forum acusticum 2023

AN OPEN-SOURCE BOUNDARY ELEMENT FRAMEWORK FOR LARGE-SCALE VISCOTHERMAL ACOUSTICS

Mikkel Paltorp[1,3,*] **Vicente Cutanda Henriquez**[2,3]
[1] Acoustic Technology Group (ACT), 2800 Kgs. Lyngby
[2] Centre for Acoustic-Mechanical Microsystems (CAMM), 2800 Kgs. Lyngby
[3] Department of Electrical and Photonics Engineering, The Technical University of Denmark

ABSTRACT

BoundaryIntegralEquations.jl is an open source software library aimed at solving the Kirchhoff-Helmholtz Integral Equation using the collocation Boundary Element Method (BEM). The software is written in the Julia programming language, making it both easy to use and maintain while also being computationally efficient. The package builds upon the ideas of the OpenBEM software, but adds on additional functionality such as the Fast Multipole Method (FMM) and sparse assembly of all matrices used in the Kirchhoff Decomposition (KD) description of viscous and thermal losses. As such the package takes the first steps towards large-scale BEM computations including viscous and thermal losses. The package is validated using simple geometries, such as cuboids and spheres, where an analytical solution exist.

Keywords: boundary element method, viscothermal losses, open-source, fast multipole method

1. INTRODUCTION

The Helmholtz equation is often met when solving time-harmonic acoustical problems. The analytical solution to this Partial Differential Equation (PDE) is limited to simple geometries such as e.g. spheres, cuboids and cylinders. As the real geometry of the world much more complex than this the solution to the PDE is more often than not approximated using a numerical scheme such as the Finite Element Method (FEM) or the Boundary Element Method (BEM).

This paper focuses on the latter method and introduces the open-source library `BoundaryIntegeralEquations.jl` aimed at solving the Helmholtz using the collocation Boundary Element Method [1]. The design of the package is inspired by the OpenBEM package written in MatLab [2]. The package itself is implemented in the Julia programming language as it was found to be good compromise of efficiency and maintainability [3]. Additionally the Julia language is free and open-source making it accessible to a larger audience than the OpenBEM package.

2. THEORY

The basic idea of the Boundary Element Method is to transform the Helmholtz equation into the Kirchoff-Helmholtz integral equation. The weak form of this integral equation is

$$\int_\Gamma \phi(\mathbf{x}) \left(\alpha\zeta(\mathbf{x})p(\mathbf{x}) - \int_\Gamma \frac{\partial G(\alpha,\mathbf{x},\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})} p(\mathbf{y})\, \mathrm{d}S_\mathbf{y} - \alpha \mathrm{i}\rho c k \int_\Gamma G(\alpha,\mathbf{x},\mathbf{y}) v_\mathbf{n}(\mathbf{y})\, \mathrm{d}S_\mathbf{y} \right) \mathrm{d}S_\mathbf{x} = 0, \mathbf{x} \in \Omega, \tag{1}$$

where i is the imaginary unit, k is the wavenumber, c is the speed of sound of the medium, ρ is the medium density, Ω is the domain of interest, $\Gamma = \partial\Omega$ is the boundary of the domain, $\zeta(\mathbf{x})$ is the integral free term at point $\mathbf{x}$, $\mathbf{n}(\mathbf{y})$ is the normal vector at point $\mathbf{y}$, $p(\mathbf{y})$ is the pressure at point $\mathbf{y}$, $v_\mathbf{n}(\mathbf{y})$ is the normal velocity at point $\mathbf{y}$, $\phi(\mathbf{x})$ is a so-called test function and $G(\alpha,\mathbf{x},\mathbf{y})$ is the Green's function

*__Corresponding author__: mpasc@dtu.dk.

defined as

$$G(\alpha, \mathbf{x}, \mathbf{y}) = \frac{\exp\left(-\alpha \mathrm{i} k \|\mathbf{x}-\mathbf{y}\|_2\right)}{4\pi\|\mathbf{x}-\mathbf{y}\|_2}, \tag{2}$$

with α being the sign of the chosen time dependency [4]. In the case of collocation BEM the test function is chosen as the sum of Dirac delta functions, i.e. that

$$\phi(\mathbf{x}) = \mathbf{a}^\top \begin{bmatrix} \delta(\mathbf{x}-\mathbf{t}_1) \\ \vdots \\ \delta(\mathbf{x}-\mathbf{t}_n) \end{bmatrix}, \tag{3}$$

where $\mathbf{a} \in \mathbb{C}^n$ is a vector of arbitrary coefficients and $\mathbf{t}_i$ is the i-th collocation point, with the set of collocation points being the collection of all interpolation nodes from the elements describing the geometry (see Figure 1 and Figure 2). The discretization of (1) is a two step process: First a discretization of the boundary into so-called elements and secondly by approximating the pressure and the velocity on each element by simple functions. Part of the first step is to represent the coordinates of element e as

$$\mathbf{x}^e(\mathbf{u}) = \mathbf{X}^e\mathbf{N}^e(\mathbf{u}) \in \Gamma^e, \ \forall \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in \mathcal{L}^e \subset \mathbb{R}^2, \tag{4}$$

where $\mathbf{X}^e$ is a matrix with columns equal to the interpolation nodes of the geometry, $\mathbf{N}^e(\mathbf{u})$ are the shape functions of the e-th element, Γ^e is the global element coordinates and $\mathcal{L}^e$ represent the local element coordinates of the e-th element. As an example, a flat (linear) triangular element will have

