EUROPEAN JOURNAL OF COMPUTATIONAL MECHANICS, 2017
VOL. 26, NO. 4, 377–393
https://doi.org/10.1080/17797179.2017.1306832

Fast Boundary Element Method for acoustics with the Sparse Cardinal Sine Decomposition

François Alouges, Matthieu Aussal and Emile Parolin

CMAP, Ecole polytechnique, CNRS, Université Paris-Saclay, Palaiseau, France

ABSTRACT
This paper presents the newly proposed method *Sparse Cardinal Sine Decomposition* that allows fast convolution on unstructured grids. We focus on its use when coupled with finite element techniques to solve acoustic problems with the (compressed) Boundary Element Method. In addition, we also compare the computational performances of two equivalent MATLAB® and Python implementations of the method. We show validation test cases in order to assess the precision of the approach. Eventually, the performance of the method is illustrated by the computation of the acoustic target strength of a realistic submarine from the Benchmark Target Strength Simulation international workshop.

ARTICLE HISTORY
Received 28 October 2016
Accepted 6 December 2016

KEYWORDS
Boundary Element Method; acoustics; quadrature; fast convolution; Non-Uniform Fast Fourier Transform

1. Introduction

Convolutions on unstructured grids are at the heart of many numerical methods used for solving problems arising from the physical modelling in many different fields. Indeed, it is one of the main ingredients for the use of the Boundary Element Method (BEM) in electrostatics, magnetostatic, acoustics, linear elasticity or electromagnetism, to quote only a few domains. Classical methods used to compute the convolution have a quadratic complexity, $\mathcal{O}(N^2)$, in terms of the number N of discretisation points in the considered object. Depending on the machine used, this usually forbids to use the method in practice when N exceeds a few tens of thousands. To overcome this limit, fast methods, such as the Fast Multipole Method (FMM) (Greengard, 1988; Greengard & Rokhlin, 1987) and the $\mathcal{H}$-Matrices (Hackbusch, 1999), have been developed to reduce the overall complexity to $\mathcal{O}(N \log N)$, to the price of a much higher implementation complexity. More recently, a new method, referred to as the *Sparse Cardinal Sine Decomposition* (SCSD) method (Alouges & Aussal, 2014; Aussal, 2014), was invented to address this particular problem in a simpler way. Originally developed for point-to-point interactions, it is based on a suitable Fourier decomposition of the Green kernel, sparse quadrature formulae and

CONTACT Emile Parolin emile.parolin@polytechnique.edu

Type-III Non-Uniform Fast Fourier Transforms (Greengard & Lee, 2004; Lee & Greengard, 2005).

We propose in this paper to extend the use of the SCSD, in particular to Fast BEM problems, where we also need to consider a finite element discretisation. We show in particular that we can use the point-to-point interaction with the integration points. For the sake of simplicity, we focus on the acoustic scattering problems, although the method can be extended to other problems.

This paper is organised as follows. We first introduce the problem of interest and explain the motivation behind the development of methods for fast numerical convolution. The basic SCSD methodology is then outlined and we provide numerical evidence of the efficiency of the method. In particular, we give comparisons between the SCSD and other established techniques such as $\mathcal{H}$-Matrix compression (Hackbusch, 1999) and FMMs (Greengard, 1988; Greengard & Rokhlin, 1987).

Eventually, we provide the reader with a complete industrial test case that consists in computing the acoustic target strength of a realistic submarine that was proposed in the Benchmark Target Strength Simulation international workshop (Schneider et al., 2003).

2. Integral equation formulation

Let us first introduce the problem of interest. We consider a surface obstacle Γ dividing the space $\mathbb{R}^3$ in two open interior Ω^i and exterior Ω^e domains. Given an incident acoustic field p_{inc} defined in $\mathbb{R}^3$, we aim at evaluating the total acoustic field $p = p_{\text{sca}} + p_{\text{inc}}$ resulting in the scattering by the obstacle and characterised by the scattered acoustic field p_{sca}. We suppose in the following that the obstacle has sound-hard or Neumann boundaries.

The scattering problem can be written as the following Boundary Value Problem (BVP)

$$\begin{aligned} -\left(\Delta p_{\text{sca}} + k^2 p_{\text{sca}}\right) &= 0, \qquad \text{in } \Omega^e, && (1)\\ \partial_n p_{\text{sca}} &= -\partial_n p_{\text{inc}}, \quad \text{on } \Gamma, && (2)\\ r(\partial_r p_{\text{sca}} - ikp_{\text{sca}}) &\to 0, \qquad \text{as } r \to \infty. && (3) \end{aligned}$$

The Equation (3) is the so-called Sommerfeld radiation condition ensuring sufficient decay at infinity of the scattered field p_{sca}.

The approach that we choose to adopt is to reformulate this BVP as a Boundary Integral Equation (BIE) on the boundary of the scatterer Γ. To do so, we define the single-layer potential $\mathcal{S}$ and double-layer potential $\mathcal{D}$, for sufficiently regular functions, as

$$\mathcal{S}\lambda(\mathbf{x}) := \int_\Gamma G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\, \mathrm{d}\Gamma_{\mathbf{y}}, \quad \mathbf{x} \in \mathbb{R}^3 \backslash \Gamma, \qquad (4)$$

$$\mathcal{D}\mu(\mathbf{x}) := \int_\Gamma \partial_{n_y} G(\mathbf{x},\mathbf{y})\mu(\mathbf{y})\ \mathrm{d}\Gamma_{\mathbf{y}}, \quad \mathbf{x} \in \mathbb{R}^3\backslash\Gamma. \tag{5}$$

Here, $G(\mathbf{x},\mathbf{y}) = e^{-ik|\mathbf{x}-\mathbf{y}|}/(4\pi\,|\mathbf{x}-\mathbf{y}|)$ is the free space Green's function in $\mathbb{R}^3$. Let p be a solution to the Helmholtz equation in $\mathbb{R}^3\backslash\Gamma = \Omega^i \cup \Omega^e$ that satisfies the Sommerfeld radiation condition at infinity. Then, it is well known that the following integral representation holds

$$p = \mathcal{S}\lambda - \mathcal{D}\mu, \qquad \text{in } \mathbb{R}^3\backslash\Gamma. \tag{6}$$

where the jumps μ and λ are defined as

$$\mu := [p] = p^i - p^e, \qquad \lambda := [\partial_n p] = \partial_n p^i - \partial_n p^e, \qquad \text{on } \Gamma. \tag{7}$$

