5.12)
      Add I_{src} to $\mathcal{I}_{\text{M2L}}(I_{\text{tar}})$.
    else
      if I_{tar} and I_{src} are not leaves in $\mathcal{T}_I$
        for all $(I_{\text{src,c}}, I_{\text{tar,c}})$ with $I_{\text{src,c}} \in \text{child}(I_{\text{src}})$, $I_{\text{tar,c}} \in \text{child}(I_{\text{tar}})$
          Call DetermineTemporalOperationLists($I_{\text{src,c}}$, $I_{\text{tar,c}}$).
      else *// Inadmissible pair*
        if I_{tar} and I_{src} are leaves in $\mathcal{T}_I$
          Add I_{src} to $\mathcal{N}(I_{\text{tar}})$.
        else if I_{tar} is a leaf in $\mathcal{T}_I$
          for all leaf descendants I_{d} of I_{src}
            Add I_{d} to $\mathcal{N}(I_{\text{tar}})$.
        else *// I_{src} is a leaf, while I_{tar} is not.*
          for all leaf descendants I_{d} of I_{tar}
            Add I_{src} to $\mathcal{N}(I_{\text{d}})$.

The data distribution is determined by an assignment of process IDs to clusters in $\mathcal{T}_I^{\text{crs}}$. We discuss a possible assignment strategy at the end of this section. For now we assume that such an assignment is given and use it to describe the data distribution and construction of distributed trees.

Mesh distribution. The space-time tensor product mesh Σ_h is decomposed into chunks corresponding to the leaves in $\mathcal{T}_I^{\text{crs}}$. Recall that these leaves are the chosen

time slices. A process is responsible for a leaf I in $\mathcal{T}_I^{\mathrm{crs}}$ if its ID is assigned to I. We provide process p with a chunk of Σ_h corresponding to a leaf I in $\mathcal{T}_I^{\mathrm{crs}}$ if

- p is responsible for I, or
- I is in the nearfield $\mathcal{N}(I_{\mathrm{tar}})$ of a leaf I_{tar} for which p is responsible.

Hence, each process p obtains a part of the mesh Σ_h related to a set of leaves $\{I_k^p\}_k$ in $\mathcal{T}_I^{\mathrm{crs}}$. We distinguish the *local mesh* of p and the *nearfield mesh* of p. The *local mesh* consists of chunks of Σ_h corresponding to local clusters in the set $\{I_k^p\}_k$. The remaining mesh obtained by p is related to non-local clusters in $\{I_k^p\}_k$ and is called the *nearfield mesh* of p. Note that the mesh Σ_h is the disjoint union of the local meshes of all n_{proc} processes.

Distributed tree construction. For the FMM we need to construct a space-time box cluster tree $\mathcal{T}_\Sigma$ and its temporal projection $\mathcal{T}_I$. In Section 6.3 we have seen that each process involved in the distributed execution of the FMM needs only a locally essential part of these trees. We start by constructing the locally essential space-time trees as introduced in Definition 6.8 in a collaborative manner. For this purpose, we provide each process with the coarse time tree $\mathcal{T}_I^{\mathrm{crs}}$ and the related process assignment information. From this tree a process p can deduce the temporal structure of the coarse initial part of $\mathcal{T}_\Sigma$ and determine its locally essential space-time clusters in this part. This allows us to use a distributed version of Algorithm 5.1 for the tree construction, where the processes collaborate to determine if a cluster Z is subdivided. Collaboration is necessary since the temporal part $\Pi_t[Z]$ of a cluster Z can contain several time slices whose meshes may be distributed among several processes. Each process counts the number of space-time elements in its local mesh which are assigned to Z. Then the total number $\#\hat{Z}$ of space-time elements in Z is determined by a global summation using communication. If $\hat{Z}$ is larger than the provided threshold $n_{\max}$, all processes can subdivide Z using, in particular, the temporal information in $\mathcal{T}_I^{\mathrm{crs}}$. Note that each process can determine if a cluster Z constructed during this process is locally essential for it and keep track of locally essential clusters only. The collaborative subdivision is stopped for a cluster Z if the temporal projection $\Pi_t[Z]$ is a leaf in the coarse tree $\mathcal{T}_I^{\mathrm{crs}}$. In this way, each process p determines a coarse version of its locally essential tree $\mathcal{T}_\Sigma^{\mathrm{LE},p}$. The remaining part of $\mathcal{T}_\Sigma^{\mathrm{LE},p}$ can then be constructed by each process p individually by further subdividing clusters whose elements are fully contained in the local or nearfield mesh of p. Clusters constructed during this local subdivision procedure are not assigned to any process a priori, since their temporal projection is not included in $\mathcal{T}_I^{\mathrm{crs}}$. Therefore, we assign the process ID of Z to all $Z_{\mathrm{c}} \in \mathrm{child}(Z)$ during this additional subdivision of clusters, and use this assignment to determine locally essential clusters.

REMARK 6.11. *To reduce the number of communication events in the collaborative construction of the space-time box cluster tree $\mathcal{T}_\Sigma$ one can use a level-wise subdivi-*

sion procedure. In this procedure, each process counts the number of local space-time elements for all its clusters at a level ℓ at once. All these numbers are communicated among all processes with a single global communication routine and summed up to determine the clusters at level ℓ which have to be subdivided. In this way, a single communication event per level in the tree suffices.

To obtain the locally essential time tree $\mathcal{T}_I^{\mathrm{LE},p}$ as introduced in Definition 6.7, each process p constructs the temporal projection $\widetilde{\mathcal{T}}_I^{\mathrm{LE},p}$ of its locally essential space-time tree $\mathcal{T}_\Sigma^{\mathrm{LE},p}$. The projection $\mathcal{T}_I$ of the global space-time tree $\mathcal{T}_\Sigma$ is obtained from these time trees $\{\widetilde{\mathcal{T}}_I^{\mathrm{LE},p}\}_p$ by a global communication between the processes. Since the coarse tree $\mathcal{T}_I^{\mathrm{crs}}$ is a subtree of $\mathcal{T}_I$, some but not all clusters in $\mathcal{T}_I$ have a process ID assigned to them. If a cluster I has a process ID but its children do not, we assign the ID of I to them. In this way, we get process assignments for all clusters in $\mathcal{T}_I$. Each process p uses these IDs to construct its locally essential tree $\mathcal{T}_I^{\mathrm{LE},p}$ from $\mathcal{T}_I$ by removing clusters which do not satisfy the properties in Definition 6.7.

REMARK 6.12. *If we follow the above strategy to distribute the space-time mesh Σ_h among all available processes and to construct the locally essential trees, each process has the necessary information for the distributed execution of the FMM in Algorithm 6.5 in general. However, this might not be the case if there exist early leaf clusters Z in the global space-time box cluster tree $\mathcal{T}_\Sigma$. We denote a leaf $Z \in \mathcal{T}_\Sigma$ as an early leaf if its temporal projection $\Pi_t[Z]$ is in the coarse time tree $\mathcal{T}_I^{\mathrm{crs}}$, but $\Pi_t[Z]$ is not a leaf in that tree. Some of the space-time elements of an early leaf Z might not be included in the local mesh or nearfield mesh of the process p which is responsible for $\Pi_t[Z]$ and Z. To overcome this problem we subdivide early leaves in the collaborative construction of $\mathcal{T}_\Sigma$ further — ignoring the fact that they contain less than $n_{\max}$ space-time elements — until the temporal components of the resulting clusters are leaves in $\mathcal{T}_I^{\mathrm{crs}}$.*

A process assignment strategy for clusters in $\mathcal{T}_I^{\mathrm{crs}}$. We still need to specify how to assign process IDs to the clusters in the coarse time tree $\mathcal{T}_I^{\mathrm{crs}}$. This process assignment does not only determine the data distribution discussed above but also the distribution of FMM tasks and the communication in the distributed parallel version of the FMM in Section 6.3. Hence, it is crucial to assign process IDs to clusters in $\mathcal{T}_I^{\mathrm{crs}}$ such that each process ends up with a similar workload and such that communication is avoided as much as possible. Here we propose a heuristic process assignment strategy for perfect binary trees $\mathcal{T}_I^{\mathrm{crs}}$, which are obtained for space-time meshes with uniform time steps in general.

To motivate our process assignment strategy we first investigate the computational effort of FMM tasks in a time tree $\mathcal{T}_I$ and recall the situations in which communication between processes is necessary. The M2L operations and nearfield operations are the

dominant operations in terms of the computational effort in the FMM in general. We will see this, for example, in the numerical experiments in Section 6.5. Hence, we want to ensure that the number of M2L and nearfield operations which every process has to execute due to our process assignment strategy is balanced. The following observations are relevant in this regard:

(i) The nearfield lists of time clusters on the same level of a time tree $\mathcal{T}_I$ have a similar size. This is illustrated in Figures 6.1a and 6.1b. An exception is the leftmost cluster on a level, i.e. the one whose lower interval bound is 0. The nearfield of this cluster is always smaller since there are not any other clusters in its history.

(ii) The sizes of the interaction lists of time clusters vary. With the typical choice of the parameter η_2 in the admissibility criterion (5.12), the right child of a cluster I has two clusters in its interaction list, while the left child of I has only one; see again Figures 6.1a and 6.1b. The two leftmost time clusters always have empty interaction lists.

(iii) Due to the refinement strategy in the construction of the space-time cluster tree $\mathcal{T}_\Sigma$, the number of space-time clusters in the set $\mathcal{Z}_{\text{assoc}}(I)$ of a time cluster $I \in \mathcal{T}_I$ increases with increasing level of I in general. Therefore, more M2L operations in $\mathcal{T}_\Sigma$ have to be executed for a cluster $I_1 \in \mathcal{T}_I$ than for a cluster $I_2 \in \mathcal{T}_I$ if the level $\ell(I_1)$ of I_1 in $\mathcal{T}_I$ is larger than the level $\ell(I_2)$ of I_2, at least in general.

Item (i) suggests that process IDs should be assigned to the leaves in a time tree $\mathcal{T}_I$ as uniformly as possible. In this way, we can balance the number of nearfield operations that each process needs to execute. Due to our mesh distribution strategy, also the memory that each process requires is balanced in this case. Due to item (iii) such a balanced process ID assignment is, in general, desirable for clusters at large levels where the M2L operations of time clusters are more costly. The varying sizes of interaction lists mentioned in item (ii) might still cause a slight imbalance which can be compensated by the process assignments for clusters at lower levels in $\mathcal{T}_I$.

To understand the effect of the process assignment strategy on the required communication in the FMM, we recall that there are two situations in which two processes need to communicate:

(a) If a cluster I and its parent $\text{par}(I)$ are assigned different process IDs, the two processes need to communicate for the M2M and L2L operations in the respective M- and L-list task.

(b) If two clusters I_{src} and I_{tar} with $I_{\text{src}} \in \mathcal{I}_{\text{M2L}}(I_{\text{tar}})$ are assigned different process IDs, the two processes need to communicate for the M2L operations in the M2L-list task of I_{tar}.

From item (a) we conclude that it is advantageous in terms of communication if we assign the same process IDs to a cluster I and its parent. If a cluster I_{src} is in the interaction list of a cluster I_{tar}, they lie in a close neighborhood of each other since their parents violate the admissibility criterion 5.12. Therefore, item (b) suggests that groups of neighboring clusters should be assigned the same process ID, if possible.

Based on the above considerations we came up with the following strategy for the assignment of n_{proc} process IDs in the set $\{0, \dots, n_{\text{proc}} - 1\}$ to clusters in a perfect binary tree $\mathcal{T}_I^{\text{crs}}$. For the sake of simplicity, we assume that $n_{\text{proc}} = 2^k$ for some $k \in \mathbb{N}$ in the description. The process IDs are assigned to the available clusters in a level-wise manner starting with the largest level $\text{depth}(\mathcal{T}_I^{\text{crs}})$. The actual assignment strategy depends on the level ℓ of the considered clusters:

- If $\ell \geq k = \log_2 n_{\text{proc}}$ there are $2^{\ell-k}$ clusters at level ℓ per process ID. We assign process IDs to groups of $2^{\ell-k}$ clusters in ascending order, i.e. the first $2^{\ell-k}$ clusters are assigned process ID 0, the next $2^{\ell-k}$ clusters process ID 1, and so on.
- If $\ell = k - 1$ there are twice as many process IDs as there are clusters at level ℓ. Each cluster I at this level is assigned the process ID of its left child.
- If $\ell < k - 1$ we first determine for each process ID p the number n_p^ℓ of clusters at levels $\widetilde{\ell} > \ell$ in $\mathcal{T}_I^{\text{crs}}$ that are assigned this ID. We group the process IDs into 2^ℓ sets of consecutive IDs, i.e. one for each cluster at level ℓ, and determine in each set a process ID p with a minimal number n_p^ℓ. The corresponding ID of the first set is assigned to the first cluster at level ℓ, the one of the second set to the second cluster, and so on.

The described process assignment strategy leads to a balanced distribution for clusters at large levels $\ell \geq k$ in the tree $\mathcal{T}_I^{\text{crs}}$ which is favorable in terms of workload balance as we observed above. By assigning neighboring clusters the same IDs and using an assignment in ascending order on each level $\ell \geq k$ we reduce the necessary communication. At level $\ell = k$ each process is responsible for a single cluster, but the workload for these clusters differs due to the differently sized interaction lists mentioned in item (ii) above. The assignment strategy at level $\ell = k - 1$ helps to compensate for the resulting workload imbalance since only processes with a lower workload at level k are responsible for a cluster at level $k - 1$. With the assignment strategy for levels $\ell < k - 1$ we try to further balance the workload between all the processes. The process assignment strategy is illustrated in Figure 6.3.

Note that the proposed strategy cannot be optimal in general. In fact, the process with ID 0 obtains the two leftmost clusters on all levels $\ell > \log_2 n_{\text{proc}}$, for which it does not have to execute any M2L operations. However, the resulting imbalance in the workload is not critical as long as there are enough other clusters assigned to process 0.

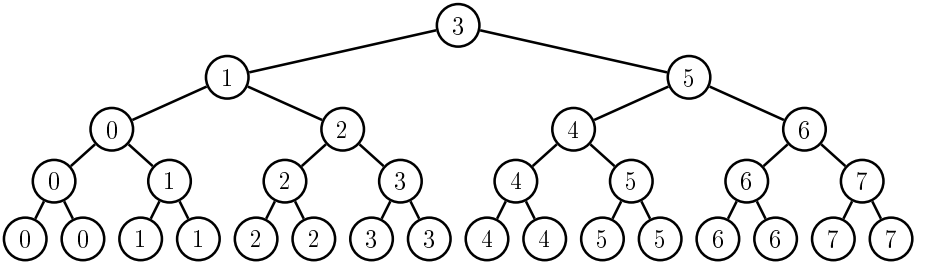


Figure 6.3: Illustration of the proposed process assignment strategy for a coarse time tree $\mathcal{T}_I^{\text{crs}}$ with depth 4 and $n_{\text{proc}} = 8$ processes. Each cluster is represented by a circle. The label of a cluster corresponds to the process ID assigned to it.

REMARK 6.13 (Process assignment strategy for general values of n_{proc}). *The proposed process assignment strategy can be generalized for values of n_{proc} which are not powers of two. In this case, we use the value $k = \lceil \log_2(n_{\text{proc}}) \rceil$ to distinguish the strategy for each level. For levels $\ell \geq k$ we split the clusters at level ℓ into n_{proc} groups, whose sizes differ between m_ℓ and $m_\ell + 1$, where m_ℓ is defined by the Euclidean division $2^\ell = m_\ell \, n_{\text{proc}} + r_\ell$ with $r_\ell \in \{0, \ldots, n_{\text{proc}-1}\}$. Process IDs are assigned to groups of clusters as before. The different group sizes introduce a slight workload imbalance at level ℓ. We can compensate for this by the process assignment at level $\ell - 1$. For clusters at levels ℓ with $\ell < k - 1$ we use the same process assignment strategies as described above.*

6.5 Numerical experiments

The parallel, task based version of the space-time FMM for shared and distributed memory systems described in Sections 6.2 and 6.3 has been implemented in the publicly available C++ library besthea [49]. In this section we present numerical experiments that demonstrate the efficiency of the presented algorithms and our implementation, in which we use SIMD vectorization to make the most of modern CPUs with vector arithmetic units. The results for the distributed memory parallelization here correspond to those published in [76] and were executed on the Salomon cluster at IT4Innovations National Supercomputing Center in Ostrava, Czech Republic. The results for the shared memory parallelization in [76] are replaced by new ones to ensure a more consistent choice of FMM parameters throughout the thesis. Since the operation of Salomon was discontinued by the end of 2021, we carried out the new experiments on the VSC-4 cluster in Vienna, Austria. More details about the

hardware specifications of these supercomputers and the compilers used on each of them are given in Appendix A.

6.5.1 Numerical experiments in shared memory

To investigate the performance of the shared memory parallelization of the space-time FMM described in Section 6.2, we consider the direct boundary integral approach in Section 3.2 to solve an initial Dirichlet boundary value problem (1.1)–(1.3) for the heat equation which is specified as follows: We choose the time interval $(0, 0.25)$, the spatial domain $\Omega = (-0.5, 0.5)^3$ and the heat capacity constant $\alpha = 1$, as well as the initial datum $u_0 = 0$ and the Dirichlet datum

$$u(\boldsymbol{x}, t) = G_1(\boldsymbol{x} - \boldsymbol{y}^*, t) \qquad \text{for } (\boldsymbol{x}, t) \in \Sigma,$$

where $\boldsymbol{y}^* = (1.5, 1.5, 1.5)^\top$ and G_1 is the fundamental solution of the heat equation in (3.4). To determine an approximation of the unknown Neumann datum $\gamma_{1,\Sigma}^{\text{int}} u$ we solve the linear system (3.24) on a space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$, where Γ_h contains 3072 congruent triangles on the surface of the cube $(-0.5, 0.5)^3$ and $\mathcal{I}_{h_t}$ consists of 128 uniform time steps. The system (3.24) is solved by using the GMRES method without a preconditioner and with a desired relative accuracy of 10^{-8}. The BEM matrices V_h and K_h are applied using the task based version of the space-time FMM in Section 6.2.

For the space-time FMM we construct a space-time box cluster tree $\mathcal{T}_\Sigma$ using Algorithm 5.1 with the cluster parameters $n_{\max} = 800$ and $c_{\text{st}} = 4.1$, and determine the operation lists of clusters in $\mathcal{T}_\Sigma$ by using Algorithm 5.2 with the spatial truncation parameter $n_{\text{tr}} = 2$ and the constant $\eta_2 = 1$ for admissibility criterion (5.12). The expansion degrees $m_t = 4$ and $m_{\boldsymbol{x}} = 12$ are used for the FMM operations in the space-time tree $\mathcal{T}_\Sigma$. All these parameters are chosen such that the FMM works efficiently and does not affect the approximation quality of the BEM in a negative way.

The linear system (3.24) was solved on a single node of the VSC-4 cluster with a varying number of threads. To ensure that each thread is assigned to a separate physical core of the two 24-core CPUs of the computing node, and to control the thread affinity we set the OpenMP environment variables `OMP_PLACES=cores` and `OMP_PROC_BIND=close`. More details about the effects of these environment variables are given in [58, Sections 2.6.2, 6.4 and 6.5].

Independent of the number of used threads, 39 iterations of the GMRES method are required to compute the approximate solution $\boldsymbol{q}$ to the system (3.24) for the considered mesh and data. The error between the corresponding discrete function $q_h \in S^{0\otimes 0}_{h_{\boldsymbol{x}}, h_t}$ and the exact Neumann datum $q(\boldsymbol{x}, t) = \gamma_{1,\Sigma}^{\text{int}} G_1(\boldsymbol{x} - \boldsymbol{y}^*, t)$ in the

relative L^2 norm is $\|q - q_h\|_{L^2(\Sigma)}/\|q\|_{L^2(\Sigma)} \approx 0.064$, which is close to the L^2 best approximation error 0.054. This indicates that our implementation is correct and that the parameters for the FMM are chosen properly.

Table 6.1 shows the measured computation times for varying numbers of threads. We list the assembly times for the matrices V_h and K_h, as well as the application times for K_h and the iteration times required for a single GMRES iteration. Note that the time it takes to apply V_h is by far the most time-consuming part of a GMRES iteration in our setting, so we can interpret the GMRES iteration times simply as application times of V_h. The application times of the matrix K_h were determined in a separate routine where K_h was applied several times in a row to ensure that initialization times of libraries like the MKL, which we use for matrix-vector multiplications of inadmissible blocks, do not spoil the measurements.

	No. threads	1	2	4	8	16	24	48
Assemble K_h	time $[s]$	3063	1587	787.6	393.1	195.9	131.5	74.2
	efficiency [%]	100.0	96.5	97.2	97.4	97.7	97.1	86.0
Apply K_h	time $[s]$	173.8	87.5	43.5	21.7	10.9	7.6	3.8
	efficiency [%]	100.0	99.3	99.9	100.1	99.7	95.3	95.3
Assemble V_h	time $[s]$	3667	1833	915.9	459.1	229.7	153.2	76.9
	efficiency [%]	100.0	100.1	100.1	99.8	99.8	99.7	99.3
GMRES it.	time $[s]$	174.8	88.1	44.3	22.0	11.1	7.48	3.87
	efficiency [%]	100.0	99.2	98.6	99.3	98.4	97.4	94.1

Table 6.1: Results of a strong scalability test on a single node of VSC-4 for a BEM problem with 393 216 space-time surface elements (128 time steps in the interval $(0, 0.25)$, 3 072 spatial elements on the surface of the cube $(-0.5, 0.5)^3$). The assembly and application times for the matrix K_h, the assembly times for the matrix V_h, and the times per GMRES iteration are presented for different numbers of OpenMP threads. The GMRES iteration time is dominated by the application time of the matrix V_h.

The assembly times of V_h and K_h indicate how long it takes to assemble the inadmissible blocks of these matrices. Recall that the assembly of these blocks and the related parallelization were shortly discussed in Section 6.2. In our implementation we use SIMD vectorization for the numerical evaluation of the occurring integrals. The details are given in [81, Section 5.2]. In Table 6.1 we see that the efficiency of the parallel matrix assembly is close to optimal for the single layer operator matrix V_h for the full range of threads. The same is true for the double layer operator matrix K_h when using up to 24 threads on a single socket of the node. When using all 48 threads

the efficiency slightly drops. This might be related to the fact that the memory access in the construction of K_h is more complicated than in the construction of V_h and that this memory access is less efficient when using both sockets of the computing node instead of one. To store the inadmissible blocks of the matrix K_h, 11.74 GiB of memory are required. Since K_h is applied only once, the memory can be freed afterwards. The inadmissible blocks of the matrix V_h require 16.28 GiB of memory. In comparison, all moments and local contributions needed for the application of the FMM require only 0.13 GiB of memory.

The application times of K_h and the times per GMRES iteration in Table 6.1 are very similar. This is not too surprising, since the same number of operations are executed in the space-time FMM for V_h and K_h. Only the nearfield operations and the kind of S2M operations differ for the matrices V_h and K_h as we mentioned in Remark 5.14 in Section 5.2.4. The S2M operations are slightly more expensive for K_h since normal derivatives of tensor product Chebyshev polynomials have to be evaluated, while the nearfield operations are slightly cheaper for K_h since the number of columns of an inadmissible block $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}} \times \check{Z}_{\mathrm{src}}}$ is smaller than the number of columns of an inadmissible block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ — the former depends on the number of spatial nodes in X_{tar} and the latter on the number of spatial elements in this box. When comparing the application and iteration times for different numbers of threads we observe an almost optimal parallel efficiency which never drops below 94%. This shows that our parallelization strategy works well for the considered example.

Remark 6.14. *In our implementation we do not check whether the time steps of a space-time tensor product mesh are uniform in time. In particular, we do not exploit the lower triangular block Toeplitz structure of the BEM matrices in such situations. By making use of this structure, the storage requirements and assembly times of the matrices V_h and K_h could be reduced significantly, while the application times would stay the same. Instead of focusing on this kind of optimization we chose to consider the space-time FMM for more general meshes as a step towards a space-time adaptive method.*

To get a better understanding of the way in which operations are executed in the task based FMM, we included the possibility to measure the execution times of individual FMM tasks and groups of FMM operations that each thread executes in our implementation. In Figure 6.4 we present the measured times for the application of V_h in the previous example when using 48 threads for the computation. As explained in the caption of the figure, each rectangle stands for a single iteration of a `taskloop` in one of the FMM tasks in Algorithms 6.3 and 6.4. For example, a red rectangle drawn in the line of a thread means that this thread executed the M2L operations for a single cluster Z_{tar} associated with a time cluster I_{tar} and all $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ in line 3 of Algorithm 6.4.

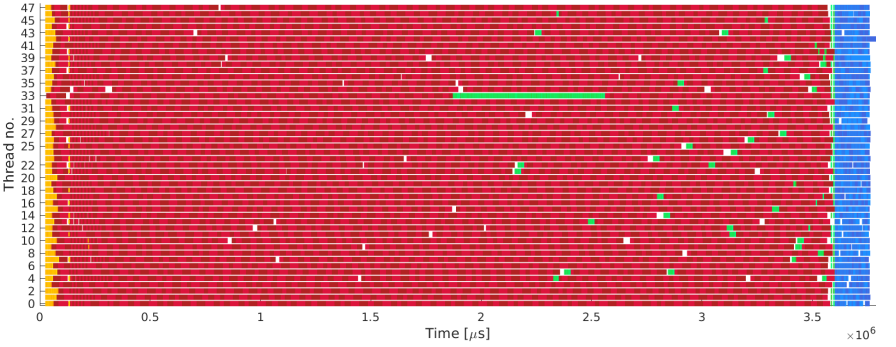


Figure 6.4: Illustration of the detailed execution times for the task based FMM for the matrix V_h while running the example from Table 6.1 on 48 threads. For each thread a line of colored rectangles is depicted. Each colored rectangle stands for a single iteration of a `taskloop` in one of the FMM tasks executed by the respective thread; see Algorithms 6.3 and 6.4. The S2M and M2M operations are depicted in shades of yellow and orange, the M2L operations in shades of red, the L2L and L2T operations in shades of green, and the nearfield operations in shades of blue. The thread responsible for scheduling the FMM tasks is thread 33 in this example.

Figure 6.4 is another indicator of the good performance of our task based parallelization. It can be clearly seen that the operations of the different FMM phases are executed in parallel. For example, some threads execute M2L operations (red) of corresponding M2L-list tasks while others are still busy with S2M and M2M operations (yellow/ orange) of M-list tasks during the starting period of the application. Also L2L operations (green) of corresponding L-list tasks are executed concurrently with M2L-list tasks. Nearfield operations (blue) of N-list tasks are executed only at the end. The reason is that we bound the number of N-list tasks that can be executed while other FMM tasks are available in our implementation as described in Section 6.2.1. For the examples in this thesis, this bound is set to zero. We choose this strict bound to ensure that other FMM tasks are always preferred, which is relevant, in particular, for the distributed parallelization as motivated in Remark 6.10. As long as enough other FMM-list tasks are available this choice is fine and might only lead to some negligible idle times before the execution of the N-list tasks, which can also be seen in Figure 6.4 at second glance. A more careful choice of the bound of N-list tasks might, however, be beneficial for examples with few tasks.

In general, one can see that the parallel execution of `tasks` is almost, but not completely perfect in our implementation. Throughout the application we can see spontaneous gaps in some of the lines in Figure 6.4, where threads are idle. Overall, these idle times make up only a negligible portion of the total runtime here. Nonetheless,

let us explain why these idle times exist in the first place. If a thread executes an FMM task like, e.g., an M2L-list task created in line 20 of Algorithm 6.2, it generates new `tasks` with the `taskloop` construct in line 2 of Algorithm 6.4. The thread can only continue with the remaining assignments of the M2L-list task after all created `tasks` have been completed. This ensures the correctness of our method but causes the observed idle times.

If the thread that is responsible for scheduling new FMM tasks has to wait for the completion of a `taskloop`, other threads might run out of `tasks` as well. Such a situation can be seen in Figure 6.5, which shows the starting period of the application in more detail. Here we see that the scheduling thread 33 becomes idle shortly before time $t = 12 \cdot 10^4\ \mu s$. From the measured execution times of FMM tasks, we can tell that this thread is waiting for the completion of an M2L-list task here until time $t \approx 13 \cdot 10^4\ \mu s$. At the same time, we see that many other threads become idle because there are not any `tasks` available for execution. At the time $t \approx 13 \cdot 10^4\ \mu s$, the thread 33 schedules new FMM tasks and the other threads start to work again. Thread 33 joins them in the execution of `tasks` again shortly before time $t = 14 \cdot 10^4\ \mu s$. The possibility that events like this occur, where multiple threads become idle at the same time, is the price we have to pay for including the scheduling thread in the execution of `tasks` by using the `taskyield` pragma as described in Section 6.2.1. One could exclude the scheduling thread from the execution of the large FMM tasks while allowing it to execute smaller `tasks` to mitigate this problem. However, we are not aware of any feature of OpenMP that allows one to hinder a certain thread from executing a specific task. Otherwise, one could also refrain from using `taskyield` and thereby let the scheduling thread focus on the scheduling of new FMM tasks until the very end of the application. If enough threads are available this might work well, but it might be less efficient for small numbers of threads.

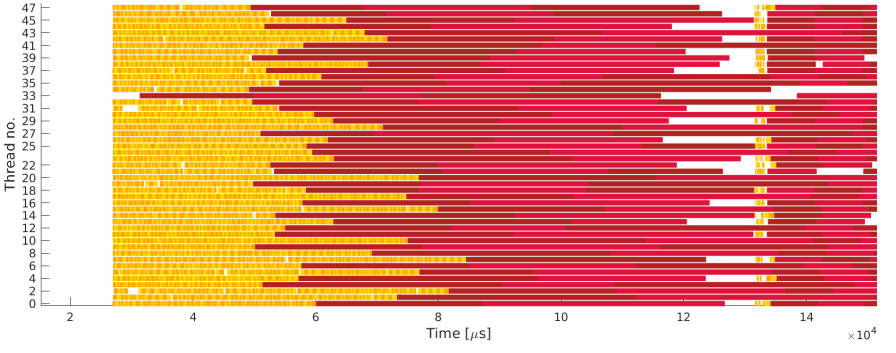


Figure 6.5: Enlarged view of the initial period of the execution times in Figure 6.4.

Despite the few small shortcomings discussed above, we have seen that our parallel, task based FMM works efficiently in practice for problems in shared memory. In the following section we investigate if this is also true for the distributed memory parallelization described in Section 6.3.

6.5.2 Numerical experiments in distributed memory

The results presented in this section correspond to those published in [76, Section 5.2]. To test the scalability of our hybrid MPI-OpenMP implementation of the distributed task based FMM in Section 6.3 we used the Salomon cluster, whose hardware specifications are given in Appendix A, to solve the linear system (3.24) related to the same initial Dirichlet boundary value problem as in Section 6.5.1, but on a finer space-time tensor product mesh Σ_h consisting of 12 288 spatial triangles and 1 024 time steps, i.e. a total of 12 582 912 space-time boundary elements.

For the distributed parallelization, the space-time mesh Σ_h needs to be distributed among the available processes. For this purpose we subdivide the mesh into 256 uniform time slices — each containing 4 time steps — and proceed as described in Section 6.4: We build a temporal tree $\mathcal{T}_I^{\text{crs}}$ for the time slices, distribute the clusters in this tree among the available processes with the assignment strategy described at the end of that section, and use the process assignments for the data distribution. For the collaborative construction of the space-time box cluster tree $\mathcal{T}_\Sigma$, or rather the corresponding locally essential parts, we use the same parameters $n_{\max} = 800$ and $c_{\text{st}} = 4.1$ as in Section 6.5.1. The remaining parameters are also chosen as in that section, i.e. $n_{\text{tr}} = 2$ and $\eta_2 = 1$ for the construction of the operation lists in the space-time tree and the expansion degrees $m_t = 4$ and $m_{\boldsymbol{x}} = 12$ for the FMM operations.

When running our code on the Salomon cluster we assigned a single MPI process to each cluster node and use all 24 available cores per node for the parallel execution of `tasks` with OpenMP. The affinity of threads on each node was set using the environment variables `KMP_AFFINITY=granularity=core,compact` and `KMP_HW_SUBSET=2s,12c`. For the computations we used between 16 and 256 nodes. The large memory demand of the considered problem did not permit us to run the problem on fewer nodes. In fact, a total of 666.2 GiB of memory is needed to store all inadmissible blocks of K_h and 890.8 GiB for the inadmissible blocks of V_h. In comparison 2.92 GiB of memory are needed to store the moments and local contributions of all clusters for the application of the FMM in this example.

In total, 60 iterations of the non-preconditioned GMRES method are required to compute the solution $\boldsymbol{q}$ in this example. The relative L^2 error between the corresponding function $q_h \in S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}$ and the exact Neumann datum q is equal to 0.031,

which is close to the relative L^2 best approximation error 0.027 of q in $S^{0\otimes 0}_{h_x,h_t}$. In Table 6.2 we present the corresponding computation times for the assembly of the system matrices V_h and K_h and the times required for a single GMRES iteration when using between 16 and 256 nodes of the Salomon cluster. To compute the efficiencies, we take the times measured when using 16 nodes as reference values. We observe that the parallel efficiency of the assembly of the system matrices V_h and K_h is almost optimal in this example. The iteration times scale also well when using up to 128 nodes. However, a clear drop in efficiency is observed for 256 nodes. This is related to the size of the considered problem. The mesh is decomposed into 256 time slices which form the leaves of the temporal tree $\mathcal{T}_I^{\text{crs}}$. Recalling the process assignment strategy at the end of Section 6.4, we see that each of the 256 processes/nodes is responsible for only a single leaf in $\mathcal{T}_I^{\text{crs}}$ and some additional clusters in the remaining tree. Hence, the number of FMM tasks that each process needs to execute is small in this situation which increases the probability of threads becoming idle. This reveals the limits of our parallelization strategy. It is remarkable that we are still able to achieve an efficiency of more than 60% when using 256 nodes under these circumstances.

	No. nodes	16	32	64	128	256
Assemble V_h	time $[s]$	769.8	385.5	194.4	97.1	50.0
	efficiency [%]	100.0	99.8	99.0	99.1	96.2
Assemble K_h	time $[s]$	502.4	252.5	128.6	63.0	31.9
	efficiency [%]	100.0	99.5	97.7	99.7	98.4
GMRES it.	time $[s]$	14.7	7.3	3.7	2.1	1.5
	efficiency [%]	100.0	101.5	99.4	89.4	62.6

Table 6.2: Results of a strong scalability test on up to 256 nodes of the Salomon cluster for a BEM problem with 12 582 912 space-time surface elements (1 024 time steps in the interval $(0, 0.25)$, 12 288 spatial elements on the surface of the cube $(-0.5, 0.5)^3$).

REMARK 6.15. *The application times of K_h are missing in Table 6.2. In fact, they are also missing in [76, Section 5] where we decided to focus only on the GMRES iteration times, since the time of a single application of K_h is negligible compared to the accumulated times of multiple applications of V_h in the GMRES. In particular, we did not run separate tests to measure the application times of K_h for that publication which would be necessary to determine the effective application times without any artifacts. Therefore, we refrain from presenting the application times of K_h determined in the original experiments. Note that they should not differ too much from the application times of V_h in general as we observed in the shared memory experiments in Section 6.5.1.*

We conclude this section by demonstrating the performance of our distributed memory parallelization of the space-time FMM for the heat equation for a more realistic example. For this purpose, we solve the same linear system (3.24) as in the last two examples related to an initial Dirichlet boundary value problem with the same initial and boundary data as before. This time, however, we consider a more complicated geometry Ω, namely a crankshaft, whose surface is discretized by a mesh Γ_h consisting of 42 888 plane spatial triangles; see Figure 6.6. Combining this mesh with 1 024 uniform time steps in the time interval $(0, 0.25)$ yields a space-time tensor product mesh Σ_h consisting of 43 917 312 space-time elements which we consider for the computations. We subdivide this mesh into 128 uniform time slices for the mesh distribution and use the parameters $n_{\max} = 800$, $c_{\mathrm{st}} = 4.5$, $n_{\mathrm{tr}} = 2$, $\eta_2 = 1$, $m_t = 3$ and $m_{\boldsymbol{x}} = 12$ for the FMM. 7 910 GiB of memory are required to store the inadmissible parts of the corresponding matrix K_h and 12 369 GiB for the matrix V_h. This high memory demand could be drastically reduced by exploiting the uniformity of the time steps, as we mentioned in Remark 6.14. Since we do not consider this in our implementation, we used 128 nodes of the Salomon cluster to set up and solve the system in (3.24). The corresponding computation times are summarized in Table 6.3. Since we do not use a preconditioner for the GMRES method in this experiment, 399 iterations are needed to achieve a relative GMRES accuracy of 10^{-6}. In total, we are able to solve the problem in less than two hours. The solution at the end of the time interval is depicted in Figure 6.6.

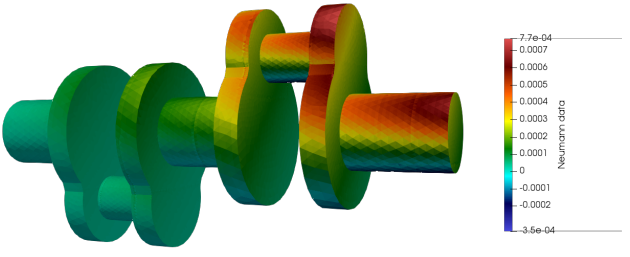


Figure 6.6: Visualization of the approximated Neumann datum on the surface of the crankshaft at time $t = 0.25$. The surface is discretized by 42 888 plane triangles.

No. utilized nodes	assembly of V_h	assembly of K_h	solution using GMRES
128	1161.58 s	1223.97 s	2927.70 s

Table 6.3: Computation times for setting up and solving the system in (3.24) for a space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ with 43 917 312 space-time surface elements (1024 time steps in the interval $(0, 0.25)$, 42 888 spatial elements), where Γ_h is a triangular mesh that describes the surface of a crankshaft.

7 A time-adaptive version of the space-time FMM

The space-time FMM described in Chapter 5 and the related pFMM in [52, 67, 68] were originally developed for space-time tensor product meshes with uniform time steps. Uniform time steps lead to BEM system matrices with a block Toeplitz structure which can be exploited to avoid recomputing several matrix entries. However, if the solution of a boundary value problem requires a small time step size to resolve some local aspects, a huge number of uniform time steps of that size might be needed to cover the whole time interval. Instead, the time step sizes can be chosen adaptively to reduce the total number of required time steps significantly and thereby the total number of degrees of freedom of the considered linear systems. The smaller system matrices can make up for the lost Toeplitz structure when considering temporally adaptive meshes. However, also the efficiency of the space-time FMM from Chapter 5 might suffer for such meshes. Therefore, we present a time-adaptive version of the FMM in this chapter.

In the context of the Laplace equation adaptive versions of the FMM have been developed, for example, in [21, 22, 54]. A key idea of these methods is to introduce additional FMM operations between boxes of different sizes for which the standard FMM operations cannot be used because of a violated admissibility criterion. These operations are based on one-sided kernel expansions and are sometimes denoted as S2L (source to local, also called Q2L) and M2T (moment to target, also called M2P) operations. In this chapter, we introduce similar new FMM operations based on temporally one-sided expansions of the heat kernel to extend the existing space-time FMM and obtain a fast method that is better suited for the treatment of temporally adaptive meshes. The corresponding results have been accepted for publication in [78].

The rest of the chapter is structured as follows. In Section 7.1 we introduce two temporally one-sided approximations of the heat kernel and analyze in which situations they are better suited for the kernel approximation than the temporally two-sided approximation in Section 5.1. In Section 7.2 we describe the new time-adaptive FMM in detail. Here we focus on the single layer operator matrix V_h, but the matrices K_h, $\mathsf{K}_h^{\top^*}$ and D_h can be treated similarly. To enable the time-adaptive FMM we modify the construction of the space-time cluster trees in Section 7.2.1, introduce new operation lists in Section 7.2.2, and new FMM operations based on the temporally

one-sided kernel expansions in Section 7.2.3. The new time-adaptive FMM algorithm for the application of V_h is presented in Section 7.2.4. In Section 7.3 we comment on the runtime complexity and the storage requirements of the new FMM operations and in Section 7.4 we discuss how the parallelization strategy from Chapter 6 can be applied to the time-adaptive FMM. Finally, we present numerical experiments for meshes with non-uniform time steps in Section 7.5 to demonstrate the benefits of the new time-adaptive FMM.

7.1 Temporally one-sided approximations of the heat kernel

The space-time FMM in Chapter 5 is based on the separable expansion (5.9) of the heat kernel (3.4) that is discussed in Section 5.1. For this expansion we have considered the heat kernel $((\boldsymbol{x},t),(\boldsymbol{y},\tau)) \mapsto G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)$ for $(\boldsymbol{x},t)$ and $(\boldsymbol{y},\tau)$ in two suitable axis-parallel, 4D boxes $Z_{\text{tar}} = X_1 \times I_1$ and $Z_{\text{src}} = X_2 \times I_2$, interpolated it in the two time intervals I_1 and I_2, and approximated it with a truncated Chebyshev expansion in X_1 and X_2. The corresponding approximation error is estimated in Theorem 5.9. In Remark 5.4 we have pointed out that the temporal convergence rate q_2 in the approximation error estimate deteriorates if the ratio $\max\{|I_1|,|I_2|\}/\operatorname{dist}(I_1,I_2)$ becomes large. This is not an issue for the space-time FMM in Chapter 5 since the stated ratio is bounded from above by a constant η_2 through the admissibility criterion (5.12) whenever we approximate the heat kernel in this way for two clusters Z_{tar} and Z_{src}. However, this admissibility criterion limits the cases in which we can use the expansion (5.9). Thus, the corresponding standard space-time FMM might become inefficient for meshes that are adaptive in time. Therefore, we introduce two new, temporally one-sided approximations of the heat kernel in this section that are accurate as long as $\min\{|I_1|,|I_2|\}/\operatorname{dist}(I_1,I_2)$ is bounded. These approximations allow us to identify new kinds of admissible blocks of BEM matrices like V_h and introduce new FMM operations that improve the performance of the FMM significantly for space-time tensor product meshes which are adaptive in time.