$$\mathbf{N}^e(\mathbf{u}) = \begin{bmatrix} 1-u_1-u_2 \\ u_1 \\ u_2 \end{bmatrix}, \ u_1 \in [0,1], \ u_2 \in [0, 1-u_1], \tag{5}$$

with the columns of $\mathbf{X}^e$ being equal to the corners of the triangle. The integral over the e-th element can be approximated using a quadrature scheme as

$$\begin{aligned} \int_{\Gamma^e} f(\mathbf{y}) \, \mathrm{d}S_{\mathbf{y}} &= \int_{\mathcal{L}^e} \mathcal{J}(\mathbf{u}) f(\mathbf{u}) \, \mathrm{d}\mathbf{u} \\ &\approx \sum_{i=1}^{Q} w_i \mathcal{J}(\mathbf{u}_i) f(\mathbf{u}_i), \end{aligned} \tag{6}$$

where $(w_i, \mathbf{u}_i)$ is a set of quadrature weights and points, Q is the number of quadrature weights/points and $\mathcal{J}(\mathbf{u})$ is the Jacobian of the parametrization. For 2D surface in 3D, which is the case for BEM, the Jacobian represents the area distortion from $\mathcal{L}^e$ to Γ^e in (4). This area distortion can be computed as

$$\mathcal{J}(\mathbf{u}) = \left\| \left(\mathbf{X}^e \frac{\mathrm{d}\mathbf{N}^e(\mathbf{u})}{\mathrm{d}u_1} \right) \times \left(\mathbf{X}^e \frac{\mathrm{d}\mathbf{N}^e(\mathbf{u})}{\mathrm{d}u_2} \right) \right\|_2. \tag{7}$$

For the second step the discretization of the pressure and normal velocity is

$$p(\mathbf{x}) \approx \mathbf{T}(\mathbf{x})\mathbf{p}, \quad v_{\mathbf{n}}(\mathbf{x}) \approx \mathbf{T}(\mathbf{x})\mathbf{v}_{\mathbf{n}}, \tag{8}$$

where $\mathbf{T}(\mathbf{x})$ is a row vector containing the global nodal interpolation functions while $\mathbf{p}$ and $\mathbf{v}_{\mathbf{n}}$ contain the nodal values of respectively the pressure and normal velocity. Taking the pressure as an example it follows further that the pressure on the e-th element can be described as

$$p(\mathbf{x}^e(\mathbf{u})) = \mathbf{T}(\mathbf{x}^e(\mathbf{u}))\mathbf{p} = \underbrace{\mathbf{T}(\mathbf{x}(\mathbf{u}))(\mathbf{L}^e)^\top}_{\mathbf{T}^e(\mathbf{u})} \underbrace{\mathbf{L}^e\mathbf{p}}_{\mathbf{p}^e}, \ \mathbf{u} \in \mathcal{L}^e$$

where $\mathbf{L}^e$ is a permutation-like matrix that extracts the relevant rows of $\mathbf{p}$ and orders them into $\mathbf{p}^e$ such that they correspond to the order of the local basis functions of $\mathbf{T}^e(\mathbf{u})$. The advantage of this description is that $\mathbf{T}^e(\mathbf{u})$ can be chosen to the same for all elements. As an example, a continuous linear interpolation of the pressures will have

$$\mathbf{T}^e(\mathbf{u}) = \begin{bmatrix} 1-u_1-u_2 & u_1 & u_2 \end{bmatrix}, \tag{9}$$

where $u_1 \in [0,1]$ and $u_2 \in [0, 1-u_1]$. The normal velocity on the e-th element can be defined analogously. Inserting the element approximations of the pressure, the normal velocity and the test function into (1), while imposing that the equality has to hold for all $\mathbf{a} \in \mathbb{C}^n$, the following linear system is generated

$$\alpha \mathrm{diag}(\zeta)\mathbf{p} - \mathbf{F}\mathbf{p} - \alpha \mathrm{i}\rho c k \mathbf{G}\mathbf{v}_{\mathbf{n}} = \mathbf{0}, \tag{10}$$

where the kth row of $\mathbf{G}$ is approximated using a quadrature scheme as

$$\begin{aligned} &\underbrace{\left(\sum_{e=1}^{N} \left(\int_{\Gamma^e} G(\mathbf{t}_k, \mathbf{y}) \mathbf{T}(\mathbf{y}) \, \mathrm{d}S_{\mathbf{y}} \right) \right)}_{k\text{th row of } \mathbf{G}} \\ &\approx \left(\sum_{e=1}^{N} \left(\sum_{i=1}^{Q(k,e)} G(\mathbf{t}_k, \mathbf{y}^e(\mathbf{u}_i)) w_i \mathcal{J}(\mathbf{u}_i) \mathbf{T}^e(\mathbf{u}_i) \right) \mathbf{L}^e \right), \end{aligned} \tag{11}$$

with a similar approximation of the kth row of $\mathbf{F}$

$$\begin{aligned}&\Bigg(\underbrace{\sum_{e=1}^{N}\left(\int_{\Gamma^e}\frac{\partial G(\mathbf{t}_k,\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}\mathbf{T}(\mathbf{y})\,\mathrm{d}S_{\mathbf{y}}\right)}_{k\text{th row of }\mathbf{F}}\Bigg)\\ &\approx\left(\sum_{e=1}^{N}\left(\sum_{i=1}^{Q(k,e)}\frac{\partial G(\mathbf{t}_k,\mathbf{y}^e(\mathbf{u}_i))}{\partial \mathbf{n}(\mathbf{y}^e(\mathbf{u}_i))}w_i\mathcal{J}(\mathbf{u}_i)\mathbf{T}^e(\mathbf{u}_i)\right)\mathbf{L}^e\right).\end{aligned}\tag{12}$$

In both cases the number of quadrature points, $Q(k,e)$, depends on the collocation point and the element.

