Graz University of Technology
Institute of Applied Mechanics

Preprint Series

Preprint No 01/2014

Fast and Data Sparse Time Domain BEM for Elastodynamics

Bernhard Kager, Martin Schanz
Institute of Applied Mechanics, Graz University of Technology

Published in *Engineering Analysis with Boundary Elements*,
50, 212–223, 2015
DOI: 10.1016/j.enganabound.2014.08.001
Latest revision: August 8, 2014

Abstract

Wave propagation is of great interest for all fields of science and engineering. Particularly, for the case of semi-infinite and infinite domains, the Boundary Element Method is an appropriate numerical method for the simulation of such problems. The presented formulation establishes a data efficient and fast boundary element formulation for the 3-d elastodynamic problem. Approximations of the inherently present temporal convolution are computed via the Convolution Quadrature Method in a nonstandard manner. Contrary to utilizing Cauchy's integral formula, this paper establishes a 'direct' convolution weight evaluation. The application of a cubic spline interpolation on these analytic functions and an appropriate clustering strategy, finally, yield a fast and data efficient formulation that is validated with numerical examples.

1 Introduction

The numerical investigation of transient problems is of great interest for all disciplines in engineering and science. A large variety of methods is available for the solution of such problems – each of them offering certain advantages and disadvantages. The Boundary Element Method (BEM) is well established in the field of transient analysis, particularly for the treatment of infinite and semi-infinite domains. Despite the major advantage, the reduction of dimensionality, the drawback of the method is evident in terms of computational complexity and storage requirements. Within the last two decades significant improvements have led to an increased competitive position of the method. The reader is recommended to consult, e.g., Ergin et al. [13], Takahashi et al. [39], Messner and Schanz [29] and Takahashi [38] – just to mention a few publications that are related to 'fast methods' dealing with wave propagation.

Due to its special character, an implementation of transient elastodynamics implicates to deal with the inherently present temporal convolution as well as with strategies to improve the efficiency in terms of computational time and memory consumption. From a general point of view, two strategies are commonly used to treat the convolution in time. The first approach exploits temporal shape functions with local support to describe the time-dependency of the respective quantities. Such procedures where used right from the beginning in basic works of, e.g., Mansur [25], Mansur and Brebbia [26] and Antes [3]. Unfortunately, this approach suffers from stability issues – a research topic addressed by authors like Siebrits and Peirce [36] and Birgisson et al. [7].

Due to works of Lubich [22, 23] in the late eighties, a new method for the treatment of the temporal convolution was given and might be regarded as the second commonly used strategy. Consequentially, Lubich and Schneider [24] applied the procedures to time-dependent boundary integral equations. Schanz and Antes [34] followed by applying it to transient elastodynamics with subsequent publications in visco- and poroelasticity [33, 32, 35]. As a result, the convolution quadrature method (CQM) found its way into boundary element formulations for a broad range of problems in science and industrial applications. The reader is recommended to consult [4] for more details and a large list of references.

The evaluation of the convolution weights, a crucial task in the application of the CQM, usually utilizes numerical approximations of Cauchy's integral formula. Most of the implementations use this strategy since it allows to use sophisticated fundamental solutions, e.g., in poroelasticity. Quite recently, a formulation for the numerical treatment of the scalar wave equation proposed by Hackbusch et al. [18] uses a 'direct' weight evaluation instead of the Cauchy integral. In this formulation, based on a BDF-2 multistep method, an analytic expression is given to compute the weights exactly. Even if the scalar-type fundamental solution is a rather simple expression, the resulting weight functions turn out to be quite involved. However, the direct computation reveals some interesting properties that correspond to the wave nature of the problem. These characteristics in turn enable efficiency improvements in terms of computational effort and storage requirements. While Hackbusch et al. [18] utilize the BDF-2 scheme as underlying multistep method for the CQM, Monegato et al. [30] show the possible practical applicability of higher order schemes – even though an essential criterion (A-stability of the underlying multistep method) is partially not met.

Yet, the direct convolution weight computation is confined to the scalar wave equation – thus,

the aim of the present paper is to extend the approach to linear elastodynamics. This paper establishes a boundary element formulation for the treatment of transient elastodynamics based on the CQM that uses the concept of direct weight evaluation. Like in the scalar case, the wave nature is recovered and a chance for efficiency improvements is given. The paper is organized in the following way: A brief introduction to both, the underlying boundary integral equation as well as the convolution quadrature method is given in sections 2 and 3, respectively. The main part, the direct computation of the tensor-valued weights is shown in section 4 accompanied by some illustrations concerning the mentioned wave nature of the problem. In section 5, the concepts for an efficient and fast implementation based on cubic spline interpolation are introduced and implementation details are provided. Finally, the proposed formulation is tested and validated with numerical examples in section 6.

Throughout the whole paper, lower subscripts indicate tensor- and vector-valued quantities while bold-faced lower case letters denote vectors in $\mathbb{R}^3$. Additionally, Einstein's summation convention is used and $(\cdot)_{i,j} = \frac{\partial}{\partial x_j}(\cdot)_i$ holds. A hat $\widehat{(\cdot)}$ denotes Laplace-transformed quantities with the Laplace parameter $s \in \mathbb{C}$ s.t. $\Re s > 0$.

2 Basic Equations

Consider a domain $\Omega \subset \mathbb{R}^3$ with boundary $\Gamma = \Gamma_N \cup \Gamma_D = \partial\Omega$ composed of both, a Neumann boundary Γ_N and Dirichlet boundary Γ_D. Assuming the domain to be occupied by a linear elastic continuum the governing equation inside the body is represented by the Lamé-Navier equation

$$\left(c_1^2 - c_2^2\right) u_{j,ji}(\mathbf{x},t) + c_2^2\, u_{i,jj}(\mathbf{x},t) + \frac{\partial^2 u_i(\mathbf{x},t)}{\partial t^2} = 0 \text{ with } i,j = 1,2,3. \tag{1}$$

Note that in equation (1) and in the sequel, body forces are assumed to vanish. The constant parameters c_1 and c_2 denote the wave velocities corresponding to the compression and the shear wave, respectively. Their relation to the density ρ and the Lamé parameters λ and μ is given via the identities $c_1^2 = (\lambda+2\mu)/\rho$ and $c_2^2 = \mu/\rho$. $u_i(\mathbf{x},t)$ denotes the displacement in the i-th direction of a material particle located at $\mathbf{x} \in \Omega$ at time $t \in (0,T) \subset \mathbb{R}^+$, where T is a fixed end time. For subsequent derivations, all initial conditions are assumed to vanish, i.e., $u_i(t=0) = \dot{u}_i(t=0) = 0$. The boundary integral equation (see [11] for instance) in its space-time representation

$$C_{ij}(\mathbf{x})\, u_j(\mathbf{x},t) = \int_\Gamma U_{ij}(\mathbf{r},t) * t_j(\mathbf{y},t)\, \mathrm{d}s_\mathbf{y} - \int_\Gamma T_{ij}(\mathbf{r},t) * u_j(\mathbf{y},t)\, \mathrm{d}s_\mathbf{y},\ \mathbf{x} \in \Gamma \tag{2}$$

serves as a prerequisite for a boundary element formulation. Equation (2) contains the time-dependent Neumann datum $t_i(\mathbf{x},t)$ and Dirichlet datum $u_i(\mathbf{x},t)$ on the boundary as well as the displacement fundamental solution $U_{ij}(\mathbf{r},t)$, the traction fundamental solution $T_{ij}(\mathbf{r},t)$ and the integral free term $C_{ij}(\mathbf{x})$. The asterisk $(*)$ denotes convolution in time and the vector valued difference between load and field point is denoted by $\mathbf{r} = \mathbf{y} - \mathbf{x}$. In the following the Euclidean distance between these points will be denoted with $r = |\mathbf{r}|$ and will be called radius.

Remark 1. Although the time-domain fundamental solutions $U_{ij}(\mathbf{r},t)$ and $T_{ij}(\mathbf{r},t)$ are well known in closed form (see [2] for instance), the special treatment of the convolution in the

sequel allows to work solely with the Laplace-transformed fundamental solutions $\hat{U}_{ij}(\mathbf{r},s)$ and $\hat{T}_{ij}(\mathbf{r},s)$ (see section 4).

3 Convolution Quadrature Method

The basic problem of finding approximations of the convolution integral

$$y(t) = f(t) * g(t) = \int_0^t f(t-\tau)\, g(\tau)\, \mathrm{d}\tau, \tag{3}$$

involving two temporal functions $f(t)$ and $g(t)$ is crucial for the time-domain boundary element method since this convolution is inherently present in the underlying boundary integral equation (2). A well established method to acquire such approximations is the convolution quadrature method developed by Lubich [22, 23]. A detailed and neatly arranged derivation of the method is given in [32]. The time interval of interest $t \in [0,T]$ is split into N_t equidistant intervals $\Delta t = {}^{T}/_{N_t}$. If the Laplace-transformed function $\hat{f}(s)$ exists, the approximation reads as

$$y(n\Delta t) = \int_0^{n\Delta t} f(n\Delta t - \tau)\, g(\tau)\, \mathrm{d}\tau \approx \sum_{m=0}^{n} \omega^{n-m} g(m\Delta t) \tag{4}$$

with $n = 0, 1, ..., N_t$. Choosing the A-stable BDF-2 scheme as underlying multistep method, the weights ω^n can be obtained via

$$\omega^n = \frac{1}{n!}\frac{\partial^n}{\partial \xi^n}\left[\hat{f}\left(\frac{\gamma(\xi)}{\Delta t}\right)\right]_{\xi=0}, \text{ with } \gamma(\xi) = \frac{1}{2}\left(\xi^2 - 4\xi + 3\right) . \tag{5}$$

Note that $\gamma(\xi)$ represents the characteristic polynomial of the chosen BDF-2 scheme and $\xi \in \mathbb{C}, |\xi| < 1$.

