Journal of Computational Physics 258 (2014) 809–832

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An interpolation-based fast-multipole accelerated boundary integral equation method for the three-dimensional wave equation

Toru Takahashi [*]

Department of Mechanical Science and Engineering, Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8603, Japan

A R T I C L E   I N F O

Article history:
Received 14 May 2013
Received in revised form 23 October 2013
Accepted 11 November 2013
Available online 18 November 2013

Keywords:
Boundary integral equation method
Boundary element method
Fast multipole method
Wave equation
Interpolation
Time domain

A B S T R A C T

A new fast multipole method (FMM) is proposed to accelerate the time-domain boundary integral equation method (TDBIEM) for the three-dimensional wave equation. The proposed algorithm is an enhancement of the interpolation-based FMM for the time-domain case, adopting the notion of the plane-wave time-domain algorithm. With the application being targeted at a low-frequency regime, the proposed time-domain interpolation-based FMM can reduce the computational complexity of the TDBIEM from $\mathcal{O}(N_s^2 N_t)$ to $\mathcal{O}(N_s^{1+\delta} N_t)$ (where $\delta = 1/3$ or $1/2$) with the help of multilevel space–time hierarchy, where N_s and N_t are the spatial and temporal degrees of freedom, respectively. The computational accuracy and speed of the proposed accelerated TDBIEM are verified in comparison with those of the conventional (direct) TDBIEM via numerical experiments.

1. Introduction

The application of the fast multipole method (FMM) [1–4] to accelerate the time-domain boundary integral equation method (TDBIEM) still remains a challenge, whereas its frequency-domain counterpart (including the static limit) has been investigated extensively [5].

In particular, for wave problems, the plane-wave time-domain (PWTD) algorithm is known as the time-domain version of the FMM [6]. The core of the PWTD algorithm is the plane-wave expansion of the fundamental solution (or the retarded Green's function for free space) of the wave equation. This expansion corresponds to the multipole expansion (in ordinary FMMs) that allows a given kernel $K(x, y)$ to be re-expressed as a degenerate form such as $\sum_n a_n(x)b_n(y)$, where x and y represent the positions of the target and source, respectively [7]. Using the plane-wave expansion, one can develop a fast method to evaluate the retarded potential or time-dependent pairwise interactions among targets and sources by systematically grouping them in space–time. The computational cost of the conventional integral equation (IE) solver (that adopts an ordinary marching-on-in-time (MOT) scheme) is $\mathcal{O}(N_s^2 N_t)$, whereas that of the multilevel PWTD-enhanced IE solver scales as $\mathcal{O}(N_s \log^n N_s \cdot N_t)$, where N_s and N_t denote the spatial and temporal degrees of freedom, respectively. The value of n is two for the complete spherical expansion and one for finite-cone representations [8]. Thus far, the PWTD algorithm has been successfully applied to accelerate the time-domain IE solvers for electromagnetics [6,8–12], acoustics [13,14], and elastodynamics [15,16]. Concurrently, the accelerated Cartesian expansion (ACE) algorithm is proposed to resolve the

* Tel./fax: +81 52 789 5333.
E-mail address: ttaka@nuem.nagoya-u.ac.jp.

0021-9991/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.11.008

breakdown of the PWTD algorithm in the low-frequency regime [17]. The ACE algorithm uses the Taylor expansion instead of the plane-wave expansion and attains a computational complexity of $\mathcal{O}(N_\mathrm{s}N_\mathrm{t})$ in that regime.

The proposed algorithm is similar to the PWTD algorithm; in particular, the mechanism of *time-gating*, which allows to a set of elapsed time-steps to be managed together at the current time-step (see Section 4), belongs to the PWTD algorithm. Specifically, because the low-frequency regime is the prime target, the proposed algorithm can be identified as an alternative for the ACE algorithm rather than that for the PWTD algorithm. However, the proposed algorithm uses neither the plane-wave expansion nor the Taylor expansion. Instead, it interpolates the fundamental solution with respect to the spatial and temporal variables for the target and source in order to reduce the fundamental solution to a degenerate form. Using this approach, one can develop another fast algorithm with a computational complexity of $\mathcal{O}(N_\mathrm{s}^{1+\delta}N_\mathrm{t})$ (where $\delta = 1/3$ or $1/2$; see Section 4.4). This algorithm is slightly slower than the PWTD and ACE algorithms but is faster than the conventional algorithm.

The class of FMMs that exploit interpolation, as in this study, is termed the *interpolation-based FMM*. In general, an FMM in this class is kernel-independent, i.e. it can manage a certain class of kernels (or pairwise interactions) in a single formulation. (Note that interpolation is not the only method to construct kernel-independent FMMs; see [18] for example.) In addition, broadly speaking, the mathematical formulation and numerical implementation of the interpolation-based FMMs are simpler than those of the classical FMMs specialised for individual kernels (e.g. [1–4,6]).

Presently, the class of FMMs includes the one-dimensional FMM [19], the black-box FMM [20], the directional FMM [21], and the FMM for low-frequency three-dimensional (3D) electromagnetics [22]. These FMMs correspond to the static or frequency-domain problems. Meanwhile, the concept of the interpolation-based FMM was extended to the transient heat problem in [23], where a time-domain FMM for the heat kernel was proposed to reduce the computational cost from $\mathcal{O}(N_\mathrm{s}^2N_\mathrm{t}^2)$ to $\mathcal{O}(N_\mathrm{s}N_\mathrm{t})$ in order to evaluate the heat potential.

From the viewpoint of kernel independency, the proposed FMM for the wave equation can be regarded as a variant of the time-domain interpolation-based FMM for the heat equation in [23]. In fact, these FMMs have some similarities in their formulations. The difference essentially originates in the properties of the kernels in relevant boundary integral equations. To be specific, the heat kernel is very smooth and decays exponentially when the distance between the target and source increases, i.e. it decays as $\exp(-r^2/(4T))$, where r and T denote distances in space and time, respectively. Hence, one can ignore pairwise interactions when r is sufficiently large. In contrast, the kernel for the wave equation is always nondifferentiable on the wavefront and decays more slowly, i.e. $1/r$. Therefore, one needs to consider a formulation that is different from the heat equation, especially in the multipole-to-local (M2L) operation.

The remainder of this paper is organised as follows: Section 2 summarises the basics of the conventional MOT-based TDBIEM for the wave equation. Section 3 highlights the mathematical aspect of the proposed interpolation-based FMM used to accelerate the TDBIEM. In Section 4, from the formulae derived in Section 3, the proposed algorithm is developed in a similar manner of the multilevel PWTD algorithm. In Section 5, the constructed fast TDBIEM is numerically investigated to validate its computational accuracy and time in comparison with the conventional TDBIEM.

2. Time-domain boundary integral equation method for the wave equation

This section summarises the conventional MOT-based TDBIEM (e.g. [24]), which is accelerated using the fast algorithm that will be described in Sections 3 and 4. Following the accurate and memory-efficient TDBIEM for elastodynamics [25,26], this study considers a TDBIEM (for the wave equation) that is based on the collocation method that uses the piecewise-constant and piecewise-linear bases for space and time, respectively. The selection of these bases is not absolutely necessary, but the formulation of the proposed fast algorithm depends on the selection to some extent.

2.1. Boundary integral equation

Let D be a domain in $\mathbb{R}^3$ with the piecewise-smooth boundary ∂D and let t denote time. Suppose that D is filled with a homogeneous and isotropic medium with a constant c (> 0), which is the wave (phase) velocity. The concerned initial–boundary value problem is to solve the field u from the wave equation

$$\triangle u(\boldsymbol{x}, t) = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}(\boldsymbol{x}, t) \quad \text{for } \boldsymbol{x} \in D,\ t > 0 \tag{1}$$

subject to the null initial condition, i.e. $u = \frac{\partial u}{\partial t} = 0$ for $t \leqslant 0$, and the typical boundary conditions, i.e. either u or $q := \frac{\partial u}{\partial n}$ is given on each point in ∂D for $t > 0$, where $\boldsymbol{n}$ is the unit outward normal to ∂D.

As is well known, this problem is reduced to solve the following boundary integral equation (BIE):

$$\frac{1}{2}u(\boldsymbol{x}, t) = \int_0^t \int_{\partial D} \left(\Gamma(\boldsymbol{x} - \boldsymbol{y}, t - s) q(\boldsymbol{y}, s) - \frac{\partial \Gamma}{\partial n_y}(\boldsymbol{x}, \boldsymbol{y}, t - s) u(\boldsymbol{y}, s) \right) \mathrm{d}s\, \mathrm{d}S_y \quad \text{for } \boldsymbol{x} \in \partial D,\ t > 0, \tag{2}$$

where $\Gamma(\boldsymbol{x}, t) := \frac{\delta(t - |\boldsymbol{x}|/c)}{4\pi|\boldsymbol{x}|}$ denotes the fundamental solution with Dirac's delta function δ.

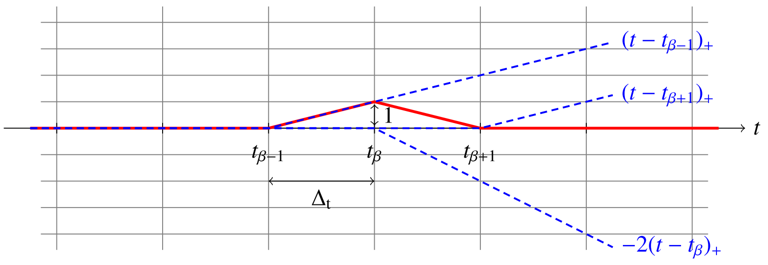


Fig. 1. Temporal piecewise-linear basis for $t = t_\beta$ (solid line). This basis can be decomposed to three truncated power functions (dashed lines).

Note that even if D is unbounded, the BIE in (2) holds without requiring any artificial condition, i.e. the absorbing boundary conditions (ABCs). This property has an advantage over major solvers, such as the finite-difference time-domain (FDTD) method and finite element method (FEM), because these methods always request such ABCs in the case of external problems. To analyse external scattering problems using the BIEM, one may append a given incident field to the right-hand side (RHS) of (2).

2.2. Discretisation

The collocation method is adopted to solve the BIE in (2). To discretise (2), the boundary is represented with N_s triangular boundary elements E_i ($1 \leqslant i \leqslant N_s$) and the temporal axis is segmented at $t_\alpha := \alpha\Delta_t$ ($\alpha = 0, 1, 2, \dots$), where Δ_t denotes a constant time-step size given *a priori*. The centre of E_i, which is denoted by $\boldsymbol{x}_i$, and t_α are selected as the spatial and temporal collocation points, respectively.

The boundary densities u and q are approximated with the piecewise-constant and piecewise-linear bases for space and time, respectively. For conciseness, u_i^α (respectively, q_i^α) denotes the value of u (respectively, q) on E_i at t_α. Then, the boundary density denoted by φ for u and q is expressed for any $\boldsymbol{x} \in E_i$ and $t > 0$ as follows:

$$\varphi(\boldsymbol{x}, t) \approx \sum_{\beta=1} \varphi_i^\beta \left(\frac{(t-t_{\beta-1})_+}{\Delta_t} - 2\frac{(t-t_\beta)_+}{\Delta_t} + \frac{(t-t_{\beta+1})_+}{\Delta_t} \right), \tag{3}$$

where each piecewise-linear basis is expressed by a linear combination of three truncated power functions with exponent one, which is denoted by $x_+ := \max(x, 0)$, to simplify the successive formulation (Fig. 1).

By substituting (3) in (2) and calculating the temporal integral, one can obtain the following discretised BIE at the αth time-step ($\alpha = 1, 2, \dots$):

$$\begin{aligned}\sum_{j=1}^{N_s} A_{ij} x_j^\alpha = \sum_{j=1}^{N_s} B_{ij} y_j^\alpha - \sum_{\beta=1}^{\alpha-1} \sum_{j=1}^{N_s} \Big(\big\{ W_{ij}^{(\alpha-\beta+1)} - 2W_{ij}^{(\alpha-\beta)} + W_{ij}^{(\alpha-\beta-1)} \big\} u_j^\beta \\ - \big\{ U_{ij}^{(\alpha-\beta+1)} - 2U_{ij}^{(\alpha-\beta)} + U_{ij}^{(\alpha-\beta-1)} \big\} q_j^\beta \Big) \quad \text{for } 1 \leqslant i \leqslant N_s,\end{aligned} \tag{4}$$

where

$$U_{ij}^{(\gamma)} := \frac{1}{\Delta_t} \int_{E_j} \int_0^{t_\gamma} \Gamma(\boldsymbol{x}_i - \boldsymbol{y}, t_\gamma - s) s \, \mathrm{d}s \, \mathrm{d}S_y = \frac{1}{4\pi c \Delta_t} \int_{E_j} \frac{(ct_\gamma - |\boldsymbol{x}_i - \boldsymbol{y}|)_+}{|\boldsymbol{x}_i - \boldsymbol{y}|} \mathrm{d}S_y, \tag{5a}$$

$$W_{ij}^{(\gamma)} := \frac{1}{\Delta_t} \int_{E_j} \int_0^{t_\gamma} \frac{\partial \Gamma}{\partial n_y}(\boldsymbol{x}_i, \boldsymbol{y}, t_\gamma - s) s \, \mathrm{d}s \, \mathrm{d}S_y = \frac{1}{4\pi c \Delta_t} \int_{E_j} \frac{ct_\gamma (\boldsymbol{x}_i - \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})}{|\boldsymbol{x}_i - \boldsymbol{y}|^3} H(ct_\gamma - |\boldsymbol{x}_i - \boldsymbol{y}|) \, \mathrm{d}S_y. \tag{5b}$$

Here, H denotes the Heaviside function, and $U_{ij}^{(\gamma)} := 0$ and $W_{ij}^{(\gamma)} := 0$ if $\gamma \leqslant 0$. Further, x_j^α (respectively, y_j^α) represents the unknown (respectively, given) u_j^α or q_j^α on E_j at the current time-step α. The coefficients A_{ij} and B_{ij} coincide with either $\pm U_{ij}^{(1)}$ or $\pm W_{ij}^{(1)}$ according to the boundary condition on E_j. Because of the truncated power functions, the spatial integrals in (5) can be calculated analytically [26], whereas the fast TDBIEM presented below considers the spatial integrals numerically.

One can solve (4) for the unknown x_j^α when the RHS is computed with u_j^β, q_j^β, and y_j^α, i.e. the data of all the elapsed time-steps β and the current time-step α.

Note that the influence coefficients $U_{ij}^{(\gamma)}$ and $W_{ij}^{(\gamma)}$ represent the source information emitted t_γ ago from E_j and subsequently observed at $\boldsymbol{x}_i$. Here, the index γ represents the time-difference and not the time-step. Consequently, matrices $\mathsf{U}^{(\gamma)}$

and $\mathsf{W}^{(\gamma)}$, which consist of elements $U_{ij}^{(\gamma)}$ and $W_{ij}^{(\gamma)}$, respectively, derive a narrow (respectively, wide) band when γ is small (respectively, large).

Finally, a useful property of the influence coefficients is worth mentioning. Although $U_{ij}^{(\gamma)}$ and $W_{ij}^{(\gamma)}$ can have non-zero values for any γ, one can prove that the triplets $U_{ij}^{(\gamma+2)} - 2U_{ij}^{(\gamma+1)} + U_{ij}^{(\gamma)}$ and $W_{ij}^{(\gamma+2)} - 2W_{ij}^{(\gamma+1)} + W_{ij}^{(\gamma)}$ in the RHS of (4) exactly vanish if the time difference t_γ is sufficiently large to satisfy $ct_\gamma > |\boldsymbol{x}_i - \boldsymbol{y}|$ for any $\boldsymbol{y} \in E_j$ for a pair of i and j. This indicates that the influence coefficients $U_{ij}^{(\gamma)}$ and $W_{ij}^{(\gamma)}$ for a sufficiently large γ are not necessarily computed. The minimum value of γ for arbitrary pairs of i and j is given by a constant

$$\gamma_{\min} := \operatorname{ceil}\left(\frac{\max_{\boldsymbol{x},\boldsymbol{y}\in\partial D}|\boldsymbol{x}-\boldsymbol{y}|}{c\Delta_{\mathrm{t}}}\right), \tag{6}$$

where the numerator in the RHS represents the size of the boundary ∂D. Note that $\gamma_{\min}$ cannot exceed N_{t}, i.e. the upper bound of $\gamma_{\min}$ is N_{t}.

