The Sparse Cardinal Sine Decomposition applied to Stokes integral equations

François Alouges[1], Matthieu Aussal[1], Aline Lefebvre-Lepot[1], Franck Pigeonneau[2] and Antoine Sellier[3]*

[1] *Centre de Mathématiques Appliquées et CNRS*
Ecole polytechnique, route de Saclay, 91128 Palaiseau Cedex, université Paris-Saclay
matthieu.aussal,francois.alouges,aline.lefebvre@polytechnique.edu

[2] *Surface du Verre et Interfaces, UMR 125 CNRS/Saint-Gobain*
39, quai Lucien Lefranc - BP 135, 93303 Aubervilliers cedex, France
franck.pigeonneau@saint-gobain.com

[3] *LadHyX*
Ecole polytechnique, route de Saclay, 91128 Palaiseau Cedex, université Paris-Saclay
sellier@ladhyx.polytechnique.fr

Abstract

Numerical simulations of two-phase flows driven by viscosity (e.g. for bubble motions in glass melting process) rely on the ability to efficiently compute the solutions to discretized Stokes equations. When using boundary element methods to track fluid interfaces, one usually faces the problem of solving linear systems with a dense matrix with a size proportional to the system number of degrees of freedom. Acceleration techniques, based on the compression of the underlying matrix and efficient matrix vector products are known (Fast Multipole Method, $\mathcal{H}$-matrices, etc.) but are usually rather cumbersome to develop. More recently, a new method was proposed, called the "Sparse Cardinal Sine Decomposition", in the context of acoustic problems to tackle this kind of problem in some generality (in particular with respect to the Green kernel of the problem). The proposed contribution aims at showing the potential applicability of the method in the context of viscous flows governed by Stokes equations.

Keywords: Stokes equations, Boundary Element Method, Fast Multipole Method

1. Introduction

In this paper, we consider a fluid flow with negligible inertial effects around a possibly moving solid body Ω in the whole space. Such a flow obeys the Stokes equations

$$\begin{cases} -\mu\Delta\mathbf{u} + \nabla p = 0 \text{ in } \mathbb{R}^3 \setminus \Omega, \\ \text{div}(\mathbf{u}) = 0 \text{ in } \mathbb{R}^3 \setminus \Omega, \end{cases} \tag{1}$$

where μ, $\mathbf{u}$, p respectively designate the fluid viscosity, velocity and pressure. Note that, by essence, (1) is restricted to Newtonian fluids for which the stress tensor Cartesian components read

$$\sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - p\delta_{ij}\,. \tag{2}$$

Boundary conditions must supplement equations (1). We consider in what follows the no-slip boundary condition stating that on the solid body boundary $\partial\Omega$ the fluid velocity is equal to the solid velocity.

Solving the problem of computing the velocity of the fluid and the solid at each point can be done using the integral representation of the velocity field $\mathbf{u}$. Namely, if $\mathbf{n}$ is the unit normal directed to the liquid, the velocity solution to (1) satisfies, for all $\mathbf{x} \in \partial\Omega$ [1] (see e.g. [9] formula (2.3.10))

$$\begin{aligned} u_j(\mathbf{x}) = &-\frac{1}{4\pi\mu}\int_{\partial\Omega}\sum_i (\boldsymbol{\sigma}\cdot\mathbf{n})_i(\mathbf{y})G_{ij}(\mathbf{x}-\mathbf{y})ds(\mathbf{y}) \\ &+\frac{1}{4\pi\mu}\int_{\partial\Omega}^{\mathcal{PV}}\sum_{i,k} u_i(\mathbf{y})T_{ijk}(\mathbf{x}-\mathbf{y})n_k(\mathbf{y})ds(\mathbf{y}) \end{aligned} \tag{3}$$

where the so-called Stokeslet G_{ij} and stresslet T_{ijk} tensors Cartesian components are respectively defined by

$$G_{ij}(\mathbf{x}) = \frac{\delta_{ij}}{|\mathbf{x}|} + \frac{x_i x_j}{|\mathbf{x}|^3}, \quad T_{ijk}(\mathbf{x}) = -6\frac{x_i x_j x_k}{|\mathbf{x}|^5} \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^3$.

