APPLICATION OF THE SPARSE CARDINAL SINE DECOMPOSITION TO 3D STOKES FLOWS

F. ALOUGES[1], M. AUSSAL[1], A. LEFEBVRE-LEPOT[1], F. PIGEONNEAU[2] & A. SELLIER[3]
[1]Centre de Mathématiques Appliquées, Ecole polytechnique, France
[2]Surface du Verre et Interfaces, Saint-Gobain, France
[3]LadHyX, Ecole polytechnique, France

ABSTRACT

In boundary element method (BEM), one encounters linear system with a dense and non-symmetric square matrix which might be so large that inverting the linear system is too prohibitive in terms of cpu time and/or memory. Each usual powerful treatment (Fast Multipole Method, H-matrices) developed to deal with this issue is optimized to efficiently perform matrix vector products. This work presents a new technique to adequately and quickly handle such products: the Sparse Cardinal Sine Decomposition. This approach, recently pioneered for the Laplace and Helmholtz equations, rests on the decomposition of each encountered kernel as series of radial Cardinal Sine functions. Here, we achieve this decomposition for the Stokes problem and implement it in MyBEM, a new fast solver for multi-physical BEM. The reported computational examples permit us to compare the advocated method against a usual BEM in terms of both accuracy and convergence.

Keywords: boundary element method, fast convolution, Stokes equations.

1 INTRODUCTION

We consider a flow around a body Ω of a Newtonian and unbounded liquid with uniform viscosity μ. Adopting henceforth the usual tensor summation notation, the Newtonian liquid has pressure p, velocity $\mathbf{u} = u_i \mathbf{e}_i$ and stress tensor $\boldsymbol{\sigma} = \sigma_{ij} \mathbf{e}_i \otimes \mathbf{e}_j$ such that, in the entire liquid domain $D = \mathbb{R}^3 \setminus \Omega$,

$$\sigma_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{1}$$

with δ the usual Kronecker delta. Neglecting inertial effects, the creeping flow $(\mathbf{u}, p)$ satisfies the following Stokes equations and far-field behavior

$$\mu\Delta\mathbf{u} = \nabla p \text{ and } \operatorname{div}(\mathbf{u}) = 0 \text{ in } D, \tag{2}$$

$$(\mathbf{u}, p) \to (\mathbf{0}, 0) \text{ far from } \Omega \tag{3}$$

One has to supplement eqns (2) and (3) with conditions on the smooth surface $\partial\Omega$ having unit normal $\mathbf{n}$ directed *into* the liquid. Those conditions depend on the nature of $\partial\Omega$: no-slip or slipping solid surface, flexible surface,...

Each component $u_j = \mathbf{u}.\mathbf{e}_j$ admits [1] in D a key integral representation involving the Oseen velocity tensor $\mathbf{G} = G_{ij}(\mathbf{X})\mathbf{e}_i \otimes \mathbf{e}_j$ and stress tensor $\mathbf{T} = T_{ijk}(\mathbf{X})\mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k$ defined, for $\mathbf{X} = X_i\mathbf{e}_i \neq \mathbf{0}$, as

$$G_{ij}(\mathbf{X}) = \delta_{ij} / |\mathbf{X}| + X_i X_j / |\mathbf{X}|^3, T_{ijk}(\mathbf{X}) = -6X_i X_j X_k / |\mathbf{X}|^5 \tag{4}$$

ISSN: 2046-0546 (paper format), ISSN: 2046-0554 (online), http://www.witpress.com/journals
DOI: 10.2495/CMEM-V5-N3-387-394

Noting $dS = dS(\mathbf{y})$, a possible *regularized* form of this representation is

$$\begin{aligned} u_j(\mathbf{x}) = &-\frac{1}{8\pi\mu}\int_{\partial\Omega}[\sigma_{ik}n_k](\mathbf{y})G_{ij}(\mathbf{y}-\mathbf{x})dS \\ &+\frac{1}{8\pi}\int_{\partial\Omega}[u_i(\mathbf{y})-u_i(\mathbf{x})]n_k(\mathbf{y})T_{ijk}(\mathbf{y}-\mathbf{x})dS,\ \mathbf{x}\in D\cup\partial\Omega. \end{aligned} \tag{5}$$