In this section we use the same notation as in Section 5.1. We consider two axis-parallel, 4D boxes $Z_{\text{tar}} = X_1 \times I_1$ and $Z_{\text{src}} = X_2 \times I_2$ where X_1 and X_2 are cubes with the same edge length $2\widetilde{h}_{\boldsymbol{x}}$, and $I_1 = [a_1,b_1]$ and $I_2 = [a_2,b_2]$ are two time intervals such that $a_1 > b_2$. The only new operators that we introduce are the one-sided interpolation operators

$$\begin{aligned}
&\mathfrak{I}^{(m)}_{\text{dir},I_1} : C(I_1 \times I_2) \to \mathcal{P}_m(I_1) \otimes C(I_2), \qquad && \mathfrak{I}^{(m)}_{\text{dir},I_1} = \mathfrak{I}^{(m)}_{I_1} \otimes \mathrm{Id},\\
&\mathfrak{I}^{(m)}_{\text{dir},I_2} : C(I_1 \times I_2) \to C(I_1) \otimes \mathcal{P}_m(I_2), \qquad && \mathfrak{I}^{(m)}_{\text{dir},I_2} = \mathrm{Id} \otimes \mathfrak{I}^{(m)}_{I_2},
\end{aligned}$$

with the one-dimensional interpolation operators $\mathfrak{I}^{(m)}_{I_1}$ and $\mathfrak{I}^{(m)}_{I_2}$ defined as in (5.1). We use these operators for the following new approximations of the heat kernel.

One-sided interpolation in the source interval I_2. By interpolating the heat kernel in the temporal variable $\tau \in I_2$ and approximating it with a truncated Chebyshev expansion (5.7) in both spatial variables $\boldsymbol{x} \in X_1$ and $\boldsymbol{y} \in X_2$ we get the expansion

$$\mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1 \times X_2} \mathfrak{I}^{(m_t)}_{\mathrm{dir},I_2}[G_\alpha](\boldsymbol{x},t,\boldsymbol{y},\tau) = \sum_{a=0}^{m_t} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu} \in \mathbb{N}_0^3: \\ |\boldsymbol{\kappa}+\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}}} E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t)\, T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x})\, T_{X_2,\boldsymbol{\kappa}}(\boldsymbol{y})\, L^{(m_t)}_{I_2,a}(\tau) \quad (7.1)$$

for $(\boldsymbol{x},t) \in Z_{\mathrm{tar}}$ and $(\boldsymbol{y},\tau) \in Z_{\mathrm{src}}$, where $\{L^{(m_t)}_{I_2,a}\}_a$ are Lagrange polynomials as defined in (5.2), and $\{T_{X_1,\boldsymbol{\nu}}\}_{\boldsymbol{\nu}}$ and $\{T_{X_2,\boldsymbol{\kappa}}\}_{\boldsymbol{\kappa}}$ are tensor product Chebyshev polynomials as in (5.6). The expansion coefficients $E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t)$ depend explicitly on $t \in I_1$ and are given by

$$E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t) = \frac{1}{(4\pi\alpha(t - \xi^{(m_t)}_{I_2,a}))^{3/2}} \prod_{j=1}^{3} E_{\kappa_j,\nu_j}(r_j, 4\alpha(t - \xi^{(m_t)}_{I_2,a})/\widetilde{h}_x^2), \quad (7.2)$$

where r_j and E_{κ_j,ν_j} are the same quantities as in (5.10) and $\{\xi^{(m_t)}_{I_2,a}\}_{a=0}^{m_t}$ are the transformed Chebyshev nodes of order $m_t{+}1$ on I_2. The superscript of $E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}$ should indicate that the corresponding approximation involves an interpolation in the source interval but does not involve an interpolation in the target interval.

One-sided interpolation in the target interval I_1. By combing an interpolation in the temporal variable $t \in I_1$ with a truncated Chebyshev expansion in both spatial variables $\boldsymbol{x} \in X_1$ and $\boldsymbol{y} \in X_2$ we obtain

$$\mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1 \times X_2} \mathfrak{I}^{(m_t)}_{\mathrm{dir},I_1}[G_\alpha](\boldsymbol{x},t,\boldsymbol{y},\tau) = \sum_{b=0}^{m_t} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu} \in \mathbb{N}_0^3: \\ |\boldsymbol{\kappa}+\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}}} E^{b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\tau)\, T_{X_1,\boldsymbol{\nu}}(\boldsymbol{x})\, T_{X_2,\boldsymbol{\kappa}}(\boldsymbol{y})\, L^{(m_t)}_{I_1,b}(t), \quad (7.3)$$

$$E^{b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\tau) = \frac{1}{(4\pi\alpha(\xi^{(m_t)}_{I_1,b} - \tau))^{3/2}} \prod_{j=1}^{3} E_{\kappa_j,\nu_j}(r_j, 4\alpha(\xi^{(m_t)}_{I_1,b} - \tau)/\widetilde{h}_x^2) \quad (7.4)$$

for $(\boldsymbol{x},t) \in Z_{\mathrm{tar}}$ and $(\boldsymbol{y},\tau) \in Z_{\mathrm{src}}$, where the expansion coefficients $E^{b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}$ depend explicitly on $\tau \in I_2$.

We plan to use the approximation (7.1) of the heat kernel for two boxes $Z_{\mathrm{tar}} = X_1 \times I_1$ and $Z_{\mathrm{src}} = X_2 \times I_2$ if the source interval I_2, in which we interpolate, is considerably smaller than the target interval I_1 and there holds $\mathrm{dist}(I_1,I_2) \gtrsim |I_2|$. Likewise, we want to use the approximation (7.3) if the target interval I_1 is considerably smaller than the source interval I_2 and there holds $\mathrm{dist}(I_1,I_2) \gtrsim |I_1|$. In the following subsection we analyze the approximation error of the two approximations (7.1) and (7.3) to show that they work well in these situations.

7.1.1 Analysis of the approximation error

The approximation error of the temporally one-sided expansions (7.1) and (7.3) can be estimated in the same way as the approximation error of the standard, temporally two-sided expansion (5.9) in Sections 5.1.1–5.1.3. We first focus on the interpolation error in time. As in Section 5.1.1 we consider the function $(t_1, t_2) \mapsto g(t_1, t_2)$ in (5.11) for this purpose, which corresponds to the heat kernel for fixed spatial points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$ and a fixed heat capacity constant α. The error that results when we consider this function g on a pair of time intervals I_1 and I_2 and interpolate it in the shorter one is estimated in the following theorem.

THEOREM 7.1 (One-sided interpolation error). *Let g be the function in (5.11). Let $\eta_1 > 0$ and $q_1 := 1 + 3/(2\,\eta_1)$. Let $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ be two non-empty intervals such that $a_1 > b_2$. Let the admissibility criterion*

$$\eta_1 \operatorname{dist}(I_1, I_2) \geq \min\{|I_1|, |I_2|\} \tag{7.5}$$

be satisfied. Let k be such that I_k is the shorter time interval. Then there exists a constant $c_1 > 0$ such that

$$\|g - \mathfrak{I}^{(m_t)}_{\mathrm{dir}, I_k}[g]\|_{\infty, I_1 \times I_2} \leq \frac{c_1}{(\alpha \operatorname{dist}(I_1, I_2))^{3/2}} q_1^{-(m_t+1)}. \tag{7.6}$$

Proof. The proof is very similar to the proof of Theorem 5.1 in Section 5.1.1. If we can show under the assumptions in Theorem 7.1 that g satisfies an estimate of the form (5.16), i.e.

$$\|\partial_{t_k}^n g\|_{\infty, I_1 \times I_2} \leq \frac{\widetilde{C}_g}{\widetilde{\gamma}_g^n} \frac{(n + \widetilde{\sigma} - 1)!}{(\widetilde{\sigma} - 1)!} \qquad \forall n \in \mathbb{N}_0$$

for constants $\widetilde{C}_g, \widetilde{\gamma}_g \in \mathbb{R}_{>0}$ and $\widetilde{\sigma} \in \mathbb{N}$, we can apply [14, Lemma 4.19] to obtain the estimate

$$\|g - \mathfrak{I}^{(m_t)}_{\mathrm{dir}, I_k}[g]\|_{\infty, I_1 \times I_2} \leq 2e\widetilde{C}_g(\Lambda_{m_t} + 1)(m_t + 2)^{\widetilde{\sigma}} \left(1 + \frac{|I_k|}{\widetilde{\gamma}_g}\right) \varrho\left(\frac{2\widetilde{\gamma}_g}{|I_k|}\right)^{-(m_t+1)},$$

where Λ_{m_t} is the Lebesgue constant defined in (5.3) and $\varrho(r) := r + \sqrt{1 + r^2}$. By using the admissibility criterion (7.5), we can show that (5.16) holds for $\widetilde{\sigma} = 1$, $\widetilde{\gamma}_g = 3 \min\{|I_1|, |I_2|\}/(4\eta_1)$ and $\widetilde{C}_g = c_{\mathrm{as}} c_\beta / (\alpha \operatorname{dist}(I_1, I_2))^{3/2}$ in the same way as we did in the proof of Theorem 5.1. Therefore,

$$\|g - \mathfrak{I}^{(m_t)}_{\mathrm{dir}, I_k}[g]\|_{\infty, I_1 \times I_2} \leq 2e\widetilde{C}_g(\Lambda_{m_t} + 1)(m_t + 2)^{\widetilde{\sigma}} \left(1 + \frac{4}{3}\eta_1\right) \varrho\left(\frac{3}{2\eta_1}\right)^{-(m_t+1)}.$$

This estimate can be simplified in the same way as the corresponding result in the proof of Theorem 5.1 to show (7.6). □

By combining the interpolation error estimate in Theorem 7.1 with the truncation error estimate of the Chebyshev expansion in space in Theorem 5.7 we can estimate the approximation errors of the temporally one-sided expansions (7.1) and (7.3) as in the proof of Theorem 5.9 in Section 5.1.3. This yields the following result.

THEOREM 7.2 (Temporally one-sided space-time expansion errors). *Let $\widetilde{c}_{\mathrm{st}} \in \mathbb{R}_{>0}$, $\eta_1 \in \mathbb{R}_{>0}$ and $q_1 := 1 + 3/(2\,\eta_1)$. Let Λ_{m_t} be the Lebesgue constant defined in (5.3) and σ be the function in (5.20). Let $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ be two non-empty intervals with $a_1 > b_2$ that satisfy the admissibility criterion (7.5). Let X_1 and X_2 be two cubes in $\mathbb{R}^3$ with edge length $2\widetilde{h}_x$ such that the criterion (5.25) holds, i.e.*

$$4\alpha \operatorname{dist}(I_1, I_2)\, \widetilde{h}_x^{-2} \geq \widetilde{c}_{\mathrm{st}}.$$

Let $Z_{\mathrm{tar}} = X_1 \times I_1$ and $Z_{\mathrm{src}} = X_2 \times I_2$. Let $k \in \{1, 2\}$ be such that I_k is the shorter time interval. Then there exist constants c_1, $c_{\boldsymbol{x}} \in \mathbb{R}_{>0}$ such that

$$\begin{aligned} \|(\mathrm{Id} - \mathfrak{S}^{(m_{\boldsymbol{x}})}_{X_1 \times X_2} \mathfrak{I}^{(m_t)}_{\mathrm{dir}, I_k})[G_\alpha]\|_{\infty, Z_{\mathrm{tar}} \times Z_{\mathrm{src}}} &\leq \frac{1}{(\alpha \operatorname{dist}(I_1, I_2))^{3/2}} \\ &\times \Big(c_1 q_1^{-(m_t+1)} + c_{\boldsymbol{x}} \Lambda_{m_t} (m_{\boldsymbol{x}} + 2)^5 \exp\Big(-(m_{\boldsymbol{x}} + 1)\sigma\Big((m_{\boldsymbol{x}} + 1)\widetilde{c}_{\mathrm{st}}/12\Big)\Big)\Big). \end{aligned} \tag{7.7}$$

REMARK 7.3. *By comparing the error estimates for the two-sided interpolation in Theorem 5.1 in Section 5.1.1 with the error estimates for the one-sided interpolation in Theorem 7.1, we see that in both estimates the error bound decreases exponentially with increasing interpolation degree m_t. The convergence rate depends on the constants η_2 and η_1 from the admissibility criteria (5.12) and (7.5), respectively, and suffers if they become large. For two fixed, separate intervals I_1 and I_2 we can choose $\eta_2 = \max\{|I_1|, |I_2|\}/\operatorname{dist}(I_1, I_2)$ and $\eta_1 = \min\{|I_1|, |I_2|\}/\operatorname{dist}(I_1, I_2)$ as the smallest constants for which (5.12) and (7.5) hold. Hence, in all situations where $\max\{|I_1|, |I_2|\}/\operatorname{dist}(I_1, I_2)$ is considerably larger than $\min\{|I_1|, |I_2|\}/\operatorname{dist}(I_1, I_2)$ the one-sided interpolation in the shorter time interval is expected to perform significantly better than the two-sided interpolation. This property carries over to the temporally one-sided space-time expansions (7.1) and (7.3) of the heat kernel when we compare them with the standard expansion (5.9).*

REMARK 7.4. *In Theorem 7.2 we use criterion (5.25) to determine a proper spatial size of the boxes for the expansions (7.1) and (7.3). This is the same criterion that we have used in Theorem 5.9 for the standard expansion (5.9), but in practice we have replaced it with the space-time configuration criterion (5.27) which requires that*

$$\frac{\widetilde{h}_{\boldsymbol{x}}^2}{4\alpha \widetilde{h}_t} \leq c_{\mathrm{st}}$$

holds for a constant c_{st} and each of the considered boxes Z_{src} and Z_{tar}, where $\widetilde{h}_{\boldsymbol{x}}$ denotes the half length of the edges of their spatial components and $\widetilde{h}_t$ the half length of

their temporal intervals; see Remark 5.10. In the setting of Theorem 7.2 we can also eliminate (5.25) *if we require instead that the space-time configuration criterion* (5.27) *is satisfied for the space-time box* Z_{tar} *or* Z_{src} *corresponding to the smaller time interval. In fact, since the spatial sizes of the two boxes coincide and the admissibility criterion* (7.5) *is satisfied, we get*

$$\frac{4\alpha \operatorname{dist}(I_1, I_2)}{\widetilde{h}_{\boldsymbol{x}}^2} \geq \frac{4\alpha \min\{|I_1|, |I_2|\}}{\widetilde{h}_{\boldsymbol{x}}^2 \eta_1} \geq \frac{2}{c_{\mathrm{st}} \eta_1}$$

in this case, which is criterion (5.25) *with* $\widetilde{c}_{\mathrm{st}} = 2(c_{\mathrm{st}}\eta_1)^{-1}$. *Therefore, we continue to use* (5.27) *instead of* (5.25) *in the rest of this work.*

7.2 Description of the time-adaptive FMM

In this section we discuss how to adapt the space-time FMM in Chapter 5 to obtain the new time-adaptive FMM which is better suited for the compression of BEM matrices related to temporally adaptive meshes. The key idea is that certain inadmissible blocks of the BEM matrices in the standard FMM can be further subdivided and partially approximated by using the temporally one-sided expansions of the heat kernel from Section 7.1. In the following sections we present this subdivision process, the resulting new operation lists, and the new FMM operations stemming from the temporally one-sided kernel expansions. We start by revisiting and extending the underlying space-time box cluster trees.

7.2.1 Extended space-time box cluster trees

In Section 5.2.1 we have described how to construct a space-time box cluster tree $\mathcal{T}_\Sigma$ to partition a space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$. The corresponding Algorithm 5.1 can also be applied to a mesh Σ_h whose temporal partition $\mathcal{I}_{h_t}$ contains non-uniform time intervals. However, the resulting tree $\mathcal{T}_\Sigma$ may contain some large leaf clusters with many associated space-time elements. In fact, in the recursive construction of $\mathcal{T}_\Sigma$ the clusters are always subdivided either with respect to time or with respect to space and time. If the temporal component I of a cluster Z contains only a single time interval (t_{j_t-1}, t_{j_t}) in $\mathcal{I}_{h_t}$, a temporal subdivision is not possible anymore, and thus Z is not further subdivided even if it still contains many space-time elements. This is undesired because large leaves correspond to large inadmissible blocks in the FMM in Chapter 5 which may break its efficiency. In this section we describe how to extend a tree $\mathcal{T}_\Sigma$ by further subdividing such kind of clusters in space. The new clusters obtained in this manner will enable us to introduce new FMM operations in the following sections.

To describe the additional subdivision of clusters in this section, we use the same notation as in Section 5.1. In particular, we use $\hat{Z}$ to denote the set of all indices $(j_t, j_{\boldsymbol{x}})$ corresponding to elements $\sigma_{j_t,j_{\boldsymbol{x}}} \in \Sigma_h$ that are assigned to a cluster Z. The quantity $n_t(\hat{Z})$ denotes the number of all distinct time-indices in $\hat{Z}$, i.e. the number of time intervals (t_{j_t-1}, t_{j_t}) contained in the temporal part I of Z. So-called temporally indivisible clusters are of particular interest.

DEFINITION 7.5. *A cluster Z is called temporally indivisible if $n_t(\hat{Z}) = 1$.*

We extend a given space-time box cluster tree $\mathcal{T}_\Sigma$ by recursively subdividing temporally indivisible clusters in space. The resulting extended space-time box cluster tree is denoted by $\mathcal{T}_\Sigma^{\text{ext}}$ to distinguish it from the original tree $\mathcal{T}_\Sigma$. The corresponding procedure is described in Algorithm 7.1.

Algorithm 7.1 Construction of an extended space-time box cluster tree $\mathcal{T}_\Sigma^{\text{ext}}$.

Require: Let a space-time tensor product mesh Σ_h be given.
Let a tree $\mathcal{T}_\Sigma$ be constructed by Algorithm 5.1 with parameters c_{st} and $n_{\max}$.
Initialize $\mathcal{T}_\Sigma^{\text{ext}}$ by $\mathcal{T}_\Sigma$.
for each temporally indivisible leaf Z in $\mathcal{T}_\Sigma$
  Call SUBDIVIDECLUSTERINSPACE(Z, $\mathcal{T}_\Sigma^{\text{ext}}$).

function SUBDIVIDECLUSTERINSPACE(Z, $\mathcal{T}_\Sigma^{\text{ext}}$)
  if $\#\hat{Z} \geq n_{\max}$ **and** a spatial subdivision is feasible
    Spatial subdivision of Z into eight children $\{Z_k\}_{k=1}^8$.
    for $k = 1, \ldots, 8$
      if $\#\hat{Z}_k \neq 0$
        Add Z_k to $\mathcal{T}_\Sigma^{\text{ext}}$ as a child of Z.
        Call SUBDIVIDECLUSTERINSPACE(Z_k, $\mathcal{T}_\Sigma^{\text{ext}}$).

A temporally indivisible leaf cluster $Z = X \times I$ in $\mathcal{T}_\Sigma$ is recursively subdivided in the routine SUBDIVIDECLUSTERINSPACE if more than $n_{\max}$ space-time elements are assigned to it and if a spatial subdivision is feasible. Recall from Section 5.2.1 that a spatial subdivision of Z is feasible, if $\max_{(j_t,j_{\boldsymbol{x}})\in\hat{Z}} \operatorname{diam}(\gamma_{j_{\boldsymbol{x}}}) \leq \widetilde{h}_{\boldsymbol{x}}(Z)$, where $\widetilde{h}_{\boldsymbol{x}}(Z)$ is the half edge length of X. For the spatial subdivision of Z we split the box $X = (\boldsymbol{a}, \boldsymbol{b}]$ uniformly into eight boxes $(\boldsymbol{a}, \widetilde{\boldsymbol{a}}], \ldots, (\widetilde{\boldsymbol{a}}, \boldsymbol{b}]$, where $\widetilde{\boldsymbol{a}} = 1/2(\boldsymbol{a} + \boldsymbol{b})$ is the center of $(\boldsymbol{a}, \boldsymbol{b}]$. The resulting boxes are combined with the time interval I to get eight space-time boxes $\{Z_k\}_{k=1}^8$. Note that the same kind of subdivision in space combined with an additional subdivision in time is used to subdivide clusters with respect to space and time in Algorithm 5.1. A box Z_k obtained by a spatial subdivision of Z is added to the extended tree $\mathcal{T}_\Sigma^{\text{ext}}$ only if it is not empty, i.e. if there are space-time elements in Σ_h assigned to it. Then it is further subdivided if possible. The construction of the extended tree $\mathcal{T}_\Sigma^{\text{ext}}$ is complete after all temporally indivisible clusters of $\mathcal{T}_\Sigma$ have been recursively subdivided in this manner.

We denote the level of a cluster Z in the extended tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ by $\ell(Z)$ and define the temporal level $\ell_t(Z)$ and the spatial level $\ell_{\boldsymbol{x}}(Z)$ of Z in the same way as in Section 5.2.1, i.e. as the number of temporal or spatial subdivisions performed to obtain Z from the root $Z^{(0)}$ of $\mathcal{T}_\Sigma^{\mathrm{ext}}$. Similarly as in that section, we also pad the clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ to ensure that a space-time element $\sigma_{j_t,j_{\boldsymbol{x}}}$ assigned to a cluster Z is fully contained in it. For this purpose, we use a similar uniform padding strategy as in Section 5.2.1, where we pad all boxes with the same spatial level $\ell_{\boldsymbol{x}}$ in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ by the same amount. In the following sections we assume that clusters are padded in this way and identify clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ with their padded versions as we did in Chapter 5.

7.2.2 Operation lists for the time-adaptive FMM

The operation lists which we have constructed in Section 5.2.2 for clusters in a space-time box cluster tree $\mathcal{T}_\Sigma$ induce a partition of the matrix V_h into admissible blocks, which we approximate in the FMM, and inadmissible blocks which are applied directly. The extension $\mathcal{T}_\Sigma^{\mathrm{ext}}$ of $\mathcal{T}_\Sigma$ and the two new expansions (7.1) and (7.3) of the heat kernel (3.4) allow us to further subdivide certain blocks which are inadmissible in the setting of Section 5.2.2, and to obtain new kinds of admissible blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ for which the standard approximation in Section 5.2.3 would not work well in general. In this section we construct new operation lists corresponding to these new kinds of admissible blocks. Their approximation is then described in Section 7.2.3.

As a first step, we need to identify pairs of clusters Z_{tar} and Z_{src} for which the temporally one-sided expansions (7.1) and (7.3) of the heat kernel are well-suited. In Theorem 7.2 and Remark 7.4 we have seen that the expansion (7.1) of the heat kernel in two clusters Z_{tar} and Z_{src} of the same spatial size approximates the heat kernel well if the temporal intervals I_{tar} and I_{src} satisfy the admissibility criterion (7.5), I_{src} is the shorter interval, and the cluster Z_{src} satisfies the space-time configuration criterion (5.27). For the expansion (7.3) the spatial sizes of Z_{tar} and Z_{src} have to coincide, the admissibility criterion (7.5) has to be satisfied, I_{tar} has to be the shorter time interval, and Z_{tar} has to satisfy the space-time configuration criterion (5.27). In both cases, it suffices to consider pairs of clusters Z_{tar} and Z_{src} where I_{src} is causally relevant for I_{tar} because for all other pairs the heat kernel is constantly zero and so are the related subblocks of V_h.

In Section 5.2.2 we have introduced the interaction lists $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ of clusters Z_{tar} in $\mathcal{T}_\Sigma$ to keep track of corresponding clusters Z_{src} for which the standard expansion (5.9) of the heat kernel is suitable. These lists were filled using a recursive subdivision procedure in Algorithm 5.2 which ensures that interactions between clusters take place on the coarsest level possible. For the new kernel expansions we introduce two new operation lists in addition to those in Section 5.2.2, namely the M2Lx and Mx2L interaction lists of clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$. A cluster Z_{src} is added to the M2Lx

interaction list $\mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}})$ of Z_{tar} if the temporally one-sided expansion (7.1) is suitable for Z_{tar} and Z_{src}, but the standard expansion (5.9) is not. Similarly, Z_{src} is added to the Mx2L interaction list $\mathcal{I}_{\text{Mx2L}}(Z_{\text{tar}})$ of Z_{tar} if the temporally one-sided expansion (7.3) is suitable, but (5.9) is not. The new lists are named after the corresponding FMM operations which we introduce in Section 7.2.3. They are filled together with the original operation lists by a recursive procedure, which is presented in Algorithm 7.2. In the following we call blocks $\mathsf{V}_h|_{\hat{Z}_{\text{tar}} \times \hat{Z}_{\text{src}}}$ *admissible* if $Z_{\text{src}} \in \mathcal{I}_{\text{M2L}}(Z_{\text{tar}})$, and *temporally one-sided admissible* if either $Z_{\text{src}} \in \mathcal{I}_{\text{Mx2L}}(Z_{\text{tar}})$ or $Z_{\text{src}} \in \mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}})$.

Algorithm 7.2 Construction of the operation lists for the time-adaptive FMM.

Require: Let $\mathcal{T}_\Sigma^{\text{ext}}$ be an extended space-time cluster tree with root $Z^{(0)}$.
Fix constants $\eta_1, \eta_2 \in \mathbb{R}_{>0}$ for the admissibility criteria (5.12) and (7.5).
Fix a constant n_{tr} for the definition of the spatial interaction areas $\mathcal{I}_{\mathcal{A}}$ in (5.31).

Call DetermineOperationListsExt($Z^{(0)}$, $Z^{(0)}$).

function DetermineOperationListsExt(Z_{src}, Z_{tar})
  if $X_{\text{src}} \in \mathcal{I}_{\mathcal{A}}(X_{\text{tar}})$ **and** I_{src} is causally relevant for I_{tar}
    if I_{src} and I_{tar} satisfy the admissibility criterion (5.12)
      Add Z_{src} to $\mathcal{I}_{\text{M2L}}(Z_{\text{tar}})$.
    else
      if Z_{tar} is not a leaf **and** Z_{tar} is not temporally indivisible
        if Z_{src} is not a leaf **and** Z_{src} is not temporally indivisible
          for all $(Z_{\text{src,c}}, Z_{\text{tar,c}})$ with $Z_{\text{src,c}} \in \text{child}(Z_{\text{src}})$, $Z_{\text{tar,c}} \in \text{child}(Z_{\text{tar}})$
            Call DetermineOperationListsExt($Z_{\text{src,c}}$, $Z_{\text{tar,c}}$).
        else // *Z_{src} is a leaf or a temporally indivisible cluster.*
          if Z_{src} is temporally indivisible
            for all $Z_{\text{tar,c}} \in \text{child}(Z_{\text{tar}})$
              Call DetermineMx2LAndNFLists(Z_{src}, $Z_{\text{tar,c}}$).
          else // *Z_{src} is not a temporally indivisible cluster but a leaf.*
            Add Z_{src} to $\mathcal{N}(Z_{\text{tar}})$.
      else // *Z_{tar} is a leaf or a temporally indivisible cluster.*
        if Z_{tar} is temporally indivisible
          if Z_{src} is not a leaf **and** Z_{src} is not temporally indivisible
            for all $Z_{\text{src,c}} \in \text{child}(Z_{\text{src}})$
              Call DetermineM2LxAndNFLists($Z_{\text{src,c}}$, Z_{tar}).
          else
            Add Z_{src} to $\mathcal{N}(Z_{\text{tar}})$.
        else // *Z_{tar} is not a temporally indivisible cluster but a leaf.*
          Add Z_{src} to $\mathcal{N}(Z_{\text{tar}})$.

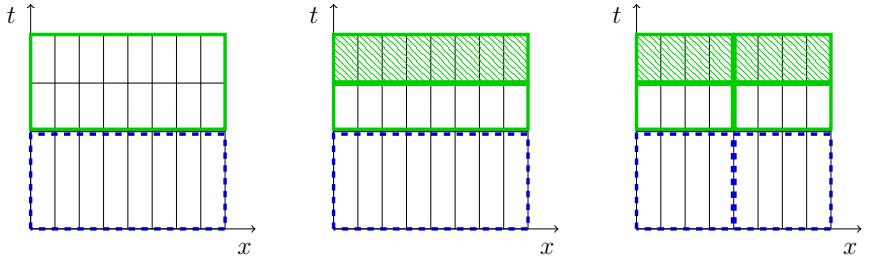
Algorithm 7.2 can be seen as an extension of Algorithm 5.2 in Section 5.2.2. The operation lists are constructed by using the recursive routine DetermineOperationListsExt that acts on pairs of clusters $(Z_{\mathrm{src}}, Z_{\mathrm{tar}})$ in the extended cluster tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$, which we denote as blocks in this context. Lines 3–10 of Algorithm 7.2 are similar to lines 3–9 in Algorithm 5.2. Here we check if I_{src} is causally relevant for I_{tar} or if a block can be neglected, which is the case if $X_{\mathrm{src}} \notin \mathcal{I}_{\mathcal{A}}(X_{\mathrm{tar}})$; see (5.31) and the related discussion. If I_{src} and I_{tar} satisfy the standard admissibility criterion (5.12), the corresponding block is admissible and we add Z_{src} to the interaction list $\mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$. Otherwise, we subdivide the block recursively. As long as Z_{src} and Z_{tar} are neither leaves nor temporally indivisible clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$, we subdivide the corresponding block as in Algorithm 5.2 by considering all pairs of blocks related to children of Z_{src} and Z_{tar} in recursive calls of the routine DetermineOperationListsExt.

Lines 11–25 in Algorithm 7.2 are new compared to Algorithm 5.2. Here we deal with the situations where Z_{src} or Z_{tar} is temporally indivisible or a leaf in $\mathcal{T}_\Sigma^{\mathrm{ext}}$. Such clusters would be leaves in the non-extended tree $\mathcal{T}_\Sigma$, so we would add Z_{src} to the nearfield $\mathcal{N}(Z_{\mathrm{tar}})$ of Z_{tar} in these cases in Algorithm 5.2. Instead, we try to identify temporally one-sided admissible blocks by special one-sided block subdivisions in Algorithm 7.2. Such a subdivision is performed in lines 12–14 and 19–21 if either Z_{src} or Z_{tar} is temporally indivisible and the respective other cluster can be further subdivided in time, i.e. it is neither a leaf nor temporally indivisible. If Z_{src} is temporally indivisible and Z_{tar} can be further subdivided in time, we can consider the blocks of V_h corresponding to Z_{src} and $Z_{\mathrm{tar,c}} \in \mathrm{child}(Z_{\mathrm{tar}})$. For these blocks a temporally one-sided expansion (7.3) instead of the full expansion (5.9) may be admissible, since the subdivision of Z_{tar} in time leads to a temporal separation between some of the children of Z_{tar} and Z_{src}, see Figures 7.1a and 7.1b. We check the blocks for this kind of admissibility in the subroutine DetermineMx2LAndNFLists in line 14, which is described in Algorithm 7.3 and further discussed below. Similarly, if Z_{tar} is temporally indivisible and Z_{src} is neither a leaf nor temporally indivisible, we can subdivide Z_{src} and treat the corresponding blocks separately in the routine DetermineM2LxAndNFLists in line 21 and Algorithm 7.4.

For some clusters Z_{src} and Z_{tar} in lines 11–25 of the routine DetermineOperationListsExt further subdivisions of the corresponding blocks are not possible or would not yield efficiently approximable subblocks anymore. This is the case if:

- Z_{src} is a leaf, which is not temporally indivisible, i.e. I_{src} contains more than one time interval in the temporal partition $\mathcal{I}_{h_t}$ corresponding to Σ_h.
- Z_{tar} is a leaf, which is not temporally indivisible.
- Z_{tar} and Z_{src} are both temporally indivisible.

In all these cases we add Z_{src} to the nearfield $\mathcal{N}(Z_{\mathrm{tar}})$ of Z_{tar}; see lines 16, 23 and 25.



a) Clusters Z_{src} and Z_{tar}. b) First subdivision case. c) Second subdivision case.

Figure 7.1: Illustration of the special subdivision of blocks/pairs $(Z_{\text{src}}, Z_{\text{tar}})$ to detect temporally one-sided admissible blocks. For the sake of simplicity, the mesh Σ_h and the clusters are drawn in 2D instead of 4D like in Figure 5.1. In a) a temporally indivisible source cluster Z_{src} is drawn with dashed blue lines and a non-leaf target cluster Z_{tar} with solid green lines. The children of Z_{tar} are either subdivided only in time (see b)), or in space and time (see c)). In both cases, Z_{src} is separated in time from the children of Z_{tar} which are hatched in green. In b), Z_{src} and the hatched child of Z_{tar} form a temporally one-sided admissible block/pair. In c), we have to subdivide Z_{src} in space to get clusters whose spatial sizes coincide with the spatial sizes of the children of Z_{tar}. Then, the spatially subdivided children of Z_{src} and the hatched children of Z_{tar} form temporally one-sided admissible blocks again.

The routine DetermineMx2LAndNFLists in Algorithm 7.3 is used to determine pairs/blocks $(Z_{\text{src}}, Z_{\text{tar}})$ for which the approximation (7.3) is appropriate. Whenever we call this routine for two candidates Z_{src} and Z_{tar}, Z_{src} is a temporally indivisible cluster whose temporal level $\ell_t(Z_{\text{src}})$ is smaller than the temporal level $\ell_t(Z_{\text{tar}})$ of Z_{tar}. The spatial sizes of the boxes Z_{tar} and Z_{src} have to coincide for the approximation (7.3), which is the case if the spatial levels $\ell_{\boldsymbol{x}}(Z_{\text{tar}})$ and $\ell_{\boldsymbol{x}}(Z_{\text{src}})$ of the clusters are the same. This is checked in line 2. Let us first assume that the spatial levels coincide. Then we can check if the temporal components I_{tar} and I_{src} satisfy the admissibility criterion (7.5). If this is the case, we have detected a temporally one-sided admissible block, and thus we add Z_{src} to the Mx2L interaction list $\mathcal{I}_{\text{Mx2L}}(Z_{\text{tar}})$ of Z_{tar}; see line 4. Otherwise, we want to recursively subdivide the current block. If Z_{tar} is a leaf or temporally indivisible, a further subdivision is not possible or would not yield admissible subblocks anymore, so we add Z_{src} to the nearfield $\mathcal{N}(Z_{\text{tar}})$ of Z_{tar}; see line 7. In the contrary case, we subdivide the current block by recursively calling the routine DetermineMx2LAndNFLists for Z_{src} and all children of Z_{tar}; see line 10. Note that we subdivide only Z_{tar} here, because Z_{src} is temporally indivisible. Thus, only a subdivision of Z_{tar} leads to a different temporal configuration of the resulting blocks which might then satisfy the admissibility criterion (7.5).

Algorithm 7.3 Determine Mx2L interaction lists by asymmetric subdivisions.

```
1: function DetermineMx2LAndNFLists(Z_src, Z_tar)
2:     if ℓ_x(Z_src) == ℓ_x(Z_tar)
3:         if I_src and I_tar satisfy the admissibility criterion (7.5)
4:             Add Z_src to I_Mx2L(Z_tar).
5:         else
6:             if Z_tar is temporally indivisible or Z_tar is a leaf
7:                 Add Z_src to N(Z_tar).
8:             else      // Subdivision of Z_tar in space and time.
9:                 for all Z_tar,c ∈ child(Z_tar)
10:                    Call DetermineMx2LAndNFLists(Z_src, Z_tar,c).
11:    else
12:        if Z_src is a leaf
13:            Add Z_src to N(Z_tar).
14:        else      // Spatial subdivision of Z_src.
15:            for all Z_src,c ∈ child(Z_src)
16:                Call DetermineMx2LAndNFLists(Z_src,c, Z_tar).
```

Let us now discuss the case when the spatial levels of the clusters Z_{src} and Z_{tar} in the routine DetermineMx2LAndNFLists in Algorithm 7.3 do not coincide; see lines 11–16. Whenever we call this routine, there holds $\ell_{\boldsymbol{x}}(Z_{\mathrm{src}}) \leq \ell_{\boldsymbol{x}}(Z_{\mathrm{tar}})$, so $\ell_{\boldsymbol{x}}(Z_{\mathrm{src}})$ is less than $\ell_{\boldsymbol{x}}(Z_{\mathrm{tar}})$ in this case. The proper spatial level is the one of Z_{tar}. This follows from Remark 7.4 and the construction of the original space-time box cluster tree $\mathcal{T}_\Sigma$, whose boxes satisfy the space-time configuration criterion (5.27). Therefore, we have to subdivide Z_{src} to obtain clusters with larger spatial levels. If Z_{src} is a leaf in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ this is not possible, and we add Z_{src} to the nearfield $\mathcal{N}(Z_{\mathrm{tar}})$ of Z_{tar} regardless of whether the temporal components satisfy (7.5) or not; see line 13. Otherwise we can subdivide Z_{src} and call the routine DetermineMx2LAndNFLists recursively for Z_{tar} and all children of Z_{src}; see line 16. Since Z_{src} is a temporally indivisible cluster, its children in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ are subdivided only with respect to space. Hence, their spatial level $\ell_{\boldsymbol{x}}$ is larger than the spatial level $\ell_{\boldsymbol{x}}(Z_{\mathrm{src}})$ of Z_{src} as desired.

To summarize, the main goal of the routine DetermineMx2LAndNFLists in Algorithm 7.3 is to identify temporally one-sided admissible blocks by recursively subdividing the target cluster Z_{tar} in time until the admissibility criterion (7.5) is satisfied for the temporal components of the resulting clusters or until a final inadmissible block is detected. During this procedure, the source cluster might need to be subdivided to ensure that the considered pairs of clusters have the same spatial level. This is also illustrated in Figure 7.1. Note that we do not check for an additional spatial truncation in Algorithm 7.3. This is motivated by the fact that the temporal configuration does not change significantly when only the target cluster is refined in time.

Since the decay of the heat kernel in space depends on the temporal configuration, an additional spatial truncation is thus not feasible.

The routine DetermineM2LxAndNFLists in Algorithm 7.4 is the analog of DetermineMx2LAndNFLists when the roles of source and target clusters are interchanged, i.e. when Z_{tar} is temporally indivisible while Z_{src} is not. In this case, we want to find blocks $(Z_{\mathrm{src}}, Z_{\mathrm{tar}})$ for which the approximation (7.1) is admissible. The recursion strategy including the choice of spatial levels is completely analogous to the one in DetermineM2LxAndNFLists.

Algorithm 7.4 Determine M2Lx interaction lists by asymmetric subdivisions.

```
1:  function DetermineM2LxAndNFLists(Z_src, Z_tar)
2:      if ℓ_x(Z_tar) == ℓ_x(Z_src)
3:          if I_src and I_tar satisfy the admissibility criterion (7.5)
4:              Add Z_src to I_M2Lx(Z_tar).
5:          else
6:              if Z_src is temporally indivisible or Z_src is a leaf
7:                  Add Z_src to N(Z_tar).
8:              else      // Subdivision of Z_src in space and time.
9:                  for all Z_src,c ∈ child(Z_src)
10:                     Call DetermineM2LxAndNFLists(Z_src,c, Z_tar).
11:     else
12:         if Z_tar is a leaf
13:             Add Z_src to N(Z_tar).
14:         else      // Spatial subdivision of Z_tar.
15:             for all Z_tar,c ∈ child(Z_tar)
16:                 Call DetermineM2LxAndNFLists(Z_src, Z_tar,c).
```

Remark 7.6. *A purely spatially refined cluster Z_{src} in the tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ can only be considered as a source cluster in the subroutine* DetermineMx2LAndNFLists *in Algorithm 7.3, if the corresponding target cluster Z_{tar} has the same spatial level. In particular, there might exist spatially refined clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ which are never considered in this routine. For the same reason, such clusters might never be considered as target clusters in the subroutine* DetermineM2LxAndNearfieldLists *in Algorithm 7.4. Hence, $\mathcal{T}_\Sigma^{\mathrm{ext}}$ might contain clusters that are never visited in the routine* DetermineOperationListsExt *and its subroutines. These clusters are not relevant for the FMM and could be removed from the tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$. On the other hand, one could use them to subdivide large inadmissible blocks into smaller parts for a better parallel efficiency as we described in Remark 5.11. For the sake of a simpler description, we assume in the following that such fine clusters do not exist or have been removed from $\mathcal{T}_\Sigma^{\mathrm{ext}}$.*

7.2.3 Block approximation and new FMM operations

In Section 5.2.3 we discussed how the kernel approximation (5.9) is used to efficiently approximate admissible blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of V_h related to clusters Z_{src} and Z_{tar} with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ in the application of V_h in the space-time FMM. In this section we consider the new temporally one-sided admissible blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ where $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z_{\mathrm{tar}})$ or $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{Mx2L}}(Z_{\mathrm{tar}})$ and use the temporally one-sided kernel expansions (7.1) and (7.3) to approximate their application. As a result, we get new kinds of FMM operations and related purely spatial moments and local contributions of clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$. As in Section 5.2.4 we introduce additional nested FMM operations for the computation and evaluation of these spatial moments and local contributions.

To derive the new FMM operations related to the temporally one-sided admissible blocks we consider a vector $\boldsymbol{q} \in \mathbb{R}^{E_t E_{\boldsymbol{x}}}$ and the local product $\boldsymbol{f}^{\mathrm{loc}} = \mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$. The entries of $\boldsymbol{f}^{\mathrm{loc}}$ are given by

$$f^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} = \sum_{(j_t,j_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{src}}} q_{j_t,j_{\boldsymbol{x}}} \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} G_\alpha(\boldsymbol{x}-\boldsymbol{y}, t-\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t \tag{7.8}$$

for all indices $(k_t, k_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{tar}}$. In Section 5.2.3 we assumed that Z_{tar} and Z_{src} are clusters with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$ and replaced the heat kernel in (7.8) with the expansion (5.9) to derive the FMM operations. Here we assume that $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ is a temporally one-sided admissible block and distinguish the two cases $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z_{\mathrm{tar}})$ and $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{Mx2L}}(Z_{\mathrm{tar}})$.