After discretization of the above-mentionned operators, e.g. by the finite element method through a Galerkin technique, we obtain the so-called BEM (boundary element method) in which a linear system of equations arises with a dense matrix. Storing the matrix induced by this discretization becomes unfeasible on classical computers when the number of degrees of freedom exceeds a few tens of thousands.

In the literature, several alternatives among which the so-called Fast Multipole Method (FMM) [4, 6] or $\mathcal{H}$-matrices [7, 8] permit to store the matrix in a compressed way and to evaluate the matrix-vector product efficiently, with a complexity $O(N_{dof}\log(N_{dof}))$ where N_{dof} stands for the number of degrees of freedom.

More recently, a new accelerating technique [1] has been proposed that takes into account the fact that the kernels are convolutions operators. The method relies on a suitable sparse integration grid in the Fourier space, and a back and forth non uniform Fast Fourier transform [3, 5]. Originally developed in the context of integral equations for acoustic problems, we hereafter evaluate its potential in the case of Stokes flows.

* The presented research was funded by Saint-Gobain Recherche.

[1] The superscript $\mathcal{PV}$ indicates the principal value of Cauchy.

2. The Sparse Cardinal Sine Decomposition (SCSD)

2.1. Principle

The aim of the SCSD is to decompose the kernel of the convolution operator in a serie of radial sinc functions. Indeed, computing a convolution with the radial sinc function defined by

$$\mathrm{sinc}(r) = \frac{\sin(r)}{r} \tag{5}$$

can be efficiently obtained through the Fourier space. This is due to the fact that the Fourier transform of the (3 dimensional) cardinal sine function is given by

$$\mathcal{F}(\mathrm{sinc}) = \frac{1}{(4\pi)^2}\delta_{\mathbb{S}^2} \tag{6}$$

where $\delta_{\mathbb{S}^2}$ stands for the Dirac mass on the unit sphere $\mathbb{S}^2$ of $\mathbb{R}^3$. Thus, if we consider the function

$$g(\mathbf{x}) = \int_{\partial\Omega} \mathrm{sinc}(|\mathbf{x}-\mathbf{y}|) f(\mathbf{y})\, d\mathbf{y}\,, \tag{7}$$

we can evaluate g at each point $\mathbf{x} \in \mathbb{R}^3$ as follows

$$\begin{aligned} g(\mathbf{x}) &= \int_{\xi\in\mathbb{S}^2}\int_{\mathbf{y}\in\partial\Omega} \exp(i(\mathbf{x}-\mathbf{y})\cdot\xi) f(\mathbf{y})\, d\mathbf{y}\, d\xi \\ &= \int_{\xi\in\mathbb{S}^2} \exp(i\mathbf{x}\cdot\xi)\left(\int_{\mathbf{y}\in\partial\Omega}\exp(-i\mathbf{y}\cdot\xi) f(\mathbf{y})\, d\mathbf{y}\right) d\xi\,. \end{aligned} \tag{8}$$

After discretization (of both $\partial\Omega$ and $\mathbb{S}^2$) the previous integrals become discrete sums and we can evaluate g at the point $(\mathbf{x}_k)$ by

$$g(\mathbf{x}_k) = \sum_{l=1}^{N_\xi} \omega_k^\xi \exp(i\mathbf{x}_k\cdot\xi_l)\left(\sum_{m=1}^{N_\mathbf{y}} \exp(-i\mathbf{y}_m\cdot\xi_l)\omega_m^\mathbf{y} f(\mathbf{y}_m)\right) \tag{9}$$

where $N_\mathbf{y}$ and N_ξ respectively stand for the number of integration points in the real and Fourier space, while $(\omega_m^\mathbf{y}, \mathbf{y}_m)_{1\le m\le N_\mathbf{y}}$ (resp. $(\omega_l^\xi, \xi_l)_{1\le m\le N_\xi}$) indicate the integration formula on $\partial\Omega$ (resp. $\mathbb{S}^2$ in the Fourier space), namely for a smooth enough function h defined on $\partial\Omega$

$$\int_{\partial\Omega} h(\mathbf{y})\, d\mathbf{y} \sim \sum_{m=1}^{N_\mathbf{y}} \omega_m^\mathbf{y} h(\mathbf{y}_m)\,. \tag{10}$$