In the previous definitions, the superscripts i, e denote, respectively, the interior and exterior traces on Γ of p. Taking the Neumann exterior trace on Γ in (6) yields the following BIE

$$\partial_n p = -H\mu + \left(-\frac{1}{2}I + D^*\right)\lambda, \qquad \text{on } \Gamma, \tag{8}$$

where I is the identity boundary operator and where the adjoint double-layer D^* and hypersingular H boundary operators are defined, for sufficiently regular functions, as

$$D^*\mu(\mathbf{x}) := \int_\Gamma \partial_{n_x} G(\mathbf{x},\mathbf{y})\mu(\mathbf{y})\ \mathrm{d}\Gamma_{\mathbf{y}}, \quad \mathbf{x} \in \Gamma, \tag{9}$$

$$H\mu(\mathbf{x}) := \int_\Gamma \partial^2_{n_x n_y} G(\mathbf{x},\mathbf{y})\mu(\mathbf{y})\ \mathrm{d}\Gamma_{\mathbf{y}}, \quad \mathbf{x} \in \Gamma. \tag{10}$$

In order to apply this result to our problem, one needs to formally prolongate the unknown p_{sca} by a function satisfying the Helmholtz equation in the interior domain Ω^i. Several formulations are then possible depending on the choice of the prolongating function. To obtain a well-posed problem, we use the Burton–Miller/Brakhage–Werner approach (Burton & Miller, 1971). It rests on a suitable prolongation that yields a proportionality relation between the two jumps on Γ: $\lambda = ik\beta\mu$, for a suitable scalar constant $\beta \in \mathbb{C}$ with a strictly positive real part. The following BIE in the unknown μ on the boundary Γ is finally obtained

$$\left(-H + ik\beta\left(-\frac{1}{2}I + D^*\right)\right)\mu = -\partial_n p_{\text{inc}}, \qquad \text{on } \Gamma. \tag{11}$$

Using Galerkin approximation, we write the Equation (11) in weak form. Namely, calling $R = \left(-H + ik\beta\left(-\frac{1}{2}I + D^*\right)\right)$, we seek μ such that for all test functions μ^t we have

$$\langle R\mu,\ \mu^t\rangle_\Gamma = -\langle \partial_n p_{\text{inc}}, \mu^t\rangle_\Gamma. \tag{12}$$

To solve numerically the integral Equation (11) or its variational counterpart (12), we introduce a discrete approximation space V_N in which to look for the unknown μ. Here, N denotes the number of degrees of freedom or equivalently the dimension of the discrete space. Let $\mathcal{T}$ be a discretisation of the boundary Γ with triangular elements. We construct the approximation space using P^1 Lagrange finite elements on the grid $\mathcal{T}$. We now seek $\mu \in V_N$ such that (12) is satisfied for all test functions $\mu^t \in V_N$.

The discrete problem now takes the form of a linear system, of size N by N and the computation of the elements of this system mostly involves the numerical evaluation of the boundary operators D^* and H. This is typically conducted via Gauss quadrature methods.[1] One is then faced with the evaluation of terms (convolution products) that take the form

$$\tilde{K}_f(\mathbf{x}_i) = \sum_{\mathbf{y}_j} K(\mathbf{x}_i, \mathbf{y}_j) f(\mathbf{y}_j), \qquad \text{for some } \mathbf{x}_i \in \Gamma, \tag{13}$$

where K is a Green kernel, f is a scalar function and the points $(\mathbf{y}_j)_{j=1}^{N_q}$ are the quadrature points on Γ. In a sense, this amounts to evaluate all interactions between two point clouds in $\mathbf{x}$ and $\mathbf{y}$. Moreover, since the Green kernel is not compactly supported, the coefficients $K(\mathbf{x}_i, \mathbf{y}_j)$ do not vanish and the support of $R\mu$ is the whole boundary Γ. As a result, the matrix involved is dense, which is one of the peculiarities of the BEM. We have rewritten the initial problem in $\mathbb{R}^3$ to the boundary Γ to the price of having now a dense matrix instead of the classical sparse matrices that are characteristic of finite element methods. As a consequence, the memory cost in a classical BEM for computing and storing the dense matrix of the linear system is $\mathcal{O}(N^2)$. Solving the linear system using direct solvers then requires typically $\mathcal{O}(N^3)$ operations, making any attempt impractical for large problems (e.g. when N is bigger than a few tens of thousands for a typical machine). Even if one turns to iterative solvers, relying only on matrix–vector products, having a dense matrix still implies a quite important cost of $\mathcal{O}(N^2)$ operations per iteration. There is therefore a need for fast and/or compression methods that aim at reducing this time and storage complexity from quadratic to quasi-linear $\mathcal{O}(N \log N)$. A successful attempt in this direction was first achieved with the FMMs (Greengard, 1988; Greengard & Rokhlin, 1987) and later with hierarchical matrix compression (Hackbusch, 1999).

A more recent approach with a similar complexity, the SCSD method, was proposed by some of the authors (Alouges & Aussal, 2014; Aussal, 2014) in the case of point-to-point interactions, and mostly with Laplace Green kernel. We aim at extending the approach to Helmholtz equation and, using the quadrature formula described above, to finite element discretisations.

3. Sparse Cardinal Sine Decomposition

The building block of the SCSD is the integral representation of the cardinal sine on the unit sphere S in $\mathbb{R}^3$, namely

$$\operatorname{sinc}(|\mathbf{x}-\mathbf{y}|) = \frac{\sin(|\mathbf{x}-\mathbf{y}|)}{|\mathbf{x}-\mathbf{y}|} = \frac{1}{4\pi}\int_S e^{i\mathbf{s}\cdot(\mathbf{x}-\mathbf{y})}\mathrm{d}S(\mathbf{s}). \tag{14}$$

We use a quadrature formula on the sphere S constructed from a set of nodes $(\mathbf{s}_m)_{1\le m\le N_s}$ associated with weights $(\omega_m)_{1\le m\le N_s}$ that enable to write, in the formula above, the cardinal sine as a sum of terms that involves $\mathbf{x}$ and $\mathbf{y}$ separately

$$\operatorname{sinc}(|\mathbf{x}-\mathbf{y}|) \approx \frac{1}{4\pi}\sum_{m=1}^{N_s}\omega_m e^{i\mathbf{s}_m\cdot\mathbf{x}}e^{-i\mathbf{s}_m\cdot\mathbf{y}}. \tag{15}$$

