The Boundary Element Method: Theoretical Formulation, Numerical Approximation, and Parallel Implementation

Type A project for the *Numerical Analysis for PDEs* course

Matteo Bonfadini [*,1], **Manuel Alfano** [**,1]

[*] *Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy (e-mail:* `matteo.bonfadini@mail.polimi.it`)
[**] *Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy (e-mail:* `manuel.alfano@mail.polimi.it`)

May 23, 2024

Abstract: The past fifty years have been marked by the evolution of computers and an enormous availability of computational power. This has boosted the development of computational methods and their application in engineering. The most popular computational method is the Finite Element Method (FEM), however around 1970 the engineering community started to develop the Boundary Element Method (BEM). This work considers the BEM theoretical formulation, and both its numerical approximation and parallel implementation based on the π-`BEM` library.

Keywords: BEM, Fundamental solution, Collocation method, Fast multipole method, MPI

1. INTRODUCTION

The term *Boundary Element Method* was coined in 1977 in three publications: Banerjee and Butterfield, Brebbia and Dominguez, and Dominguez. The mathematical foundations were established by Betti in 1872 and Somigliana in 1886 for elasticity problems. Additionally, Green in 1828 and Fredholm in 1903 made foundational contributions to potential problems.

A fair question to ask is "why do we need the BEM since we already have the FEM that solves engineering problems?". The answer is that modeling with finite elements can be ineffective and laborious for certain classes of problems. So the FEM, despite the generality of its application in engineering problems, is not free of drawbacks. The most important advantages of preferring BEM over FEM are:

(1) The solution is mathematically expressed as a continuous mathematical formula, namely a Boundary Integral Equation (BIE). Therefore the associated numerical method, the BEM, benefits of a discretization only over the boundary Γ of the problem domain Ω, leading to a significant reduction in the number of degrees of freedom of the numerical model (see Fig. 1).
(2) The method is particularly effective in computing accurately the derivatives of the field function (e.g., fluxes, strain, stresses, moments). Instead, in FE methods the accuracy drops considerably in areas of large gradients.

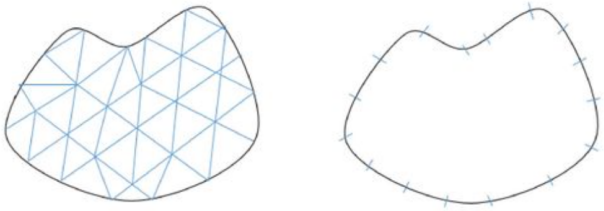

Fig. 1. FEM versus BEM domain discretization.

However, we mention that in mixed Neumann Dirichlet problems with Lipschitz domain there arises an issue when considering continuous elements. A possible solution is the so-called double node technique, where, thanks to the splitting of a physical node into more computational nodes, continuity is preserved only on the solution, while its normal gradient is allowed to have a jump across physical edges.
(3) For infinite domains, the problem is formulated simply as an exterior one. In this manner, computer programs developed for finite domains can be used, with just a few modifications, to solve problems in infinite domains. This is not possible with the FEM.
(4) Even if nowadays there a lot of advanced professional FE softwares equipped with automatic and adaptable mesh generators, the BEM is well suited for solving problems in domains with geometric peculiarities, such as cracks and holes. Moreover, BEM is more fea-

[1] We deeply thank for the opportunity Prof. P.F.Antonietti, Dr. F. Regazzoni, Dr. C. B. Leimer Saglio, and for the support Dr. I. Mazzieri.

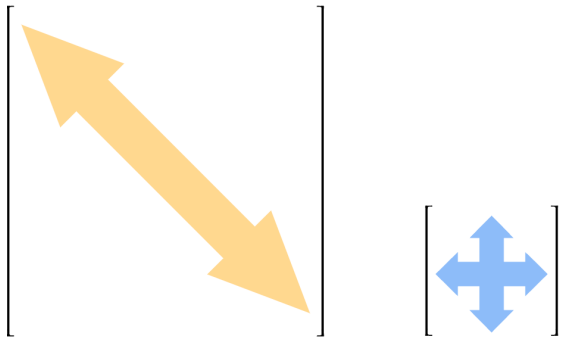

Fig. 2. FEM versus BEM coefficient matrix.

sible for problems described by differential equations of fourth or higher order (e.g., plate equation).

On the other hand, the BEM exhibits the following main disadvantages:

(6) Application of the BEM requires the establishment of the BIE. This is possible only if the problem is linear and its fundamental solution can be established, such as for Laplace equation, Helmholtz equation, and Stokes system. Hence, the method cannot be used for problems whose fundamental solution is either unknown or cannot be determined. Such are, for example, problems described by differential equations with variable coefficients.

(7) The numerical implementation of the BEM results in systems of linear algebraic equations whose coefficient matrices are fully populated and nonsymmetrical. Moreover, if one considers mixed Neumann Dirichlet boundary value problems the final linear system is generally ill conditioned. In a FEM model, however, the corresponding matrices have some very nice properties, they are banded and symmetric. This drawback of the BEM is counterbalanced by a smaller dimension of its matrices (see Fig. 2).

A number of steps are possible to improve the applicability of BEM. We begin by discussing high order elements, which reduce the number of degrees of freedom needed to achieve a certain tolerance in problems with smooth solutions. High order BEMs are more attractive when compared to their FE counterparts, since increasing the order in finite elements leads to a denser, more ill conditioned system matrix. In the case of BEM the matrices, accordingly to drawback (7), are already full, and this is no more a disadvantage. If the solution is non smooth, one can combine high order elements with local refinement techniques to achieve the same result.

As the size of problem increases, none of these techniques alone is enough to reduce memory problems. Then it is usually applied a domain decomposition by exploiting distributed memory techniques, which reduce the number of degree of freedom handled by each processor. Moreover, in order to optimize BEM matrix-vector product operations from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ it has been developed the so-called Fast Multipole Method (FMM), which thanks to a hierarcical structure (see Fig. 3) can decompose short and long range interaction.

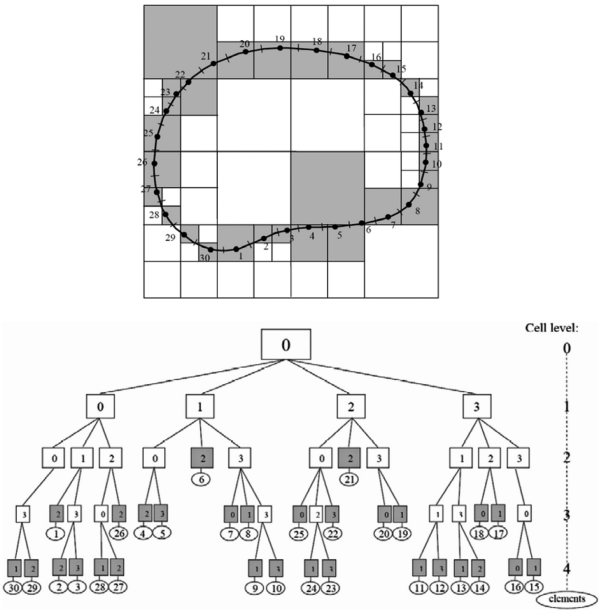

Fig. 3. Hierarchical quad-tree structure for a 2D boundary element mesh.