Inspecting eqn (5) shows that when looking at **u** in the entire liquid domain D it is sufficient to gain on the body boundary $\partial\Omega$ the velocity **u** and the surface traction $\boldsymbol{\sigma}.\mathbf{n}$. In practice, those two key surface quantities are obtained by injecting the boundary conditions prescribed on $\partial\Omega$ either in eqn (5) for **x** on $\partial\Omega$ or in the following equivalent relation

$$\begin{aligned} 8\pi u_j(\mathbf{x}) = &-u_i(\mathbf{x})\int_{\partial\Omega}^{PV} n_k(\mathbf{y})T_{ijk}(\mathbf{y}-\mathbf{x})dS \\ &+\int_{\partial\Omega}^{PV}[u_i n_k](\mathbf{y})T_{ijk}(\mathbf{y}-\mathbf{x})dS-\int_{\partial\Omega}\left[\frac{\sigma_{ik}n_k}{\mu}\right](\mathbf{y})G_{ij}(\mathbf{y}-\mathbf{x})dS,\ \mathbf{x}\in\partial\Omega \end{aligned} \tag{6}$$

where the superscript *PV* indicates the principal value of the integral.

In practice, the discretization of eqn (5) on $\partial\Omega$ or eqn (6) is done by a collocation or Galerkin BEM (boundary element method) which approximates $\partial\Omega$ with boundary elements. On those elements one locates N nodal points at which three Cartesian components amongst u_i and $\mathbf{e}_i.\boldsymbol{\sigma}.\mathbf{n}$ are unknown. Denoting by $N_d = 3N$, the number of degrees of freedom, one arrives at a linear system $\mathbf{A}.\mathbf{v} = \mathbf{b}$ with N_d–unknown vector **v** and dense and non-symmetric square $N_d \times N_d$ matrix **A**. For N_d typically less than 10000 one can use a LU factorization. For N_d larger one resorts to a generalized minimal iterative residual method (such as GMRS), which reduces the task to the evaluation of products $\mathbf{A}.\mathbf{q}$ for many vectors **q**. This can be efficiently done by storing only a relevant approximation of **A** by so-called Fast Multipole Method [2, 3] or H-matrices approach [4, 5]. Employing those methods each matrix-vector product $\mathbf{A}.\mathbf{q}$ is approximated in only $O(N_d \log N_d)$ operations.

Recently, a new accelerating technique has been proposed in Alouges and Aussal [6] for the boundary-integral equations encountered in potential and Helmholtz problems. This method appeals to a suitable sparse integration grid in the Fourier space, and a back and forth non-uniform Fast Fourier transform [7, 8]. This work extends the procedure to the boundary-integral eqn (6) obtained for the Stokes problem and investigates its abilities in terms of both error and cpu time.

2 THE SPARSE CARDINAL SINE DECOMPOSITION (SCSD)

2.1 Principle for a radial kernel

For the cardinal sine function sinc defined by $\mathrm{sinc}(t) = \sin(t)/t$, we look at evaluating the convolution operator g with radial kernel sinc defined as

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

This can be done calculating the usual three-dimensional Fourier transform $F(g)$ of g and then operating the inverse Fourier transform with

$$g(\mathbf{x}) = \frac{1}{(2\pi)^3}\int_{\mathbb{R}^3} e^{i\mathbf{x}\cdot\boldsymbol{\xi}}F(\mathrm{g})(\boldsymbol{\xi})d\boldsymbol{\xi}. \tag{8}$$