Case 1: Let $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z_{\mathrm{tar}})$. By the construction of the M2Lx interaction lists in Algorithms 7.2 and 7.4, Z_{tar} is temporally indivisible in this case and, therefore, $\hat{Z}_{\mathrm{tar}}$ contains only a single time-index k_t. We use the temporally one-sided expansion (7.1) to replace the heat kernel in (7.8) and get

$$f^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} \approx \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} \sum_{a=0}^{m_t} \sum_{\substack{\boldsymbol{\kappa},\boldsymbol{\nu}\in\mathbb{N}_0^3:\\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|\le m_{\boldsymbol{x}}}} E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t) T_{X_{\mathrm{tar}},\boldsymbol{\nu}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\,\mu_{a,\boldsymbol{\kappa}}(Z_{\mathrm{src}}) \tag{7.9}$$

for all $(k_t, k_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{tar}}$, where the moments $\boldsymbol{\mu}(Z_{\mathrm{src}})$ are the same as in (5.38), i.e.

$$\mu_{a,\boldsymbol{\kappa}}(Z_{\mathrm{src}}) = \sum_{(j_t,j_{\boldsymbol{x}})\in\hat{Z}_{\mathrm{src}}} q_{j_t,j_{\boldsymbol{x}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} T_{X_{\mathrm{src}},\boldsymbol{\kappa}}(\boldsymbol{y}) L^{(m_t)}_{I_{\mathrm{src}},a}(\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau$$

for all $a \in \{0,\dots,m_t\}$ and $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \le m_{\boldsymbol{x}}$. The temporal integral in (7.9) is evaluated by using a Gauß–Legendre quadrature rule with $\rho_t + 1$ points, i.e.

$$\int_{t_{k_t-1}}^{t_{k_t}} E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t)\,\mathrm{d}t \approx \sum_{b=0}^{\rho_t} \omega_{k_t,b} E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\zeta_{k_t,b}), \tag{7.10}$$

where $\{\zeta_{k_t,b}\}_{b=0}^{\rho_t}$ and $\{\omega_{k_t,b}\}_{b=0}^{\rho_t}$ are the corresponding Gauß–Legendre quadrature points and weights on the time interval (t_{k_t-1}, t_{k_t}). In this way, we get an approximation $\widetilde{\boldsymbol{f}}^{\mathrm{loc}}$ of $\boldsymbol{f}^{\mathrm{loc}}$ which we compute in three steps:
S2M: We compute the moments $\boldsymbol{\mu}(Z_{\mathrm{src}})$ by (5.38).
M2Lx: We compute the *spatial local contributions* $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ by

$$\lambda_{\boldsymbol{\nu}}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}, Z_{\mathrm{src}}) = \sum_{b=0}^{\rho_t} \omega_{k_t,b} \sum_{\substack{\boldsymbol{\kappa}\in\mathbb{N}_0^3:\\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|\le m_{\boldsymbol{x}}}} \sum_{a=0}^{m_t} E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\zeta_{k_t,b}) \mu_{a,\kappa}(Z_{\mathrm{src}}) \tag{7.11}$$

for all $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \le m_{\boldsymbol{x}}$, where the coefficient function $t \mapsto E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(t)$ is given in (7.2).
Lx2T: For all $k_{\boldsymbol{x}}$ such that $(k_t, k_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{tar}}$ we evaluate

$$\widetilde{f}^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} = \sum_{|\boldsymbol{\nu}|\le m_{\boldsymbol{x}}} \int_{\gamma_{k_{\boldsymbol{x}}}} T_{X_{\mathrm{tar}},\boldsymbol{\nu}}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \lambda_{\boldsymbol{\nu}}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}, Z_{\mathrm{src}}). \tag{7.12}$$

Case 2: Let $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{Mx2L}}(Z_{\mathrm{tar}})$. In this case, Z_{src} is temporally indivisible and $\hat{Z}_{\mathrm{src}}$ contains only a single time-index j_t. We substitute the heat kernel in (7.8) with the temporally one-sided expansion in (7.3) and proceed similarly as in case 1 to obtain an approximation $\widetilde{\boldsymbol{f}}^{\mathrm{loc}}$ of $\boldsymbol{f}^{\mathrm{loc}}$ in the following three steps:
S2Mx: We compute the *spatial moments* $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z_{\mathrm{src}})$ by

$$\mu_{\boldsymbol{\kappa}}^{(\boldsymbol{x})}(Z_{\mathrm{src}}) = \sum_{j_{\boldsymbol{x}} \,:\, (j_t, j_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{src}}} q_{j_t,j_{\boldsymbol{x}}} \int_{\gamma_{j_{\boldsymbol{x}}}} T_{X_{\mathrm{src}},\boldsymbol{\kappa}}(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \tag{7.13}$$

for all $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \le m_{\boldsymbol{x}}$.
Mx2L: We compute the local contributions $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$

$$\lambda_{b,\boldsymbol{\nu}}(Z_{\mathrm{tar}}, Z_{\mathrm{src}}) = \sum_{a=0}^{\rho_t} \sum_{\substack{\boldsymbol{\kappa}\in\mathbb{N}_0^3:\\ |\boldsymbol{\kappa}+\boldsymbol{\nu}|\le m_{\boldsymbol{x}}}} \omega_{j_t,a} E^{b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\zeta_{j_t,a}) \mu_{\boldsymbol{\kappa}}^{(\boldsymbol{x})}(Z_{\mathrm{src}}) \tag{7.14}$$

for $b \in \{0, \dots, m_t\}$ and $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \le m_{\boldsymbol{x}}$, where $\{\zeta_{j_t,a}\}_{a=0}^{\rho_t}$ and $\{\omega_{j_t,a}\}_{a=0}^{\rho_t}$ are Gauß–Legendre quadrature points and weights on the time interval (t_{j_t-1}, t_{j_t}), and $\tau \mapsto E^{b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\tau)$ is the coefficient function given in (7.4).
L2T: We evaluate the local contributions $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ by (5.40), i.e. we compute

$$\widetilde{f}^{\mathrm{loc}}_{k_t,k_{\boldsymbol{x}}} = \sum_{b=0}^{m_t} \sum_{|\boldsymbol{\nu}|\le m_{\boldsymbol{x}}} \lambda_{b,\boldsymbol{\nu}}(Z_{\mathrm{tar}}, Z_{\mathrm{src}}) \int_{t_{k_t-1}}^{t_{k_t}} \int_{\gamma_{k_{\boldsymbol{x}}}} T_{X_{\mathrm{tar}},\boldsymbol{\nu}}(\boldsymbol{x}) L^{(m_t)}_{I_{\mathrm{tar},b}}(t) \,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \,\mathrm{d}t$$

for all $(k_t, k_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{tar}}$.

The abbreviations Mx and Lx in the M2Lx, Lx2T, S2Mx and Mx2L operations indicate that these operations involve purely spatial moments and local contributions. For example, in the M2Lx operation (7.11) a standard moment (M) is transformed into a spatial local contribution (Lx). These FMM operations and the corresponding purely spatial moments and local contributions are new compared to the FMM operations in Section 5.2.3. This is related to the fact that the expansions (7.1) and (7.3) of the heat kernel incorporate an interpolation in only one of the temporal variables instead of an interpolation in both variables as the expansion (5.9) does. However, all three expansions include an additional truncated Chebyshev expansion in both spatial variables. Thus, we have an expansion in space and time in Z_{src} and only an expansion in space in Z_{tar} in the case of (7.1), and vice versa in the case of (7.3). The spatial local contributions in (7.11) and (7.12) and the spatial moments in (7.13) and (7.14) originate from this purely spatial expansion in the respective cluster. Note that the spatial moments depend explicitly on the time interval (t_{j_t-1}, t_{j_t}) contained in I_{src} and the spatial local contributions depend explicitly on the time interval (t_{k_t-1}, t_{k_t}) contained in I_{tar}. Since these time clusters contain only a single time interval whenever we construct spatial moments or spatial local contributions we do not indicate this dependence in the notation.

In comparison to the standard S2M and L2T operations in (5.38) and (5.40), the temporal integrals over the Lagrange polynomials are missing in the S2Mx and Lx2T operations in (7.13) and (7.12). The corresponding integration in time is included in the Mx2L and M2Lx operations in (7.14) and (7.11) instead where it is carried out by a numerical quadrature; see (7.10). It is not surprising that these Mx2L and M2Lx operations are similar to the M2L operation in (5.39), since the additional interpolation in the kernel expansion (5.9), which leads to the M2L operation, can be interpreted as an alternative quadrature formula in the context of (7.10). However, we can freely choose the quadrature formula in (7.10) for the M2Lx operations — and likewise for the Mx2L operations — and adapt it for individual time intervals to obtain better accuracy. The discussion in Remark 7.3 indicates that quadrature formulae with higher accuracies are required in those situations, where we use the temporally one-sided expansions. Alternatively, we could analytically evaluate the integrals of the expansion coefficients $E^{a,}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\cdot_t)$ and $E^{,b}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(\cdot_\tau)$ for the M2Lx and Mx2L operations, respectively. However, this would destroy the product structure of the coefficients and prohibit an efficient execution as in [69, Section 4.3] which we use for the Mx2L and M2Lx operations in the same way as for the M2L operations in our implementation.

The newly introduced spatial moments $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ and spatial local contribution $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ of clusters $Z \in \mathcal{T}_\Sigma^{\mathrm{ext}}$ can be computed and evaluated in a nested way like the standard moments $\boldsymbol{\mu}(Z)$ and standard local contributions $\boldsymbol{\lambda}(Z)$ in Section 5.2.4. The corresponding new FMM operations are called Mx2Mx and Lx2Lx operations in this work and can be derived like the space-time M2M and L2L operations in (5.49) and (5.51)

by using the identity (5.47) in Section 5.2.4, i.e. a suitable change of basis of the involved Chebyshev polynomials. Furthermore, the spatial moments $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ of a cluster Z can be used to compute its standard moments $\boldsymbol{\mu}(Z)$ if both are needed, and standard local contributions of Z can be transformed into spatial local contributions of Z. We describe the related Mx2M and L2Lx operations together with the Mx2Mx and Lx2Lx operations in the following.

Mx2Mx: Let $Z = X \times I$ be a temporally indivisible cluster in $\mathcal{T}_\Sigma^{\mathrm{ext}}$, whose children are obtained by a purely spatial subdivision. Its spatial moments $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ can be computed by

$$\boldsymbol{\mu}^{(\boldsymbol{x})}(Z) = \sum_{\substack{Z_\mathrm{c} \in \mathrm{child}(Z) \\ Z_\mathrm{c} = X_\mathrm{c} \times I}} \boldsymbol{\mu}^{(\boldsymbol{x})}(Z, Z_\mathrm{c}),$$

where the spatial moments $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z, Z_\mathrm{c})$ of Z related to a child $Z_\mathrm{c} = X_\mathrm{c} \times I$ are computed by the Mx2Mx operation

$$\mu_{\boldsymbol{\nu}}^{(\boldsymbol{x})}(Z, Z_\mathrm{c}) = \sum_{\boldsymbol{\kappa} \leq \boldsymbol{\nu}} q_{\boldsymbol{\kappa},\boldsymbol{\nu}}^{(\boldsymbol{x})}(X_\mathrm{c}, X) \mu_{\boldsymbol{\kappa}}^{(\boldsymbol{x})}(Z_\mathrm{c}) \tag{7.15}$$

for all $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}$. The coefficients $q_{\boldsymbol{\kappa},\boldsymbol{\nu}}^{(\boldsymbol{x})}(X_\mathrm{c}, X)$ which are used here are those in (5.47).

Mx2M: For a temporally indivisible cluster $Z = X \times I$ in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ we can compute the standard moments $\boldsymbol{\mu}(Z)$ defined in (5.38) from the spatial moments $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ defined in (7.13) by including the temporal integrals of the Lagrange polynomials. This yields the Mx2M operation

$$\mu_{a,\boldsymbol{\kappa}}(Z) = \mu_{\boldsymbol{\kappa}}^{(\boldsymbol{x})}(Z) \int_{t_{j_t-1}}^{t_{j_t}} L_{I,a}^{(m_t)}(\tau) \, \mathrm{d}\tau \tag{7.16}$$

for all $a \in \{0, \ldots, m_t\}$ and $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \leq m_{\boldsymbol{x}}$, where (t_{j_t-1}, t_{j_t}) is the only time interval contained in the temporal part I of the temporally indivisible cluster Z. Note that such a conversion is only necessary for a cluster Z if its parent is not temporally indivisible. The standard moments of all other temporally indivisible clusters are not needed.

L2Lx: For a temporally indivisible cluster $Z = X \times I$ in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ we can transform the standard local contributions $\boldsymbol{\lambda}(Z)$ into spatial local contributions $\boldsymbol{\lambda}^{(\boldsymbol{x}),\mathrm{trf}}(Z)$ by using the L2Lx operation

$$\lambda_{\boldsymbol{\nu}}^{(\boldsymbol{x}),\mathrm{trf}}(Z) = \sum_{b=0}^{m_t} \lambda_{b,\boldsymbol{\nu}}(Z) \int_{t_{k_t-1}}^{t_{k_t}} L_{I,b}^{(m_t)}(t) \, \mathrm{d}t \tag{7.17}$$

for all $\boldsymbol{\nu} \in \mathbb{N}_0^3$ with $|\boldsymbol{\nu}| \leq m_{\boldsymbol{x}}$, where (t_{k_t-1}, t_{k_t}) is the only time interval contained in the temporal part I of the temporally indivisible cluster Z. This corresponds to

an evaluation of the local contributions $\boldsymbol{\lambda}(Z)$ with respect to time only. The resulting spatial local contributions $\boldsymbol{\lambda}^{(\boldsymbol{x}),\mathrm{trf}}(Z)$ can be added to the spatial local contributions $\widetilde{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(Z)$, which are those obtained by summing up all spatial local contributions related to M2Lx operations for Z and $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z)$ in (7.11), i.e.

$$\widetilde{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(Z) = \sum_{Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z)} \boldsymbol{\lambda}^{(\boldsymbol{x})}(Z, Z_{\mathrm{src}}). \tag{7.18}$$

In this way, we get the spatial local contributions

$$\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z) = \widetilde{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(Z) + \boldsymbol{\lambda}^{(\boldsymbol{x}),\mathrm{trf}}(Z)$$

that can be evaluated with a single Lx2T operation or further processed with an Lx2Lx operation. Note that the standard local contributions $\boldsymbol{\lambda}(Z)$ of a temporally indivisible cluster Z will be computed in the later algorithm only if the parent of Z is not temporally indivisible. In particular, only for such clusters an L2Lx operation will be executed.

Lx2Lx: Let $Z = X \times I$ be a temporally indivisible cluster in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ and let $Z_{\mathrm{c}} = X_{\mathrm{c}} \times I$ be a child of Z obtained by a purely spatial subdivision. The spatial local contributions $\boldsymbol{\lambda}^{(\boldsymbol{x})}$ can be transformed into spatial local contributions $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{c}}, Z)$ of Z_{c} by the Lx2Lx operation

$$\lambda_{\boldsymbol{\kappa}}^{(\boldsymbol{x})}(Z_{\mathrm{c}}, Z) = \sum_{\boldsymbol{\nu} \geq \boldsymbol{\kappa}} q_{\boldsymbol{\kappa},\boldsymbol{\nu}}^{(\boldsymbol{x})}(X_{\mathrm{c}}, X) \lambda_{\boldsymbol{\nu}}(Z) \tag{7.19}$$

for all $\boldsymbol{\kappa} \in \mathbb{N}_0^3$ with $|\boldsymbol{\kappa}| \leq m_{\boldsymbol{x}}$, where the coefficients $q_{\boldsymbol{\kappa},\boldsymbol{\nu}}^{(\boldsymbol{x})}(X_{\mathrm{c}}, X)$ are given in (5.47). We can add $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{c}}, Z)$ and the spatial local contributions $\widetilde{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(Z_{\mathrm{c}})$ of Z_{c} that are defined as in (7.18) to get the spatial local contributions $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{c}})$, which can be evaluated by a single Lx2T operation or further processed by additional Lx2Lx operations.

7.2.4 The time-adaptive space-time FMM

In Algorithm 7.5 we present the new time-adaptive version of the space-time FMM in Chapter 5. This algorithm can be seen as an extension of Algorithm 5.3. The additional subdivision of formerly inadmissible blocks by Algorithms 7.3 and 7.4 leads to new FMM operations which allow for a more efficient treatment of space-time tensor product meshes which are adaptive in time. The related new purely spatial moments and local contributions are compatible with the standard moments and local contributions which allows us to compute and evaluate all of them in a nested way to avoid redundant computations.

Algorithm 7.5 The time-adaptive FMM for the approximate evaluation of $\boldsymbol{f} = \mathsf{V}_h \boldsymbol{q}$.

Require: Let a space-time box cluster tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ as in Algorithm 7.1 be given.
Let the operation lists be constructed by Algorithm 7.2.
Let expansion degrees m_t and $m_{\boldsymbol{x}}$ and a parameter ρ_t (for (7.10)) be given.

Initialize $\boldsymbol{f} = \boldsymbol{0}$.
Call ForwardTransformationTimeAdaptiveFMM( ). *// Alg. 7.6*

for all boxes $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma^{\mathrm{ext}}$ *// Multiplication phase*
  if Z_{tar} is temporally indivisible
    Initialize the spatial local contributions by setting $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}) = \boldsymbol{0}$.
    if $\mathrm{par}(Z_{\mathrm{tar}})$ is not temporally indivisible
      Initialize the local contributions by setting $\boldsymbol{\lambda}(Z_{\mathrm{tar}}) = \boldsymbol{0}$.
  else *// Z_{tar} is not temporally indivisible.*
    Initialize the local contributions by setting $\boldsymbol{\lambda}(Z_{\mathrm{tar}}) = \boldsymbol{0}$.
  for all boxes $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2L}}(Z_{\mathrm{tar}})$
    M2L: Add the result $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ from (5.39) to $\boldsymbol{\lambda}(Z_{\mathrm{tar}})$.
  for all boxes $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{Mx2L}}(Z_{\mathrm{tar}})$
    Mx2L: Add the result $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ from (7.14) to $\boldsymbol{\lambda}(Z_{\mathrm{tar}})$.
  for all boxes $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z_{\mathrm{tar}})$
    M2Lx: Add the result $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ from (7.11) to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}})$.

for all levels $\ell = 1, \ldots, \mathrm{depth}(\mathcal{T}_\Sigma^{\mathrm{ext}})$ *// Backward transformation*
  for all boxes $Z \in \mathcal{T}_\Sigma^{\mathrm{ext}}$ with $\ell(Z) == \ell$
    if $\mathrm{par}(Z)$ is temporally indivisible:
      Lx2Lx: Add the result $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z, \mathrm{par}(Z))$ from (7.19) to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$.
    else
      if Z results from $\mathrm{par}(Z)$ by a temporal refinement:
        Temporal L2L: Add the result $\boldsymbol{\lambda}(Z, \mathrm{par}(Z))$ from (5.50) to $\boldsymbol{\lambda}(Z)$.
      else
        Space-time L2L: Add the result $\boldsymbol{\lambda}(Z, \mathrm{par}(Z))$ from (5.51) to $\boldsymbol{\lambda}(Z)$.
      if Z is temporally indivisible:
        L2Lx: Add the result $\boldsymbol{\lambda}^{(\boldsymbol{x}),\mathrm{trf}}$ from (7.17) to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$.
for all leaves $Z \in \mathcal{T}_\Sigma^{\mathrm{ext}}$
  if Z is temporally indivisible:
    Lx2T: Evaluate $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ by (7.12) and add the result to $\boldsymbol{f}|_{\hat{Z}}$.
  else
    L2T: Evaluate $\boldsymbol{\lambda}(Z)$ by (5.40) and add the result to $\boldsymbol{f}|_{\hat{Z}}$.

for all $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma^{\mathrm{ext}}$ *// Nearfield evaluation*
  for all $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$
    Nearfield operation: Add the product $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}} \boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.

Algorithm 7.6 The forward transformation phase in Algorithm 7.5

function ForwardTransformationTimeAdaptiveFMM
  for all $Z \in \mathcal{T}_\Sigma^{\text{ext}}$
    if Z is temporally indivisible:
      Initialize the spatial moments by setting $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z) = \mathbf{0}$.
    else
      Initialize the moments by setting $\boldsymbol{\mu}(Z) = \mathbf{0}$.
  for all leaves $Z \in \mathcal{T}_\Sigma^{\text{ext}}$
    if Z is temporally indivisible:
      S2Mx: Compute $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ by (7.13).
    else
      S2M: Compute $\boldsymbol{\mu}(Z)$ by (5.38).
  for all levels $\ell = \text{depth}(\mathcal{T}_\Sigma^{\text{ext}}), \ldots, 1$
    for all $Z \in \mathcal{T}_\Sigma^{\text{ext}}$ with $\ell(Z) == \ell$
      if $\text{par}(Z)$ is temporally indivisible:
        Mx2Mx: Compute $\boldsymbol{\mu}^{(\boldsymbol{x})}(\text{par}(Z), Z)$ by (7.15).
        Add $\boldsymbol{\mu}^{(\boldsymbol{x})}(\text{par}(Z), Z)$ to $\boldsymbol{\mu}^{(\boldsymbol{x})}(\text{par}(Z))$.
      else
        if Z is temporally indivisible
          Mx2M: Compute $\boldsymbol{\mu}(Z)$ from $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ by (7.16).
        if Z results from $\text{par}(Z)$ by a temporal subdivision:
          Temporal M2M: Compute $\boldsymbol{\mu}(\text{par}(Z), Z)$ by (5.48).
        else
          Space-time M2M: Compute $\boldsymbol{\mu}(\text{par}(Z), Z)$ by (5.49).
          Add $\boldsymbol{\mu}(\text{par}(Z), Z)$ to $\boldsymbol{\mu}(\text{par}(Z))$.

The time-adaptive FMM in Algorithm 7.5 is split into the same four phases as the standard space-time FMM in Algorithm 5.3. In the new forward transformation phase, which is covered in Algorithm 7.6, we need to distinguish between temporally indivisible clusters and others. Spatial moments $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ are computed for temporally indivisible leaf clusters Z by S2Mx operations and passed to their temporally indivisible ancestors by Mx2Mx operations. For a temporally indivisible cluster Z whose parent is not temporally indivisible also the standard moments $\boldsymbol{\mu}(Z)$ are needed, which are obtained by an Mx2M operation. The resulting moments are used to compute the moments of clusters at lower levels in the tree with suitable M2M operations just like in the original Algorithm 5.3. If a leaf cluster Z in $\mathcal{T}_\Sigma^{\text{ext}}$ is not temporally indivisible, we compute its moments $\boldsymbol{\mu}(Z)$ directly by an S2M operation, and further process them with M2M operations. In this way, all the required spatial moments and standard moments of clusters in $\mathcal{T}_\Sigma^{\text{ext}}$ are computed in the forward transformation phase.

In the subsequent multiplication phase of Algorithm 7.5, we execute all the M2L operations as in Algorithm 5.3 and, in addition, all new M2Lx and Mx2L operations. The resulting local contributions and spatial local contributions are evaluated in a nested way in the backward transformation phase. Standard local contributions $\boldsymbol{\lambda}(Z)$ are passed from a cluster Z to its children by suitable L2L operations. As soon as a temporally indivisible cluster is encountered, the local contributions are transformed into spatial local contributions with an L2Lx operation. The spatial local contributions $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ are then passed from a cluster Z to its descendants by Lx2Lx operations. In the last part of the backward transformation phase, the spatial local contributions of temporally indivisible leaf cluster are evaluated by an Lx2T operation, while for all other leaf clusters the standard local contributions are evaluated by an L2T operation like in Algorithm 5.3. Finally, in the nearfield evaluation phase, all inadmissible blocks are applied directly just like in the corresponding phase of Algorithm 5.3.

7.3 Complexity analysis for newly approximated blocks

In this section we give an overview of the runtime complexities of the individual operations in the time-adaptive FMM. Furthermore, we compare the costs of the standard FMM in Algorithm 5.3 with the costs of the new time-adaptive FMM in Algorithm 7.5 for certain groups of temporally one-sided admissible blocks to better understand the benefits provided by the latter.

Note that we do not intend to give a complete complexity analysis of the full time-adaptive FMM here. The reason is that the operations in the time-adaptive FMM highly depend on the specific adaptive decomposition of the considered time interval $(0, T)$ which makes a general analysis without any restrictive assumptions difficult. Furthermore, an additional temporal nearfield compression is, in general, necessary to reduce the costs of nearfield operations related to coarse time steps, which may dominate the overall costs of the time-adaptive FMM. Such a nearfield compression scheme is discussed in detail in Chapter 8.

The runtime complexities of the new FMM operations are listed in Table 7.1. To enable a simple comparison we also list the runtime complexities of the original FMM operations in this table. Recall that we estimated the complexity of the S2M and L2T operations in Section 5.2.3 where we also discussed that the complexity $\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)^2)$ of the M2L operations is only achieved if they are executed as proposed in [69, Section 4.3]. Executing the M2Lx and Mx2L operations in the same way requires $\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)(\rho_t+1))$ arithmetic operations. The Mx2Mx and Lx2Lx operations have the complexity given in Table 7.1 if the product structure of the coefficients $q^{(\boldsymbol{x})}_{\boldsymbol{\kappa},\boldsymbol{\nu}}(X_{\mathrm{c}}, X)$ in (7.15) and (7.19) is used to split up the sums in the

computation; see also [67, Sections 4.2 and 4.3]. The same holds for the space-time M2M and L2L operations. The complexity of all other FMM operations can be estimated by simply counting the number of multiplications in the respective equations. By comparing the numbers in Table 7.1 we conclude that the new FMM operations are similarly efficient as the related standard FMM operations.

Operations	Runtime complexity
S2Mx/Lx2T ((7.13)/(7.12))	$\mathcal{O}(\#\hat{Z}\binom{m_{\boldsymbol{x}}+3}{3})$
Mx2Mx/Lx2Lx ((7.15)/(7.19))	$\mathcal{O}(\binom{m_{\boldsymbol{x}}+3}{3}\frac{m_{\boldsymbol{x}}}{4})$
Mx2M/L2Lx ((7.16)/(7.17))	$\mathcal{O}(\binom{m_{\boldsymbol{x}}+3}{3}(m_t+1))$
M2Lx/Mx2L ((7.11)/(7.14))	$\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)(\rho_t+1))$
S2M/L2T ((5.38)/(5.40))	$\mathcal{O}(\#\hat{Z}\binom{m_{\boldsymbol{x}}+3}{3}+n_t(\hat{Z})\binom{m_{\boldsymbol{x}}+3}{3}(m_t+1))$
Temporal M2M/L2L ((5.48)/(5.50))	$\mathcal{O}(\binom{m_{\boldsymbol{x}}+3}{3}(m_t+1)^2)$
Space-time M2M/L2L ((5.49)/(5.51))	$\mathcal{O}(\binom{m_{\boldsymbol{x}}+3}{3}(m_t+1)(\frac{m_{\boldsymbol{x}}}{4}+(m_t+1)))$
M2L ((5.39))	$\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)^2)$

Table 7.1: Runtime complexities of all FMM operations in the time-adaptive space-time FMM for the heat equation for given expansion orders $m_{\boldsymbol{x}}$ and m_t. The cluster Z appearing in the estimated costs for the S2M, L2T, S2Mx and Lx2T operations is the related source or target cluster and (ρ_t+1) is the number of quadrature points used in the M2Lx and Mx2L operations; cf. (7.10).

To study the effects of the approximation of temporally one-sided admissible blocks of V_h we consider an extended space-time cluster tree $\mathcal{T}_\Sigma^{\text{ext}}$ corresponding to a tensor-product mesh $\Sigma_h = \Gamma_h \times \mathcal{I}_{h_t}$, where $\mathcal{I}_{h_t}$ is a non-uniform partition of the time interval $(0,T)$ and Γ_h is a sufficiently fine spatial mesh with similarly sized elements. We focus on two fixed time clusters I_{src} and I_{tar} related to clusters in $\mathcal{T}_\Sigma^{\text{ext}}$ which satisfy the admissibility criterion (7.5) but not the admissibility criterion (5.12), and assume that I_{tar} is temporally indivisible, i.e. it contains only a single time interval of the partition $\mathcal{I}_{h_t}$. The temporally one-sided admissible blocks of V_h related to these time intervals are given by

$$\mathcal{B}(I_{\text{src}}, I_{\text{tar}}) = \{V_h|_{\hat{Z}_1\times\hat{Z}_2} : Z_1 = X_1 \times I_{\text{tar}},\ Z_2 = X_2 \times I_{\text{src}},\ Z_2 \in \mathcal{I}_{\text{M2Lx}}(Z_1)\}. \quad (7.20)$$

In the following, we compare the costs for the direct application of these blocks in the case of the standard FMM with the costs of the corresponding new time-adaptive FMM operations. Note that the analysis for interchanged roles of I_{src} and I_{tar} — i.e. a temporally indivisible cluster I_{src} and blocks $\mathsf{V}_h|_{\hat{Z}_1\times\hat{Z}_2}$ with $Z_2 \in \mathcal{I}_{\text{Mx2L}}(Z_1)$ — can be done following the same lines due to the symmetric character of the related FMM operations, which is why it is sufficient to focus on the above case.

We start by estimating the number of blocks in (7.20) or rather the corresponding pairs of clusters. From the construction of the operation lists in Algorithm 7.2 it follows that these clusters have the same spatial level $\ell_{\boldsymbol{x}}$. Due to the uniform spatial refinements in the construction of the extended space-time box cluster tree $\mathcal{T}_\Sigma^{\text{ext}}$ in Algorithm 7.1, the spatial components of the clusters with spatial level $\ell_{\boldsymbol{x}}$ are contained in a regular grid $\mathcal{G}^{\ell_{\boldsymbol{x}}}$ consisting of $8^{\ell_{\boldsymbol{x}}}$ boxes, as we already noted for the non-extended trees in Section 5.2.2. Since each target cluster Z_{tar} related to a block in (7.20) is a product of a spatial cluster in $\mathcal{G}^{\ell_{\boldsymbol{x}}}$ and the time interval I_{tar}, the number N_{tar} of all such target clusters is bounded by

$$N_{\text{tar}} \leq \#\mathcal{G}^{\ell_{\boldsymbol{x}}} = 8^{\ell_{\text{x}}}.$$

For each target cluster $Z_{\text{tar}} = X_{\text{tar}} \times I_{\text{tar}}$ we need to estimate the number of clusters in the M2Lx interaction list $\mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}})$ with temporal component I_{src}. This number of clusters is influenced by the truncation in space applied in the construction of the operation lists in Algorithm 7.2, which is determined at the level of the coarsest temporally indivisible ancestor $Z_{\text{tar}}^{\text{anc}} = X_{\text{tar}}^{\text{anc}} \times I_{\text{tar}}$ of Z_{tar} in $\mathcal{T}_\Sigma^{\text{ext}}$. In fact all relevant pairs of clusters $(Z_{\text{src}}, Z_{\text{tar}})$ with $Z_{\text{src}} \in \mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}})$ are obtained by recursive subdivisions of this cluster $Z_{\text{tar}}^{\text{anc}}$ and a suitable source cluster $Z_{\text{src}}^{\text{anc}} = X_{\text{src}}^{\text{anc}} \times I_{\text{src}}^{\text{anc}}$, whose spatial component $X_{\text{src}}^{\text{anc}}$ is contained in the interaction area $\mathcal{I}_{\mathcal{A}}(X_{\text{tar}}^{\text{anc}})$ due to the truncation in space executed in line 3 of Algorithm 7.2. The number of clusters in $\mathcal{I}_{\mathcal{A}}(X_{\text{tar}}^{\text{anc}})$ is bounded by $(2n_{\text{tr}}+1)^3$ in general, and is approximately equal to $(2n_{\text{tr}}+1)^2$ if the spatial clusters at this level are fine enough to resolve the surface Γ, which we assume for the sake of simplicity in the following. With this assumption, it also follows that each cluster $Z_{\text{src}}^{\text{anc}} = X_{\text{src}}^{\text{anc}} \times I_{\text{src}}^{\text{anc}}$ with $X_{\text{src}}^{\text{anc}} \in \mathcal{I}_{\mathcal{A}}(X_{\text{tar}}^{\text{anc}})$ has approximately $4^{d_{\boldsymbol{x}}}$ descendants with temporal component I_{src} and spatial level $\ell_{\boldsymbol{x}}$, where $d_{\boldsymbol{x}} = \ell_{\boldsymbol{x}} - \ell_{\boldsymbol{x}}(Z_{\text{tar}}^{\text{anc}})$ corresponds to the number of spatial refinements required to obtain these descendants. These are the clusters in $\mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}})$ with temporal component I_{src} and their number is approximately equal to $(2n_{\text{tr}}+1)^2 4^{d_{\boldsymbol{x}}}$. Hence, we conclude that the number of blocks in (7.20) satisfies

$$\#\mathcal{B}(I_{\text{src}}, I_{\text{tar}}) \approx N_{\text{tar}}(2n_{\text{tr}}+1)^2 4^{d_{\boldsymbol{x}}}.$$

If the blocks in (7.20) are stored and applied without any approximation the related storage and runtime complexity is directly proportional to the number of entries of these blocks. This number is

$$\mathcal{O}(\#\mathcal{B}(I_{\text{src}}, I_{\text{tar}}) n_t(I_{\text{src}}) n_{\ell_{\boldsymbol{x}}}^2), \tag{7.21}$$

where we denote the number of time steps contained in I_{src} by $n_t(I_{\text{src}})$ and the average number of spatial elements contained in a cluster with spatial level $\ell_{\boldsymbol{x}}$ by $n_{\ell_{\boldsymbol{x}}}$.

In the time-adaptive FMM the blocks in (7.20) are approximated by computing the moments of the involved source clusters, applying an M2Lx operation for each block,

and evaluating the resulting spatial local contributions. As we have seen in Table 7.1, the runtime complexity of a single M2Lx operation is $\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)(\rho_t+1))$, so the runtime complexity for the execution of the M2Lx operations for all blocks in (7.20) is

$$\mathcal{O}(\#\mathcal{B}(I_{\mathrm{src}}, I_{\mathrm{tar}})(m_{\boldsymbol{x}}+1)^4(m_t+1)(\rho_t+1)). \tag{7.22}$$

Due to the nested computation and evaluation of (spatial) moments and (spatial) local contributions in the time-adaptive FMM, the costs of these operations should not be estimated in a blockwise manner to avoid counting the same costs multiple times. On a global level all (spatial) moments and (spatial) local contributions of clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ are computed or evaluated only once in a nested way in the time-adaptive FMM. Furthermore, the costs of the newly introduced operations are similar to the costs of the standard FMM operations, as we have seen in Table 7.1. Therefore, one can show that the overall storage and runtime costs for the computation of all the moments by S2M, S2Mx, Mx2Mx, Mx2M, and M2M operations and evaluation of local contributions by L2L, L2Lx, Lx2Lx, Lx2T, and L2T operations scale linearly in the number of space-time elements $E_tE_{\boldsymbol{x}}$ as in the case of the standard space-time FMM or pFMM; see [67, Section 5.4] and [51, Section 4.5.3]. In particular, it is reasonable to focus on the costs of the M2Lx operations in (7.22) when comparing the costs of the application of the blocks in (7.20) in the time-adaptive FMM with the costs of their direct application in (7.21).

By comparing the costs in (7.21) and (7.22) we see that the approximation of the considered blocks in the time-adaptive FMM is more efficient than the direct application if

$$\mathcal{O}((m_{\boldsymbol{x}}+1)^4(m_t+1)(\rho_t+1)) < n_{\ell_{\boldsymbol{x}}}^2 n_t(I_{\mathrm{src}}).$$

While the costs on the left-hand side are constant if the expansion degrees m_t and $m_{\boldsymbol{x}}$ and the number of quadrature points ρ_t+1 is fixed, the costs on the right-hand side depend on the considered blocks. The number $n_t(I_{\mathrm{src}})$ of time steps in I_{src} influences the costs of the direct application, but if the considered temporal partition is not highly non-uniform, this influence is rather small. The influence of $n_{\ell_{\boldsymbol{x}}}$, i.e. the average number of spatial elements contained in the clusters with spatial level $\ell_{\boldsymbol{x}}$ in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ that are associated with the considered blocks, is more pronounced. If $\ell_{\boldsymbol{x}}$ is small, $n_{\ell_{\boldsymbol{x}}}$ may be large. In this case, the approximation in the time-adaptive FMM is significantly more efficient than the direct application, while for large spatial levels $\ell_{\boldsymbol{x}}$ and small numbers $n_{\ell_{\boldsymbol{x}}}$ the direct application might be faster. Note that the approximation of the blocks in the time-adaptive FMM may still be favorable in the latter case since the costly computation of the matrix entries in the assembly of the blocks in (7.20) is omitted and since the costs for storing the related moments and local contributions in the time-adaptive FMM are also significantly lower than the costs for storing the entries themselves in general. In the numerical examples in Section 7.5 we will observe these effects.

7.4 Parallelization of the time-adaptive FMM

The task based parallelization scheme for the space-time FMM from Chapter 6 can be adapted to obtain a parallel version of the time-adaptive FMM in Algorithm 7.5. The necessary modifications are described in this section. For the sake of simplicity, we focus on the shared memory parallelization and comment on the distributed memory parallelization only at the end of the section.

The starting point for the derivation of the parallel space-time FMM in Chapter 6 was the temporal version of the FMM in Algorithm 6.1. We obtained this version by associating space-time clusters in the underlying space-time tree $\mathcal{T}_\Sigma$ with temporal clusters using projections and defining FMM operations for these time clusters. For the time-adaptive FMM we follow the same approach. By applying the temporal projection Π_t in (6.1) to the extended space-time box cluster tree $\mathcal{T}_\Sigma^{\text{ext}}$ we obtain the same temporal tree $\mathcal{T}_I$ as for the related non-extended tree $\mathcal{T}_\Sigma$. For a time cluster $I \in \mathcal{T}_I$ we define the set of all associated space-time clusters in $\mathcal{T}_\Sigma^{\text{ext}}$ by

$$\mathcal{Z}_{\text{assoc}}^{\text{ext}}(I) = \{Z \in \mathcal{T}_\Sigma^{\text{ext}} : \Pi_t[Z] = I\}. \tag{7.23}$$

Note that a temporally indivisible cluster $Z = X \times I$ in $\mathcal{T}_\Sigma$ and all its spatially refined descendants in $\mathcal{T}_\Sigma^{\text{ext}}$ are associated with the same temporal cluster I. Hence, the set $\mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ of a temporally indivisible clusters I may contain entire subtrees of $\mathcal{T}_\Sigma^{\text{ext}}$ in general.

For a time cluster I_{tar} in $\mathcal{T}_I$ we define the operation lists related to the time-adaptive FMM as in Section 6.1 by projections. The M2L interaction list $\mathcal{I}_{\text{M2L}}(I_{\text{tar}})$ is defined by (6.3) and the nearfield $\mathcal{N}(I_{\text{tar}})$ by (6.4). In addition we define the M2Lx interaction list of I_{tar} by

$$\begin{aligned}\mathcal{I}_{\text{M2Lx}}(I_{\text{tar}}) := \{I_{\text{src}} \in \mathcal{T}_I : \exists Z_{\text{tar}} \in \mathcal{T}_\Sigma^{\text{ext}} \text{ and } Z_{\text{src}} \in \mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}}) \\ \text{with } \Pi_t[Z_{\text{src}}] = I_{\text{src}} \text{ and } \Pi_t[Z_{\text{tar}}] = I_{\text{tar}}\}\end{aligned} \tag{7.24}$$

and the Mx2L interaction list by

$$\begin{aligned}\mathcal{I}_{\text{Mx2L}}(I_{\text{tar}}) := \{I_{\text{src}} \in \mathcal{T}_I : \exists Z_{\text{tar}} \in \mathcal{T}_\Sigma^{\text{ext}} \text{ and } Z_{\text{src}} \in \mathcal{I}_{\text{Mx2L}}(Z_{\text{tar}}) \\ \text{with } \Pi_t[Z_{\text{src}}] = I_{\text{src}} \text{ and } \Pi_t[Z_{\text{tar}}] = I_{\text{tar}}\}.\end{aligned} \tag{7.25}$$

To obtain a temporal version of the time-adaptive FMM in Algorithm 7.5 we need to define corresponding operations in the temporal tree $\mathcal{T}_I$. While the operations of the standard FMM in Definition 7.7 can be reused, we need to introduce additional ones for the new operations in the time-adaptive FMM like the M2Lx and Mx2L operations and related sets of spatial moments and local contributions for clusters $I \in \mathcal{T}_I$. This is done in Definition 7.7 and Algorithm 7.7. Note that in the routines of Algorithm 7.7

we execute operations in a level-wise manner in the underlying extended space-time cluster tree $\mathcal{T}_\Sigma^{\text{ext}}$ to ensure a correct execution in the case that the set $\mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ of a cluster $I \in \mathcal{T}_I$ contains entire subtrees of $\mathcal{T}_\Sigma^{\text{ext}}$.

DEFINITION 7.7. *For a temporally indivisible time cluster I we define the set of spatial moments of its associated space-time clusters by*

$$\underline{\boldsymbol{\mu}}^{(\boldsymbol{x})}(I) := \{\boldsymbol{\mu}^{(\boldsymbol{x})}(Z) : Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)\} \tag{7.26}$$

and similarly the set of spatial local contributions by

$$\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I) := \{\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z) : Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)\}. \tag{7.27}$$

Furthermore, we define the following FMM operations for clusters in $\mathcal{T}_I$ in addition to the operations in Definition 6.2:

- *Extended S2M operations for temporally indivisible clusters I: Execute the required S2Mx, Mx2Mx and Mx2M operations for the clusters in $\mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ in a level-wise manner as outlined in Algorithm 7.7.*
- *M2Lx operations for I with $\mathcal{I}_{\text{M2Lx}}(I) \neq \emptyset$: For each $Z_{\text{tar}} \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ and each $Z_{\text{src}} \in \mathcal{I}_{\text{M2Lx}}(Z_{\text{tar}})$ compute $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\text{tar}}, Z_{\text{src}})$ by the M2Lx operation in (7.11) and add the result to the corresponding spatial local contributions in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$.*
- *Mx2L operations for I with $\mathcal{I}_{\text{Mx2L}}(I) \neq \emptyset$: For each $Z_{\text{tar}} \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ and each $Z_{\text{src}} \in \mathcal{I}_{\text{Mx2L}}(Z_{\text{tar}})$ compute $\boldsymbol{\lambda}(Z_{\text{tar}}, Z_{\text{src}})$ by the Mx2L operation in (7.14) and add the result to the corresponding local contributions in $\underline{\boldsymbol{\lambda}}(I)$.*
- *Extended L2T operations for temporally indivisible clusters I: Execute the required L2Lx, Lx2Lx and Lx2T operations for the clusters in $\mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ in a level-wise manner as outlined in Algorithm 7.7.*

With the newly defined FMM operations for clusters in $\mathcal{T}_I$ we can introduce a temporal version of the time-adaptive FMM in Algorithm 7.5 similar to Algorithm 6.1. Instead of explicitly describing this temporal version we continue with the description of the corresponding task based algorithm. The concept of this algorithm is the same as the one in Section 6.2: We group FMM operations of time clusters in $\mathcal{T}_I$ into FMM tasks and execute them based on individual dependencies using a task scheduler and a second level of OpenMP `tasks` for a better load distribution. For this purpose we define two new tasks for clusters in $\mathcal{T}_I$ in addition to the M-list, M2L-list, L-list and N-list tasks from Section 6.2:

- The M2Lx-list task for $I \in \mathcal{T}_I$. This task includes the M2Lx operations between the clusters I and $I_{\text{src}} \in \mathcal{I}_{\text{M2Lx}}(I)$ and in addition the extended L2T operations for I. It can be executed once the sets of moments $\underline{\boldsymbol{\mu}}(I_{\text{src}})$ of the clusters $I_{\text{src}} \in \mathcal{I}_{\text{M2Lx}}(I)$ are fully available.

