A fast-multipole unified technique for the analysis of continuum mechanics problems with the boundary element methods

Hélvio de Farias Costa Peixoto

Larissa Simões Novelino

Ney Augusto Dumont

hfcpeixoto@gmail.com

larissanovelino@hotmail.com

dumont@puc-rio.br

Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

Rua Marquês de São Vicente, 225, Gávea, 22451-900, Rio de Janeiro - RJ, Brazil

Abstract. *This paper presents a fast-multipole implementation of the boundary element method for problems with many millions of degrees of freedom. It can lower the number of operations from $O(N^2)$ to $O(N \log N)$ in the iterative solution of a problem with N degrees of freedom. The memory allocation is also very small, as there is no need to store large matrices such as required by other numerical methods. The proposed implementation is based on a consistent development of the conventional, collocation boundary element method (BEM), which leads to a computationally less intensive (and conceptually more accurate) formulation for large-scale 2D and 3D problems of potential and elasticity. This formulation is especially advantageous for problems that require complicated fundamental solutions and curved boundaries. A scheme is used for expanding a generic fundamental solution about hierarchical levels of source and field poles, which is particularly advantageous to make the fast multipole technique seamlessly applicable to different fundamental solutions. The hierarchical tree of poles is built upon a topological concept of superelements inside superelements, which in part circumvents the need of evaluating geometrical distances between elements. The formulation is assessed and validated in terms of some simple, although very large, 2D potential problems with general geometry and topology for constant, linear and quadratic elements. Since iterative solvers are not addressed in this first step of numerical simulations, an isolated efficiency assessment of the implemented fast multipole technique becomes possible.*

Keywords: *Boundary elements, fast multipole method, numerical methods*

1 INTRODUCTION

The fast multipole method (FMM), as developed by Greengard and Rokhlin (1987), was elected one of the top ten algorithms of the 20th century (Dongarra & Sullivan, 2000). Although it has been initially conceived for particle simulation problems involving coulombic and gravitational fields, it also turned out efficient in the solution of boundary integral equations (Nishimura, 2002). The FMM, combined with an iterative solver such as the GMRES, can speed up the complete solution time of a problem with N unknowns from order $O(N^2)$ to $O(N \log N)$, while requiring computer storage that is only a small fraction of what would be allocated for a different numerical method.

Liu (2009), who also claims that an overall reduction to $O(N)$ may be possible, presents a thorough description of the FMM as applied to different types of BEM problems. A short review of the method is given by Liu *et al.* (2011), a comprehensive review is given by Nishimura (2002) and a tutorial has been prepared by Liu & Nishimura (2006), to mention but a few key references.

The present research work is part of the Ph.D and M.Sc. works of the two first authors (Peixoto, 2014; Novelino, 2015), and is mainly concerned with the application of the FMM to problems with generally curved boundaries, in a framework that is almost completely independent from the underlying fundamental solution. The basic concept of the FMM, with the expansion of the fundamental solution about successive layers of source and field poles, is described in a compact algorithm that is more straightforward to lay out and promises to be more efficient than the ones available in the technical literature. This paper is a sequel of the References (Dumont & Peixoto, 2014; Peixoto *et al.*, 2015) and brings as main contribution the implementation for curved boundary elements. Owing to space restrictions only the basic aspects of the proposed implementations (Novelino, 2015) are described in the present paper, with a more extensive manuscript being prepared for publication in the near future.

2 Proposed FM algorithm for a general, complex function

The following basic definitions are used in the present developments to represent a general complex function $f(z)$:

- $z - z_0$ = difference between the source point z_0 and the field point z.
- z_{c^k}, $k = 1, 2, ...n_c$: hierarchical levels of poles about which the fundamental solution will be successively expanded for the field point z (then, by definition, $z_{c^0} \equiv z$).
- z_{L^l}, $l = 1, 2, ...n_L$: hierarchical levels of poles about which the fundamental solution will be successively expanded for the source point z_0 (by definition, $z_{L^0} \equiv z_0$).