3.1 Conventional Approach of Weight Computation

At this stage, the common approach for the weight computation utilizes Cauchy's integral formula to obtain the partial derivatives with respect to ξ in equation (5). Thus, ω^n is

$$\omega^n = \frac{1}{2\pi \mathrm{i}} \int_{|\xi|=\mathcal{R}} \hat{f}\left(\frac{\gamma(\xi)}{\Delta t}\right) \xi^{-(n+1)}\, \mathrm{d}\xi. \tag{6}$$

$\mathcal{R}$ denotes the radius of a circle within the domain of analyticity of the Laplace-domain function $\hat{f}\left(\frac{\gamma(\xi)}{\Delta t}\right)$. Commonly, a trapezoidal rule is used to evaluate the integral in equation (6).

One reason for the frequent usage of Cauchy's integral formula is that this strategy can be applied for all kinds of multistep methods and functions $\hat{f}(s)$ – even complicated ones like, e.g., poroelastic fundamental solutions. As a consequence, the derivation of closed form expressions for the n-th partial derivative in (5) can be avoided. Nonetheless, this leads to another approximation that is introduced into the formulation – with additional computational effort.

Contrary to that, another strategy is the direct evaluation of the weight functions. Much more effort has to be invested in finding closed form expressions for the n-th partial derivative with respect to ξ in (5). Even for simple expressions like for the scalar wave equation (see Hackbusch et al. [18]), the resulting weight functions get quite involved. Fortunately, this strategy reveals some interesting properties that can be used for efficiency improvements. The aim of the next section is to employ this strategy for elastodynamics to finally come up with an efficient and fast boundary element formulation.

Remark 2. The proposed formulation does not utilize Cauchy's integral formula at all. Instead, the partial derivatives are evaluated exactly for each timestep n.

4 Direct Weight Computation for Elastodynamics

For the sake of readability the crucial ideas are shown solely for the displacement fundamental solution. The same strategies can be applied to the traction fundamental solution. Its expressions are listed in A.2. For the elastodynamic case the convolution involving the displacement fundamental solution and the Neumann datum is

$$U_{ij}(\mathbf{r},n\Delta t)*t_j(\mathbf{y},n\Delta t)=\int_0^{n\Delta t}U_{ij}(\mathbf{r},n\Delta t-\tau)\,t_j(\mathbf{y},\tau)\,\mathrm{d}\tau\approx\sum_{m=0}^{n}\omega_{ij}^{n-m}(\mathbf{r})\,t_j(\mathbf{y},m\Delta t)\,. \tag{7}$$

As stated in section 3, a successful application of the CQM requires the elastodynamic fundamental solution in Laplace-domain that is defined according to [9] as

$$\hat{U}_{ij}(\mathbf{r},s)=\frac{1}{4\pi\rho c_2^2}\left(\delta_{ij}\psi(r,s)-r_{,i}r_{,j}\chi(r,s)\right), \tag{8}$$

with scalar functions

$$\begin{aligned}\psi(r,s)&=\left(\frac{c_2^2}{s^2r^3}+\frac{c_2}{sr^2}+\frac{1}{r}\right)\exp\left(-\frac{sr}{c_2}\right)-\frac{c_2^2}{c_1^2}\left(\frac{c_1^2}{s^2r^3}+\frac{c_1}{sr^2}\right)\exp\left(-\frac{sr}{c_1}\right)\\ \chi(r,s)&=\left(\frac{3c_2^2}{s^2r^3}+\frac{3c_2}{sr^2}+\frac{1}{r}\right)\exp\left(-\frac{sr}{c_2}\right)-\frac{c_2^2}{c_1^2}\left(\frac{3c_1^2}{s^2r^3}+\frac{3c_1}{sr^2}+\frac{1}{r}\right)\exp\left(-\frac{sr}{c_1}\right),\end{aligned} \tag{9}$$

which depend solely on the Euclidean distance r and not on any direction $\mathbf{r}$. In the sequel $\alpha=1,2$ is an index for the wave velocities. With the aid of equations (5) and (8), the tensor-valued weights are defined by

$$\omega_{ij}^n(\mathbf{r})=\frac{1}{n!}\frac{\partial^n}{\partial\xi^n}\left[\hat{U}_{ij}\left(\mathbf{r},\frac{\gamma(\xi)}{\Delta t}\right)\right]_{\xi=0}. \tag{10}$$

Using the representation of the fundamental solution with the scalar functions $\chi(r,s)$ and $\psi(r,s)$ (9) and the abbreviations

$$\begin{aligned}\overset{n}{\psi}(r)&=\frac{1}{n!}\frac{\partial^n}{\partial\xi^n}\left[\psi\left(r,\frac{\gamma(\xi)}{\Delta t}\right)\right]_{\xi=0}\\ \overset{n}{\chi}(r)&=\frac{1}{n!}\frac{\partial^n}{\partial\xi^n}\left[\chi\left(r,\frac{\gamma(\xi)}{\Delta t}\right)\right]_{\xi=0}\end{aligned} \tag{11}$$

equation (10) might be rewritten to

$$\omega_{ij}^{n}(\mathbf{r}) = \frac{1}{4\pi\rho c_2^2}\left(\delta_{ij}\overset{n}{\psi}(r) - r_{,i}r_{,j}\overset{n}{\chi}(r)\right) . \tag{12}$$

Taking into account equations (9), the terms to consider are essentially of the form $\exp\left(-\frac{rs}{c_\alpha}\right)s^{-k}$ with $k = 0,1,2$. Hence, the expressions to be handled in (11) have the form

$$\overset{(n)}{P}{}_\alpha^k(r) = \frac{1}{n!}\frac{\partial^n}{\partial\xi^n}\left[\exp\left(-\frac{r\gamma(\xi)}{c_\alpha\Delta t}\right)\left(\frac{\gamma(\xi)}{\Delta t}\right)^{-k}\right]_{\xi=0} . \tag{13}$$

Thus, the weight is computed for a fixed radius r as a linear combination of the quantities (13) with constant prefactors. For $k = 0$ and a BDF-2 as multistep method, an explicit expression can be found in [18] involving Hermite polynomials where the n-th polynomial is denoted $H_n(x)$, particularly,

$$\overset{(n)}{P}{}_\alpha^0(r) = \frac{1}{n!}\left(\frac{r}{2c_\alpha\Delta t}\right)^{\frac{n}{2}}\exp\left(-\frac{3r}{2c_\alpha\Delta t}\right)H_n\left(\sqrt{\frac{2r}{c_\alpha\Delta t}}\right) . \tag{14}$$

Applying the general Leibniz rule for differentiation yields the two remaining terms

$$\overset{(n)}{P}{}_\alpha^1(r) = \Delta t\sum_{m=0}^{n}\left(1-3^{-(m+1)}\right)\overset{(n-m)}{P}{}_\alpha^0(r) \tag{15}$$

$$\overset{(n)}{P}{}_\alpha^2(r) = (\Delta t)^2\sum_{m=0}^{n}3^{-(2+m)}\left(4+m+3^{(m+2)}m\right)\overset{(n-m)}{P}{}_\alpha^0(r) . \tag{16}$$

Using equations (14), (15) and (16), the scalar functions $\overset{n}{\psi}(r)$ and $\overset{n}{\chi}(r)$ are constructed as linear combinations with prefactors corresponding to equations (9)

$$\overset{n}{\psi}(r) = \left(\frac{c_2^2}{r^3}\overset{(n)}{P}{}_2^2(r) + \frac{c_2}{r^2}\overset{(n)}{P}{}_2^1(r) + \frac{1}{r}\overset{(n)}{P}{}_2^0(r)\right) - \frac{c_2^2}{c_1^2}\left(\frac{c_1^2}{r^3}\overset{(n)}{P}{}_1^2(r) + \frac{c_1}{r^2}\overset{(n)}{P}{}_1^1(r)\right) \tag{17}$$

$$\overset{n}{\chi}(r) = \left(\frac{3c_2^2}{r^3}\overset{(n)}{P}{}_2^2(r) + \frac{3c_2}{sr^2}\overset{(n)}{P}{}_2^1(r) + \frac{1}{r}\overset{(n)}{P}{}_2^0(r)\right) - \frac{c_2^2}{c_1^2}\left(\frac{3c_1^2}{r^3}\overset{(n)}{P}{}_1^2(r) + \frac{3c_1}{r^2}\overset{(n)}{P}{}_1^1(r) + \frac{1}{r}\overset{(n)}{P}{}_1^0(r)\right) . \tag{18}$$

Local Support and Elastic Waves The scalar functions $\overset{n}{\psi}(r)$ and $\overset{n}{\chi}(r)$ exhibit local support in the Euclidean distance r and, consequently, the weight $\omega_{ij}^n(\mathbf{r})$ as well. Figures 1 and 2 illustrate the behavior using the parameters

$$c_1 = 1\,\mathrm{m/s}\,,\; c_2 = \sqrt{\frac{1}{2}}\,\mathrm{m/s} \quad \text{and} \quad r_3 = 0\,.$$

In figure 1, the scalar functions $\overset{n}{\psi}(r)$ and $\overset{n}{\chi}(r)$ are shown for different timesteps n. Clearly, the fact that the functions return non-zero values solely in a certain range of r becomes visible.