Algorithm 7.7 Extended S2M and L2T operations in $\mathcal{T}_I$.

function ExtendedS2MOperations(I)
  Let $\ell_{\boldsymbol{x}}^{\min} := \min\{\ell_{\boldsymbol{x}}(Z) : Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)\}$ and $\ell_{\boldsymbol{x}}^{\max} := \max\{\ell_{\boldsymbol{x}}(Z) : Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)\}$.
  for all leaves $Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$
    S2Mx: Compute $\boldsymbol{\mu}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\mu}}^{(\boldsymbol{x})}(I)$ by (7.13).
  for $\ell_{\boldsymbol{x}} = \ell_{\boldsymbol{x}}^{\max}, \ldots, \ell_{\boldsymbol{x}}^{\min} - 1$
    for all $Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ with spatial level $\ell_{\boldsymbol{x}}$
      Mx2Mx: Compute $\boldsymbol{\mu}^{(\boldsymbol{x})}(\text{par}(Z), Z)$ by (7.15).
      Add $\boldsymbol{\mu}^{(\boldsymbol{x})}(\text{par}(Z), Z)$ to $\boldsymbol{\mu}^{(\boldsymbol{x})}(\text{par}(Z))$ in $\underline{\boldsymbol{\mu}}^{(\boldsymbol{x})}(I)$.
  for all $Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ with spatial level $\ell_{\boldsymbol{x}}^{\min}$
    Mx2M: Compute $\boldsymbol{\mu}(Z)$ in $\underline{\boldsymbol{\mu}}(I)$ by (7.16).
function ExtendedL2TOperations(I)
  Let $\ell_{\boldsymbol{x}}^{\min} := \min\{\ell_{\boldsymbol{x}}(Z) : Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)\}$ and $\ell_{\boldsymbol{x}}^{\max} := \max\{\ell_{\boldsymbol{x}}(Z) : Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)\}$.
  for all $Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ with spatial level $\ell_{\boldsymbol{x}}^{\min}$
    L2Lx: Add the result $\boldsymbol{\lambda}^{(\boldsymbol{x}),\text{trf}}(Z)$ from (7.17) to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$.
  for $\ell_{\boldsymbol{x}} = \ell_{\boldsymbol{x}}^{\min} + 1, \ldots, \ell_{\boldsymbol{x}}^{\max}$
    for all $Z \in \mathcal{Z}_{\text{assoc}}^{\text{ext}}(I)$ with spatial level $\ell_{\boldsymbol{x}}$
      Lx2Lx: Compute $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z, \text{par}(Z))$ by (7.19).
      Add $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z, \text{par}(Z))$ to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$.
  for all leaves Z in $\mathcal{T}_{\Sigma}^{\text{ext}}$
    Lx2T: Evaluate $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$ by (7.12). Add the result to $\boldsymbol{f}|_{\hat{Z}}$.

- The Mx2L-list task for $I \in \mathcal{T}_I$. This task includes the Mx2L operations between the clusters I and $I_{\text{src}} \in \mathcal{I}_{\text{Mx2L}}(I)$ and in addition the (extended) L2T operations for I, if the list of associated clusters $Z_{\text{assoc}}^{\text{ext}}(I)$ contains leaf clusters. It can be executed once the sets of spatial moments $\underline{\boldsymbol{\mu}}^{(\boldsymbol{x})}(I_{\text{src}})$ of the clusters $I_{\text{src}} \in \mathcal{I}_{\text{Mx2L}}(I)$ are fully available.

In the Mx2L-list task of a cluster I we execute extended L2T operations only if I is temporally indivisible. Otherwise, the standard L2T operations from Definition 6.2 are used. Also in the M2L-list tasks and L-list tasks of temporally indivisible clusters we need to replace the standard L2T operations with the extended L2T operations in Algorithm 7.7 and the standard S2M operations in the M-list tasks of temporally indivisible clusters with the extended S2M operations. Note that the (extended) L2T operations are now included in up to four different FMM tasks of a cluster I, namely the M2L-list, L-List, Mx2L-list, and M2Lx-list task of I, but may be executed only once. By specifying that the (extended) L2T operations are only executed in the last scheduled task we can execute these tasks in any order once their individual dependencies are satisfied, and still ensure that the corresponding (spatial) local contributions are fully computed before their evaluation.

In the parallel task based version of the time-adaptive FMM we need to execute M2Lx-list tasks for all temporally indivisible clusters I with non-empty M2Lx interaction lists $\mathcal{I}_{\mathrm{M2Lx}}(I)$ and Mx2L-list tasks for all clusters I with non-empty Mx2L interaction lists $\mathcal{I}_{\mathrm{Mx2L}}(I)$. To keep track of all these tasks we collect the corresponding clusters in the M2Lx-list and the Mx2L-list, respectively. The additional M-list, M2L-list, L-list and N-list can be constructed as described in Section 6.2 with two small modifications:

- In the recursive construction of the M-list, a cluster I should be added to this list not only if I is in the M2L interaction list $\mathcal{I}_{\mathrm{M2L}}(I_{\mathrm{tar}})$ of another cluster I_{tar} or if $\mathrm{par}(I)$ is in the M-list, but also if there exists a cluster I_{tar} such that $I \in \mathcal{I}_{\mathrm{Mx2L}}(I_{\mathrm{tar}})$ or $I \in \mathcal{I}_{\mathrm{M2Lx}}(I_{\mathrm{tar}})$.
- In the recursive construction of the L-list, a cluster I should be added to this list not only if $\mathcal{I}_{\mathrm{M2L}}(\mathrm{par}(I))$ is non-empty or $\mathrm{par}(I)$ is in the L-list, but also if $\mathcal{I}_{\mathrm{Mx2L}}(\mathrm{par}(I))$ is non-empty.

A sketch of the parallel task based version of the time-adaptive FMM is given in Algorithm 7.8. The main difference in comparison to the parallel version of the

Algorithm 7.8 Sketch of a parallel, task based version of the time-adaptive FMM.

1: Initialize $\boldsymbol{f} = \mathbf{0}$.
2: **for** all $I \in \mathcal{T}_I$
3: Initialize $\underline{\boldsymbol{\mu}}(I)$, $\underline{\boldsymbol{\mu}}^{(\boldsymbol{x})}(I)$, $\underline{\boldsymbol{\lambda}}(I)$ and $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$ by zeros as necessary.
4: Fill the `M_list`, `M2L_list`, `L_list`, `Mx2L_list`, `M2Lx_list` and `N_list`.
5: `#pragma omp parallel`
6: `#pragma omp single`
7: **while** the FMM task lists are not empty
8: [I, `list`] = FindNextReadyFMMTask(`M_list`, `L_list`, `M2L_list`, `Mx2L_list`, `M2Lx_list`)
9: **if** `list` $\in$ {`M_list, L_list, M2L_list`}
10: Proceed as in lines 9–20 of Algorithm 6.2.
11: **else if** `list == Mx2L_list`
12: Remove I from the `Mx2L_list`.
13: `#pragma omp task` *// + depend clause.*
14: Mx2LListTask(I)
15: **else if** `list == M2Lx_list`
16: Remove I from the `M2Lx_list`.
17: `#pragma omp task`
18: M2LxListTask(I)
19: **else** *// Schedule one of the remaining `N_list` tasks.*
20: Proceed as in lines 22–26 of Algorithm 6.2.

standard space-time FMM in Algorithm 6.2 is that the new Mx2L-list and M2Lx-lists need to be taken into account in the task scheduling procedure. Ready M-list, L-list, and M2L-list tasks are handled by creating corresponding OpenMP `tasks` in the same way as in Algorithm 6.2. The concrete tasks have to be slightly modified though, as we will discuss later. Also for ready M2Lx-list and Mx2L-list tasks we create corresponding OpenMP `tasks` in lines 11–18, while N-list tasks are only scheduled for execution if there are not any ready FMM tasks in the other five lists. Note that since the L-list, M2L-list and Mx2L-list tasks of a cluster I modify the same set of local contributions $\underline{\boldsymbol{\lambda}}(I)$ we use OpenMP `depend` clauses as discussed in Remark 6.4 to ensure that they are not executed contemporaneously.

The Mx2L-list task for a cluster $I \in \mathcal{T}_I$ is specified in Algorithm 7.9. In this task we execute the Mx2L operations of all space-time clusters associated with I in parallel using the OpenMP `taskloop` construct, see lines 2–5. This is similar to the execution of the M2L operations in the M2L-list task in Algorithm 6.4. The subsequent (extended) L2T operations in the Mx2L-list task are only executed if all other tasks for the cluster I from the M2L-list, L-list, and M2Lx-list have been executed already. We check this in lines 7 and 22, respectively, where we distinguish between temporally indivisible clusters and others.

For the parallelization of the extended L2T operations of a temporally indivisible cluster I in lines 8–20 of Algorithm 7.9 we make use of the OpenMP `taskloop` construct. First, the L2Lx and Lx2Lx operations are executed in parallel in a level-wise manner. As we already mentioned in Section 6.2 the `taskloop` construct introduces an implicit barrier that ensures that the spatial local contributions are computed in the correct level-wise order. The subsequent Lx2T operations are also parallelized by a `taskloop`. When adding the local vectors obtained by these Lx2T operations to the result vector $\boldsymbol{f}$, atomic operations are used to prevent race conditions between threads. If I is not temporally indivisible, the standard L2T operations are similarly parallelized in lines 22–25.

Algorithm 7.9 also includes a sketch of the M2Lx-list task of a cluster I. In this task we execute the M2Lx operations of all space-time clusters associated with I in parallel using the OpenMP `taskloop` construct; see lines 27–31. The subsequent extended L2T operations are handled exactly in the same way as in the Mx2L-list task.

Note that the M-list, L-list and M2L-list tasks have to be slightly adapted for the time-adaptive FMM. In the M2L-list and L-list tasks, the (extended) L2T operations have to be handled as in the Mx2L-list task in lines 7–25 of Algorithm 7.9. Also the M-list tasks need to be adapted to distinguish between temporally indivisible clusters and others. If a cluster I is not temporally indivisible, the standard S2M operations can be executed in parallel as in Algorithm 6.3. For temporally indivisible clusters, on the other hand, the extended S2M operations in Algorithm 7.7 need to be executed.

Algorithm 7.9 Mx2L-list and M2Lx-list tasks in Algorithm 7.8.

1: **function** Mx2LListTask(I)
    // Execution of the Mx2L operations for I:
2: `#pragma omp taskloop`
3: **for** all $Z_{\mathrm{tar}} \in \mathcal{Z}^{\mathrm{ext}}_{\mathrm{assoc}}(I)$
4: **for** all $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{Mx2L}}(Z_{\mathrm{tar}})$
5: Mx2L: Compute $\boldsymbol{\lambda}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ by (7.14) and add it to $\boldsymbol{\lambda}(Z_{\mathrm{tar}})$ in $\underline{\boldsymbol{\lambda}}(I)$.
6: **if** I is temporally indivisible
7: **if** $\underline{\boldsymbol{\lambda}}(I)$ is complete and potential M2Lx operations have been executed
// Extended L2T operations are executed only after potential
// L-list, M2L-list or M2Lx-list tasks.
8: Initialize $\ell^{\min}_{\boldsymbol{x}}$ and $\ell^{\max}_{\boldsymbol{x}}$ as in line 12 of Algorithm 7.7.
9: `#pragma omp taskloop`
10: **for** all $Z \in \mathcal{Z}^{\mathrm{ext}}_{\mathrm{assoc}}(I)$ with spatial level $\ell^{\min}_{\boldsymbol{x}}$
11: L2Lx: Add the result $\boldsymbol{\lambda}^{(\boldsymbol{x}),\mathrm{trf}}(Z)$ from (7.17) to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$.
12: **for** $\ell_{\boldsymbol{x}} = \ell^{\min}_{\boldsymbol{x}} + 1, \ldots, \ell^{\max}_{\boldsymbol{x}}$
13: `#pragma omp taskloop`
14: **for** all $Z \in \mathcal{Z}^{\mathrm{ext}}_{\mathrm{assoc}}(I)$ with spatial level $\ell_{\boldsymbol{x}}$
15: Lx2Lx: Compute $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z, \mathrm{par}(Z))$ by (7.19).
16: Add $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z, \mathrm{par}(Z))$ to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$.
17: `#pragma omp taskloop`
18: **for** all leaves $Z \in \mathcal{Z}^{\mathrm{ext}}_{\mathrm{assoc}}(I)$
19: Lx2T: Evaluate $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z)$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$ by (7.12).
20: Add the result to $\boldsymbol{f}|_{\hat{Z}}$ atomically.
21: **else if** $\mathcal{Z}_{\mathrm{assoc}}(I)$ contains leaves in $\mathcal{T}^{\mathrm{ext}}_{\Sigma}$
22: **if** $\underline{\boldsymbol{\lambda}}(I)$ is complete *// Only after potential M2L- and L-list operations.*
// Execution of the L2T operations for I:
23: `#pragma omp taskloop`
24: **for** all leaves $Z \in \mathcal{Z}^{\mathrm{ext}}_{\mathrm{assoc}}(I)$
25: L2T: Evaluate $\boldsymbol{\lambda}(Z)$ in $\underline{\boldsymbol{\lambda}}(I)$ by (5.40).
Add the result to $\boldsymbol{f}|_{\hat{Z}}$ atomically.

26: **function** M2LxListTask(I)
    // Execution of the M2Lx operations for I:
27: `#pragma omp taskloop`
28: **for** all $Z_{\mathrm{tar}} \in \mathcal{Z}^{\mathrm{ext}}_{\mathrm{assoc}}(I)$
29: **for** all $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{M2Lx}}(Z_{\mathrm{tar}})$
30: M2Lx: Compute $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ by (7.11).
31: Add $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}}, Z_{\mathrm{src}})$ to $\boldsymbol{\lambda}^{(\boldsymbol{x})}(Z_{\mathrm{tar}})$ in $\underline{\boldsymbol{\lambda}}^{(\boldsymbol{x})}(I)$.
32: Handle extended L2T operations as in lines 7–20 of the Mx2L-list task.

These extended S2M operations can be parallelized similarly to the extended L2T operations in Algorithm 7.9.

REMARK 7.8 (Priorities of Mx2L-and M2Lx tasks). *In Section 6.2.1 we have discussed that the load balance of the parallel, task based version of the space-time FMM can be improved by assigning priorities to the corresponding FMM tasks to influence the order in which they are executed. Based on the dependencies of the individual FMM tasks we have motivated that M-list tasks have the highest priority, followed by L-list tasks and M2L-list tasks, and that N-list tasks have the lowest priorities. For the prioritization of tasks in the parallel version of the time-adaptive FMM discussed in this section, we need to assign priorities to the new Mx2L-list and M2Lx-list tasks as well. Mx2L-list tasks are similar to M2L-list tasks. In fact, both modify the set of local contributions of a cluster and can be executed as soon as certain (spatial) moments are available. Therefore, we assign the same priority to Mx2L-list tasks as to M2L-list tasks, and prioritize Mx2L-list tasks for clusters on lower levels over Mx2L-list tasks for clusters on higher levels in the time tree. M2Lx-list tasks, on the other hand, are only used to update spatial local contributions of a cluster which are not required for the execution of any other tasks. Therefore, they have a similarly low priority as N-list tasks.*

REMARK 7.9 (Distributed memory parallelization). *The task based version of the time-adaptive FMM presented in this section can be extended as in Section 6.3 to allow for a parallel execution on distributed memory systems. In that section we distributed the clusters in the time tree $\mathcal{T}_I$ and the corresponding FMM tasks among all the available processes and handled the necessary inter-process communication in the execution of the FMM on the level of the time tree. The same can be done in the case of the time-adaptive FMM with the following modifications:*

- *The locally essential time trees and locally essential space-time trees of a process p introduced in Definitions 6.7 and 6.8 need to be extended such that the clusters contained in the Mx2L and M2Lx interaction lists of clusters assigned to p are contained in its locally essential trees.*

- *The inter-process communication has to be extended. Each process has to send the moments of clusters assigned to it also to those processes which need them for the execution of M2Lx operations. Likewise, a process has to send spatial moments of clusters assigned to it to those processes which need them for the execution of Mx2L operations.*

- *Instead of the process assignment strategy in Section 6.4 a more elaborated strategy is necessary to obtain a good load balance in the case of space-time meshes with adaptively refined temporal partitions.*

7.5 Numerical experiments

The time-adaptive FMM algorithm presented and analyzed in the previous sections has been implemented in the publicly available C++ library besthea [49] using the parallelization approach in Section 7.4. In this section we consider numerical experiments for space-time tensor product meshes with non-uniform time steps to show the benefits of the new time-adaptive FMM over the standard space-time FMM considered in Chapters 5 and 6. The results here correspond to those published in [78].

The following parameters are chosen for the standard and time-adaptive FMM in this section: When constructing a space-time cluster tree $\mathcal{T}_\Sigma$ by Algorithm 5.1 or an extended tree $\mathcal{T}_\Sigma^{\text{ext}}$ by Algorithm 7.1 we choose $n_{\max} = 800$ and $c_{\text{st}} = 4.5$. For the construction of the operation lists in Algorithms 5.2 and 7.2, respectively, we choose the truncation parameter $n_{\text{tr}} = 2$ and the parameters $\eta_1 = \eta_2 = 1$ for the admissibility criteria (5.12) and (7.5). Finally, we choose the expansion degrees $m_t = 4$ and $m_{\boldsymbol{x}} = 12$, and the parameter $\rho_t = 3$ for the numerical quadrature in (7.10), i.e. a four-point rule, which proves to be sufficiently accurate in our examples.

First experiment: Exponential decay in time. In the first experiment we use the direct boundary integral approach from Section 3.2 to solve an initial Dirichlet boundary value problem (1.1)–(1.3) for the heat equation. We consider the space-time cylinder $Q = \Omega \times (0, T)$ with the spatial domain $\Omega = (-0.5, 0.5)^3$, the time horizon $T = 0.25$, the heat capacity constant $\alpha = 1$, the Dirichlet datum $g = 0$, and the initial datum u_0 chosen such that the exact solution is given by

$$u(\boldsymbol{x}, t) = \exp(-3\pi^2 t) \prod_{j=1}^{3} \sin(\pi(x_j + 0.5)), \quad \text{for all } (\boldsymbol{x}, t) \in Q. \tag{7.28}$$

Our goal is to compare the standard space-time FMM and the new time-adaptive FMM for the application of V_h when solving the linear system (3.24) to obtain an approximation q_h of the Neumann datum $q = \gamma_{1,\Sigma}^{\text{int}} u$. To set up this system, a discretization Σ_h of the lateral surface Σ of Q is needed and in addition a discretization Ω_h of the domain Ω due to the inhomogeneous initial condition. We use an admissible volume mesh Ω_h consisting of 196 608 similarly sized tetrahedra and a space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$, where Γ_h is the conforming surface mesh corresponding to Ω_h and contains 12 288 uniform triangles. To account for the exponential decay in time of the solution u in (7.28) we choose the non-uniform partition $\mathcal{I}_{h_t}$ illustrated in Figure 7.2 which consists of 20 intervals whose sizes increase as time advances. We constructed this partition $\mathcal{I}_{h_t}$ by using an adaptive refinement algorithm that was tailored to reduce the L^2 projection error of the temporal part $q_t(t) = \exp(-3\pi^2 t)$ of the exact solution (7.28) for piecewise constant basis functions. The resulting space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ consists of 245 760 space-time boundary elements.

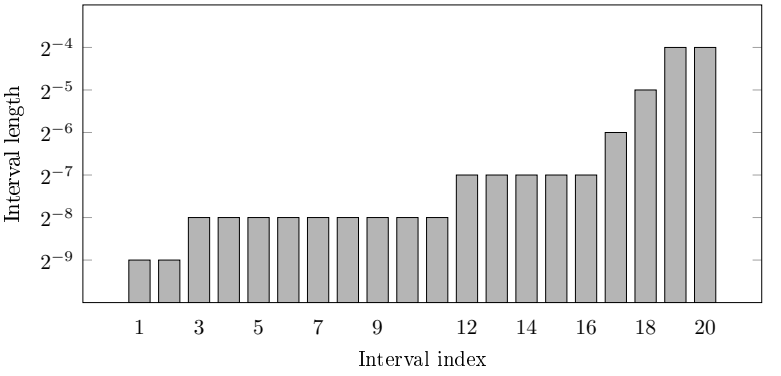


Figure 7.2: The time mesh $\mathcal{I}_{h_t}$ consisting of 20 time steps in the time interval $(0, 0.25)$ adapted to $q_t(t) = \exp(-3\pi^2 t)$. For each interval its length is depicted.

For the computation of the right-hand side of the linear system (3.24) we assemble the initial operator matrix M_h^0 as outlined in Section 3.3. The system is solved by using the GMRES method with a diagonal preconditioner and a desired relative accuracy of 10^{-8}. The computations were carried out on the local workstation Babbage at the Institute of Applied Mathematics at TU Graz; see Appendix A for the hardware details. The results are presented in Table 7.2.

	Standard FMM	Time-adaptive FMM
Storage nearfield V_h [GiB]	39.9	32.0
Setup time V_h $[s]$	307.3	217.6
GMRES it. time $[s]$	1.36	1.42
No. GMRES iterations	49	49
Total time $[s]$	374.0	286.9
Rel. L^2 error BEM	0.0530951	0.0530951

Table 7.2: Results of the first experiment of Section 7.5 where the linear system (3.24) is solved for a space-time tensor product mesh consisting of the non-uniform partition of the time interval $(0, 0.25)$ depicted in Figure 7.2 and a uniform spatial mesh on the surface of the cube $\Omega = (-0, 5.0.5)^3$ consisting of 12 288 triangles. The listed total times include the setup times for V_h and the times for the solution of the considered linear system, but not the required times for the construction of the right-hand side of the system, which takes about 250 seconds when using a suitable fast method for the application of M_h^0.

Let us first compare the storage requirements of the standard and time-adaptive FMM. In the second line of Table 7.2 we list the memory required to store the inadmissible nearfield blocks of V_h in these two fast methods. Here we see that the memory demand is reduced by roughly 20 percent for the time-adaptive FMM. Note that only 21.6 MiB are needed in this particular example to store all the moments and local contributions in the case of the standard FMM and only 22.1 MiB in the case of the time-adaptive FMM with the additional spatial local contributions, so the memory required to store the inadmissible blocks is by far the dominating part.

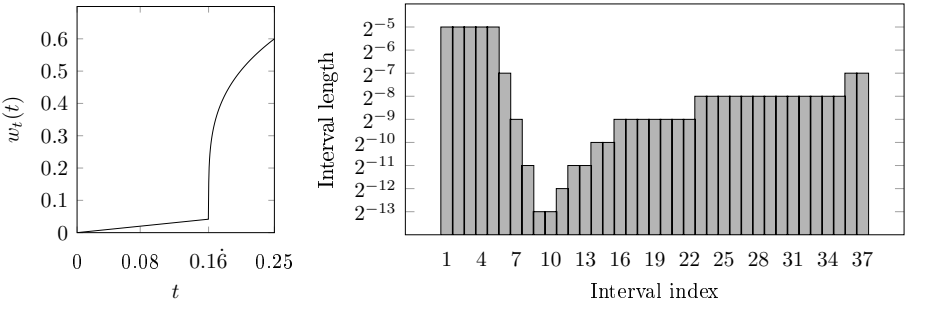
In lines 3–6 of Table 7.2 we compare the execution times when assembling and solving the system (3.24) using the standard and time-adaptive FMM. In both cases, the GMRES method requires 49 iterations to achieve a relative accuracy of 10^{-8}. The time for a single GMRES iteration, which is dominated by the time needed for the application of V_h, is slightly lower for the standard FMM in this example. This is not completely unexpected, since the application of inadmissible dense matrix blocks is simple and efficient if they are already computed. The newly introduced Mx2L and M2Lx operations are more efficient than the related nearfield operations for source and target clusters containing many space-time boundary elements but can be less efficient for clusters containing few elements, as we discussed in Section 7.3. The corresponding gains and losses seem to balance out in this example. However, the assembly of the inadmissible blocks requires considerably more time than the GMRES solver and is significantly faster in the time-adaptive version, due to the reduced number of nearfield entries which have to be computed. Therefore, the new time-adaptive FMM outperforms the standard version in this example.

To compare the accuracy of the standard and time-adaptive FMM we consider the relative L^2 approximation errors $\|q - q_h\|_{L^2(\Sigma)}/\|q\|_{L^2(\Sigma)}$ between the known Neumann datum q and the approximate solutions q_h obtained by solving (3.24) in the last line of Table 7.2. These values do not differ in the specified six significant digits. Furthermore, they are close to the relative L^2 best approximation error of the Neumann datum which is 0.0467157. Therefore, we conclude that the new time-adaptive FMM operations are sufficiently accurate and that the time-adaptive FMM is the better option in this particular example.

Second experiment: Rapidly changing boundary data. In the second experiment we consider a boundary value problem where the boundary data change rapidly over time. For this purpose we consider the function

$$w_t(t) = \frac{t}{4} + \left(t - \frac{1}{6}\right)^{1/4} \mathbb{1}_{(1/6,\infty)}(t), \quad t \in (0,T), \tag{7.29}$$

where $T = 0.25$. The function w_t is depicted in Figure 7.3a. It is non-smooth at time $t = 1/6$ when the part $(t - 1/6)^{1/4}$ is "turned on" by the indicator function $\mathbb{1}_{(1/6,\infty)}$.



a) w_t from (7.29). b) Interval lengths of the adaptive mesh $\mathcal{I}_{h_t}$ for w_t in (7.29).

Figure 7.3: Visualization of the rapidly changing function w_t in (7.29) in the time interval $(0, 0.25)$ and a suitable mesh $\mathcal{I}_{h_t}$ on this interval that resolves the non-smooth behavior of w_t at time $t = 1/6$ well.

We set

$$w(\boldsymbol{x},t) = w_t(t) w_{\boldsymbol{x}}(\boldsymbol{x}), \text{ for all } (\boldsymbol{x},t) \in \Gamma \times (0,T), \quad w_{\boldsymbol{x}}(\boldsymbol{x}) = \frac{\boldsymbol{n}(\boldsymbol{x}) \cdot (2,-3,1)^\top}{10}, \tag{7.30}$$

where Γ is the surface of the crankshaft domain Ω from Section 6.5.2 depicted in Figure 6.6, and $\boldsymbol{n}$ is the outer unit normal vector on Γ. Applying the single layer potential operator $\widetilde{V}$ from (3.2) to w yields a solution of the heat equation (1.1) in the space-time cylinder $Q = \Omega \times (0,T)$ with homogeneous initial conditions and Dirichlet trace $g := Vw$. Therefore, we can regard w as the density that is obtained when solving the initial Dirichlet boundary value problem (1.1)–(1.3) for the heat equation with initial datum $u_0 = 0$ and Dirichlet datum g using the first indirect boundary integral approach described in Section 3.3. In the following, we discretize this boundary integral equation and solve the corresponding linear system

$$\mathsf{V}_h \boldsymbol{w} = \boldsymbol{g}. \tag{7.31}$$

For the discretization we use the space-time tensor product mesh $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ where Γ_h is a surface mesh of the crankshaft geometry consisting of $E_{\boldsymbol{x}} = 10\,722$ triangles and $\mathcal{I}_{h_t}$ is the non-uniform partition of the time interval $(0,T)$ depicted in Figure 7.3b consisting of $E_t = 37$ subintervals. As in the previous experiment, we constructed this partition $\mathcal{I}_{h_t}$ by using an adaptive refinement algorithm to reduce the L^2 projection error of w_t for piecewise constant basis functions. Note that $\mathcal{I}_{h_t}$ is highly adaptive and that the intervals with the smallest lengths are those close to the time $t = 1/6$.

Since an analytic form of the Dirichlet datum g is not known, we compute the right-hand side $\boldsymbol{g}$ of (7.31) as follows: We uniformly refine the mesh Σ_h once with respect

to space and twice with respect to time to obtain the mesh

$$\hat{\Sigma}_h = \{\hat{\sigma}_{k_t,k_{\boldsymbol{x}}} = \hat{\gamma}_{k_{\boldsymbol{x}}} \times (\hat{t}_{k_t-1}, \hat{t}_{k_t}) : k_t \in \{1,\dots,4E_t\}, k_{\boldsymbol{x}} \in \{1,\dots,4E_{\boldsymbol{x}}\}\}.$$

By applying the single layer matrix $\hat{\mathsf{V}}_h$ for this mesh $\hat{\Sigma}_h$ to the coefficients $\hat{\boldsymbol{w}}$ of the L^2 projection $\hat{w}_h \in S^{0\times 0}_{h_x,h_t}(\hat{\Sigma}_h)$ of w we obtain the coefficients $\hat{\boldsymbol{g}} = \hat{\mathsf{V}}_h\hat{\boldsymbol{w}}$ which we restrict to the mesh Σ_h by setting

$$g_{j_t,j_{\boldsymbol{x}}} = \sum_{(k_t,k_{\boldsymbol{x}})\in \mathrm{child}(j_t,j_{\boldsymbol{x}})} \hat{g}_{k_t,k_{\boldsymbol{x}}}, \qquad \mathrm{child}(j_t,j_{\boldsymbol{x}}) := \{(k_t,k_{\boldsymbol{x}}) : \hat{\sigma}_{k_t,k_{\boldsymbol{x}}} \subset \sigma_{j_t,j_{\boldsymbol{x}}}\}$$

for all index pairs $(j_t, j_{\boldsymbol{x}})$ corresponding to space-time elements $\sigma_{j_t,j_{\boldsymbol{x}}} \in \Sigma_h$ to get the vector $\boldsymbol{g}$.

We solve the linear system (7.31) by using the GMRES method with a diagonal preconditioner and a desired relative accuracy of 10^{-8}, and compare the resulting runtimes, storage requirements, and approximation errors when using the standard FMM and the time-adaptive FMM for the application of V_h. The computations were carried out using a single node of the VSC-4 cluster: a standard node for the time-adaptive FMM and a fat node for the standard FMM due to the higher memory demand; see Appendix A for the hardware details. The results are presented in Table 7.3.

	Standard FMM	Time-adaptive FMM
Storage nearfield V_h [GiB]	93.2	66.6
Setup time V_h $[s]$	495.3	302.3
GMRES it. time $[s]$	1.27	1.16
No. GMRES iterations	98	98
Total time $[s]$	619.4	415.6
Rel. L^2 error BEM	0.013129	0.013129

Table 7.3: Results of the solution of the linear system (7.31) for a space-time tensor product mesh consisting of the non-uniform partition of the time interval $(0, 0.25)$ depicted in Figure 7.3b and a mesh on the surface of the crankshaft geometry Ω depicted in Figure 6.6 with 10 722 triangles.

The difference in performance between the standard and the time-adaptive FMM is slightly more pronounced in this experiment than in the previous one. The memory required to store the inadmissible blocks of V_h is reduced by almost 29 percent. Note that the moments and local contributions require only 28.8 MiB of storage in the standard FMM and the additional spatial moments and local contributions in the case of the time-adaptive FMM require only 2.4 MiB. Due to the smaller

number of inadmissible blocks, the assembly time is significantly lower for the time-adaptive FMM. In contrast to the previous experiment, also the application times and the related GMRES iteration times are lower when the time-adaptive FMM is used instead of the standard FMM. Since 98 GMRES iterations were needed to solve the system in both cases, we end up with the total times in the sixth row of Table 7.3 for the assembly of V_h and the solution of the system (7.31). As in the previous experiment we see that the time-adaptive FMM outperforms the standard FMM, while the approximation quality does not suffer when introducing the new FMM operations.

Finally, we want to compare the results in Table 7.3 with results obtained by solving the same boundary integral equation on a suitable temporally uniform mesh. For this purpose we consider the space-time mesh $\Sigma_{h,u} = \Gamma_h \otimes \mathcal{I}_{h_t,\mathrm{u}}$, where Γ_h is the same crankshaft surface mesh as above and $\mathcal{I}_{h_t,\mathrm{u}}$ is the uniform time mesh on $(0,T)$ consisting of 512 uniform time steps with length 2^{-11}. Note that $\Sigma_{h,u}$ contains 5 489 664 space-time boundary elements while the time-adaptive mesh Σ_h above contains only 396 714 space-time elements. The large number of uniform time steps in $\mathcal{I}_{h,\mathrm{u}}$ is required to resolve the rapid change of w_t around the point $t = 1/6$ reasonably well. The mesh $\mathcal{I}_{h_t,\mathrm{u}}$ is slightly coarser than the adaptive mesh $\mathcal{I}_{h_t}$ in Figure 7.3b around $t = 1/6$ but considerably finer otherwise. The L^2 approximation errors obtained by projecting the function w_t to the spaces of piecewise constant functions for these two temporal meshes are similar, but the error is slightly larger around time $t = 1/6$ for the uniform mesh.

When we discretize the boundary integral equation $Vw = g$ using the uniform space-time mesh $\Sigma_{h,\mathrm{u}}$ we obtain the system

$$\mathsf{V}_{h,\mathrm{u}} \boldsymbol{w}_{\mathrm{u}} = \boldsymbol{g}_{\mathrm{u}}, \tag{7.32}$$

where we compute the right-hand side $\boldsymbol{g}_{\mathrm{u}}$ similarly as the right-hand side $\boldsymbol{g}$ of (7.31) above. We solve this system by using the GMRES method with a diagonal preconditioner and the standard FMM for the application of $\mathsf{V}_{h,\mathrm{u}}$. Note that using the time-adaptive FMM would not bring any benefits here due to the temporal uniformity of the mesh $\Sigma_{h,\mathrm{u}}$. For the computations we used 16 standard nodes of the VSC-4 cluster due to the large size of the problem. The results are given in Table 7.4.

Let us compare the results in Table 7.4 with the results in Table 7.3 when the time-adaptive FMM is used for the approximation of the matrix V_h. Even though we use 16 nodes for the computations on the temporally uniform mesh, more memory per node is needed to store the inadmissible blocks of $\mathsf{V}_{h,\mathrm{u}}$ than for storing all the inadmissible blocks of the matrix V_h for the temporally adaptive mesh on a single node. However, this comparison is slightly unfair, since we do not exploit the temporal uniformity of the mesh $\mathsf{V}_{h,\mathrm{u}}$ in our implementation and, therefore, we store identical blocks of $\mathsf{V}_{h,\mathrm{u}}$ multiple times. For the same reason, a direct comparison of the setup times is not very

meaningful. On the other hand, a comparison of the application times or GMRES iteration times is reasonable since the uniformity of the mesh cannot be exploited in a similar way in the application of the matrices. In the case of the temporally uniform mesh, we use the sixteenfold hardware power for the computations, but nonetheless, the time for a single GMRES iteration is about twice as long as for the computations on the temporally adaptive mesh. Due to this fact and the larger number of required GMRES iterations, the solution of the system (7.32) for the temporally uniform mesh takes about 350 seconds instead of 113 seconds for the temporally adaptive mesh despite the additional parallelization. At the same time, the approximation quality of the resulting BEM solutions is comparable. This shows that the temporally adaptive mesh is far better suited for the solution of the considered boundary integral equation with rapidly changing boundary data.

	Standard FMM (16 nodes)
Storage nearfield $\mathsf{V}_{h,\mathrm{u}}$ (average per node) [GiB]	79.90
Setup time $\mathsf{V}_{h,\mathrm{u}}$ $[s]$	406.12
GMRES it. time $[s]$	2.29
No. GMRES iterations	153
Total time $[s]$	756.40
Rel. L^2 error BEM	0.011851

Table 7.4: Results of the solution of the system (7.32) for the mesh $\Sigma_{h,\mathrm{u}}$ consisting of 512 uniform time steps in the interval $(0, 0.25)$ and 10 722 triangles on the surface of the crankshaft domain Ω.

In summary, we have seen that the new time-adaptive FMM provides a better matrix compression for temporally non-uniform meshes than the standard space-time FMM, which helps to reduce the storage requirements and runtimes when solving related boundary integral equations. The only difference between the two methods is the treatment of temporally one-sided admissible blocks, which are fully assembled in the case of the standard FMM and approximated in the case of the time-adaptive FMM. Hence, the time-adaptive FMM can be seen as a valuable extension of the standard FMM. Since the storage requirements — or, equivalently, the number of inadmissible blocks corresponding to pairs of clusters that are not temporally separated — can still be quite large in the case of the time-adaptive FMM, we present a method for the additional compression of this temporal nearfield in the next chapter.

8 An ACA based nearfield compression scheme

At the end of Chapter 5 we have seen that the space-time FMM can significantly reduce the runtime complexity of the application of BEM matrices like the single layer operator matrix V_h: For a space-time tensor product mesh with E_t similarly sized time steps and $E_{\boldsymbol{x}}$ similarly sized spatial elements and lowest order test and trial spaces the costs are reduced from $\mathcal{O}((E_t E_{\boldsymbol{x}})^2)$ to $\mathcal{O}(E_t E_{\boldsymbol{x}} m_t^2 m_{\boldsymbol{x}}^4)$, where m_t and $m_{\boldsymbol{x}}$ are the expansion degrees of the FMM operations. A requirement for this reduction is that the spatial and temporal mesh widths $h_{\boldsymbol{x}}$ and h_t satisfy the relation $h_t \sim h_{\boldsymbol{x}}^2$. If $h_{\boldsymbol{x}}$ is too small in relation to h_t the efficiency of the space-time FMM suffers and costs of order $\mathcal{O}(E_t E_{\boldsymbol{x}}^2)$ are to be expected in the worst case. This is because the approximations of the heat kernel in (5.1) and (7.1), on which the efficient approximation of blocks of V_h is based, rely on a separation in time of the clusters corresponding to the blocks. If the number of time steps of the considered mesh is too small, there exist large blocks of V_h whose clusters are not separated in time. These large temporal nearfield blocks are not approximated in the standard space-time FMM and are responsible for its bad performance.

The suboptimal performance of the space-time FMM for meshes with too fine spatial mesh resolutions has already been discussed in [52] for the related pFMM. To overcome this issue an additional fast method for the application of the temporal nearfield has been described in that work. In that fast method, the integrals of the entries of inadmissible blocks of V_h — or other BEM matrices — are transformed and subdivided in time in a sophisticated way to obtain two parts: A part of singular integrals, which are sufficiently localized in space to allow for an efficient direct evaluation, and another part of regular integrals, which are efficiently treated by combining numerical quadrature in time with fast Gauß transforms in space based on truncated Chebyshev expansions as in the pFMM. In the numerical examples in [52, Section 5] it was shown that the linear complexity of the pFMM can be reestablished up to a logarithmic factor for meshes with fine spatial mesh resolutions by using the proposed additional fast method for the application of the temporal nearfield.

In this chapter we present an alternative method for the compression of the temporal nearfield in the space-time FMM. Our approach is based on the partially pivoted adaptive cross approximation (ACA) which is a method that allows one to compute a low-rank approximation of a given matrix in a pure algebraic way using only few of the original matrix entries and is described, for example, in [10, 13]. The partially pivoted ACA can be used instead of kernel approximations in BEM to approximate

suitable admissible blocks of the corresponding system matrices with an overall quasi-linear complexity in the number of rows and columns of the considered matrix. This is described in more detail, for example, in [62, Chapter 3] or [11]. In the context of the heat equation, a fast method alternative to the pFMM and the related space-time FMM in this thesis was developed in [74] where admissible blocks of a BEM matrix like V_h corresponding to temporally separated clusters are approximated using a kernel expansion in time and the ACA in space. Our approach in this chapter is different: Starting from the space-time FMM in Chapter 7 we determine suitable inadmissible blocks of V_h which can be approximated by the ACA — if necessary by additional block refinements. The partially pivoted ACA is then directly applied to these blocks. The ranks of the resulting low-rank matrices can be further reduced by using an additional truncated singular value decomposition. This kind of recompression is well known from the literature — see e.g. [12, Section 2.2] — but we propose a new criterion to determine the related recompression ranks. The resulting nearfield compression scheme is similarly efficient but easier to implement than the nearfield compression scheme in [52] since it requires only an implementation of the rather simple partially pivoted ACA algorithm — and algorithms to compute a singular value decomposition of a matrix available in standard math libraries like LAPACK [6] — while the method in [52] introduces additional refinements of time steps and new integrals to evaluate in a spatial multi-level setting. Furthermore, the method in [52] relies on the uniform prismatic structure of the elements of the considered space-time mesh, while the black-box nature of the ACA might allow us to use our new nearfield compression scheme also for more general space-time meshes.

The rest of this chapter is structured as follows. In Section 8.1 we give a basic description of the partially pivoted ACA before motivating in Section 8.2 why it can be used to approximate certain originally inadmissible blocks of the matrix V_h. By identifying and approximating these blocks in the context of the space-time FMM from Chapter 7, we obtain the new nearfield compression scheme mentioned in the previous paragraph, which is described in Section 8.3. In Section 8.4 we describe the additional recompression strategy and in Section 8.5 we analyze its effect on the approximation quality of V_h. The chapter is concluded by Section 8.6 where we present various numerical experiments revealing the positive effects of the new nearfield compression scheme with and without the additional recompression.

8.1 The adaptive cross approximation

The space-time FMM from Chapter 5 and its extension from Chapter 7 rely on the fact that suitable blocks of BEM matrices like V_h can be approximated and applied in an efficient way by suitable FMM operations. These approximations of matrix blocks can be understood as approximations by low-rank matrices. In linear algebra, the

rank of a matrix $\mathsf{A} \in \mathbb{R}^{m\times n}$ is defined as the dimension of the image space of A and is bounded by $\min\{m,n\}$. A is called a low-rank matrix, if its rank k is significantly smaller than $\min\{m,n\}$. It is well known, that such a low-rank matrix A admits a representation

$$\mathsf{A} = \mathsf{U}\mathsf{W}^\top = \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{w}_i^\top, \tag{8.1}$$

where $\mathsf{U} \in \mathbb{R}^{m\times k}$ and $\mathsf{W} \in \mathbb{R}^{n\times k}$ are suitable matrices, and $\{\boldsymbol{u}_i\}_{i=1}^k$ and $\{\boldsymbol{w}_i\}_{i=1}^k$ are their columns; see [11, Theorem 1.2]. The low-rank approximations of matrix blocks in Chapters 5 and 7 have been constructed by replacing the heat kernel in the integrals defining the matrix entries by the separable approximations (5.9), (7.1) and (7.3) of the heat kernel. An alternative method that yields such low-rank approximations is the adaptive cross approximation.

The adaptive cross approximation (ACA) has been introduced in [10, 13]. It allows one to construct a low-rank approximation of a given matrix in a purely algebraic way by using only the original matrix entries. For certain matrices — like suitable blocks of BEM matrices generated by asymptotically smooth kernels — this is even possible without assembling the full matrix in advance. Instead, relevant matrix entries can be computed on demand during the approximation. The resulting method is denoted as partially pivoted ACA and is sketched in Algorithm 8.1. This version is based on the one in [11, Section 3.4.1, Algorithm 3.1].