While (10) can be solved directly, a naive discretization will result in $\mathbf{G},\mathbf{F}\in\mathbb{C}^{n\times n}$ being fully populated matrices. Due to the $\mathcal{O}(n^2)$ scaling of the memory, as well as the matrix vector products, this naive approach will only be possible for relatively small number of Degrees of Freedom ($n<50.000$). In the following section we show how either the Fast Multipole Method (FMM) [1] or Hierarchical matrices ($\mathcal{H}$-matrices) [5] can be utilized to reduce the scaling respect to both memory and computation from $O(n^2)$ to $O(n\log(n))$.

2.1 Acceleration Methods

In order to simplify the description of how to accelerate the Boundary Element (BE) computations we assume that the integration of each element requires the same Q quadrature points to be approximately correct. This will be the source of numerical error for every element close to the collocation point. This error can, however, be dealt with by a so-called near field correction step (as shown in (15)). As a result of this assumption it is possible to describe the product of $\mathbf{G}$ and $\mathbf{F}$ with a known vector $\mathbf{z}$ easily. In order to avoid too large expressions in the following we again zoom in on the multiplication with the kth rows. Starting with $\mathbf{G}$ it follows that

$$\begin{aligned}&\Bigg(\underbrace{\sum_{e=1}^{N}\left(\int_{\Gamma^e}G(\mathbf{t}_k,\mathbf{y})\mathbf{T}(\mathbf{y})\,\mathrm{d}S_{\mathbf{y}}\right)}_{k\text{th row of }\mathbf{G}}\Bigg)\mathbf{z}\\ &\approx\left(\sum_{j=1}^{NQ}G(\mathbf{t}_k,\mathbf{y}_j)\underbrace{w_j\mathcal{J}(\mathbf{u}_j)\mathbf{T}^{e(j)}(\mathbf{u}_j)\mathbf{L}^{e(j)}}_{j\text{th row of }\mathbf{C}}\right)\mathbf{z}\\ &=\begin{bmatrix}G(\mathbf{t}_k,\mathbf{y}_1) & \dots & G(\mathbf{t}_k,\mathbf{y}_{NQ})\end{bmatrix}\mathbf{C}\mathbf{z},\end{aligned}\tag{13}$$

where the subscript j refers to an ordering of the collection of Gaussian points from all elements and $e(j)$ is a function that returns the element number that Gaussian point j is located on. The matrix $\mathbf{C}$ is sparse and can be thought of as a transformation of $\mathbf{z}$ into coefficients $\mathbf{c}=\begin{bmatrix}c_1 & c_2 & \dots & c_{NQ}\end{bmatrix}^\top$. A similar approach can be applied to $\mathbf{F}$ resulting in

$$\begin{aligned}&\Bigg(\underbrace{\sum_{e=1}^{N}\left(\int_{\Gamma^e}\frac{\partial G(\mathbf{t}_k,\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}\mathbf{T}(\mathbf{y})\,\mathrm{d}S_{\mathbf{y}}\right)}_{k\text{th row of }\mathbf{F}}\Bigg)\mathbf{z}\\ &\approx\left(\sum_{j=1}^{NQ}\nabla G(\mathbf{t}_k,\mathbf{y}_j)\cdot\mathbf{n}(\mathbf{y}_j)\underbrace{w_j\mathcal{J}(\mathbf{u}_j)\mathbf{T}^{e(j)}(\mathbf{u}_j)\mathbf{L}^{e(j)}}_{j\text{th row of }\mathbf{C}}\right)\mathbf{z}\\ &=\begin{bmatrix}\nabla G(\mathbf{t}_k,\mathbf{y}_1)\cdot\mathbf{n}(\mathbf{y}_1) & \dots & \nabla G(\mathbf{t}_k,\mathbf{y}_{NQ})\cdot\mathbf{n}(\mathbf{y}_{NQ})\end{bmatrix}\mathbf{C}\mathbf{z},\end{aligned}\tag{14}$$

where $\mathbf{C}$ is the same transform as for the $\mathbf{G}$ operator. In order to fix the numerical inaccuracies caused by the elements close to each collocation we add a so-called near field correction matrix, $\mathbf{S}$, to the matrix product. As such multiplying with either $\mathbf{G}$ and $\mathbf{F}$ is as follows

$$\mathbf{A}=\mathbf{B}\mathbf{C}+\mathbf{S},\quad \mathbf{S},\mathbf{A}\in\mathbb{C}^{n\times n},\ \mathbf{B},\mathbf{C}^\top\in\mathbb{C}^{n\times NQ},\tag{15}$$

where $\mathbf{A}$ represent either $\mathbf{G}$ or $\mathbf{F}$, $\mathbf{B}$ is the fully populated part for which multiplication can be accelerated by the FMM or a $\mathcal{H}$-matrix, $\mathbf{C}$ is the coefficient map and $\mathbf{S}$ is the near field correction. In short $\mathbf{S}$ corrects the computation by subtracting the wrong integration done by using only Q Gaussian points and adds on the correct integration instead. Note that both $\mathbf{S}$ and $\mathbf{C}$ are sparse matrices, meaning that their assembly and matrix-vector-products scale $\mathcal{O}(n\tau)$ where $\tau\ll n$. As such using an approximate scheme for $\mathbf{B}$ is all that is needed to reduce the memory and computational scaling of $\mathbf{G}$ and $\mathbf{F}$.