The above definitions of a pole z_{c^k} that is *close* (lower case c) to the field point z and of a pole z_{L^l} that is *local* (upper case L) to the source point z_0 follow the notation introduced by Liu (2009). In the following developments, each *close* pole z_{c^k} and each *local* pole z_{L^l} are actually array representations of different hierarchical levels of poles, as illustrated in Fig. 1, where the attached superscripts (here omitted, for simplicity) denote an individual pole in the array.

The expression of a generic fundamental solution for 2D problems is initially expanded about the *close* pole $z_{c^{n_c}}$ (of highest level, as developed next) using n terms:

$$f(z-z_0) = \sum_{i=0}^{n} \frac{1}{i!}(z-z_{c^{n_c}})^i D^{(i)} f(z_{c^{n_c}}-z_0) + O(z-z_{c^{n_c}})^{n+1} \tag{1}$$

where $D^{(0)}f(z) = f(z)$ and $D^{(i)}f(z) = \partial^i f(z)/\partial z^i$.

The truncated form of Eq. (1) is conveniently written as

$$f(z-z_0) = \sum_{i=1}^{n+1} \frac{1}{(i-1)!} P_i(z-z_{c^{n_c}}) Q_i(z_{c^{n_c}}-z_0) \tag{2}$$

for truncation order $O(z-z_{c^{n_c}})^{n+1}$ and with the arrays of functions $P(z)$ and $Q(z)$ defined for a generic argument z as

$$P(z) = [\ 1 \quad z \quad z^2 \quad z^3 \quad \cdots \quad z^{n+1}\ ] \tag{3}$$

$$Q(z) = [\ f(z) \quad \frac{\partial f(z)}{\partial z} \quad \frac{\partial f^2(z)}{\partial z^2} \quad \frac{\partial f^3(z)}{\partial z^3} \quad \cdots \quad \frac{\partial f^{n+1}(z)}{\partial z^{n+1}}\ ] \tag{4}$$

As a matter of illustration, one may want to evaluate in the scheme on the upper left of Fig. 1 the influence $f\,(z(\xi)-z_0^4)$ of a source attached to the nodal point numbered as z_0^4 on a certain point $z(\xi)$ along a given boundary segment. This might be done either directly or, in the present context, via an expansion of $z(\xi)$ about the pole z_c such as in Eq. (2). The evaluation for the influence of $N=6$ source points $z_0^1 \cdots z_0^6$ on $N=6$ field segments $z(\xi)$ is represented on the upper left of Fig. 1. The numerical effort for the complete evaluation of $\int_{\Gamma_f} f\,(z(\xi)-z_0)\,\mathrm{d}\Gamma$ along Γ_f, as indicated in the figure, for all source points is then proportional to N^2 evaluations of the type $f(z-z_0) \equiv Q_1(z-z_0)$. However, the evaluation in terms of Eq. (2) – dash lines for $Q(z)$ and solid green lines for $P(z)$ – requires an effort proportional to $(n+1)N$ evaluations of $P_i(z)$ plus $(n+1)N$ evaluations of $Q_i(z)$, where n is the number of expansion terms:

$$\int_{\Gamma_f} f\,(z(\xi)-z_0)\,\mathrm{d}\Gamma = \sum_{i=1}^{n+1} \frac{1}{(i-1)!} Q_i(z_{c^{n_c}}-z_0) \int_{\Gamma_f} P_i(z-z_{c^{n_c}})\mathrm{d}\Gamma, \tag{5}$$

keeping in mind that the evaluation of a function $Q_i(z)$ (which involves transcendental functions, in general) may require by far more computational effort than the evaluation of $P_i(z)$, which is just a binomial. However, the above illustration is just an ideal case for z_c far from the source points and close to the field boundary segment Γ_f, which does not happens in practice.