The implemented algorithm for the convolution appeals to the following steps :

- Evaluate f at the integration points $(\mathbf{y}_m)_m$, and multiply by the weights $(\omega_m^\mathbf{y})_m$.
- Compute the Fourier transform on the unit sphere at the (integration) points $(\xi_l)_l$

$$\hat{f}_l = \sum_m \exp(-i\mathbf{y}_m\cdot\xi_l)\omega_m^\mathbf{y} f(\mathbf{y}_m). \tag{11}$$

- Multiply by the weights $(\omega_l^\xi)_l$ and proceed to the inverse Fourier transform

$$g(\mathbf{x}_k) \sim \sum_l \exp(i\mathbf{x}_k\cdot\xi_l)\omega_l^\xi \hat{f}_l\,. \tag{12}$$

Formulas (11) and (12) can be efficiently computed using the non-uniform FFT of type 3 [5] ensuring the global complexity of the algorithm to be $O((N_\xi + N_\mathbf{y})\log(N_\xi + N_\mathbf{y}))$.

Extending the concept to more general kernels $K(r)$ than the sinc is done by decomposing the (assumed to be radial) kernel K as a finite sum of sinc functions as

$$K(r) = \sum_i \alpha_i \frac{\sin(\lambda_i r)}{r}\,. \tag{13}$$

In [1] is proposed a way to effectively compute both $(\lambda_i)_i$ and the weights $(\alpha_i)_i$ for classical kernels (e.g. the Laplace kernel $1/4\pi r$ or the Helmholtz kernel $\exp(ikr)/4\pi r$) in order that formula (13) is precise to any given tolerance ϵ in a range $r \in [R_{\min}, R_{\max}]$. Local interactions (those for which $|\mathbf{x}-\mathbf{y}| < R_{\min}$ are taken into account by adding to the discretized operator a local corrective sparse matrix.

It is then shown that the sum (13) can be computed at once by considering a suitable integration grid in the Fourier space which is assembled by collecting the ones of each $sinc$ functions that appear in the formula together. The global convolution obeys the same algorithm than before, except for the Fourier grid which ressorts to more points.

Nevertheless, the Stokes kernels $\mathbf{G}$ and $\mathbf{T}$ being not radial (see (4)), we need to describe new ideas to come up with a suitable decomposition that would generalize the method. We explain hereafter the strategy that we have used.

2.2. Application to Stokes kernels

As shown in (4), we split the Oseen tensor $\mathbf{G}$ into two parts as $\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2$, where $\mathbf{G}_1$ is radial and corresponds to the vectorial Laplacian, while $\mathbf{G}_2$ writes

$$\mathbf{G}_2(\mathbf{x}) = \frac{\mathbf{x}\otimes\mathbf{x}}{|\mathbf{x}|^3}\,. \tag{14}$$

We first apply the methodology described in [1] to the kernel G_1. This provides us with a set of integration weights and points $(\omega_j^\xi, \xi_j)_{1\le j\le N_\xi}$ and a formula

$$\frac{1}{|\mathbf{x}|} \sim \sum_{j=1}^{N_\xi} \omega_j^\xi \exp(i\mathbf{x}\cdot\xi_j) \tag{15}$$

which is valid[2] for all $\mathbf{x}$ such that $|\mathbf{x}| \in [R_{\min}, R_{\max}]$. Differentiating (13) leads to

$$\frac{\mathbf{x}}{|\mathbf{x}|^3} \sim \sum_{j=1}^{N_\xi} -i\xi_j\omega_j^\xi \exp(i\mathbf{x}\cdot\xi_j)\,. \tag{16}$$