2.2 Viscous and thermal losses

The inclusion of viscous and thermal losses can be achieved utilizing the Kirchoff Decomposition [6]. This approach splits the problem into three separate Helmholtz equations, commonly referred to as modes, as follows

$$\text{Acoustic Mode:}\quad (\Delta+k_a^2)p_a(\mathbf{x})=0,\tag{16}$$

$$\text{Thermal Mode:}\quad (\Delta+k_h^2)p_h(\mathbf{x})=0,\tag{17}$$

$$\text{Viscous Mode:}\quad (\Delta+k_v^2)\mathbf{v}_v(\mathbf{x})=\mathbf{0},\tag{18}$$

where the viscous mode is divergence free ($\nabla\cdot\mathbf{v}_v(\mathbf{x})=0$). The three modal wavenumbers ($k_a,\ k_h$ & k_v) all depend on the lossless wavenumber (k) as well as the physical properties of the fluid, such as the thermal conductivity, specific heat capacity under constant pressure and

the shear/bulk viscosity coefficients [7]. The three modes are coupled through the isothermal and no-slip boundary conditions. The first boundary condition states that there is no change of the surface temperature while the second boundary condition states that the fluid sticks to the surface. Mathematically the two boundary conditions are given as

$$\begin{aligned} p_a(\mathbf{x})\tau_a + p_h(\mathbf{x})\tau_h = 0, \\ \nabla p_a(\mathbf{x})\phi_a + \nabla p_h(\mathbf{x})\phi_h + \mathbf{v}_v(\mathbf{x}) = \mathbf{v}_s(\mathbf{x}), \end{aligned} \tag{19}$$

where $\mathbf{x} \in \Gamma$, $\mathbf{v}_s(\mathbf{x})$ is prescribed surface velocity at point $\mathbf{x}$ and like the modal wavenumbers the constants τ_a, τ_h, ϕ_a, and ϕ_h depends on the lossless wavenumber and the physical properties of the fluid. The total pressure and velocity can be extracted from the three modes as follows

$$p_t = p_a + p_h, \tag{20}$$

$$\mathbf{v}_t = \mathbf{v}_a + \mathbf{v}_h + \mathbf{v}_v. \tag{21}$$

Each of the three modes give rise to a linear system of equations similar to that of (10). Writing this out it means that the following must be true

$$\begin{aligned} \mathbf{H}_a\mathbf{p}_a + \mathbf{G}_a\tfrac{\partial \mathbf{p}_a}{\partial \mathbf{n}} = \mathbf{0}, \quad \mathbf{H}_h\mathbf{p}_h + \mathbf{G}_h\tfrac{\partial \mathbf{p}_h}{\partial \mathbf{n}} = \mathbf{0}, \\ \mathbf{H}_v\mathbf{v} + \mathbf{G}_v\tfrac{\partial \mathbf{v}}{\partial \mathbf{n}} = \mathbf{0}, \end{aligned} \tag{22}$$

where $\mathbf{H}_i = \alpha \mathrm{diag}(\zeta_i) - \mathbf{F}_i$ and $\mathbf{G}_i = -\alpha \mathrm{i}\rho c k \mathbf{G}_i$ for $i \in \{a, h, v\}$. Note that the viscous mode is special as it is a result of a vector Helmholtz equation. In the above it was chosen to separate the x, y and z components as $\mathbf{v} = \begin{bmatrix} \mathbf{v}_x & \mathbf{v}_y & \mathbf{v}_z \end{bmatrix}^\top$ and $\frac{\partial \mathbf{v}}{\partial \mathbf{n}} = \begin{bmatrix} \frac{\partial \mathbf{v}_x}{\partial \mathbf{n}} & \frac{\partial \mathbf{v}_y}{\partial \mathbf{n}} & \frac{\partial \mathbf{v}_z}{\partial \mathbf{n}} \end{bmatrix}^\top$. Using this description will result in $\mathbf{G}_v$ and $\mathbf{H}_v$ being block diagonal matrices with the three blocks being identical and computed similar to the acoustical and thermal mode but with wavenumber k_v. In order to assert the null-divergence of the viscous mode, as well as applying the no-slip boundary condition, we need to be able to compute the gradient. The idea in both cases is to split the gradient computation into a tangential part (denoted by $^\parallel$) and a orthogonal part (denoted by $^\perp$). In the case of the divergence this means that

$$\nabla \cdot \mathbf{v}_v = \nabla^\parallel \cdot \mathbf{v}_v + \nabla^\perp \cdot \mathbf{v}_v, \tag{23}$$

while for the gradients this means that

$$\nabla p_a = \nabla^\parallel p_a + \nabla^\perp p_a, \quad \nabla p_h = \nabla^\parallel p_h + \nabla^\perp p_h. \tag{24}$$

It turns out that since the Boundary Element discretization only prescribes values on the surface, then the tangential part of the gradient is extracted directly from the interpolation functions. Likewise the orthogonal part can be extracted from the Boundary Element discretization of the normal derivatives by multiplication with the normals. As such the divergence can be computed as

$$\nabla \cdot \mathbf{v} = \mathbf{D}_r\mathbf{v} + \mathbf{N}^\top \frac{\partial \mathbf{v}}{\partial \mathbf{n}}, \tag{25}$$