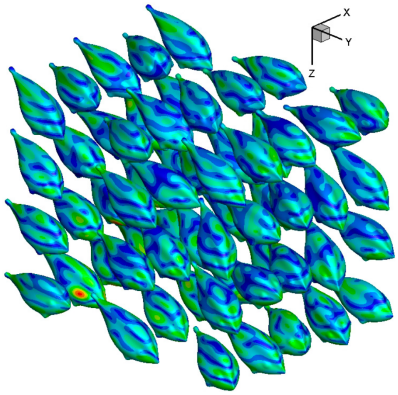

Fig. 4. Scattering of a multiple fish model.

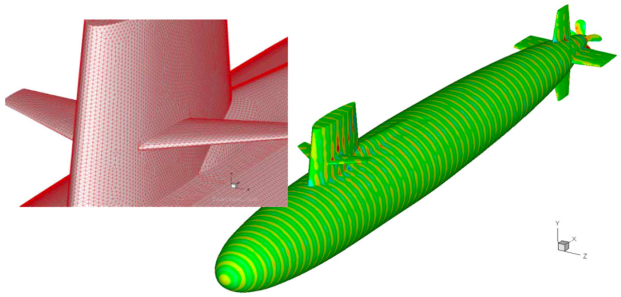

Fig. 5. BEM model of the Skipjack submarine impinged upon by an incident wave in the direction $(1, 0, -1)$.

Nowadays, boundary integral formulation and in particular the Fast Multipole BEM (FMBEM) have been applied to problems involving hydrodynamic flows (e.g., Fig. 4), flow around aerodynamic lifting bodies, structural mechanics, electrostatics, quantum mechanics, and acoustics (e.g., Fig. 5-6). See [2],[3] for more delailed explanations.

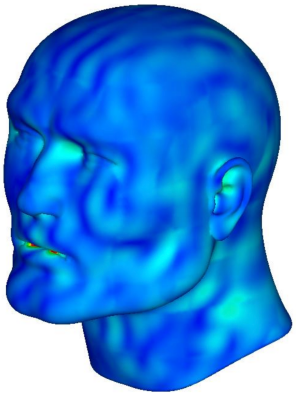

Fig. 6. BEM mesh and sound-pressure plots for a human-head model.

In ths work, we are going to present for a model Laplace problem the theoretical background of BEM methodologies, as well as convergence and scalability tests of the parallel implementation carried out in [4].

2. BOUNDARY INTEGRAL FORMULATION

We consider a bounded open domain $\Omega \subseteq \mathbb{R}^3$ with Lipschitz boundary $\Gamma = \partial\Omega$, and we want to solve potential problems described by the Laplace equation

$$\Delta\phi = 0 \qquad \text{in } \Omega. \tag{1}$$

We couple Eq. (1) with the following boundary conditions:

$$\phi = f_D \qquad \text{on } \Gamma_D, \tag{2a}$$

$$\frac{\partial\phi}{\partial n} = f_N \qquad \text{on } \Gamma_N, \tag{2b}$$

where $\frac{\partial\phi}{\partial n} = \nabla\phi \cdot \mathbf{n}$, $\Gamma_D \cup \Gamma_N = \Gamma$, $\Gamma_D \cap \Gamma_N = \varnothing$, and $\Gamma_D \neq \varnothing$.

2.1 Laplace Equation in Distributional Form

Before dealing with Eq. (1), let us consider the following equation:

$$-\Delta\phi(\mathbf{y}) = \delta_{\mathbf{x}} \qquad \forall\, \mathbf{y} \in \mathbb{R}^3. \tag{3}$$

This is known as the Poisson equation in the whole space. Here, $\phi(\mathbf{y})$ represents the potential produced at a point $\mathbf{y}$ in the domain Ω generated by a unit point source placed at point $\mathbf{x}$, i.e., the Dirac Delta function $\delta_{\mathbf{x}}$. A solution of Eq. (3) is called the fundamental solution of the laplacian, and it can be determined as follows.

We write Eq. (3) in polar coordinates with origin at $\mathbf{x}$. Since this solution is axisymmetric with respect to the source, it is independent of the polar angle θ, thus the three-dimensional laplacian is

$$\Delta\phi = \phi_{rr} + \frac{2}{r}\,\phi_r,$$

where $r = |\mathbf{x} - \mathbf{y}|$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. The right-hand side vanishes at all points of the plane, except at the origin $r = 0$ (or $\mathbf{y} = \mathbf{x}$), where it has infinite value. Then Eq. (3) is written as

$$\phi_{rr} + \frac{2}{r}\,\phi_r = 0 \qquad \forall\, r > 0.$$

The change of variables $\phi_r = v$ gives us

$$v' + \frac{2}{r}\,v = 0 \quad \rightsquigarrow \quad \int \frac{v'}{v} = \int -\frac{2}{r}$$
$$\rightsquigarrow \quad \log(v) = -2\log(r) + c \quad \rightsquigarrow \quad v = \frac{c}{r^2},$$

then

$$\phi = \int \frac{c}{r^2} = \frac{c_1}{r} + c_2.$$

Since we want a particular solution we may set $c_2 = 0$, and to determine c_1 we perform an integration by parts on Eq. (3) over a ficticious unit circular domain Ω:

$$\int_\Omega -\Delta\phi\ \varphi\ \mathrm{d}\Omega = -\int_{\partial\Omega} \frac{\partial\phi}{\partial n}\ \varphi\ \mathrm{d}S + \int_\Omega \nabla\phi \cdot \nabla\varphi\ \mathrm{d}\Omega$$

which, for $\varphi \equiv 1$, reads

$$\int_\Omega \delta_{\mathbf{x}}\ \mathrm{d}\Omega = \int_{\partial\Omega} \frac{\partial\phi}{\partial n}\ \mathrm{d}S.$$

Due to the axisymmetric nature of the problem

$$\frac{\partial\phi}{\partial n} = \frac{\partial\phi}{\partial r} = \frac{c_1}{r^2}$$

thus, after using the definition of Delta function and performing an integration in radial coordinates, we are left with

$$1 = 4\pi \int_0^1 \rho^2\ v(\rho)\ \mathrm{d}\rho = 4\pi \int_0^1 \rho^2\ \frac{c_1}{\rho^2}\ \mathrm{d}\rho = 4\pi c_1.$$

We discovered $c_1 = \frac{1}{4\pi}$, hence, the fundamental solution becomes $\phi = \frac{1}{4\pi r}$, which is also known in the literature as the free space Green's function:

$$G(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi\,|\mathbf{x} - \mathbf{y}|} \qquad \forall\, \mathbf{y} \in \mathbb{R}^3. \tag{4}$$

One can easily prove (see for example [1]) that this expression is the solution of $-\Delta G(\mathbf{x}, \cdot) = \delta_{\mathbf{x}}$ in $\mathcal{D}'(\mathbb{R}^3)$, i.e., in the distributional sense.