In the ACA in Algorithm 8.1 a matrix $\mathsf{A} \in \mathbb{R}^{m\times n}$ is approximated by generating a sequence of low-rank matrices $\{\mathsf{S}_k\}_k$ of the form $\mathsf{S}_k = \mathsf{U}_k\mathsf{W}_k^\top$. In the first step, a suitable index pair (i_1, j_1) is chosen and the vectors $\boldsymbol{w}_1 \in \mathbb{R}^n$ and $\boldsymbol{u}_1 \in \mathbb{R}^m$ are constructed as $\boldsymbol{w}_1 = \mathsf{A}[i_1, j_1]^{-1}\mathsf{A}[i_1, :]$ and $\boldsymbol{u}_1 = \mathsf{A}[:, j_1]$, i.e. $\boldsymbol{w}_1$ is the i_1^{th} row of A scaled by $\mathsf{A}[i_1, j_1]^{-1}$ and $\boldsymbol{u}_1$ is the j_1^{th} column of A. This yields the first approximation $\mathsf{S}_1 = \boldsymbol{u}_1\boldsymbol{w}_1^\top$ of A. The residuum $\mathsf{R}_1 = \mathsf{A} - \mathsf{S}_1$ is a matrix whose i_1^{th} row and j_1^{th} column vanish. The approximations $\{\mathsf{S}_k\}_k$ are then constructed iteratively: Once S_k is computed, a suitable index pair (i_{k+1}, j_{k+1}) is determined and new vectors $\boldsymbol{w}_{k+1} \in \mathbb{R}^n$ and $\boldsymbol{u}_{k+1} \in \mathbb{R}^m$ are created by setting $\boldsymbol{w}_{k+1} = \mathsf{R}_k[i_{k+1}, j_{k+1}]^{-1}\mathsf{R}_k[i_{k+1}, :]$ and $\boldsymbol{u}_{k+1} = \mathsf{R}_k[:, j_{k+1}]$, where $\mathsf{R}_k = \mathsf{A} - \mathsf{S}_k$; see lines 5–7 and 11 in Algorithm 8.1 for $\boldsymbol{w}_{k+1}$ and lines 12–14 for $\boldsymbol{u}_{k+1}$. This yields the next approximation $\mathsf{S}_{k+1} = \sum_{\ell=1}^{k+1} \boldsymbol{u}_\ell\boldsymbol{w}_\ell^\top$ and the residuum $\mathsf{R}_{k+1} = \mathsf{A} - \mathsf{S}_{k+1}$ which contains zeros in all rows $i \in \{i_\ell\}_{\ell=1}^{k+1}$ and columns $j \in \{j_\ell\}_{\ell=1}^{k+1}$. Note that once S_k is computed only the i_{k+1}^{th} row and j_{k+1}^{th} column of A are needed to compute the required entries of R_{k+1} and the update S_{k+1}. The pairs of rows and columns that are constructed in each step can be seen as crosses, which explains the name of the method.

The choice of the indices i_{k+1} and j_{k+1} for the construction of the matrix S_{k+1} from S_k in the ACA is crucial for the efficiency and correctness of the method. From the construction described in the last paragraph it is clear that the index i_{k+1}

should be chosen from the set $\{1,\dots,m\}\setminus\{i_1,\dots,i_k\}$ and the index j_{k+1} from the set $\{1,\dots n\}\setminus\{j_1,\dots,j_k\}$. Once i_{k+1} is chosen and the unscaled i_{k+1}^{th} row $\widetilde{\boldsymbol{w}}_{k+1}$ of the residual matrix R_k has been computed, the index j_{k+1} is chosen such that $|(\widetilde{\boldsymbol{w}}_{k+1})_{j_{k+1}}| = \max_j\{|(\widetilde{\boldsymbol{w}}_{k+1})_j|\}$; see line 10 in Algorithm 8.1. This choice is motivated in [11, Lemma 3.34]. Note that it is possible that $\widetilde{\boldsymbol{w}}_{k+1} = \boldsymbol{0}$. This is not bad, since it means that an additional row of the matrix R_k is zero and, therefore, an additional row of A is already approximated well. However, it also means that we need to choose a different row index i_{k+1} to determine the vector $\boldsymbol{w}_{k+1}$ for the matrix S_{k+1}. We keep track of all previously selected row indices in the set $\mathcal{R}$ in line 8 of Algorithm 8.1. A new index i_{k+1} is then chosen in line 20 such that $|(\boldsymbol{u}_k)_{i_{k+1}}|$ is maximal among all values $|(\boldsymbol{u}_k)_i|$ with $i\notin\mathcal{R}$; similarly as in [62].

Algorithm 8.1 Partially pivoted adaptive cross approximation (ACA) of $\mathsf{A}\in\mathbb{R}^{m\times n}$.

Require: Let a parameter ε_{ACA} for the stopping criterion (8.3) be given.
Let a bound $k_{\max}$ for the maximal rank of the approximation be given.

1: Initialize $k=0$ and the row index set $\mathcal{R}=\emptyset$.
2: Initialize the row index $i_{k+1}=1$.
3: Initialize the squared Frobenius norm n_{frob}^2 of the approximation by $n_{\text{frob}}^2=0$.
4: **do**
5: Set $\boldsymbol{w}_{k+1}=\mathsf{A}[i_{k+1},:]^\top$.
6: **for** all $\ell=1,\dots,k$
7: Update $\boldsymbol{w}_{k+1}\leftarrow\boldsymbol{w}_{k+1}-(\boldsymbol{u}_\ell)_{i_{k+1}}\boldsymbol{w}_\ell$.
8: Update $\mathcal{R}\leftarrow\mathcal{R}\cup\{i_{k+1}\}$.
9: **if** $\boldsymbol{w}_{k+1}$ does not vanish:
10: Set $j_{k+1}=\text{argmax}_{j=1,\dots,n}\{|(\boldsymbol{w}_{k+1})_j|\}$.
11: Update $\boldsymbol{w}_{k+1}\leftarrow(\boldsymbol{w}_{k+1})_{j_{k+1}}^{-1}\boldsymbol{w}_{k+1}$.
12: Set $\boldsymbol{u}_{k+1}=\mathsf{A}[:,j_{k+1}]$.
13: **for** $\ell=1,\dots,k$
14: Update $\boldsymbol{u}_{k+1}\leftarrow\boldsymbol{u}_{k+1}-(\boldsymbol{w}_\ell)_{j_{k+1}}\boldsymbol{u}_\ell$.
// Update the squared Frobenius norm n_{frob}^2 of the approximation.
15: **for** $\ell=1,\dots,k$
16: Update $n_{\text{frob}}^2\leftarrow n_{\text{frob}}^2+2(\boldsymbol{u}_\ell\cdot\boldsymbol{u}_{k+1})(\boldsymbol{w}_\ell\cdot\boldsymbol{w}_{k+1})$.
17: Update $n_{\text{frob}}^2\leftarrow n_{\text{frob}}^2+|\boldsymbol{u}_{k+1}|^2|\boldsymbol{w}_{k+1}|^2$.
18: Update $k\leftarrow k+1$.
19: **if** $\mathcal{R}\neq\{1,\dots,m\}$
20: Set $i_{k+1}=\text{argmax}_{i\in\{1,\dots,m\}\setminus\mathcal{R}}\{|(\boldsymbol{u}_k)_i|\}$.
21: **while** (8.3) is violated **and** $k<\min\{k_{\max},n\}$ **and** $\mathcal{R}\neq\{1,\dots,m\}$.
22: **return** $\mathsf{S}_k=\mathsf{U}_k\mathsf{W}_k^\top$. // $\mathsf{U}_k=[\boldsymbol{u}_1,\dots,\boldsymbol{u}_k]$, $\mathsf{W}_k=[\boldsymbol{w}_1,\dots,\boldsymbol{w}_k]$.

The approximation can be stopped for a certain rank k^* if the corresponding residuum is small enough. To determine this one could, for example, compute the Frobenius norm $\|\mathsf{R}_{k^*}\|_F = (\sum_{i=1}^{m}\sum_{j=1}^{n} R_{k^*}[i,j]^2)^{1/2}$ of R_{k^*} and demand that

$$\|\mathsf{R}_{k^*}\|_F \leq \widetilde{\varepsilon}_{\mathrm{ACA}} \, \|\mathsf{A}\|_F \tag{8.2}$$

for a sufficiently small constant $\widetilde{\varepsilon}_{\mathrm{ACA}} > 0$. Since the full matrix A needs to be computed to check (8.2), we stop the approximation instead if

$$|\boldsymbol{u}_{k^*}|^2 |\boldsymbol{w}_{k^*}|^2 \leq \varepsilon_{\mathrm{ACA}} \|\mathsf{S}_{k^*}\|_F^2 \tag{8.3}$$

for another small constant $\varepsilon_{\mathrm{ACA}} > 0$. Under certain assumptions on the matrix A and the construction of the approximations $\{\mathsf{S}_k\}_k$, the stopping criterion (8.3) implies the estimate (8.2); see [11, Sections 3.4.1 and 3.4.2]. We will further comment on this in Remark 8.2. Note that the norm $\|\mathsf{S}_{k+1}\|_F^2$ can be computed from $\|\mathsf{S}_k\|_F^2$ in each iteration step using the formula

$$\begin{aligned}\|\mathsf{S}_{k+1}\|_F^2 = \|\mathsf{U}_{k+1}\mathsf{W}_{k+1}^\top\|_F^2 &= \sum_{i=0}^{k+1}\sum_{j=0}^{k+1}(\boldsymbol{u}_i \cdot \boldsymbol{u}_j)(\boldsymbol{w}_i \cdot \boldsymbol{w}_j) \\ &= \|\mathsf{S}_k\|_F^2 + 2\sum_{i=0}^{k}(\boldsymbol{u}_i \cdot \boldsymbol{u}_{k+1})(\boldsymbol{w}_i \cdot \boldsymbol{w}_{k+1}) + |\boldsymbol{u}_{k+1}|^2|\boldsymbol{w}_{k+1}|^2,\end{aligned}$$

which can be found in [11, cf. p. 11, Equation (1.4)]. We compute the norm of S_{k+1} in this way in lines 15–17 of Algorithm 8.1, where it is denoted as n_{frob}.

Even if the stopping criterion (8.3) is not satisfied, there are two additional cases in which the computation of the low-rank approximation of A is stopped:

(i) The rank k has reached a certain threshold $k_{\max}$ or is equal to the number of columns n of A.

(ii) All rows of A have been considered, i.e. $\mathcal{R} = \{1, \ldots, m\}$.

If $k = n$ or $\mathcal{R} = \{1, \ldots, m\}$, $\mathsf{U}_k\mathsf{W}_k^\top$ corresponds to the full matrix A which means that the low-rank approximation has failed. By introducing a reasonable threshold $k_{\max}$ one can also stop the approximation earlier when it is foreseeable that the ACA will not yield an efficient approximation of A. In all these cases the full matrix A should be used instead of the low-rank approximation S_k returned in Algorithm 8.1, which we denote as ACA low-rank matrix in the following.

The runtime complexity of the ACA in Algorithm 8.1 is $\mathcal{O}((\#\mathcal{R})^2(m+n))$. This is stated in [11, Section 3.4.1] and can also be seen by counting the number of floating point operations in the algorithm. During the computation $\mathcal{O}(\#\mathcal{R}(m+n))$ entries of A are accessed. If these entries are computed on demand and the computation is expensive, the complexity of the ACA scales instead like $\mathcal{O}(\#\mathcal{R}(m+n))$;

see [11, Remark 3.31]. For a block A of a BEM matrix this is typically the case since for each entry a boundary integral needs to be evaluated. For the resulting approximation $(m+n)k$ entries of the matrices U_k and W_k need to be stored, which is more efficient than storing the original mn entries of A if the rank k satisfies $k < \min\{m,n\}/2$.

REMARK 8.1. *It is well known that the choice of the row indices $\{i_k\}_k$ in line 20 of Algorithm 8.1 combined with the stopping criterion* (8.3) *might cause the ACA to fail for certain matrices A. In fact, some parts of the matrix A might not be approximated at all by the resulting approximation S_k in certain pathological situations; see [11, Section 3.4.3] where a block of a double layer operator matrix is considered as an example. Different row selection strategies that allow one to overcome this deficiency are discussed in [11, Section 3.4.3] and [40, Section 3.2]. For the sake of simplicity we do not discuss them here, and apply the ACA only to suitable blocks of the single layer operator matrix V_h in the following sections.*

REMARK 8.2. *If the residual matrices $\{\mathsf{R}_k\}_k$ considered in the ACA satisfy an estimate of the form $\|\mathsf{R}_{k+1}\|_F \leq \eta\|\mathsf{R}_k\|_F$ for some value $\eta < 1$ and if the stopping criterion* (8.3) *is satisfied for a rank k^* and $\varepsilon_{\mathrm{ACA}} = (1-\eta)\widetilde{\varepsilon}_{\mathrm{ACA}}/(1+\widetilde{\varepsilon}_{\mathrm{ACA}})$, one can show that the estimate* (8.2) *is satisfied, i.e. S_{k^*} approximates A well; see [11, p. 141f.]. Let A be the Galerkin matrix for an integral operator $\mathcal{K}$, i.e.*

$$\begin{aligned} \mathsf{A}[i,j] &= \langle \varphi_i, \mathcal{K}\psi_j\rangle_\Gamma, \\ \mathcal{K}\psi_j(\boldsymbol{x}) &= \int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\psi_j(\boldsymbol{y})\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}, \end{aligned} \tag{8.4}$$

for some functions $\{\varphi_i\}_{i=1}^m$ with supports in a box X_{tar} and $\{\psi_j\}_{j=1}^n$ with supports in X_{src}. Let $\{\mathsf{R}_k\}_k$ be the residual matrices obtained in the adaptive cross approximation of such a matrix A. An upper bound for $|\mathsf{R}_k[i,j]|$ is derived in [11, Theorem 3.37] under suitable assumptions on the row indices $\{i_k\}_k$ chosen in the ACA. An essential part of this bound is the best approximation error

$$\max_{j=1,\ldots,n} \inf_{p\in\operatorname{span}\Xi} \|\mathcal{K}\psi_j - p\|_{\infty,X_{\mathrm{tar}}} \tag{8.5}$$

where Ξ is an arbitrary set of functions $\{\xi_\ell\}_{\ell=1}^{k'}$ with $\xi_1 = 1$, and k' is the number of vanishing rows of R_k. If one can show that this best approximation error (8.5) *converges exponentially to zero, an estimate of the form $\|\mathsf{R}_{k+1}\|_F \leq \eta\|\mathsf{R}_k\|_F$ is satisfied. For kernel functions $(\boldsymbol{x},\boldsymbol{y}) \mapsto K(\boldsymbol{x},\boldsymbol{y})$ which are asymptotically smooth — see Definition 5.2 — and boxes X_{src} and X_{tar} which satisfy an admissibility criterion like*

$$\eta_{\boldsymbol{x}} \operatorname{dist}(X_{\mathrm{tar}}, X_{\mathrm{src}}) \geq \max\{\operatorname{diam}(X_{\mathrm{tar}}), \operatorname{diam}(X_{\mathrm{src}})\} \tag{8.6}$$

for a suitable constant $\eta_{\boldsymbol{x}} > 0$, such an exponential decay of the best approximation error can be proven using polynomial interpolation. This is shown in [11, Section 3.3] and [14, Section 4.4].

8.2 The applicability of the ACA for the single layer operator matrix

In this section we motivate why the ACA from Section 8.1 can be used to efficiently compress certain blocks $\mathsf{V}_h|_{\hat{Z}_{\text{tar}}\times\hat{Z}_{\text{src}}}$ of the single layer operator matrix V_h. We are particularly interested in those blocks which are inadmissible, and thus not approximated in the FMM in Chapters 5 and 7. For our motivation we consider a special setting: We assume that Z_{src} and Z_{tar} are temporally indivisible clusters, i.e. their temporal components I_{src} and I_{tar} contain only a single time interval of the partition $\mathcal{I}_{h_t}$ of the space-time tensor product mesh Σ_h; see Definition 7.5. This situation is particularly relevant for inadmissible blocks of V_h and the meshes Σ_h mentioned in the introduction of this chapter for which the spatial mesh width $h_{\boldsymbol{x}}$ is considerably finer than the temporal mesh width h_t in the sense that the relation $h_t \sim h_{\boldsymbol{x}}^2$ is violated. We show that for such temporally indivisible clusters Z_{src} and Z_{tar} the block $\mathsf{V}_h|_{\hat{Z}_{\text{tar}}\times\hat{Z}_{\text{src}}}$ can be interpreted as a matrix generated by a purely spatial kernel function. By showing that this kernel function is asymptotically smooth we will conclude that the ACA can be used to compute low-rank approximations of such blocks if the spatial components of Z_{tar} and Z_{src} are suitably separated.

Let $\mathsf{V}_h|_{\hat{Z}_{\text{tar}}\times\hat{Z}_{\text{src}}}$ be a block of the single layer operator matrix V_h corresponding to two clusters Z_{tar} and Z_{src}. Its entries are given in (3.25), i.e. there holds

$$\begin{aligned}\mathsf{V}_h|_{\hat{Z}_{\text{tar}}\times\hat{Z}_{\text{src}}}&[(k_t-1)E_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)E_{\boldsymbol{x}}+j_{\boldsymbol{x}}]\\ &=\int_{t_{k_t-1}}^{t_{k_t}}\int_{\gamma_{k_{\boldsymbol{x}}}}\int_{t_{j_t-1}}^{t_{j_t}}\int_{\gamma_{j_{\boldsymbol{x}}}} G_\alpha(\boldsymbol{x}-\boldsymbol{y},t-\tau)\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\tau\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}\,\mathrm{d}t\end{aligned}$$

for all $(k_t,k_{\boldsymbol{x}})\in\hat{Z}_{\text{tar}}$ and all $(j_t,j_{\boldsymbol{x}})\in\hat{Z}_{\text{src}}$. As in previous sections, we use global indices to access the entries of the block here to simplify the description. In Section 3.3 we discussed that the integrals in (3.25) are evaluated in practice by an analytic integration in the two time-variables combined with quadrature routines in space. This is described in more detail in [81]. Here we present the resulting formulae after the analytic integration in time. We define the time-integrated kernel

$$F_{\alpha,k_t,j_t}(\boldsymbol{r}) := \int_{t_{k_t-1}}^{t_{k_t}}\int_{t_{j_t-1}}^{t_{j_t}} G_\alpha(\boldsymbol{r},t-\tau)\,\mathrm{d}\tau\,\mathrm{d}t \tag{8.7}$$

for $\boldsymbol{r}\in\mathbb{R}^3$ and distinguish three cases. If $k_t<j_t$, $F_{\alpha,k_t,j_t}(\boldsymbol{r})=0$ for all $\boldsymbol{r}\in\mathbb{R}^3$ due to the causality of the heat kernel G_α. If $k_t>j_t$ there holds

$$\begin{aligned}F_{\alpha,k_t,j_t}(\boldsymbol{r}) =&G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},t_{k_t}-t_{j_t})-G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},t_{k_t}-t_{j_t-1})\\ &-G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},t_{k_t-1}-t_{j_t})+G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},t_{k_t-1}-t_{j_t-1})\end{aligned} \tag{8.8}$$

for all $\boldsymbol{r} \in \mathbb{R}^3$, where $G_\alpha^{\mathrm{d}\tau\mathrm{d}t}$ is defined by

$$G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},\delta) = \frac{1}{4\pi}\left[\left(\frac{|\boldsymbol{r}|}{2\alpha^2} + \frac{\delta}{\alpha|\boldsymbol{r}|}\right)\operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2\sqrt{\alpha\delta}}\right) + \frac{\sqrt{\delta}}{\sqrt{\pi\alpha^3}}\exp\left(-\frac{|\boldsymbol{r}|^2}{4\alpha\delta}\right)\right], \tag{8.9}$$

$$G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},0) = \lim_{\delta\to 0} G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},\delta) = \frac{|\boldsymbol{r}|}{8\pi\alpha^2} \tag{8.10}$$

for all $\boldsymbol{r} \in \mathbb{R}^3 \setminus \{\boldsymbol{0}\}$ with the error function

$$\operatorname{erf}(x) := \frac{2}{\sqrt{\pi}}\int_0^x \exp(-y^2)\,\mathrm{d}y. \tag{8.11}$$

If instead $k_t = j_t$, we have

$$F_{\alpha,k_t,k_t}(\boldsymbol{r}) = h_{t,k_t} G_\alpha^{\mathrm{d}\tau}(\boldsymbol{r},0) - G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},h_{t,k_t}) + G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},0) \tag{8.12}$$

for all $\boldsymbol{r} \in \mathbb{R}^3$ with $h_{t,k_t} = t_{k_t} - t_{k_t-1}$ as defined in (2.13) and $G_\alpha^{\mathrm{d}\tau}(\boldsymbol{r},0)$ given by

$$G_\alpha^{\mathrm{d}\tau}(\boldsymbol{r},0) = \frac{1}{4\pi\alpha|\boldsymbol{r}|}. \tag{8.13}$$

Note that

$$F_{\alpha,k_t,k_t}(\boldsymbol{r}) \sim \frac{h_{t,k_t}}{4\pi\alpha|\boldsymbol{r}|} \qquad \text{as } \boldsymbol{r} \to \boldsymbol{0}, \tag{8.14}$$

while $F_{\alpha,k_t,j_t}(\boldsymbol{r})$ stays bounded as $\boldsymbol{r} \to 0$ if $j_t < k_t$. This follows, since $G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},0) \to 0$ as $\boldsymbol{r} \to \boldsymbol{0}$ and since

$$\lim_{\boldsymbol{r}\to\boldsymbol{0}} G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{r},\delta) = \frac{\sqrt{\delta}}{2\sqrt{\pi^3\alpha^3}}$$

for all $\delta > 0$; see [81, Section 4].

By using the function F_{α,k_t,j_t} we can rewrite the entries of the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ as

$$\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}[(k_t-1)E_{\boldsymbol{x}}+k_{\boldsymbol{x}},(j_t-1)E_{\boldsymbol{x}}+j_{\boldsymbol{x}}] = \int_{\gamma_{k_{\boldsymbol{x}}}}\int_{\gamma_{j_{\boldsymbol{x}}}} F_{\alpha,k_t,j_t}(\boldsymbol{x}-\boldsymbol{y})\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{y}}\,\mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \tag{8.15}$$

for all $(k_t,k_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{tar}}$ and $(j_t,j_{\boldsymbol{x}}) \in \hat{Z}_{\mathrm{src}}$. If we assume that Z_{src} and Z_{tar} are two temporally indivisible clusters, there exists only one time-index k_t in $\hat{Z}_{\mathrm{tar}}$ and one time-index j_t in $\hat{Z}_{\mathrm{src}}$, and thus the block in (8.15) is generated by a single kernel function F_{α,k_t,j_t}. In this representation, the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ has exactly the same form as a block of a single layer operator matrix of an elliptic PDE like the Laplace equation when a Galerkin method with piecewise constant basis functions in space is used for the discretization; cf. Equation (8.4) in Remark 8.2 and [11, Section 3.1]. Due to (8.14), the time-integrated kernel function $(\boldsymbol{x},\boldsymbol{y}) \mapsto F_{\alpha,k_t,k_t}(\boldsymbol{x}-\boldsymbol{y})$ is even asymptotically equivalent to a scaled version of the fundamental solution of the

Laplace equation $(\boldsymbol{x},\boldsymbol{y}) \mapsto (4\pi|\boldsymbol{x}-\boldsymbol{y}|)^{-1}$ as $\boldsymbol{x}-\boldsymbol{y} \to \boldsymbol{0}$. The ACA presented in Section 8.1 is known to be well suited for the approximation of blocks of the single layer operator matrix of the Laplace equation corresponding to clusters that satisfy an admissibility criterion like (8.6); see for example the numerical experiments in [62, Section 4.2]. This follows, in particular, from the results which we have mentioned in Remark 8.2. Those results are valid for more general matrices generated by asymptotically smooth functions. By showing that the time-integrated kernel functions F_{α,k_t,j_t} in (8.7) are asymptotically smooth, we motivate that the ACA can also be used to approximate suitable nearfield blocks of V_h which are inadmissible in the context of the FMM in Chapter 7. Note that the asymptotic smoothness of the time integrals of the heat kernel in 2+1D was shown in [60, Corollary 7.2.2].

To show that the kernel functions $(\boldsymbol{x},\boldsymbol{y}) \mapsto F_{\alpha,k_t,j_t}(\boldsymbol{x}-\boldsymbol{y})$ are asymptotically smooth for all $k_t, j_t \in \{1,\dots,E_t\}$ with $j_t \le k_t$, we use a few basic results about asymptotically smooth functions which we list in the following propositions.

PROPOSITION 8.3 ([35, Section E.1]). *Let $d \in \mathbb{N}$. The functions $(\boldsymbol{x},\boldsymbol{y}) \mapsto |\boldsymbol{x}-\boldsymbol{y}|^{-1}$ and $(\boldsymbol{x},\boldsymbol{y}) \mapsto |\boldsymbol{x}-\boldsymbol{y}|$ are asymptotically smooth on $\mathbb{R}^d \times \mathbb{R}^d$.*

PROPOSITION 8.4 ([35, Theorem E.8]). *Let $r_A > 0$ and $A := (-r_A, r_A) \subset \mathbb{R}$. Let $f : A \setminus \{0\} \to \mathbb{R}$ be asymptotically smooth in the sense that there exists an $s \in \mathbb{R}$ and suitable constants C, ρ and γ such that*

$$\left|\frac{\mathrm{d}^\nu}{\mathrm{d}r^\nu} f(r)\right| \le h_{\mathrm{as}}(\nu)|r|^{-\nu-s} \qquad \text{with } h_{\mathrm{as}}(\nu) := C\nu!\nu^\rho\gamma^\nu \tag{8.16}$$

for all $r \in A \setminus \{0\}$ and $\nu \in \mathbb{N}$. Let $d \in \mathbb{N}$ and $X, Y \subset \mathbb{R}^d$ be such that $|\boldsymbol{x}-\boldsymbol{y}| \le r_A$ for all $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$. Then the function $F : X \times Y \to \mathbb{R}$ defined by $F(\boldsymbol{x},\boldsymbol{y}) := f(|\boldsymbol{x}-\boldsymbol{y}|)$ for all $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$ is asymptotically smooth.

PROPOSITION 8.5 ([35, Remark E.11 and Theorem E.12]). *Let $d \in \mathbb{N}$ and $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^d$ be two non-empty, bounded, open sets. Let $f : X \times Y \to \mathbb{R}$ and $g : X \times Y \to \mathbb{R}$ be two asymptotically smooth functions. Then the sum $f+g$ and the product fg of f and g are asymptotically smooth on $X \times Y$.*

In the next proposition we show that the two functions $(\boldsymbol{x},\boldsymbol{y}) \mapsto \exp(-|\boldsymbol{x}-\boldsymbol{y}|^2/(4\alpha\delta))$ and $(\boldsymbol{x},\boldsymbol{y}) \mapsto \operatorname{erf}(|\boldsymbol{x}-\boldsymbol{y}|(2\sqrt{\alpha\delta}))$, which are parts of the function $G_\alpha^{\mathrm{d}\tau\mathrm{d}t}$ in (8.9), are asymptotically smooth.

PROPOSITION 8.6. *Let $X, Y \subset \mathbb{R}^3$ be two non-empty, bounded, open sets. Then*

$$(\boldsymbol{x},\boldsymbol{y}) \mapsto F_1(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4\alpha\delta}\right) \qquad \text{for } (\boldsymbol{x},\boldsymbol{y}) \in X \times Y, \tag{8.17}$$

$$(\boldsymbol{x},\boldsymbol{y}) \mapsto F_2(\boldsymbol{x},\boldsymbol{y}) = \operatorname{erf}\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|}{2\sqrt{\alpha\delta}}\right) \qquad \text{for } (\boldsymbol{x},\boldsymbol{y}) \in X \times Y, \tag{8.18}$$

are asymptotically smooth for all $\delta > 0$ and $\alpha > 0$.

Proof. For the functions $f_1 : \mathbb{R} \to \mathbb{R}$ and $f_2 : \mathbb{R} \to \mathbb{R}$ defined by

$$f_1(r) = \exp\left(-\frac{r^2}{4\alpha\delta}\right), \qquad f_2(r) = \operatorname{erf}\left(\frac{r}{2\sqrt{\alpha\delta}}\right)$$

there holds $F_1(\boldsymbol{x}, \boldsymbol{y}) = f_1(|\boldsymbol{x} - \boldsymbol{y}|)$ and $F_2(\boldsymbol{x}, \boldsymbol{y}) = f_2(|\boldsymbol{x} - \boldsymbol{y}|)$. According to Proposition 8.4 it is therefore sufficient to show that f_1 and f_2 are asymptotically smooth on $\mathbb{R}$ in the sense of (8.16) to conclude that F_1 and F_2 are asymptotically smooth on $X \times Y$.

Let us first consider f_1 and estimate $|(\mathrm{d}/\mathrm{d}r)^\nu f_1(r)|$ for all $\mathbb{R} \setminus \{0\}$ and $\nu \in \mathbb{N}$. Since

$$(\mathrm{d}/\mathrm{d}r)^\nu f_1(r) = (-1)^\nu (\mathrm{d}/\mathrm{d}r)^\nu f_1(-r)$$

holds for all $\nu \in \mathbb{N}_0$ and $r \in \mathbb{R}$ we can focus on the case $r > 0$. The function $z \mapsto f_1(z)$ is holomorphic on $\mathbb{C}$. Therefore, we can use Cauchy's integral formula to get

$$\frac{\mathrm{d}^\nu}{\mathrm{d}z^\nu} f_1(r) = \frac{\nu!}{2\pi i} \int_{\partial B(r,R)} \frac{f_1(\zeta)}{(\zeta - r)^{\nu+1}} \mathrm{d}\zeta$$

for all $r \in \mathbb{R}_{>0}$ and $R \in \mathbb{R}_{>0}$, where $B(r, R) \subset \mathbb{C}$ is the ball with center r and radius R. We can estimate $|f_1(\zeta)| \leq 1$ if $\Re(-\zeta^2/(4\alpha\delta)) \leq 0$, i.e. $\Re(\zeta^2) \geq 0$. If we write ζ in polar coordinates $\zeta = \varrho e^{i\varphi}$ with $\varrho \in \mathbb{R}_{\geq 0}$ and $\varphi \in (-\pi, \pi]$, we see that this is the case if and only if $|\varphi| \leq \pi/4$. In particular, it is satisfied for all $\zeta \in \partial B(r, R)$ with $R \leq r/\sqrt{2}$, because for such values R the ball $B(r, R)$ is fully contained in the set $\{z = \varrho e^{i\varphi} \in \mathbb{C} : |\varphi| \leq \pi/4\}$. Hence, for $R \leq r/\sqrt{2}$ we can estimate

$$\left|\frac{\mathrm{d}^\nu}{\mathrm{d}z^\nu} f_1(r)\right| \leq \frac{\nu!}{2\pi} \int_{\partial B(r,R)} \frac{|f_1(\zeta)|}{|\zeta - r|^{\nu+1}} \mathrm{d}\zeta \leq \frac{\nu!}{2\pi} \int_{\partial B(r,R)} \frac{1}{R^{\nu+1}} \mathrm{d}\zeta = \frac{\nu!}{R^\nu}.$$

Since $R \mapsto R^{-\nu}$ is monotonically decreasing, we can minimize this bound by choosing $R = r/\sqrt{2}$. In this way, we obtain the estimate

$$\left|\frac{\mathrm{d}^\nu}{\mathrm{d}z^\nu} f_1(r)\right| \leq \nu!\, 2^{\nu/2} r^{-\nu}$$

for all $\nu \in \mathbb{N}$, which shows that f_1 is asymptotically smooth.

To show that f_2 is asymptotically smooth too, we proceed in the same manner. The function f_2 satisfies $(\mathrm{d}/\mathrm{d}r)^\nu f_2(r) = (-1)^{\nu+1} (\mathrm{d}/\mathrm{d}r)^\nu f_2(-r)$ for all $\nu \in \mathbb{N}_0$ and $r \in \mathbb{R}$, so we can focus on the case $r > 0$ again. Since $z \mapsto f_2(z)$ is holomorphic on $\mathbb{C}$, we can estimate

$$\left|\frac{\mathrm{d}^\nu}{\mathrm{d}z^\nu} f_2(r)\right| \leq \frac{\nu!}{2\pi} \int_{\partial B(r,R)} \frac{|f_2(\zeta)|}{|\zeta - r|^{\nu+1}} \mathrm{d}\zeta \tag{8.19}$$

for all $r \in \mathbb{R}_{>0}$ and $R \in \mathbb{R}_{>0}$. We can rewrite $f_2(\zeta)$ as

$$\begin{aligned} f_2(\zeta) &= \operatorname{erf}\left(\frac{\zeta}{2\sqrt{\alpha\delta}}\right) = \frac{2}{\sqrt{\pi}} \int_0^{\frac{\zeta}{2\sqrt{\alpha\delta}}} \exp(-y^2)\,\mathrm{d}y \\ &= \frac{\zeta}{\sqrt{\pi\alpha\delta}} \int_0^1 \exp\left(-\frac{\zeta^2}{4\alpha\delta}\tau^2\right)\mathrm{d}\tau = \frac{\zeta}{\sqrt{\pi\alpha\delta}} \int_0^1 f_1(\zeta\tau)\,\mathrm{d}\tau, \end{aligned}$$

where we used the definition of the error function in (8.11), which is also valid for complex arguments, and the substitution $y = \tau\zeta/(2\sqrt{\alpha\delta})$. From this representation it follows that

$$|f_2(\zeta)| \leq \frac{|\zeta|}{\sqrt{\pi\alpha\delta}} \int_0^1 |f_1(\zeta\tau)|\,\mathrm{d}\tau \leq \frac{r+R}{\sqrt{\pi\alpha\delta}} \tag{8.20}$$

for all $\zeta \in \partial B(r, R)$ with $R \leq r/\sqrt{2}$, where we used that $|\zeta| \leq r + R$ and $|f_1(\zeta\tau)| \leq 1$ for all these values of ζ and all $\tau \in [0, 1]$. By using the estimate (8.20) in (8.19) it follows that

$$\left|\frac{\mathrm{d}^\nu}{\mathrm{d}z^\nu} f_2(r)\right| \leq \nu!\, \frac{r+R}{R^\nu\sqrt{\pi\alpha\delta}}.$$

The function $R \mapsto (r+R)R^{-\nu}$ is monotonically decreasing for all $r > 0$ and $\nu \in \mathbb{N}$, so we can minimize the bound in the last estimate by setting $R = r/\sqrt{2}$ and obtain the estimate

$$\left|\frac{\mathrm{d}^\nu}{\mathrm{d}z^\nu} f_2(r)\right| \leq \nu! \left(1 + \frac{1}{\sqrt{2}}\right) \frac{1}{\sqrt{\alpha\pi\delta}}\, 2^{\nu/2} r^{-\nu+1},$$

which proves that f_2 is asymptotically smooth. □

The asymptotical smoothness of the functions F_1 and F_2 in Proposition 8.6 implies that we can approximate them well on pairs of boxes X_{tar} and X_{src} in $\mathbb{R}^3$ that satisfy an admissibility criterion like (8.6), which ensures a separation in space. For F_1 we had already shown a stronger result in Section 5.1.2. In fact, the heat kernel $(\boldsymbol{x}, \boldsymbol{y}) \mapsto G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau)$ for a fixed time difference $\delta = t - \tau$ is just a scaled version of the function F_1, and in Theorem 5.7 we had shown that it can be approximated well by a truncated Chebyshev expansion. For this purpose, a separation of the boxes X_{tar} and X_{src} in space is not necessary. Only the sizes of the boxes have to be chosen suitably with respect to $\delta = t - \tau$ as stated in (5.19). Proposition 8.6 is helpful nonetheless since we can use it together with the results in the previous propositions to show that the more complicated time-integrated kernel functions in (8.7) are asymptotically smooth.

THEOREM 8.7. *Let $k_t \in \{1, \ldots, E_t\}$ and $j_t \in \{1, \ldots, k_t - 1\}$. Let $X, Y \subset \mathbb{R}^3$ be two non-empty, bounded, open sets. Then the functions $(\boldsymbol{x}, \boldsymbol{y}) \mapsto F_{\alpha,k_t,j_t}(\boldsymbol{x} - \boldsymbol{y})$ and $(\boldsymbol{x}, \boldsymbol{y}) \mapsto F_{\alpha,k_t,k_t}(\boldsymbol{x} - \boldsymbol{y})$ with F_{α,k_t,j_t} defined in (8.8) and F_{α,k_t,k_t} defined in (8.12) are asymptotically smooth on $X \times Y$.*

Proof. Let us first consider the function $(\boldsymbol{x},\boldsymbol{y}) \mapsto F_{\alpha,k_t,j_t}(\boldsymbol{x}-\boldsymbol{y})$ in (8.8) for $j_t < k_t$. Due to Proposition 8.5 we can conclude that it is asymptotically smooth on $X \times Y$ if we can show that the function $(\boldsymbol{x},\boldsymbol{y}) \mapsto G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{x}-\boldsymbol{y},\delta)$ is asymptotically smooth on $X \times Y$ for all $\delta \geq 0$. For $\delta = 0$ this is clear due to Proposition 8.3 since

$$G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{x}-\boldsymbol{y},0) = \frac{|\boldsymbol{x}-\boldsymbol{y}|}{8\pi\alpha^2};$$

see (8.10). For $\delta > 0$ we have

$$G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{x}-\boldsymbol{y},\delta) = \frac{|\boldsymbol{x}-\boldsymbol{y}|}{8\pi\alpha^2} F_2(\boldsymbol{x},\boldsymbol{y}) + \frac{\delta}{4\pi\alpha|\boldsymbol{x}-\boldsymbol{y}|} F_2(\boldsymbol{x},\boldsymbol{y}) + \frac{\sqrt{\delta}}{\sqrt{\pi\alpha^3}} F_1(\boldsymbol{x},\boldsymbol{y})$$

with the asymptotically smooth functions F_1 and F_2 from Proposition 8.6. According to Proposition 8.3, the function $(\boldsymbol{x},\boldsymbol{y}) \mapsto |\boldsymbol{x}-\boldsymbol{y}|^{-1}$ is asymptotically smooth on $X \times Y$ too. In particular, the function $(\boldsymbol{x},\boldsymbol{y}) \mapsto G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{x}-\boldsymbol{y},\delta)$ is the sum of three products of asymptotically smooth functions on $X \times Y$, and thus it is itself asymptotically smooth on $X \times Y$ because of Proposition 8.5. Therefore, we have proven the assertion for F_{α,k_t,j_t} with $j_t < k_t$. For the function $(\boldsymbol{x},\boldsymbol{y}) \mapsto F_{\alpha,k_t,k_t}(\boldsymbol{x}-\boldsymbol{y})$ it follows in the same way using the decomposition

$$F_{\alpha,k_t,k_t}(\boldsymbol{x}-\boldsymbol{y}) = \frac{h_{t,k_t}}{4\pi\alpha|\boldsymbol{x}-\boldsymbol{y}|} - G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{x}-\boldsymbol{y},h_{t,k_t}) + G_\alpha^{\mathrm{d}\tau\mathrm{d}t}(\boldsymbol{x}-\boldsymbol{y},0). \qquad \square$$

To summarize, we have shown that for two temporally indivisible space-time clusters Z_{tar} and Z_{src} the entries of the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ are of the form (8.4), and, more importantly, that the corresponding kernel function F_{α,k_t,j_t}, which is determined by the time-index k_t in the index set $\hat{Z}_{\mathrm{tar}}$ and j_t in $\hat{Z}_{\mathrm{src}}$, is asymptotically smooth for all cases $j_t \leq k_t$. According to Remark 8.2 we can therefore assume that the ACA can be used to efficiently approximate such a block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ if the spatial components X_{tar} and X_{src} of Z_{tar} and Z_{src} satisfy an admissibility criterion like (8.6).

8.3 The ACA nearfield compression in the space-time FMM

The ACA from Section 8.1 can be used to compress certain inadmissible blocks of the single layer operator matrix V_h in the space-time FMM from Chapter 7 to obtain a more efficient fast method for the application of V_h. In this section we describe the necessary changes in the setup and application of the FMM for this additional nearfield compression.

Let $\mathcal{T}_\Sigma^{\mathrm{ext}}$ be an extended space-time box cluster tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ as constructed in Algorithm 7.1 and let the operation lists for clusters in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ be constructed by Algorithm 7.2. We want to determine originally inadmissible blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ of V_h

with $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$ for which a compression by means of the ACA is applicable. According to the findings in Section 8.2 this is the case for temporally indivisible clusters Z_{tar} and Z_{src}, whose spatial components satisfy an admissibility criterion like (8.6). In our application we use a different admissibility criterion. We require that Z_{tar} and Z_{src} have the same spatial level $\ell_{\boldsymbol{x}}$, which we defined at the end of Section 7.2.1. Then their spatial components X_{tar} and X_{src} are contained in a regular grid $\mathcal{G}^{\ell_{\boldsymbol{x}}}$ and we can use the grid distance introduced in (5.30) in Section 5.2.2 to define the spatial separation criterion

$$\mathrm{griddist}(X_{\mathrm{tar}}, X_{\mathrm{src}}) > 1. \tag{8.21}$$

Note that there exists a constant $\eta_{\boldsymbol{x}}$ such that (8.6) is satisfied for all pairs of boxes X_{tar} and X_{src} in the grid $\mathcal{G}^{\ell_{\boldsymbol{x}}}$ that satisfy (8.21), even if the boxes are additionally padded as described at the end of Section 7.2.1.

An illustration of the criterion (8.21) is given in Figure 8.1, where we consider a fixed spatial target box X_{tar} and mark all the source boxes for which (8.21) is satisfied. Recall that when we construct the operation lists in Algorithm 7.1 we neglect interactions between clusters Z_{src} and Z_{tar} if X_{src} is not contained in the interaction area $\mathcal{I}_{\mathcal{A}}(X_{\mathrm{tar}})$ of X_{tar} defined in (5.31), i.e. if $\mathrm{griddist}(X_{\mathrm{tar}}, X_{\mathrm{src}}) > n_{\mathrm{tr}}$. This is indicated by the hatched boxes in Figure 8.1 for which (8.21) is satisfied, but which are not contained in the interaction area of X_{tar} in this particular example, so they can be ignored.