Actually, noticing that $F(\text{sinc}) = 2\pi^2\delta_{\mathbb{S}}2$, with $\delta_{\mathbb{S}}2$ the Dirac mass on the unit sphere $\mathbb{S}^2$ of $\mathbb{R}^3$, it turns out that $F(g)$ is a radial function given by $F(g)(\xi) = 2\pi^2 F(f)(\xi)\delta_{\mathbb{S}}2(\xi)$ where $\xi = |\boldsymbol{\xi}|$. Accordingly, eqn (8) becomes

$$\begin{aligned} g(\mathbf{x}) &= \frac{1}{4\pi}\int_{\boldsymbol{\xi}\in\mathbb{S}^2}\int_{\mathrm{y}\in\partial\Omega}\exp(i(\mathbf{x}-\mathbf{y})\cdot\boldsymbol{\xi})f(\mathbf{y})d\mathbf{y}\; d\boldsymbol{\xi} \\ &= \frac{1}{4\pi}\int_{\boldsymbol{\xi}\in\mathbb{S}^2} e^{i\mathbf{x}\cdot\boldsymbol{\xi}}\left(\int_{\mathbf{y}\in\partial\Omega} e^{-i\mathbf{y}\cdot\boldsymbol{\xi}} f(\mathbf{y})d\mathbf{y}\right)d\boldsymbol{\xi}. \end{aligned} \tag{9}$$

In eqn (9), the integration over $\mathbb{S}^2$ (or $\partial\Omega$) is evaluated by a numerical quadrature with N_{ξ} (or N_{y}) points $\boldsymbol{\xi}_l$ (or $\mathbf{y}_{\boldsymbol{m}}$) with associated weight ω_l^{ξ} (or ω_m^{y}). Thus, at point $\mathbf{x}_k$ on $\partial\Omega$ one finally gets the approximation

$$g(\mathbf{x}_k) \sim \sum_{l=1}^{N\xi}\omega_l^{\xi}\exp(i\mathbf{x}_k\cdot\boldsymbol{\xi}_l)\left(\sum_{m=1}^{N_y}\exp(-i\mathbf{y}_m\cdot\boldsymbol{\xi}_l)\omega_m^{\mathrm{y}}f(\mathbf{y}_m)\right). \tag{10}$$

Once f has been computed at points $(\mathbf{y}_{\boldsymbol{m}})_m$ of $\partial\Omega$, we then successively calculate the following discrete Fourier and inverse Fourier transforms

$$\hat{f}_l = \sum_{l=1}^{N_{\mathrm{y}}}\exp(-i\mathbf{y}_m\cdot\boldsymbol{\xi}_i)\omega_m^{\mathrm{y}}f(\mathbf{y}_m), \sum_{l=1}^{N\xi}\exp(i\mathrm{x}_k\cdot\boldsymbol{\xi}_l)\omega_l^{\xi}\hat{f}_l \tag{11}$$

a task which is efficiently achieved using the non-uniform FFT of type 3 [8]. Accordingly, the algorithm global complexity to evaluate $g(\mathbf{x}_k)$ is $O((N_{\xi}+N_{\mathrm{y}})\log(N_{\xi}+N_y))$.

As shown in Alouges and Aussal [6], it is possible to extend the procedure to the approximation of the convolution integral

$$g_K(\mathbf{x}) = \int_{\partial\Omega}K(|\mathbf{X}|)f(\mathbf{y})\,\mathrm{d}S, \mathbf{X} = \mathbf{y}-\mathbf{x} \tag{12}$$

when the kernel K admits a sparse cardinal sine decomposition of the form

$$K(r) \sim \sum_{p=1}^{P}\beta_p\,\text{sinc}\,(\lambda_p r)\;\text{ for }\; r \text{ large enough.} \tag{13}$$