2.3. Issues of conventional TDBIEM

In the conventional TDBIEM or *direct method*, computing multiple matrix-vector products ($\mathsf{W}^{(\cdot)}\mathsf{u}^{\cdot}$ and $\mathsf{U}^{(\cdot)}\mathsf{q}^{\cdot}$) in the RHS of (4) is the most time-consuming process. The computational complexity scales as $\mathcal{O}(N_{\mathrm{s}}^2 N_{\mathrm{t}})$, where N_{t} denotes the total number of time-steps; the exponent of N_{t} is not two because the number of the elapsed time-steps β to be considered is bounded by the constant $\gamma_{\min}$ for every α. The quadratic complexity in space restricts the conventional TDBIEM from analysing large-scale problems. The FMM presented in Sections 3 and 4 aims at reducing the computational cost.

On the other hand, in general the cost required to solve (4) for the current unknown vector x^α does not matter in general because the matrix A is associated with the narrow band matrices $\mathsf{U}^{(1)}$ and $\mathsf{W}^{(1)}$. Hence, achieving the solution requires a computational cost of $\mathcal{O}(N_{\mathrm{s}})$ every time-step, resulting in $\mathcal{O}(N_{\mathrm{s}} N_{\mathrm{t}})$ for all time-steps.

Another issue of the conventional TDBIEM is the significant memory consumption. To avoid calculation of the same time-difference in different time-steps, it is usual to store the non-vanishing coefficients $W_{ij}^{(\gamma)}$ and $U_{ij}^{(\gamma)}$ for every time-difference γ (only if $\gamma < \gamma_{\min}$). Therefore, the memory requirement is $\mathcal{O}(N_{\mathrm{s}}^2 \gamma_{\min})$, although the exponent of N_{s} is in fact less than two because $\mathsf{U}^{(\gamma)}$ and $\mathsf{W}^{(\gamma)}$ are sparse when γ is small. If it is impossible to store all values, one may store a part of the coefficients and recalculate the unstored coefficients; however, such recalculations increase the computational time. It should be noted that regardless of which non-zero coefficients are stored, the computational complexity of the conventional TDBIEM is $\mathcal{O}(N_{\mathrm{s}}^2 N_{\mathrm{t}})$ because matrix-vector products, each of which incurs a cost of $\mathcal{O}(N_{\mathrm{s}}^2)$, are performed (at most $\gamma_{\min}$ times) for every time-step in the RHS of (4).

To resolve the memory consumption issue, it is worth mentioning that the memory-saving algorithm [25] can reduce the upper bound of $\gamma_{\min}$ in (6) from N_{t} to $N_{\mathrm{t}}/2$ without any computational penalty. Although this study exploits this efficient algorithm, memory shortage (thus, the recalculation of some influence coefficients) can still arise in large-scale problems. For the sake of readability, the details of this auxiliary algorithm are not described in this paper.

3. Formulation of time-domain interpolation-based FMM

Let us develop a fast method to evaluate the RHS of (4). To this end, this study proposes an interpolation-based FMM, which is based on approximating the integral kernel(s) of interest through interpolation. This section derives the formulae to construct the FMM discussed in Section 4.

3.1. Interaction between two clusters in space–time

For simplicity, (4) is equivalently rewritten with respect to a single time-difference $\alpha - \beta + 1$ as follows:

$$\sum_{j=1}^{N_{\mathrm{s}}} A_{ij} x_j^\alpha = \sum_{\beta=1}^{\alpha}\sum_{j=1}^{N_{\mathrm{s}}}\left(U_{ij}^{(\alpha-\beta+1)}\tau_j^\beta - W_{ij}^{(\alpha-\beta+1)}\sigma_j^\beta\right), \tag{7}$$

where, to simplify the notation, σ_j^β and τ_j^β are defined as follows:

$$\sigma_j^\beta := u_j^\beta - 2u_j^{\beta-1} + u_j^{\beta-2}, \qquad \tau_j^\beta := q_j^\beta - 2q_j^{\beta-1} + q_j^{\beta-2}.$$

Here, $u_j^\beta = q_j^\beta = 0$ if $\beta \leqslant 0$ because of the null initial condition. Further, in the definitions of $\sigma_j^{\cdot}$ and $\tau_j^{\cdot}$, the unknown density at the current time-step α, i.e. either u_j^α or q_j^α for each j, is set to zero because it is exactly x_j^α in the left-hand side (LHS) of (7). Note that σ_j^β and τ_j^β are already known at the current time-step α.

Now, let us focus on one part of the RHS of (7), i.e. the interaction between two clusters that are separated from each other in space–time (see Fig. 2). Specifically, let O and S be separated cubes (or *cells*) with the same edge length $2h_{\mathrm{s}}(=: d_{\mathrm{s}})$.

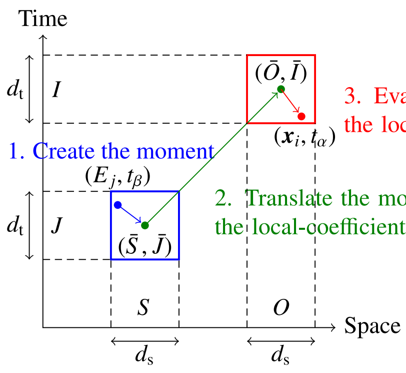


Fig. 2. Schematic illustration to evaluate the layer potential in (8), which is associated with the observation cluster $O \times I$ and the source cluster $S \times J$, through the three FMM-steps formulated in Section 3.3. The space is represented in one dimension for explanation.

The centres of O and S are denoted by $\bar{O}$ and $\bar{S}$, respectively. Further, let I and J be non-overlapping time-intervals with the same length $2h_t(=: d_t)$ and centres $\bar{I}$ and $\bar{J}$, respectively. Then, the RHS of (7) regarding the influence from the source cluster $S \times J$ to the observation (target) cluster $O \times I$, where $\bar{I} > \bar{J}$ is assumed without the loss of generality because of the causality, is represented as follows:

$$\frac{1}{4\pi c\Delta_t}\sum_{t_\beta\in J}\sum_{E_j\subset S}\left(\tau_j^\beta\int_{E_j}U(\boldsymbol{x}_i,\boldsymbol{y},t_\alpha,t_\beta-\Delta_t)\,\mathrm{d}S_y-\sigma_j^\beta\int_{E_j}\boldsymbol{W}(\boldsymbol{x}_i,\boldsymbol{y},t_\alpha,t_\beta-\Delta_t)\cdot\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}S_y\right)\quad\text{for } \boldsymbol{x}_i\in O,\ t_\alpha\in I, \tag{8}$$

where U and $\boldsymbol{W}$ are referred to as the single- and double-layer kernels, respectively, and are defined from (5) as follows:

$$U(\boldsymbol{x},\boldsymbol{y},t,s) := \frac{(c(t-s)-|\boldsymbol{x}-\boldsymbol{y}|)_+}{|\boldsymbol{x}-\boldsymbol{y}|}, \tag{9a}$$

$$\boldsymbol{W}(\boldsymbol{x},\boldsymbol{y},t,s) := \nabla_y U(\boldsymbol{x},\boldsymbol{y},t,s) = \frac{c(t-s)(\boldsymbol{x}-\boldsymbol{y})}{|\boldsymbol{x}-\boldsymbol{y}|^3}H\bigl(c(t-s)-|\boldsymbol{x}-\boldsymbol{y}|\bigr). \tag{9b}$$

In the following sections, several formulae of the proposed interpolation-based FMM used to compute (8) are described.

3.2. Approximation of U and $\boldsymbol{W}$ with interpolation

According to the methodology adopted in previous studies on the interpolation-based FMM [19–23] (where the time-domain case is investigated only in [23]), the kernels U and $\boldsymbol{W}$ in (9) are approximated using certain interpolation functions. The approximated kernels are expressed in a degenerate form [7], i.e. in the form of the separation of variables, which is necessary to develop an FMM-like algorithm. Unlike the previous studies (especially, [23]), the kernels under investigation lack smoothness on the *wavefront*, i.e. the sphere $|\boldsymbol{x}-\boldsymbol{y}| = c(t-s)$ with centre $\boldsymbol{y}$ and radius $c(t-s)$. More precisely, U is nondifferentiable and $\boldsymbol{W}$ $(=\nabla_y U)$ is discontinuous on the wavefront. These properties can lead to undesirable oscillations when the kernels are interpolated, as will be observed in Section 5.1. Nonetheless, the numerical examples in Sections 5.2 and 5.3 will show that the proposed interpolation-based FMM can work well for the TDBIEM for the wave equation.

Note that, because U and $\boldsymbol{W}$ in (9) are obtained from the temporal convolution of the fundamental solution with a given temporal basis (recall (5)), the smoothness of U and $\boldsymbol{W}$ originates in that of the piecewise-linear basis selected in this study. Hence, if a smoother basis is used, the accuracy of interpolation can improve. However, thus far, the present author has not discovered any appropriate base that is not only sufficiently smooth but also computationally cost-effective.

Let us consider an interpolation scheme such that the given function $f(x)$ defined in $[-1,1]$ is approximated by

$$\widetilde{f}(x) = \sum_{i<p} f\left(\omega_i^p\right)\ell_i(x), \tag{10}$$

where $\sum_{i<p}$ abbreviates $\sum_{i=0}^{p-1}$, $\ell_i(x)$ represents the prescribed interpolants, and $\omega_i^p \in [-1,1]$ denotes the prescribed nodes where f is sampled.

Applying any interpolation in the form of (10) to the eight variables of U (viz. x_1, x_2, x_3, y_1, y_2, y_3, t, and s) individually, one can approximate U as

$$\widetilde{U}(\boldsymbol{x},\boldsymbol{y},t,s) = \sum_{a<p_s}\sum_{b<p_s}\sum_{m<p_t}\sum_{n<p_t} U_{a,b,m,n}(O,S,I,J)\ell_a\left(\frac{\boldsymbol{x}-\bar{O}}{h_s}\right)\ell_b\left(\frac{\boldsymbol{y}-\bar{S}}{h_s}\right)\ell_m\left(\frac{t-\bar{I}}{h_t}\right)\ell_n\left(\frac{s-\bar{J}}{h_t}\right), \tag{11}$$

where p_s and p_t are the prescribed number of nodes for spatial and temporal interpolations, respectively. Also, the values of U at the $p_s^6 p_t^2$ nodes are denoted by

$$U_{a,b,m,n}(O,S,I,J) := U\big(\bar{O}+h_s\boldsymbol{\omega}_a^{p_s},\ \bar{S}+h_s\boldsymbol{\omega}_b^{p_s},\ \bar{I}+h_t\omega_m^{p_t},\ \bar{J}+h_t\omega_n^{p_t}\big)$$

with the nodal vector $\boldsymbol{\omega}_v^{p_s} := (\omega_{v_1}^{p_s}, \omega_{v_2}^{p_s}, \omega_{v_3}^{p_s}) \in [-1,1]^3$, and the following abbreviation is used:

$$\sum_{v<p_s} \ell_v(\boldsymbol{z}) := \sum_{v_1<p_s}\sum_{v_2<p_s}\sum_{v_3<p_s} \ell_{v_1}(z_1)\ell_{v_2}(z_2)\ell_{v_3}(z_3)$$

for $v=a,b$ and $\boldsymbol{z} := (z_1,z_2,z_3) \in [-1,1]^3$. The RHS of (11) is in the form of the separation of variables. Indeed, the four independent variables $\boldsymbol{x}\in O$, $\boldsymbol{y}\in S$, $t\in I$, and $s\in J$ are separated by four individual interpolants. Note that one can adopt different interpolation schemes (interpolants or nodes) for different variables in general, although this study uses the same scheme (see Section 5.1).

On the other hand, the double-layer kernel $\boldsymbol{W}$ is not interpolated directly. Instead, because $\boldsymbol{W} = \nabla_y U$ holds, the approximation of $\boldsymbol{W}$, which is denoted by $\widetilde{\boldsymbol{W}}$, is obtained from $\nabla_y \widetilde{U}$, where ∇_y operates on ℓ_b in (11). Note that the discontinuous double-layer kernel is consequently smoothed across the wavefront according to the smoothness of ℓ_b.

3.3. FMM-like expression for the layer potential in (8)

The interpolated kernels $\widetilde{U}$ and $\widetilde{\boldsymbol{W}}$ in the previous section are substituted in U and $\boldsymbol{W}$ in the layer potential of interest in (8) to yield the following FMM-like expression:

$$\frac{1}{4\pi c\Delta_t}\sum_{t_\beta\in J}\sum_{E_j\subset S}\left(\tau_j^\beta\int_{E_j} U(\boldsymbol{x}_i,\boldsymbol{y},t_\alpha,t_\beta-\Delta_t)\,\mathrm{d}S_y - \sigma_j^\beta\int_{E_j}\boldsymbol{W}(\boldsymbol{x}_i,\boldsymbol{y},t_\alpha,t_\beta-\Delta_t)\cdot\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}S_y\right)$$
$$\approx \sum_{a<p_s}\sum_{m<p_t}\ell_a\left(\frac{\boldsymbol{x}_i-\bar{O}}{h_s}\right)\ell_m\left(\frac{t_\alpha-\bar{I}}{h_t}\right)L_{a,m}(O,I) \quad \text{for } \boldsymbol{x}_i\in O,\ t_\alpha\in I. \tag{12}$$

This is the so-called *local expansion* of the layer potential and $L_{a,m}$ is the *local-coefficient* computed by the *multipole-to-local (M2L) formula*

$$L_{a,m}(O,I) := \sum_{b<p_s}\sum_{n<p_t} U_{a,b,m,n}(O,S,I,J)M_{b,n}(S,J), \tag{13}$$

where $M_{b,n}$ represents the so-called *multipole moment*:

$$M_{b,n}(S,J) := \frac{1}{4\pi c\Delta_t}\sum_{t_\beta\in J}\sum_{E_j\subset S}\ell_n\left(\frac{t_\beta-\Delta_t-\bar{J}}{h_t}\right)\int_{E_j}\left(\tau_j^\beta\ell_b\left(\frac{\boldsymbol{y}-\bar{S}}{h_s}\right) - \sigma_j^\beta\nabla_y\ell_b\left(\frac{\boldsymbol{y}-\bar{S}}{h_s}\right)\cdot\boldsymbol{n}(\boldsymbol{y})\right)\mathrm{d}S_y. \tag{14}$$

Eqs. (12), (13), and (14) are the fundamental formulae that will be used to develop the proposed FMM-like algorithm.

Finally, it is worth noting that the M2L translation in (13) can be accelerated by the fast Fourier transform (FFT). To this end, this study assumes that the nodes $\omega_i^{p_\cdot}$ are equidistant. Then, because the kernel U in (9a) is a translation-invariant function [20] with respect to space and time, the M2L translator $U_{a,b,m,n}$ in (13) can be re-expressed as $U_{a-b,m-n}$, where $a-b\in(-p_s,p_s)^3$ and $m-n\in(-p_t,p_t)$ are the two new indices replacing the four original indices, and the M2L translation is therefore rewritten as $L_{a,m} = \sum_b\sum_n U_{a-b,m-n}M_{b,n}$. This can be identified as a 4D discrete Fourier convolution, and can be computed efficiently using the 4D FFT. The computational cost can be reduced from $\mathcal{O}(p_s^6p_t^2)$ to $\mathcal{O}(p_s^3p_t(\log p_s+\log p_t))$. This study utilises the FFTW library [27,28], specifically, the routines `fftw_execute_dft_{r2c,c2r}` from the class of 'multi-dimensional DFTs of real data'. This technique is called the FFT-based diagonalisation method and it was originally proposed for the interpolation-based FMM in the frequency domain [22].