where $\mathbf{N} = \begin{bmatrix} \mathrm{diag}(\mathbf{n}_x) & \mathrm{diag}(\mathbf{n}_y) & \mathrm{diag}(\mathbf{n}_z) \end{bmatrix}^\top$ with $\mathbf{n}_x$, $\mathbf{n}_y$ and $\mathbf{n}_z$ being respectively the vectors collecting the x, y and z components of the normals at the collocation points and $\mathbf{D}_r = \begin{bmatrix} \mathbf{D}_x & \mathbf{D}_y & \mathbf{D}_z \end{bmatrix}$ with the sparse matrices $\mathbf{D}_x$, $\mathbf{D}_y$ and $\mathbf{D}_z$ being computed directly from the Boundary Element interpolation using the interpolation function derivative approach [6]. Using the same idea for the gradients it is found that

$$\nabla \mathbf{p}_a = \mathbf{D}_c\mathbf{p}_a + \mathbf{N}\frac{\partial \mathbf{p}_a}{\partial \mathbf{n}}, \ \nabla \mathbf{p}_h = \mathbf{D}_c\mathbf{p}_h + \mathbf{N}\frac{\partial \mathbf{p}_h}{\partial \mathbf{n}}, \tag{26}$$

where $\mathbf{D}_c = \begin{bmatrix} \mathbf{D}_x^\top & \mathbf{D}_y^\top & \mathbf{D}_z^\top \end{bmatrix}^\top$.

Putting everything together it turns out that the acoustical pressure can be computed by solving the following linear system of equations

$$\begin{aligned} &\left[\mathbf{G}_a\left(\mu_a\left(\mathbf{RN}\right)^{-1}\mathbf{RD}_c + \mu_h\mathbf{G}_h^{-1}\mathbf{H}_h\right) - \phi_a\mathbf{H}_a\right]\mathbf{p}_a \\ &= \mathbf{G}_a\left(\mathbf{RN}\right)^{-1}\mathbf{Rv}_s. \end{aligned} \tag{27}$$

where $\mu_h = {}^{\tau_a\phi_h}/_{\tau_h}$, $\mu_a = \phi_a - \mu_h$ and $\mathbf{v}_s = \begin{bmatrix} \mathbf{v}_{s_x} & \mathbf{v}_{s_y} & \mathbf{v}_{s_z} \end{bmatrix}^\top$ is the boundary velocities stacked with respect to the x, y, and z direction and

$$\mathbf{R} = \mathbf{D}_r - \mathbf{N}^\top\mathbf{G}_v^{-1}\mathbf{H}_v. \tag{28}$$

An important detail to mention is that only the acoustical mode, corresponding to $\mathbf{G}_a$ and $\mathbf{H}_a$, are fully populated matrices while the remaining matrices in (27) are sparse. This means that simply utilizing one of the acceleration strategies for the acoustical mode is enough to make (27) solvable for even relatively large problems. Using the no-slip boundary condition it is possible to extract the viscous velocity as

$$\mathbf{v} = \mathbf{v}_s - \left(\mu_a\mathbf{D}_c + \mu_h\mathbf{N}\mathbf{G}_h^{-1}\mathbf{H}_h - \phi_a\mathbf{N}\mathbf{G}_a^{-1}\mathbf{H}_a\right)\mathbf{p}_a. \tag{29}$$

The normal and tangential components of the viscous velocity, as described above, will later be used to verify the simulation results (Figure 7).

3. IMPLEMENTATION

This section gives an overview of the currently supported elements types, mesh file formats, and acceleration methods.

3.1 Element Types

The software package currently supports triangular and quadrilateral elements. For the geometry both linear and quadratic elements are supported. Additionally the global interpolation functions for the pressure and normal derivative, as shown in (8), can be chosen to be discontinuous constant, linear and quadratic. For continuous elements interpolation nodes are being put on the boundary of the element while for the discontinuous elements the interpolation nodes are exclusively put inside the domain of the elements. The different setups of geometric and interpolation elements of triangles can be seen in Figure 1 while the elements for the quadrilaterals can be seen in Figure 2.

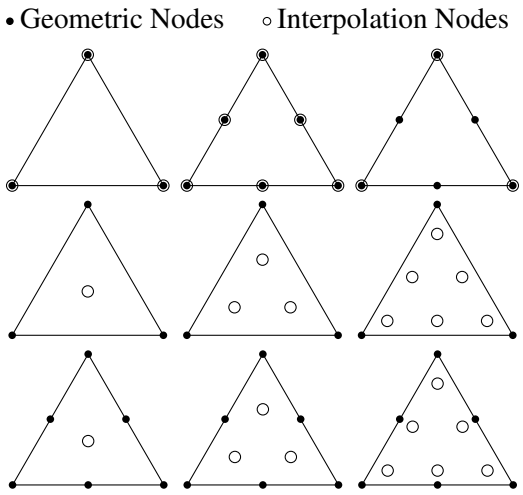

Figure 1. Top row: Linear and quadratic continuous interpolation. Middle row: Linear Geometry with discontinuous Constant, linear and quadratic interpolation. Bottom row: Quadratic Geometry with discontinuous constant, linear and quadratic interpolation.

3.2 Meshes

Currently flat triangular panels can be imported through the `.obj`, `.stl`, `.ply`, `.off` and `.2DM` file formats

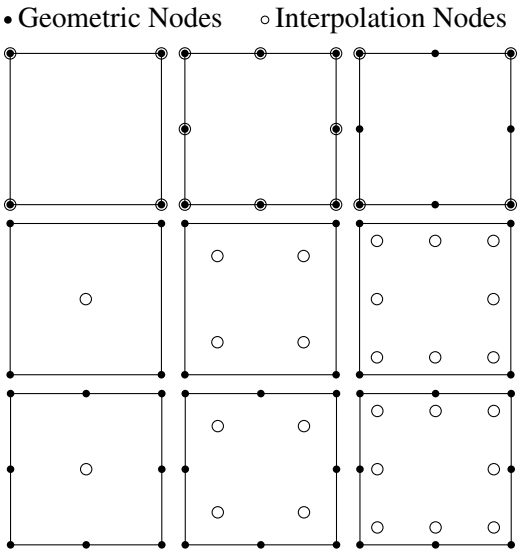

Figure 2. Top row: Linear and quadratic continuous interpolation. Middle row: Linear Geometry with discontinuous Constant, linear and quadratic interpolation. Bottom row: Quadratic Geometry with discontinuous constant, linear and quadratic interpolation.

using the `MeshIO.jl` package. Additionally both triangular and quadrilateral panels of linear and quadratic order can be imported using the `.mphtxt` file format from COMSOL Multiphysics® [8].