Notice that with such an equation, the (fast) convolution with the cardinal sine is already at hand. Indeed, for all $i = 1,\cdots,N_q$

$$\sum_{j=1}^{N_q}\operatorname{sinc}(|\mathbf{x}_i-\mathbf{y}_j|)f_j \approx \frac{1}{4\pi}\sum_{m=1}^{N_s}e^{i\mathbf{s}_m\cdot\mathbf{x}_i}\left(\omega_m\left(\sum_{j=1}^{N_q}e^{-i\mathbf{s}_m\cdot\mathbf{y}_j}f_j\right)\right). \tag{16}$$

Therefore, the convolution with the cardinal sine follows the three steps:

- Knowing the set of values (f_j), compute $g_m = \sum_{j=1}^{N_q}e^{-i\mathbf{s}_m\cdot\mathbf{y}_j}f_j$ for all $m \in \{1,\cdots,N_s\}$. This can be done with a Type-III Non-Uniform Fast Fourier Transforms (NUFFT) (Greengard & Lee, 2004; Lee & Greengard, 2005) with a complexity $O(N_q\log(N_q))$.
- Compute $\tilde{g}_m = \omega_m g_m$ for all $m \in \{1,\cdots,N_s\}$. This step is of complexity $O(N_s)$.
- From $(\tilde{g}_m)_{1\le m\le N_s}$, compute $\frac{1}{4\pi}\sum_{m=1}^{N_s}e^{i\mathbf{s}_m\cdot\mathbf{x}_i}\tilde{g}_m$. This evaluation can be done in $O(N_s\log(N_s))$ using an inverse (or sometimes called adjoint) Type-III Non-Uniform Fast Fourier Transforms.

For a general (radial) kernel $K(\mathbf{x},\mathbf{y}) = K(|\mathbf{x}-\mathbf{y}|)$, the generalisation of the preceding idea to K consists in decomposing K as a sum of cardinal sine functions. More precisely, we look for two series $(\alpha_n)_{n=1,\cdots,N_K}$ and $(\rho_n)_{n=1,\cdots,N_K}$ in order to approximate K as

$$K(|\mathbf{x}-\mathbf{y}|) \approx \sum_{n=1}^{N_K}\alpha_n\operatorname{sinc}(\rho_n|\mathbf{x}-\mathbf{y}|). \tag{17}$$

Notice that once this approximation is known, one can adapt (15) as

$$\operatorname{sinc}(\rho_n|\mathbf{x}-\mathbf{y}|) \approx \frac{1}{4\pi}\sum_{m=1}^{N_s}\omega_m e^{i\rho_n\mathbf{s}_m\cdot\mathbf{x}}e^{-i\rho_n\mathbf{s}_m\cdot\mathbf{y}}$$

$$= \frac{1}{4\pi} \sum_{m=1}^{N_s'(n)} \omega_m e^{i\mathbf{s}'_{m,n}\cdot\mathbf{x}} e^{-i\mathbf{s}'_{m,n}\cdot\mathbf{y}} \tag{18}$$

with $\mathbf{s}'_{m,n} = \rho_n \mathbf{s}_m$. It is worth noting that the formula becomes a quadrature formula on the dilated sphere $\rho_n S$. The number of points (and the weights), here denoted by $N_s'(n)$, needs to be adapted depending on the required precision and the radius ρ_n of the sphere. For our applications, we use a Gauss–Legendre quadrature on the sphere, although other choices could be, a priori, possible.

Collecting all terms in (17), we obtain

$$K(|\mathbf{x}-\mathbf{y}|) \approx \frac{1}{4\pi} \sum_{n=1}^{N_K} \sum_{m=1}^{N_s'(n)} \alpha_n \omega_m e^{i\mathbf{s}'_{m,n}\cdot\mathbf{x}} e^{-i\mathbf{s}'_{m,n}\cdot\mathbf{y}}, \tag{19}$$

which is of a form similar to (15) and for which a procedure similar to the one given above for sinc can be applied.

The preceding explanations give an overview of the method. Many details have been omitted that are very important for the method to work in practice. In particular, a precise control of the error needs to be done. Besides the error chosen by the user for the NUFFT, the algorithm has two sources of error that have been studied in Alouges and Aussal (2014):

- The error of each of the spherical quadrature (18). This one is easily controlled by the maximum value of $|\mathbf{x}-\mathbf{y}|$. Depending on the clouds of points to which they, respectively belong, we call $R_{\max}$ the maximal distance between $\mathbf{x}$ and $\mathbf{y}$.
- The error in the cardinal sine decomposition (17). This point is more subtle. It turns out that the approximation as a series of cardinal sine functions of traditional kernel (e.g. Helmholtz kernel) is actually more difficult to achieve for *small* values of $|\mathbf{x}-\mathbf{y}|$. We therefore fix a value $R_{\min}$ and look for an approximation that works up to a given precision in the range

$$|\mathbf{x}-\mathbf{y}| \in [R_{\min}, R_{\max}]\,. \tag{20}$$

Since the SCSD quadrature is only valid for far interactions $|\mathbf{x}-\mathbf{y}| \geq R_{\min}$, the close interaction terms ($|\mathbf{x}-\mathbf{y}| \leq R_{\min}$) are inexact and need to be corrected. Fortunately, those terms correspond to short-range interactions and can be efficiently taken into account by assembling a sparse correcting matrix. The performance of the method then relies on choosing the right value for $R_{\min}$ that balances the number of points in the Fourier grid $\mathbf{s}'_{m,n}$ and therefore the time spent in the NUFFT computations with the size of this sparse matrix.

The SCSD method has been applied with success to several problems with radial Green kernels, and not only to the exterior Neumann problem which is the focus of this paper. In particular, we refer the reader to applications of the method to Laplace equation in Alouges and Aussal (2014) and to Stokes equations in Alouges, Aussal, Lefebvre-Lepot, Pigeonneau, and Sellier (2016). For other applications in acoustics, see also Aussal (2014).

4. Implementation details

To test and evaluate the SCSD method, two similar implementations were carried out. A first implementation was done in MATLAB (2013), the *MyBEM* code, and a second implementation in Python (Python Software Foundation, 2016), the *PyBEM* code. In this section, we sketch the structure of both codes and give some implementation details.

The two libraries are divided in several modules that implement the different parts of the finite element method.

(1) Module 1: Mesh handling. The libraries do not encapsulate meshing capabilities. The input of the code must therefore be already constructed meshes, in the form of two arrays, one containing the vertex coordinates and one containing the elements defined from their vertex indices. The module contains all routines to read and import mesh objects from mesh files, and to export the computed solution. Several formats are supported (.ply, .vtk, .msh). More importantly, this module contains the functions to compute useful mesh-related quantities that are used by the code such as element normals, element surfaces and so on.