3.4. M2M and L2L formulae

To construct a multi-level FMM, one needs the so-called multipole-to-multipole (M2M) and local-to-local (L2L) formulae that translate the multipole moment and local coefficient, respectively, across two levels in the space–time hierarchy, which will be defined in Section 4.1. These formulae can be derived with the help of the interpolation of interpolants [23], i.e.

$$\ell_i\left(\frac{t-s}{2}\right) \approx \sum_{j<p}\ell_i\left(\frac{\omega_j^p-s}{2}\right)\ell_j(t) = \sum_{j<p}D_{i,j}(s)\ell_j(t), \tag{15}$$

where $s,t\in[-1,1]$ and $D_{i,j}(s) := \ell_i(\frac{\omega_j^p-s}{2})$.

Now, let S' be one of the eight sub-cells (children) of S (see Fig. 3). Denote the edge length of S' by $2h_s'$, where $h_s' = h_s/2$ is assumed. Similarly, let J' be one of the two sub-intervals of J. Denote the length of J' by $2h_t$, where $h_t' = h_t/2$. Then, one can derive the following M2M formula by applying (15) to (14):

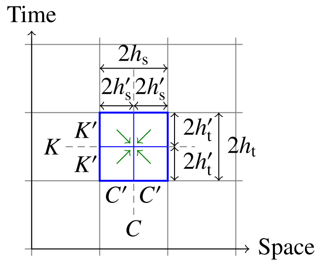
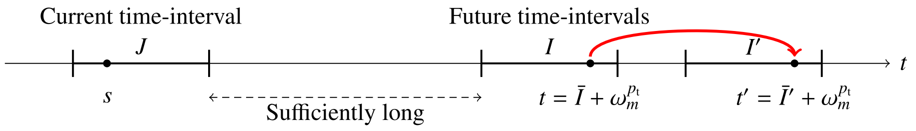


Fig. 3. Schematic illustration for the M2M and L2L translations. Let $C = S$ and $K = J$ in the M2M translation in Eq. (16) and $C = O$ and $K = I$ in the L2L translation in Eq. (17). The space is represented in one dimension for explanation.



Current time-interval
Future time-intervals
J
I
I'
t
s
Sufficiently long
$t = \bar{I} + \omega_m^{p_\mathrm{t}}$
$t' = \bar{I}' + \omega_m^{p_\mathrm{t}}$

Fig. 4. Schematic illustration of computing $L_{a,m}(O, I')$ (i.e. the mth component of the local-coefficient for a time interval I') from $L_{a,m}(O, I)$ (i.e. the same component of the local-coefficient for a time-interval I) according to Eq. (18a).

$$M_{b,n}(S, J) \approx \sum_{b'<p_\mathrm{s}} \sum_{n'<p_\mathrm{t}} D_{b,b'}\left(\frac{\bar{S} - \bar{S}'}{h'_\mathrm{s}}\right) D_{n,n'}\left(\frac{\bar{J} - \bar{J}'}{h'_\mathrm{t}}\right) M_{b',n'}\left(S', J'\right), \tag{16}$$

where the identities $\frac{\boldsymbol{y}-\bar{S}}{h_\mathrm{s}} = \frac{1}{2}\left(\frac{\boldsymbol{y}-\bar{S}'}{h'_\mathrm{s}} - \frac{\bar{S}-\bar{S}'}{h'_\mathrm{s}}\right)$ and $\frac{t_{\beta-1}-\bar{J}}{h_\mathrm{t}} = \frac{1}{2}\left(\frac{t_{\beta-1}-\bar{J}'}{h'_\mathrm{t}} - \frac{\bar{J}-\bar{J}'}{h'_\mathrm{t}}\right)$ are considered. Further, $\sum_{b'<p_\mathrm{s}} D_{b,b'}(\boldsymbol{z})$ represents

$$\sum_{b_1<p_\mathrm{s}} \sum_{b_2<p_\mathrm{s}} \sum_{b_3<p_\mathrm{s}} D_{b_1,b'_1}(z_1) D_{b_2,b'_2}(z_2) D_{b_3,b'_3}(z_3)$$

for $\boldsymbol{z} := (z_1, z_2, z_3) \in [-1, 1]^3$.

Similarly, let O' be a sub-cell of O and I' be a sub-interval of I (see Fig. 3). Then, one can obtain the following L2L formula:

$$L_{a',m'}\left(O', I'\right) \approx \sum_{a<p_\mathrm{s}} \sum_{m<p_\mathrm{t}} D_{a,a'}\left(\frac{\bar{O} - \bar{O}'}{h'_\mathrm{s}}\right) D_{m,m'}\left(\frac{\bar{I} - \bar{I}'}{h'_\mathrm{t}}\right) L_{a,m}(O, I). \tag{17}$$

Note that the prescribed parameters p_s and p_t are assumed to be constant through all the levels in the hierarchy, which implies that the target of the proposed method is the low-frequency regime. Because the wavelength (respectively, period) becomes relatively small as the spatial (respectively, temporal) scale becomes large, the approximation parameters p_s and p_t should be sufficiently large to be able to approximate the wave field at the largest scale, i.e. at level 2.

3.5. Translation of the local-coefficient with respect to time

In principle, the source information for a certain time-interval must be transmitted to all time-intervals in the future. The M2L formula in (13) can be used for this purpose; however its naïve application leads to a computational cost of $\mathcal{O}(N_\mathrm{t}^2)$, which is entirely unacceptable.

This issue can be avoided by computing the local-coefficient in a different manner from that in (13). Suppose that I is far from J such that $c(t - s) > |\boldsymbol{x} - \boldsymbol{y}|$ holds for any $\boldsymbol{x} \in O$, $t \in I$, $\boldsymbol{y} \in S$, and $s \in J$ (see Fig. 4). Then, the kernel U is a linear function with respect to t as follows:

$$U(\boldsymbol{x}, \boldsymbol{y}, t, s) = \frac{c(t - s) - |\boldsymbol{x} - \boldsymbol{y}|}{|\boldsymbol{x} - \boldsymbol{y}|}.$$

(Note that this property was used to derive (6).) Let I' be a time-interval such that $\bar{I}' > \bar{I}$ (thus, I' is also sufficiently separated from J). Then, one can obtain

$$U\left(\boldsymbol{x}, \boldsymbol{y}, t', s\right) = U(\boldsymbol{x}, \boldsymbol{y}, t, s) + \frac{c}{|\boldsymbol{x} - \boldsymbol{y}|}\left(t' - t\right) \quad \text{for } t \in I,\ t' \in I'.$$

Substituting this equation in (13) where I is replaced with I', one can obtain the following formula to compute the local-coefficient for I' from that for I:

Code 1 Main

```
Input parameters; basically, $c$, $\partial D$ (boundary model and conditions), $N_s$, $\Delta_t$, $N_t$, $p_s$, $p_t$ and $Z$.
Build the space–time hierarchy.
Initialise all the RHS vectors: let $r^\alpha = 0$ for $\alpha \in [1, N_t)$.
for time-step $\alpha = 1$ to $N_t - 1$ do
  Calculate the near-field component of the RHS vector $r^\alpha$ for the current time-step $\alpha$ according to Code 2.
  Solve Eq. (7) or $Ax^\alpha = r^\alpha$ to obtain the unknown vector $x^\alpha$.
  Output the solution $x^\alpha$.
  Calculate the far-field component of the RHS vectors $r^\beta$ for the future time-steps (i.e. $\beta > \alpha$) according to Codes 3 and 4.
end for
```

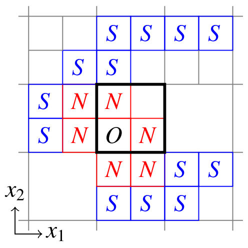


Fig. 5. Example of cells S in the interaction-list $\mathscr{I}(O)$ and cells N in the neighbour-list $\mathscr{N}(O)$. The cell with a thick border is O's parent. The white cells at the same level as O represent empty cells. Meanwhile, a white cell at the same level as O's parent is either an empty cell or a leaf and such a leaf is neither in $\mathscr{I}(O)$ nor $\mathscr{N}(O)$, but in the neighbour-list of O's parent. This illustration is in two dimensions for explanation.

$$L_{a,m}(O, I') = L_{a,m}(O, I) + \dot{L}_a(O, I)(\bar{I}' - \bar{I}), \tag{18a}$$

where

$$\dot{L}_a(O, I) := \sum_{b<p_s} \sum_{n<p_t} \frac{c}{|(\bar{O} + h_s \boldsymbol{\omega}_a^{p_s}) - (\bar{S} + h_s \boldsymbol{\omega}_b^{p_s})|} M_{b,n}(S, J) \tag{18b}$$

represents the first-order time derivative of the local-coefficient for I. As described in Section 4.3.2, one can perform the M2L operation with the computational cost of $\mathcal{O}(N_t)$ by using another M2L formula in (18), which is meant for the condition $\bar{I} \gg \bar{J}$, together with the usual M2L formula in (13), which is meant for the condition $\bar{I} \sim \bar{J}$.

Note that in the RHS of (18b), the summation over n can be initially computed. If ω^{p_s} are equidistant nodes, the resulting RHS is of the form of discrete convolution with respect to b, and thus can be computed efficiently by the 3D FFT.

4. Algorithm

Using the formulae derived in the previous section, one can construct a fast algorithm to solve the BIE in (2) with the computational complexity of $\mathcal{O}(N_s^{1+\delta} N_t)$, where δ is 1/3 or 1/2 according to the assumption of the spatial distribution of N_s boundary elements. The details of the algorithm are described below along with four pseudo codes.

4.1. Main routine

Code 1 presents the main routine of the proposed TDBIEM. After the setup stage that includes the creation of the space–time hierarchy, the unknown vector x^α is solved from (7) or its matrix form $Ax^\alpha = r^\alpha$ for every time-step α $(= 1, \ldots, N_t - 1)$. Prior to the solution of (7), the RHS vector r^α is computed using near- and far-field calculations. Here, the evaluation of the RHS vectors using the multipole and local expansions is termed the *far-field calculation*. This calculation is based on the formulae given in Section 3 and the details are described in Section 4.3. On the other hand, the remaining calculation is termed the *near-field calculation* and is performed in the conventional MOT manner, as described in Section 4.2.

In the remaining part of this section, the space–time hierarchy and relevant concepts such as interaction-list, neighbour-list, time-intervals, and time-gating are described.

Similar to ordinary FMMs, the proposed FMM utilises an adaptive (or non-uniform) octree for space to create the hierarchy of boundary elements. According to the conventional manner, an octree is constructed by assigning the maximum number of boundary elements per leaf, denoted by Z. The maximum (finest) level of the constructed octree is denoted by $l_{\max}$. Further, the length of edges of the cells (cubes) at level l $(= 0, \ldots, l_{\max})$ is denoted by $2h_s^{(l)} (=: d_s^{(l)})$ as per the notations used in Section 3.

With the help of the hierarchy of cells, the *interaction-list* of a certain cell O, which is denoted by $\mathscr{I}(O)$, is defined as the set of non-empty cells S at the same level as O. In particular, such a cell S in $\mathscr{I}(O)$ must satisfy that S is not adjacent to O and that S's parent is adjacent to O's parent. An interaction-list contains a maximum of 189 cells. Meanwhile, the set of non-empty cells that are at the same level as O and adjacent to O is termed the *neighbour-list* of O and is denoted by $\mathscr{N}(O)$. There are at most 27 neighbour cells in $\mathscr{N}(O)$, which includes O itself. Fig. 5 illustrates an example.

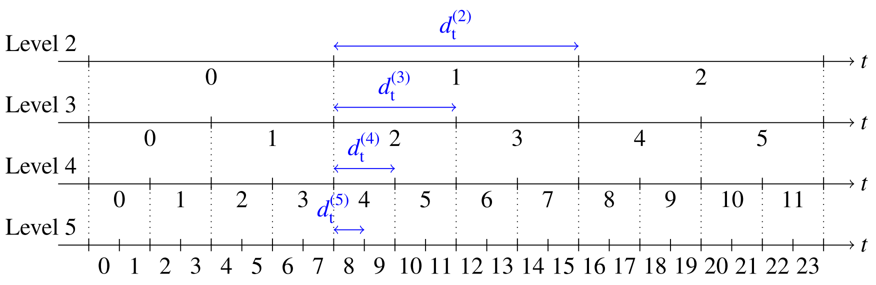


Fig. 6. Example of the time hierarchy. The numbers represent the indices of time-intervals for each level.

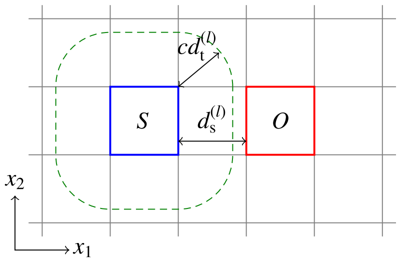


Fig. 7. The region surrounded by the dashed line represents the region where the source influence (information) that radiates from the source cell S, which is one of the closest cells to O in the interaction-list $\mathscr{I}(O)$, can reach within a given time $d_t^{(l)}$ (the length of the given time-interval). Because $d_s^{(l)} \geqslant cd_t^{(l)}$ is guaranteed by Eqs. (19) and (20), the influence from any $S \subset \mathscr{I}(O)$ never reaches the inside of O within the time $d_t^{(l)}$. This figure illustrates a two-dimensional case for explanation.

The hierarchy of time-intervals is created from the abovementioned spatial hierarchy. The length of the time-intervals associated with (spatial) level l is denoted by $2h_t^{(l)} (=: d_t^{(l)})$. Then, the number of time-steps per $d_t^{(l)}$, which is denoted by $M_t^{(l)}$, is recursively determined as follows:

$$M_t^{(l_{\max})} := \operatorname{floor}\left(\frac{d_s^{(l_{\max})}}{c\Delta_t}\right) \quad \text{and} \quad M_t^{(l)} := 2M_t^{(l+1)} \quad \text{for } 2 \leqslant l < l_{\max}, \tag{19}$$

where the levels 0 and 1 are ignored because the proposed algorithm does not use any cells at these levels, as in ordinary FMMs. From (19), $d_t^{(l)}$ is determined as

$$d_t^{(l)} := M_t^{(l)} \Delta_t \quad \text{for } 2 \leqslant l \leqslant l_{\max}. \tag{20}$$

From (20), the ith time-interval at level l is defined by

$$I_i^{(l)} := \left(id_t^{(l)}, (i+1)d_t^{(l)}\right] \quad \text{for } i = 0, 1, 2, \ldots.$$

Fig. 6 illustrates the hierarchy (basically, a binary tree) of time-intervals. Obviously, each time-interval consists of two sub-intervals, except for the lowest level $l_{\max}$; in general, the sub-intervals of $I_i^{(l)}$ are given by $I_{2i}^{(l+1)}$ and $I_{2i+1}^{(l+1)}$.

Note that the relationship $d_s^{(l)} \geqslant cd_t^{(l)}$, which is obtained from (19) and (20), guarantees that for every observation cell O at level l, influence (information) from any source cells $S \subset \mathscr{I}(O)$ takes time $d_t^{(l)}$ or more before arriving at O (see Fig. 7). Therefore, one can delay the instant to perform the far-field calculation for the underlying time-interval $I_i^{(l)}$ until the end of the time-interval. At the end, it is allowed to create the multipole moment for $I_i^{(l)}$ and translate it to the local-coefficient for future time-intervals, i.e. $I_{i+1}^{(l)}, I_{i+2}^{(l)}, \ldots$. This method is borrowed from the PWTD algorithm [6] and is referred to as *time-gating*.