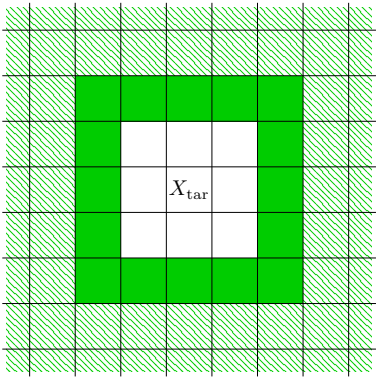


Figure 8.1: Illustration of the spatial separation criterion (8.21) for spatial boxes in 2D. For the fixed target box X_{tar} and any of the source boxes filled with green color (8.21) is satisfied. For the source boxes hatched in green (8.21) is also satisfied, but they are not contained in the interaction area $\mathcal{I}_A(X_{\mathrm{tar}})$ of X_{tar} defined in (5.31) if $n_{\mathrm{tr}} = 2$ is chosen as truncation parameter.

REMARK 8.8. *Numerical experiments indicate that also for originally inadmissible blocks corresponding to clusters Z_{tar} and Z_{src} which are not temporally indivisible we can apply the ACA to determine a potential low-rank approximation, as long as the spatial components of these clusters satisfy* (8.21)*. Therefore, we do not require that the clusters are temporally indivisible when we determine the blocks which are admissible for a compression by the ACA in the following.*

In Algorithm 8.2 we present the procedure to determine the *ACA admissible nearfield blocks* of V_h, i.e. all previously inadmissible blocks of V_h that can be compressed by the ACA. To keep track of these blocks we introduce the new ACA interaction lists $\mathcal{I}_{\text{ACA}}$ for clusters in $\mathcal{T}_\Sigma^{\text{ext}}$ which we fill during this procedure. We assume that the nearfield lists of clusters in $\mathcal{T}_\Sigma^{\text{ext}}$ have already been constructed using Algorithm 7.2. Then we consider all pairs of clusters Z_{tar} and Z_{src} with $Z_{\text{src}} \in \mathcal{N}(Z_{\text{tar}})$ to identify related ACA admissible nearfield blocks of V_h. First, we check if the spatial levels of the clusters agree; see line 3. If this is the case, we remove Z_{src} from the nearfield $\mathcal{N}(Z_{\text{tar}})$ and call the routine DETERMINEACAINTERACTIONLISTS for this pair of clusters.

Algorithm 8.2 Construction of ACA interaction lists

Require: Let the operation lists of clusters in $\mathcal{T}_\Sigma^{\text{ext}}$ be constructed by Algorithm 7.2.

1: **for** all $Z_{\text{tar}} \in \mathcal{T}_\Sigma^{\text{ext}}$
2: **for** all $Z_{\text{src}} \in \mathcal{N}(Z_{\text{tar}})$
3: **if** $\ell_{\boldsymbol{x}}(Z_{\text{src}}) == \ell_{\boldsymbol{x}}(Z_{\text{tar}})$:
4: Remove Z_{src} from $\mathcal{N}(Z_{\text{tar}})$.
5: Call DETERMINEACAINTERACTIONLISTS(Z_{src}, Z_{tar}).

6: **function** DETERMINEACAINTERACTIONLISTS(Z_{src}, Z_{tar})
7: **if** X_{src} and X_{src} satisfy (8.21):
8: Add Z_{src} to $\mathcal{I}_{\text{ACA}}(Z_{\text{tar}})$.
9: **else**
10: **if** Z_{src} and Z_{tar} are not leaves in $\mathcal{T}_\Sigma^{\text{ext}}$:
11: **for** all $(Z_{\text{src,c}}, Z_{\text{tar,c}})$ with $Z_{\text{src,c}} \in \text{child}(Z_{\text{src}})$, $Z_{\text{tar,c}} \in \text{child}(Z_{\text{tar}})$
12: Call DETERMINEACAINTERACTIONLISTS($Z_{\text{src,c}}$, $Z_{\text{tar,c}}$).
13: **else**
14: Add Z_{src} to $\mathcal{N}(Z_{\text{tar}})$.

In the routine DETERMINEACAINTERACTIONLISTS we check if the spatial separation criterion (8.21) is satisfied for the spatial components X_{tar} and X_{src} of Z_{tar} and Z_{src}. If this is true, the corresponding block can be approximated by the ACA, so we add Z_{src} to the ACA interaction list $\mathcal{I}_{\text{ACA}}(Z_{\text{tar}})$; see line 8. Otherwise we want to further subdivide the block $\mathsf{V}_h|_{\hat{Z}_{\text{tar}} \times \hat{Z}_{\text{src}}}$. If Z_{tar} and Z_{src} are not leaves in $\mathcal{T}_\Sigma^{\text{ext}}$, we

do this in the usual way by recursively calling the routine DETERMINEACAINTERACTIONLISTS for all pairs of children of Z_{tar} and Z_{src}; see lines 11–12. If either Z_{tar} or Z_{src} is a leaf, we stop the subdivision. In this case, the block is inadmissible, so we add Z_{src} to the nearfield $\mathcal{N}(Z_{\mathrm{tar}})$; see line 14. In this way, we partition the set of all originally inadmissible blocks of V_h that we have determined in Algorithm 7.2 into a set of ACA admissible nearfield blocks and a remaining set of blocks for which an approximation is not feasible according to our criteria.

REMARK 8.9. *The spatial separation criterion* (8.21) *can only be used for boxes* X_{tar} *and* X_{src} *corresponding to clusters* Z_{tar} *and* Z_{src} *whose spatial levels coincide. This is an implicit requirement in the routine* DETERMINEACAINTERACTIONLISTS *in Algorithm 8.2. For a pair of clusters* Z_{tar} *and* Z_{src} *with* $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$ *we check it explicitly in line 3 before calling the routine the first time. In line 12 we call it recursively without checking the spatial levels of the involved child clusters. In fact, this is not necessary: For the clusters for which we call* DETERMINEACAINTERACTIONLISTS *the first time there holds* $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$*, so we know that either* Z_{tar} *or* Z_{src} *is a leaf in* $\mathcal{T}_\Sigma^{\mathrm{ext}}$*, or both are temporally indivisible according to the construction of the operation lists in Algorithm 7.2. Only if both clusters are temporally indivisible, they can have children in* $\mathcal{T}_\Sigma^{\mathrm{ext}}$*. These children are obtained from* Z_{tar} *and* Z_{src} *by purely spatial refinements, so their spatial levels coincide if the spatial levels of* Z_{tar} *and* Z_{src} *are equal. Hence, whenever we reach line 12 in Algorithm 8.2 the considered child clusters have the same spatial level.*

REMARK 8.10. *By constructing the ACA interaction lists according to Algorithm 8.2 we compress only blocks* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ *by the ACA for which the spatial levels* $\ell_{\boldsymbol{x}}(Z_{\mathrm{tar}})$ *and* $\ell_{\boldsymbol{x}}(Z_{\mathrm{src}})$ *of the clusters* Z_{tar} *and* Z_{src} *coincide. We decided on this restriction since it helps to simplify the description and convey the main ideas. However, the ACA can also be applied to blocks* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ *where* $\ell_{\boldsymbol{x}}(Z_{\mathrm{tar}}) \neq \ell_{\boldsymbol{x}}(Z_{\mathrm{src}})$*, as long as the clusters are sufficiently separated in space. The criterion*

$$\widetilde{\eta}_{\boldsymbol{x}} \operatorname{dist}(X_1, X_2) \geq \min\{\operatorname{diam}(X_1), \operatorname{diam}(X_2)\} \tag{8.22}$$

with a constant $\widetilde{\eta}_{\boldsymbol{x}} > 0$ *is suitable to determine such clusters; cf. [11, Equation (3.35)]. We can use this criterion and one-sided subdivisions of blocks in the construction of the ACA interaction lists to determine additional blocks that can be approximated by the ACA. E.g., if* (8.22) *is not satisfied for the spatial components* X_{tar} *and* X_{src} *of a pair of clusters* Z_{tar} *and* Z_{src} *and if only* Z_{tar} *is a leaf, we can consider all blocks* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{c}}}$ *with* $Z_{\mathrm{c}} \in \operatorname{child} Z_{\mathrm{src}}$ *and compress them by the ACA if* X_{tar} *and* X_{c} *satisfy* (8.22)*. This approach is related to the temporally one-sided FMM operations which we introduced in Chapter 7. In the numerical experiments in Section 8.6 we will use this enhancement since it helps to further reduce the storage requirements of the method. Note that the criterion* (8.22) *can again be replaced by a grid criterion like* (8.21)*, if* X_{tar} *and* X_{src} *are contained in two regular grids, where one is obtained from the other by uniform refinements.*

To set up the new fast method we assemble and store the blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of V_h for all clusters Z_{tar} and Z_{src} with Z_{src} contained in the updated nearfield list $\mathcal{N}(Z_{\mathrm{tar}})$ as we did in previous chapters. During this assembly phase, we also approximate each ACA admissible nearfield block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$ by using the partially pivoted ACA in Algorithm 8.1 and obtain an ACA low-rank matrix

$$\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}} = \mathsf{U}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\mathsf{W}^{\top}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}.$$

If the ACA algorithm terminated because the stopping criterion (8.3) was satisfied for a small rank k, the matrix $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ is an efficient low-rank approximation of the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$. In this case we store the matrices $\mathsf{U}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ and $\mathsf{W}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ and use them when applying the block. If the ACA algorithm terminated because of any of the other conditions in Algorithm 8.1 the low-rank compression failed and we assemble and store the full block instead.

The new fast method is presented in Algorithm 8.3. It is an enhancement of the time-adaptive FMM in Algorithm 7.5, which uses the ACA low-rank approximations $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ of successfully approximated blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$ for a more efficient nearfield evaluation. In particular, only the nearfield evaluation differs from the time-adaptive FMM in Algorithm 7.5.

Algorithm 8.3 The space-time FMM for V_h with an ACA nearfield compression.

Require: Let all the requirements in Algorithm 7.5 be met.
Let the ACA interaction lists be constructed by Algorithm 8.2.
Let blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$ be approximated by Algorithm 8.1.

1: Execute lines 1–31 of Algorithm 7.5.
// Nearfield evaluation
2: **for** all $Z_{\mathrm{tar}} \in \mathcal{T}_{\Sigma}^{\mathrm{ext}}$
3: **for** all $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$
4: **if** $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ was successfully compressed by the ACA:
// Compressed nearfield operation:
5: Compute $\boldsymbol{w} = \mathsf{W}^{\top}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$.
6: Add the product $\mathsf{U}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\boldsymbol{w}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.
7: **else**
8: Nearfield operation: Add the product $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.
9: **for** all $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$
10: Nearfield operation: Add the product $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.

When applying a block of the matrix V_h in the nearfield evaluation phase in Algorithm 8.3 we have to distinguish between compressed and uncompressed blocks. To identify the blocks of V_h compressed by the ACA we consider the ACA interaction lists $\mathcal{I}_{\mathrm{ACA}}$ of clusters in $\mathcal{T}_{\Sigma}^{\mathrm{ext}}$; see line 3. If the compression of a block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$

by the ACA was successful, we apply the ACA low-rank matrix $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ to the corresponding part of the source vector $\boldsymbol{q}$ in lines 4–6, where we use the low-rank representation of $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ and compute the product in two steps for the sake of efficiency. If the compression was not successful, the full block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ has been assembled instead and is applied to the corresponding part of the source vector in line 8. All other uncompressed blocks of V_h correspond to pairs of clusters $(Z_{\mathrm{src}}, Z_{\mathrm{tar}})$ with $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$ and are applied in lines 9–10.

The additional nearfield compression in the fast method in Algorithm 8.3 allows us to overcome the storage and runtime complexity $\mathcal{O}(E_t E_{\boldsymbol{x}}^2)$ of the fast methods in Algorithms 5.3 and 7.5 for meshes where the spatial mesh width $h_{\boldsymbol{x}}$ and the temporal mesh width h_t satisfy $h_{\boldsymbol{x}}^2 \ll h_t$. In fact, we can interpret the nearfield compression as an approximation of each originally inadmissible nearfield block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of V_h in the original FMM by a hierarchical matrix ($\mathcal{H}$-matrix) using the ACA as described in [11]. If such a block has $\mathcal{O}(E_{\boldsymbol{x}}^2)$ entries, the complexity to compute this $\mathcal{H}$-matrix approximation is $\mathcal{O}(E_{\boldsymbol{x}} \log(E_{\boldsymbol{x}}) |\log(\varepsilon_{\mathrm{ACA}})|^6)$, where $\varepsilon_{\mathrm{ACA}}$ is the accuracy in the stopping criterion (8.3), and the complexity to store and apply the approximated block is reduced from $\mathcal{O}(E_{\boldsymbol{x}}^2)$ to $\mathcal{O}(E_{\boldsymbol{x}} \log(E_{\boldsymbol{x}}) |\log(\varepsilon_{\mathrm{ACA}})|^3)$; see [11, Section 3.4.4]. In particular, if there are $\mathcal{O}(E_t)$ originally inadmissible blocks in the FMM in Algorithm 5.3 with $\mathcal{O}(E_{\boldsymbol{x}}^2)$ entries, the complexity to store and apply all these blocks is reduced from $\mathcal{O}(E_t E_{\boldsymbol{x}}^2)$ to $\mathcal{O}(E_t E_{\boldsymbol{x}} \log(E_{\boldsymbol{x}}) |\log(\varepsilon_{\mathrm{ACA}})|^3)$.

Remark 8.11. *The described nearfield compression agrees well with our parallelization strategy in Chapter 6. In fact, each process assembles and applies a local part of all inadmissible blocks in our parallel FMM individually. To incorporate the nearfield compression, each process can use a local variant of Algorithm 8.2 to determine all ACA admissible nearfield blocks among its share of inadmissible blocks, approximate them, and apply the compressed blocks in the parallel application of V_h together with related inadmissible blocks whenever a corresponding nearfield operation is executed.*

8.4 A recompression strategy to improve the nearfield compression

While the ACA in Algorithm 8.1 typically yields an adequate low-rank approximation S_k when it is applied to a suitable matrix $\mathsf{A} \in \mathbb{R}^{m\times n}$, the rank k of this approximation does not need to be optimal in general. A recompression of the matrix S_k by means of a truncated singular value decomposition can be used to reduce the rank of S_k in a post processing step; see e.g. [12, Section 2.2]. In this section we describe such a recompression in the context of the fast method from Section 8.3. The novelty

of our approach is the criterion that we use to choose the recompression ranks, which is based on a comparison of an approximated block with a suitable diagonal block of the matrix V_h.

To compute a singular value decomposition (SVD) of a low-rank matrix $\mathsf{S}_k = \mathsf{U}\mathsf{W}^\top$ with $\mathsf{U} \in \mathbb{R}^{m\times k}$ and $\mathsf{W} \in \mathbb{R}^{n\times k}$ we proceed as described in [12, Section 2.2]. First, we compute the QR decompositions $\mathsf{U} = \mathsf{Q}_\mathsf{U}\mathsf{R}_\mathsf{U}$ and $\mathsf{W} = \mathsf{Q}_\mathsf{W}\mathsf{R}_\mathsf{W}$, where $\mathsf{Q}_\mathsf{U} \in \mathbb{R}^{m\times k}$ and $\mathsf{Q}_\mathsf{W} \in \mathbb{R}^{n\times k}$ are orthogonal matrices and $\mathsf{R}_\mathsf{U}, \mathsf{R}_\mathsf{W} \in \mathbb{R}^{k\times k}$ are upper triangular matrices. Secondly, we compute a singular value decomposition of $\mathsf{R}_\mathsf{U}\mathsf{R}_\mathsf{W}^\top$, which has the form $\hat{\mathsf{U}}\hat{\Sigma}\hat{\mathsf{W}}^\top$, where $\hat{\mathsf{U}}$ and $\hat{\mathsf{W}}$ are orthogonal matrices and $\hat{\Sigma}$ is a diagonal matrix containing the singular values of $\mathsf{R}_\mathsf{U}\mathsf{R}_\mathsf{W}^\top$. The singular values $\sigma_j = \hat{\Sigma}[j,j]$ with $j \in \{1,\dots,k\}$ are non-negative real numbers and are typically arranged in descending order, i.e. $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq 0$, which we assume here. From this singular value decomposition of $\mathsf{R}_\mathsf{U}\mathsf{R}_\mathsf{W}^\top$ we get a singular value decomposition $\mathsf{S}_k = \widetilde{\mathsf{U}}\hat{\Sigma}\widetilde{\mathsf{W}}^\top$ with the orthogonal matrices $\widetilde{\mathsf{U}} = \mathsf{Q}_U\hat{\mathsf{U}} \in \mathbb{R}^{m\times k}$ and $\widetilde{\mathsf{W}} = \mathsf{Q}_\mathsf{W}\hat{\mathsf{W}} \in \mathbb{R}^{n\times k}$.

We obtain an approximation of the matrix S_k with rank $r < k$ from the singular value decomposition $\mathsf{S}_k = \widetilde{\mathsf{U}}\hat{\Sigma}\widetilde{\mathsf{W}}^\top$ by setting all entries $\hat{\Sigma}[j,j]$ with $j > r$ to zero. This truncation yields the approximation $\widetilde{\mathsf{S}}_r = \widetilde{\mathsf{U}}_r\widetilde{\mathsf{W}}_r^\top$ of S_k with $\widetilde{\mathsf{U}}_r \in \mathbb{R}^{m\times r}$ and $\widetilde{\mathsf{W}}_r \in \mathbb{R}^{n\times r}$, where the columns of $\widetilde{\mathsf{U}}_r$ correspond to the first r columns of $\widetilde{\mathsf{U}}$ and the j^{th} column of $\widetilde{\mathsf{W}}_r$ corresponds to the j^{th} column of $\widetilde{\mathsf{W}}$ scaled by the singular value σ_j. One can show that

$$\|\mathsf{S}_k - \widetilde{\mathsf{U}}_r\widetilde{\mathsf{W}}_r^\top\|_2 = \sigma_{r+1}; \tag{8.23}$$

see [14, Lemma 5.19]. Here, $\|\cdot\|_2$ denotes the spectral norm of a matrix defined by

$$\|A\|_2 = \max_{\boldsymbol{x}\in\mathbb{R}^m\setminus\{\mathbf{0}\}} \frac{|A\boldsymbol{x}|}{|\boldsymbol{x}|} \tag{8.24}$$

and $|\boldsymbol{v}|$ is the Euclidean norm of the vector $\boldsymbol{v}$ as in the rest of this work.

We want to use the described recompression to reduce the ranks of the ACA low-rank matrices $\mathsf{S}_{k,Z_{\text{tar}}\times Z_{\text{src}}}$ in the context of the fast method from Section 8.3. While this kind of recompression is not new, we propose a new criterion to choose the ranks of the recompressed matrices. For a given ACA low-rank matrix $\mathsf{S}_{k,Z_{\text{tar}}\times Z_{\text{src}}}$ we choose the recompression rank r as the smallest value such that the $(r+1)^{\text{th}}$ singular value σ_{r+1} of $\mathsf{S}_{k,Z_{\text{tar}}\times Z_{\text{src}}}$ satisfies

$$\sigma_{r+1} < \varepsilon_{\text{rec}}\|\mathsf{V}_h|_{\hat{Z}_{\text{tar}}\times\hat{Z}_{\text{tar}}}\|_2 \tag{8.25}$$

for a sufficiently small constant $\varepsilon_{\text{rec}} > 0$ and obtain the approximation

$$\widetilde{\mathsf{S}}_{r,Z_{\text{tar}}\times Z_{\text{src}}} = \widetilde{\mathsf{U}}_{r,Z_{\text{tar}}\times Z_{\text{src}}}\widetilde{\mathsf{W}}_{r,Z_{\text{tar}}\times Z_{\text{src}}}^\top.$$

Choosing the recompression rank r by (8.25) means that we use $\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2$ as a reference value to determine the accuracy of the recompression instead of the largest singular value of $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$, which is often used for this purpose. Our choice is motivated by the fact that the blocks on the diagonal of V_h are the most relevant ones in the sense that they contain the entries with the largest absolute values among all the entries of V_h. This follows from the definition of the entries of V_h in (3.25) and the decay of the heat kernel G_α in (3.4) in space and time. In Section 8.5 we will show that this kind of recompression is sufficiently accurate.

The spectral norm of the diagonal block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ in (8.25) corresponds to its largest singular value, which is the square root of the largest eigenvalue λ_1 of the product $(\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}})^\top\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$. We can approximate λ_1 by applying the power iteration to this matrix, which is described, for example, in [20, Section 4.1]. Independently of the considered stopping criterion, this method yields an approximation $\widetilde{\lambda}_1$ of λ_1 which is less than or equal to $\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2^2$, so (8.25) is satisfied if we ensure that the first truncated singular value σ_{r+1} satisfies $\sigma_{r+1} < \varepsilon_{\mathrm{rec}}(\widetilde{\lambda}_1)^{1/2}$. Note that a less accurate approximation of the spectral norm leads to a more restrictive truncation criterion. By bounding the number of iterations of the power iteration in practice for all blocks by a suitable constant n_{pow} we might therefore increase the recompression ranks of some blocks but do not reduce the accuracy of the recompression.

If a cluster Z_{tar} has children in $\mathcal{T}_\Sigma^{\mathrm{ext}}$, the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ is not assembled in the fast method in Section 8.3, because Z_{tar} is not contained in $\mathcal{N}(Z_{\mathrm{tar}})$ due to the subdivision of blocks in Algorithm 8.2. This means that we cannot estimate the norm $\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2$ as described above. Therefore, we do not use the criterion (8.25) for the choice of the recompression rank r for such clusters, but require instead that

$$\sigma_{r+1} < \varepsilon_{\mathrm{rec}} \max\{\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{d}}\times\hat{Z}_{\mathrm{d}}}\|_2 : Z_{\mathrm{d}} \text{ is a leaf in } \mathcal{T}_\Sigma^{\mathrm{ext}} \text{ and a descendant of } Z_{\mathrm{tar}}\}. \quad (8.26)$$

Note that if (8.26) is satisfied, (8.25) follows since for every descendant Z_{d} of Z_{tar} the block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{d}}\times\hat{Z}_{\mathrm{d}}}$ is a subblock of $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ and, therefore,

$$\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{d}}\times\hat{Z}_{\mathrm{d}}}\|_2 \le \|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2.$$

REMARK 8.12. *The criteria* (8.25) *and* (8.26) *might even be satisfied for the recompression rank* $r = 0$ *if the spectral norm of the ACA low-rank matrix* $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ *approximating a block* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ *satisfies*

$$\|\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\|_2 \le \varepsilon_{\mathrm{rec}}\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2. \quad (8.27)$$

In this case we discard the matrix $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$.

To incorporate the recompression of the ACA low-rank matrices in the fast method in Section 8.3 we need to modify the way in which we assemble the blocks of V_h. Since

we may need the spectral norms of some diagonal blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ for the recompression described above, we first assemble all blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ with $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$. The spectral norms of blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ are then determined for all leaves $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma^{\mathrm{ext}}$ by using the power iteration as described above and stored to access them later. Once this is done, we can consider the ACA admissible nearfield blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ of V_h. For each of these blocks we use the ACA in Algorithm 8.1 to construct an ACA low-rank matrix $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$. If the approximation is successful, we recompress the resulting low-rank matrix by a truncated SVD as described above where we determine the recompression rank r by using either the criterion (8.25) if Z_{tar} is a leaf in $\mathcal{T}_\Sigma^{\mathrm{ext}}$ or (8.26) otherwise. If the compression by the standard ACA is not successful, we assemble and store the full block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ instead.

In the fast method in Algorithm 8.3 only minor changes are necessary to take the additional recompression of ACA low-rank matrices into account. In fact, we only need to replace each ACA low-rank matrix $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ by the corresponding recompressed matrix $\widetilde{\mathsf{S}}_{r,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ in the application. The resulting method is presented in Algorithm 8.4.

Algorithm 8.4 The fast method in Algorithm 8.3 with the additional recompression.

Require: Let all the requirements in Algorithm 7.5 be met.
Let the ACA interaction lists be constructed by Algorithm 8.2.
Let blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$ be compressed by Algorithm 8.1 and recompressed by a truncated SVD choosing the rank by (8.25) or (8.26).

1: Execute lines 1–31 of Algorithm 7.5.
// Nearfield evaluation
2: **for** all $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma^{\mathrm{ext}}$
3: **for** all $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$
4: **if** $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ was successfully compressed by the ACA with recompression:
5: **if** the recompression rank r is greater than zero:
// Compressed nearfield operation:
6: Compute $\boldsymbol{w} = \widetilde{\mathsf{W}}_{r,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}^{\top}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$.
7: Add the product $\widetilde{\mathsf{U}}_{r,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\boldsymbol{w}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.
8: **else**
9: Nearfield operation: Add the product $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.
10: **for** all $Z_{\mathrm{src}} \in \mathcal{N}(Z_{\mathrm{tar}})$
11: Nearfield operation: Add the product $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}$ to $\boldsymbol{f}|_{\hat{Z}_{\mathrm{tar}}}$.

Let us discuss the computational effort of the additional recompression. As described above we first estimate the spectral norms of the blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ for all leaves $Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma$. The number of space-time elements assigned to a leaf Z_{tar}, i.e. $\#\hat{Z}_{\mathrm{tar}}$,

is bounded by a constant $n_{\max}$ due to the construction of the extended space-time box cluster tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ in Algorithm 7.1. Thus, approximating the spectral norm of the corresponding diagonal block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ by applying n_{pow} iterations of the power iteration to $(\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}})^\top\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ requires $\mathcal{O}(n_{\max}^2 n_{\mathrm{pow}})$ operations. If we assume that $\mathcal{T}_\Sigma^{\mathrm{ext}}$ has $\mathcal{O}(E_{\boldsymbol{x}}E_t)$ leaves, the spectral norms of all corresponding diagonal blocks of V_h can therefore be approximated in $\mathcal{O}(E_tE_{\boldsymbol{x}}n_{\max}^2 n_{\mathrm{pow}})$ operations, i.e. with linear complexity. Note that, in general, the effective costs for the assembly of the related blocks will dominate the costs of the applied power iterations if n_{pow} is reasonably bounded.

Once the spectral norm estimates are available, the ACA low-rank blocks are assembled and recompressed. Computing the singular value decomposition for the recompression of a single low-rank matrix $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ with rank k as described at the beginning of this section requires $\mathcal{O}((\#\hat{Z}_{\mathrm{tar}}+\#\hat{Z}_{\mathrm{src}}+k)k^2)$ operations; see for example [12, Section 2.2]. Recall that the computation of $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ with the ACA requires $\mathcal{O}((\#\mathcal{R})^2(\#\hat{Z}_{\mathrm{tar}}+\#\hat{Z}_{\mathrm{src}}))$ operations, where $\#\mathcal{R}\geq k$ is the number of considered rows in Algorithm 8.1, and that we bound the rank k of each block by a parameter $k_{\max}$. Therefore, the costs of the additional singular value decomposition and its truncation scale like the costs of computing the initial low-rank approximation. This shows that the additional recompression strategy does not reduce the efficiency of the fast method in Section 8.3. While we cannot show that it improves the storage and runtime complexity of that algorithm asymptotically, we will see its positive effects in the numerical experiments in Section 8.6.

REMARK 8.13 (A heuristic extension of the recompression strategy). *The idea of our recompression strategy is to consider all ACA admissible nearfield blocks* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ *of* V_h*, apply the ACA in Algorithm 8.1 to compress them, and recompress them if Algorithm 8.1 terminated because the stopping criterion* (8.3) *was satisfied, i.e. the approximation was successful. The blocks for which the approximation by the ACA stopped before* (8.3) *was satisfied because the rank of the approximation reached the bound* $k_{\max}$ *are not recompressed but fully assembled. In fact, if the stopping criterion* (8.3) *is not satisfied, one cannot expect that the low-rank approximation computed by the ACA is sufficiently accurate in general. However, if we could show that an estimate like*

$$\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}-\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\|_2\leq\varepsilon_{\mathrm{rec}}\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2 \tag{8.28}$$

holds for a low-rank approximation $\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ *of a block* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}$ *for which* (8.3) *was not satisfied, we could recompress it like the successfully approximated blocks using the appropriate criterion* (8.25) *or* (8.26) *to determine the recompression rank* r *and get an approximation* $\widetilde{\mathsf{S}}_{r,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}$ *that is accurate in the sense that*

$$\begin{aligned}&\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}-\widetilde{\mathsf{S}}_{r,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\|_2\\&\quad\leq\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}-\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\|_2+\|\mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}-\widetilde{\mathsf{S}}_{r,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\|_2\leq 2\varepsilon_{\mathrm{rec}}\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2.\end{aligned}$$

Note that the spectral norm of the diagonal block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ *is used as a reference value on the right-hand side of* (8.28)*. Hence, it is a rather weak requirement that might also be satisfied if the estimate*

$$\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}} - \mathsf{S}_{k,Z_{\mathrm{tar}}\times Z_{\mathrm{src}}}\|_F \leq \widetilde{\varepsilon}_{\mathrm{ACA}}\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\|_F$$

related to the stopping criterion (8.3) *of the ACA is not satisfied. In Section 8.6.2 we consider numerical experiments where the ACA fails for certain blocks. There we motivate why the estimate* (8.28) *might hold for these blocks and the related low-rank approximations, and investigate the effect of their recompression on the efficiency and accuracy of the fast method.*

Remark 8.14. *If one wants to apply the ACA nearfield compression with the additional recompression to the double layer operator matrix* K_h*, the recompression criteria* (8.25) *and* (8.26) *need to be adapted because diagonal blocks of* K_h *can be zero. In fact, if all the elements of the spatial part* Γ_h *of* Σ_h *which are contained in the spatial part* X_{tar} *of* Z_{tar} *lie in a plane, the integral kernel* $\alpha\,\partial_{\boldsymbol{n_y}}G_\alpha$ *of* K_h *in* (5.33) *vanishes on* $\Gamma_h \cap X_{\mathrm{tar}}$ *and the diagonal block* $\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}$ *is zero. However, not all diagonal blocks of* K_h *are zero at once, and the gradient of the heat kernel decays in space and time similarly as the heat kernel itself. Thus, the criterion*

$$\sigma_{r+1} \leq \varepsilon_{\mathrm{rec}}\max\{\|\mathsf{K}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2 : Z_{\mathrm{tar}} \text{ is a leaf in } \mathcal{T}_\Sigma^{\mathrm{ext}}\}$$

should be a suitable substitute for the recompression criteria (8.25) *and* (8.26) *when considering* K_h*. The same is true for the adjoint double layer operator matrix* $\mathsf{K}_h^{\top_{\boldsymbol{x}}}$ *and the matrix* D_h *of the hypersingular operator.*

8.5 Analysis of the recompression error

In this section we analyze the effect of the additional recompression of ACA low-rank matrices in the fast method in Algorithm 8.4 on the approximation of the matrix V_h. The results which we present can be used to conclude that when solving a BEM system like (3.24) and using this fast method with suitably chosen parameters for the application of V_h in an iterative solver, the approximation quality of the obtained solution is not reduced.

To be able to analyze the approximation error we define the matrices corresponding to the fast methods in Algorithm 8.3 and Algorithm 8.4.

Definition 8.15. *Let* $\mathsf{V}_h \in \mathbb{R}^{E_t E_{\boldsymbol{x}}\times E_t E_{\boldsymbol{x}}}$ *be the single layer operator matrix defined in* (3.25)*. We define* $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}} \in \mathbb{R}^{E_t E_{\boldsymbol{x}}\times E_t E_{\boldsymbol{x}}}$ *as the matrix corresponding to the fast method for the application of* V_h *in Algorithm 8.3 and* $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}} \in \mathbb{R}^{E_t E_{\boldsymbol{x}}\times E_t E_{\boldsymbol{x}}}$ *as the matrix corresponding to the fast method in Algorithm 8.4.*

To simplify the discussion, we assume that the parameters in Algorithm 8.3 and in all the corresponding preparatory routines are chosen such that

$$\|\mathsf{V}_h - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}\|_2 \leq \widetilde{\varepsilon}, \tag{8.29}$$

i.e. the spectral norm error between V_h and its approximation in the fast method in Algorithm 8.3 that includes the ACA nearfield compression but not the additional recompression is bounded by a sufficiently small constant $\widetilde{\varepsilon} > 0$. Such an estimate can be shown by deriving an estimate like (8.29) for each block of the partition of V_h induced by the operation lists constructed in Algorithms 7.2 and 8.2, and combining the blockwise results to obtain the global estimate (8.29); see for example [14, Section 4.6]. The approximation of an admissible block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ with Z_{src} contained in one of the interaction lists $\mathcal{I}_{\mathrm{M2L}}$, $\mathcal{I}_{\mathrm{M2Lx}}$, or $\mathcal{I}_{\mathrm{Mx2L}}$ of Z_{tar} is obtained by replacing the heat kernel in the integrals (3.25) defining the entries of V_h by the respective kernel expansions (5.9), (7.1) or (7.3) as we have seen in Sections 5.2.3 and 7.2.3. By using the results about the approximation errors of these kernel expansions in Theorems 5.9 and 7.2, one can estimate the approximation errors of the related blocks; see [14, Lemma 4.44]. The blockwise error estimates for the approximation of the ACA admissible nearfield blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$ are obtained from the error analysis in [11, Section 3.4].

In Theorem 8.16 we estimate the approximation error related to the recompression of all the ACA low-rank blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$, which is $\|\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}} - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}\|_2$.

THEOREM 8.16. *Let $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ be the matrices related to the fast methods in Algorithms 8.3 and 8.4 as introduced in Definition 8.15 and let*

$$\widetilde{\mathsf{V}}_h^{\mathrm{diff}} := \widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}} - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}.$$

Let $\varepsilon_{\mathrm{rec}} > 0$ be the constant in the criteria (8.25) and (8.26) that are used to determine the recompression ranks of all the ACA low-rank blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$. Then there exists a constant c_{diff} depending on the underlying space-time box cluster tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ (see Remark 8.17) such that

$$\|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}\|_2 \leq c_{\mathrm{diff}}\, \varepsilon_{\mathrm{rec}}\, \|\mathsf{V}_h\|_2. \tag{8.30}$$

Proof. The matrix $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ is obtained from $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ by recompressing all the low-rank blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ that are obtained by successful applications of the partially pivoted ACA in Algorithm 8.1 to ACA admissible nearfield blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ of V_h corresponding to clusters Z_{tar} and Z_{src} with $Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$. In particular, there holds

$$\|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}\|_2 \leq \varepsilon_{\mathrm{rec}} \|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{tar}}}\|_2 \tag{8.31}$$

for all these blocks due to the recompression criteria (8.25) and (8.26), and the entries of all the other blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{diff}}$ are zero. Let

$$\mathcal{A}_{\mathrm{tar}} := \{Z \in \mathcal{T}_\Sigma^{\mathrm{ext}} : \mathcal{I}_{\mathrm{ACA}}(Z) \neq \emptyset\} \tag{8.32}$$

and $\boldsymbol{q} \in \mathbb{R}^{E_t E_{\mathfrak{w}}}$ with $|\boldsymbol{q}| = 1$. To simplify the discussion, let us first assume that the index sets of clusters in $\mathcal{A}_{\mathrm{tar}}$ are pairwise disjoint, i.e. $\hat{Z}_1 \cap \hat{Z}_2 = \emptyset$ for all Z_1 and Z_2 in $\mathcal{A}_{\mathrm{tar}}$ with $Z_1 \neq Z_2$. Then there holds

$$\left|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}\boldsymbol{q}\right|^2 = \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \left|(\widetilde{\mathsf{V}}_h^{\mathrm{diff}}\boldsymbol{q})|_{\hat{Z}_{\mathrm{tar}}}\right|^2 = \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \left| \sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} \widetilde{\mathsf{V}}_h^{\mathrm{diff}}|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}\right|^2. \tag{8.33}$$

By using the triangle inequality, the consistency of the matrix norm $\|\cdot\|_2$ with respect to the Euclidean vector norm $|\cdot|$, and (8.31) we further get

$$\begin{aligned}
\left|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}\boldsymbol{q}\right|^2 &\leq \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \left(\sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} \left|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}\right| \right)^2 \\
&\leq \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \left(\sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} \left\|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{src}}}\right\|_2 |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}| \right)^2 \\
&\leq \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \left(\sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} \varepsilon_{\mathrm{rec}} \left\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\right\|_2 |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}| \right)^2.
\end{aligned}$$

Let $n_{\mathrm{src}}^{\mathrm{ACA}}$ be a bound for the number of source clusters in the ACA interaction list $\mathcal{I}_{\mathrm{ACA}}(Z)$ of all clusters $Z \in \mathcal{A}_{\mathrm{tar}}$. We can continue the estimation of $|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}\boldsymbol{q}|^2$ by using the estimate $\|\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}}\times\hat{Z}_{\mathrm{tar}}}\|_2 \leq \|\mathsf{V}_h\|_2$ for all $Z_{\mathrm{tar}} \in \mathcal{A}_{\mathrm{tar}}$ and the Cauchy–Schwarz inequality to obtain

$$\begin{aligned}
\left|\widetilde{\mathsf{V}}_h^{\mathrm{diff}}\boldsymbol{q}\right|^2 &\leq \varepsilon_{\mathrm{rec}}^2 \left\|\mathsf{V}_h\right\|_2^2 \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \left(\sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}| \right)^2 \\
&\leq \varepsilon_{\mathrm{rec}}^2 \left\|\mathsf{V}_h\right\|_2^2 \sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} n_{\mathrm{src}}^{\mathrm{ACA}} \sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}|^2.
\end{aligned} \tag{8.34}$$

To further simplify this expression, we reorder the double sum in the last line. For this reason we define the set

$$\mathcal{A}_{\mathrm{src}} := \{Z \in \mathcal{T}_\Sigma^{\mathrm{ext}} : \exists Z_{\mathrm{tar}} \in \mathcal{T}_\Sigma^{\mathrm{ext}} \text{ such that } Z \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})\} \tag{8.35}$$

and get

$$\begin{aligned}
\sum_{Z_{\mathrm{tar}}\in\mathcal{A}_{\mathrm{tar}}} \sum_{Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}|^2 &= \sum_{Z_{\mathrm{src}}\in\mathcal{A}_{\mathrm{src}}} \sum_{Z_{\mathrm{tar}}:Z_{\mathrm{src}}\in\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}|^2 \\
&\leq \sum_{Z_{\mathrm{src}}\in\mathcal{A}_{\mathrm{src}}} |\boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}}|^2 n_{\mathrm{tar}}^{\mathrm{ACA}},
\end{aligned}$$

where $n_{\mathrm{tar}}^{\mathrm{ACA}}$ is a bound for the number of target clusters Z_{tar} in whose ACA interaction list $\mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})$ a source cluster Z_{src} is contained. If we assume that also the index sets of clusters in $\mathcal{A}_{\mathrm{src}}$ are pairwise disjoint there holds

$$\sum_{Z_{\mathrm{src}} \in \mathcal{A}_{\mathrm{src}}} \left| \boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}} \right|^2 n_{\mathrm{tar}}^{\mathrm{ACA}} \leq n_{\mathrm{tar}}^{\mathrm{ACA}} |\boldsymbol{q}|^2 = n_{\mathrm{tar}}^{\mathrm{ACA}} \tag{8.36}$$

and together with (8.34) we obtain

$$\left| \widetilde{\mathsf{V}}_h^{\mathrm{diff}} \boldsymbol{q} \right| \leq (n_{\mathrm{src}}^{\mathrm{ACA}} n_{\mathrm{tar}}^{\mathrm{ACA}})^{1/2} \, \varepsilon_{\mathrm{rec}} \left\| \mathsf{V}_h \right\|_2,$$

which is the estimate that we wanted to show.

We have to show a similar estimate in the case that the index sets of clusters in $\mathcal{A}_{\mathrm{tar}}$ and $\mathcal{A}_{\mathrm{src}}$ are not pairwise disjoint, which is possible due to the refinement of blocks in the construction of the ACA interaction lists in Algorithm 8.2. Let us first consider the sum in (8.36), i.e.

$$\sum_{Z_{\mathrm{src}} \in \mathcal{A}_{\mathrm{src}}} \left| \boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}} \right|^2.$$

For each element index pair $(j_t, j_{\boldsymbol{x}})$ and each level ℓ there exists at most one cluster Z at level ℓ of $\mathcal{T}_\Sigma^{\mathrm{ext}}$ such that $(j_t, j_{\boldsymbol{x}}) \in \hat{Z}$. Therefore, the number of clusters in $\mathcal{A}_{\mathrm{src}}$ which contain a fixed index pair $(j_t, j_{\boldsymbol{x}})$ is bounded by a constant $c_{j_t, j_{\boldsymbol{x}}}$ and the maximum

$$c_{\max} = \max\{c_{j_t, j_{\boldsymbol{x}}} : j_t \in \{1, \ldots, E_t\}, j_{\boldsymbol{x}} \in \{1, \ldots, E_{\boldsymbol{x}}\}\}$$

is bounded by $\mathrm{depth}(\mathcal{T}_\Sigma^{\mathrm{ext}})$. As a consequence, we get

$$\sum_{Z_{\mathrm{src}} \in \mathcal{A}_{\mathrm{src}}} \left| \boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}} \right|^2 \leq c_{\max} \sum_{j_t=1}^{E_t} \sum_{j_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}} q_{j_t, j_{\boldsymbol{x}}}^2 = |\boldsymbol{q}|^2 c_{\max} = c_{\max},$$

since we chose $\boldsymbol{q}$ such that $|\boldsymbol{q}| = 1$. In a similar way, we get the estimate

$$\left| \widetilde{\mathsf{V}}_h^{\mathrm{diff}} \boldsymbol{q} \right|^2 \leq c_{\max} \sum_{Z_{\mathrm{tar}} \in \mathcal{A}_{\mathrm{tar}}} \left| \sum_{Z_{\mathrm{src}} \in \mathcal{I}_{\mathrm{ACA}}(Z_{\mathrm{tar}})} \widetilde{\mathsf{V}}_h^{\mathrm{diff}}|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}} \boldsymbol{q}|_{\hat{Z}_{\mathrm{src}}} \right|^2$$

instead of the equality in (8.33). The remaining steps of the first part of the proof can be repeated without further modifications to obtain

$$\left| \widetilde{\mathsf{V}}_h^{\mathrm{diff}} \boldsymbol{q} \right| \leq c_{\max} (n_{\mathrm{src}}^{\mathrm{ACA}} n_{\mathrm{tar}}^{\mathrm{ACA}})^{1/2} \, \varepsilon_{\mathrm{rec}} \left\| \mathsf{V}_h \right\|_2, \tag{8.37}$$

which is (8.30) with $c_{\mathrm{diff}} = c_{\max} (n_{\mathrm{src}}^{\mathrm{ACA}} n_{\mathrm{tar}}^{\mathrm{ACA}})^{1/2}$. □

REMARK 8.17. *The constants $n_{\mathrm{src}}^{\mathrm{ACA}}$ and $n_{\mathrm{tar}}^{\mathrm{ACA}}$ in the estimate* (8.37) *are sometimes denoted as sparsity constants. Under suitable assumptions on the mesh one can show that these constants are bounded; cf. for example [11, Example 1.36]. The remaining constant $c_{\max}$ in* (8.37) *is bounded by the maximal number of refinements of an originally inadmissible block of V_h in the recursive construction of the ACA interaction lists in Algorithm 8.2 and scales like $\mathcal{O}(\log(E_{\boldsymbol{x}}))$ for globally quasi-uniform meshes with $E_{\boldsymbol{x}}$ spatial elements.*

The right-hand side of the estimate (8.30) in Theorem 8.16 includes the spectral norm $\|\mathsf{V}_h\|_2$ of the non-approximated single layer operator matrix V_h. We estimate this spectral norm next to get a better understanding of the approximation error in (8.30).