(2) Module 2: Implementation of numerical quadrature rules, to evaluate boundary integrals on surface triangular meshes. Numerical quadrature is achieved using Gauss–Legendre quadrature rules of several orders. For typical applications, we use a rule of order two, based on three Gauss points per triangle.

(3) Module 3: Finite Elements. The two codes support P^0 and P^1 Lagrange finite elements on surface triangular meshes. Perhaps one originality of our codes lies in the implementation of the finite elements basis functions in the form of sparse matrices. Such matrices allow for the direct evaluation on the quadratures nodes of the basis functions from a vector of DOF values. For instance, let $\mathbf{x} = (x_i)_{i=1}^M$ be the vector containing the M quadrature nodes on the whole mesh. Let also $\mathbf{v} = (v_i)_{i=1}^N$ be the vector containing the N DOF values associated with the N basis functions $(\phi_i)_{i=1}^N$. An element v of the approximation space is then written as

$$v = \sum_{i=1}^{N} v_i \phi_i. \tag{21}$$

The sparse matrix Φ, of size (M, N), allowing for the evaluation of the function $v(\mathbf{x}) = \big(v(x_i)\big)_{i=1}^M$ on the quadrature nodes $\mathbf{x}$ is such that

$$v(\mathbf{x}) = \Phi \mathbf{v}. \tag{22}$$

Further, let $\mathbf{w} = (w_i)_{i=1}^{M}$ denote the vector containing the M quadrature weights associated to the corresponding quadrature nodes $\mathbf{x}$. Then, the evaluation of the boundary integral on Γ of v can then simply computed as $\int_\Gamma v \mathrm{d}\Gamma = \mathbf{w}^T \Phi \mathbf{v}$, where the superscript ^{T} denotes the transpose. Similar sparse matrices corresponding to the evaluation of the basis functions composed with differential operators (divergence, gradient) can also be computed.

(4) Module 4: Classical BEM. This module enables the direct assembly of the dense matrices arising in the classical BEM.
(5) Module 5: implementation of the SCSD method (Alouges & Aussal, 2014), in particular the routines for computing the radial quadrature of the Kernels, the Fourier points and weights and the close interaction correction matrix. The NUFFT (Non-Uniform Fast Fourier Transform) (Dutt & Rokhlin, 1993; Greengard & Lee, 2004; Lee & Greengard, 2005) which is at the core of the SCSD matrix–vector product is not written in MATLAB or Python. We use a Fortran code obtained from L. Greengard's website (http://www.cims.nyu.edu/cmcl/nufft/nufft.html). The NUFFT Fortran routine is interfaced in MATLAB using a Mex file and in Python using the Fortran to Python interface generator, *f2py*.
(6) Module 6: implementation of a $\mathcal{H}$-matrix compression algorithm (Hackbusch, 1999). The routines present in this module can be used in particular for assembling the cluster tree and computing the low rank approximations of the blocks via the Adaptive Cross Approximation algorithm. No $\mathcal{H}$-matrix algebra is yet implemented in the libraries.
(7) Module 7: interface of the FMM routine by L. Greengard (http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html). The Fortran code is wrapped in MATLAB using a Mex file and has been interfaced in *MyBEM* for the sake of comparison with the SCSD. Note that this version of the Fortran routine only computes the point-to-point interactions via the FMM and does not solve the integral equations. This module is not present in *PyBEM*.
(8) Module 8: definition of linear operators. This module allows to compose linear operators from the objects constructed using the BEM, SCSD, $\mathcal{H}$-matrix and FMM modules and defines for each method the matrix–vector product associated. Object-oriented capabilities of MATLAB and Python languages are used to be able to write linear operators as they appear in the equations.
(9) Module 9: regularisation of operators arising from the BEM. This module contains functions used for correcting the numerical errors arising from the numerical quadrature rules in the presence of singularities. This correction rests on semi-analytical integration of singularities and takes the form of a sparse matrix.

Let us now expose how the operators are defined in our libraries. Suppose that we want to evaluate the matrix arising from the discretisation of the single-layer BIO in its variational form $\left[\langle S\phi_i, \phi_j\rangle_\Gamma\right]_{i,j=1}^N$ within a Galerkin framework. Let $\mathbf{G} = \left[G(x_i, x_j)\right]_{i,j=1}^M$ be the matrix of the Green's kernel evaluated at the quadrature nodes $\mathbf{x} = (x_i)_{i=1}^M$. Let also $\Omega = \mathrm{diag}(\mathbf{w})$ be the diagonal matrix constructed with the M quadrature weights, $\mathbf{w} = (w_i)_{i=1}^M$. Then, the Galerkin matrix of the single-layer BIO is numerically computed as

$$\left[\langle S\phi_i, \phi_j\rangle_\Gamma\right]_{i,j=1}^N = \left(\Omega\Phi\right)^T \mathbf{G} \left(\Omega\Phi\right). \tag{23}$$

Note that here $\Omega\Phi$ is a sparse matrix. The methods implemented in the libraries differ on the way the Galerkin matrix is computed or approximated. When using the classical BEM, the matrix $\mathbf{G}$ is assembled directly and the Galerkin matrix is readily obtained by right and left multiplication by $\Omega\Phi$ and its transpose. The $\mathcal{H}$-matrix algorithm provides a (hierarchical) block approximation of the Galerkin matrix using low rank matrices. The matrix vector product is defined by the product of these low rank matrices with the vector of DOFs. The SCSD on the other hand only provides an approximation of the matrix $\mathbf{G}$. When solving a linear system involving the Galerkin matrix, the matrix–vector product is obtained by first applying the sparse matrix $\Omega\Phi$ to the vector of DOFs, then applying the SCSD matrix–vector product approximation (NUFFT, multiplication by the Fourier weights, inverse NUFFT) and finally multiplicating by the transposed matrix $(\Omega\Phi)^T$.

Both implementations rely on an extensive use of vectorised operations to obtain the maximum efficiency. Vectorisation is embedded natively in MATLAB and is achieved in Python using the *numpy* package, together with the corresponding *scipy* package for some linear algebra functions. It is interesting to note that if MATLAB includes some native multi-threading, python programs are single-threaded due to the Global Interpreter Lock (GIL). However, *numpy* and *scipy* functions are outside the scope of the GIL, providing some level of multi-threading. The library *MyBEM* includes in addition multicore parallelisation which is achieved using the Parallel Computing Toolbox.