Thus, if we consider a vector field $\mathbf{f}$ defined on $\partial\Omega$ and $\mathbf{g}$ the convolution of $\mathbf{f}$ with G_2, we can write componentwise, for $i = 1, 2, 3$

$$\begin{aligned} g_i(\mathbf{x}) &:= \int_{\partial\Omega}\sum_{j=1}^3 G_{2,ij}(\mathbf{x}-\mathbf{y}) f_j(\mathbf{y})\, d\mathbf{y} \\ &= \int_{\partial\Omega}\sum_{j=1}^3 \frac{(x_i-y_i)}{|\mathbf{x}-\mathbf{y}|^3}(x_j-y_j) f_j(\mathbf{y})\, d\mathbf{y} \\ &= \sum_{j=1}^3 \mathbf{x}_j \int_{\partial\Omega} \frac{(x_i-y_i)}{|\mathbf{x}-\mathbf{y}|^3} f_j(\mathbf{y})\, d\mathbf{y} \\ &\qquad - \sum_{j=1}^3 \int_{\partial\Omega} \frac{(x_i-y_i)}{|\mathbf{x}-\mathbf{y}|^3} y_j f_j(\mathbf{y})\, d\mathbf{y}\,, \end{aligned} \tag{17}$$

[2] The choice of $R_{\min}$ and $R_{\max}$ is explained in [1].

and each term can be computed using the fast algorithm described before together with the decomposition (16).

Similarly, as far as the term involving the stress tensor $\mathbf{T}$ is concerned, we use the identity

$$\begin{aligned}\frac{x_i x_j x_k n_k}{|\mathbf{x}|^5} &= \frac{1}{3}\left(x_k n_k \frac{\partial^2}{\partial x_i \partial x_j}\frac{1}{|\mathbf{x}|} + x_k n_k \frac{\delta_{ij}}{|\mathbf{x}|^3}\right) \\ &= \frac{1}{3}\left(x_k n_k \frac{\partial^2}{\partial x_i \partial x_j}\frac{1}{|\mathbf{x}|} - \delta_{ij} n_k \frac{\partial}{\partial x_k}\frac{1}{|\mathbf{x}|}\right) \quad (18)\end{aligned}$$

to deduce decomposition of the term appearing in (3) as

$$\begin{aligned}v_j(\mathbf{x}) &= \int_{\partial\Omega} \sum_{i,k} T_{ijk}(\mathbf{x}-\mathbf{y}) n_k(\mathbf{y}) u_i(\mathbf{y})\, d\mathbf{y} \\ &= -2\sum_{i,k} x_k \int_{\partial\Omega}\left(\frac{\partial^2}{\partial x_i \partial x_j}\frac{1}{|\mathbf{x}|}\right)(\mathbf{x}-\mathbf{y}) n_k(\mathbf{y}) u_i(\mathbf{y})\, d\mathbf{y} \\ &\quad +2\sum_{i,k}\int_{\partial\Omega}\left(\frac{\partial^2}{\partial x_i \partial x_j}\frac{1}{|\mathbf{x}|}\right)(\mathbf{x}-\mathbf{y}) n_k(\mathbf{y}) y_k u_i(\mathbf{y})\, d\mathbf{y} \\ &\quad +2\sum_{k}\int_{\partial\Omega}\left(\frac{\partial}{\partial x_k}\frac{1}{|\mathbf{x}|}\right)(\mathbf{x}-\mathbf{y}) n_k(\mathbf{y}) u_j(\mathbf{y})\, d\mathbf{y}\,. \quad (19)\end{aligned}$$

Each term in the right-hand side is eventually treated separately as before.

3. Numerical validation

In the following numerical tests, the solid body is an ellipsoid with surface $\partial\Omega$ defined by

$$\frac{x_1^2}{a_1^2} + \frac{x_2^2}{a_2^2} + \frac{x_3^2}{a_3^2} = 1, \quad \forall \mathbf{x} \in \partial\Omega \quad (20)$$

with $a_1 = 5$, $a_2 = 3$ and $a_3 = 2$. The surface is approximated as $\partial\Omega_h$ using flat triangular boundary elements. The normal to $\partial\Omega$ is approximated by the normal to $\partial\Omega_h$. We plot in Figure 1 an example of mesh approximating $\partial\Omega$. We use P^1 boundary elements then, if N is the number of vertex of the mesh, we have $N_{dof} = 3N$. The computations are run using a parallel MATLAB code.