2.2 BEM for the Laplace Equation

Here we derive the solution of the Laplace equation (1) with mixed boundary conditions (2).

We can multiply the Laplace equation by an arbitrary test function φ and integrate by parts twice:

$$0 = \int_\Omega -\Delta\phi\ \varphi\ \mathrm{d}\Omega = -\int_\Gamma \frac{\partial\phi}{\partial n}\ \varphi\ \mathrm{d}S + \int_\Omega \nabla\phi\ \nabla\varphi\ \mathrm{d}\Omega$$
$$= -\int_\Gamma \frac{\partial\phi}{\partial n}\ \varphi\ \mathrm{d}S + \int_\Gamma \frac{\partial\varphi}{\partial n}\ \phi\ \mathrm{d}S - \int_\Omega \Delta\varphi\ \phi\ \mathrm{d}\Omega$$

(one could directly apply the Green's second identity). Choosing $\varphi \equiv G$, the free space Green's function defined in Eq. (4), Laplace Eq. (1) becomes

$$\int_\Omega -\Delta G\ \phi(\mathbf{y})\ \mathrm{d}\Omega(\mathbf{y}) = \int_\Gamma \left[\frac{\partial\phi}{\partial n}(\mathbf{y})\ G - \frac{\partial G}{\partial n}\ \phi(\mathbf{y})\right] \mathrm{d}S(\mathbf{y}).$$

Since we know that $-\Delta G(\mathbf{x}, \mathbf{y}) = \delta_{\mathbf{x}}$,

$$\int_\Omega \delta_{\mathbf{x}}\ \phi(\mathbf{y})\ \mathrm{d}\Omega(\mathbf{y}) = \int_\Gamma \left[\frac{\partial\phi}{\partial n}(\mathbf{y})\ G - \frac{\partial G}{\partial n}\ \phi(\mathbf{y})\right] \mathrm{d}S(\mathbf{y})$$

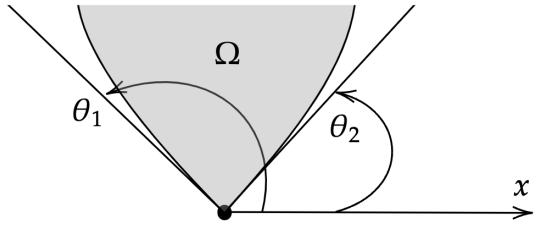

Fig. 7. Solid angle related to a corner point of a nonsmooth boundary.

hence, by the definition of Dirac Delta, we are left with

$$\phi(\mathbf{x}) = \int_\Gamma \left[\frac{\partial \phi}{\partial n}(\mathbf{y})\ G(\mathbf{x},\mathbf{y}) - \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})\ \phi(\mathbf{y}) \right] \mathrm{d}S(\mathbf{y}) \quad (5)$$

$\forall\, \mathbf{x} \in \Omega$. This expression is the integral representation of the solution for the Laplace equation at any point $\mathbf{x}$ inside the domain Ω in terms of the boundary values of ϕ and its normal derivative $\partial\phi/\partial n$. It is apparent from the boundary conditions (2a) and (2b), that only one of the quantities ϕ or $\partial\phi/\partial n$ is prescribed at a point $\mathbf{y}$ on the boundary. Consequently, it is not yet possible to determine the solution from the integral representation (5). Therefore, we let $\mathbf{x}$ lie on the boundary Γ: we notiche that the kernels $G(\mathbf{x},\mathbf{y})$ and $\frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})$ become weakly singular (but integrable) and singular respectively. Hence, considering the Cauchy Principal Value (CPV) of the $\frac{\partial G}{\partial n}$ singular integral, we can write

$$\alpha(\mathbf{x})\phi(\mathbf{x}) = \int_\Gamma \frac{\partial \phi}{\partial n}(\mathbf{y})\ G(\mathbf{x},\mathbf{y})\ \mathrm{d}S(\mathbf{y}) - \int_\Gamma^{\mathrm{PV}} \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})\ \phi(\mathbf{y})\ \mathrm{d}S(\mathbf{y}), \quad (6)$$

where the coefficient $\alpha(\mathbf{x})$ is obtained fromt the CPV evaluation of the singular integral, and it represents the fraction of solid angle (see Fig. 7) with which the domain Ω is seen from the boundary point $\mathbf{x}$:

$$\alpha(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{x} \text{ inside } \Omega, \\ \frac{\theta_1 - \theta_2}{2\pi} & \text{for } \mathbf{x} \text{ on the boundary } \Gamma, \\ 0 & \text{for } \mathbf{x} \text{ outside } \Omega. \end{cases}$$

In particular, $\alpha = 1/2$ for smooth boundary points.

By explicitly writing the boundary conditions using the characteristic function $\chi_A(\mathbf{x})$, which is one if $\mathbf{x} \in A$ and zero otherwise, we obtain

$$\chi_{\Gamma_N}(\mathbf{x})\alpha(\mathbf{x})\phi(\mathbf{x}) - \int_{\Gamma_N} f_N(\mathbf{y})\ G(\mathbf{x},\mathbf{y})\ \mathrm{d}S(\mathbf{y}) + \int_{\Gamma_N}^{\mathrm{PV}} \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})\ \phi(\mathbf{y})\ \mathrm{d}S(\mathbf{y}) =$$
$$= -\chi_{\Gamma_D}(\mathbf{x})\alpha(\mathbf{x}) f_D(\mathbf{x}) + \int_{\Gamma_D} \frac{\partial \phi}{\partial n}(\mathbf{y})\ G(\mathbf{x},\mathbf{y})\ \mathrm{d}S(\mathbf{y}) - \int_{\Gamma_D}^{\mathrm{PV}} \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})\ f_D(\mathbf{y})\ \mathrm{d}S(\mathbf{y}).$$

Grouping all the unknowns in the left-hand side we finally deduce

$$\chi_{\Gamma_N}(\mathbf{x})\alpha(\mathbf{x})\phi(\mathbf{x}) - \int_{\Gamma_D} \frac{\partial \phi}{\partial n}(\mathbf{y})\ G(\mathbf{x},\mathbf{y})\ \mathrm{d}S(\mathbf{y}) + \int_{\Gamma_N}^{\mathrm{PV}} \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})\ \phi(\mathbf{y})\ \mathrm{d}S(\mathbf{y}) =$$
$$= -\chi_{\Gamma_D}(\mathbf{x})\alpha(\mathbf{x}) f_D(\mathbf{x}) + \int_{\Gamma_N} f_N(\mathbf{y})\ G(\mathbf{x},\mathbf{y})\ \mathrm{d}S(\mathbf{y}) - \int_{\Gamma_D}^{\mathrm{PV}} \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y})\ f_D(\mathbf{y})\ \mathrm{d}S(\mathbf{y}). \quad (7)$$

Eq. (7) simply is the trace of Eq. (5), namely it is the BIE with which it is possible to derive the potential where its normal derivative is known, and viceversa.

3. DISCRETISATION

We use standard Lagrangian finite element spaces on Γ as basis functions both for the geometry and the unknowns ϕ and $\frac{\partial \phi}{\partial n} =: \gamma$. The approximation can be divided in 6 main steps.