5. Numerical results

5.1. Validation

We first present a validation test run by the two codes and for which an analytical solution is available. It consists in the acoustic scattering of an incident plane wave $p_{\text{inc}} = e^{-i\mathbf{k}\cdot\mathbf{x}}$, with $\mathbf{k} = (k, 0, 0)$ by the unit sphere in $\mathbb{R}^3$. We plot the magnitude of the total pressure field $p = p_{\text{sca}} + p_{\text{inc}}$ on the boundary of the scatterer and in the near field in Figure 1 for $k = 4$.

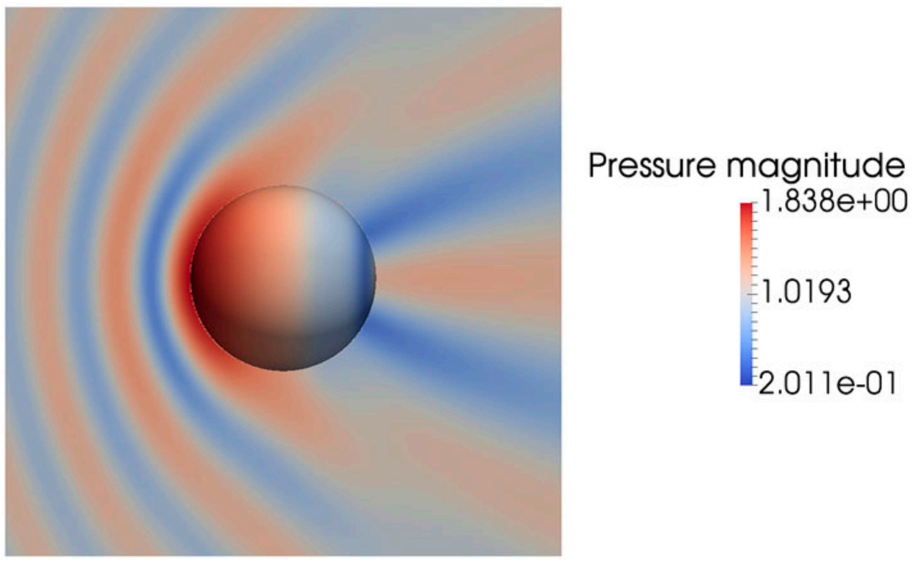

Figure 1. Magnitude of the acoustic pressure $p = p_{\mathrm{sca}} + p_{\mathrm{inc}}$ on the boundary and in the near field due to the scattering of an incident plane wave by the unit sphere with wavenumber $k = 4$.

Another quantity of interest in applications is the far field pattern. In the far field, one can obtain an asymptotic expansion of the integral representation (6). The scattered pressure field p_{sca} reads, for large arguments $|\mathbf{r}|$,

$$p_{\mathrm{sca}}(\mathbf{r}) = \frac{e^{ik|\mathbf{r}|}}{|\mathbf{r}|} p^{\infty}_{\mathrm{sca}}(\mathbf{r}) + \mathcal{O}\left(\frac{1}{|\mathbf{r}|^2}\right), \tag{24}$$

$$p^{\infty}_{\mathrm{sca}}(\mathbf{r}) = \mathcal{S}^{\infty}\lambda(\mathbf{r}) - \mathcal{D}^{\infty}\mu(\mathbf{r}), \tag{25}$$

with

$$\mathcal{S}^{\infty}\lambda(\mathbf{r}) := \frac{1}{4\pi}\int_{\Gamma} e^{-ik\mathbf{r}\cdot\mathbf{y}}\lambda(\mathbf{y})\,\mathrm{d}\Gamma_{\mathbf{y}}, \tag{26}$$

$$\mathcal{D}^{\infty}\mu(\mathbf{r}) := -\frac{ik}{4\pi}\int_{\Gamma} n_y \cdot \mathbf{y} e^{-ik\mathbf{r}\cdot\mathbf{y}}\mu(\mathbf{y})\,\mathrm{d}\Gamma_{\mathbf{y}}, \tag{27}$$

and λ and μ are solutions of the integral Equation (11). We plot in Figure 2 the quantity $|p_{\mathrm{sca}}(\mathbf{r}(\alpha))|^2$ in decibels, representing the far field pattern in 361 evenly distributed directions $\mathbf{r}(\alpha) = (\cos\alpha, \sin\alpha, 0)$, with aspect angle $\alpha \in [-\pi, \pi]$. This is usually referred to as the bistatic Radar Cross Section (RCS). To obtain more quantitative results, we show convergence in the infinity norm of the approximated far field pattern $p^{\infty}_{\mathrm{sca},h}$ towards the reference analytical solution p^{*} as the grid is refined. Note that the dependence of the approximate solutions with respect to the grid size h, which we choose to be the maximum edge size in the considered grids, is here introduced explicitly in the notations. Specifically, the approximate and reference far field solutions are sampled in 36001 uniformly

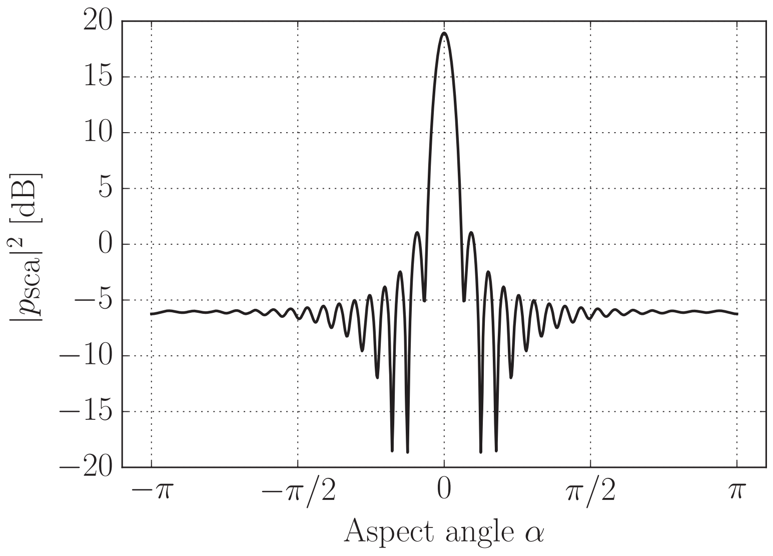

Figure 2. Bistatic RCS of a unit sphere ($k = 20$).