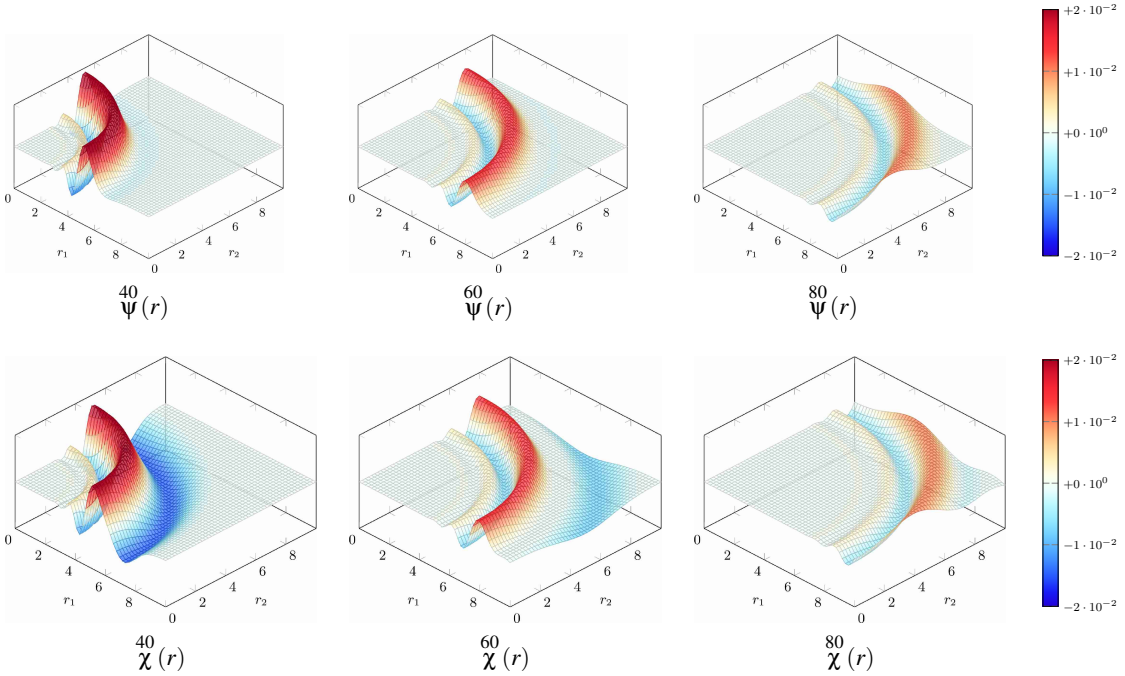


Figure 1: Surface plots at different timesteps n

The weights $\omega_{ij}^{n}(\mathbf{r})$ can be viewed as expressions that are obtained by convolution of the time domain fundamental solution and shape functions in time. A discussion about these similarities can be found in [22, 23]. Based on these observations it becomes obvious that from a physical point of view the local support of $\omega_{ij}^{n}(\mathbf{r})$ corresponds to the existence of the two elastic waves.

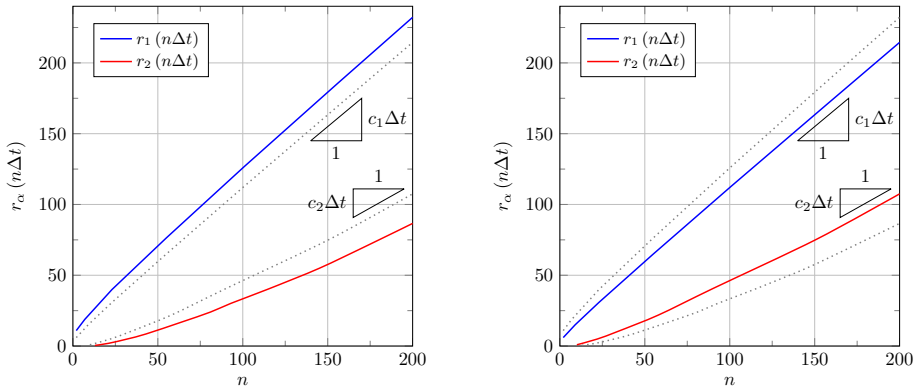
For subsequent derivations, it is essential to locate the position of the wave fronts, thus the support radii $r_\alpha(n\Delta t)$ are introduced. With the aid of these measures a distinction whether the weight function returns 'zero' (negligible) or 'non-zero' values can be made. A radius r is defined to be part of the support if

$$\sqrt{\overset{n}{\chi}(r)^2 + \overset{n}{\psi}(r)^2} \geq \varepsilon_{\text{supp}} \tag{19}$$

holds true. The evaluation of the n-th weight returns significant values solely if r lies in between the two support radii $r_1(n\Delta t)$ and $r_2(n\Delta t)$. Thus, the n-th weight is assumed to give a 'zero' contribution if $r \geq r_1(n\Delta t)$ or $r \leq r_2(n\Delta t)$ holds.

Figure 2 illustrates these radii for two different accuracies and a timestep size $\Delta t = 1\,s$. Indeed, the asymptotic behavior shows the correspondence to the compression and shear wave velocity (slope).

Recurrence Relations Bearing in mind an algorithmic implementation, a recursive computation of the weight functions is an appropriate way to ensure efficiency. The principle strategy for the derivation of the recurrence relations is shown for the first expression in A.1.



(a) $\varepsilon_{\text{supp}} = 10^{-6}$ (b) $\varepsilon_{\text{supp}} = 10^{-3}$

Figure 2: Wave front radii $r_\alpha\,(n\Delta t)$ for different support detection accuracies $\varepsilon_{\text{supp}}$

Subsequently, the required recurrences are listed:

1. Recurrence Relation for $\overset{(n)}{P^0_\alpha}(r)$

   With the aid of the recursive definition of the Hermite polynomial [1] simple computations lead to the following recursion formulas.

$$\overset{(0)}{P^0_\alpha}(r) = \exp\left(-\frac{3r}{2c_\alpha\Delta t}\right), \quad \overset{(1)}{P^0_\alpha}(r) = \frac{2r}{c_\alpha\Delta t}\overset{(0)}{P^0_\alpha}(r), \quad \overset{(n+1)}{P^0_\alpha}(r) = \frac{1}{(n+1)}\frac{r}{c_\alpha\Delta t}\left(2\overset{(n)}{P^0_\alpha}(r) - \overset{(n-1)}{P^0_\alpha}(r)\right)$$

   Remark 3. It must be stated clearly that in the case of large r or small $c_\alpha\Delta t$, the recursion might fail in an implementation with double precision data types. This numerical issue can be overcome by logarithm techniques commonly used, e.g., for the computation of binomial coefficients. For the problem sizes as shown in the following, this effect has not occurred.

2. Recurrence Relations for $\overset{(n)}{P^1_\alpha}(r)$

$$\alpha_0(r) = \overset{(0)}{P^0_\alpha}(r), \qquad \alpha_{n+1}(r) = \alpha_n(r) + \overset{(n+1)}{P^0_\alpha}(r)$$
$$\overset{(0)}{P^1_\alpha}(r) = \frac{2}{3}\Delta t\overset{(0)}{P^0_\alpha}(r), \qquad \overset{(n+1)}{P^1_\alpha}(r) = \frac{2\Delta t}{3}\left(\alpha_n(r) + \overset{(n+1)}{P^0_\alpha}(r)\right) + \frac{1}{3}\overset{(n)}{P^1_\alpha}(r)$$

3. Recurrence Relations for $\overset{(n)}{P^2_\alpha}(r)$

$$\beta_0(r) = \frac{4}{81}\overset{(0)}{P^0_\alpha}(r), \qquad \beta_{n+1}(r) = \frac{1}{3}\beta_n(r) + \frac{4}{81}\overset{(n+1)}{P^0_\alpha}(r)$$

$$\delta_0(r) = \frac{52}{27}\overset{(0)}{P^0_\alpha}(r), \qquad \delta_{n+1}(r) = \delta_n(r) + \beta_n(r) + \frac{52}{27}\overset{(n+1)}{P^0_\alpha}(r)$$

$$\zeta_0(r) = \frac{28}{9}\overset{(0)}{P^0_\alpha}(r), \qquad \zeta_{n+1}(r) = \zeta_n(r) + \delta_n(r) + \frac{28}{9}\overset{(n+1)}{P^0_\alpha}(r)$$

$$\overset{(0)}{P^2_\alpha}(r) = \frac{4}{9}(\Delta t)^2\overset{(0)}{P^0_\alpha}(r), \qquad \overset{(n+1)}{P^2_\alpha}(r) = \frac{(\Delta t)^2}{3}\left(\zeta_n(r) + \frac{4}{3}\overset{(n+1)}{P^0_\alpha}(r)\right) + \frac{1}{3}\overset{(n)}{P^2_\alpha}(r)$$

5 Spatial Discretization and Efficiency Improvements

The aim of this section is to establish strategies to perform an efficient algorithmic implementation of the boundary element formulation based on the results of section 4.

5.1 Spatial Discretization and Integration

In order to establish a fully discretized version of the space-time dependent boundary integral equation (2), a spatial discretization has to be performed. The shape functions for displacements are chosen to be linear and continuous whereas the tractions are chosen to be constant over the element and thus discontinuous. The convolution quadrature time discretization followed by a collocation scheme in space finally yields a the well known Toeplitz structure that can be solved recursively. The reader is recommended to consult, e.g., [25, 11] for more details about the solution procedures

The singular spatial integration is done numerically with the use of Duffy's transformation [12]. Since this method is suitable solely for weakly singular kernels, a regularization technique is used for the traction fundamental solution (see [21, 15, 20]). The corresponding expressions are listed in the appendix. Note that the jump $C_{ij}(\mathbf{x})$ at the spatial point $\mathbf{x} \in \Gamma$ on the boundary is computed according to [27].

5.2 Cubic Spline Interpolation

A crucial task from a computational point of view is the assemblage of the system matrix for each timestep n since the weight functions are expensive to evaluate. The largest part of the computation is spent for the evaluation of the recurrences given in section 4. To overcome this drawback a cubic spline interpolation is used on these functions. Note that for any discrete domain the Euclidean distance r between two points is bounded by $r \leq r_{\max}$. This in turn defines the interpolation interval that is further subdivided into n_r equidistant intervals of size Δr with $r_j = j\Delta r$ and $j = 0, 1, ..., n_r$.