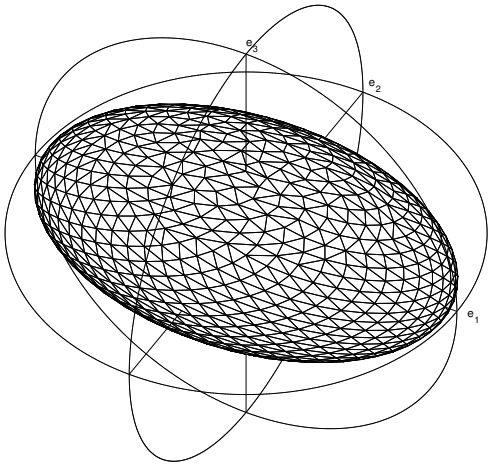

Figure 1: Mesh of the ellipsoidal domain Ω for $N = 10^3$ vertices.

Several tests of the SCSD method were performed. For each test, the convergence of the method as well as the CPU time when the number of degrees of freedom N increases is examined. We compare the results against the ones obtained using a full BEM method. To do so, we introduce the size of the mesh as $h := (h_{\min} + h_{\max})/2$ where $h_{\min}$ (resp. $h_{\max}$) is the length of the smaller (resp. larger) edge of the mesh. Tests are run for N between 200 and 5000 for BEM and between 200 and 50000 for SCSD.

3.1. The Stokeslet

We define the vector field $\mathbf{Q}_1$ on $\partial\Omega$ by, $\forall \mathbf{x} \in \partial\Omega$

$$(Q_1)_j(\mathbf{x}) := \int_{\partial\Omega} \sum_i G_{ij}(\mathbf{x}-\mathbf{y}) n_i(\mathbf{y}) d\mathbf{y} \quad (21)$$

where $\mathbf{n}(\mathbf{y})$ is the outer normal to $\partial\Omega$ at point $\mathbf{y}$. It is known that (see e.g. [9] formula (2.1.4)):

$$\forall \mathbf{x} \in \partial\Omega, \quad \mathbf{Q}_1 = 0. \quad (22)$$

Then, we examine the behavior when N increases of the $L^2(\partial\Omega)$ numerical error

$$err_1 := \sqrt{\int_{\partial\Omega} |\mathbf{Q}_1(\mathbf{x})|^2 d\mathbf{x}}. \quad (23)$$

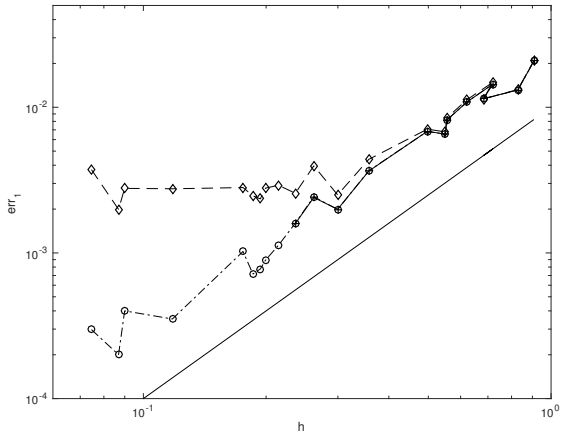

Figure 2: Test for the Stokeslet : err_1 versus h. BEM (–+–) and SCSD for $\epsilon = 10^{-3}$ (– –◇– –) and $\epsilon = 10^{-4}$ (– · –○– · –). The unit slope is also given as a straight line (—).

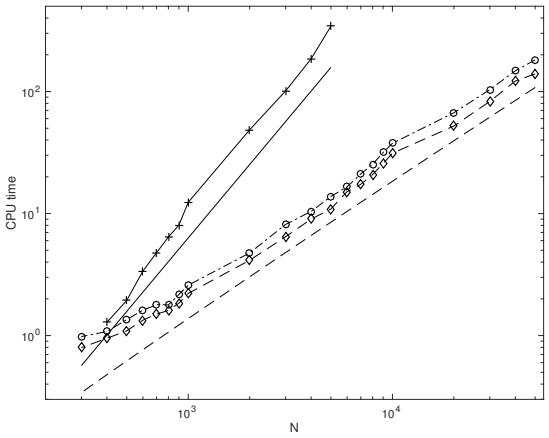

Figure 3: Test for the Stokeslet : CPU time versus N. BEM (– +–) and SCSD for $\epsilon = 10^{-3}$ (– –◇– –) and $\epsilon = 10^{-4}$ (– · –○– · –). $N\log(N)$ (– –) and N^2 (—) are also displayed.