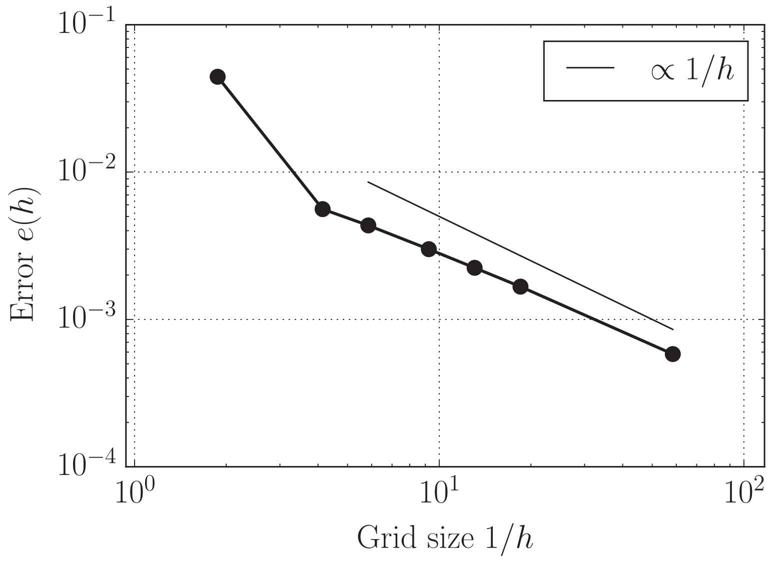

Figure 3. Convergence of the relative error in the infinity norm with respect to the grid size (h) in the computation of the bistatic RCS of a unit sphere ($k = 2$).

distributed directions and we compute the relative error

$$e(h) = \frac{||p^{\infty}_{\mathrm{sca},\,h} - p^{*}||_{\infty}}{||p^{*}||_{\infty}}. \tag{28}$$

The convergence of the error $e(h)$ with respect to h is reported in Figure 3, for a constant wavenumber $k = 2$.

Table 1. Characteristics of the machine used.

Processor	Intel® Xeon™ E5-2667 v3
CPUs Max/Used	16/12
Cache	20 M
Frequency	3.2 GHz
Memory	128 Go

5.2. Performance

We now compare numerically the performance of the SCSD with respect to the classical BEM and other established techniques: $\mathcal{H}$-Matrix compression (Hackbusch, 1999) and FMMs (Greengard, 1988; Greengard & Rokhlin, 1987). For the purpose of this comparison, we use the simple validation test case of the unit sphere as the obstacle. In all the results presented in this section, one run consists in assembling the problem, solving the integral Equation (11) for one incidence and computing the bistatic RCS in 361 evenly distributed directions. The wavenumber k is here set to 2. The number of nodes in the meshes ranges from $N = 10^3$ to $N = 5\ 10^5$. The characteristics of the machine used The MATLAB version is R2013a. The Python version is 2.7.5 with *numpy 1.11.1* and *scipy 0.17.1*. This benchmark is conducted with the fastest versions of the codes available. A maximum of 12 CPUs, out of the 16 available, are used by the pool of workers in MATLAB (Table 1).

The computing times of the different methods with respect to the total number of degrees of freedom are reported in Figure 4. The maximum memory usage of the software during the runs is recorded and plotted with respect to the total number of degrees of freedom. The result is given in Figure 5. Note that because all these figures account for one complete run (therefore including the precomputations, the preconditionning, the iterations and the radiation,) the complexity of the underlying methods used does not appear on the graph presented. These results highlight the good performance of the SCSD method. We remark that it performs similarly compared to the other methods sharing the same complexity, the FMM and H-matrix compression. The more in-depth parallelisation of the MATLAB code explains its better efficiency compared to the Python version. Note that rather than the computing time, it is really the storage requirement of the classical BEM, scaling like $\mathcal{O}\left(N^2\right)$, that puts a limit on the problem size one can achieve with such a method.

5.3. BeTSSi workshop test case

We finally consider the case of a submarine model from the Benchmark Target Strength Simulation (BeTSSi) international workshop (Schneider et al., 2003). In undersea warfare, being able to develop stealthy submarines is a long-standing concern. As such, the goal is to develop efficient predictive tools for evaluating acoustic target strength of submarines. Several approaches are typically used to address underwater acoustic scattering. Due to the difficulty of the problem at

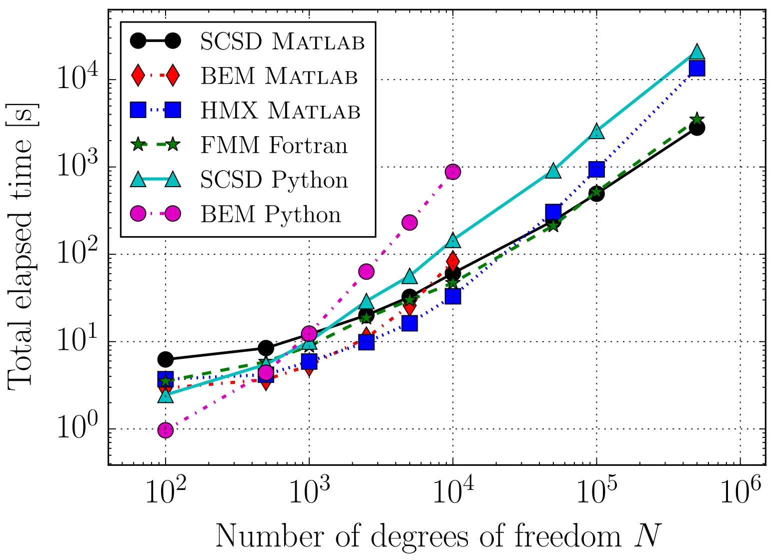

Figure 4. CPU time with respect to the number of degrees of freedom N.
Notes: Results for the SCSD method with *MyBEM* (SCSD MATLAB) and *PyBEM* (SCSD Python), the BEM with *MyBEM* (BEM MATLAB) and *PyBEM* (BEM Python), the $\mathcal{H}$-matrix approach of *MyBEM* (HMX MATLAB) and the FMM from the Fortran routine of Greengard (http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html) interfaced with *MyBEM* (FMM Fortran).