The necessary steps to compute the interpolation are shown according to [10] exemplarily for the function $\overset{(n)}{P^0_\alpha}(r)$. The cubic spline interpolation of this function in $r \in [r_j, r_{j+1}]$ reads as

$$\begin{aligned}\overset{(n)}{P^0_\alpha}(r) \approx & \frac{1}{6\Delta r}\left((r_{j+1}-r)^3 m_j + (r-r_j)^3 m_{j+1}\right) \\ & + \frac{1}{\Delta r}\left((r_{j+1}-r)\overset{(n)}{P^0_\alpha}(r_j) + (r-r_j)\overset{(n)}{P^0_\alpha}(r_{j+1})\right) \\ & + \frac{\Delta r}{6}\left((r_{j+1}-r)m_j + (r-r_j)m_{j+1}\right).\end{aligned}$$

The coefficients m_j can be computed as solution of the linear system of equations

$$\mathbf{Mm} = \mathbf{p}\,, \tag{20}$$

where $m_0 = m_{n_r} = 0$ is set. The right hand side vector is defined by

$$p_k = \frac{6}{(\Delta r)^2}\left(\overset{(n)}{P^0_\alpha}(r_{k-1}) - 2\overset{(n)}{P^0_\alpha}(r_k) + \overset{(n)}{P^0_\alpha}(r_{k+1})\right) \tag{21}$$

and the matrix is

$$M_{k\ell} = \begin{cases} 4 & \text{if } k = \ell \\ 1 & \text{if } k = \ell \pm 1 \\ 0 & \text{else} \end{cases} \tag{22}$$

with indices $k, \ell = 1, ..., n_r - 1$. The interpolation is done for every timestep, for all functions $\overset{(n)}{P^i_\alpha}(r)$ with $i = 1, 2, 3$. Note that the inverse ${M_{k\ell}}^{-1}$ is set up once recursively and reused.

5.3 Geometrical Clustering and Efficient Storage

The proposed formulation is based on the $\mathcal{H}$-matrix concept (cf. e.g. [8, 31, 5]). Within this approach, the nodal degrees of freedom (dofs) are hierarchically grouped to sets (clusters) via a Principal Component Analysis (PCA) resulting in a balanced clustertree. This is achieved via a recursive procedure that starts from a root cluster including all degrees of freedom. Within each step, the dofs are split into two subsets based on geometric information provided by the PCA.

Combinations of two clustertrees in turn define the blockclustertree that serves as a basic structure for the hierarchical subdivision of the $\mathcal{H}$-matrix. The geometric information of the clusters Cl_x and Cl_y along with the local support information allow for a distinction whether a block is a negligible 'zero' block or not. Figure 3 shows a pair of such clusters and its corresponding subblock in the matrix.

The subsequently denoted 'efficient storage scheme' relies on the following steps that are performed prior to any matrix entry evaluation:

1. compute the support radii $r_\alpha(n\Delta t)$ at all timesteps n with predefined accuracy $\varepsilon_{\text{supp}}$

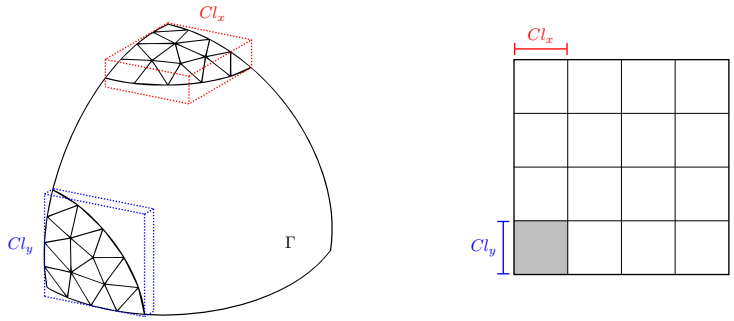


Figure 3: Cluster Cl_x and Cl_y with the corresponding matrix block

2. create a clustertree
3. create a blockclustertree for each timestep n
   - 3.1 for each cluster Cl_x and Cl_y compute the axis aligned bounding box (dotted lines in figure 3) recursively
   - 3.2 for each cluster pairing (Cl_x, Cl_y), compute the minimum and maximum occurring radius, i.e., the smallest possible distance $d_{\min}$ and the largest possible distance $d_{\max}$ of two points residing in the two axis aligned boxes
   - 3.3 for each cluster pairing (Cl_x, Cl_y), determine whether the block is
     - a zero block - $(d_{\min} > r_1(n\Delta t)) \vee (d_{\max} < r_2(n\Delta t))$,
     - a fully populated block - $d_{\max} < r_1(n\Delta t) \wedge r_2(n\Delta t) < d_{\min}$,
     - a partially populated block - else,

     based on the results of section 4 (Note that the recursive algorithm stops if a block is either fully populated or identified to be a zero block.)
4. with the aid of the blockclustertree the hierarchical matrix is computed for each timestep n

Both, the computation of zero-blocks as well as the storage of these blocks in the matrix are avoided with this strategy. The nonzero blocks might be stored densely or in low-rank representation. The effect of a low rank representation is studied in section 6.

Remark 4. Note that the bounding boxes of the clusters have to contain the whole corresponding nodal support.

Remark 5. It should be stated that a partially populated block is refined recursively at most to leaf clusters of size $b_{\min}$. This might result at most in a storage of 'zero' entry blocks of leaf cluster size b_{min}.

5.4 Further Improvements and used Libraries

A possible improvement of the method is the application of low rank matrix compression strategies for fully populated sub-blocks of the hierarchical matrices. Therefore, a singular value decomposition (SVD) is applied on those blocks. Note that the SVD is chosen for simplicity and robustness reasons since the problem sizes are rather small. The second part of the results is devoted to this investigation.

The implementation makes use of HyENA [28], a C++ based BEM-library written at the Institute of Applied Mechanics at Graz University of Technology. The library is capable to treat a large set of partial differential equations including transient problems via the convolution quadrature method.

Additionally, the implementation utilizes the slightly adopted hierarchical matrix library *AHMED* which is a project of Mario Bebendorf [6]. The hierarchical storage concept is used as well as the fast routines that come along with this library (i.e., fast matrix-vector product, iterative solver routines, etc.).

6 Numerical Examples

This section presents two numerical examples. First, a Dirichlet Problem is evaluated to verify the proposed algorithm and, secondly, a mixed problem is analyzed. Both examples are computed for several refinement levels with uniform refinement in space and time. For all problems the relations

$$\beta = \frac{c_1 \Delta t}{r_e} = \frac{1}{3} \quad \text{and} \quad c_1 = 1\,\mathrm{m/s},\ \ c_2 = \sqrt{\frac{1}{2}}\,\mathrm{m/s},\ \ \rho = 1\,\mathrm{kg/m^3} \tag{23}$$

are chosen, where r_e is the average edge length of the triangulation. Furthermore, the wave speeds are chosen according to a vanishing Poisson effect and the minimum leaf cluster size is set to $b_{\min} = 9$.

6.1 Dirichlet Problem

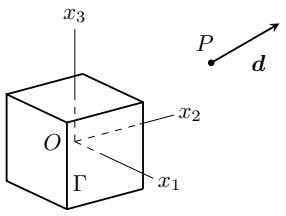


Figure 4: Dirichlet Problem

Domain and Parameters The Dirichlet Problem is computed for a cube of edge length $1\,m$ centered at the origin (figure 4) with parameters as given in (23). Essential prerequisites for this study are analytic functions in space and time that describe the displacements and tractions respectively. While the displacement function $\mathbf{u}(\mathbf{x},t)$ allows to prescribe the Dirichlet datum

(illustrated in figure 5), the traction function $\mathbf{t}(\mathbf{x},t)$ is used for the error computation. Such functions are also known as full-space functions and the definition is given in A.3. Note that $\mathbf{t}^{\ell}(\mathbf{x},t)$ denotes the computed approximation for the ℓ-th level of the mesh refinement.

Memory The computations shown in this section are based on parameters listed in table 1a, where N_t denotes the number of timesteps that are computed, n_e refers to the number of elements of the triangulation and n_t is the relevant number of timesteps (i.e., the number of timesteps until the shear wave has left the domain – the temporal cut-off). Two measures that both make use of this cut-off are defined in bytes:

1. mem_{dns} refers to the amount of memory consumed for dense storage.
2. mem_{eff} utilizes the efficient storage scheme introduced in 5.3.

Storage requirements are shown in table 1b for different levels of mesh refinement where the fourth level yields a reduction of 51%.

Error The absolute and relative space-time error is defined by

$$\text{err}_{abs}^{\ell} = \left(\Delta t \sum_{n=0}^{N_t} ||\mathbf{t}(\mathbf{x}, n\Delta t) - \mathbf{t}^{\ell}(\mathbf{x}, n\Delta t)||_{L_2(\Gamma)}^2 \right)^{1/2}, \tag{24a}$$

$$\text{err}_{rel}^{\ell} = \text{err}_{abs}^{\ell} \left(\Delta t \sum_{n=0}^{N_t} ||\mathbf{t}(\mathbf{x}, n\Delta t)||_{L_2(\Gamma)}^2 \right)^{-1/2} \text{with } \mathbf{x} \in \Gamma. \tag{24b}$$

The order of convergence

$$\text{eoc} = \log_2 \left(\frac{\text{err}_{rel}^{\ell}}{\text{err}_{rel}^{(\ell+1)}} \right) \tag{25}$$

is listed in the tables for several spatial refinement levels ℓ as well. In the proposed formulation two approximations are introduced: the efficient storage and the spline interpolation. The influence of both are studied in table 2 by showing the space-time errors of three different formulations:

1. A first formulation uses none of both approximations, i.e., neither the spline interpolation for the weight evaluation nor an efficient storage is used. This formulation serves as reference and is denoted as *exact, dense* (see table 2a).
2. The second formulation uses the efficient storage but no spline interpolation. The results of this formulation are presented in table 2b and are denoted by *exact, efficient*. They reveal the fact that the negligence of zero blocks works properly and, consequently, no additional error is induced.
3. Finally, the third formulation that makes use of both the spline interpolation as well as the efficient storage scheme. The results in table 2c denoted by *interpolation, efficient* show that the order of convergence is preserved for the chosen number of interpolation points n_r.