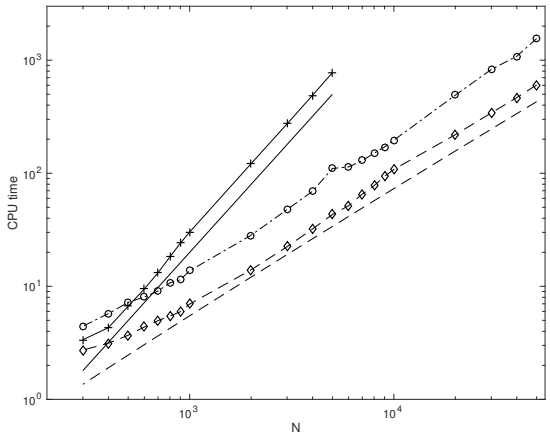

Figure 4: Test for the stresslet : CPU time versus N. BEM (–+–) and SCSD for $\epsilon = 10^{-3}$ (– –◇– –) and $\epsilon = 10^{-4}$ (– · –○– · –). $N\log(N)$ (– –) and N^2 (—) are also displayed.

The computed error is seen to be of order h (see Figure 2). When the mesh size h goes to zero, a saturation is observed, due to the tolerance ϵ used in the SCSD algorithm. The CPU time behaves as $O(N)$ (see Figure 3). We see that when the tolerance is sufficiently small, the SCSD is as precise as the BEM. However, as expected, since BEM has a $O(N^2)$ complexity, computations can not be performed using this method when N becomes too large.

3.2. The stresslet

A similar test is run for the stresslet. Let $\mathbf{w} = \mathbf{e}_1 \times \mathbf{x}$ be the rotation around $\mathbf{e}_1 = (1,0,0)$. We define the vector field $\mathbf{Q}_2$ on $\partial\Omega$ by, $\forall \mathbf{x} \in \partial\Omega$,

$$(Q_2)_j(\mathbf{x}) := \frac{1}{8\pi}\int_{\partial\Omega}\sum_{i,k} w_i(\mathbf{y})(\mathbf{y})T_{ijk}(\mathbf{x}-\mathbf{y})n_k(\mathbf{y})ds(\mathbf{y}). \tag{24}$$

We have (see e.g. [9] formula (2.1.13)):

$$\forall \mathbf{x} \in \partial\Omega, \quad \mathbf{Q}_2(\mathbf{x}) = -\frac{\mathbf{w}(\mathbf{x})}{2}, \tag{25}$$

and we look at the corresponding error :

$$err_2 := \sqrt{\int_{\partial\Omega}\left|\mathbf{Q}_2(\mathbf{x}) + \frac{\mathbf{w}(\mathbf{x})}{2}\right|^2 d\mathbf{x}}. \tag{26}$$

Due to the correction for T, err_2 is almost constant versus N and is close to the precision ϵ chosen for the SCSD method and close to 10^{-14} for BEM. Again the CPU time, it is of order $O(N\log(N))$ for SCSD, $O(N^2)$ for BEM (see Figure 4).

3.3. Integral representation

Let us now consider the fundamental solution $\mathbf{u}_0$ to the Stokes problem in $\mathbb{R}^3$, with the Dirac source term $\mathbf{f} = \delta_{\mathbf{x}_0}\mathbf{e}_1$ where $\mathbf{x}_0 \in \Omega$. We have

$$u_{0,j}(\mathbf{x}) = \frac{1}{8\pi}G_{1j}(\mathbf{x}-\mathbf{x}_0) \tag{27}$$

and

$$(\boldsymbol{\sigma}_0 \cdot \mathbf{n})_j(\mathbf{x}) = \frac{1}{8\pi}\sum_k T_{1jk}(\mathbf{x}-\mathbf{x}_0)n_k(\mathbf{x}). \tag{28}$$

Since $\mathbf{u}_0$ is solution to (1) in $\mathbb{R}^3 \setminus \Omega$, it can also be expressed on $\partial\Omega$ using the integral representation (3).