3.3 Fast Operators

The implementation currently supports applying $\mathbf{G}$ and $\mathbf{F}$ using the Fast Multipole Method [1] (through the FMM3D library [9]) and the $\mathcal{H}$-matrix approach [5] (using the `HMatrices.jl` package [10]). Additionally there is an experimental implementation of the Interpolated Factored Green's Function approach [11] (using the `IFGF.jl` package [12]).

4. EXAMPLES

In the following examples, it was chosen to set the speed of sound as $c = 343\,\mathrm{m\,s^{-1}}$ and the medium density of air as $\rho = 1.21\,\mathrm{kg\,m^{-3}}$. Lastly, we note that a more in-depth descriptions of each the three examples, including the code, can be found in the online documentation of the software.

4.1 Rigid sphere with radius equal to 1m

The acoustic scattering of a rigid sphere coming from a plane wave traveling in the z-direction can be computed analytically as [13]

$$p^{\text{analytical}}(r,\theta) = P_0 \exp\left(\mathrm{i}kr\cos(\theta)\right) - P_0 \sum_n^{\infty} \mathrm{i}^n (2n+1) \frac{j_n'(ka)}{h_n'(ka)} P_n(\cos(\theta)) h_n(kr),$$

where r is the distance to the origin, θ is the colatitude angle, P_0 is the magnitude of the plane wave, j_n is the spherical Bessel function of the first kind, h_n is the spherical Hankel function of the first kind and a is the radius of the sphere.

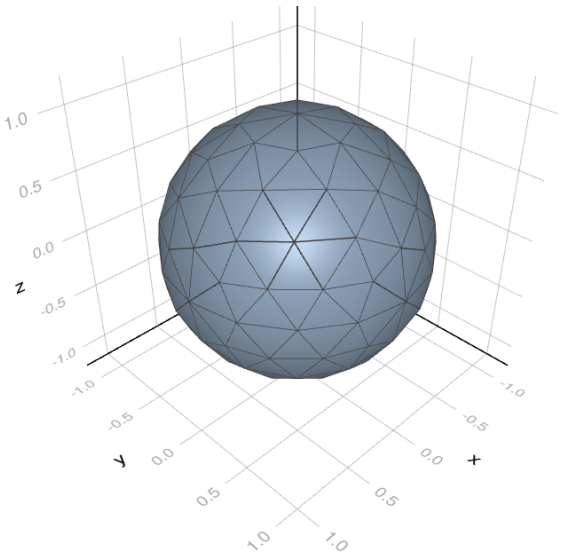

Figure 3. Visualization of the sphere mesh. Note that while the mesh is quadratic only flat triangles are plotted due to limitations in the used plotting library.

In the simulation we set $P_0 = 1\,\mathrm{Pa}$ and $f = 100\,\mathrm{Hz}$. The geometry of the sphere was approximated using 246 quadratic triangular elements with the interpolation function set to be discontinuous linear (Figure 3). As a result the simulation had 738 Degrees of Freedom. Furthermore given that the sphere is rigid we set $\mathbf{v_n} = \mathbf{0}$ in (10). As such this simulation solves

$$(\alpha \mathrm{diag}(\zeta) - \mathbf{F})\,\mathbf{p} = \mathbf{p}^{\text{incident}}, \tag{30}$$

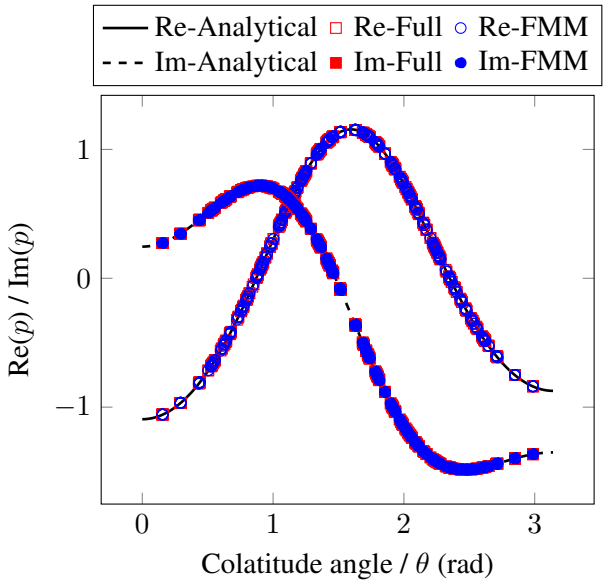

Figure 4. Real and imaginary part of the surface pressure as a function of the angle for a rigid sphere with radius 1 m at 100 Hz.

where $\mathbf{p}^{\text{incident}}$ is the pressure of the plane wave at the collocation nodes. The results of the simulation can be seen in Figure 4.