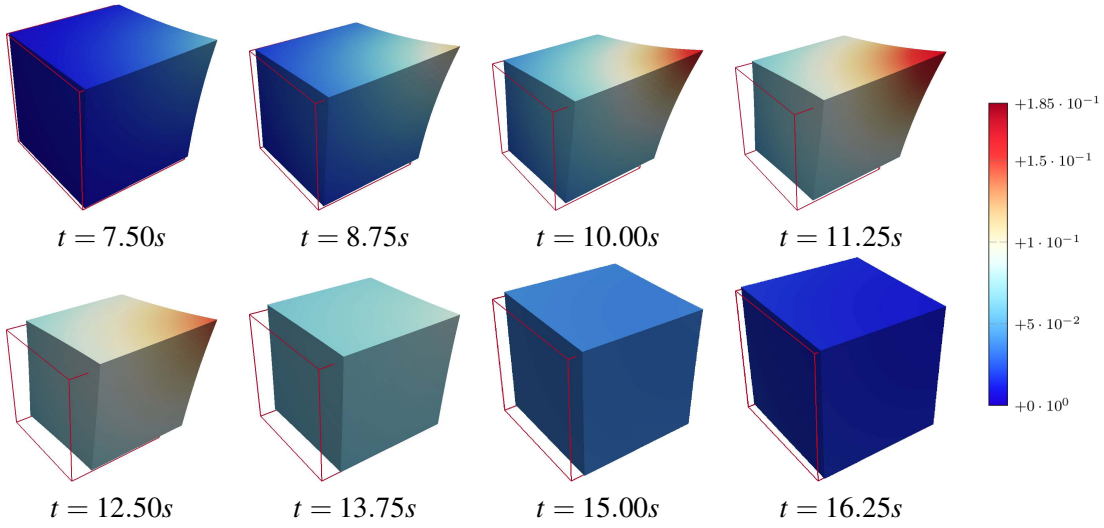


Figure 5: Prescribed displacement boundary conditions $|\mathbf{u}(\mathbf{x},t)|$

A counterexample is given in table 3 where due to an improper choice of interpolation points the convergence cannot be achieved any more. The efficient storage schemes presented in here rely on a support detection accuracy of $\varepsilon_{\text{supp}} = 10^{-3}$.

ℓ	N_t	Δt	n_e
0	100	$1/3$	12
1	200	$1/6$	48
2	400	$1/12$	192
3	800	$1/24$	768
4	1600	$1/48$	3072

(a) Parameters

n_t	mem_{eff}	mem_{dns}	ratio
23	4.21E+05	4.15E+05	1.02
36	9.13E+06	9.46E+06	0.96
57	1.89E+08	2.33E+08	0.81
98	3.97E+09	6.32E+09	0.63
171	8.55E+10	1.75E+11	0.49

(b) Memory

Table 1: Dirichlet Problem

6.2 Singular Value Decomposition Tests

A possible improvement is investigated in this section – a further reduction of memory via low rank approximations of fully populated blocks. The distinction whether a block is fully populated or not is known prior to the computation of its entries and identified within the setup of the blockclustertree. Again, the Dirichlet Problem is investigated in terms of low rank compression of fully populated blocks. Table 4 shows the results $\text{mem}_{eff,SVD}$ for the computed levels and different SVD accuracies ε_{SVD}. The compression is slightly improved in the fourth level with $\varepsilon_{SVD} = 10^{-3}$ – a reduction of about 6 percent – still, the order of convergence is preserved. Larger problems however might require a higher accuracy of the SVD to maintain the order of

ℓ	err_{abs}	err_{rel}	eoc
0	1.02E-1	5.43E-1	-
1	5.31E-2	2.83E-1	0.94
2	2.58E-2	1.38E-1	1.04
3	1.25E-2	6.67E-2	1.05
4	6.00E-3	3.20E-2	1.06

(a) Exact, dense

err_{abs}	err_{rel}	eoc
1.02E-1	5.46E-1	-
5.36E-2	2.86E-1	0.93
2.61E-2	1.39E-1	1.04
1.25E-2	6.67E-2	1.06
6.00E-3	3.20E-2	1.06

(b) Exact, efficient

n_r	err_{abs}	err_{rel}	eoc
120	1.02E-1	5.43E-1	-
240	5.31E-2	2.83E-1	0.94
480	2.58E-2	1.38E-1	1.04
960	1.24E-2	6.61E-2	1.06
1920	6.10E-3	3.25E-2	1.02

(c) Interpolation, efficient

Table 2: Convergence rates for different formulations

ℓ	err_{abs}	err_{rel}	eoc
0	1.02E-1	5.43E-1	-
1	5.31E-2	2.83E-1	0.94
2	2.58E-2	1.38E-1	1.04
3	1.25E-2	6.67E-2	1.05
4	6.00E-3	3.20E-2	1.06

(a) Exact, dense

n_r	err_{abs}	err_{rel}	eoc
10	1.32E-1	7.02E-1	-
20	7.60E-2	4.05E-1	0.79
40	4.50E-2	2.40E-1	0.76
80	2.89E-2	1.54E-1	0.64
160	2.01E-2	1.07E-1	0.52

(b) Interpolation, efficient

Table 3: Counterexample for improperly chosen number of interpolation points n_r

convergence since the number of approximated blocks increases significantly. For the presented problem sizes, a compression does not pay off but when it comes to larger problems, methods like the ACA (see [5]) might be advantageous.

ℓ	$\text{mem}_{eff,SVD}$	ratio	eoc
0	4.21E+05	1.02	-
1	9.13E+06	0.96	0.94
2	1.89E+08	0.81	1.04
3	3.97E+09	0.63	1.06
4	8.53E+10	0.49	1.02

(a) $\varepsilon_{SVD} = 10^{-9}$

$\text{mem}_{eff,SVD}$	ratio	eoc
4.21E+05	1.02	-
9.13E+06	0.96	0.94
1.89E+08	0.81	1.04
3.97E+09	0.63	1.06
8.47E+10	0.48	1.02

(b) $\varepsilon_{SVD} = 10^{-6}$

$\text{mem}_{eff,SVD}$	ratio	eoc
4.21E+05	1.02	-
9.13E+06	0.96	0.94
1.89E+08	0.81	1.04
3.91E+09	0.62	1.06
7.62E+010	0.43	1.02

(c) $\varepsilon_{SVD} = 10^{-3}$

Table 4: Memory and convergence results for SVD compression

6.3 Mixed Problem

Domain and Parameters A rod of dimensions $1\,m \times 3\,m \times 1\,m$ (figure 6) is investigated with the same parameters as for the Dirichlet problem (23). At $x_2 = 0\,m$ the displacements are zero and at $x_2 = 3\,m$, the surface is loaded by $t_2 = 1H(t)\ {}^{N}/_{m^2}$ with $H(t)$ being the Heaviside function. On the remaining boundary the tractions are set to zero. For this kind of setting the one-dimensional analytic solutions for the displacements $u_2(x,t)$ and tractions $t_2(x,t)$ are known (see [16] for

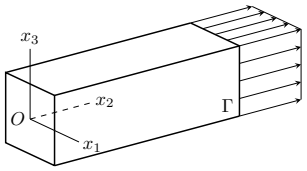


Figure 6: Mixed Problem

instance). To compare the numerical results with the analytic solutions a vanishing Poisson's ratio $\nu = 0$ is required.

Memory Like in the previous section, the memory consumption is listed for different levels in table 5b (parameters are chosen according to table 5a). The memory consumption is significantly reduced by 56% in the third level with a support detection accuracy of $\varepsilon_{\text{supp}} = 10^{-3}$. The number of interpolation points n_r is chosen according to the results of subsection 6.1. A further illustration is given in figure 11a where the semilogarithmic representation and the different incline of the two curves have to be taken into account.

Remark 6. Note that due to the results of subsection 6.2 the numerical examples of this subsection do not make use of low rank approximation techniques. Solely the efficient storage scheme developed in subsection 5.3 is used.

Error The difference between the analytical displacement solution and the center point at the loaded end ($x_2 = 3\,m$) is measured as well as the difference between the analytical tractions and the center point of the clamped end ($x_2 = 0\,m$)

$$e_u^\ell(n\Delta t) = u_2^\ell(3,0,0,n\Delta t) - u_2(3,n\Delta t) \tag{26a}$$

$$e_t^\ell(n\Delta t) = t_2^\ell(0,0,0,n\Delta t) - t_2(0,n\Delta t)\,. \tag{26b}$$

The qualitative results for the displacements and tractions are shown in figures 7, 8, 9 and 10 as well as their corresponding errors. The expected behavior, an error reduction between the levels becomes visible for the displacements. Regarding the tractions it must be said that the approximation of the discontinuity certainly causes troubles – nevertheless, apart from the overshoot at the rising and falling edges, the overall result is better for the third level.

Timings The speedup of the formulation is compared with the standard CQM routine implemented in HyENA (see [34] for details). Integration orders for singular and regular integration are kept equal for both formulations and the matrices are assembled up to the number of relevant timesteps of the efficient formulation (i.e., the same cut-off is used). $t_{ass,eff}$ denotes the assembling time with the proposed algorithm, likewise the timings for the HyENA standard CQM formulation are denoted $t_{ass,H}$ (both quantities are measured in seconds). It can be seen in table 6 that the efficient formulation – already for the 3rd level – takes only half the computational time for the matrix assemblage compared to the HyENA standard formulation. A further illustration of the assembly timings is given in figure 11b where it becomes obvious that the break even point is reached in level 1.