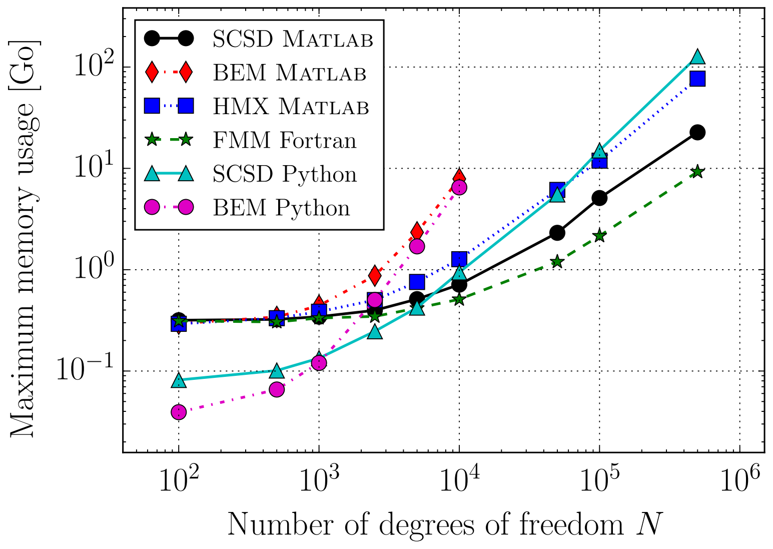

Figure 5. Maximum memory used with respect to the number of degrees of freedom N.
Notes: Results for the SCSD method with *MyBEM* (SCSD MATLAB) and *PyBEM* (SCSD Python), the BEM with *MyBEM* (BEM MATLAB) and *PyBEM* (BEM Python), the $\mathcal{H}$-matrix approach of *MyBEM* (HMX MATLAB) and the FMM from the Fortran routine of Greengard (http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html) interfaced with *MyBEM* (FMM Fortran).

hand and the high frequency regime considered, Kirchhoff diffraction and ray-tracing codes are often used. In contrast to our approach, involving BIEs, such codes only compute an approximation of the scattering by the obstacle. In this context, the precision of BEM is appreciated and BEM solutions may be used as references, at the expense of a more important computational cost.

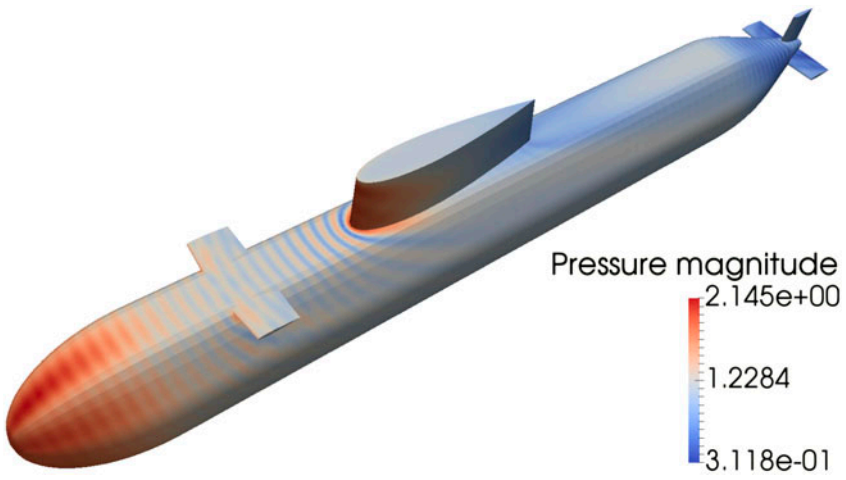

Figure 6. Magnitude of the acoustic pressure $p = p_{\text{sca}} + p_{\text{inc}}$ on the boundary at 1 kHz ($\alpha = 0°$).

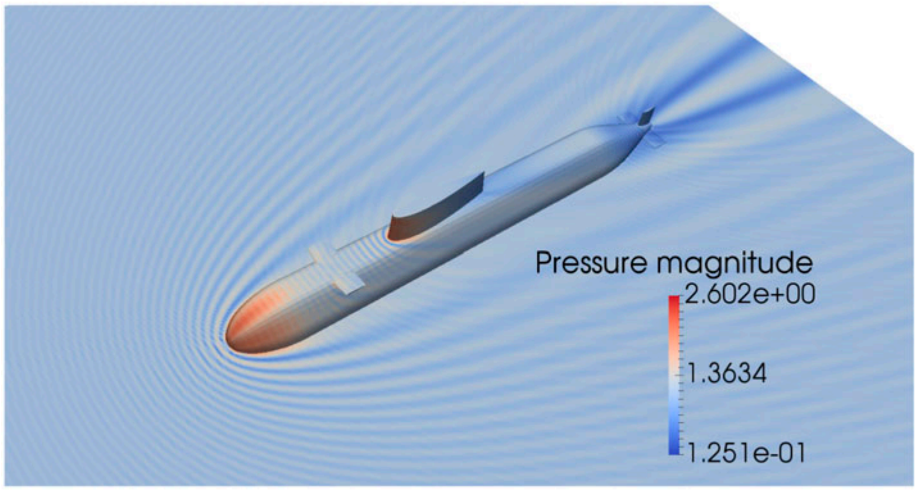

Figure 7. Magnitude of the acoustic pressure $p = p_{\text{sca}} + p_{\text{inc}}$ in the *Oxy* plane at 1 kHz ($\alpha = 0°$).

Several different objects are included in this benchmark. Among the possible choices, we choose a rather complicated shape, a complete submarine model. This is a more challenging test case, close to industrial applications. The mesh used for the computations was provided by ESI group. We consider an incident field in the form of a plane wave, as a model for the signal generated by radar devices. The incident wave then takes the form of $p_{\text{inc}}(\mathbf{x}) = e^{-i\mathbf{k}\cdot\mathbf{x}}$ with $\mathbf{k} = (-\cos\alpha, -\sin\alpha, 0)$ where α is the aspect angle, measured from the x-axis. Although such computations were not required in the benchmark, we provide the acoustic pressure magnitude $p = p_{\text{sca}} + p_{\text{inc}}$ on the boundary of the submarine and in the *Oxy* plane. The representations are given, respectively, in Figure 6 and 7.

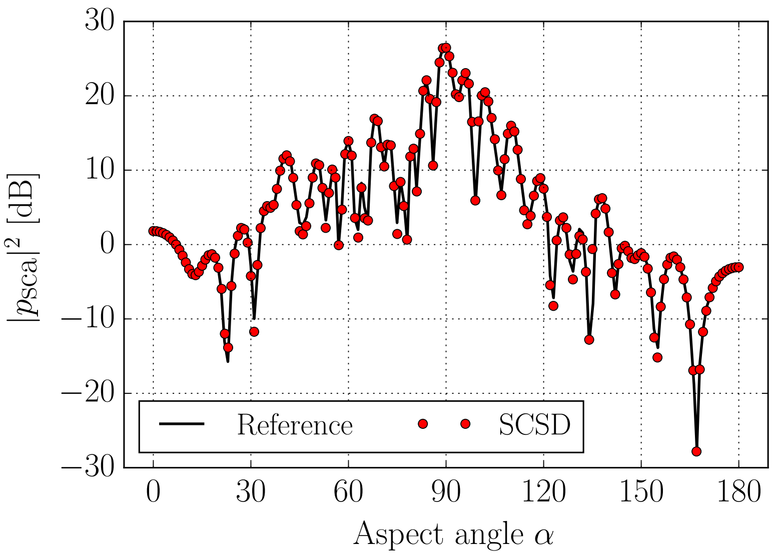

Figure 8. Monostatic target strength $TS(\alpha)$ at 200 Hz (elevation $\beta = 0°$).
Notes: Results obtained with the SCSD (dots) and reference from the BEM code *AVAST* (Nell and Gilroy, 2003) (solid line).