4.2. Near-field calculation

The near-field calculation is performed at every time-step (see Code 2). Let α be the current time-step and O be a cell at level l. Then, if O or $S \subset \mathscr{N}(O)$ is a leaf, one directly computes the relevant part of r^{α}, i.e.

$$\sum_{\beta=\alpha-\gamma_{\min}^{(l)}}^{\alpha} \sum_{E_j \subset S} \left(U_{ij}^{(\alpha-\beta+1)} \tau_j^{\beta} - W_{ij}^{(\alpha-\beta+1)} \sigma_j^{\beta}\right) \quad \text{for every } \boldsymbol{x}_i \in O, \tag{21}$$

Code 2 Near-field calculation

```
The current time-step is $\alpha$.
for level $l = 2$ to $l_{\max}$ do
  for cell $O$ in level $l$ do
    for cell $S$ in the neighbour-list $\mathcal{N}(O)$ do
      if $O$ or $S$ is a leaf then
        for collocation point $\boldsymbol{x}_i$ in $O$ do
          for boundary element $E_j$ in $S$ do
            for time-step $\beta = \alpha - \gamma_{\min}^{(l)}$ to $\alpha$ do
              Calculate (21), i.e. the contribution from $(E_j, t_\beta)$ to $(\boldsymbol{x}_i, t_\alpha)$. The result is accumulated to the $i$th component of the current RHS vector $\mathsf{r}^\alpha$, which already contains the contribution for the far-field calculation.
            end for
          end for
        end for
      end if
    end for
  end for
end for
```

Code 3 Upward pass in the far-field calculation

```
The current time-step is $\alpha$.
for level $l = l_{\max}$ to 2 do
  if $\alpha$ mod $M_{\mathrm{t}}^{(l)}$ is 0 then
    Denote the index of the current time-interval $\frac{\alpha}{M_{\mathrm{t}}^{(l)}} - 1$ by $i$ ($\geqslant 0$).
    for cell $S$ in level $l$ do
      if $S$ is a leaf then
        Compute $M(S, I_i^{(l)})$ according to Eq. (14).
      else
        Compute $M(S, I_i^{(l)})$ from $M(S', I_{2i}^{(l+1)})$ and $M(S', I_{2i+1}^{(l+1)})$ for every child $S'$ according to Eq. (16).
      end if
    end for
  end if
end for
```

where not all but only the last $\gamma_{\min}^{(l)}$ time-steps may be considered because the influence from S passes over O after a specific period. Similar to $\gamma_{\min}$ in (6), the constant $\gamma_{\min}^{(l)}$ is given by

$$\gamma_{\min}^{(l)} := \mathrm{ceil}\left(\frac{2\sqrt{3}d_{\mathrm{s}}^{(l)}}{c\Delta_{\mathrm{t}}}\right), \tag{22}$$

where the numerator in the RHS represents the maximum distance between O and $S \subset \mathcal{N}(O)$ at level l.

Using the memory-saving algorithm [25] from the conventional TDBIEM (recall Section 2.3), the proposed TDBIEM stores non-zero coefficients $U_{ij}^{(\cdot)}$ and $W_{ij}^{(\cdot)}$ in (21) as much as possible to avoid calculating the same coefficients in different time-steps. The proposed TDBIEM requires less memory than the conventional TDBIEM because $\gamma_{\min}^{(l)} < \gamma_{\min}$ holds. However, even if non-zero coefficients are stored, the computational complexity given in Section 4.4 does not decrease because the complexity of computing the relevant matrix-vector products $\mathsf{U}^{(\cdot)}\tau^{\cdot}$ and $\mathsf{W}^{(\cdot)}\sigma^{\cdot}$ is essentially the same as that of filling the matrices.

4.3. Far-field calculation

The far-field calculation is performed through the upward pass and the subsequent downward pass.

4.3.1. Upward pass

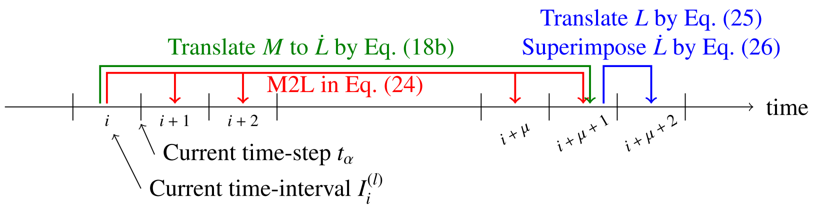
Starting from level $l_{\max}$ towards 2, one computes multipole moments (see Code 3). This computation is in fact done for levels l such that the current time t_α meets the end of the current time-interval, which is denoted here by $I_i^{(l)}$. (Recall the note in Section 4.1 for the instant at which the far-field calculation is performed.) For all the cells at such levels l, their moments for $I_i^{(l)}$ are computed.

In particular, if S is a leaf at level l, one computes S's moment for the current time-interval $I_i^{(l)}$ according to (14), where the spatial integral is numerically evaluated using the triangular Gaussian quadrature [29]; the three-points formula is used in the actual computation in Section 5.

Alternatively, if S is not a leaf, one computes the moment of S for $I_i^{(l)}$ by collecting the moments of S's children for $I_i^{(l)}$'s sub-intervals (i.e. $I_{2i}^{(l+1)}$ and $I_{2i+1}^{(l+1)}$) according to the M2M formula in (16).

Code 4 Downward pass in the far-field calculation

```
The current time-step is $\alpha$.
for level $l=2$ to $l_{\max}$ do
  if $\alpha$ mod $M_t^{(l)}$ is 0 then
    Denote the index of the current time-interval $\frac{\alpha}{M_t^{(l)}}-1$ by $i$ ($\geqslant 0$).
    for cell $O$ in level $l$ do
      for cell $S$ in $\mathscr{I}(O)$ do
        Compute $L(O, I_{i+1}^{(l)}), \ldots, L(O, I_{i+\mu+1}^{(l)})$ from $M(S, I_i^{(l)})$ according to Eq. (24).
        Compute $\dot{L}(O, I_{i+\mu+1}^{(l)})$ from $M(S, I_i^{(l)})$ according to Eq. (18b).
      end for
      Compute $L(O, I_{i+\mu+2}^{(l)})$ according to Eq. (25).
      Compute $\dot{L}(O, I_{i+\mu+2}^{(l)})$ according to Eq. (26).
      if $O$ is a leaf then
        Evaluate the filed for any $\boldsymbol{x}_i \in O$ and $t_\beta \in I_{i+1}^{(l)}$ with $L(O, I_{i+1}^{(l)})$ according to Eq. (12) (where $I$ and $t_\alpha$ read $I_{i+1}^{(l)}$ and $t_\beta$, respectively).
      else
        Compute $L(O', I_{2(i+1)}^{(l+1)})$ and $L(O', I_{2(i+1)+1}^{(l+1)})$ from $L(O, I_{i+1}^{(l)})$ for every child $O'$ according to Eq. (17).
      end if
    end for
  end if
end for
```



Translate L by Eq. (25)
Superimpose $\dot{L}$ by Eq. (26)
Translate M to $\dot{L}$ by Eq. (18b)
M2L in Eq. (24)
time
i
$i+1$
$i+2$
$i+\mu$
$i+\mu+1$
$i+\mu+2$
Current time-step t_α
Current time-interval $I_i^{(l)}$

Fig. 8. Translation from the multipole moment for the current time-interval $I_i^{(l)}$ to the local-coefficients for future time-intervals.

4.3.2. Downward pass

Beginning from level 2 towards $l_{\max}$, one computes the local-coefficients and evaluates the RHS vectors r^β for the future time-steps β ($>\alpha$) (see Code 4). The instant at which the downward pass is performed is the same as that for the upward pass.

First, for every cell O at each relevant level, the local-coefficient of O is computed from the moments of S in $\mathscr{I}(O)$ according to the M2L formula in (13). As noted in Section 4.1, S for the current time-interval $I_i^{(l)}$ cannot interact with O during the same time-interval. Hence, one translates S's moment for the current time-interval $I_i^{(l)}$ to O's local-coefficients for the future time-intervals, i.e. $I_{i+1}^{(l)}, I_{i+2}^{(l)}, \ldots$.

However, the M2L formula in (13) is not allowed to manage all future time-intervals from the viewpoint of the computational cost, as mentioned in Section 3.5. To avoid this issue, the naïve M2L formula in (13) is applied to a finite number of future time-intervals $I_{i+1}^{(l)}, \ldots, I_{i+\mu+1}^{(l)}$, which are relatively close to the current time-interval $I_i^{(l)}$. Meanwhile, another M2L formula in (18) is applied to the remaining future time-intervals $I_{i+\mu+2}^{(l)}, I_{i+\mu+3}^{(l)}, \ldots$, which are sufficiently away from the current time-interval $I_i^{(l)}$.

The above constant μ is determined so that the prerequisite of (18) is satisfied for this case. Namely, $c(t-s) > |\boldsymbol{x}-\boldsymbol{y}|$ must be satisfied for any $\boldsymbol{x} \in O$, $t \in I_{i+\mu+1}^{(l)}$, $\boldsymbol{y} \in S \subset \mathscr{I}(O)$, and $s \in I_i^{(l)}$. Because $\min(t-s) = \mu M_t^{(l)} \Delta_t$ and $\max|\boldsymbol{x}-\boldsymbol{y}| = 4\sqrt{3} d_s^{(l)}$ hold, one can determine μ such that it is minimised, i.e.

$$\mu = \operatorname{floor}\left(\frac{4\sqrt{3} d_s^{(l)}}{c M_t^{(l)} \Delta_t}\right) = 8, \tag{23}$$

where (19) is used. Note that μ is independent of levels.

Let us state the M2L operation in detail. On the one hand, the moment for the current time-interval $I_i^{(l)}$ is translated to the local-coefficients for $I_{i+1}^{(l)}, \ldots, I_{i+\mu+1}^{(l)}$ as follows (see Fig. 8):

$$L_{a,m}(O, I_j^{(l)}) \mathrel{+}= \sum_{S \subset \mathscr{I}(O)} \sum_{b<p_s} \sum_{n<p_t} U_{a,b,m,n}(O, S, I_j^{(l)}, I_i^{(l)}) M_{b,n}(S, I_i^{(l)}) \quad \text{for } j = i+1, \ldots, i+\mu+1, \tag{24}$$

where '+=' represents adding the LHS by the RHS. This incremental operation is necessary because the LHS may already have a value given by (i) the M2L translations for the elapsed time-intervals and/or (ii) the L2L translation from the upper level $l-1$ for the current time-interval (see below).

On the other hand, the local-coefficients for the other time-intervals, i.e. $I^{(l)}_{i+\mu+2}, I^{(l)}_{i+\mu+3}, \ldots$, are computed using the second M2L formula in (18). However, if one simply applies this formula to all these time-intervals, the computational cost becomes $\mathcal{O}(N_\mathrm{t}^2)$ after all computations. To achieve linear complexity, one may apply (18a) only to the first time-interval $I^{(l)}_{i+\mu+2}$, i.e. one may translate the local-coefficient for $I^{(l)}_{i+\mu+2}$ from that for $I^{(l)}_{i+\mu+1}$ as follows:

$$L_{a,m}\bigl(O, I^{(l)}_{i+\mu+2}\bigr) \mathrel{+}= L_{a,m}\bigl(O, I^{(l)}_{i+\mu+1}\bigr) + \dot{L}_a\bigl(O, I^{(l)}_{i+\mu+1}\bigr) d^{(l)}_\mathrm{t}, \tag{25}$$

where the rate $\dot{L}_a(O, I^{(l)}_{i+\mu+1})$ is computed by (18b) where $I = I^{(l)}_i$ and $J = I^{(l)}_{i+\mu+1}$. Once (25) is computed, one superimposes the rate for $I^{(l)}_{i+\mu+1}$ to that for the next time-interval $I^{(l)}_{i+\mu+2}$ as follows:

$$\dot{L}_a\bigl(O, I^{(l)}_{i+\mu+2}\bigr) \mathrel{+}= \dot{L}_a\bigl(O, I^{(l)}_{i+\mu+1}\bigr). \tag{26}$$

The procedure given by (25) and (26) is equivalent to the direct application of the M2L translation from the moment for $I^{(l)}_i$ to the local-coefficients for $I^{(l)}_{i+\mu+2}, I^{(l)}_{i+\mu+3}, \ldots$.

Note that the rate in the RHS of (26) is no longer used after (26) has been computed. Hence, the memory required for these rates is considerably small. Moreover, once a multipole moment for a certain time-interval is used in (24) and (18b), it can be eliminated unless it is used in the M2M translation in the next time-interval. As a result, one may provide two memory spaces to the multipole moment of each cell; one for the even time-intervals and the other for odd time-intervals.

Next, if O is not a leaf, O's local-coefficient for the next time-interval $I^{(l)}_{i+1}$ is translated to its children according to (17). Alternatively, if O is a leaf, one evaluates the field (the layer potential) for all the time-steps in $I^{(l)}_{i+1}$ with the local-coefficient for $I^{(l)}_{i+1}$ according to (12). Note that, once a local-coefficient for $I^{(l)}_{i+1}$ is used in either operation, it is no longer used, and thus may be immediately discarded. As a result, one may store local-coefficients only for $\mu + 2$ $(= 10)$ time-intervals in every cell.

4.4. Computational complexity

To calculate the computational complexity of the proposed algorithm (say, the multilevel time-domain interpolation-based FMM), it is assumed that N_s boundary elements are distributed either (i) uniformly or (ii) flatly in a fixed domain. In the case of (i), the resulting octree is uniform and thus there are 8^l cells at level l. Meanwhile, in the case of (ii), the octree is regarded as a uniform quad-tree and thus there are 4^l cells at level l. In addition, it is assumed that the number of boundary elements per leaf is constant, i.e. $Z \sim 1$, where Z is defined in Section 4.1. Then, because all the leaves exist in the finest level l_max, $N_\mathrm{s} = Z \cdot 8^{l_\mathrm{max}}$ and $N_\mathrm{s} = Z \cdot 4^{l_\mathrm{max}}$ hold in the case of (i) and (ii), respectively. Moreover, recall that $\gamma^{(l)}_\mathrm{min} \propto d^{(l)}_\mathrm{s} \sim 2^{-l}$, $d^{(l)}_\mathrm{t} \propto M^{(l)}_\mathrm{t} \sim 2^{-l}$, and $\mu, p_\mathrm{s}, p_\mathrm{t} \sim 1$. Thus, the computational complexity is calculated as $\mathcal{O}(N_\mathrm{s}^{4/3} N_\mathrm{t})$ and $\mathcal{O}(N_\mathrm{s}^{3/2} N_\mathrm{t})$ in the case of (i) and (ii), respectively (For details, see Appendix A). Therefore, the proposed TDBIEM is essentially faster than the conventional one that has the computational complexity of $\mathcal{O}(N_\mathrm{s}^2 N_\mathrm{t})$.

Note that the time-step size Δ_t does not explicitly appear in the above calculation, implying that Δ_t can be arbitrarily chosen. However, an excessive Δ_t can make $M^{(l_\mathrm{max})}_\mathrm{t}$ in (19) zero and thus the algorithm fails. Hence, Δ_t is assumed to be sufficiently small for any N_s and l under consideration.

4.5. Optimisation of codes

The fast and conventional TDBIEM codes used in this paper are parallelised by multiple threads running on a computer. Similar to ordinary FMMs, the fast TDBIEM has a parallelism with respect to cells at each level. Meanwhile, the conventional TDBIEM has a parallelism with respect to the row index i and column index j (in fact, only j is considered) when storing the influence coefficients $U^{(\gamma)}_{ij}$ and $W^{(\gamma)}_{ij}$ in (5) and another parallelism with respect to i when computing the matrix-vector products in (4). In the codes, the loops corresponding to these parallelisms are parallelised by inserting the OpenMP directive '`#omp parallel for`' for the case of the C/C++ language [30].

In spite of the FFT acceleration, the M2L translation in (13) or (24) is often time-consuming because of the large pre-factor $189(\mu + 1)$ and the overhead of FFT. To relieve this, an easy and effective optimisation is to pre-compute the translator $U_{a-b,m-n}$ (recall Section 3.3). Specifically, one may first compute $U_{a-b,m-n}(O, S, I, J)$ for all the 316 $(= 7^3 - 3^3)$ pairs of O and S and all the $\mu + 1 (= 9)$ pairs of I and J. Here, each translator is a real square matrix with dimension $(2p_\mathrm{s} - 1)^3(2p_\mathrm{t} - 1)$. Subsequently, one may apply the 4D-FFT to each translator and store the Fourier transformed data. Note that this pre-computation is independent of levels.