The selected accuracy level of the approximation (9) dictates the values of P and coefficients β_p and λ_{p}. In addition, it is also possible to get

$$\text{sinc}\,(\lambda_p\,|\,\mathbf{X}\,|) \sim \sum_{l=1}^{N_{pl}}W_{pl}\exp(i\mathbf{X}\cdot\boldsymbol{\xi}_{pl}) \text{ for } |\,\mathbf{X}\,| \text{ large enough} \tag{14}$$

with points $\boldsymbol{\xi}_{pl}$ on $\mathbb{S}^2$ and relevant weights W_{pl}. More precisely, using eqns (13) and (14) one can actually obtain $|K(|\mathbf{X}|) - K_{\mathbf{a}}(|\mathbf{X}|)| \le \varepsilon$ with ε a prescribed small tolerance and $|\mathbf{X}|$ in the range $[R_{\min}, \mathrm{R}_{\max}]$. The approximating kernel K_a is readily defined as

$$K_a(|\,\mathbf{X}\,|) = \sum_{p=1}^{P}\sum_{l=1}^{N_{pl}}\beta_p W_{pl}\exp(i\mathbf{X}\cdot\boldsymbol{\xi}_{pl}). \tag{15}$$

The trick then consists in using the decomposition

$$g_K(\mathbf{x}) = \int_{\partial\Omega}[K(|\,\mathbf{X}\,|)-K_a(|\,\mathbf{X}\,|)]f(\mathbf{y})dS + \int_{\partial\Omega}K_a(|\,\mathbf{X}\,|)f(\mathbf{y})dS. \tag{16}$$

In getting the operator g_K, the first integral in eqn (16) is calculated by only keeping into account the part of $\partial\Omega$ for which $|\mathbf{X}| \le R_{min}$ (local interactions which then add to the discretized operator g_K a sparse matrix contribution), whereas the second integral is calculated over the entire surface $\partial\Omega$ using eqn (15) and the definition $\mathbf{X} = \mathbf{y} - \mathbf{x}$. Consequently, the convolution operator g_K obeys the same algorithm as the previous one for K = sinc except for the Fourier grid on $\mathbb{S}^2$ which resorts to more points $\xi_{pl.}$

For a given tolerance ε the associated values of P, λ_p, β_p, W_{pl}, ξ_{pl} and $(R_{\min}, R_{\max})$ have been obtained in Alouges and Aussal [6] for the Laplace kernel $K(r) = 1/(4\pi r)$ and the Helmholtz kernel $K(r) = \exp(ikr)/(4\pi r)$.

2.2 Application to Stokes kernels

Unfortunately, the Stokes kernels **G** and **T,** with Cartesian components defined by eqns (4), are clearly not radial ones. In a first attempt recently proposed in Alouges *et al.* [9] to generalize the approach to those kernels too many computations are still needed and this results in a pretty slow algorithm. Therefore, we propose and test in this paper another formulation which was found to improve the performances of the underlying algorithm.

Denoting by $\mathbf{Id} = \mathbf{e}_j \otimes \mathbf{e}_j$ the identity tensor, we write $\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2$ with the following definitions

$$\mathbf{G}_1(\mathbf{x}) = \frac{2}{|\mathbf{x}|}, \mathbf{G}_2(\mathbf{x}) = \frac{\mathbf{x}\otimes\mathbf{x}}{|\mathbf{x}|^3} - \frac{\mathbf{Id}}{|\mathbf{x}|}. \tag{17}$$

Actually, the above decomposition presents two basic merits:

(i) First, $\mathbf{G}_1$ is a radial kernel for which we apply the technique described in Alouges and Aussal [6]. This provides us with a set of integration weights and points $(\omega_j^{\xi,1}, \xi_j^1)_{1\le j\le N_\xi^1}$ and the formula

$$\frac{2}{|\mathbf{x}|} \sim \sum_{j=1}^{N_\xi^1} \omega_j^{\xi,1} \exp(i\mathbf{x}\cdot\xi_j^1) \tag{18}$$

which is valid[1] for all **x** such that $|\mathbf{x}|$ is in the range $[R_{\min}, R_{\max}]$.