3.1 Computational Mesh Generation

We define a quadrilateral computational mesh meant as a regular [2] decomposition Γ_h of the boundary Γ. Then we progressively refine this very coarse mesh, by exploiting a proper *a posteriori* error estimator, to obtain a solution with desired accuracy.

3.2 Definition of the Discrete Spaces

The right spaces to analyse Eq. (6) are

$$\phi \in V = H^{1/2}(\Gamma) \qquad \gamma \in Q = H^{-1/2}(\Gamma)$$

In a conforming setting we choose $V_h \subseteq V,\ Q_h \subseteq Q$ as

$$V_h := \left\{ \phi_h \in L^2(\Gamma_h) \,:\, \phi_h\big|_K \in \mathbb{Q}^r(K),\ K \in \Gamma_h \right\}, \quad (8a)$$
$$Q_h := \left\{ \gamma_h \in L^2(\Gamma_h) \,:\, \gamma_h\big|_K \in \mathbb{Q}^s(K),\ K \in \Gamma_h \right\}. \quad (8b)$$

It can be shown that using the same FE discretization for both unknowns does not lead to big disadvantages, thus

$$V_h = Q_h \equiv \mathrm{span}\{\psi_i\}_{i=1}^{N_V}.$$

3.3 Collocation of the BIE

Two approaches are possible in order to discretize Eq. (7). The *Galerkin BEM* is a natural choice and it consists in writing the problem in variational form; however, this implies a second integration of weakly singular kernels, which would increase the computational complexity of the BEM. Therefore, we focus on *collocation methods*. The idea is to directly use the discretized ϕ_h and γ_h instead of the continuous ϕ and γ, and to collocate the BIE on a number of points equal to the number of unknowns, for example on the support of the Lagrangian basis functions.

We collocate Eq. (6) [3] on the nodes x_i of the boundary mesh, obtaining

$$\alpha(x_i)\phi(x_i) = \sum_{K\in\Gamma_h} \int_K \gamma(y)\ G(x_i,y)\ \mathrm{d}S(y) + \\ - \sum_{K\in\Gamma_h} \int_K \frac{\partial G}{\partial n}(x_i,y)\ \phi(y)\ \mathrm{d}S(y)$$

for every $i = 1,\dots,N_V$. Then by a quadrature formula which exploit the geometry of the boundary instead of using generic quadrature weights we get

$$\alpha(x_i)\phi(x_i) = \sum_{K\in\Gamma_h} \sum_{q=1}^{N_q} \gamma(x_q) G(x_i,x_q) J^K(x_q) + \\ - \sum_{K\in\Gamma_h} \sum_{q=1}^{N_q} \frac{\partial G}{\partial n}(x_i,x_q)\phi(x_q) J^K(x_q),$$

where J^K is the determinant of the first fundamental form for each reference cell $\hat{K}$. Now, as we anticipated before, we substitute ϕ with its approximation, i.e. $\phi(x_q) \approx \phi_h =$

[2] For every cell K there exist a mapping from a reference cell $\hat{K}$ to K with positive determinant of the Jacobian.

[3] It's simpler to impose the boundary conditions after assembling the linear system.

$\sum_j \phi_j \psi_j(x_q)$. We do the same thing for the flux $\frac{\partial \phi}{\partial n}$, and this yields

$$\alpha(x_i)\phi_j\psi_j(x_i) = \sum_{K\in\Gamma_h}\sum_{q=1}^{N_q}\gamma_j\psi_j(x_q)G(x_i,x_q)J^K(x_q) + - \sum_{K\in\Gamma_h}\sum_{q=1}^{N_q}\frac{\partial G}{\partial n}(x_i,x_q)\phi_j\psi_j(x_q)J^K(x_q),$$

and finally we obtain the linear system

$$(\alpha + N)\hat{\phi} - D\hat{\gamma} = 0, \tag{9}$$

where

- α is the diagonal matrix of values $\alpha(x_i)$;
- $D_{ij} = \sum_K \sum_q G(x_i, x_q)\psi_j^q J^K$;
- $N_{ij} = \sum_K \sum_q \frac{\partial G}{\partial n}(x_i, x_q)\psi_j^q J^K$;
- $\hat{\phi}$, $\hat{\gamma}$ are the nodal values of the unknowns.

3.4 Imposition of the Boundary Conditions

To impose the boundary condition for ϕ_h and γ_h, we define the additional vectors

$$\tilde{\phi} = \begin{cases} 0 & \text{on } \Gamma_D \\ \hat{\phi} & \text{on } \Gamma_N \end{cases} \qquad \tilde{\gamma} = \begin{cases} \hat{\gamma} & \text{on } \Gamma_D \\ 0 & \text{on } \Gamma_N \end{cases} \tag{10a}$$

$$\bar{\phi} = \begin{cases} f_D & \text{on } \Gamma_D \\ 0 & \text{on } \Gamma_N \end{cases} \qquad \bar{\gamma} = \begin{cases} 0 & \text{on } \Gamma_D \\ f_N & \text{on } \Gamma_N \end{cases} \tag{10b}$$

With these defintions we can decouple the action of the two unknowns: while (10a) clarify the real unknowns, (10b) help setting the known values. The decoupled system is

$$\begin{cases} (\alpha + N)\hat{\phi} - D\hat{\gamma} = 0 & \text{on } \Gamma_D \\ (\alpha + N)\hat{\phi} - D\hat{\gamma} = 0 & \text{on } \Gamma_N \end{cases}$$

therefore

$$\underbrace{(\alpha + N)\tilde{\phi} - D\tilde{\gamma}}_{A\tilde{t}} = \underbrace{-(\alpha + N)\bar{\phi} + D\bar{\gamma}}_{b}. \tag{11}$$

To wrap up, we obtained the linear system $A\tilde{t} = b$, where $\tilde{t}$ simply groups the unknowns:

$$\tilde{t}_i = \begin{cases} \hat{\gamma}_i & \text{if } x_i \in \Gamma_D \\ \hat{\phi}_i & \text{if } x_i \in \Gamma_N \end{cases}$$

Notice that the system in Eq. (11) is *one to one* with Eq. (7), as one would expect.

3.5 Solution of the Linear System

In order to solve system (11), we don't assemble the matrix A but we use the Generalized Minimal Residuals (GMRES), which is an iterative method based on Krylov subspace methods. By using least squares (and a preconditioner to accelerate the convergence), we minimize the residual $r^{(k)} = b - At^{(k)}$ till desired accuracy. This procedure works well in BEM frameworks, because it allows you to deal with nonsymmetric and fully populated matrices, and without explicitly forming them, saving a lot of memory.

3.6 Post Process

We compute an error analysis to verify the proper convergence of the method and if necessary we proceed with a local refinement; then we recover the normal derivative $\frac{\partial \phi}{\partial n}$ (or simply the gradient $\nabla\phi$) by computing the derivative over Eq. (6), and finally we output the solution.