In addition, it is effective to fuse the three loops (summations) over spatial variables (e.g. b_1, b_2, and b_3 in the M2M operation in (16)) into a loop of length p_s^3 because SIMD units (i.e. Intel's AVX unit in this study [31]) can work efficiently for long loops. This is certainly effective to compute (12), (14), (16), and (17).

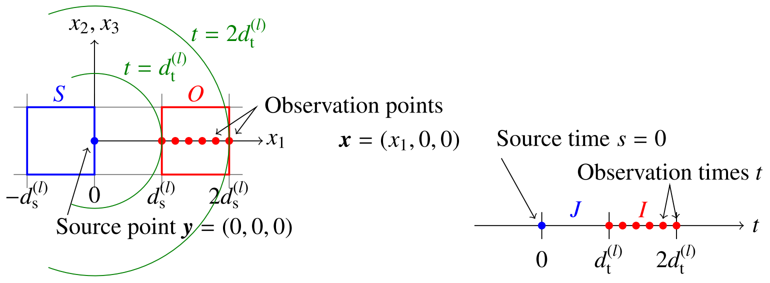


Fig. 9. Configuration to compute the error resulting from the cubic Hermite interpolation.

5. Numerical experiments

This section addresses the numerical aspects of the proposed interpolation-based TDBIEM. First, the interpolation of the kernels U and $\boldsymbol{W}$ in (5) is examined in Section 5.1. Next, in Section 5.2, the accuracy and performance of the proposed TD-BIEM are validated by performing analysis for an external scattering problem. Finally, in Section 5.3, a simulation regarding architectural acoustics is demonstrated to observe the feasibility of the proposed method to solve practical problems. All computations are performed in double precision.

5.1. Interpolation

This study employs the cubic Hermite interpolation (CHI; see Appendix B) for both spatial and temporal interpolations of U in (11), using equidistant nodes, i.e. $\omega_i^p := \frac{2i}{p-1} - 1$ for $0 \leqslant i < p$, which is necessary to use the FFT-based diagonalisation method. However, because the CHI requires the data of the first derivative of a function to be interpolated, it does not obey (10), which assumes that a given function f is interpolated only with its values. To resolve this problem, the first derivative is approximated by a finite difference as described in Appendix B.

In contrast to global interpolations (e.g. Lagrange interpolation), local interpolations that includes the CHI are relatively robust near nondifferentiable or discontinuous points, whereas global interpolations can exhibit severe oscillations (over/under shoots) in general. In addition, the computational cost of local interpolations is less than that of global ones; the cost of a local (respectively, global) interpolation is $\mathcal{O}(p)$ (respectively, $\mathcal{O}(p^2)$) to evaluate $\widetilde{f}(x)$ in (10) for a certain x.

Let us investigate the error resulting from CHI. Now, $U(\boldsymbol{x}, \boldsymbol{y}, t, s)$ and its gradient are examined with respect to $(\boldsymbol{x}, t)$ in the observation cluster $O \times I$ and $(\boldsymbol{y}, s)$ in the source cluster $S \times J$ for a certain level l (see Fig. 9). To make the two clusters interact strongly, let O and S be separated by the length $d_s^{(l)}$ (i.e. S is one of the nearest cells in the interaction-list of O) and let I be next to J, where $J := (0, d_t^{(l)}]$ and $I := (d_t^{(l)}, 2d_t^{(l)}]$ without loss of generality. Here, it is assumed that $d_s^{(l)}$ and $d_t^{(l)}$ satisfy $d_s^{(l)} = cd_t^{(l)}$, which is derived from (19) and (20) if one ignores a small variation owing to the floor operation in (19). For the source, let $\boldsymbol{y} := (0,0,0)$ and $s := 0$. With respect to the observer, let $\boldsymbol{x}$ be the 50 equally-spaced points on the x_1-axis in O and let t be the 50 equally-spaced points in I.

Because the wavefront goes through $O \times I$ in the above configuration, the present test is a strict one from the viewpoint of interpolation. Note that it is not allowed to adjust the positions of interpolation nodes ($\omega_i^{p.}$) according to the location of wavefront because $\omega_i^{p.}$ must be given *a priori*.

For the sake of generality, U and the gradient $\frac{\partial U}{\partial y_1}$ (the other components are zero in the current configuration) are re-expressed with the normalised distance $\rho := \frac{|\boldsymbol{x}-\boldsymbol{y}|}{d_s^{(l)}} = \frac{x_1}{d_s^{(l)}}$ and time $\tau := \frac{t-s}{d_t^{(l)}} = \frac{t}{d_t^{(l)}}$ as follows:

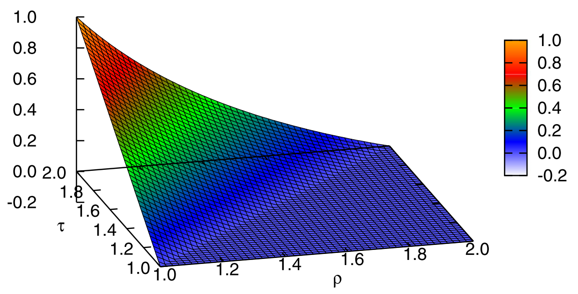
$$U = \frac{(\tau - \rho)_+}{\rho}, \qquad \frac{\partial U}{\partial y_1} = \frac{1}{d_s^{(l)}} \frac{\tau}{\rho^2} H\left(\frac{\tau}{\rho} - 1\right),$$

where $d_s^{(l)} = cd_t^{(l)}$ is used.

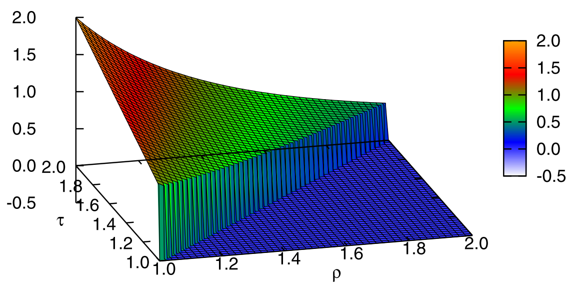
For example, to observe how CHI works in the case of $p_s = p_t = 8$, Fig. 10 plots the absolute error of the interpolated functions $\widetilde{U}$ and $d_s^{(l)} \frac{\partial \widetilde{U}}{\partial y_1}$ (where $d_s^{(l)}$ is used to eliminate the dependency on levels) against the exact values. Obviously, the error is large near the wavefront, i.e. the line $\rho = \tau$. Note that the error is slightly large near the line $\rho = 1$. This is because the order of the finite difference used in CHI at the end nodes is not two, but one (see (B.2)).

Let us consider the general case. In the same configuration as the above, the relative error of the interpolated functions compared to the exact functions are computed for various p_s and p_t. For simplicity, $p_s = p_t$ is assumed and the common value is designated by $p_{s,t}$. The relative error is computed with the l^2-norm as follows:

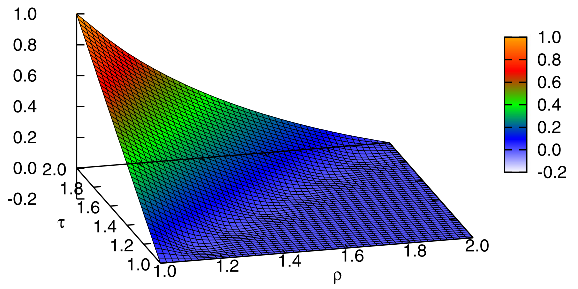
$$\sqrt{\frac{\sum_{\boldsymbol{x} \in O} \sum_{t \in I} (\widetilde{f}(\boldsymbol{x}, \boldsymbol{0}, t, 0) - f(\boldsymbol{x}, \boldsymbol{0}, t, 0))^2}{\sum_{\boldsymbol{x} \in O} \sum_{t \in I} f(\boldsymbol{x}, \boldsymbol{0}, t, 0)^2}} \qquad \text{for } (\widetilde{f}, f) = (\widetilde{U}, U) \text{ or } \left(\frac{\partial \widetilde{U}}{\partial y_1}, \frac{\partial U}{\partial y_1}\right). \tag{27}$$



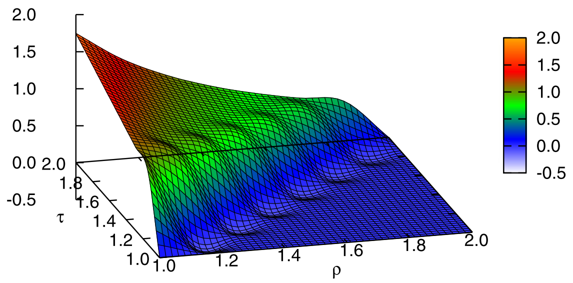
(a) U



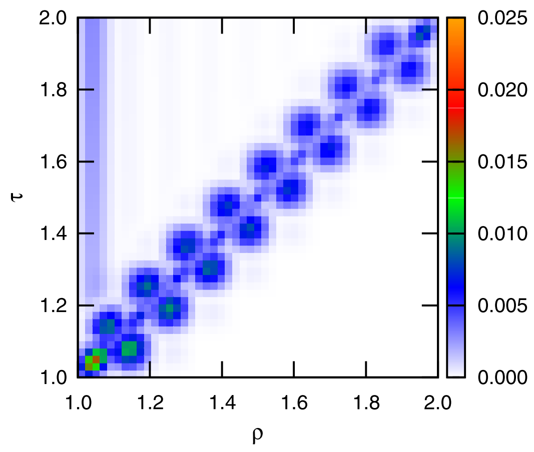
(b) $d_{\mathrm{s}}^{(l)} \frac{\partial U}{\partial y_1}$



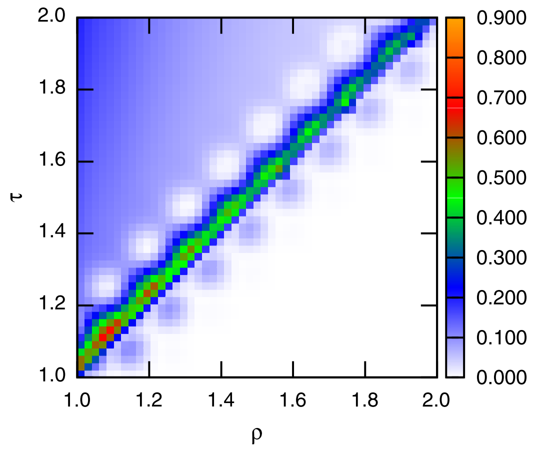
(c) $\widetilde{U}$ with $p_{\mathrm{s}} = p_{\mathrm{t}} = 8$



(d) $d_{\mathrm{s}}^{(l)} \frac{\partial \widetilde{U}}{\partial y_1}$ with $p_{\mathrm{s}} = p_{\mathrm{t}} = 8$



(e) Absolute error of $\widetilde{U}$ with $p_{\mathrm{s}} = p_{\mathrm{t}} = 8$



(f) Absolute error of $d_{\mathrm{s}}^{(l)} \frac{\partial \widetilde{U}}{\partial y_1}$ with $p_{\mathrm{s}} = p_{\mathrm{t}} = 8$

Fig. 10. Comparison of the exact and interpolated values of $U(\boldsymbol{x}, \boldsymbol{y}, t, s)$ and $d_{\mathrm{s}}^{(l)} \frac{\partial U}{\partial y_1}(\boldsymbol{x}, \boldsymbol{y}, t, s)$, where $\boldsymbol{y} = \boldsymbol{0}$, $s = 0$, $x_1 \in [d_{\mathrm{s}}^{(l)}, 2d_{\mathrm{s}}^{(l)}]$ (i.e. $\rho \in [1, 2]$), $x_2 = x_3 = 0$, and $t \in (d_{\mathrm{t}}^{(l)}, 2d_{\mathrm{t}}^{(l)}]$ (i.e. $\tau \in [1, 2]$).

Note that the level-dependent parameter $d_{\mathrm{s}}^{(l)}$ in $\frac{\partial U}{\partial y_1}$ is cancelled in this relative error, and therefore, the result shown below is independent of levels.

Fig. 11 plots the relative errors of U and $\frac{\partial U}{\partial y_1}$ against $p_{\mathrm{s,t}}$. It should be pointed out that both relative errors decrease monotonically as $p_{\mathrm{s,t}}$ increases. However, the decreasing rate especially for $\frac{\partial U}{\partial y_1}$ is very slow, i.e. the rate is between $-\frac{1}{2}$ and -1. On the other hand, the computational time scales as $\mathcal{O}(p_{\mathrm{s,t}}^4 \log p_{\mathrm{s,t}})$ owing to the FFT-accelerated M2L translation (or $\mathcal{O}(p_{\mathrm{s,t}}^8)$ if the M2M or L2L translation is considered). Consequently, using a large value of p_{s} or p_{t} can be infeasible in the current framework. This provides a possible direction for future work.

5.2. Validation

This section investigates the accuracy and speed of the CHI-based TDBIEM in comparison with the conventional TDBIEM.

Let us solve an external scattering problem for a sphere D with diameter one and centre $(\frac{1}{2}, 0, 0)$ in an infinite medium with the velocity c as one (Fig. 12). The sphere is irradiated with the sinusoidal plane pulse u^{in} in the $-x_1$ side from $t = 0$, i.e. $u^{\mathrm{in}}(\boldsymbol{x}, t) = A\{1 - \cos \frac{2\pi}{\Lambda}(x_1 - ct)\}$, where $A = \Lambda = \frac{1}{2}$ and $\cos x$ is equal to $\cos x$ if $0 \leqslant x \leqslant 2\pi$ and 1 otherwise. Here, Λ

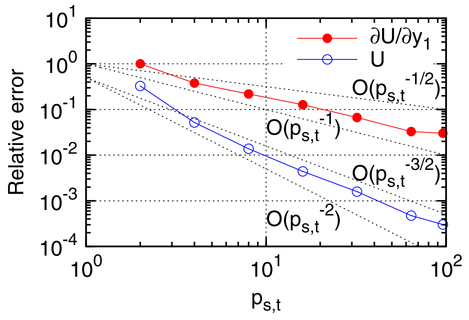


Fig. 11. Relative error of the interpolated functions compared to the exact functions with respect to the l^2-norm. Here, $p_{s,t} = 2, 4, 8, 16, 32, 64$, and 96 are examined.

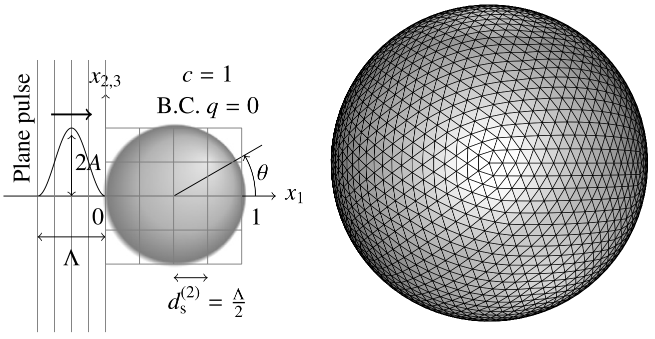


Fig. 12. (Left) Schematic illustration of the scattering problem to be solved in Section 5.2. (Right) Boundary element mesh for Case 3 or $N_s = 5120$.

Table 1
Discretisation parameters and related quantities used in Section 5.2.