(ii) Second, the kernel $\mathbf{G}_2$ is the Hessian matrix of the radial function g_2: $\mathbf{X} \to -|\mathbf{X}|$. Achieving the SCSD of g_2 leads to a new quadrature $(\omega_j^{\xi,2}, \xi_j^2)_{1\le j\le N_\xi^2}$ such that

$$g_2(\mathbf{x}) = -|\mathbf{x}| \sim \sum_{j=1}^{N_\xi^2} \omega_j^{\xi,2} \exp(i\mathbf{x}\cdot\xi_j^2). \tag{19}$$

Differentiating eqn (19) twice immediately yields the approximation

$$\frac{\mathbf{x}\otimes\mathbf{x}}{|\mathbf{x}|^3} - \frac{\mathbf{Id}}{|\mathbf{x}|} \sim -\sum_{j=1}^{N_\xi^2} \xi_j^2 \otimes \xi_j^2 \omega_j^{\xi,2} \exp(i\mathbf{x}\cdot\xi_j^2). \tag{20}$$

The eqns (18) and (20) finally permit us to apply the method.

In a similar fashion, for the Cartesian components of the Green stress tensor **T** we use this time the identity

$$\frac{x_i x_j x_k}{|\mathbf{x}|^5} = \frac{1}{3}\frac{\partial^3}{\partial x_i \partial x_j \partial x_k}|\mathbf{x}| + \delta_{ij}\frac{x_k}{|\mathbf{x}|^3} + \delta_{ik}\frac{x_j}{|\mathbf{x}|^3} + \delta_{jk}\frac{x_i}{|\mathbf{x}|^3}. \tag{21}$$

[1] The choice of $R_{\min}$ and $R_{\max}$ is explained in Alouges *et al*. [6].

At that stage, the third derivative of $|\mathbf{x}|$ is computed as before (see how to deduce eqn (20) for eqn (19)) while the other terms are essentially coordinates of the gradient of $|\mathbf{x}|^{-1}$ (use this time eqn (18)). As before, all those terms are computed using the SCSD.

3 NUMERICAL VALIDATION

Henceforth, we benchmark the proposed strategy for a solid ellipsoid with surface $\partial\Omega$ defined by

$$\frac{x_1^2}{a_1^2}+\frac{x_2^2}{a_2^2}+\frac{x_3^2}{a_3^2}=1 \text{ for } \mathbf{x} \text{ on } \partial\Omega;\ (a_1,a_2,a_3)=(5,3,2). \tag{22}$$

As illustrated in Fig. 1, the surface $\partial\Omega$ is approximated as $\partial\Omega_h$ using flat P_1 triangular boundary elements with typical size h which is the average length of the sides of the boundary elements. The normal to $\partial\Omega_h$ is approximated by the normal to the surface $\partial\Omega_h$ on which we put N nodes (recall that $N_d = 3N$). The computations are run for a Galerkin approach using a GMRS iterative solver and a parallel MATLAB code. For each test, the SCSD method convergence error (versus h) and CPU time (versus N) are compared against the ones obtained by a full BEM method and termed BEM. Computations are run for N between 200 and 5000 for BEM and for N between 200 and 50000 for SCSD. Moreover, different accuracy level (tolerance) ε for the SCSD are taken while the prescribed residual for the GMRS solver is taken to be in getting the operator ε.

3.1 The stresslet

We first test the stresslet contribution, i.e. the second integral appearing in eqn (6). This is done taking on $\partial\Omega$ a rotation with velocity $\mathbf{w} = \mathbf{e}_1 \wedge \mathbf{x} = w_i\mathbf{e}_i$. Hence, we introduce on $\partial\Omega$ the vector field $\mathbf{Q} = Q_j\mathbf{e}_j$ by

$$Q_j(\mathbf{x}) = \frac{1}{8\pi}\int_{\partial\Omega}^{PV}[\omega_i n_k](\mathbf{y})T_{ijk}(\mathbf{y}-\mathbf{x})dS. \tag{23}$$