4. RESULTS

We analyse the convergence of the numerical scheme to a known analytical solution

$$\phi(x,y,z) = \frac{-1}{4\pi\sqrt{(x-1)^2 + (y-0.5)^2 + (z-0.5)^2}} \tag{12}$$

for a mixed Dirichlet-Neumann problem on the domain configurations illustrated in Fig. 8 and Fig. 12.

The first test case considers a sphere of radius $R = 1$ and centered in $O = (0, 0, 0)$. On the left of Fig. 8 we show the initial grid, represented by a simple cube. Dirichlet boundary conditions are applied to the red faces, while Neumann boundary conditions are applied to the blue faces. The right image represents the mesh obtained after 3 global refinement cycles. The possibility to introduce a very coarse initial discretisation of the domain is a key feature of the `π-BEM` library.

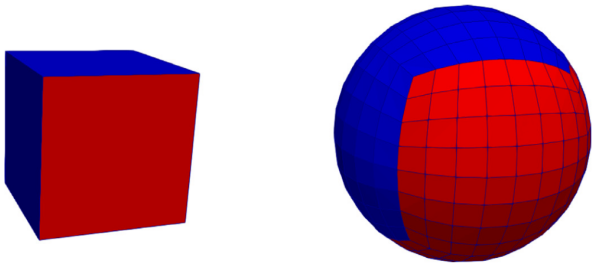

Fig. 8. Mesh configuration for the sphere test case with mixed boundary conditions.

While Fig. 9 shows the computed solution ϕ over 98304 cells, Fig. 10 presents the convergence analysis for both the potential (on the left) and its normal derivative (on the right). We obtained a first order convergence on ϕ using Q_1 elements for both L_2 and L_∞ norm.

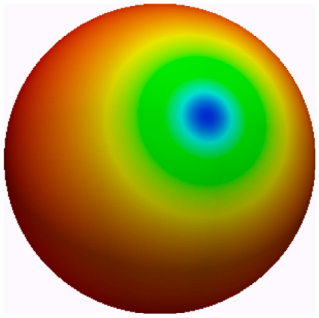

Fig. 9. Sphere test case. The colours represent the magnitude of the exact solution ϕ.

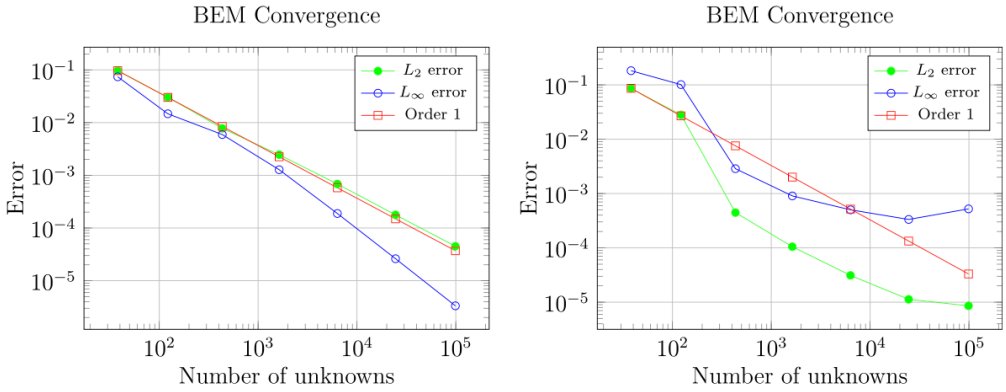

Fig. 10. Convergence analysis for the error in a mixed Dirichlet-Neumann problem using Q_1 boundary elements and the spherical mesh. On the left we plot the analysis for the variable ϕ, on the right we depict the errors for $\frac{\partial \phi}{\partial n}$.

For the normal derivative, we observe a more irregular behavior: initial superconvergence (greater than linear) followed by a significant decrease during the final refinement step. A closer look at the error distribution (see Fig. 11) suggests that the progressive refinement of an initially cubic mesh over a sphere results in the presence of few stretched cells located at the original mesh vertices. To make things worse such nodes are located at the interface between different boundary condition regions. This situation is not improved throughout the refinements of the grid, as the angle of such stretched cells remain constant for each mesh, which results in a constant error.

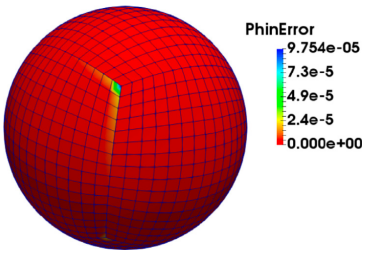

Fig. 11. Error analysis for the potential normal derivative in the sphere test case.

The second test case takes into account the truncated pyramid illustrated in Fig. 12.

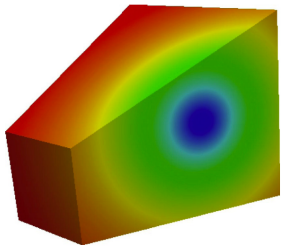

Fig. 12. Truncated pyramid geometry. The colours represent the magnitude of the exact solution ϕ.

The carried out convergence analysis is shown in Fig. 13. Even if the domain in not the simple sphere, we obtained a first order error for ϕ. In the case of the potential normal derivative we observe the occurrence of an error plateaux for the L_∞ norm in corresponding to the last refinement level considered. A possible cause can be assessed through an inspection of the local $\frac{\partial \phi}{\partial n}$ error distribution presented in Fig. 14. As can be seen, the maximum error occurs in correspondence with the edge characterised by the sharpest angle of the figure.

5. PARALLELISATION

At this point, we aim to develop an efficient parallelization strategy to achieve greater computational efficiency. Initially, we analyzed the bottlenecks in the straightforward implementation of BEM, followed by assessing the performance of the parallelization using both strong and weak scaling analysis.

5.1 Matrix Assembling and Domain decomposition

Table 1 with computational times refers to the truncated pyramid test case on a single CPU. This test case involves using the BEM to evaluate the solver's performance on a geometrically complex domain shaped like the one in Fig. 12. It requires handling sharp edges and corners with high accuracy, which poses significant computational challenges.

Table 1. Profiling on a single CPU.

Function	Time (s)
Assemble cycle	68.44
Solvetime	5.741
Post process (Gradient Recovery)	0.1645
Total time	78.6

As we can see, matrix assembly is the major part of the overall program computationally demanding, for this reason, it has to be parallelized. We used a collocation scheme to solve the boundary integral formulation. Provided that the whole computational grid is available on each processor, every line of the matrix can be assembled independently. This is because the process of matrix assembling in BEM involves calculating the interactions between each basis function and each collocation point across the entire domain. Since each row of the system matrix is basically an integral involving a collocation point and all basis functions, and can be computed independently of others, this matrix assembling task is ideal for parallel execution. For this, we utilized MPI to distribute the task