Case	N_s	Δ_s Min.–Max.	Δ_t	N_t	l_{max} [†]	$M_t^{(l_{max})}$ [†]
1	320	0.138–0.163	0.04	100	2	6
2	1280	0.069–0.084	0.02	200	2	12
3	5120	0.035–0.042	0.01	400	3	24
4	20 480	0.017–0.021	0.005	800	4	48
5	81 920	0.009–0.011	0.0025	1600	5	96

[†] Determined by $Z = 100$ in the case of the proposed TDBIEM.

represents the length of the support of the pulse. Now, let $q = 0$ on the sphere. Then, this problem is reduced to solve the following BIE with respect to the total field u:

$$\frac{1}{2}u(\boldsymbol{x},t) = u^{\mathrm{in}}(\boldsymbol{x},t) - \int_0^t \int_{\partial D} \frac{\partial \Gamma}{\partial n_y}(\boldsymbol{x},\boldsymbol{y},t-s)u(\boldsymbol{y},s)\,\mathrm{d}s\,\mathrm{d}S_y \quad \text{for } \boldsymbol{x} \in \partial D,\ 0 < t < T,$$

where the duration T is set to 4, which is sufficiently long to observe that the field u on the sphere is excited and then it almost vanishes. Note that this analysis aims at verifying the double-layer potential because it can be less accurate than the single-layer potential when their kernels are approximated with interpolation, as seen in the previous section.

The exact solution of this transient problem is not available in a closed form; however the numerical solution can be computed by applying a numerical Laplace inverse transform to the corresponding exact solution in the frequency domain (e.g. [32]). This numerical solution is referred to as the semi-analytical solution.

Table 1 shows the five cases of different N_s to discretise the sphere. Following an empirical manner for the MOT scheme, Δ_t is scaled such that $\Delta_t \sim \Delta_s/c$ is satisfied, where Δ_s denotes the edge lengths of boundary elements. The total number of time-steps (N_t) is given by T/Δ_t in each case.

To solve $\mathsf{A}x^\alpha = \mathsf{r}^\alpha$ in (4) at every time-step, the iterative solver GMRES [33] is used without preconditioning. The iteration is stopped when the relative residual $\|\mathsf{A}x^\alpha - \mathsf{r}^\alpha\|_2/\|\mathsf{r}^\alpha\|_2$ is less than 10^{-5}, where $\|\cdot\|_2$ denotes the l^2-norm. Note that because A is a narrowly banded matrix (Section 2.3), there are only a few number of iterations until convergence in any case, and thus, the fraction of the solution time to the entire computational time is negligibly small.

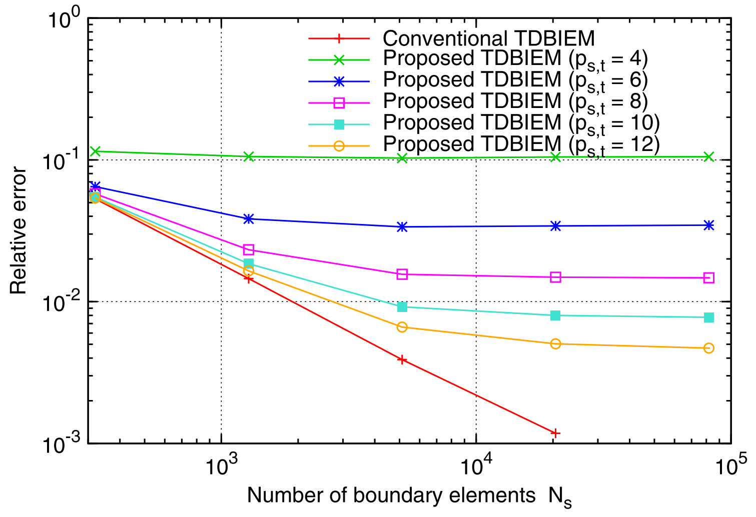


Fig. 13. Relative error of the numerical solutions compared to the semi-analytical solution.

With respect to the proposed TDBIEM, let $p_{s,t} = 4$, 6, 8, 10, and 12 for comparison. Note that $2p_s$ (respectively, $2p_t$) interpolation nodes are considered per Λ, i.e. the support length of the pulse (respectively, per Λ/c, i.e. the duration of the pulse) at level 2, which is the most critical level in view of the accuracy of interpolation (recall Section 3.4). The octree is created by assuming $Z = 100$ (recall Section 4.1). Then, as N_s increases, $l_{\max}$ (i.e. the depth of the octree) increases from 2 to 5, and $M_t^{(l_{\max})}$ (i.e. the number of time-steps per time-interval at the maximum level $l_{\max}$) increases from 6 to 96 (see Table 1).

The above configuration satisfies the assumptions in Section 4.4, where the computational complexity of the proposed TDBIEM is discussed. In fact, the domain D is fixed and the parameters Z, p_s, and p_t are constant through all the cases. However, it is not trivial if the distribution of the boundary elements on a spherical surface ∂D is flat or uniform in D, although the numerical result will approve the flat distribution, as seen subsequently.

In the following analysis, a desktop computer with sixteen cores in two Intel Xeon CPUs (Model: E5-2687W clocked at 3.1 GHz) and 128 GB RAM was used. The amount of RAM was sufficient for the conventional TDBIEM to store all the non-zero influence coefficients ($W_{ij}^{(\gamma)}$) in Cases 1–3. In Case 4, the coefficients for $\gamma \leqslant 140$ could be stored and the others were re-calculated. Meanwhile, Case 5 was not carried out because the estimated computational time was extremely long ($\gtrsim 50$ days). The Intel C++ Composer (version 13.1.0) was used as the compiler.

Fig. 13 plots the relative error of the numerical solutions compared to those of the semi-analytical solution. Here, the relative error is computed with $33N_t$ data with regard to the l^2-norm as follows:

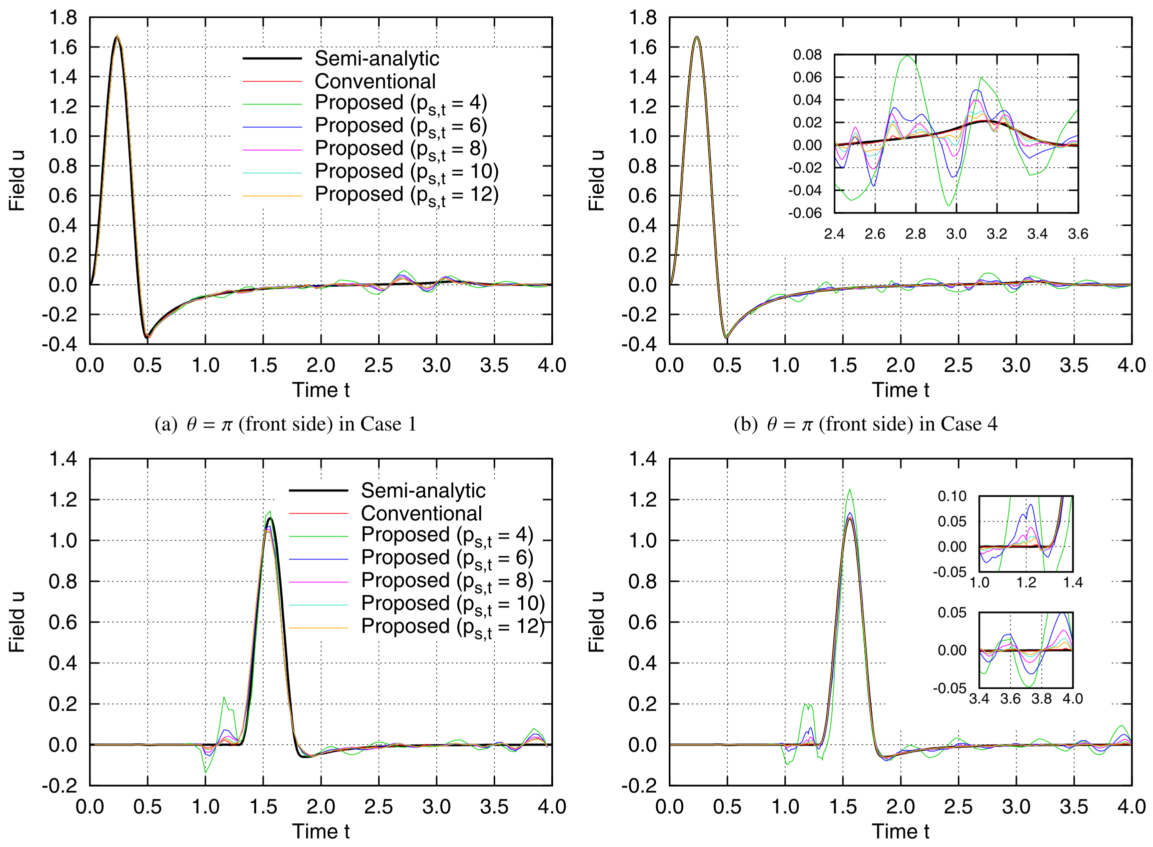
$$\sqrt{\frac{\sum_{0\leqslant i<33}\sum_{0\leqslant\alpha<N_t}(u(\boldsymbol{x}(\theta_i),t_\alpha)-v(\boldsymbol{x}(\theta_i),t_\alpha))^2}{\sum_{0\leqslant i<33}\sum_{0\leqslant\alpha<N_t}(v(\boldsymbol{x}(\theta_i),t_\alpha))^2}},$$

where u and v represent the numerical and semi-analytical solutions, respectively, and $\boldsymbol{x}(\theta_i)$ denotes the position of the collocation point whose angle θ from the x_1-axis (see Fig. 12) is the closest to $\frac{\pi i}{32}$ in all N_s collocation points. In Fig. 13, the error of the conventional TDBIEM is interpreted as the discretisation error. On the other hand, the proposed TDBIEM is dominated by the *approximation error* because of the interpolation. In fact, the proposed TDBIEM is likely to converge to the conventional TDBIEM as $p_{s,t}$ increases for every N_s.

Because the curve for every $p_{s,t}$ is approximately flat for Case 5 in Fig. 13, the errors of Case 5 can represent the approximation error of the proposed method. These errors are fitted to nearly $5.1p_{s,t}^{-2.8}$. The rate of -2.8 is better than that of the double-layer kernel in the previous test in Section 5.1. This is because the previous analysis considered only the discontinuous case (i.e. the kernel interpolated in a space–time always involves a wavefront), whereas this analysis also includes the continuous case.

To visualise the relative errors, the profiles of u for $\theta = \pi$ (the front side of the sphere) in Cases 1 and 4 are shown in Figs. 14(a) and 14(b), respectively, where the numerical solutions use the data at the collocation point whose angle θ is the closest to π in all N_s points. Likewise, Figs. 14(c) and 14(d) show the profiles for $\theta = 0$ (back side). Compared to the conventional TDBIEM, the proposed TDBIEM has larger oscillations after the pulse has passed ($t \approx 3.0$ in the front side and 3.8 in the back side) and larger over/under shoots in the back side. However, the deviations can be basically reduced by increasing $p_{s,t}$. This tendency is consistent in Fig. 13.

It should be noted that late-time instability is not observed even when the duration T is doubled. More precisely, the conventional method and proposed method (using $p_{s,t} = 4, 6, 8$) are stable until $T = 8$ in Cases 1–3 and 1–5, respectively.



(a) $\theta = \pi$ (front side) in Case 1

(b) $\theta = \pi$ (front side) in Case 4

(c) $\theta = 0$ (back side) in Case 1

(d) $\theta = 0$ (back side) in Case 4

Fig. 14. Comparison of the profiles of the total field u on the front and back sides of the sphere. The insets in (b) and (d) clearly show that the solution of the proposed TDBIEM converges to that of the conventional TDBIEM, which is in good agreement with the semi-analytical solution.

Table 2
Total computational time to perform the present analysis. The numbers in the brackets show the speedup of the proposed TDBIEM with respect to the conventional one.

Case	N_s	N_t	Conventional TDBIEM [s]	Proposed TDBIEM [s]				
				$p_{s,t} = 4$	$p_{s,t} = 6$	$p_{s,t} = 8$	$p_{s,t} = 10$	$p_{s,t} = 12$
1	320	100	0.4	0.3	1.0	4.1	19.0	45.9
				(1.4)	(0.4)	(0.1)	(0.0)	(0.0)
2	1280	200	14.2	3.1	3.8	6.9	22.3	49.7
				(4.5)	(3.7)	(2.1)	(0.6)	(0.3)
3	5120	400	595.7	76.9	80.1	93.4	187.4	397.3
				(7.8)	(7.4)	(6.4)	(3.2)	(1.5)
4	20 480	800	144 089.1†	597.0	660.8	870.2	1876.6	4214.1
			19 061.1♯	(31.9)	(28.8)	(21.9)	(10.2)	(4.5)
5	81 920	1600	n/a	6707.7	7120.3	8769.8	16 375.5	34 949.9
			609 956.4♯	(90.9)	(85.7)	(69.6)	(37.2)	(17.5)

† This time includes the additional time owing to the recalculation of the influence coefficients $W_{ij}^{(\gamma)}$ such as $\gamma > 140$, and it is not used to calculate the speedup of the proposed method.

♯ These times are ideal ones that are estimated by extrapolating Case 3 according to the scale of $N_s^2 N_t$ and are used to calculate the speedup in Cases 4 and 5.

Next, let us observe the run-time aspect of the proposed method. The total computational time required to perform each case is shown in Table 2. Note that the computational time of the conventional method in the fourth case is exceptionally long. This is because of the recalculation of influence coefficients owing to memory shortage. If all the non-zero coefficients could be stored in Cases 4 and 5, their computational times would be 32 $(= 4^2 \cdot 2)$ and 32^2 times longer, respectively, than that of Case 3. Even in comparison with these ideal times of the conventional method, the proposed method is relatively fast in Cases 4 and 5 for any $p_{s,t}$ under consideration. For clarity, the same table shows the speedup of the proposed method relative to the conventional one.

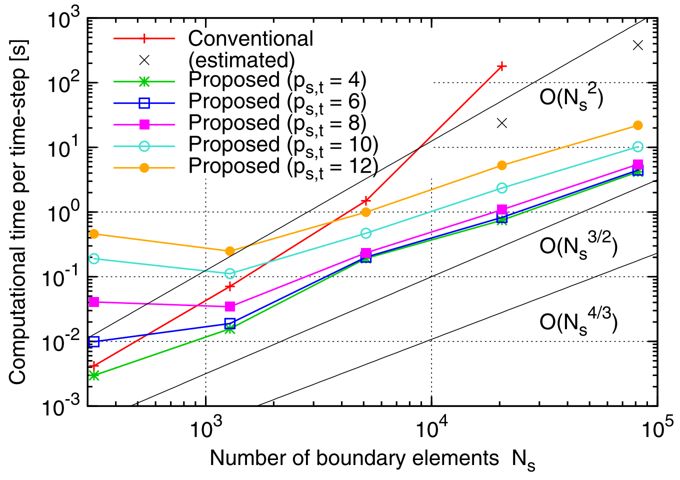


Fig. 15. Comparison for computational time per time-step. Two plots of 'estimated' are obtained by extrapolating Case 3 of the conventional method according to its complexity of $\mathcal{O}(N_s^2)$.

Fig. 15 shows the computational time per time-step, which is defined as the total computational time divided by N_t. (The actual computational time per time-step fluctuates at every time-step due to the far-field calculation, which is performed at every time-interval (not time-step); however, the computational time per time-step defined above can represent an averaged value because the far-field calculation is periodically performed.) In the figure, the observed computational complexity of the proposed TDBIEM is less than the quadratic order for every $p_{s,t}$. The complexity appears to be 4/3 rather than 3/2, implying that the distribution of the boundary elements is uniform rather than flat. On the other hand, the complexity of the conventional method is indeed $\mathcal{O}(N_s^2)$ as seen from the first three cases.

Finally, it should be noted that the performance of the proposed method is unsatisfactory when it is compared with that of the conventional method in terms of the same accuracy. For example, Fig. 13 and Table 2 show that the proposed method for $p_{s,t} = 12$ spent 34 949.9 seconds in Case 5 ($N_s = 81\,920$) and the resulting error was approximately 5×10^{-3}. For the same case, the conventional method would be 17 ($\approx 609\,956.4/34\,949.9$) times slower, where it is assumed that the method can use sufficient memory. However, Table 2 indicates that $N_s = 5120$ (Case 3) is sufficient for the conventional method to obtain the same error. Hence, the conventional method is 59 ($\approx 34\,949.9/595.7$) times faster than the proposed method in achieving the same accuracy.