From Pozrikidis [1] one has the analytical result

$$\mathbf{Q}(\mathbf{x}) = -\mathbf{w}(\mathbf{x})/2 \text{ for } \mathbf{x} \text{ on } \partial\Omega \tag{24}$$

and we accordingly define the error Err as the following quantity

$$Err = [\int_{\partial\Omega}|\mathbf{Q}(\mathbf{x})+\mathbf{w}(\mathbf{x})/2|^2\ dS]^{1/2}. \tag{25}$$

The computations give $Err = O(\varepsilon)$ for SCSD and $Err \sim 10^{-14}$ for BEM. Moreover, as shown in Fig. 1, the cpu time CPU behaves as N^2 for BEM and as $N \log(N)$ for SCSD.

3.2 The Stokeslet

We now test a term similar to the last integral occurring in eqn (6). To do so, we this time define on $\partial\Omega$ the vector field $\mathbf{Q}$ by

$$Q_j(\mathbf{x}) = \int_{\partial\Omega} G_{ij}(\mathbf{y}-\mathbf{x})n_i(\mathbf{y})dS \text{ for } \mathbf{x} \text{ on } \partial\Omega. \tag{26}$$

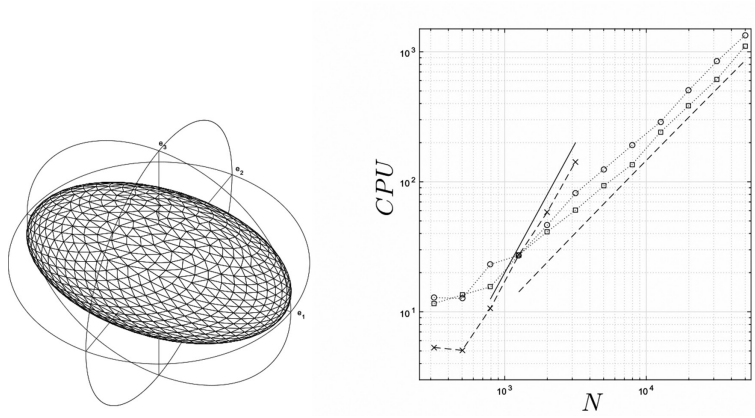

Figure 1: Surface mesh $\partial\Omega_h$ for the ellipsoid. CPU time versus N for the Stresslet test: BEM (×) and SCSD for $\varepsilon = 10^{-3}$(□) or $\varepsilon = 10^{-4}$(o)**.** Functions $N\log(N)$ (dashed line) and N^2 (solid line) are also plotted.

As it is well known [1], one has in theory $\mathbf{Q(x)} = \mathbf{0}$ whatever $\mathbf{x}$ located on the boundary $\partial\Omega$. Thus, we now define the error Err as

$$Err = [\int_{\partial\Omega} |\,\mathbf{Q(x)}\,|^2\ dS]^{1/2}. \tag{27}$$

Both Err and the CPU time are plotted in Fig. 2. Note that $Err = O(h^2)$ for BEM while, not surprisingly, $Err = O(\varepsilon)$ for SCSD. As ε decreases from $\varepsilon = 10^{-2}$ to $\varepsilon = 10^{-4}$ the SCSD converges to BEM. As regard the CPU time, is appears that BEM and SCSD behave as N^2 and $N\log(N)$, respectively.