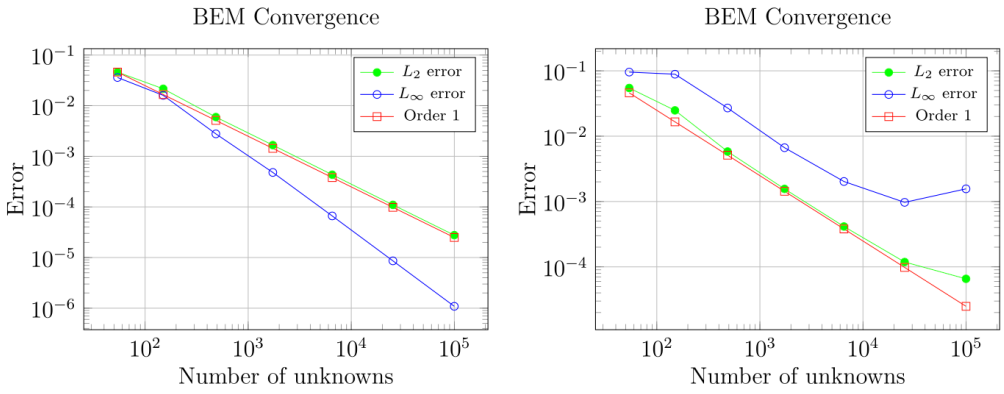

Fig. 13. Convergence analysis for the error in a mixed Dirichlet-Neumann problem using Q_1 boundary elements and the truncated pyramid mesh. On the left we plot the analysis for the variable ϕ, on the right we depict the errors for $\frac{\partial \phi}{\partial n}$.

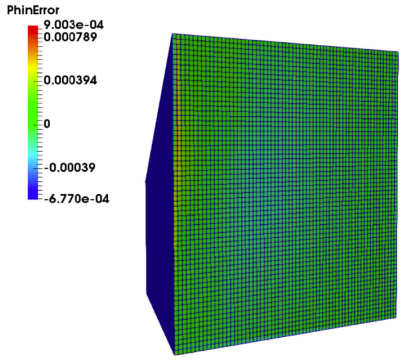

Fig. 14. Error analysis for the potential normal derivative in the truncated pyramid test case.

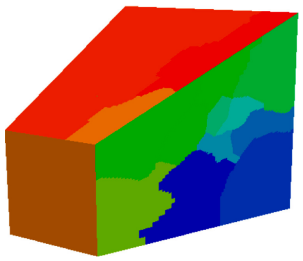

Fig. 15. Domain decomposition between 16 different MPI processors.

of assembling different rows of the matrix to multiple processors. To preserve the parallelisation efficiency, it is mandatory to achieve a proper work balance between different processors.

Fig. 15 indeed represents the domain decomposition between 16 different MPI processors. This non-trivial domain decomposition is due to the local refinement. It is important to highlight the main difference between BEM and FEM methods, indeed the final matrix in BEM is a smaller but dense matrix, and this is due to the fact that each matrix entry depends on information on the entire boundary of the domain, so every processor still needs access to the full discretisation of the boundary of the geometry, that has not to be confused with the domain decomposition that is related with the collocation points that every MPI processor has to take into account for a proper workbalance as we mentioned. It could seems that sharing the whole domain to all processor is a bottleneck, but is not, since it is just the boundary domain to be shared, in opposite with the FEM parallelisation method, in which it is mandatory not to share the whole domain. Finally the problem is solved using a parallel implementation of a preconditioned GMRES solver. The preconditioner used is derived from an Incomplete Gauss factorization, which helps reduce the computational load and iteration count necessary for solving.

5.2 Strong and Weak Scaling

The analysis proceeds by evaluating the strong and weak scalability of the previously described problem. We executed a scaling analysis with a total of 40 processors. However, the infiniband network drivers only allow for 16 MPI processors and we will see how this can influence scalability results.

Fig. 16 represents the strong scalability test. As we can see, in the real problem, it is the gradient recovery and the solve computation that led to a sub-optimal total scalability, and this is even more present in the left graph. This is due to the communication overhead among processors in the MPI environment because matrix-vector multiplication requires a lot of communication. Notice that the assembly is almost optimal because it doesn't require any communication. The super optimality in the right test is due to cache resources. We can see an increase in the total performance when considering more degrees of freedom, and this is reasonable because it implies that every MPI processor has to take into account more inner computation, leading to the communication time becoming less impactful. However, at the same time using more processors involves more communication among them. We could also notice that even if the solve speed-up is really below the ideal, this doesn't affect so much the overall performance, and this is due to the relative importance of the assembling versus solving cycles.

Moving forward with the weak scaling, we report the weak scalability analysis in Fig. 17.

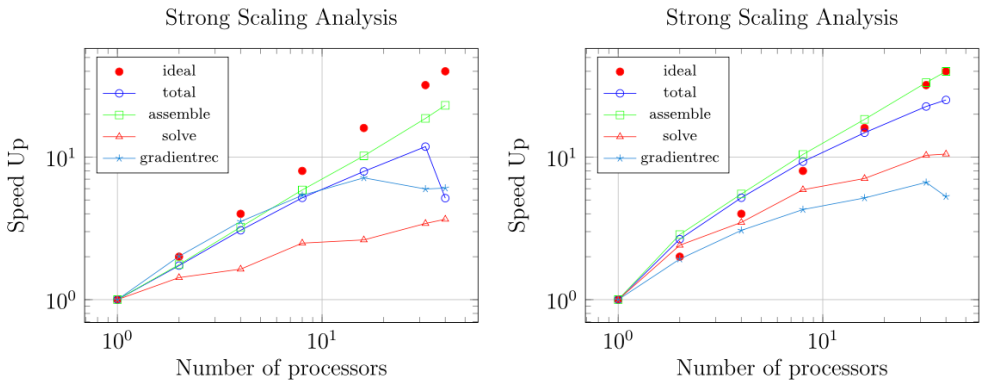

Fig. 16. Strong scalability test. In particular, on the left there is the analysis using 6534 degrees of freedom and on the right the analysis using 25350. In blue with circles we plot the total scalability, in green with squares the performances of the full matrix assembling, in cyan with stars the gradient recovery scalability, and in red with triangles the performance of the linear solver. The red dots represent the ideal speed-up that is of course linear.

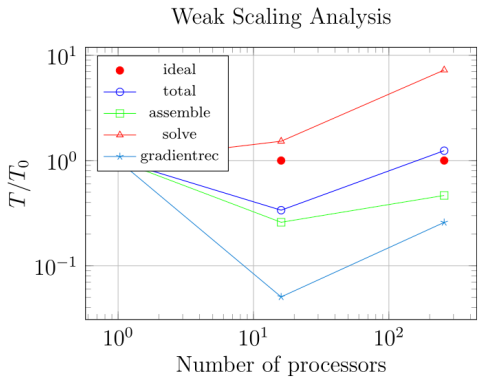

Fig. 17. Weak scalability test. In blue with circles we plot the total performance, in green with squares the timing ratio of the full matrix assembling, in cyan with stars the gradient recovery performance, and in red with triangles the performance of the linear solver. The red dots represents the ideal timings.

The analysis is carried out using up to 256 MPI processors and 25350 degrees of freedom. In particular, we wanted to ensure that each processor handles approximately the same number of matrix entries, thus maintaining a fixed workload per processor in the scaling analysis. If the workload per processor is fixed, we expect that the execution time should remain constant regardless of the number of processors, and the results highlight the desired output. Indeed, the assembling routine has a superoptimal behavior. This is probably due to the structure of the assembling cycle because each processor has to loop over all the cells N times less. However, if we increase the number of MPI processors to 256, we increase dramatically the computational costs for the linear solver, explaining the suboptimal behavior of the solving phase.