Nevertheless, it is very important to solve a given problem within an acceptable time. If N_s and/or N_t is large, the application of the conventional method is in fact difficult. Note that, in order to reduce the computational time, it is generally difficult to decrease the N_s of a given model (much less a complicated model unlike the present sphere model) with preserving its shape. Therefore, to manage such complicated (thus large-scale) models, the proposed method is promising with respect to its high computational speed and low memory consumption. Although the accuracy of the fast method is relatively low, the improvement is not impossible if larger parameters (p_s and p_t) are used at the cost of computational time.

5.3. Demonstration

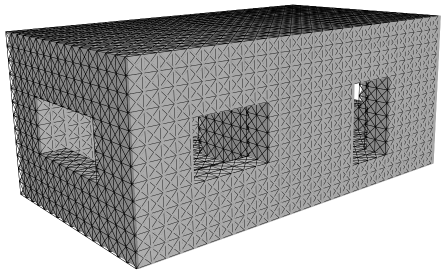
To observe the applicability of the proposed TDBIEM in more complicated problems, let us consider a model of a rectangular-parallelepiped house (dimensions $1.4 \times 0.8 \times 0.6$) located in an infinite medium (Fig. 16). The house has two rooms connected to each other and is open to the outside space through an entrance and three windows. The boundary condition $q = 0$ is applied to all the surfaces of the model. In this analysis, the following point source $f(\boldsymbol{x}, t)$ is given:

$$f(\boldsymbol{x}, t) := A\left(1 - \cos\frac{2\pi ct}{\Lambda}\right)\delta(\boldsymbol{x} - \boldsymbol{q}),$$

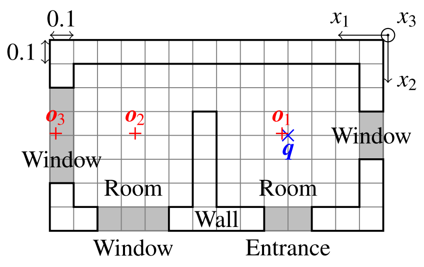
where the position $\boldsymbol{q}$ of the source is set to the centre of the RHS room in Fig. 16, i.e. $\boldsymbol{q} := (0.4, 0.4, 0.3)$. Further, $A = \Lambda = \frac{1}{2}$. The BIE to be solved here is given by the previous BIE (see Section 5.2) where u^{in} is formally replaced with

$$\int_0^t \int_D \Gamma(\boldsymbol{x} - \boldsymbol{y}, t - s) f(\boldsymbol{y}, s)\, \mathrm{d}s\, \mathrm{d}V_y = \frac{A}{4\pi|\boldsymbol{x} - \boldsymbol{q}|}\left(1 - \cos\frac{2\pi}{\Lambda}\bigl(|\boldsymbol{x} - \boldsymbol{q}| - ct\bigr)\right).$$

In the following case, let $c = 1$, $\Delta_t = 0.04$, and $N_t = 512$. Thus, $T := N_t\Delta_t = 20.48$. For the fast TDBIEM, let $l_{\max}$ (the maximum level of the hierarchy) be two and $p_{s,t} = 6$ or 8.



(a) Boundary element mesh using 12704 triangular elements. The dimensions are $1.4 \times 0.8 \times 0.6$.



(b) Layout of the house (top view). The x_3-coordinate of the floor and ceiling of the rooms are 0.1 and 0.5, respectively. Each square of dimension 0.1×0.1 is discretised with 16 boundary elements as shown in the left figure.

Fig. 16. Model of a rectangular-parallelepiped house with two rooms. The following observation points are considered: $\boldsymbol{o}_1 = (0.43, 0.39, 0.10)$, $\boldsymbol{o}_2 = (1.08, 0.39, 0.10)$, and $\boldsymbol{o}_3 = (1.38, 0.39, 0.20)$, where $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ are near the centres of the floor of the rooms and $\boldsymbol{o}_3$ is on the stand of the window in the LHS room.

Fig. 17 shows the profiles of u at three observation points $\boldsymbol{o}_{1,2,3}$ (see Fig. 16 for their positions). The proposed TDBIEM that uses $p_{s,t} = 6$ deviates significantly from the conventional TDBIEM with time. Meanwhile, $p_{s,t} = 8$ is more stable and in good agreement with the conventional TDBIEM. In this analysis, the value of u ranged from -1.23 to 1.48 in the proposed TDBIEM that uses $p_{s,t} = 8$, whereas it ranged from -1.28 to 1.49 in the conventional TDBIEM. Fig. 18 shows the snapshots of u inside the house in the case of the fast TDBIEM that uses $p_{s,t} = 8$. The corresponding animation is available from the journal's webpage.

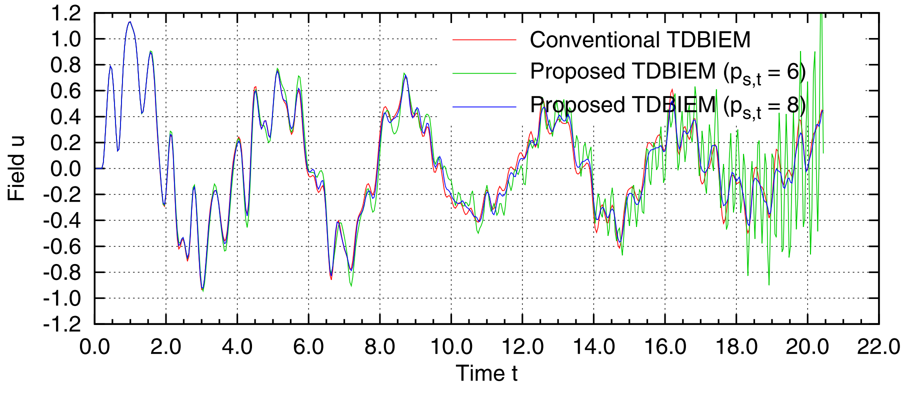
Note that the conventional TDBIEM (and thus the proposed TDBIEM) can become unstable with time, although no solution is considered for this late-time instability issue in this paper. In fact, when N_t is doubled (then, $T = 0.04 \cdot 1024 = 40.96$), the conventional method diverged gradually after $t \approx 30$. However, with respect to its early-time behaviour shown in Figs. 17 and 18, the conventional method seems reliable as a reference for the proposed method from the physical viewpoint that (i) the field u inside the house decays gradually because of the energy radiation to the outside space through the entrance and windows and (ii) the profiles at $\boldsymbol{o}_2$ and $\boldsymbol{o}_3$ (recall their locations relative to the source location $\boldsymbol{q}$) are similar in shape but different in phase (because of the propagation distance from the source) and amplitude (because of the radiation effect).

To verify the stability of the proposed TDBIEM, further investigation is necessary. It is worth noting that a large $p_{s,t}$ (if available) can be a remedy for the divergence of the solution. In addition, a small Δ_t compared to Δ_s/c can cause instability with time. In this regard, because a very small Δ_t can also make the conventional TDBIEM unstable, the instability of both TDBIEMs may be caused by the collocation method or low-order elements. Therefore, it may be interesting to enhance the proposed framework to the Galerkin method and high-order elements.

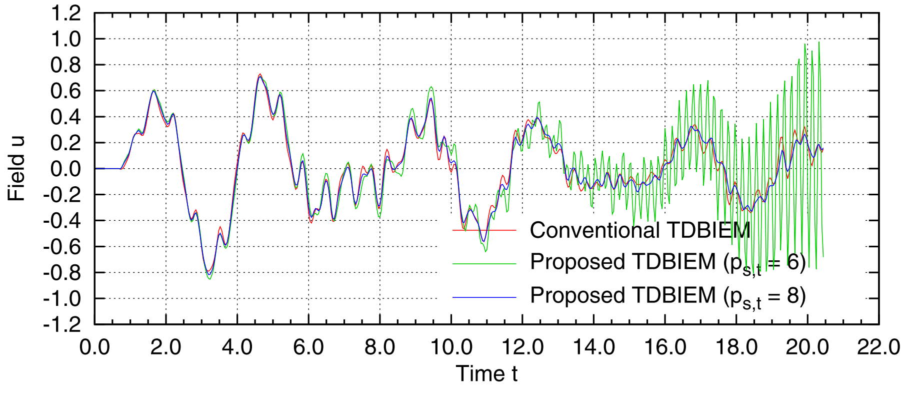
With respect to the computational time, the conventional TDBIEM spent 1171 s, while the proposed TDBIEM with $p_{s,t} = 8$ spent 531 s. Furthermore, when $l_{\max}$ was changed to three (respectively, four), the proposed method spent 333 s (respectively, 607 s) with a slight change in u; u ranged from -1.22 to 1.52 (respectively, from -1.21 to 1.49). Hence, a 3.5 ($\approx 1171/333$) times speedup was achieved by the fast TDBIEM in this simulation.

6. Conclusion

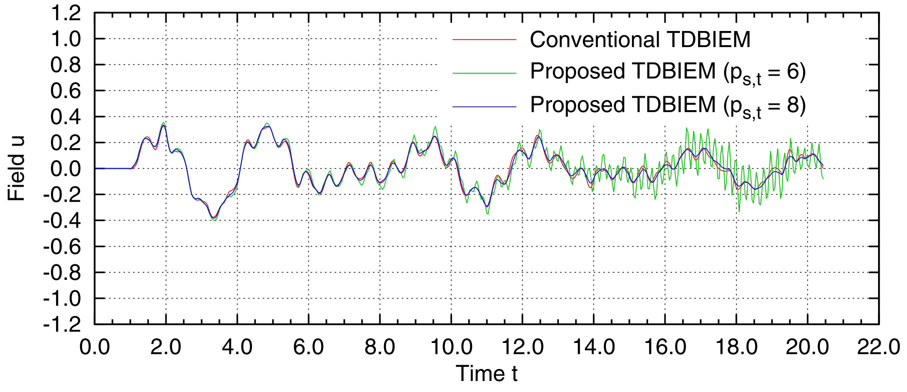
A novel fast time-domain boundary integral equation method (TDBIEM) to solve initial–boundary value problems for the three-dimensional wave equation in the low-frequency regime is proposed in this paper, based on the studies of the plane-wave time-domain (PWTD) algorithm for wave problems [6,14] and the interpolation-based fast multipole method (FMM) particularly for the transient heat problem [23]. The cubic Hermite interpolation, which uses a finite difference for the first-order derivative, was applied to re-express the fundamental solution of the wave equation in the form of the separation of variables with respect to both spatial and temporal variables. Similar to ordinary FMMs, the degenerate form that was derived enabled us to formulate the (so-called) multipole and local expansions as well as the translation formulae of multipole-moments and local-coefficients across two levels of a multilevel space–time hierarchy, which is a compound of an octree for space and a binary tree for time. By using these interpolation-based formulae systematically on a multilevel hierarchy in a similar manner to the PWTD algorithm, one can construct a fast algorithm to evaluate the retarded potential with respect to the elapsed time in the boundary integral equation for the wave equation. The proposed algorithm can reduce the computational complexity of the conventional (marching-on-in-time-based) TDBIEM, which is based on the collocation method and adopts the piecewise-constant spatial basis and the piecewise-linear temporal basis, from $\mathcal{O}(N_s^2 N_t)$ to $\mathcal{O}(N_s^{1+\delta} N_t)$, where N_s and N_t denote the numbers of boundary elements and time-steps, respectively, and the exponent δ is 1/3 or 1/2 according to the assumption of the spatial distribution of boundary elements (Section 4.4). In comparison with the PWTD-based TDBIEM, the proposed interpolation-based TDBIEM is semi-fast; however, the numerical implementation is easy because the formulation is simple. The result of some numerical experiments indicates that the fast TDBIEM can control the precision by the interpolation parameters (p_s and p_t), although this paper does not conduct any theoretical



(a) $\boldsymbol{o}_1 = (0.43, 0.39, 0.10)$.



(b) $\boldsymbol{o}_2 = (1.08, 0.39, 0.10)$.



(c) $\boldsymbol{o}_3 = (1.38, 0.39, 0.20)$.

Fig. 17. Comparison of the profiles of the total field u at three observation points. From an acoustic viewpoint, the echo of the given pulse decays because of the sound propagation to the outside. This is consistent with the results of the conventional TDBIEM and the proposed one that uses $p_{s,t} = 8$, while $p_{s,t} = 6$ makes the proposed TDBIEM relatively unstable.

error analysis. In addition, the proposed solver significantly outperformed the conventional one in large-scale problems ($\gtrsim 10^4$) with the relative error $\sim 10^{-2}$ for p_s, $p_t \lesssim 10$. In summary, the proposed TDBIEM is likely to be useful for solving large-scale problems, for example, in acoustics.

A direction for future work is to reduce the computational cost of the time-consuming M2L translation (see Sections 4.5 and 5.1). This reduction allows not only to shorten the CPU time, but also to use large p_s and p_t, which will allow overcoming the potential instability of the proposed algorithm in long-time simulations. A possible candidate is some low-rank approximation such as a truncated singular value decomposition or adaptive cross approximation investigated for the black-box and the directional FMMs [21]. Furthermore, the application of the proposed interpolation-based FMM to other wave problems such as electromagnetics and elastodynamics is an interesting and important future direction with respect to science and engineering.

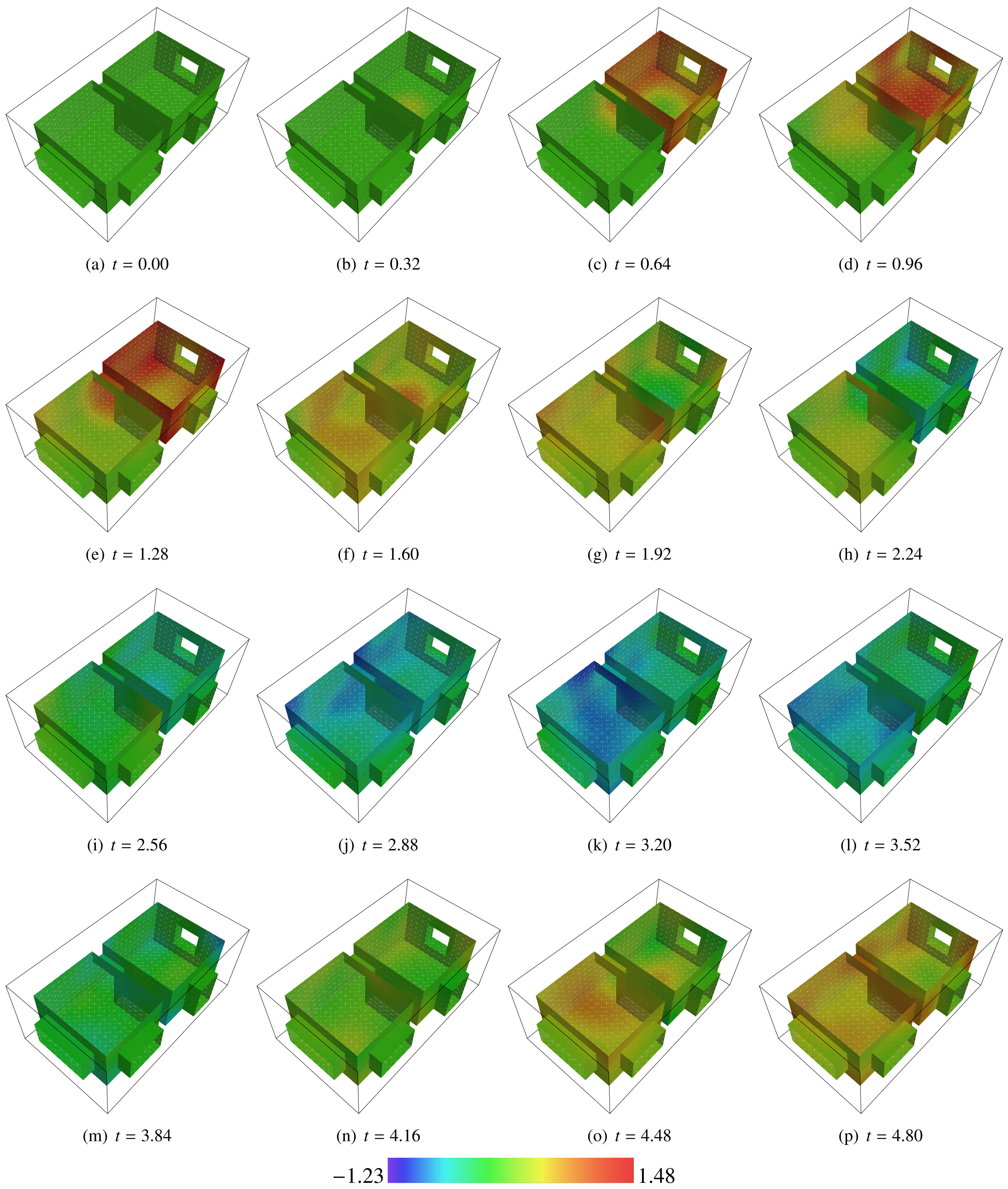


Fig. 18. Snapshots of the distribution of u from the 0th to 120th time-step every 8 time-steps. This is the result of the analysis with the proposed TDBIEM using $p_{s,t} = 8$, where u ranges from -1.23 to 1.48. To observe the inside of the house, the outside walls and roof are removed.