We compute this integral representation using SCSD and BEM and compare the convergence of the methods and the corresponding CPU times. To study the convergence, we define err_3 as the $L^2(\partial\Omega)$ norm between $\mathbf{u}_0$ and its integral representation. In order to avoid symmetry effects, we choose $\mathbf{x}_0 = (1,2,0.5)$ which is inside the ellipsoid.

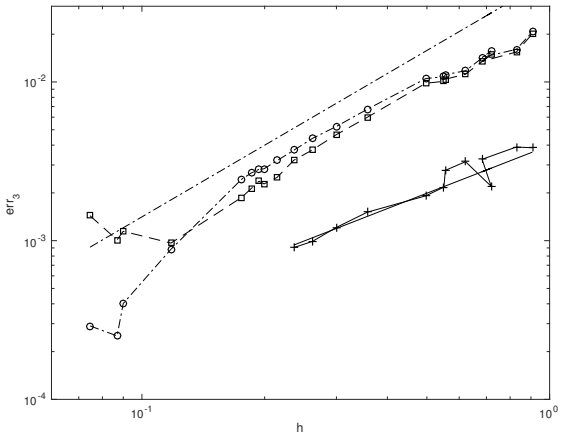

Figure 5: Test for the integral representation : err_3 versus h. BEM (–+–) and SCSD for $\epsilon = 10^{-2}$ (– –□– –) and $\epsilon = 10^{-4}$ (– · –○– · –). Lines of slope 1 (—) and 1.5 (– · –) are also displayed.

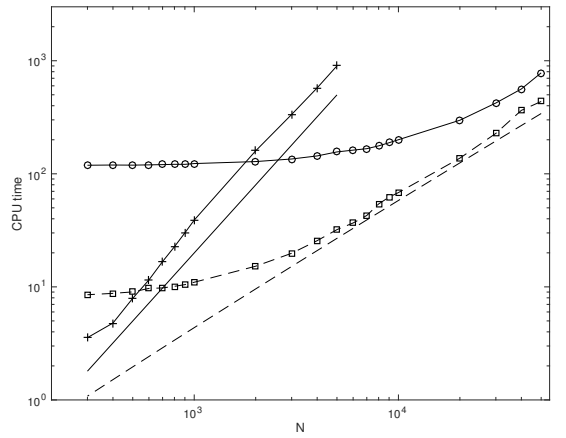

Figure 6: Test for the integral representation : CPU time versus N. BEM (–+–) and SCSD for $\epsilon = 10^{-2}$ (– –□– –) and $\epsilon = 10^{-4}$ (– · –○– · –). $N\log(N)$ (– –) and N^2 (—) are also displayed.

The associated plots of err_3 and the CPU time are given in Figures 5 and 6. Observe for err_3 a saturation due to the precision parameter for SCSD while CPU time behaves as $N\log(N)$.

3.4. The Dirichlet to Neumann problem

Let us now consider the Dirichlet to Neumann problem: we suppose that $\mathbf{u}_0$ is known and we compute the corresponding traction $\boldsymbol{\sigma}_0 \cdot \mathbf{n}$ by solving (3). From (21), we see that $\boldsymbol{\sigma}_0 \cdot \mathbf{n}$ is computed up to a multiple of $\mathbf{n}$. If we denote by $(\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num}$ the numerical field, we compute the constant λ minimizing the $L^2(\partial\Omega)$ norm between $\boldsymbol{\sigma}_0 \cdot \mathbf{n}$ and $(\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num} + \lambda\mathbf{n}$. Then, the numerical error is defined as

$$err_4 = \sqrt{\int_{\partial\Omega} |(\boldsymbol{\sigma}_0 \cdot \mathbf{n})(\mathbf{x}) - ((\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num}(\mathbf{x}) + \lambda\mathbf{n}(\mathbf{x}))\,|^2 d\mathbf{x}}$$