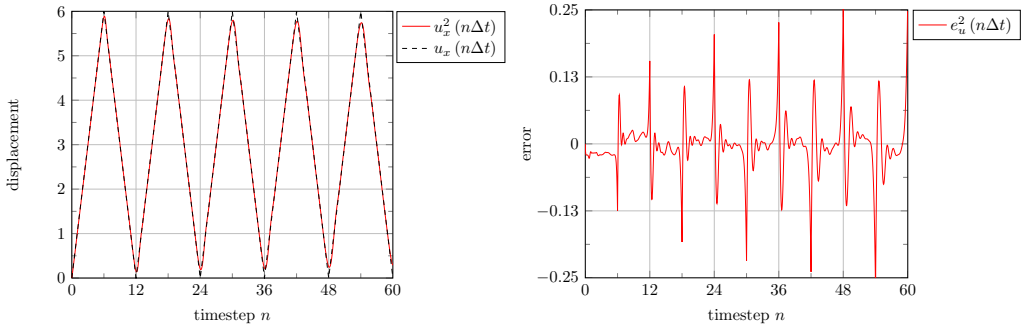
ℓ	N_t	Δt	n_e
0	360	$1/6$	112
1	720	$1/12$	448
2	1440	$1/24$	1792
3	2880	$1/48$	7168

(a) Parameters

n_t	mem_{eff}	mem_{dns}	ratio
55	1.15E+07	1.37E+07	0.84
94	2.75E+08	3.88E+08	0.71
160	6.36E+09	1.08E+10	0.59
296	1.41E+11	3.24E+11	0.44

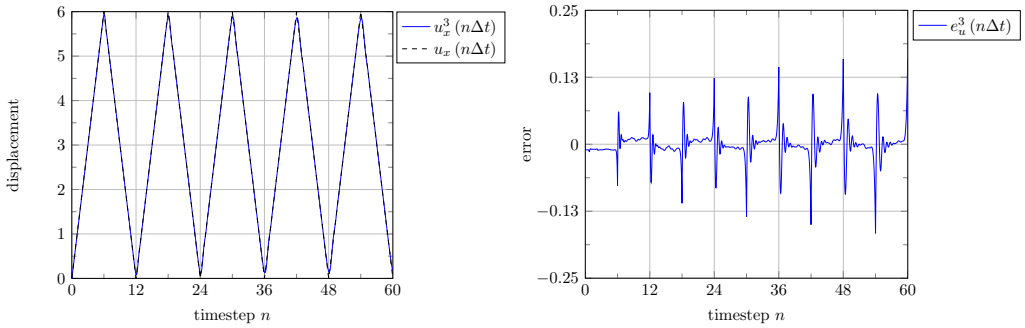
(b) Memory

Table 5: Mixed Problem



(a) Displacements at the free end (b) Error

Figure 7: Displacement results for refinement level 2

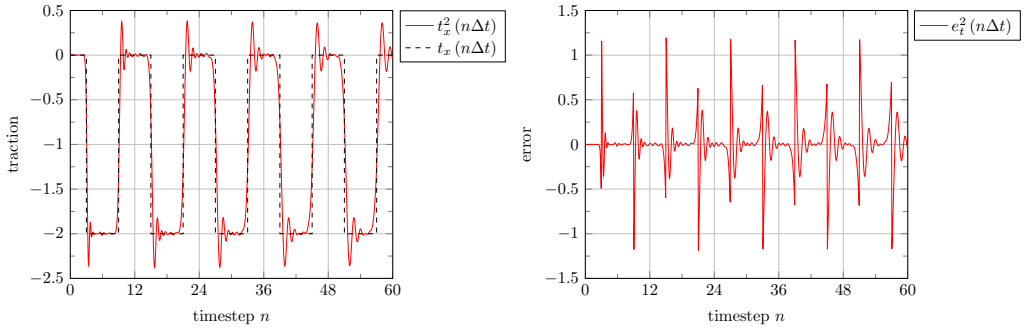


(a) Displacements at the free end (b) Error

Figure 8: Displacement results for refinement level 3

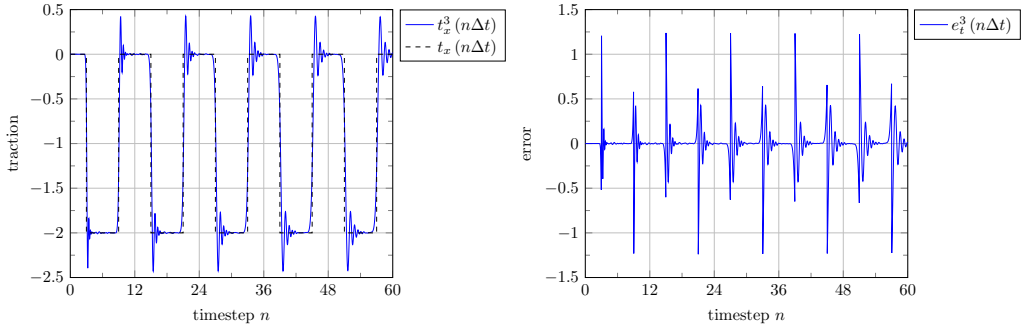
ℓ	n_e	n_t	$t_{ass,eff}$	$t_{ass,H}$	ratio
0	112	55	10	8	1.23
1	448	94	137	158	0.87
2	1792	160	1329	2571	0.52

Table 6: Assembly times for efficient and standard formulation



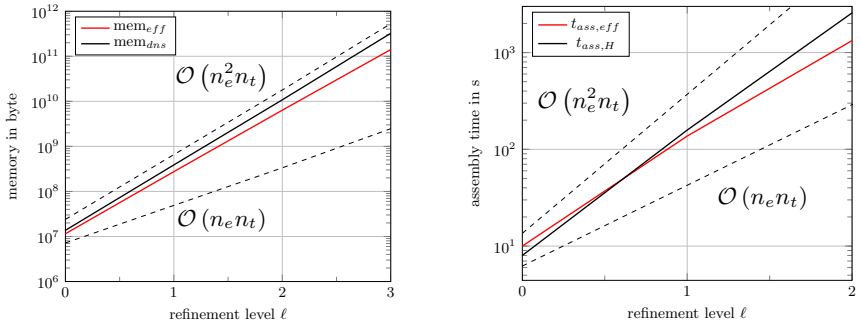
(a) Tractions at the clamped end (b) Error

Figure 9: Traction results for refinement level 2



(a) Tractions at the clamped end (b) Error

Figure 10: Traction results for refinement level 3



(a) Memory consumption (b) Assembly timings

Figure 11: Plot of memory consumption and assembly timings

7 Conclusion

This paper has established a boundary element formulation for the treatment of time-dependent problems in linear elastodynamics with the aid of the convolution quadrature method. The approach of direct weight computation was successfully extended to the elastodynamic problem and the local support information was utilized to avoid the computation and storage of 'zero'-blocks in the matrix assemblage. This could be achieved by detecting such blocks prior to the matrix computation. The strategy of direct evaluation avoids the use of Cauchy's integral formula, thus the approximation of this integral becomes obsolete. However, from a computational point of view, the direct weight evaluation is costly and was thus sped up via an interpolation scheme, hence, another approximation was introduced. The results of the Dirichlet problem though show that if the interpolation parameters are chosen appropriately the order of convergence is preserved.

A second example with mixed boundary expressions was presented as well and significant improvements in terms of memory consumption and assembling times have been achieved.

Acknowledgement The authors gratefully acknowledge the financial support by the Austrian Science Fund (FWF) under Grant P-22510.

A

A.1 Derivation of the Recurrence Relation for $\overset{(n)}{P^0_\alpha}(r)$

As a starting point serves equation (14)

$$\overset{(n)}{P^0_\alpha}(r) = \frac{1}{n!}\left(\frac{r}{2c_\alpha\Delta t}\right)^{\frac{n}{2}} \exp\left(-\frac{3r}{2c_\alpha\Delta t}\right) H_n\left(\sqrt{\frac{2r}{c_\alpha\Delta t}}\right) .$$

The increment $n+1$ in combination with the recursive definition of the Hermite polynomial [1] yields

$$\begin{aligned}
\overset{(n+1)}{P^0_\alpha}(r) &= \frac{1}{(n+1)!}\left(\frac{r}{2c_\alpha\Delta t}\right)^{\frac{n+1}{2}} \exp\left(-\frac{3r}{2c_\alpha\Delta t}\right) H_{n+1}\left(\sqrt{\frac{2r}{c_\alpha\Delta t}}\right) \\
&= \frac{1}{(n+1)}\frac{1}{n!}\sqrt{\frac{r}{2c_\alpha\Delta t}}\left(\frac{r}{2c_\alpha\Delta t}\right)^{\frac{n}{2}} \exp\left(-\frac{3r}{2c_\alpha\Delta t}\right) H_{n+1}\left(\sqrt{\frac{2r}{c_\alpha\Delta t}}\right) \\
&= \frac{1}{(n+1)}\frac{2r}{c_\alpha\Delta t}\underbrace{\frac{1}{n!}\left(\frac{r}{2c_\alpha\Delta t}\right)^{\frac{n}{2}} \exp\left(-\frac{3r}{2c_\alpha\Delta t}\right) H_n\left(\sqrt{\frac{2r}{c_\alpha\Delta t}}\right)}_{\overset{(n)}{P^0_\alpha}(r)}
\end{aligned}$$

$$
\begin{aligned}
&- \frac{1}{(n+1)} \frac{r}{c_\alpha \Delta t} \underbrace{\frac{1}{(n-1)!} \left(\frac{r}{2 c_\alpha \Delta t} \right)^{\frac{n-1}{2}} \exp\left(-\frac{3r}{2 c_\alpha \Delta t} \right) H_{n-1}\left(\sqrt{\frac{2r}{c_\alpha \Delta t}} \right)}_{\overset{(n-1)}{P^0_\alpha}(r)} \\
&= \frac{1}{(n+1)} \frac{r}{c_\alpha \Delta t} \left(\overset{(n)}{2P^0_\alpha}(r) - \overset{(n-1)}{P^0_\alpha}(r) \right) .
\end{aligned}
$$

Deriving the recurrences for the expressions $\overset{(1)}{P^0_\alpha}(r)$ and $\overset{(2)}{P^0_\alpha}(r)$ follows the same strategy but is rather cumbersome. For the sake of readability it is therefore omitted.