PROPOSITION 8.18. *Let $\Sigma_h = \Gamma_h \otimes \mathcal{I}_{h_t}$ be a space-time tensor product mesh for a lateral space-time boundary Σ as in Section 2.4 and $h_{\boldsymbol{x}}$ and h_t be the global spatial and temporal mesh sizes of Σ_h as defined in* (2.12) *and* (2.14)*, respectively. Let $V : H^{-1/2,-1/4}(\Sigma) \to H^{1/2,1/4}(\Sigma)$ be the single layer operator defined in* (3.8) *and let the constant c_2^V be a bound for its operator norm. Let $\mathsf{V}_h \in \mathbb{R}^{E_t E_{\boldsymbol{x}} \times E_t E_{\boldsymbol{x}}}$ be the BEM matrix of V for the space of piecewise constant test and trial functions $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ as in Section 3.3. Then*

$$\|\mathsf{V}_h\|_2 \leq c_2^V h_t h_{\boldsymbol{x}}^2. \tag{8.38}$$

Proof. Let $\boldsymbol{q}$ and $\boldsymbol{w}$ be two vectors in $\mathbb{R}^{E_t E_{\boldsymbol{x}}} \setminus \{\mathbf{0}\}$ and q_h and w_h be the corresponding functions in the space $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ of piecewise constant functions on Σ_h as in (2.17). To estimate the spectral norm of V_h we consider the inner product $\boldsymbol{q} \cdot \mathsf{V}_h \boldsymbol{w}$ and use the identity

$$\boldsymbol{q} \cdot \mathsf{V}_h \boldsymbol{w} = \langle q_h, V w_h \rangle_\Sigma,$$

where V is the single layer operator in (3.8). Due to the continuity of the duality product $\langle \cdot, \cdot \rangle_\Sigma$ on $H^{-1/2,-1/4}(\Sigma) \times H^{1/2,1/4}(\Sigma)$ and the boundedness of V as an operator from $H^{-1/2,-1/4}(\Sigma)$ to $H^{1/2,1/4}(\Sigma)$ there holds

$$\begin{aligned} \langle q_h, V w_h \rangle_\Sigma &\leq \|q_h\|_{H^{-1/2,-1/4}(\Sigma)} \|V w_h\|_{H^{1/2,1/4}(\Sigma)} \\ &\leq c_2^V \|q_h\|_{H^{-1/2,-1/4}(\Sigma)} \|w_h\|_{H^{-1/2,-1/4}(\Sigma)}. \end{aligned} \tag{8.39}$$

The functions w_h and q_h are contained in $L^2(\Sigma)$, so we can estimate their norms in $H^{-1/2,-1/4}(\Sigma)$ by their norms in $L^2(\Sigma)$. In this way, we obtain

$$\begin{aligned} \|q_h\|^2_{H^{-1/2,-1/4}(\Sigma)} \leq \|q_h\|^2_{L^2(\Sigma)} &= \sum_{j_t=1}^{E_t} \sum_{j_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}} \int_{t_{j_t-1}}^{t_{j_t}} \int_{\gamma_{j_{\boldsymbol{x}}}} q^2_{j_t,j_{\boldsymbol{x}}} \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}} \, \mathrm{d}t \\ &\leq h_t h_{\boldsymbol{x}}^2 \sum_{j_t=1}^{E_t} \sum_{j_{\boldsymbol{x}}=1}^{E_{\boldsymbol{x}}} q^2_{j_t,j_{\boldsymbol{x}}} = h_t h_{\boldsymbol{x}}^2 |\boldsymbol{q}|^2 \end{aligned}$$

and an analogous estimate for $\|w_h\|_{H^{-1/2,-1/4}(\Sigma)}$. By using these estimates and the one in (8.39) it follows that

$$\boldsymbol{q} \cdot \mathsf{V}_h \boldsymbol{w} = \langle q_h, V w_h \rangle_\Sigma \leq c_2^V h_t h_{\boldsymbol{x}}^2 |\boldsymbol{q}||\boldsymbol{w}|$$

for all $\boldsymbol{q}, \boldsymbol{w} \in \mathbb{R}^{E_t E_{\boldsymbol{x}}} \setminus \{\mathbf{0}\}$. We can reorder the terms in this inequality and insert $\boldsymbol{q} = \mathsf{V}_h \boldsymbol{w}$ to get

$$\frac{|\mathsf{V}_h \boldsymbol{w}|}{|\boldsymbol{w}|} \leq c_2^V h_t h_{\boldsymbol{x}}^2$$

for all $\boldsymbol{w} \in \mathbb{R}^{E_t E_{\boldsymbol{x}}} \setminus \{\mathbf{0}\}$ which yields the desired estimate (8.38) for $\|\mathsf{V}_h\|_2$. □

From Theorem 8.16 and Proposition 8.18 it follows that

$$\|\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}} - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}\|_2 \leq \varepsilon_{\mathrm{rec}}\, c_{\mathrm{diff}}\, c_2^V h_t h_{\boldsymbol{x}}^2,$$

i.e. the error related to the additional recompression of blocks approximated by the ACA can be controlled by choosing a sufficiently small constant $\varepsilon_{\mathrm{rec}}$ for the recompression criteria (8.25) and (8.26). Together with (8.29) we obtain the estimate

$$\begin{aligned} \|\mathsf{V}_h - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}\|_2 &\leq \|\mathsf{V}_h - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}\|_2 + \|\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}} - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}\|_2 \\ &\leq \widetilde{\varepsilon} + \varepsilon_{\mathrm{rec}}\, c_{\mathrm{diff}}\, c_2^V h_t h_{\boldsymbol{x}}^2 =: \varepsilon_{\mathrm{FMM,rNC}}\, h_t h_{\boldsymbol{x}}^2. \end{aligned} \tag{8.40}$$

Let us assume that the constant $\varepsilon_{\mathrm{FMM,rNC}}$ in (8.40) is sufficiently small and that the mesh Σ_h is part of a sequence of meshes that are globally quasi-uniform in space and globally quasi-uniform in time. Then we want to show that a system like (3.24) is still uniquely solvable if we replace V_h by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$, and that the approximation error of the solution of this perturbed system is still quasi-optimal in the energy norm $\|\cdot\|_{H^{-1/2,-1/4}(\Sigma)}$ like the solution of the non-perturbed system. Such a result is proven, for example, in [11, Section 3.3.4.1] in a general setting, but also in [53, Section 4] in the context of the parabolic FMM for the heat equation. In the latter work the authors derive their results by comparing the bilinear forms related to the non-approximated BEM matrices with the bilinear forms obtained by approximating these matrices with the parabolic FMM. In our setting, the corresponding bilinear forms $a(\cdot,\cdot)$ and $\widetilde{a}_h^{\mathrm{FMM,rNC}}(\cdot,\cdot)$ on $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h) \times S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ are defined by

$$a(q_h, w_h) := \boldsymbol{w}^\top \mathsf{V}_h \boldsymbol{q}, \qquad \widetilde{a}_h^{\mathrm{FMM,rNC}}(q_h, w_h) := \boldsymbol{w}^\top \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}} \boldsymbol{q}$$

where $\boldsymbol{q}$ and $\boldsymbol{w}$ are the coefficient vectors of the functions q_h and w_h in $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$. Since we assumed that Σ_h is globally quasi-uniform in space and time, there exists a constant c_{uni} such that

$$|\boldsymbol{q}|^2 \leq c_{\mathrm{uni}}\, h_t^{-1} h_{\boldsymbol{x}}^{-2} \|q_h\|^2_{L^2(\Sigma)}.$$

By using this estimate and the one in (8.40) we get

$$|a(q_h, w_h) - \widetilde{a}_h^{\mathrm{FMM,rNC}}(q_h, w_h)| = |\boldsymbol{w}^\top (\mathsf{V}_h - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}})\boldsymbol{q}| \leq \|\mathsf{V}_h - \widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}\|_2 |\boldsymbol{q}||\boldsymbol{w}| \\ \leq c_{\mathrm{uni}}^2 \, \varepsilon_{\mathrm{FMM,rNC}} \|q_h\|_{L^2(\Sigma)} \|w_h\|_{L^2(\Sigma)}.$$

In [53, Sections 4.1 and 4.2] an estimate of the same form is used as a starting point to show the existence, uniqueness, and quasi-optimality in the energy norm of solutions of perturbed BEM systems using, in particular, the first Strang lemma. One can follow the same lines to conclude that these results also hold in our setting for sufficiently small values of $\varepsilon_{\mathrm{FMM,rNC}}$. To be more precise, for the quasi-optimality of the lowest order approximation error in the energy norm $\|\cdot\|_{H^{-1/2,-1/4}(\Sigma)}$ we need to ensure that

$$\varepsilon_{\mathrm{FMM,rNC}} = \mathcal{O}\left((h_{\boldsymbol{x}}^2 + h_t) \left(\max\left\{ \frac{h_{\boldsymbol{x}}}{\sqrt{h_t}}, \frac{\sqrt{h_t}}{h_{\boldsymbol{x}}} \right\} \right)^{-1/2} \right);$$

cf. [53, Equation (4.4)].

8.6 Numerical experiments

In this section we present several numerical experiments to show the benefits of the proposed ACA based nearfield compression in the FMM. The experiments are subdivided into two sections. In Section 8.6.1 we consider numerical experiments for sequences of nested space-time tensor product meshes with a fixed number E_t of uniform time steps and a varying number $E_{\boldsymbol{x}}$ of spatial elements. Here we will see that the storage requirements and runtimes of the original FMM scale like $\mathcal{O}(E_t E_{\boldsymbol{x}}^2)$ and are significantly reduced by the fast methods in Algorithms 8.3 and 8.4. In Section 8.6.2 we revisit numerical experiments from Sections 6.5 and 7.5 to show that the additional nearfield compression reduces the effective storage requirements and runtimes also for more general examples.

If not stated otherwise, we choose the parameters $n_{\max} = 800$ and $c_{\mathrm{st}} = 4.1$ in all experiments when constructing an extended space-time box cluster tree $\mathcal{T}_\Sigma^{\mathrm{ext}}$ for a mesh Σ_h by Algorithm 7.1 and the parameters $n_{\mathrm{tr}} = 2$, $\eta_1 = \eta_2 = 1$, $m_t = 4$ and $m_{\boldsymbol{x}} = 12$ for the setup and application of the FMM operations from Chapters 5 and 7. For the nearfield compression by the ACA we set $k_{\max} = 150$ to bound the approximation ranks of the approximated blocks and use $\varepsilon_{\mathrm{ACA}} = 10^{-5}$ for the stopping criterion (8.3). When using the additional recompression from Section 8.4 we use $\varepsilon_{\mathrm{rec}} = 10^{-5}$ for the recompression criteria (8.25) and (8.26) and $n_{\mathrm{pow}} = 20$ iterations of the power iteration to estimate the spectral norms of the diagonal blocks appearing in these criteria. The basic implementation of the ACA which we use is taken from [18]. All occurring systems of linear equations are solved using the GMRES method with a relative accuracy of 10^{-8}.

8.6.1 Experiments for spatially refined meshes

As we mentioned in the introduction of Chapter 8, the standard FMM in Algorithm 5.3 and its extension to non-uniform temporal meshes in Algorithm 7.5 may not efficiently compress BEM matrices for space-time tensor product meshes whose spatial and temporal mesh widths $h_{\boldsymbol{x}}$ and h_t are such that $h_{\boldsymbol{x}}^2 \ll h_t$. In this section we present numerical experiments which show that our nearfield compression scheme based on the ACA in Section 8.1 allows one to overcome this deficiency like the nearfield compression scheme proposed in [52] does.

REMARK 8.19. *The number of time steps of the meshes in the following two numerical experiments is rather low. Hence, for each ACA admissible nearfield block* $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ *of* V_h *the clusters* Z_{src} *and* Z_{tar} *are temporally indivisible and the application of the ACA for the nearfield compression is justified according to Section 8.2.*

First experiment: A sequence of meshes refined uniformly in space. In the first experiment we consider a partition $\mathcal{I}_{h_t}$ of the time interval $(0, 0.25)$ consisting of $E_t = 16$ uniform time steps and a triangular mesh Γ_h of the surface of the cube $(0.5, 0.5)^3$ with $E_{\boldsymbol{x}} = 768$ congruent triangles. The spatial mesh is uniformly refined several times, while the time steps are not modified. In this way, we get a sequence of space-time tensor product meshes on which we solve the linear system (3.24) for the same initial Dirichlet boundary value problem with zero initial datum as in Section 6.5.1 using the GMRES method. For the application of the matrix V_h we use the fast methods in Algorithms 8.3 and 8.4 that include the nearfield compression by the ACA and in the case of the latter also the recompression from Section 8.4. To simplify the discussion we identify these methods with the related matrices $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ as introduced in Definition 8.15, and the standard FMM in Algorithm 5.3 with the matrix $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ when referring to them. For the computation of the right-hand side in (3.24) the matrix K_h is applied by using a variant of the standard FMM in Algorithm 5.3, where we apply the nearfield blocks directly after their assembly and discard them afterwards to reduce the memory demand. The computations were executed on the local workstation Babbage using all 32 available CPU cores; see Appendix A for more hardware details. The obtained results are presented in Tables 8.1 and 8.3 and discussed in detail in the following paragraphs. Since we are interested only in the effect of the nearfield compression introduced in this chapter, we do not report on the application times of the FMM approximation of K_h.

We start by comparing the storage requirements when using the different methods to approximate the matrix V_h. The numbers in lines 4–6 of Table 8.1 correspond to the memory required to store all the inadmissible blocks and all the ACA admissible nearfield blocks computed during the setup phase of the fast methods. Note that the numbers for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ were determined without assembling the actual blocks due to their

large sizes. In fact, we can see that these storage requirements scale quadratically with the number of spatial elements $E_{\boldsymbol{x}}$. This is due to the small number of time steps of the considered meshes. In fact, already for the mesh with 12 288 space-time elements all leaves of the corresponding space-time box cluster tree $\mathcal{T}_\Sigma$ are temporally indivisible, i.e. they contain only a single time-step of the temporal partition $\mathcal{I}_{h_t}$. By refining the space-time mesh only with respect to space, the numbers of space-time elements in these clusters increase and so do the corresponding blocks of the matrix V_h. While we can subdivide these clusters when constructing the extended space-time box cluster tree $\mathcal{T}_\Sigma^{\text{ext}}$ by Algorithm 7.1, we do not get any new admissible pairs of clusters satisfying a temporal admissibility criterion like (5.12) or (7.5) in this way, so neither in the standard FMM of Section 5.2.4 nor in its extension to temporally adaptive meshes in Section 7.2.4 we can subdivide and approximate the increasing inadmissible blocks. Therefore, we observe the quadratic increase in the storage requirements of $\widetilde{\mathsf{V}}_h^{\text{FMM}}$.

No. space-time elements $E_t E_{\boldsymbol{x}}$	12 288	49 152	196 608	786 432	3 145 728
No. spatial elements $E_{\boldsymbol{x}}$	768	3072	12288	49 152	196 608
No. time elements E_t	16	16	16	16	16
Storage $\widetilde{\mathsf{V}}_h^{\text{FMM}}$ [GiB]	0.14	2.18	34.88	558.0	8928
Storage $\widetilde{\mathsf{V}}_h^{\text{FMM,NC}}$ [GiB]	0.14	2.18	10.65	58.47	304.5
Storage $\widetilde{\mathsf{V}}_h^{\text{FMM,rNC}}$ [GiB]	0.14	2.18	6.29	35.46	191.7

Table 8.1: Results of the first experiment of Section 8.6.1 where the linear system (3.24) is solved for a sequence of space-time meshes of the time interval $(0, 0.25)$ and the surface of the cube $(-0.5, 0.5)^3$. The storage requirements of the FMM in Algorithm 5.3 ($\widetilde{\mathsf{V}}_h^{\text{FMM}}$) are compared with the storage requirements of the fast methods in Algorithm 8.3 ($\widetilde{\mathsf{V}}_h^{\text{FMM,NC}}$) and Algorithm 8.4 ($\widetilde{\mathsf{V}}_h^{\text{FMM,rNC}}$).

The storage requirements of $\widetilde{\mathsf{V}}_h^{\text{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\text{FMM,rNC}}$ do not differ from those of $\widetilde{\mathsf{V}}_h^{\text{FMM}}$ for the first two considered meshes. This has to do with our choice of the parameter $n_{\max}$ in the construction of the underlying extended space-time box cluster trees and its resulting structure: The maximal spatial level of the tree is zero for the first mesh and one for the second mesh. Therefore, a separation in space as we require it for the approximation of blocks by the ACA in (8.21) is not possible for clusters in these trees. Only for the considered meshes with more than 3 072 spatial elements the maximal spatial level of clusters in the tree is larger than one and the nearfield compression starts to show its effects. In fact, we see that the storage requirements of $\widetilde{\mathsf{V}}_h^{\text{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\text{FMM,rNC}}$ are significantly reduced in the fourth column of Table 8.1 and all following columns. In Figure 8.2 we plot the different storage requirements per space-time element, i.e. the total storage requirements divided by $E_{\boldsymbol{x}} E_t$, on a logarithmic

scale together with two reference curves. Here we can see that the elementwise storage requirements of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ scale like $\mathcal{O}(E_{\boldsymbol{x}} E_t)$ which means that its total storage requirements scale quadratically as we have already observed before. By comparing the curves of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ with the $\mathcal{O}(\log^3(E_{\boldsymbol{x}} E_t))$ reference curve we conclude that the total storage requirements of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ scale like $\mathcal{O}(E_{\boldsymbol{x}} E_t \log^3(E_{\boldsymbol{x}} E_t))$ for the considered meshes.

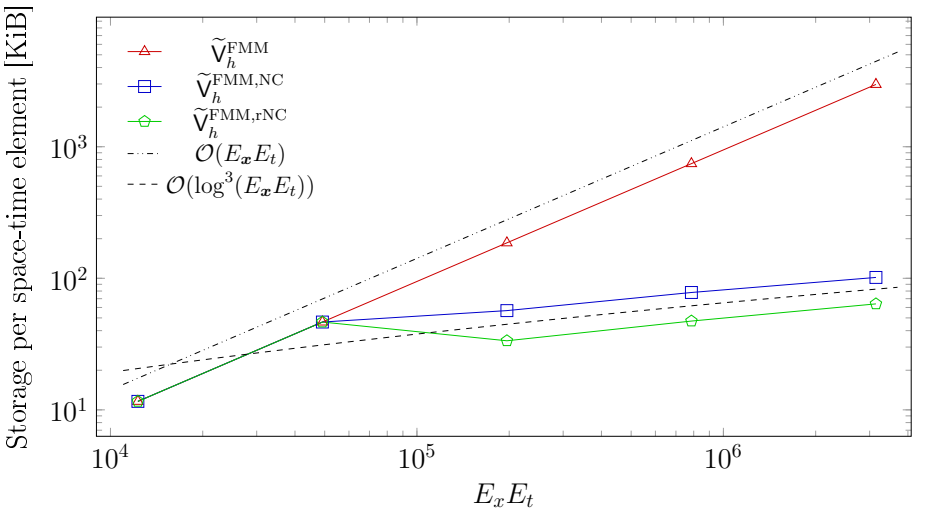


Figure 8.2: Plot of the storage requirements per space-time element of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$, $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ for the results in Table 8.1 together with related reference curves.

When comparing the storage requirements of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ in the fourth, fifth, and sixth columns of Table 8.1 we see that the additional recompression in the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ reduces the effective storage requirements by roughly 40 percent compared to $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$. To get a better understanding of this recompression, we consider the matrix V_h for the mesh with 786 432 space-time elements and take a closer look at how the ACA admissible nearfield blocks of V_h are approximated in the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$.

In Table 8.2 we see that the 243 536 ACA admissible nearfield blocks require a total of 526.6 GiB when they are stored without any approximation, so only 31.4 GiB of the 558 GiB of blocks which are stored in the case of the standard FMM are not approximated in the nearfield compression. In the case of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$, i.e. the nearfield compression with the ACA but without any recompression, all blocks are successfully approximated and the storage requirements are reduced from 526.6 GiB to 27.04 GiB.

By the additional recompression in the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ these storage requirements are further reduced to 4.04 GiB. In Table 8.2 we see that 40 160 blocks with a total memory demand of 230.4 GiB before the approximation can be discarded during the recompression because their recompression rank r equals zero; see Remark 8.12. The other 203 376 blocks require only 4.04 GiB of storage instead of 296.1 GiB after their approximation and the additional recompression.

		No. blocks	Storage [GiB] before approx.	Storage [GiB] after approx.
$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$	ACA admissible	243 536	526.6	27.04
	Compressed	243 536	526.6	27.04
$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$	ACA admissible	243 536	526.6	4.04
	Compressed	203 376	296.1	4.04
	Discarded	40 160	230.4	0

Table 8.2: Details about the nearfield compression of V_h for the mesh with 786 432 space-time elements in the experiment in Table 8.1. By *ACA admissible* we denote all the ACA admissible blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ of V_h. *Compressed* blocks are those that are successfully approximated by the ACA with a rank greater than zero and *discarded* blocks are those whose recompression rank r equals zero.

In Table 8.3 we present additional details of the conducted computations. Lines 4 and 5 contain the times required for the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$, i.e. for the assembly of the inadmissible blocks and the approximation of the ACA admissible nearfield blocks. The times for the smallest mesh are suboptimal due to the small number of clusters in the corresponding space-time cluster tree and the way in which we parallelize the assembly (see Section 6.2) which is not optimal for such small meshes if many CPU cores are used. The assembly times for the larger meshes scale similarly with respect to the number of space-time elements as the storage requirements of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ in Table 8.1. This is to be expected since the computational effort of assembling inadmissible blocks and ACA admissible nearfield blocks is proportional to the number of computed matrix entries in general. In the case of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ the ACA admissible nearfield blocks are approximated by first using the partially pivoted ACA as for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and recompressing them afterwards. Therefore, the setup times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ cannot behave as well as the corresponding storage requirements and will be larger than the setup times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ in general. However, the results in Table 8.3 indicate that the costs of the additional recompression are rather low. In fact, we see that the setup times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are almost identical for the mesh with 196 608 space-time elements, which is the first mesh for which the nearfield compression is in effect. For the two larger meshes the times for

the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are only slightly larger than the corresponding setup times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$. This validates our analysis of the costs of the recompression at the end of Section 8.4.

In lines 6–8 of Table 8.3 we present the required numbers of iterations and times per single iteration when solving the linear system (3.24) with the GMRES method and using Algorithms 8.3 and 8.4 for the application of V_h. For the first three meshes a large part of the computation time corresponds to FMM operations for admissible blocks. The number of these FMM operations does not vary for the considered sequence of meshes and the costs of the M2M, M2L and L2L operations stay constant when we refine the meshes in space. This explains why the GMRES iteration times, which are dominated by the application times of V_h, scale better than the assembly times and storage requirements for the first three meshes. Only for the last two meshes the application of the inadmissible blocks and ACA admissible nearfield blocks becomes dominant in the fast methods and we observe a quasi-linear increase in the iteration times. For these larger meshes the differences between the iteration times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$, which are caused by the better compression rates of the latter, are also more pronounced. Therefore, and due to the mild increase in the number of GMRES iterations, the total times required for the setup and solution are lower for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ than for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$; see lines 9 and 10 of Table 8.3.

No. space-time elements $E_t E_{\boldsymbol{x}}$	12 288	49 152	196 608	786 432	3 145 728
No. spatial elements $E_{\boldsymbol{x}}$	768	3072	12288	49 152	196 608
No. time elements E_t	16	16	16	16	16
Setup time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ $[s]$	2.02	16.85	79.56	413.9	2311
Setup time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	1.97	16.90	79.28	416.7	2327
GMRES it. time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ $[s]$	0.08	0.12	0.25	1.01	4.93
GMRES it. time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	0.078	0.12	0.19	0.78	3.38
No. GMRES iterations (both)	26	33	42	52	68
Total time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ $[s]$	4.1	20.7	89.9	466	2706
Total time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	4.0	20.9	87.4	457	2601
Rel. L^2 error BEM (both)	0.146	0.082	0.061	0.055	0.054
Rel. L^2 projection error	0.120	0.075	0.059	0.054	0.053

Table 8.3: Execution times and approximation errors for the computations in the first numerical experiment of Section 8.6.1. The listed total times include the setup times for V_h and the times for the solution of the considered linear system, but not the required times for the construction of the right-hand side of the system.

Finally, we consider the relative approximation errors $\|q - q_h\|_{L^2(\Sigma)}/\|q\|_{L^2(\Sigma)}$ between the known Neumann datum q and the approximate solutions q_h of the systems (3.24) for the different meshes in line 11 of Table 8.3. The differences between these errors when solving (3.24) and approximating V_h by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ or $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are very small, and, in particular, not observable when considering only the first three decimal places. E.g., for the mesh with 786 432 space-time elements the approximation error for the solution obtained when using $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ is 0.0549592, and 0.0549597 when using $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$. This shows that the additional recompression of the low-rank matrices is sufficiently accurate. In line 12 of Table 8.3 we present the relative approximation errors obtained by projecting q to the ansatz space $S^{0\otimes 0}_{h_{\boldsymbol{x}},h_t}(\Sigma_h)$ for each of the meshes as reference values. The BEM approximation errors are close to these best approximation errors — in particular for the larger meshes — which demonstrates the correctness of our implementation and the proposed methods. However, we also see that the approximation errors do not really decrease anymore after a few purely spatial mesh refinements, which is due to the too large time step size.

In conclusion, our first experiment shows that the fast methods in Algorithms 8.3 and 8.4 with the additional nearfield compression are capable of reducing the prohibitively large storage and runtime complexity $\mathcal{O}(E_t E_{\boldsymbol{x}}^2)$ of the space-time FMM in Section 5.2.4 for meshes with a fine spatial resolution. The additional recompression proves to be efficient in terms of the required computation time as well as the achieved storage reduction. Furthermore, our experiment indicates that both methods provide an approximation of the application of V_h that is sufficiently accurate for solving linear systems in BEM and not compromising the quasi-best approximation property of the related solutions. The academic nature of the experiment should be noted, though. The sequence of spatially refined meshes was only considered to show the benefits of the nearfield compression, but it is suboptimal for the solution of the considered initial boundary value problem, as the development of the approximation errors in Table 8.3 indicates. However, in real-life applications where a fine spatial resolution is necessary to resolve complicated geometries and a relatively coarse temporal resolution is sufficient, our fast methods should perform similarly well. Another use case is considered in the next experiment.

Second experiment: Meshes refined adaptively in space. In the second experiment we consider an initial Dirichlet boundary value problem for the prismatic L-shaped domain

$$\Omega = (-0.5, 0) \times (-0.5, 0.5) \times (0, 0.5) \cup (0, 0.5) \times (0, 0.5) \times (0, 0.5)$$

and the time interval $(0, T)$ with $T = 0.25$ which we want to solve by using the direct boundary integral approach from Section 3.2. The initial and boundary data are chosen such that the exact solution of the homogeneous heat equation is

$$u(r, \varphi, z, t) = J_{2/3}(r) \sin(2\varphi/3) \exp(-2t/3), \tag{8.41}$$

where $J_{2/3}$ is the Bessel function of the first kind of order $2/3$ and (r, φ, z) are cylinder coordinates in $\mathbb{R}^3$. We constructed this solution by using the method of separation of variables similarly as in [34, Proposition 4.4.2.2], where the well-known singular solutions of the Laplace equation for polygonal domains are derived. As stated in [3, p.360, Equation 9.1.7]

$$J_{2/3}(r) \sim \frac{1}{\Gamma(5/3)} \left(\frac{r}{2}\right)^{2/3} \qquad \text{for } 0 < r \ll \sqrt{5/3}.$$

Hence, the function u has a singularity along the edge $\{(0,0,z) \in \mathbb{R}^3 : z \in (0, 0.5)\}$ for all times $t \in (0,T)$ and, in particular, the Neumann datum which we want to determine is non-smooth. When such a non-smooth function is approximated on a sequence of uniformly refined meshes a reduced order of convergence of the approximation errors is to be expected. To obtain better convergence rates one can consider adaptively refined meshes instead, which is done in an adaptive BEM; see e.g. [31] for such a method in the context of the heat equation in two spatial dimensions.

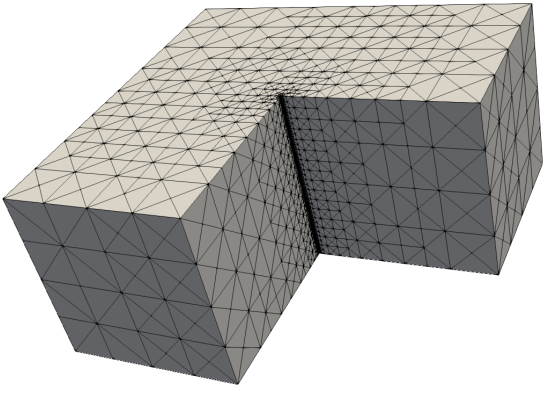


Figure 8.3: An adaptively refined surface mesh of the L-shaped domain Ω.

In our experiment we do not apply an adaptive BEM in the strict sense, but consider a sequence $\{\Gamma_{h,j}\}_j$ of adaptively refined spatial meshes; see e.g. Figure 8.3. These meshes were constructed using an adaptive BEM in [38, Section 6] to resolve the singularity of the Neumann datum of the solution $v(r, \varphi, z) = r^{2/3} \sin(2\varphi/3)$ of the Laplace equation on the L-shaped domain Ω, which has the same singular behavior along the edge $\{(0,0,z) \in \mathbb{R}^3 : z \in (0, 0.5)\}$ as our considered solution u of the heat equation. By combining these spatial meshes with a partition $\mathcal{I}_{h_t}$ of the time interval $(0,T)$ consisting of 16 uniform time steps, we obtain a sequence of space-time

tensor product meshes for which we solve the linear system (3.24) for the considered initial Dirichlet boundary value problem by using the GMRES method with a diagonal preconditioner. The computations were executed on the workstation Babbage using all 32 available CPU cores. The results are given in Table 8.4.

REMARK 8.20. *Since the initial datum* $u(r,\varphi,z,0) = J_{2/3}(r)\sin(2\varphi/3)$ *of the initial Dirichlet boundary value problem which we consider is non-zero, we need a mesh* Ω_h *of the domain* Ω *to discretize the initial datum and to construct the initial operator matrix* M_h^0 *defined in* (3.28)*. We use a sufficiently fine globally quasi-uniform volume mesh* Ω_h *consisting of 118 784 tetrahedra for all examples. Note that this mesh and the adaptively refined surface meshes* $\{\Gamma_{h,n}\}_n$ *are not conforming, i.e. the intersection of a tetrahedron* $T_{\Omega,k} \in \Omega_h$ *and a triangle* γ_j *in any of the meshes* $\{\Gamma_{h,n}\}_n$ *is not necessarily a vertex, edge, or triangle of the same surface mesh. Our results indicate that by using these combinations of meshes and evaluating the entries of* M_h^0 *as outlined in Section 3.3 we are able to compute the right-hand side of the system* (3.24) *accurately enough to not affect the approximation error of the BEM solution in a negative way. Since we are primarily interested in the effect of the nearfield compression of the fast methods in Algorithms 8.3 and 8.4 for* V_h*, we do not discuss the application times of* M_h^0 *or the double layer operator matrix* K_h *in the following.*

In Table 8.4 we see that the storage requirements of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are significantly lower than the storage requirements of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ for the considered spatially adaptive meshes, at least once a certain mesh size is reached. When comparing the storage requirements in Table 8.4 with the storage requirements from the first experiment in Table 8.1 we see that for a comparable amount of space-time elements they are quite similar. The same holds for the assembly and iteration times in Tables 8.3 and 8.4. The required numbers of GMRES iterations are slightly higher for the adaptive meshes in the second experiment, but due to the diagonal preconditioning, they are still reasonable. By comparing the L^2 approximation errors of the obtained BEM solutions with the L^2 projection errors of the actual Neumann datum in Table 8.4 we furthermore see that our matrix approximations are sufficiently accurate to preserve the quasi-optimality of the BEM solutions also in the second experiment. Note that here the approximation errors are reduced without any observable stagnation when increasing the number of adaptive space elements, although we keep the number of time steps fixed as in the first experiment. In fact, increasing the number of time steps does not bring any noteworthy benefit, which we checked by computing the related projection errors. This means that the spatial approximation quality is the limiting factor in this experiment, which is not surprising due to the singular behavior of the approximated Neumann datum in space. In conclusion, the results in Table 8.4 reveal that our new nearfield compression scheme allows one to recover the quasi-linear storage and runtime complexity of the FMM for the application of V_h also in the case of space-time meshes which are adaptively refined in space.

No. space-time elements $E_t E_{\boldsymbol{x}}$	13 696	54 592	210 816	791 040
No. spatial elements $E_{\boldsymbol{x}}$	856	3 412	13 176	49 440
No.time elements E_t	16	16	16	16
Storage $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ [GiB]	0.17	2.69	40.1	564.6
Storage $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ [GiB]	0.17	2.41	15.2	57.3
Storage $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ [GiB]	0.17	2.30	13.2	42.0
Setup time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ $[s]$	2.16	18.82	109.18	411.62
Setup time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	2.42	19.54	110.14	412.24
GMRES it. time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ $[s]$	0.058	0.10	0.31	0.95
GMRES it. time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	0.062	0.10	0.30	0.78
No. GMRES iterations (both)	58	78	98	117
Total time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ $[s]$	5.51	26.7	139.3	523
Total time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	6.03	27.2	139.6	504
Rel. L^2 error BEM (both)	0.162	0.125	0.098	0.078
Rel. L^2 projection error	0.151	0.120	0.094	0.075

Table 8.4: Results of the second experiment of Section 8.6.1 where the linear system (3.24) is solved for a sequence of spatially adaptive meshes for the time interval $(0, 0.25)$ and the surface of the L-shaped domain Ω. The matrix $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ represents the original FMM in Algorithm 5.3, $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ the fast method in Algorithm 8.3 and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ the one in Algorithm 8.4. The listed total times include the setup times for V_h and the times for the solution of the considered linear system, but not the required times for the construction of the right-hand side of the system.

8.6.2 Revisiting numerical experiments from previous chapters

In this section we revisit various numerical experiments from Chapters 6 and 7 and solve the related boundary integral equations by using the fast methods in Algorithms 8.3 and 8.4 for the approximation of V_h to study the effects of the additional nearfield compression in more general situations. In contrast to the experiments from the previous section, the clusters corresponding to ACA admissible nearfield blocks can contain more than one time step in the following experiments. While we have not explicitly shown that the partially pivoted ACA in Algorithm 8.1 is suitable for the approximation of such blocks, the results which we are going to present indicate that this is the case. In the following experiments we will also use the heuristic extension of the recompression strategy in Remark 8.13 and motivate why it is applicable.

First experiment: The cube example from Section 6.5.2. In the first experiment we consider the space-time tensor product mesh Σ_h from the distributed memory scalability test in Section 6.5.2 which consists of a surface mesh of the cube $(-0.5, 0.5)^3$ with 12 288 triangles and a partition of the time interval $(0, 0.25)$ into 1 024 uniform time steps. We solve the same boundary integral equation as in that section, or rather the corresponding discrete linear system (3.24) using the GMRES method. For the application of V_h we use the parallel FMM in Algorithm 6.5 and distributed parallel versions of the fast methods in Algorithm 8.3 and Algorithm 8.4. As in Section 8.6.1 we identify the three fast methods in the following discussion with the matrices $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$, $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$, respectively. To run the experiment we used 16 nodes of the VSC-4 cluster; see Appendix A for the hardware details.

By choosing the FMM parameters as discussed in the introduction of Section 8.6 we obtain a space-time cluster tree $\mathcal{T}_\Sigma$ for the space-time tensor product mesh Σ_h whose leaf clusters have the same level and contain four time steps. The ACA admissible nearfield blocks of V_h determined in Algorithm 8.2 correspond to pairs of such clusters. To start, we investigate how well the nearfield compression of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ works in this setting. In Table 8.5 we present the number of ACA admissible nearfield blocks of V_h and an overview of the successfully and unsuccessfully approximated parts of these blocks for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$.

	No. blocks	Storage [GiB] before approx.	Storage [GiB] after approx.
ACA admissible	3 139 584	577.9	217.8
Compressed	2 476 656	488.0	127.9
Uncompressed	662 928	89.9	89.9

Table 8.5: Details about the nearfield compression of V_h for the first experiment in Section 8.6.2 when using the fast method in Algorithm 8.3 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$). By *ACA admissible* we denote all ACA admissible nearfield blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ of V_h. *Compressed* blocks are those that are successfully approximated by the ACA with a rank greater than zero and *uncompressed* blocks are those which were not efficiently compressed and are thus fully assembled in the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$.

In Table 8.5 we see that the compression achieved by the ACA is rather bad in this example. For roughly 21 percent of all the ACA admissible nearfield blocks the approximation even failed. This rather large amount is the reason why we use the heuristic extension of the recompression from Remark 8.13 when setting up $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$. To motivate why this extension is applicable, we examine for which blocks the ACA fails. For this purpose we subdivide the ACA admissible nearfield blocks into equivalence classes based on the relative position of the corresponding space-time clusters

$Z_{\mathrm{tar}} = X_{\mathrm{tar}} \times I_{\mathrm{tar}}$ and $Z_{\mathrm{src}} = X_{\mathrm{src}} \times I_{\mathrm{src}}$. Since all leaf clusters in the space-time cluster tree $\mathcal{T}_\Sigma$ have the same level, the number of different relative configurations of such pairs of clusters is bounded by 250:

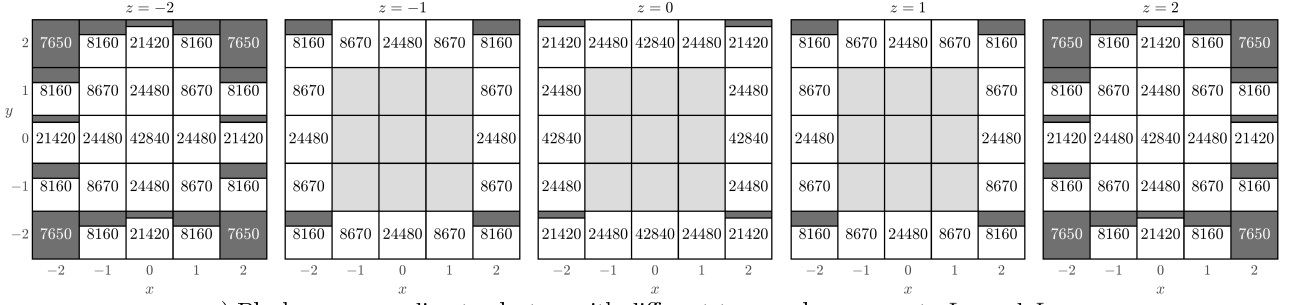
- 2 different configurations in time: ACA admissible nearfield blocks are inadmissible in the standard FMM in Chapter 5. Thus, I_{src} is either equal to I_{tar} or it is the causally relevant neighboring cluster of I_{tar}.

- At most 125 different configurations in space: If the grid distance of X_{tar} and X_{src} defined in (5.30) is greater than the chosen truncation parameter $n_{\mathrm{tr}} = 2$ the interaction between Z_{tar} and Z_{src} is neglected in the FMM (recall the definition of the interaction areas in (5.31)).

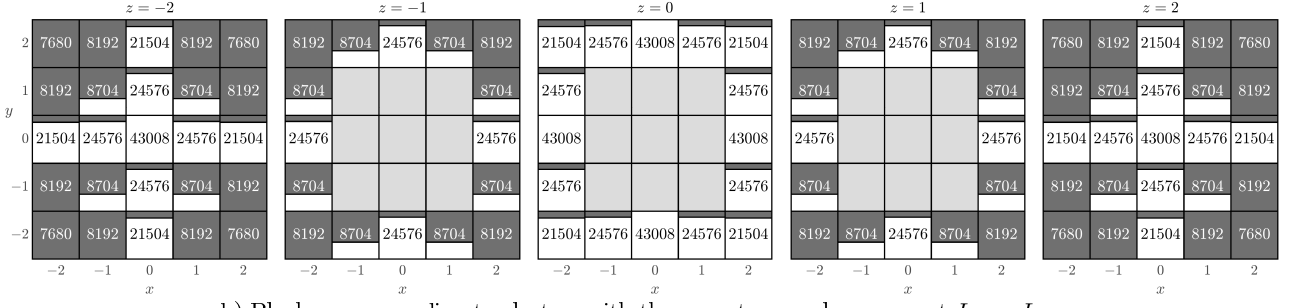
For each configuration we count how many of the corresponding ACA admissible nearfield blocks of V_h have been approximated successfully and how many have not. The result is illustrated in Figure 8.4. Here we see that the successful approximation of a block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ by the ACA depends on the relative position of the corresponding clusters. The partially pivoted ACA fails more frequently to provide an efficient low-rank approximation if the temporal components of the clusters Z_{tar} and Z_{src} coincide and if their spatial components are farther separated. The heat kernel $(\boldsymbol{x}, t), (\boldsymbol{y}, \tau) \mapsto G_\alpha(\boldsymbol{x} - \boldsymbol{y}, t - \tau)$ in (3.4) attains small values for such pairs of clusters due to its exponential decay in space, which is more pronounced for small differences in time. Hence, the entries of the corresponding block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ are particularly small compared to the entries of the related diagonal block $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{tar}}}$. This means that already a small relative accuracy of the ACA low-rank approximation $\mathsf{S}_{k, Z_{\mathrm{tar}} \times Z_{\mathrm{src}}}$ of $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ is sufficient to ensure that the estimate (8.28) in Remark 8.13 is satisfied, which serves as a motivation to use the heuristic extension of the recompression strategy for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ in the following.