The target strength $TS(\mathbf{r})$ at range $\mathbf{r}$ is the quantity of interest for evaluating the stealth capabilities of submarines. It is defined as

$$TS(\mathbf{r}) = 10 \log \frac{|\mathbf{r} - \mathbf{r}_0|^2 p_{\mathrm{sca}}^2(\mathbf{r})}{p_{\mathrm{inc}}^2(\mathbf{r}_0)}, \tag{29}$$

where $\mathbf{r}_0$ is the geometric centre of the obstacle. The target strength is typically evaluated in the far field, for large $|\mathbf{r}|$, where it become effectively independent of $|\mathbf{r}|$.

The computation of this quantity was the task assigned in the benchmark. Specifically, we need to compute the monostatic target strength $TS(\alpha)$ at 200 Hz evaluated in the far field at elevation angle $\beta = 0°$, with respect to the source and receiver aspect angle α running from 0° to 180° with 1° steps. This represents 181 different integral Equations (11) to solve for, with only the right-hand side (RHS) of the equation changing from one another. To perform this efficiently, we use the Multi-Generalized Conjugate Residual allowing for simultaneous resolution of all incidences (Simon, 2003; Soudais, 1994). In this algorithm, one single new descent direction is computed from a different RHS at each iteration (corresponding to the maximum residual) and is orthonormalised with respect to the previous ones. This is the descent direction that is then used for all the given RHS. This approach implies that only one matrix–vector product is computed at each iteration, therefore saving computational time. The algorithm is not restarted during convergence. We use as preconditioner of the system the sparse local correction matrix arising in the SCSD method, after performing an incomplete LU decomposition. The monostatic target strength is given in

Figure 8 and is in very good agreement with the reference, reported from Nell and Gilroy (2003).

6. Conclusion and future works

We presented a recently developed method for fast numerical convolution in the context of integral equations within BEMs. It is based on a suitable Fourier decomposition of the Green's kernel as a sum of cardinal sine. The final algorithm relies on Non-Uniform Fast Fourier Transforms for fast evaluation and exhibits a similar complexity as other established techniques. The performance of the method allows to tackle real-life size problems. In particular, we computed the monostatic target strength at 200 Hz of the BeTSSi submarine model.

Current and future research work on related subjects includes extension of the SCSD to other physics, e.g. electromagnetism, elastodynamics or Stokes flow. Another research direction focuses on Finite Element Method and BEM coupling. Some implementation efforts are also ongoing, in particular parallelisation on distributed memory systems and on GPUs.

Notes

1. We draw the attention of the reader that a special treatment needs to be done in order to account for the singularity of the Green kernel $G(\mathbf{x}, \mathbf{y})$ when $\mathbf{x} \sim \mathbf{y}$. Indeed, classical quadrature formulae are not accurate enough and one needs to take care of integration points that are close one to another.

Acknowledgements

We gratefully thank Nicolas Zerbib and Lassen Mebarek from ESI Group for having provided us with the BeTSSi submarine mesh.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the French DGA (Direction Générale de l'Armement).

References

Alouges, F., & Aussal, M. (2014). The Sparse Cardinal Sine Decomposition and its application for fast numerical convolution. *Numerical Algorithms*, 1–22.

Alouges, F., Aussal, M., Lefebvre-Lepot, A., Pigeonneau, F., & Sellier, A. (2016, May 22–27). The Sparse Cardinal Sine Decomposition applied to Stokes integral equations. *International Conference on Multiphase Flow*. 9, Florence, Italy.

Aussal, M. (2014). *Méthodes numériques pour la spatialisation sonore, de la simulation à la synthèse binaurale* [Numerical methods for sound spatialisation, from simulation to binaural synthesis] (PhD Thesis). Ecole polytechnique, Palaiseau.

Burton, A. J., & Miller, G. F. (1971). The application of integral equation methods to the numerical solution of some exterior boundary value problems. *Proceedings of the Royal Society of London, Series A, 323*, 201–210.

Dutt, A., & Rokhlin, V. (1993). Fast Fourier transforms for nonequispaced data. *SIAM Journal of Scientific Computing, 14*, 1368–1393.

Greengard, L., & Rokhlin, V. (1987). A fast algorithm for particle simulations. *Journal of Computational Physics, 73*, 325–348.

Greengard, L. (1988). *The rapid evaluation of potential fields in particle systems*. Cambridge: MIT press.

Greengard, L., & Lee, J. Y. (2004). Accelerating the non-uniform Fast Fourier Transform. *SIAM Review, 46*, 443–455.

Hackbusch, W. (1999). A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing, 62*, 89–108.

Lee, J. Y., & Greengard, L. (2005). The type 3 non-uniform FFT and its applications. *Journal of Computational Physics, 206*, 1–7.

Nell, C. W., & Gilroy, L. E. (2003). An improved BASIS model for the BeTSSi submarine. *DRDC Atlantic TR, 199*, 1–20.

Schneider, H.G., Berg, R., Gilroy, L., Karasalo, I., MacGillivray, I., Morshuizen, M.T., & Volker, A. (2003). Acoustic scattering by a submarine: Results from a benchmark target strength simulation workshop. *International Congress on Sound and Vibration*, *10*, 2475–2482.

Simon, J. (2003). Extension des méthodes multipoles rapides: résolution pour des seconds membres multiples et applications aux objets diélectriques, [Extension of fast multipole methods: solving multiple right-hand-sides and applications to dielectric objects] Thèse de l'université de Versailles Saint-Quentin-en-Yvelines (PhD thesis). Université de Versailles Saint-Quentin-en-Yvelines, Versaille.

Soudais, P. (1994). Iterative solution of a 3D scattering problem from arbitrary shaped multi-dielectric and multi-conducting bodies. *IEEE Transactions on Antennas and Propagation, 42*, 954—959.

Matlab. (2013). Version 8.1.0 (R2013a). Natick, MA: The MathWorks. Retrieved from http://www.mathworks.com

Python Software Foundation. (2016). Python language reference (Version 2.7.). Retrieved from http://www.python.org