A.2 Traction Fundamental Solution

The traction fundamental solution is defined by

$$
\hat{T}_{ij}(\mathbf{r}, s) = \mathcal{T}_{jk}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y})) \hat{U}_{ki}(\mathbf{r}, s) \ , \tag{27}
$$

where $\mathcal{T}_{ij}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y}))$ is the traction operator (i.e., Hooke's law) and $\mathbf{n}(\mathbf{y})$ denotes the outward normal vector located at point $\mathbf{y}$ and $i, j, k = 1, 2, 3$. Contrary to the standard notation, a representation that involves the *Günter derivatives* [17] is given by

$$
\mathcal{T}_{ij}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y})) = 2\mu \mathcal{M}_{ij}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y})) + (\lambda + 2\mu) n_i(\mathbf{y}) \frac{\partial}{\partial y_j} - \mu n_j(\mathbf{y}) \frac{\partial}{\partial y_i} + \mu \delta_{ij} \frac{\partial}{\partial \mathbf{n}(\mathbf{y})} \tag{28}
$$

with

$$
\mathcal{M}_{ij}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y})) = n_j(\mathbf{y}) \frac{\partial}{\partial y_i} - n_i(\mathbf{y}) \frac{\partial}{\partial y_j} \text{ and } \mathbf{y} \in \Gamma \, . \tag{29}
$$

Thus, the traction fundamental solution for elastodynamics results in

$$
\begin{aligned}
\hat{T}_{ij}(\mathbf{r}, s) &= 2\mu \mathcal{M}_{jk}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y})) \left(\hat{U}_{ki}(\mathbf{r}, s) - \frac{1}{4\pi} \delta_{ki} \frac{1}{r} \exp\left(\frac{-sr}{c_1} \right) \right) \\
&+ \frac{1}{4\pi} n_i(\mathbf{y}) \frac{\partial}{\partial y_j} \left(\frac{1}{r} \exp\left(\frac{-sr}{c_1} \right) - \frac{1}{r} \exp\left(\frac{-sr}{c_2} \right) \right) \\
&+ \frac{1}{4\pi} \delta_{ij} \frac{\partial}{\partial \mathbf{n}(\mathbf{x})} \left(\frac{1}{r} \exp\left(\frac{-sr}{c_2} \right) \right) \qquad (30) \\
&= 2\mu \mathcal{M}_{jk}(\partial_{\mathbf{y}}, \mathbf{n}(\mathbf{y})) \left(\hat{U}_{ki}(\mathbf{r}, s) - \frac{1}{4\pi} \delta_{ki} \frac{1}{r} \exp\left(\frac{-sr}{c_1} \right) \right) \\
&+ \frac{1}{4\pi} \frac{1}{r^2} \frac{n_i(\mathbf{y}) r_j}{r} \left(- \left(1 + \frac{sr}{c_1} \right) \exp\left(\frac{-sr}{c_1} \right) + \left(1 + \frac{sr}{c_2} \right) \exp\left(\frac{-sr}{c_2} \right) \right) \\
&+ \frac{1}{4\pi} \frac{1}{r^2} \frac{n_k(\mathbf{y}) r_k}{r} \delta_{ij} \left(-1 - \frac{sr}{c_2} \right) \exp\left(\frac{-sr}{c_2} \right) . \qquad (31)
\end{aligned}
$$

This representation is suitable to perform an integration by parts technique in the sequel. The theoretical framework can be found in [37] and applications in elastodynamics in [19, 15, 20]. Note that the last two terms in equation (30) exhibit a weakly singular behavior if r tends to zero.

Similar to the approach in section 4, the approximation of the convolution involving the traction fundamental solution is defined by

$$\int_\Gamma T_{ij}(\mathbf{r},t) * u_j(\mathbf{y},t)\,\mathrm{d}s_\mathbf{y} \approx \sum_{m=0}^{n} \int_\Gamma \theta_{ij}^{n-m}(\mathbf{r},\mathbf{n}(\mathbf{y}))\,u_j(\mathbf{y},m\Delta t)\,\mathrm{d}s_\mathbf{y}\,. \tag{32}$$

Applying the same strategies, the weights $\theta_{ij}^n(\mathbf{r},\mathbf{n}(\mathbf{y}))$ are obtained

$$\begin{aligned}\theta_{ij}^n(\mathbf{r},\mathbf{n}(\mathbf{y})) = 2\mu\mathcal{M}_{jk}(\partial_\mathbf{y},\mathbf{n}(\mathbf{y}))\left(\omega_{ki}^n(\mathbf{r}) - \frac{1}{4\pi}\frac{1}{r}\delta_{ki}\overset{(n)}{P}{}_1^0(r)\right)\\ + \frac{1}{4\pi}\frac{1}{r^2}\frac{n_i(\mathbf{y})\,r_j}{r}\left(-\overset{(n)}{P}{}_1^0(r) - \frac{r}{c_1}\overset{(n)}{P}{}_1^{-1}(r) + \overset{(n)}{P}{}_2^0(r) + \frac{r}{c_2}\overset{(n)}{P}{}_2^{-1}(r)\right)\\ + \frac{1}{4\pi}\frac{1}{r^2}\frac{n_k(\mathbf{y})\,r_k}{r}\delta_{ij}\left(-\overset{(n)}{P}{}_2^0(r) - \frac{r}{c_2}\overset{(n)}{P}{}_2^{-1}(r)\right)\,.\end{aligned} \tag{33}$$

Provided that $u_i(\mathbf{y},n\Delta t)$ is differentiable with respect to $\mathbf{y}$ and Γ is closed, a weakly singular representation of the double layer potential can be established (i.e., the *Günter derivatives* are shifted towards the displacements via a partial integration)

$$\int_\Gamma \theta_{ij}^{n-m}(\mathbf{r},\mathbf{n}(\mathbf{y}))\,u_j(\mathbf{y},m\Delta t)\,\mathrm{d}s_\mathbf{y} \tag{34}$$

$$\begin{aligned}&= \int_\Gamma 2\mu\left(\omega_{ki}^{n-m}(\mathbf{r}) - \frac{1}{4\pi}\frac{1}{r}\delta_{ki}\overset{(n-m)}{P}{}_1^0(r)\right)\mathcal{M}_{kj}(\partial_\mathbf{y},\mathbf{n}(\mathbf{y}))\,u_j(\mathbf{y},m\Delta t)\,\mathrm{d}s_\mathbf{y}\\ &+ \int_\Gamma \frac{1}{4\pi}\frac{1}{r^2}\frac{n_i(\mathbf{y})\,r_j}{r}\left(-\overset{(n-m)}{P}{}_1^0(r) - \frac{r}{c_1}\overset{(n-m)}{P}{}_1^{-1}(r)\right)u_j(\mathbf{y},m\Delta t)\,\mathrm{d}s_\mathbf{y}\\ &+ \int_\Gamma \frac{1}{4\pi}\frac{1}{r^2}\frac{n_i(\mathbf{y})\,r_j}{r}\left(+\overset{(n-m)}{P}{}_2^0(r) + \frac{r}{c_2}\overset{(n-m)}{P}{}_2^{-1}(r)\right)u_j(\mathbf{y},m\Delta t)\,\mathrm{d}s_\mathbf{y}\\ &+ \int_\Gamma \frac{1}{4\pi}\frac{1}{r^2}\frac{n_k(\mathbf{y})\,r_k}{r}\delta_{ij}\left(-\overset{(n-m)}{P}{}_2^0(r) - \frac{r}{c_2}\overset{(n-m)}{P}{}_2^{-1}(r)\right)u_j(\mathbf{y},m\Delta t)\,\mathrm{d}s_\mathbf{y}\,.\end{aligned} \tag{35}$$

The additional expression $\overset{(n)}{P}{}_\alpha^{-1}(r)$ can be computed

$$\overset{(n)}{P}{}_\alpha^{-1}(r) = \frac{1}{\Delta t}\sum_{i=0}^{2} F_i\,\overset{(n-i)}{P}{}_\alpha^0(r) \text{ with} \tag{36}$$

$$F_0 = \frac{3}{2}\,,\ F_1 = -2\,,\ F_2 = \frac{1}{2} \text{ and } \overset{(-2)}{P}{}_\alpha^0(r) = \overset{(-1)}{P}{}_\alpha^0(r) = 0\,.$$

A.3 Full-Space Solutions $\mathbf{u}(\mathbf{x},t)$ and $\mathbf{t}(\mathbf{x},t)$

Two basic singular solutions $u_{ij}(\mathbf{x},t,\boldsymbol{\xi}|f)$ and $t_{ijk}(\mathbf{x},t,\boldsymbol{\xi}|f)$ with $i,j,k=1,2,3$, provided in [14], serve as a starting point for the construction of $\mathbf{u}(\mathbf{x},t)$ and $\mathbf{t}(\mathbf{x},t)$. According to the notation found in [14], $\boldsymbol{\xi}$ denotes the source point and $\mathbf{x}$ the observation point while $f(t)$ is a twice continuously differentiable function with respect to t.