	No. blocks	Storage [GiB] before approx.	Storage [GiB] after approx.
ACA admissible	3 139 584	577.9	13.7
Compressed	1 728 864	358.5	13.7
Discarded	1 410 720	219.4	0

Table 8.6: Details about the nearfield compression of V_h for the first experiment in Section 8.6.2 when using the fast method in Algorithm 8.4 with the heuristic extension of the recompression in Remark 8.13 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$). By *ACA admissible* we denote all the ACA admissible blocks $\mathsf{V}_h|_{\hat{Z}_{\mathrm{tar}} \times \hat{Z}_{\mathrm{src}}}$ of V_h. *Compressed* blocks are those that are successfully approximated by the ACA with a rank greater than zero and *discarded* blocks are those whose recompression rank r equals zero.



a) Blocks corresponding to clusters with different temporal components I_{src} and I_{tar}.



b) Blocks corresponding to clusters with the same temporal component $I_{\text{src}} = I_{\text{tar}}$.

Figure 8.4: Visualization of the blocks of $\widetilde{\mathrm{V}}_h^{\mathrm{FMM,NC}}$ approximated by the ACA in the first experiment of Section 8.6.2. Based on the relative position of their associated clusters $Z_{\text{tar}} = X_{\text{tar}} \times I_{\text{tar}}$ and $Z_{\text{src}} = X_{\text{src}} \times I_{\text{src}}$, the blocks are grouped into equivalence classes which are represented by the small squares. The position (x,y,z) of a square represents the relative position of X_{src} with respect to X_{tar} (0,0,0) for all blocks in its class, and the number inside a square represents the number of corresponding blocks. The colors indicate how many of them have been approximated successfully (white) or unsuccessfully (dark gray) by the ACA. If a square is split into two differently colored parts, the ratio of their sizes corresponds to the success rate of the approximation for the related equivalence class. Light gray squares without any number correspond to clusters that are not separated in space and thus not approximated.

In Table 8.6 we see the effect of the additional recompression in the nearfield compression of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$. The memory required to store all the ACA admissible nearfield blocks is reduced to 13.7 GiB, which is significantly less than the 577.9 GiB required to store the non-approximated blocks in the case of the standard FMM or the 217.8 GiB required to store the ACA admissible nearfield blocks in the case of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$. About 38 percent of the ACA admissible nearfield blocks are even discarded during the recompression because the determined recompression rank is zero.

	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$
Storage requirement [GiB]	890.8	530.7	326.6
Setup time $[s]$	299.8	212.3	195.2
GMRES iteration time $[s]$	6.14	5.80	5.50
No. GMRES iterations	59	59	59
Setup and solution time $[s]$	661.9	554.4	519.7
rel. L^2 approximation error	0.0297163	0.0297163	0.0297164

Table 8.7: Storage requirements, execution times and approximation errors for the computations in the first numerical experiment of Section 8.6.2 where the linear system (3.24) is solved for a space-time tensor product mesh with 1 024 time steps in the interval $(0, 0.25)$ and 12 288 triangles on the surface of the cube $(-0.5, 0.5)^3$. The matrix V_h is approximated by the FMM in Algorithm 6.5 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$), the fast method in Algorithm 8.3 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$) and the fast method in Algorithm 8.4 with the heuristic extension of the recompression in Remark 8.13 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$).

In Table 8.7 we compare the storage requirements, the setup and solution times, and the achieved accuracy when solving the system (3.24) for the considered boundary integral equation and approximating V_h by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$, $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$. Note that storing all non-zero entries of V_h without any compression would require 590 400 GiB of memory for this particular example. With the approximation of V_h by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ in the standard FMM we can already reduce this exorbitant amount to 890.8 GiB of storage for the inadmissible blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$. The new fast methods yield further reductions. Indeed, the memory required to store the inadmissible blocks and ACA admissible nearfield blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and, in particular, $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ is significantly less than the memory required for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$, which the results in Tables 8.5 and 8.6 already suggested. As a consequence, the setup times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are considerably lower than the setup time of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$. At first glance, it is surprising that the setup time of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ is about eight percent lower than the setup time of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ despite the additional recompression. The reason is that all the ACA admissible nearfield blocks for which the approximation by the ACA failed are recompressed in the case of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ due to our heuristic recompression scheme, while they are fully assembled

in the case of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$. Note that in our code we recompute all the entries of such blocks and do not reuse the entries which we already determined in the ACA, so a slight reduction of the setup times of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ would still be possible.

The times required for a single GMRES iteration of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$, $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ listed in Table 8.7 do not differ as much as the corresponding setup times. This is because the FMM operations of admissible blocks require the most time in the application of the three fast methods and are the same for all three of them. The total time required for the setup and solution of the linear system (3.24) is nonetheless about 16 percent lower for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ and about 21 percent lower for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ than the corresponding time for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$. At the same time, the accuracy of the matrix approximation does not suffer for the new fast methods, as the relative L^2 approximation errors $\|q - q_h\|_{L^2(\Sigma)}/\|q\|_{L^2(\Sigma)}$ in the last line of Table 8.7 indicate. When using $\widetilde{\mathsf{V}}_h^{\mathrm{FMM}}$ and $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,NC}}$ for the approximation of V_h these approximation errors do not differ in the specified six significant digits, and when using $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ only the last of these digits is different. This validates our assumptions on the applicability of the heuristic extension of the recompression strategy for the considered example.

Second experiment: The crankshaft example from Section 6.5.2. To investigate the effect of the nearfield compression in the case of a more realistic example, we revisit the second numerical example from Section 6.5.2, where we solved the linear system (3.24) on a space-time tensor product mesh with 1 024 uniform time steps in the time interval $(0, 0.25)$ and 42 888 triangles on the surface of the crankshaft depicted in Figure 6.6. We apply a parallel version of the fast method in Algorithm 8.4 to approximate the matrix V_h in (3.24). As in the previous experiments of this chapter we denote the resulting approximation by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$. To allow for an easier comparison of the new results with the results in Section 6.5.2 we choose the same FMM parameters as in that section and in addition the new parameters for the nearfield compression of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ as stated in the introduction of Section 8.6. The system (3.24) is solved using the GMRES method with a diagonal preconditioner. For the computations we used 64 nodes of the VSC-4 cluster. The resulting setup time for $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$, solution time, and required memory to store the inadmissible blocks and compressed ACA admissible nearfield blocks of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are presented in Table 8.8.

In the standard FMM in Section 6.5.2, 12 369 GiB of memory were required to store the inadmissible blocks of V_h which is why even 128 nodes of VSC-4 do not provide enough RAM to store these blocks. In contrast, only 2 711 GiB are required in the case of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$, which allowed us to use 64 nodes of VSC-4. The computation times in Table 8.8 and Table 6.3 are not really comparable due to the different hardware specifications and the diagonal preconditioner used in this section, which reduces the number of required iterations from 399 to 112 to achieve a relative accuracy

below 10^{-6} in the GMRES. Nevertheless, we note that the setup of $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ and the solution of the system (3.24) using this approximation of V_h requires only about fifteen minutes. The relative L^2 approximation error of the resulting approximate solution q_h is roughly 0.01237, while the L^2 best approximation error is 0.00837. Hence, we conclude that also in the second experiment the nearfield compression with the heuristic extension of the recompression proves to be sufficiently accurate and significantly more efficient than the standard FMM.

No. nodes	storage [GiB]	assembly time $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ $[s]$	solution time $[s]$
64	2711	415.3	490.4

Table 8.8: Storage requirements and execution times for the computations in the second numerical experiment of Section 8.6.2 where the linear system (3.24) is solved for a space-time tensor product mesh with 1 024 time steps in the time interval $(0, 0.25)$ and 42 888 triangles on the surface of a crankshaft using the fast method in Algorithm 8.4 with the heuristic extension of the recompression in Remark 8.13 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$) to approximate V_h.

Third and fourth experiment: The experiments from Section 7.5. In the last two experiments of this section we revisit the numerical experiments from Section 7.5 to study the effects of our nearfield compression in the case of temporally non-uniform meshes. To allow for a simple comparison we use the same FMM parameters as in that section and the additional parameters for the ACA nearfield compression listed at the beginning of Section 8.6 when solving the related boundary integral equations. Furthermore, we used the same hardware to run the computations, i.e. the local workstation Babbage for the first experiment and a single node of the VSC-4 cluster for the second experiment. The results are presented in Tables 8.9 and 8.10, where we compare the time-adaptive FMM ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,ta}}$) with the fast method in Algorithm 8.4 ($\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$), which differ only in the treatment of the temporal nearfield.

In both tables we clearly see the benefits of the additional nearfield compression. The storage requirements for the inadmissible blocks are drastically reduced. In relation, the reduction of the setup times is slightly less pronounced, since we approximate ACA admissible blocks first by the partially pivoted ACA before recompressing them. Nonetheless, the reduction is still significant. The GMRES iteration times are also reduced by a considerable amount in both experiments, which shows that a large part of the original application times is related to the application of inadmissible nearfield blocks. While the storage requirements and runtimes are reduced, the approximation quality of the obtained BEM solutions is almost identical for the two fast methods in both experiments. This clearly demonstrates the usefulness of our new nearfield compression scheme.

	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,ta}}$	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$
Storage nearfield V_h [GiB]	32.0	8.1
Setup time V_h $[s]$	217.6	108.0
GMRES it. time $[s]$	1.42	1.09
No. GMRES iterations	49	49
Total time $[s]$	286.9	161.4
Rel. L^2 error BEM	0.0530951	0.0530957

Table 8.9: Results obtained by revisiting the first experiment from Section 7.5 (exponential decay in time). The results labeled by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,ta}}$ are the same as in Table 7.2 when the time-adaptive FMM is used for the approximation of V_h. The results labeled by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are obtained by using the fast method in Algorithm 8.4 with the heuristic extension of the recompression in Remark 8.13.

	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,ta}}$	$\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$
Storage nearfield [GiB]	66.6	28.3
Setup time $[s]$	302.3	170.4
GMRES it. time $[s]$	1.16	0.77
No. GMRES iterations	98	98
Total time $[s]$	415.6	246.3
Rel. L^2 error BEM	0.013129	0.013130

Table 8.10: Results obtained by revisiting the second experiment from Section 7.5 (rapidly changing boundary data). The results labeled by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,ta}}$ are the same as in Table 7.3 when the time-adaptive FMM is used for the approximation of V_h. The results labeled by $\widetilde{\mathsf{V}}_h^{\mathrm{FMM,rNC}}$ are obtained by using the fast method in Algorithm 8.4 with the heuristic extension of the recompression in Remark 8.13.

Let us summarize our findings in the four experiments in this section. We have seen that the new fast methods with the additional nearfield compression can also be used in more general settings, where clusters corresponding to ACA admissible nearfield blocks contain more than a single time step. In the first experiment we observed that the approximation of the ACA admissible nearfield blocks was not optimal for the fast method in Algorithm 8.3 and failed for a relatively large part of the blocks. Nonetheless, the obtained matrix approximation proved to be more efficient and at the same time similarly accurate as the approximation obtained by the standard FMM in Algorithm 6.5. By identifying the blocks for which the ACA failed, we motivated that the heuristic extension of the recompression scheme in Remark 8.13

is applicable for the fast method in Algorithm 8.4. This recompression proved to be more efficient in the first experiment while being still similarly accurate. This impression was solidified in the other three numerical experiments, where we applied the fast method with the heuristic recompression for space-time tensor product meshes with more complicated spatial geometries and non-uniform temporal partitions. Together with the results from Section 8.6.1 the results in this section show that the additional nearfield compression proposed in this chapter is a useful addition to the standard space-time FMM in Chapter 5 and its extension to non-uniform temporal meshes in Chapter 7.

9 Conclusions and Outlook

In this work space-time boundary element methods for the solution of initial boundary value problems of the transient heat equation in three spatial dimensions have been considered. The main results are a new, rigorous proof of the integration by parts formula for the hypersingular boundary integral operator and several significant enhancements of an alternative version of the parabolic fast multipole method, namely an efficient parallelization in space and time, an extension for temporally adaptive meshes, and a novel compression strategy for the temporal nearfield.

After presenting some preliminary results in the first chapters, we provided a general version of the integration by parts formula for the hypersingular boundary integral operator of the heat equation in Chapter 4 together with a rigorous proof. We showed that the formula which is commonly used in the literature includes an integral term that is not well-defined because the related kernel function is not Lebesgue integrable. In the general formula provided in Chapter 4 this integral term is replaced by a bilinear form $b(\cdot,\cdot)$, which is well-defined for general functions in the considered Sobolev space $H^{1/2,1/4}(\Sigma)$, but difficult to evaluate due to its definition as a continuous extension. We derived a suitable formula for the evaluation of this bilinear form for certain types of functions including the typical tensor product discretization spaces. In this way, we justified the results found in the literature.

When it comes to fast methods for the application of BEM matrices for the heat equation we achieved several significant improvements. Our starting point was the pFMM in [52, 67, 68] which we covered in a slightly modified form in Chapter 5. While we considered the same expansion of the heat kernel and a similar clustering strategy to derive the FMM operations, we did not exploit the lower triangular block structure of the matrices to solve the related systems of linear equations in a forward-sweeping manner. Instead, we reorganized the operations to obtain a more common FMM that realizes the full matrix-vector multiplication at once. This allowed us to parallelize the method in Chapter 6 not only with respect to space but also with respect to time. For the parallelization we used a task based execution scheme where the temporal structure of the FMM is exploited to distribute the workload and to handle the communication between the involved processes efficiently. We implemented the parallel FMM in C++ using MPI and OpenMP. The parallel efficiency of our method and implementation was demonstrated in several numerical experiments where we used up to 256 nodes of the Salomon cluster in Ostrava (CZ) and showed close to optimal scalability. A refinement of the parallelization approach to incorporate an

additional decomposition and distribution in space to further improve the scalability is left for future considerations.

In Chapter 7 we extended the FMM to improve its performance when dealing with space-time tensor product meshes that are adaptive in time. For this purpose we considered temporally one-sided expansions of the heat kernel and analyzed the corresponding approximation errors. Based on these expansions we introduced new FMM operations to compress previously inadmissible blocks of the considered BEM matrices. The resulting time-adaptive FMM was parallelized similarly as the standard FMM and implemented in C++. In our numerical experiments we showed that it outperforms the standard FMM for space-time tensor product meshes that are adaptive in time.

To further improve the performance of the FMM we presented a novel method for the compression of the temporal nearfield in Chapter 8. Our method is based on an additional subdivision of inadmissible blocks of the BEM matrices in the FMM and subsequent low-rank approximations obtained by using the partially pivoted ACA. In the description we focused on the single layer operator matrix V_h. By showing that certain blocks of this matrix are generated by an asymptotically smooth spatial kernel function we motivated that the ACA can be applied for their approximation. We suggested using a truncated SVD with a novel truncation criterion to recompress the low-rank matrices obtained by the ACA and analyzed the related approximation errors. In various numerical experiments we showed that the resulting nearfield compression scheme can significantly improve the compression achieved by the FMM not only for space-time meshes with a fine spatial resolution but also for more general meshes. We are confident that the presented nearfield compression method can also be used for other BEM matrices than V_h if a more sophisticated row selection strategy is used in the ACA as discussed in Remark 8.1.

The improvements of the FMM achieved in this work are an important step towards the development of efficient adaptive space-time boundary element methods for the heat equation. First results regarding a related adaptive BEM can be found in [31]. In that work, a posteriori error estimates are introduced which drive the non-local mesh refinement of the proposed adaptive algorithm. The adaptive algorithm and the considered numerical examples cover only the spatially two-dimensional case. When transferring the ideas to the spatially three-dimensional case, a fast method is needed for the solution of the related systems because the problem size is prohibitively limited otherwise. The numerical experiments in this thesis suggest that our enhanced FMM provides a good matrix compression also for temporally or spatially adaptive meshes and, therefore, it is suitable for use in an adaptive BEM. Depending on the structure of the adaptive meshes, further modifications of the method might be necessary, though. In fact, the new temporally one-sided FMM operations in the time-adaptive FMM exploit the tensor-product structure of the considered meshes

and need to be modified for more general meshes, in particular tetrahedral meshes or other non-prismatic meshes. The black box nature of the ACA might allow us to use our nearfield compression scheme also for this kind of meshes, but further work is necessary to justify the application in such situations. However, before tetrahedral meshes can be considered in an adaptive BEM, suitable quadrature routines need to be developed to efficiently evaluate the occurring singular integrals for matrix entries corresponding to neighboring elements. This is a challenging task in itself.

There are a variety of other interesting topics which one could consider in future works. For example, we have not presented an FMM for the initial potential operator matrices M_h^0 and M_h^1 in this work, because they differ considerably from the other BEM matrices due to the integrals over the domain Ω appearing in their definition. Nonetheless, the same concepts used to derive the FMM in this work can be used to derive an FMM for these operators if an additional cluster tree for the volume mesh Ω_h is constructed. In fact, we even used such a fast method in the numerical experiments in Chapters 7 and 8. However, a new parallelization strategy needs to be developed for these fast methods, because the parallelization strategy in Chapter 6 cannot be directly applied due to the structural differences between the methods. The development and analysis of new preconditioning strategies for the solution of the linear systems in the space-time BEM and the consideration of higher order polynomial discretization spaces are also interesting research topics that are left for future considerations.

A Hardware and compiler specifications

The following hardware and compilers were used to run the numerical experiments in this thesis.

Babbage is a local workstation at the Institute of Applied Mathematics at TU Graz, Austria. It is equipped with two 16-core Intel Xeon Gold 5218 processors and 384 GiB of RAM. For the compilation on Babbage we used the Intel compiler v2021.2.0.

Salomon was a supercomputer at IT4Innovations National Supercomputing Center in Ostrava, Czech Republic. It consisted of 1009 compute nodes equipped with two 12-core Intel Xeon E5-2680v3 processors and 128 GiB of RAM that were interconnected by InfiniBand FDR networks. It was operated until the end of 2021. For the compilation on Salomon we used the Intel compiler v19.1.1.

VSC-4 is a supercomputer at the Vienna Scientific Cluster in Vienna, Austria. It consists of 790 compute nodes equipped with two 24-core Intel Skylake Platinum 8174 processors that are interconnected with 100 Gbit/s OmniPath networks. 700 of the nodes at VSC-4 are standard nodes with 96 GiB of RAM, 78 are fat nodes with 384 GiB of RAM, and 12 are very fat nodes with 768 GiB of RAM. For the compilation on VSC-4 we used the Intel compiler v19.1.3.304.

References

[1] M. Abduljabbar, M. Al Farhan, N. Al-Harthi, R. Chen, R. Yokota, H. Bagci, and D. Keyes, *Extreme scale FMM-accelerated boundary integral equation solver for wave scattering*, SIAM J. Sci. Comput. 41, 3 (2019), pp. C245–C268, https://doi.org/10.1137/18M1173599.

[2] M. Abduljabbar, G. S. Markomanolis, H. Ibeid, R. Yokota, and D. Keyes, *Communication reducing algorithms for distributed hierarchical N-body problems with boundary distributions*, in High Performance Computing. ISC High Performance 2017, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, eds., Vol. 10266, Lecture Notes in Computer Science, Cham, 2017, Springer, pp. 79–96, https://doi.org/10.1007/978-3-319-58667-0_5.

[3] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 9th Dover printing ed., 1964.

[4] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner, and T. Takahashi, *Task-based FMM for multicore architectures*, SIAM J. Sci. Comput. 36, 1 (2014), pp. C66–C93, https://doi.org/10.1137/130915662.

[5] E. Agullo, B. Bramas, O. Coulaud, M. Khannouz, and L. Stanisic, *Task-based fast multipole method for clusters of multicore processors*, research report, RR8970, hal-01387482v4, Inria Bordeaux Sud-Ouest, 2017, https://hal.archives-ouvertes.fr/hal-01387482.

[6] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 3rd ed., 1999, https://doi.org/10.1137/1.9780898719604.

[7] W. Arendt, C. J. K. Batty, M. Hieber, and F. Neubrander, *Vector-valued Laplace Transforms and Cauchy Problems*, No. 96 in Monographs in Mathematics, Birkhäuser, Basel, 2nd ed., 2011, https://doi.org/10.1007/978-3-0348-0087-7.

[8] D. N. Arnold and P. J. Noon, *Coercivity of the single layer heat potential*, J. Comput. Math. 7, 2 (1989), pp. 100–104.

[9] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, *StarPU: A unified platform for task scheduling on heterogeneous multicore architectures*, Concurrency Computat.: Pract. Exper. 23 (2011), pp. 187–198, https://doi.org/10.1002/cpe.1631.

[10] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math. 86, 4 (2000), pp. 565–589, https://doi.org/10.1007/PL00005410.

[11] M. Bebendorf, *Hierarchical Matrices*, No. 63 in Lecture Notes in Computational Science and Engineering, Springer, Berlin, Heidelberg, 2008, https://doi.org/10.1007/978-3-540-77147-0.

[12] M. Bebendorf and S. Kunis, *Recompression techniques for adaptive cross approximation*, Journal of Integral Equations and Applications 21, 3 (2009), pp. 331 – 357, https://doi.org/10.1216/JIE-2009-21-3-331.

[13] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing 70, 1 (2003), pp. 1–24, https://doi.org/10.1007/s00607-002-1469-6.

[14] S. Börm, *Efficient Numerical Methods for Non-local Operators*, No. 14 in Tracts in Mathematics, European Mathematical Society, Zürich, 2010, https://doi.org/10.4171/091.

[15] S. Börm and J. Bendoraityte, *Distributed $\mathcal{H}^2$-matrices for non-local operators*, Comput. Visual. Sci. 11 (2008), pp. 237–249, https://doi.org/10.1007/s00791-008-0095-z.

[16] C. Bourgeois and S. Nicaise, *Biorthogonal wavelet approximation methods for the heat equation*, in Problems and Methods in Mathematical Physics. Operator Theory: Advances and Applications, J. Elschner, I. Gohberg, and B. Silbermann, eds., Vol. 121, Basel, 2001, Birkhäuser, https://doi.org/10.1007/978-3-0348-8276-7_6.

[17] C. Bourgeois and R. Schneider, *Biorthogonal wavelets for the direct integral formulation of the heat equation*. Preprint, SFB393/00-14, 2000.

[18] J. Breuer, *Schnelle Randelementmethoden zur Simulation von elektrischen Wirbelstromfeldern sowie ihrer Wärmeproduktion und Kühlung*, PhD thesis, Universität Stuttgart, 2005, https://doi.org/10.18419/opus-4746.

[19] R. M. Brown, *The method of layer potentials for the heat equation in Lipschitz cylinders*, Amer. J. Math. 111, 2 (1989), pp. 339–379, https://doi.org/10.2307/2374513.

[20] S. Börm and C. Mehl, *Numerical Methods for Eigenvalue Problems*, De Gruyter, Berlin, Boston, 2012, https://doi.org/10.1515/9783110250374.

[21] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Stat. Comput. 9, 4 (1988), pp. 669–686, https://doi.org/10.1137/0909044.

[22] H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys. 155, 2 (1999), pp. 468–498, https://doi.org/10.1006/jcph.1999.6355.

[23] A. Chernov and A. Reinarz, *Sparse grid approximation spaces for space-time boundary integral formulations of the heat equation*, Comput. Math. Appl. 78, 11 (2019), pp. 3605–3619, https://doi.org/10.1016/j.camwa.2019.06.036.

[24] M. Costabel, *Boundary integral operators for the heat equation*, Integral Equations and Operator Theory 13, 4 (1990), pp. 498–552, https://doi.org/10.1007/BF01210400.

[25] F. Cruz, M. Knepley, and L. Barba, *PetFMM—A dynamically load-balancing parallel fast multipole library*, Int. J. Numer. Methods. Eng. 85, 4 (2011), pp. 403–428, https://doi.org/10.1002/nme.2972.

[26] J. Diestel and J. J. Uhl, *Vector Measures*, No. 15 in Mathematical Surveys and Monographs, American Mathematical Society, Providence, 1977.

[27] S. Dohr, *Distributed and Preconditioned Space-Time Boundary Element Methods for the Heat Equation*, PhD thesis, Graz University of Technology, 2019, https://tinyurl.com/dohrphdthesis.

[28] S. Dohr, K. Niino, and O. Steinbach, *Space-time boundary element methods for the heat equation*, in Space-time methods – applications to partial differential equations, U. Langer and O. Steinbach, eds., No. 25 in Radon Series on Computational and Applied Mathematics, De Gruyter, Berlin, 2019, pp. 1–60, https://doi.org/10.1515/9783110548488.

[29] S. Dohr, J. Zapletal, G. Of, M. Merta, and M. Kravčenko, *A parallel space-time boundary element method for the heat equation*, Comput. Math. Appl. 78, 9 (2019), pp. 2852–2866, https://doi.org/10.1016/j.camwa.2018.12.031.

[30] L. C. Evans, *Partial Differential Equations*, Oxford University Press, Oxford, 2nd ed., 1998.

[31] G. Gantner and R. van Venetië, *Adaptive space-time BEM for the heat equation*, Comput. Math. Appl. 107 (2022), pp. 117–131, https://doi.org/10.1016/j.camwa.2021.12.022.

[32] L. Greengard and P. Lin, *Spectral approximation of the free-space heat kernel*, Appl. Comput. Harmon. Anal. 9, 1 (2000), pp. 83–97, https://doi.org/10.1006/acha.2000.0310.

[33] L. Greengard and J. Strain, *A fast algorithm for the evaluation of heat potentials*, Comm. Pure Appl. Math. 43, 8 (1990), pp. 949–963, https://doi.org/10.1002/cpa.3160430802.

[34] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, No. 69 in Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 2011.

[35] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Vol. 49, Springer Series in Computational Mathematics, Springer, Heidelberg, 2015, https://doi.org/10.1007/978-3-662-47324-5.

[36] H. D. Han, *The boundary integro-differential equations of three-dimensional Neumann problem in linear elasticity*, Numer. Math. 68, 2 (1994), pp. 269–281, https://doi.org/10.1007/s002110050061.

[37] H. Harbrecht and J. Tausch, *A fast sparse grid based space-time boundary element method for the nonstationary heat equation*, Numer. Math. 140, 1 (2018), pp. 239–264, https://doi.org/10.1007/s00211-018-0963-5.

[38] M. Karkulik, G. Of, and D. Praetorius, *Convergence of adaptive 3D BEM for weakly singular integral equations based on isotropic mesh-refinement*, Numer. Methods Partial Differ. Equ. 29, 6 (2013), pp. 2081–2106, https://doi.org/10.1002/num.21792.

[39] G. Kratochwill, *Analyse und Vergleich verschiedener Varianten der schnellen Gauß–Transformation*, Master's thesis, Technische Universität Graz, Graz, 2018, https://permalink.obvsg.at/tug/AC15175955.

[40] M. Kravčenko, M. Merta, and J. Zapletal, *Distributed fast boundary element methods for Helmholtz problems*, Appl. Math. Comput. 362 (2019), 124503, https://doi.org/10.1016/j.amc.2019.06.017.

[41] V. D. Kupradze, T. G. Gegelia, M. O. Basheleishvili, and T. V. Burchuladze, *Three-Dimensional Problems of the Mathematical Theory of Elasticity and Thermoelasticity*, No. 25 in North-Holland Series in applied Mathematics and Mechanics, North-Holland, Amsterdam, 1979.

[42] S. Larsson and C. Schwab, *Compressive space-time Galerkin discretizations of parabolic partial differential equations*. arxiv:1501.04514, 2015, https://arxiv.org/abs/1501.04514.

[43] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros, *A massively parallel adaptive fast-multipole method on heterogeneous architectures*, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, New York, 2009, Association for Computing Machinery, pp. 1–12, https://doi.org/10.1145/1654059.1654118.

[44] J. L. Lions and E. Magenes, *Non-Homogeneous Boundary Value Problems and Applications*, Vol. I, No. 181 in Grundl. Math. Wissen., Springer, Berlin-Heidelberg-New York, 1972, https://doi.org/10.1007/978-3-642-65161-8.

[45] J. L. Lions and E. Magenes, *Non-Homogeneous Boundary Value Problems and Applications*, Vol. II, No. 182 in Grundl. Math. Wissen., Springer, Berlin-Heidelberg-New York, 1972, https://doi.org/10.1007/978-3-642-65393-3.

[46] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, Chapman and Hall/CRC, New York, 1st ed., 2002, https://doi.org/10.1201/9781420036114.

[47] A. Maue, *Zur Formulierung eines allgemeinen Beugungsproblems durch eine Integralgleichung*, Z. Phys. 126 (1949), pp. 601–618.

[48] W. McLean, *Strongly Elliptic Systems and Boundary Integral Equations*, Cambridge University Press, Cambridge, 2000.

[49] M. Merta, G. Of, R. Watschinger, and J. Zapletal, *besthea*. https://github.com/zap150/besthea, 2020.

[50] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard; Version 3.1*, 2015, https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf. [Online; accessed May 12 2022].

[51] M. Messner, *A Fast Multipole Galerkin Boundary Element Method for the Transient Heat Equation*, Vol. 23, Monographic Series TU Graz: Computation in Engineering and Science, Verlag der Technischen Universität Graz, Graz, 2014, https://doi.org/10.3217/978-3-85125-350-4.

[52] M. Messner, M. Schanz, and J. Tausch, *A fast Galerkin method for parabolic space-time boundary integral equations*, J. Comput. Phys. 258 (2014), pp. 15–30, https://doi.org/10.5555/2799696.2799938.

[53] M. Messner, M. Schanz, and J. Tausch, *An efficient Galerkin boundary element method for the transient heat equation*, SIAM J. Sci. Comput. 37, 3 (2015), pp. A1554–A1576, https://doi.org/10.1137/151004422.

[54] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White, *Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory*, SIAM J. Sci. Comput. 15, 3 (1994), pp. 713–735, https://doi.org/10.1137/0915046.

[55] J.-C. Nédélec, *Integral equations with non integrable kernels*, Integral Equations and Operator Theory 5 (1982), pp. 562–572, https://doi.org/10.1007/BF01694054.

[56] J.-C. Nédélec, *Acoustic and electromagnetic equations*, Vol. 144, Applied Mathematical Sciences, Springer, New York, 2001, https://doi.org/10.1007/978-1-4757-4393-7.

[57] P. J. Noon, *The Single Layer Heat Potential and Galerkin Boundary Element Methods for the Heal Equation*, PhD thesis, University of Maryland, 1988.

[58] OpenMP Architecture Review Board, *OpenMP Application Programming Interface 5.0 Specification*, 2018, https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf. [Online; accessed May 12 2022].

[59] W. Pogorzelski, *Integral Equations and Their Applications*, Vol. I, No. 88 in International Series of Monographs in Pure and Applied Mathematics, Pergamon Press, Oxford, 1966.

[60] A. Reinarz, *Sparse Space-time Boundary Element Methods for the Heat Equation*, PhD thesis, University of Reading, 2015, https://centaur.reading.ac.uk/49315.

[61] T. J. Rivlin, *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*, Wiley, New York, 2nd ed., 1990.

[62] S. Rjasanow and O. Steinbach, *The Fast Solution of Boundary Integral Equations*, Mathematical and Analytical Techniques with Applications to Engineering, Springer, New York, 1st ed., 2007, https://doi.org/10.1007/0-387-34042-4.

[63] S. A. Sauter and C. Schwab, *Boundary Element Methods*, Vol. 39, Springer Series in Computational Mathematics, Springer, Berlin, Heidelberg, 2010, https://doi.org/10.1007/978-3-540-68093-2.

[64] F.-J. Sayas, T. S. Brown, and M. E. Hassell, *Variational Techniques for Elliptic Partial Differential Equations*, Taylor & Francis Group, Boca Raton, 2019, https://doi.org/10.1201/9780429507069.

[65] J. Schuchart, K. Tsugane, J. Gracia, and M. Sato, *The impact of taskyield on the design of tasks communicating through MPI*, in Evolving OpenMP for Evolving Architectures. IWOMP 2018, B. R. de Supinski, P. Valero-Lara, X. Martorell, S. Mateo Bellido, and J. Labarta, eds., Vol. 11128, Lecture Notes in Computer Science, Cham, 2018, Springer, pp. 3–17, https://doi.org/10.1007/978-3-319-98521-3_1.

[66] O. Steinbach, *Numerical Approximation Methods for Elliptic Boundary Value Problems. Finite and Boundary Elements*, Springer, New York, 2008, https://doi.org/10.1007/978-0-387-68805-3.

[67] J. Tausch, *A fast method for solving the heat equation by layer potentials*, J. Comput. Phys. 224, 2 (2007), pp. 956–969, https://doi.org/10.1016/j.jcp.2006.11.001.

[68] J. Tausch, *Fast Nyström methods for parabolic boundary integral equations*, in Fast Boundary Element Methods in Engineering and Industrial Applications, U. Langer, M. Schanz, O. Steinbach, and W. Wendland, eds., No. 63 in Lecture Notes in Applied and Computational Mechanics, Springer, Berlin, Heidelberg, 2012, pp. 185–219, https://doi.org/10.1007/978-3-642-25670-7_6.

[69] J. Tausch and A. Weckiewicz, *Multidimensional fast Gauss transforms by Chebyshev expansions*, SIAM J. Sci. Comput. 31, 5 (2009), pp. 3547–3565, https://doi.org/10.1137/080732729.

[70] S. Thomas, A. Matthew, and B. Maciej, *High Performance Computing. Modern Systems and Practices*, Morgan Kaufmann, Cambridge, 2018, https://doi.org/10.1016/C2013-0-09704-6.

[71] F. Treves, *Topological Vector Spaces, Distributions and Kernels*, Academic Press, New York, 3rd ed., 1970.

[72] G. C. Verchota, *Layer potentials and boundary value problems for Laplace's equation on Lipschitz domains*, PhD thesis, University of Minnesota, 1982, https://www.proquest.com/docview/303065322.

[73] G. C. Verchota, *Layer potentials and regularity for the Dirichlet problem for Laplace's equation in Lipschitz domains*, J. Funct. Anal. 59, 3 (1984), pp. 572–611, https://doi.org/10.1016/0022-1236(84)90066-1.

[74] S. Wang, *The Boundary Element Method for Parabolic Equation and Its Implementation in BEM++*, PhD thesis, Southern Methodist University, 2020, https://scholar.smu.edu/hum_sci_mathematics_etds/7.

[75] M. S. Warren and J. K. Salmon, *Astrophysical N-body simulations using hierarchical tree data structures*, in Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, Washington, DC, 1992, IEEE, pp. 570–576, https://doi.org/10.1109/SUPERC.1992.236647.

[76] R. Watschinger, M. Merta, G. Of, and J. Zapletal, *A parallel fast multipole method for a space-time boundary element method for the heat equation*, SIAM J. Sci. Comput. 44, 4 (2022), pp. C320–C345, https://doi.org/10.1137/21M1430157.

[77] R. Watschinger and G. Of, *An integration by parts formula for the bilinear form of the hypersingular boundary integral operator for the transient heat equation in three spatial dimensions*, J. Integral Equ. Appl. 34, 1 (2022), pp. 103–133, https://doi.org/10.1216/jie.2022.34.103.

[78] R. Watschinger and G. Of, *A time-adaptive space-time FMM for the heat equation*, Comput. Methods Appl. Math (2022), https://doi.org/https://doi.org/10.1515/cmam-2022-0117.

[79] R. Yokota and L. A. Barba, *A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems*, J. High Perform. Comput. Appl. 26, 4 (2012), pp. 337–346, https://doi.org/10.1177/1094342011429952.

[80] M. Zank, *Inf-Sup Stable Space-Time Methods for Time-Dependent Partial Differential Equations*, Vol. 36, Monographic Series TU Graz: Computation in Engineering and Science, Verlag der Technischen Universität Graz, Graz, 2019, https://doi.org/10.3217/978-3-85125-721-2.

[81] J. Zapletal, R. Watschinger, G. Of, and M. Merta, *Semi-analytic integration for a parallel space-time boundary element method modeling the heat equation*, Comput. Math. Appl. 103 (2021), pp. 156–170, https://doi.org/10.1016/j.camwa.2021.10.025.

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 1 Steffen Alvermann

Effective Viscoelastic Behavior of Cellular Auxetic Materials
2008
ISBN 978-3-902465-92-4

Vol. 2 Sendy Fransiscus Tantono

The Mechanical Behaviour of a Soilbag under Vertical Compression
2008
ISBN 978-3-902465-97-9

Vol. 3 Thomas Rüberg

Non-conforming FEM/BEM Coupling in Time Domain
2008
ISBN 978-3-902465-98-6

Vol. 4 Dimitrios E. Kiousis

Biomechanical and Computational Modeling of Atherosclerotic Arteries
2008
ISBN 978-3-85125-023-7

Vol. 5 Lars Kielhorn

A Time-Domain Symmetric Galerkin BEM for Viscoelastodynamics
2009
ISBN 978-3-85125-042-8

Vol. 6 Gerhard Unger

Analysis of Boundary Element Methods for Laplacian Eigenvalue Problems
2009
ISBN 978-3-85125-081-7

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 7 Gerhard Sommer

Mechanical Properties of Healthy and Diseased Human Arteries
2010
ISBN 978-3-85125-111-1

Vol. 8 Mathias Nenning

Infinite Elements for Elasto- and Poroelastodynamics
2010
ISBN 978-3-85125-130-2

Vol. 9 Thanh Xuan Phan

Boundary Element Methods for Boundary Control Problems
2011
ISBN 978-3-85125-149-4

Vol. 10 Loris Nagler

Simulation of Sound Transmission through Poroelastic Plate-like Structures
2011
ISBN 978-3-85125-153-1

Vol. 11 Markus Windisch

Boundary Element Tearing and Interconnecting Methods for Acoustic and Electromagnetic Scattering
2011
ISBN 978-3-85125-152-4

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 12 Christian Walchshofer

Analysis of the Dynamics at the Base of a Lifted Strongly Buoyant Jet Flame Using Direct Numerical Simulation
2011
ISBN 978-3-85125-185-2

Vol. 13 Matthias Messner

Fast Boundary Element Methods in Acoustics
2012
ISBN 978-3-85125-202-6

Vol. 14 Peter Urthaler

Analysis of Boundary Element Methods for Wave Propagation in Porous Media
2012
ISBN 978-3-85125-216-3

Vol. 15 Peng Li

Boundary Element Method for Wave Propagation in Partially Saturated Poroelastic Continua
2012
ISBN 978-3-85125-236-1

Vol. 16 Andreas Jörg Schriefl

Quantification of Collagen Fiber Morphologies in Human Arterial Walls
2013
ISBN 978-3-85125-238-5

Vol. 17 Thomas S. E. Eriksson

Cardiovascular Mechanics
2013
ISBN 978-3-85125-277-4

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 18 Jianhua Tong

Biomechanics of Abdominal Aortic Aneurysms
2013
ISBN 978-3-85125-279-8

Vol. 19 Jonathan Rohleder

Titchmarsh–Weyl Theory and Inverse Problems for Elliptic Differential Operators
2013
ISBN 978-3-85125-283-5

Vol. 20 Martin Neumüller

Space-Time Methods
2013
ISBN 978-3-85125-290-3

Vol. 21 Michael J. Unterberger

Microstructurally-Motivated Constitutive Modeling of Cross-Linked Filamentous Actin Networks
2013
ISBN 978-3-85125-303-0

Vol. 22 Vladimir Lotoreichik

Singular Values and Trace Formulae for Resolvent Power Differences of Self-Adjoint Elliptic Operators
2013
ISBN 978-3-85125-304-7

Vol. 23 Michael Meßner

A Fast Multipole Galerkin Boundary Element Method for the Transient Heat Equation
2014
ISBN 978-3-85125-350-4

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 24 Lorenz Johannes John

Optimal Boundary Control in Energy Spaces
2014
ISBN 978-3-85125-373-3

Vol. 25 Hannah Weisbecker

Softening and Damage Behavior of Human Arteries
2014
ISBN 978-3-85125-370-2

Vol. 26 Bernhard Kager

Efficient Convolution Quadrature based Boundary Element Formulation for Time-Domain Elastodynamics
2015
ISBN 978-3-85125-382-5

Vol. 27 Christoph M. Augustin

Classical and All-floating FETI Methods with Applications to Biomechanical Models
2015
ISBN 978-3-85125-418-1

Vol. 28 Elias Karabelas

Space-Time Discontinuous Galerkin Methods for Cardiac Electromechanics
2016
ISBN 978-3-85125-461-7

Vol. 29 Thomas Traub

A Kernel Interpolation Based Fast Multipole Method for Elastodynamic Problems
2016
ISBN 978-3-85125-465-5

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 30 Matthias Gsell

Mortar Domain Decomposition Methods for Quasilinear Problems and Applications
2017
ISBN 978-3-85125-522-5

Vol. 31 Christian Kühn

Schrödinger operators and singular infinite rank perturbations
2017
ISBN 978-3-85125-551-5

Vol. 32 Michael H. Gfrerer

Vibro-Acoustic Simulation of Poroelastic Shell Structures
2018
ISBN 978-3-85125-573-7

Vol. 33 Markus Holzmann

Spectral Analysis of Transmission and Boundary Value Problems for Dirac Operators
2018
ISBN 978-3-85125-642-0

Vol. 34 Osman Gültekin

Computational Inelasticity of Fibrous Biological Tissues with a Focus on Viscoelasticity, Damage and Rupture
2019
ISBN 978-3-85125-655-0

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 35 Justyna Anna Niestrawska

Experimental and Computational Analyses of Pathological Soft Tissues – Towards a Better Understanding of the Pathogenesis of AAA
2019
ISBN 978-3-85125-678-9

Vol. 36 Marco Zank

Inf-Sup Stable Space-Time Methods for Time-Dependent Partial Differential Equations
2020
ISBN 978-3-85125-721-2

Vol. 37 Christoph Irrenfried

Convective turbulent near wall heat transfer at high Prandtl numbers
2020
ISBN 978-3-85125-724-3

Vol. 38 Christopher Albert

Hamiltonian Theory of Resonant Transport Regimes in Tokamaks with Perturbed Axisymmetry
2020
ISBN 978-3-85125-746-5

Vol. 39 Daniel Christopher Haspinger

Material Modeling and Simulation of Phenomena at the Nano, Micro and Macro Levels in Fibrous Soft Tissues of the Cardiovascular System
2021
ISBN 978-3-85125-802-8

Monographic Series TU Graz

Computation in Engineering and Science

Vol. 40 Markus Alfons Geith

Percutaneous Coronary Intervention
2021
ISBN 978-3-85125-801-1

Vol. 41 Dominik Pölz

Space-Time Boundary Elements for Retarded Potential Integral Equations
2021
ISBN 978-3-85125-811-0

Vol. 42 Douglas Ramalho Queiroz Pacheco

Stable and stabilised finite element methods for incompressible flows of generalised Newtonian fluids
2021
ISBN 978-3-85125-856-1

Vol. 43 Peter Schlosser

Superoscillations and their Schrödinger time evolution
2022
ISBN 978-3-85125-930-8

Vol. 44 Raphael Watschinger

Fast space-time boundary element methods for the heat equation
2023
ISBN 978-3-85125-949-0