4.2 Plane wave in cube with side lengths equal to 1m

The analytical description of the pressure from a plane wave is given by

$$p(\mathbf{x}) = P_0 \exp\left(-\mathrm{i}\mathbf{k}\cdot\mathbf{x}\right), \tag{31}$$

where P_0 is the magnitude of the plane wave, $\mathbf{k}$ is the wave vector and $\mathbf{x}$ is the position vector. If a plane wave is perfectly aligned with the sides of an open duct the pressure should behave the same as (31), i.e. a plane wave in the free field. In the simulation we align the wave with the x-direction and set $P_0 = 1\,\mathrm{Pa}$ and $f = 54.59\,\mathrm{Hz}$. Given the flat sides of the cube it was chosen to use 224 linear elements for the geometry while the chosen interpolation was set to be discontinuous quadratic resulting in 1344 Degrees of Freedom. For the simulation the side at $x = 0$ is applied a pressure condition equal to P_0 while the side at $x = 1$ is applied a ρc impedance condition. The remaining boundaries are set as rigid ($\mathbf{v}_n = \mathbf{0}$). The resulting linear system of equations becomes

$$\left(\mathbf{H}\mathrm{diag}(\mathbf{p}_0^{\mathbb{C}}) - \alpha \mathrm{i}\rho c k \mathbf{G}\mathrm{diag}(\mathbf{y})\right)\mathbf{z} = -\mathbf{H}\mathrm{diag}(\mathbf{p}_0), \tag{32}$$

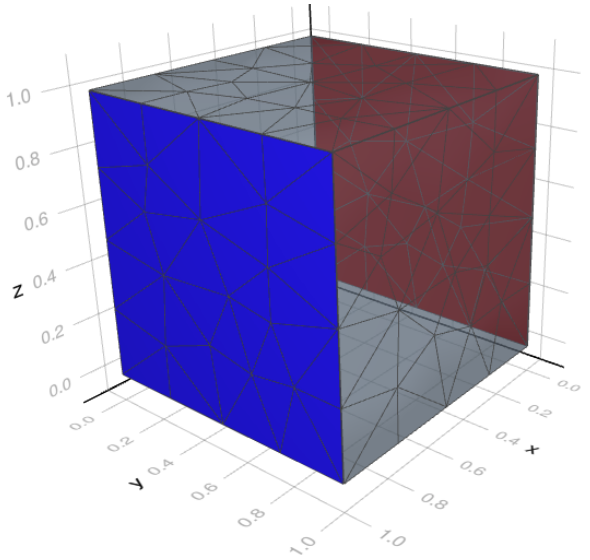

Figure 5. Visualization of the cube mesh. The red side ($x = 0$) is applied a pressure condition while the blue side ($x = 1$) is applied an impedance condition.

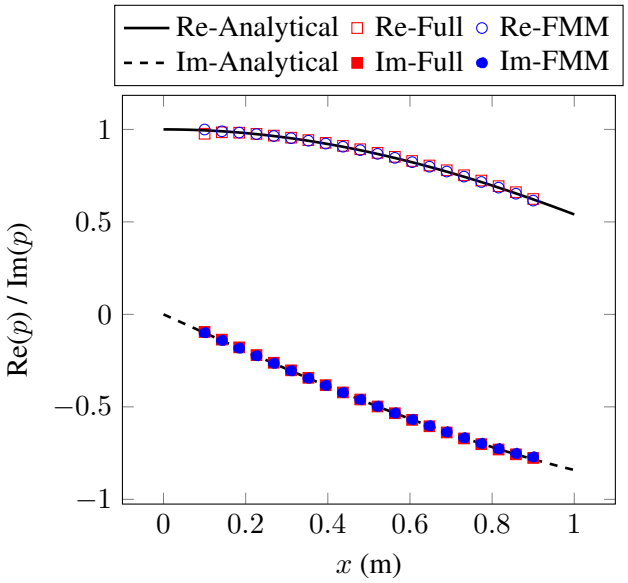

Figure 6. Real and imaginary part of the pressure on the line $(x, 0.5, 0.5)$ inside of the duct of side lengths 1 m at 54.59 Hz.

where $\mathbf{H} = (\alpha \text{diag}(\zeta) - \mathbf{F})$, $\mathbf{p}_0$ is a vector containing P_0 at the collocation points positioned at $x = 0$, $\mathbf{p}_0^{\mathbf{C}} = \mathbf{1} - \mathbf{p}_0/P_0$ and $\mathbf{y}$ contains the prescribed admittance at the collocation points positioned at $x = 1$. Finally, (32) is solved iteratively without ever forming the matrix-matrix products. From the solution ($\mathbf{z}$) the unknown surface pressures and normal velocities are extracted. Using this the pressure inside the cube is evaluated using (1) directly. The results can be seen in Figure 6.

4.3 Oscillating sphere with radius equal to 1m including viscothermal losses

The analytical solution of a sphere oscillating in the z-direction in a viscous fluid is described in section 6.9 of [14]. While the solution is stated as a infinite series it is dominated by the first order terms. As such the analytical solution for the acoustical pressure, normal velocity and tangential velocity can be approximated accurately as

$$
\begin{aligned}
p_a^{\text{analytical}}(r,\theta) &\approx -3\rho c A_1 h_1^{(1)}(kr)\cos(\theta) \\
|\mathbf{v_n}|^{\text{analytical}}(r,\theta) &\approx -\frac{6\mathrm{i}kB_1}{k_v r} h_1(k_v r)\cos(\theta) \\
|\mathbf{v_t}|^{\text{analytical}}(r,\theta) &\approx -\frac{3\mathrm{i}B_1\left(k_v r h_1'(k_v r) + h_1(k_v r)\right)}{r}\sin(\theta),
\end{aligned}
$$

where r is the distance to the origin, θ is the colatitude angle, h_1 is the spherical Hankel function of the first kind, k_v is the viscous wavenumber, and the coefficients A_1 and B_1 is computed as the solution to

$$
\begin{bmatrix} v_z/(3\mathrm{i}k) \\ a v_z/(3\mathrm{i}) \end{bmatrix} = \begin{bmatrix} h_1'(ka) & -2h_1(k_v a)/(ka) \\ h_1(ka) & -\left(k_v a h_1'(k_v a) + h_1(k_v a)\right) \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \end{bmatrix},
$$

where a is the radius of the sphere and v_z is the speed in the z-direction.