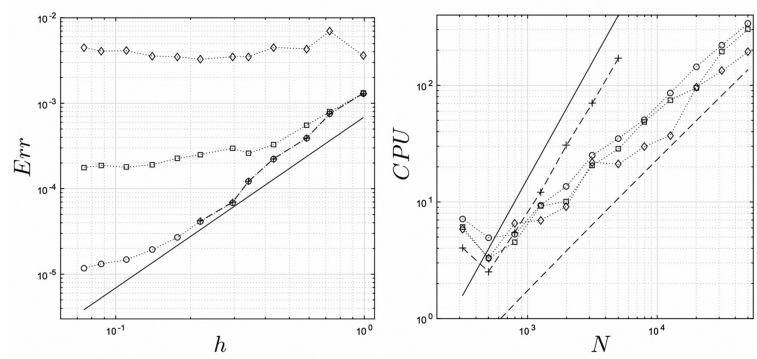

Figure 2: Error and CPU for the Stokeslet test. BEM (+) and SCSD for $\varepsilon = 10^{-2}$(◊), $\varepsilon = 10^{-3}$(□) and $\varepsilon = 10^{-4}$(o). Functions $N\log(N)$ (dashed line) and either h^2 or N^2 (solid lines) are also plotted.

3.3 Integral representation

We test the whole relation (6) for the Stokes flow $(\mathbf{u}_0, p_0)$, obeying eqns (2) and (3) and thus also eqn (6), produced by a force point with unit strength $\mathbf{e}_1$ located inside the ellipsoid at point $\mathbf{x_0} = (1, 2, 0.5)$. Clearly, for this flow and associated stress tensor σ_0 we have

$$u_{0,j}(\mathbf{x}) = \mathbf{u}_0 \cdot \mathbf{e}_j = \frac{G_{1j}(\mathbf{x}-\mathbf{x}_0)}{8\pi\mu}, \; [\boldsymbol{\sigma}_0 \cdot \mathbf{n}]_j(\mathbf{x}) = \frac{T_{1jk}(\mathbf{x}-\mathbf{x}_0)n_k(\mathbf{x})}{8\pi}. \tag{28}$$

The error *Err* is the $L^2(\partial\Omega)$ norm between $\mathbf{u}_0$ and its integral representation given by eqn (6). Both the computed error *Err* and cpu time *CPU* are plotted in Fig. 3. It appears that $Err = O(h)$ for BEM and SCSD with tolerance $\varepsilon = 10^{-4}$. In addition, the cpu time is order N^2 or $N \log(N)$ for BEM or SCSD, respectively.

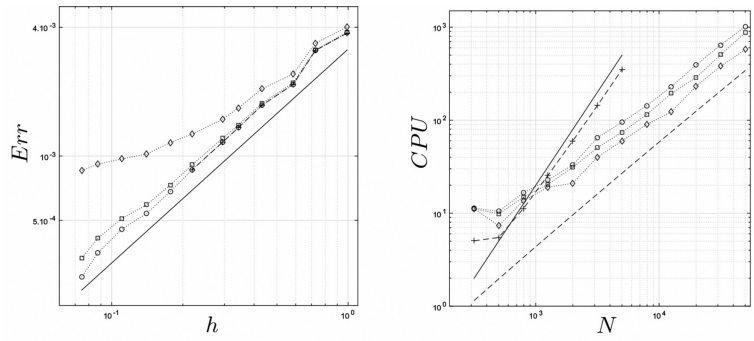

Figure 3: Error and CPU for the integral representation. BEM (+) and SCSD for $\varepsilon = 10^{-2}$(◊), $\varepsilon = 10^{-3}$(□) and $\varepsilon = 10^{-4}$(o). Functions $N \log(N)$ (dashed line) and N (solid line for error) or N^2 (solid line for CPU) are also plotted.