6. ACCELERATED BEM

We now focus on a more efficient BEM solver. Indeed, the cost for the solution of standard BEMs remains quadratic: $O(N^2)$, due to the dense structure of their system matrices. This fact poses limitations on the number of degrees of freedom that can be considered, both in terms of memory requirements and computational times. Exploiting the structure of the fundamental solution G to approximate the convolution integrals when two cells are well separated leads to the well-known Fast Multipole Method (FMM), which reduces the computational cost from $O(N^2)$ to $O(N)$.

6.1 Fast Multipole Method

First, we briefly introduce the FMM. The algorithm was originally derived for N-body problems and it takes into account N evaluation points (nodes) and M charges (sources) that are distinct in space. N-body problems are typical in electromagnetic applications where one is interested in computing the force exerted by the charges at the evaluation points. The computation is based upon pairwise evaluations of electromagnetic potential which coincide with the Green functions considered in this paper. This results in the evaluation of NM source-target distances. If the source and target points coincide, we recover the quadratic cost $O(N^2)$. The Fast Multipole Method is based on the consideration that an harmonic expansion of the electromagnetic potential of a single charge allows the separation into distinct factors of the effects of the source and evaluation point respectively. In principle, these factors can be computed separately only once for each source and node point, leading to a $O(N)$ computational cost. In practice, the harmonic series converges only when target and node points are well separated. Thus, one important building block of the FMM is a hierarchical space subdivision that is used to decide whether harmonic expansions or direct potential evaluations are to be used. Observing that the electromagnetic potential of a single charge coincides with the free space Green functions employed in most BEM discretisations, it is possible to modify the FMM algorithm and integrate it into a BEM solver so that we recover the N-Body problem. It's important to highlight that even if we are considering a complex geometry case and local refinement, we are still in the FMM framework. This just led to an algorithm that carefully manages the cells in order to maintain the harmonic convergence, and this is also valid for the weakly singular integrals coming from the boundary integral formulation: we have to deal with specific quadrature formulas, so in such conditions it

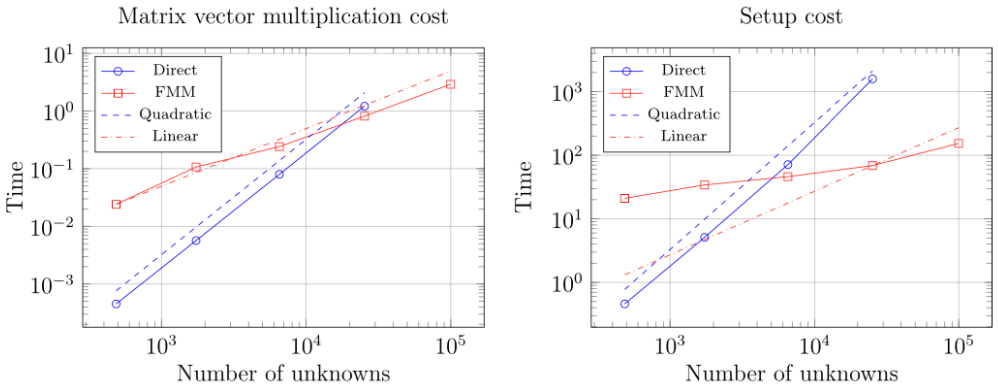

Fig. 18. Computational cost comparison between BEM and BEM-FMM. The figure on the left reports a comparison of a single matrix vector multiplication, the one on the right reports a comparison of the setting time needed by the method.

is not possible to exploit an external FMM library without implementing massive changes while the π-BEM adapted these characteristics. We proceed with a comparison between our FMM-BEM and the direct BEM formulation in terms of accuracy and computational costs.

Fig. 18 represents the computational cost comparison between BEM and BEM-FMM. On the left, it reports a comparison of a single matrix-vector multiplication, and on the right, the setting time needed by the method. As we can see, we obtain the expected linear behavior even if the break-even point is roughly located at 10^4 unknowns. For what concerns the accuracy, the FMM algorithm introduces an error associated with the truncation of the harmonic series expansions of the type

$$e_{\text{FMM}} \leq C \left|\frac{r}{z}\right|^{p+1} \leq \left(\frac{1}{2}\right)^{p+1},$$

where r represents the radius of the sphere used to generate the Multipole expansion, z is the minimum distance between such a sphere and the evaluation node, and p is the truncation order of the Multipole expansion. In FMM algorithm the expansion is only used for sources and nodes lying in *well separated* boxes.

In Fig. 19 we report the convergence rate of the FMM to the direct matrix vector product evaluation (left plot) and to the standard BEM solution (right plot). Also in this case we can see the expected behavior, as the L_2 and L_∞ errors lead below the exponential error bounds.

6.2 FMM Hybrid Parallelisation

Here we challenge the parallelisation of FMM-BEM. A key-step is the computation of the so called *Locally Essential Tree* (LET) that drives the overall MPI communication in an optimal way, but one of our implementation choices is the automatic domain decomposition which doesn't guarantee the construction of such hierarchical tree. This led us to use a multithreaded parallelisation, also due to the replicated data structures among nodes. We restrict the distributed memory parallelisation to the most demanding phases of the algorithm by enforcing that the number of threads and the number of MPI processes is kept well balanced at all times.

Before going on is necessary an overview on the key step of the Fast Multipole Algorithm: in essence, the so-called hierarchical tree is built, which is the name given to the subdivision of the domain into cells that are in turn divided into smaller cells (higher levels). The last level is represented by the leaves. The algorithm efficiently leverages the multipole expansion at various levels following an ascending phase and a descending phase in which it traverses the levels of the tree.

One of the most important parameters in the FMM is the number B of evaluation points allowed per leaf of the tree. We want to get some insights on the better tuning of the parameter B to ease the overall computational time of the BEM-FMM algorithm. Greengard and Gropp (see [5]) derived a simplified analysis of the computational costs to recover an optimal size for the block. We repeat such simplified analysis for the presented FMM in the hybrid shared (TBB) distributed (MPI) memory framework. By defining

$$\begin{aligned} N &= \text{number of unknowns},\\ B &= \text{number of unknowns in a leaf},\\ p &= \text{number of MPI processors},\\ t &= \text{number of TBB threads}, \end{aligned}$$

the FMM can be sketched as:

$$T_{\text{FMM}} = T_{\text{ascending}} + T_{\text{descending}} + T_{\text{direct}} + T_{\text{comm}}, \quad (13)$$

where $T_{\text{ascending}}$ represents the time needed by the ascending phase, $T_{\text{descending}}$ is the time needed by the descending phase, T_{direct} represents the time needed by the short range interactions, while T_{comm} is the time required by the parallel communications or synchronisations. If we expand all the terms in Eq. (13) we obtain

$$\begin{aligned} T_{\text{FMM}} =& K_1 \frac{N}{t} + K_2 \log_8\left(\frac{N}{B}\right)\frac{N}{Bt} + K_3 \log_8\left(\frac{N}{B}\right)\frac{N}{Btp} \\ &+ K_4 \frac{N}{tp} + K_5 \frac{NB}{tp} + e(B,p,t), \end{aligned} \quad (14)$$

where K_1, K_2, K_3, K_4, K_5 are experimentally constants that depend on the characteristic of the underlying computational architecture considered, and the function e represents the communication and synchronization cost. Eventually, it can be easily recover the optimal B by