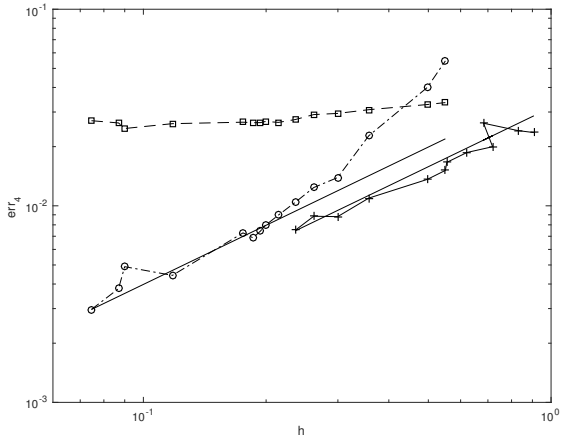

Figure 7: Test for the Dirichlet to Neumann problem : err_4 versus h. BEM (–+–) and SCSD for $\epsilon = 10^{-2}$ (– –□– –) and $\epsilon = 10^{-4}$ (– · –○– · –).

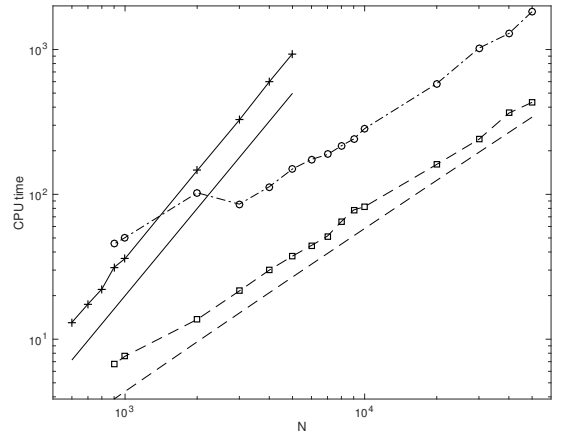

Figure 8: Test for the Dirichlet to Neumann problem : CPU time versus N. BEM (–+–) and SCSD for $\epsilon = 10^{-2}$ (– –□– –) and $\epsilon = 10^{-4}$ (– · –○– · –). $N\log(N)$ (– –) and N^2 (—) are also displayed.

Again, observe in Figures 7 and 8 the convergence of the method, a saturation of the error depending of the parameter chosen for the SCSD algorithm and $O(N\log N)$ behavior of CPU time.

4. Conclusion

The proposed approach has been nicely tested for the case of one solid body (as regards for the error and the CPU time). At the oral presentation, additional results for two interacting particles (spheres or ellipsoids) will be also presented and discussed.

References

[1] Alouges, F. and Aussal, M. *The sparse cardinal sine decomposition and its application for fast numerical convolution*, Numerical Algorithms, 1-22, 2015.

[2] Batchelor, G.K., *An Introduction to Fluid Dynamics*, first ed., Cambridge Mathematical Library, Cambridge, 1967.

[3] Dutt, A. et Rokhlin, V.: *Fast Fourier transforms for nonequispaced data.* SIAM J. Sci. Comput. 14, 1993.

[4] L. Greengard and V. Rokhlin, *The Rapid Evaluation of Potential Fields in Three Dimensions*, Lecture Notes in Mathematics, Springer Verlag, 1988.

[5] Lee, J.-Y. et Greengard, L.: *The type 3 nonuniform fft and its application.* J. Comput. Phys., 206, 1–5, 2005.

[6] Greengard, L.: *The rapid evaluation of potential fields in particle systems*. MIT Press, 1988.

[7] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. Part I. Introduction to H-matrices*, Computing, 1999.

[8] Hackbusch, W.: *Hierarchische Matrizen*. Springer, 2009.

[9] Pozrikidis, C., *Boundary Integral and Singularity Methods for Linearized Viscous Flow*, Cambridge University Press, Cambridge, 1992.