In the simulation we set $f = 100\,\text{Hz}$ and $v_z = 0.01\,\text{m}\,\text{s}^{-1}$. The mesh used was the same as for the scattering case (Figure 3). This time, however, utilizing continuous quadratic interpolation functions resulting in 494 Degrees of Freedom. Given the velocity in the z-direction the following surface velocity vector is created

$$
\mathbf{v}_s = \begin{bmatrix} \mathbf{0} & \mathbf{0} & v_z\mathbf{1} \end{bmatrix}^\top \in \mathbb{R}^{3n}, \tag{33}
$$

where $\mathbf{0}$ denotes a vector filled with zeros of length n and $\mathbf{1}$ denotes a vector filled with ones of length n. In order to compute the acoustical, $\mathbf{p}_a$, we solve (27) directly while $\mathbf{v}$ is computed afterwards using (29). The results of the simulation can be seen in Figure 7.

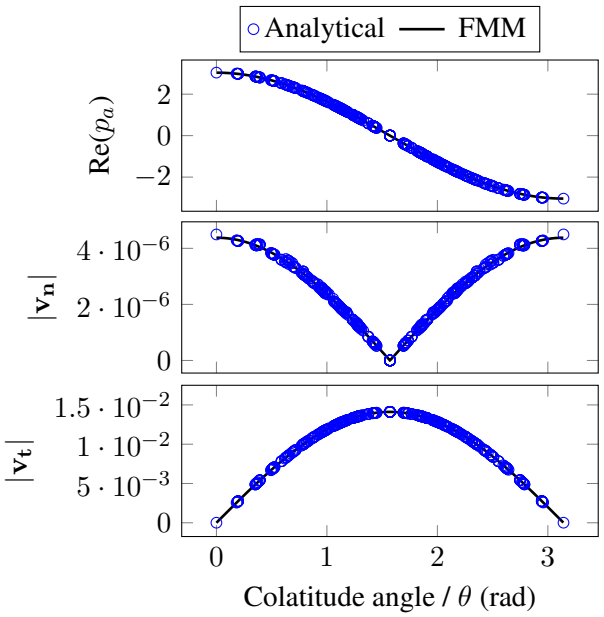

Figure 7. Real part of the acoustical pressure as well as absolute value of the normal and tangential viscous velocity for a sphere with 1 m radius in at 100 Hz.

5. CONCLUSIONS

In this paper some of the current features and implementation details of the open-source software `BoundaryIntegralEquations.jl` were explained. The package were validated on three different examples where a good compliance between the analytical solution and the simulation results. While the features are not expected to change we suggest that the reader look online for the most recent version of both the software and documentation.

6. ACKNOWLEDGMENTS

Many thanks to Simone Preuss (Lehrstuhl für Akustik mobiler Systeme, TUM) for many good discussions on BEM and the inclusion of viscothermal losses. It is always a pleasure.

7. REFERENCES

[1] L. Yijun, *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, 2009.

[2] V. C. Henríquez and P. M. Juhl, "OpenBEM - an open source Boundary Element Method software in Acoustics," *Proceedings of Internoise 2010*, vol. 7, pp. 5796–5805, 2010.

[3] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A Fresh Approach to Numerical Computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[4] S. Marburg, "The Burton and Miller Method: Unlocking Another Mystery of Its Coupling Parameter," *Journal of Computational Acoustics*, vol. 24, no. 1, p. 1550016, 2016.

[5] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*. Springer, 2008.

[6] V. Cutanda Henríquez and P. R. Andersen, "A Three-Dimensional Acoustic Boundary Element Method Formulation with Viscous and Thermal Losses Based on Shape Function Derivatives," *Journal of Computational Acoustics*, vol. 26, no. 3, 2018.

[7] M. Brunaeu, P. Herzog, J. Kergomard, and J. Polack, "General formulation of the dispersion equation in bounded visco-thermal fluid, and application to some simple geometries," *Wave Motion*, vol. 11, no. 5, pp. 441–451, 1989.

[8] COMSOL AB, Stockholm, Sweden., "COMSOL Multiphysics®," 2021.

[9] T. Askham, Z. Gimbutas, L. Greengard, L. Lu, J. Magland, D. Malhotra, M. O'Neil, M. Rachh, and V. Rohklin, "FMM3D." `https://github.com/flatironinstitute/FMM3D`, 2023.

[10] L. M. Faria, "HMatrices.jl." `https://github.com/WaveProp/HMatrices.jl`, 2023.

[11] C. Bauinger and O. P. Bruno, ""Interpolated Factored Green Function" Method for accelerated solution of Scattering Problems," *Journal of Computational Physics*, vol. 430, p. 110095, 2021.

[12] L. M. Faria, "IFGF.jl." `https://github.com/WaveProp/IFGF.jl`, 2023.

[13] F. Ihlenburg, *Finite Element Analysis of Acoustic Scattering*. Springer, 1998.

[14] S. Temkin, *Elements of Acoustics*. Acoustical Society of America, 2001.