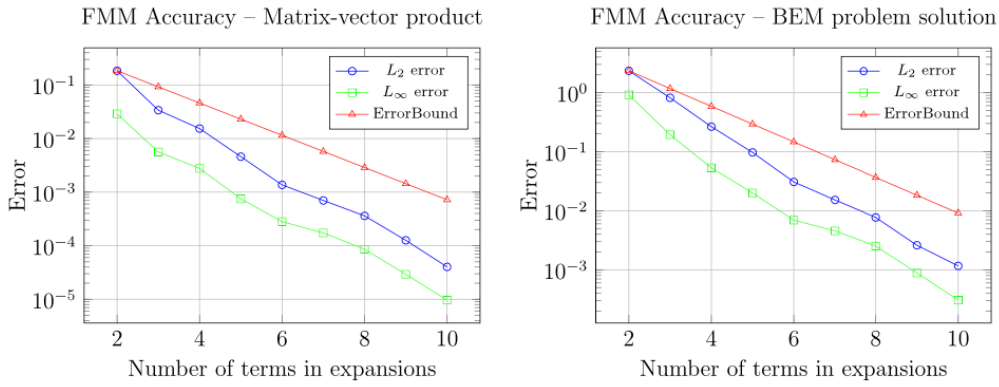

Fig. 19. Convergence analysis for the Multiple expansion using 486 unknowns on the pyramid test case. On the left we report the error of a single matrix vector multiplication, on the right we plot the error on the overall solution.

computing the partial derivative of T_{FMM} with respect to B:

$$B_{\text{opt}} = \sqrt{\frac{(K_2 + K_3/p)\log_8(N)}{K_5/p}}. \tag{15}$$

Fig. 20 represents analysis of the time needed by the overall FMM, considering 1 MPI processor and up to 10 threads per processor, with the number of evaluation nodes per block B varying from 20 to 140. In blue are reported theoretical results using Eq. (14), in green we can see the experimental results. That drastic loss of performance we see is due to the cache-misses. The result is promising since we get that $B_{\text{opt}} = 52.5440911254$.

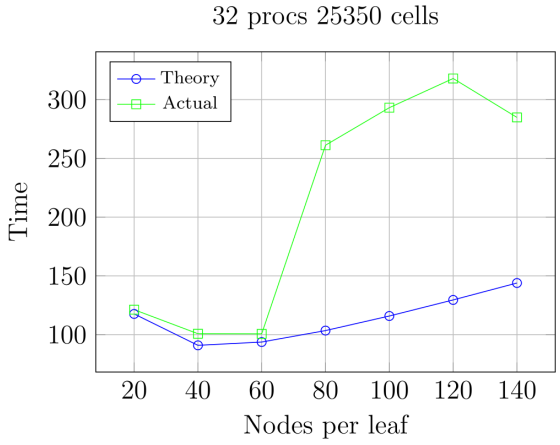

Fig. 20. Analysis of the time needed by the overall FMM.

So for the further analysis we are considering B equal to 60. We highlight that B_{opt} depends on the number of MPI processor. However, since our FMM is mostly shared memory parallelised, we are not interested in requiring many MPI processors, thus $B = 60$ is a good choice for the scaling analysis.

6.3 FMM Scaling

The strong scaling of BEM-FMM algorithm is run with hybrid TBB-MPI parallelisation up to 2 nodes, each node is composed by two 10 cores, as the previous ones. The top of Fig. 21 represents the strong scaling analysis of the

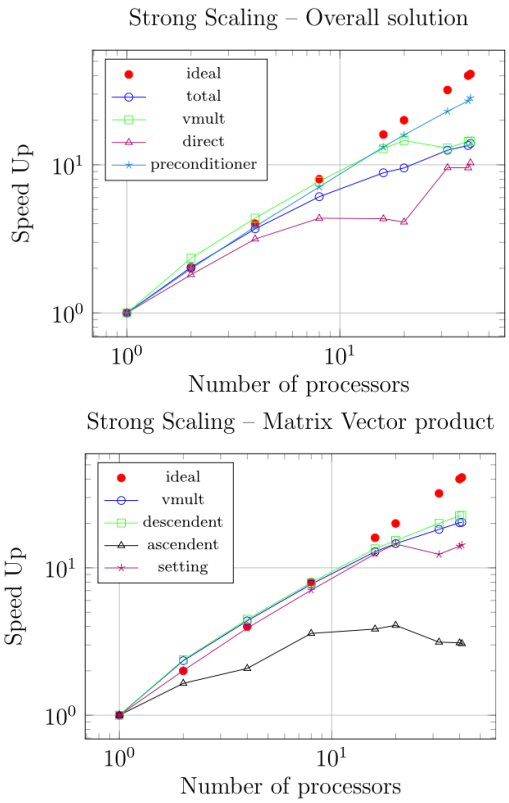

Fig. 21. Strong Scalability test using 98306 degrees of freedom.

overall algorithm while the bottom illustrates the scalability concerning a single matrix vector multiplication. These plots highlight the scalability issue for short range interactions, and the matrix vector products. Part of the performance loss is induced by the augmented number of iterations needed by the iterative solver when MPI is involved, as suggested by the bottom plot. That plot shows how our preconditioner is not optimal with rispect to the number of MPI processes. We highlight how the trade-off between computational efficiency and memory is still being studied for a better implementation of the FMM-BEM algorithm, even though we believe the main improvement should come from a better preconditioner.

7. CONCLUSIONS

In this work, after briefly introducing the history and the main advantages of the Boundary Element Method, we first analysed the scheme in all its theoretical features. Then we presented two test case solved with the π-BEM library, we faced parallelisation challenges and we showed how a fast hybrid implementation can be derived.

REFERENCES

[1] L. C. Evans. *Partial Differential Equations*, volume 19. American Mathematical Society, 1993.

[2] J. T. Katsikadelis. *The Boundary Element Method for Engineers and Scientist*. Academic Press, 2 edition, 2016.

[3] Y. J. Liu, S. Mukherjee, N. Nishimura, M. Schanz, W. Ye and A. Sutradhar and E. Pan and A. Dumont and A. Frangi and A. Saez. *Recent Advances and Emerging Applications of the Boundary Element Method*. Applied Mechanics Reviews, 64, 2011.

[4] A. Mola, N. Giuliani, L. Heltai. *π-BEM: A flexible parallel implementation for adaptive, geometry aware, and high order boundary element methods*. Advances in Engineering Software, 121:39–58, 2018.

[5] L. Greengard, W. D. Gropp. *A parallel version of the fast multipole method*, volume 20. Computers Math. Applic. 7:63-71, 1990.