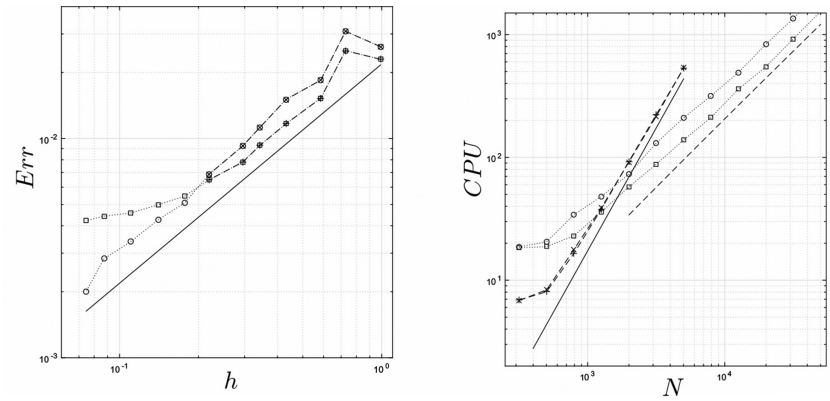

Figure 4: Error and CPU for the Dirichlet to Neumann problem. BEM with $\varepsilon = 10^{-3}$(+) or $\varepsilon = 10^{-4}$(×) and SCSD for $\varepsilon = 10^{-3}$(D) and $\varepsilon = 10^{-4}$(o). Functions $N \log(N)$ (dashed line) and N (solid line for error) or N^2 (solid line for CPU) are also plotted.

3.4 The Dirichlet to Neumann problem

Finally, this last test concerns the Dirichlet to Neumann problem, i.e. we provide the velocity $\mathbf{u}_0$ on the ellipsoid boundary $\partial\Omega$ and compute the resulting traction $(\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num}$ there from the boundary-integral eqn (6). This traction [1] is defined up to a constant multiple of **n.** Denoting by $(\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num}$ the computed traction, we first calculate the constant λ which minimizes the $L^2(\partial\Omega)$ norm of $\boldsymbol{\sigma}_0 \cdot \mathbf{n} - (\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num} - \lambda\mathbf{n}$. Then, the numerical error *Err* is defined as

$$Err = \left\{ \int_{\partial\Omega} \left| (\boldsymbol{\sigma}_0 \cdot \mathbf{n}) - (\boldsymbol{\sigma}_0 \cdot \mathbf{n})_{num} - \lambda \mathbf{n} \right|^2 dS \right\}^{1/2}. \tag{29}$$

The resulting *Err* and cpu time *CPU,* given in Fig. 4, exhibit the same trends as the ones observed for the previous integral representation test.

4 CONCLUSION

A new sparse cardinal sine decomposition for 3D Stokes flow has been proposed, implemented and also compared, both in terms of accuracy and cpu time, against a classical BEM solver. It is different from and more efficient than the one recently proposed and tested in Alouges *et al.* [9]. We aim in future at implementing this technique also to the *regularized* boundary-integral eqn (5).

REFERENCES

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

[2] Greengard, L. & Rokhlin, V., *The Rapid Evaluation of Potential Fields in Three Dimensions*, *In Vortex Methods*, Springer Verlag, pp. 121–141, 1988.

[3] Greengard, L., *The Rapid Evaluation of Potential Fields in Particle Systems,* MIT Press, 1988.

[4] Hackbusch, W., *A Sparse Matrix Arithmetic Based on H-Matrices. Part I. Introduction to H-matrices,* Computing, **62**(2), pp. 89–108, 1999.

[5] Hackbusch, W., *Hierarchische Matrizen,* Springer, 2009.

[6] Alouges, F. & Aussal, M., The sparse cardinal sine decomposition and its application for fast numerical convolution. *Numerical Algorithms,* **70**(2), pp. 427–448, 2015. http://dx.doi.org/10.1007/s11075-014-9953-6

[7] Dutt, A. & Rokhlin, V., Fast fourier transforms for nonequispaced data. *SIAM Journal on Scientific Computing*, **14**(6), pp. 1368–1393, 1993. http://dx.doi.org/10.1137/0914081

[8] Lee, J.-Y. & Greengard, L., The type 3 nonuniform FFT and its application. *Journal of Computational Physics*, **206**(1), pp. 1–5, 2005. http://dx.doi.org/10.1016/j.jcp.2004.12.004

[9] Alouges, F., Aussal, M., Lefebvre-Lepot, A., Pigeonneau, F. & Sellier, A. The sparse cardinal sine decomposition applied to Stokes integral equations. *Proceedings of the ICMF-2016, 9th International Conference on Multiphase Flow*, Florence, 2016.