The solutions given in [14] are formulated for a generic source function f in time. The numerical examples presented in section 6.1 use

$$f(t) = \exp\left(-a(t-ab)^2\right) \tag{37}$$

with constant parameters a and b. To evaluate $u_{ij}(\mathbf{x},t,\boldsymbol{\xi}|f)$ and $t_{ijk}(\mathbf{x},t,\boldsymbol{\xi}|f)$ the function itself as well as the temporal derivatives are required. Moreover, an integration is required that is listed for the sake of completeness

$$\begin{aligned}\int_{1/c_1}^{1/c_2} \beta f(t-r\beta)\,\mathrm{d}\beta = \frac{1}{2ar^2}\Big(&\exp\left(-q_1^2(r)\right) - \exp\left(-q_2^2(r)\right) \\ &+ \sqrt{a\pi}(ab-t)\left(\operatorname{erf}(q_1(r)) - \operatorname{erf}(q_2(r))\right)\Big)\,,\end{aligned} \tag{38}$$

where $\operatorname{erf}(x)$ denotes the Error Function

$$\begin{aligned}\operatorname{erf}(x) &= \frac{2}{\sqrt{\pi}}\int_0^x \exp\left(-t^2\right)\mathrm{d}t \quad \text{and} \\ q_\alpha(r) &= \frac{\sqrt{a}}{c_\alpha}\left(abc_\alpha + r - c_\alpha t\right)\,.\end{aligned} \tag{39}$$

Having $u_{ij}(\mathbf{x},t,\boldsymbol{\xi}|f)$ and $t_{ijk}(\mathbf{x},t,\boldsymbol{\xi}|f)$ at hand, the analytic functions are constructed with the aid of a direction vector $\mathbf{d} = (1,1,1)^T$ and the outward normal $\mathbf{n}(\boldsymbol{\xi})$ by

$$u_i(\mathbf{x},t) = u_{ij}(\mathbf{x},t,\boldsymbol{\xi}|f)\,d_j \text{ and } t_i(\mathbf{x},t) = t_{ijk}(\mathbf{x},t,\boldsymbol{\xi}|f)\,n_j(\mathbf{x})\,d_k\,, \tag{40}$$

where finally all directions are aggregated to get $\mathbf{u}(\mathbf{x},t)$ and $\mathbf{t}(\mathbf{x},t)$. Note that for the presented example, the parameters are set to $a = 0.1$ and $b = 100$, the chosen source point is $P(1,1,1)$ and $\boldsymbol{\xi} = \overline{OP}$ (vector $\mathbf{d}$ and point P are illustrated in figure 4).

Remark 7. The source point P has to be chosen such that the homogeneous boundary conditions are met (i.e., far enough from the domain).

References

[1] Milton Abramowitz and Irene Stegun. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. Dover Publications Inc., 1972.

[2] Jan D. Achenbach. *Wave Propagation in Elastic Solids*. Applied Mathematics and Mechanics. North-Holland Publishing Company - Amsterdam, London, 1973.

[3] Heinz Antes. *Anwendung der Methode der Randelemente in der Elastodynamik und Fluiddynamik*, volume 9 of *Mathematische Methoden in der Technik*. B.G. Teubner Stuttgart, 1988.

[4] Lehel Banjai and Martin Schanz. Wave propagation problems treated with convolution quadrature and bem. In *Fast Boundary Element Methods in Engineering and Industrial Applications*, volume 63 of *Lecture Notes in Applied and Computational Mechanics*, pages 145–184. Springer Berlin Heidelberg, 2012.

[5] Mario Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering (LNCSE)*. Springer-Verlag, 2008.

[6] Mario Bebendorf. Another software library on hierarchical matrices for elliptic differential equations (AHMED). `http://bebendorf.ins.uni-bonn.de/AHMED.html`, 2014.

[7] Björn Birgisson, Eduard Siebrits, and Anthony P. Peirce. Elastodynamic direct boundary element methods with enhanced numerical stability properties. *International Journal for Numerical Methods in Engineering*, 46:871–888, 1999.

[8] Steffen Börm, Lars Graseyk, and Wolfgang Hackbusch. Hierarchical matrices. Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig, 2006. Lecture note no. 21.

[9] Thomas A. Cruse and Frank J. Rizzo. A Direct Formulation and Numerical Solution of the General Transient Elastodynamic Problem. I. *Journal of Mathematical Analysis and Applications*, 22:244–259, 1968.

[10] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer Berlin Heidelberg, 2006.

[11] Jose Dominguez. *Boundary Elements in Dynamics*. Computational Mechanics Publications, Southampton, Boston, 1993.

[12] Michael Duffy. Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM Journal on Numerical Analysis*, 19(6):1260–1262, 1982.

[13] Arif A. Ergin, Balasubramaniam Shanker, and Eric Michielssen. Fast evaluation of three-dimensional transient wave fields using diagonal translation operators. *Journal of Computational Physics*, 146(1):157 – 180, 1998.

[14] Ahmed C. Eringen and Erdogan S. Suhubi. *Elastodynamics*, volume 2. Academic Press, 1975.

[15] Attilio Frangi. Elastodynamics by BEM : A new direct Formulation. *International Journal for Numerical Methods in Engineering*, 45:721–740, 1999.

[16] Karl F. Graff. *Wave Motion in Elastic Solids*. Dover Publications, 1991.

[17] Nikolai M. Günter. *Potential theory, and its applications to basic problems of mathematical physics*. Frederick Ungar Publishing, New York, 1967.

[18] Wolfang Hackbusch, Wendy Kress, and Stefan A. Sauter. Sparse Convolution Quadrature for Time Domain Boundary Integral Formulations of the Wave Equation by Cutoff and Panel-Clustering. In *Boundary Element Analysis: Mathematical Aspects and Applications*, volume 29, pages 113–134. Springer Berlin Heidelberg, 2007.

[19] Houde Han. The boundary integro-differential equations of three-dimensional Neumann problem in linear elasticity. *Numerische Mathematik*, 68(2):269–281, 1994.

[20] Lars Kielhorn. *A Time-Domain Symmetric Galerkin BEM for Viscoelastodynamics*. Monographic Series TU Graz, Graz, 2009.

[21] Victor D. Kupradze and T. V. Burchuladze. The dynamical Problems of the Theory of Elasticity and Thermoelasticity. *Journal of Soviet Mathematics*, 7(3):415–500, 1977.

[22] Christian Lubich. Convolution Quadrature and Discretized Operational Calculus. I. *Numerische Mathematik*, 52:129–145, 1988.

[23] Christian Lubich. Convolution Quadrature and Discretized Operational Calculus. II. *Numerische Mathematik*, 52:413–425, 1988.

[24] Christian Lubich and Reinhold Schneider. Time discretization of parabolic boundary integral equations. *Numerische Mathematik*, 63:455–481, 1992.

[25] Webe J. Mansur. *A Time Stepping Technique to Solve Wave Propagation Problems Using the Boundary Element Method*. PhD thesis, University of Southampton, 1993.

[26] Webe J. Mansur and Carlos A. Brebbia. Transient Elastodynamics. In C.A. Brebbia, editor, *Time-dependent and Vibration Problems*, chapter 5, pages 124–155. Springer Berlin Heidelberg, 1985.

[27] Vladislav Mantic. A new formula for the C-matrix in the Somigliana identity. *Journal of Elasticity*, 33:191–201, 1992.

[28] Ma. Messner, Mi. Messner, F. Rammerstorfer, and P. Urthaler. Hyperbolic and elliptic numerical analysis BEM library (HyENA). `http://www.mech.tugraz.at/HyENA`, 2010. [Online; accessed 22-January-2010].

[29] Matthias Messner and Martin Schanz. An accelerated symmetric time-domain boundary element formulation for elasticity. *Engineering Analysis with Boundary Elements*, 34(11): 944–955, November 2010. ISSN 09557997.

[30] Giovanni Monegato, Letizia Scuderi, and Marija P. Stanić. Lubich convolution quadratures and their application to problems described by space-time BIEs. *Numerical Algorithms*, 56(3):405–436, 2010.

[31] Sergej Rjasanow and Olaf Steinbach. *The Fast Solution of Boundary Integral Equations*. Springer, 2006.

[32] Martin Schanz. *Wave Propagation in Viscoelastic and Poroelastic Continua: A Boundary Element Approach*, volume 2 of *Lecture Notes in Applied Mechanics*. Springer-Verlag Berlin Heidelberg, 2001.

[33] Martin Schanz and Heinz Antes. A new visco- and elastodynamic time domain Boundary Element formulation. *Computational Mechanics*, 20:452–459, 1997.

[34] Martin Schanz and Heinz Antes. Application of Operational Quadrature Methods in time domain boundary element methods. *Meccanica*, 32:179–186, 1997.

[35] Martin Schanz, Heinz Antes, and Thomas Rüberg. Convolution quadrature boundary element method for quasi-static visco- and poroelastic continua. *Computers & Structures*, 83 (10-11):673–684, 2005.

[36] Eduardo Siebrits and Anthony P. Peirce. Implementation and application of elastodynamic boundary element discretizations with improved stability properties. *Engineering Computations*, 14:669–695, 1997.

[37] Olaf Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems*. Springer New York, 2008.

[38] Toru Takahashi. A wideband fast multipole accelerated boundary integral equation method for time-harmonic elastodynamics in two dimensions. *International Journal for Numerical Methods in Engineering*, 91(5):531–551, 2012.

[39] Toru Takahashi, Naoshi Nishimura, and S. Kobayashi. A fast {BIEM} for three-dimensional elastodynamics in time domain. *Engineering Analysis with Boundary Elements*, 27(5):491 – 506, 2003.