Acknowledgements

The author would like to thank Dr. H. Yoshikawa of Kyoto University for his useful discussions especially on the conventional TDBIEM. This work was supported by JSPS KAKENHI (Grand-in-Aid for Challenging Exploratory Research) Grant Number 24656072.

Appendix A. Calculation of the computational complexity of the proposed TDBIEM

From the assumptions mentioned in Section 4.4, the numbers of floating-number operations (flops) for the major stages can be calculated. First, let us consider the case where boundary elements are distributed uniformly in a fixed 3D domain. In this case, there are 8^l cells at level l and $N_s = Z \cdot 8^{l_{\max}}$, i.e. $2^{l_{\max}} = (N_s/Z)^{1/3}$ holds. Then, the flops are counted as follows:

1. Near-field calculation in (21):

$$\begin{aligned}&(\text{flops per leaf and time-step}) \times (\text{\# of leaves}) \times (\text{\# of time-steps})\\ &\sim Z \cdot 27Z \cdot \gamma_{\min}^{(l_{\max})} \times 8^{l_{\max}} \times N_t\\ &\sim Z^{4/3} N_s^{2/3} N_t.\end{aligned}$$

2. Creation of moments in (14):

$$\begin{aligned}&(\text{flops per leaf and time-interval}) \times (\text{\# of leaves}) \times (\text{\# of time-intervals})\\ &\sim Z M_t^{(l_{\max})} p_s^3 p_t \times 8^{l_{\max}} \times \frac{N_t}{M_t^{(l_{\max})}}\\ &\sim p_s^3 p_t N_s N_t.\end{aligned}$$

3. M2M translation in (16):

$$\begin{aligned}&\sum_{l=2}^{l_{\max}-1} (\text{flops per cell and time-interval}) \times (\text{\# of cells}) \times (\text{\# of time-intervals})\\ &\sim \sum_{l=2}^{l_{\max}-1} p_s^6 p_t^2 \cdot 8 \times 8^l \times \frac{N_t}{M_t^{(l)}}\\ &\sim Z^{-4/3} p_s^6 p_t^2 N_s^{4/3} N_t.\end{aligned}$$

4. M2L translation in (24):

$$\begin{aligned}&\sum_{l=2}^{l_{\max}} (\text{flops per cell and time-interval}) \times (\text{\# of cells}) \times (\text{\# of time-intervals})\\ &\sim \sum_{l=2}^{l_{\max}} p_s^3 p_t (\log p_s + \log p_t) \cdot 189 \cdot (\mu + 1) \times 8^l \times \frac{N_t}{M_t^{(l)}}\\ &\sim Z^{-4/3} p_s^3 p_t (\log p_s + \log p_t) N_s^{4/3} N_t.\end{aligned}$$

   Here, the FFT-based diagonalisation method [22] is considered (see Section 3.3).
5. Another M2L translation in (25) and (26): The dominant part is the calculation of the rate of local-coefficients in (18b), which is necessary for (26). When the FFT is used for the spatial summations, the number of flops of (18b) is given by

$$\begin{aligned}&\sum_{l=2}^{l_{\max}} (\text{flops per cell and time-interval}) \times (\text{\# of cells}) \times (\text{\# of time-intervals})\\ &\sim \sum_{l=2}^{l_{\max}} \left(p_s^3 p_t + p_s^3 \log p_s\right) \cdot 189 \cdot 1 \times 8^l \times \frac{N_t}{M_t^{(l)}}\\ &\sim Z^{-4/3} p_s^3 (p_t + \log p_s) N_s^{4/3} N_t.\end{aligned}$$

6. L2L translation in (17): This is essentially the same as the M2M translation.
7. Evaluate the layer-potential by local-coefficients in (12):

$$\begin{aligned}&(\text{flops per leaf and time-interval}) \times (\text{\# of leaves}) \times (\text{\# of time-intervals})\\ &\sim Z M_t^{(l_{\max})} p_s^3 p_t \times 8^{l_{\max}} \times \frac{N_t}{M_t^{(l_{\max})}}\\ &\sim p_s^3 p_t N_s N_t.\end{aligned}$$

From this calculation, the computational complexity of the proposed method is determined as $\mathcal{O}(p_s^6 p_t^2 N_s^{4/3} N_t)$ in the case of uniform distribution. This complexity originates in the M2M and L2L; however, the most of the run-time is spent for the M2L translation whenever $p_{s,t} \sim 10$ in this study.

Note that the second and seventh items assume that the cost to compute a value of an interpolant ℓ in (12) and (14) is independent of p_s or p_t. In general, the use of local interpolations such as the cubic Hermite interpolation is assumed, which is actually used in this study.

In the case that boundary elements are distributed on a surface in a fixed 3D domain, one can show that the computational complexity is $\mathcal{O}(p_s^6 p_t^2 N_s^{3/2} N_t)$ because there exists 4^l cells at level l, and thus $N_s = Z \cdot 4^{l_{\max}}$, i.e. $2^{l_{\max}} = (N_s/Z)^{1/2}$ holds.

Appendix B. Cubic Hermite interpolation

Consider a function $f(x)$ defined on $[-1, 1]$. Let f_k and $\dot{f}_k$ be the function value and its derivative at node ω_k^p, respectively, where $-1 \leqslant \omega_0^p < \cdots < \omega_{p-1}^p \leqslant 1$. In the cubic Hermite interpolation, the approximated function, which is denoted by $\widetilde{f}(x)$ on the interval $[\omega_k^p, \omega_{k+1}^p]$ $(k = 0, \ldots, p-2)$, is given by

$$\widetilde{f}(x) = A(t) f_k + B(t)\big(\omega_{k+1}^p - \omega_k^p\big)\dot{f}_k + C(t) f_{k+1} + D(t)\big(\omega_{k+1}^p - \omega_k^p\big)\dot{f}_{k+1}, \tag{B.1}$$

where $t := \frac{x-\omega_k^p}{\omega_{k+1}^p-\omega_k^p} \in [0, 1]$, and A, B, C, and D are Hermite basis functions defined as

$$A(t) := 2t^3 - 3t^2 + 1, \qquad B(t) := t^3 - 2t^2 + t, \qquad C(t) := -2t^3 + 3t^2, \qquad D(t) := t^3 - t^2.$$

Here, these functions are determined such that $\widetilde{f}(\omega_k^p) = f_k$ and $\dot{\widetilde{f}}(\omega_k^p) = \dot{f}_k$ are satisfied.

Because the proposed method does not use derivatives (see (10)), each $\dot{f}$ is approximated as follows with the second-order (three-point) finite difference at each node, except for the end nodes:

$$\dot{f}_0 \approx s_0, \qquad \dot{f}_i \approx \frac{s_{i-1} + s_i}{2} \quad \text{for } i = 1, \ldots, p-2, \qquad \dot{f}_{p-1} \approx s_{p-2}, \tag{B.2}$$

where $s_i := \frac{f_{i+1}-f_i}{\omega_{i+1}^p-\omega_i^p}$ for $i = 0, \ldots, p-2$. Substituting (B.2) in (B.1), one can obtain $\widetilde{f}(x) = \sum_{i=0}^{p-1} f(\omega_i^p)\ell_i(x)$, where the interpolants $\ell_i(x)$ are cubic polynomials consisting of A, B, C, and/or D.

Appendix C. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.jcp.2013.11.008.

References

[1] V. Rokhlin, Rapid solution of integral equations of classical potential theory, J. Comput. Phys. 60 (2) (1985) 187–207, http://dx.doi.org/10.1016/0021-9991(85)90002-6.

[2] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 325–348, http://dx.doi.org/10.1016/0021-9991(87)90140-9.

[3] V. Rokhlin, Rapid solution of integral equations of scattering theory in two dimensions, J. Comput. Phys. 86 (2) (1990) 414–439, http://dx.doi.org/10.1016/0021-9991(90)90107-C.

[4] V. Rokhlin, Diagonal forms of translation operators for the Helmholtz equation in three dimensions, Appl. Comput. Harmon. Anal. 1 (1993) 82–93, http://dx.doi.org/10.1006/acha.1993.1006.

[5] Y.J. Liu, S. Mukherjee, N. Nishimura, M. Schanz, W. Ye, A. Sutradhar, E. Pan, N.A. Dumont, A. Frangi, A. Saez, Recent advances and emerging applications of the boundary element method, Appl. Mech. Rev. 64 (3) (2011) 030802, http://dx.doi.org/10.1115/1.4005491.

[6] A. Ergin, B. Shanker, E. Michielssen, Fast evaluation of three-dimensional transient wave fields using diagonal translation operators, J. Comput. Phys. 146 (1) (1998) 157–180, http://dx.doi.org/10.1006/jcph.1998.5908.

[7] R. Kress, Linear Integral Equations, second edition, Appl. Math. Sci., vol. 82, Springer Verlag, New York, 1999.

[8] A. Ergin, B. Shanker, E. Michielssen, The plane-wave time-domain algorithm for the fast analysis of transient wave phenomena, IEEE Antennas Propag. Mag. 41 (4) (1999) 39–52, http://dx.doi.org/10.1109/74.789736.

[9] K. Aygün, S.E. Fisher, A.A. Ergin, B. Shanker, E. Michielssen, Transient analysis of multielement wire antennas mounted on arbitrarily shaped perfectly conducting bodies, Radio Sci. 34 (4) (1999) 781–796, http://dx.doi.org/10.1029/1998RS900040.

[10] W. Chew, E. Michielssen, J.M. Song, J.M. Jin (Eds.), Fast and Efficient Algorithms in Computational Electromagnetics, Artech House, Inc., Norwood, MA, USA, 2001.

[11] B. Shanker, A. Ergin, M. Lu, E. Michielssen, Fast analysis of transient electromagnetic scattering phenomena using the multilevel plane wave time domain algorithm, IEEE Trans. Antennas Propag. 51 (3) (2003) 628–641, http://dx.doi.org/10.1109/TAP.2003.809054.

[12] K. Aygun, B. Fischer, J. Meng, B. Shanker, E. Michielssen, A fast hybrid field-circuit simulator for transient analysis of microwave circuits, IEEE Trans. Microw. Theory Tech. 52 (2) (2004) 573–583, http://dx.doi.org/10.1109/TMTT.2003.821929.

[13] A.A. Ergin, B. Shanker, E. Michielssen, Fast transient analysis of acoustic wave scattering from rigid bodies using a two-level plane wave time domain algorithm, J. Acoust. Soc. Am. 106 (1999) 2405–2416, http://dx.doi.org/10.1121/1.428077.

[14] A.A. Ergin, B. Shanker, E. Michielssen, Fast analysis of transient acoustic wave scattering from rigid bodies using the multilevel plane wave time domain algorithm, J. Acoust. Soc. Am. 107 (2000) 1168–1178, http://dx.doi.org/10.1121/1.428406.

[15] T. Takahashi, N. Nishimura, S. Kobayashi, A fast BIEM for three-dimensional elastodynamics in time domain, Eng. Anal. Bound. Elem. 27 (5) (2003) 491–506, http://dx.doi.org/10.1016/S0955-7997(02)00157-1.
[16] Y. Otani, T. Takahashi, N. Nishimura, A fast boundary integral equation method for elastodynamics in time domain and its parallelisation, in: M. Schanz, O. Steinbach (Eds.), Boundary Element Analysis, in: Lect. Notes Appl. Comput. Mech., vol. 29, Springer, Berlin Heidelberg, 2007, pp. 161–185, doi:10.1007/978-3-540-47533-07.
[17] M. Vikram, B. Shanker, Fast evaluation of time domain fields in sub-wavelength source/observer distributions using accelerated Cartesian expansions (ACE), J. Comput. Phys. 227 (2) (2007) 1007–1023, http://dx.doi.org/10.1016/j.jcp.2007.08.017.
[18] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626, http://dx.doi.org/10.1016/j.jcp.2003.11.021.
[19] A. Dutt, M. Gu, V. Rokhlin, Fast algorithms for polynomial interpolation, integration, and differentiation, SIAM J. Numer. Anal. 33 (5) (1996) 1689–1711, http://dx.doi.org/10.1137/0733082.
[20] W. Fong, E. Darve, The black-box fast multipole method, J. Comput. Phys. 228 (2009) 8712–8725, http://dx.doi.org/10.1016/j.jcp.2009.08.031.
[21] M. Messner, B. Bramas, O. Coulaud, E. Darve, Optimized M2L kernels for the Chebyshev interpolation based fast multipole method, http://arxiv.org/abs/1210.7292, Oct. 2012.
[22] D. Schobert, T. Eibert, Low-frequency surface integral equation solution by multilevel Green's function interpolation with fast Fourier transform acceleration, IEEE Trans. Antennas Propag. 60 (3) (2012) 1440–1449, http://dx.doi.org/10.1109/TAP.2011.2180306.
[23] J. Tausch, A fast method for solving the heat equation by layer potentials, J. Comput. Phys. 224 (2) (2007) 956–969, http://dx.doi.org/10.1016/j.jcp.2006.11.001.
[24] M. Bonnet, Boundary Integral Equation Methods for Solids and Fluids, John Wiley & Sons, West Sussex, 1999.
[25] H. Yoshikawa, N. Nishimura, An improved implementation of time domain elastodynamic BIEM in 3D for large scale problems and its application to ultrasonic NDE, Electron. J. Bound. Elem. 1 (2) (2003) 201–217, http://reaper64.scc-net.rutgers.edu/journals/index.php/ejbe/article/view/758/1979.
[26] H. Yoshikawa, Study on the application of the time-domain boundary integral equation method to the non-destructive evaluation using laser ultrasonic measurement, Ph.D. thesis, Kyoto University, September 2003 (in Japanese).
[27] M. Frigo, S. Johnson, The design and implementation of FFTW3, Proc. IEEE 93 (2) (2005) 216–231, http://dx.doi.org/10.1109/JPROC.2004.840301.
[28] M. Frigo, S.G. Johnson, FFTW (manual for version 3.3.3), http://www.fftw.org, November 2012.
[29] M. Abramowitz, I. Stegun (Eds.), Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables, eighth Dover printing, Dover, New York, 1972.
[30] The OpenMP API Specification for Parallel Programming, http://openmp.org/wp.
[31] Intel Advanced Vector Extensions (Intel AVX), http://software.intel.com/en-us/avx.
[32] J.J. Bowman, T.B.A. Senior, P.L.E. Uslenghi, Electromagnetic and Acoustic Scattering by Simple Shapes, revised edition, Hemisphere Publishing Corp., New York, 1987.
[33] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869, http://dx.doi.org/10.1137/0907058.