For the sake of generality, let the derivatives $D^{(i)}f(z_{c^{n_c}}-z_0)$ be also expanded for the source point z_0 about the *local* point $z_{L^{n_L}}$ (of highest level, as to be also shown subsequently) using m terms:

$$D^{(i)}f(z_{c^{n_c}}-z_0) = \sum_{j=0}^{m} \frac{1}{j!}(z_{L^{n_L}}-z_0)^j D^{(i+j)} f(z_{c^{n_c}}-z_{L^{n_L}}) + O(z_{L^{n_L}}-z_0)^{m+1} \tag{6}$$

Substituting for $D^{(i)}f(z_{c^{n_c}} - z_0)$ in Eq. (1) according to above, it results

$$\begin{aligned} f(z - z_0) = & \sum_{i=0}^{n} \tfrac{1}{i!}(z - z_{c^{n_c}})^i \sum_{j=0}^{m} \tfrac{1}{j!}(z_{L^{n_L}} - z_0)^j D^{(i+j)} f(z_{c^{n_c}} - z_{L^{n_L}}) \\ + & \; O(z - z_{c^{n_c}})^{n+1} + O(z_{L^{n_L}} - z_0)^{m+1} \end{aligned} \tag{7}$$

As given in this equation, it is in principle indifferent which expansion is carried out first. The indicated sum of order of errors stresses the relevancy of the distances between the respective points of expansion for the accuracy of the problem. This accuracy can be assessed case by case, but in general the higher derivatives of the fundamental solution tend rapidly to zero when evaluated for large arguments, and the truncation errors turn out to be proportional to $|(z - z_{c^{n_c}})/(z - z_0)\,|^{n+1}$ and $|(z_{L^{n_L}} - z_0)/(z - z_0)\,|^{m+1}$.

The truncated form of Eq. (7) is conveniently written as

$$f(z - z_0) = \sum_{i=1}^{n+1} \frac{1}{(i-1)!} P_i(z - z_{c^{n_c}}) \sum_{j=1}^{m+1} \frac{1}{(j-1)!} P_j(z_{L^{n_L}} - z_0) Q_{i+j-1}(z_{c^{n_c}} - z_{L^{n_L}}) \tag{8}$$

for truncation order given by $\max\left(|(z - z_{c^{n_c}})/(z - z_0)\,|^{n+1}, |(z_{L^{n_L}} - z_0)/(z - z_0)\,|^{m+1}\right)$.

Equation (2) is undoubtedly more economical than Eq. (8) and is in fact the simplest and fastest way of evaluating a function such as in Eq. (5). However, Eq. (8) is given as the starting point for a general procedure that leads to a computationally fast and economical evaluation of a given fundamental solution $f(z - z_0)$ for a very large number of source points z_0 and of field points z by means of an approximate expression that can be as accurate as required. The expansion about successive levels of poles is dealt with in the next section.

2.1 Expansion of $P_i(z - z_{c^k})$ and $P_j(z_{L^{n_l}} - z_0)$ about successive poles

The next developments are illustrated on the upper right of Fig. 1 and shall lead to the complete generalization of the fast multipole technique. This figure shows three levels of close poles (with four, two and one poles on each level) and two levels of local poles (with two and one poles on each level). This generalization is at least in part required because, for a closed boundary, $z^1_{c^3}$ may be sufficiently far only from a certain subboundary of Γ_s. Then, hierarchical level of poles must be implemented, which are sufficiently far from certain subboundaries while building up the contribution for higher-order poles.

The superscripts n_c and n_L in Eq. (8) turn out to be the highest levels of field and source poles for an expansion to be carried out hierarchically about successive layers of poles. Moreover, the binomials $P_i(\cdot)$ introduced in Eq. (8) are actually to be directly evaluated according to their definition in Eq. (3) only when the argument refers to a coordinate difference for two consecutive levels of poles, such as $z_{c^{k-1}} - z_{c^k}$, $k = 1, \cdots, n_c$ in the definition of $P_i(z - z_{c^k})$, or $z_{L^l} - z_{L^{l-1}}$, $l = 1, \cdots, n_L$ in the definition of $P_j(z_{L^{n_L}} - z_0)$. When the argument of $P_i(\cdot)$ refers to poles that are not on consecutive layers, it is expanded in terms of a lower-level pole $z_{c^{l-1}}$ exactly as

$$P_i(z - z_{c^l}) = \sum_{j=1}^{i} C_{j,i+1-j} P_j(z - z_{c^{l-1}}) P_{i+1-j}(z_{c^{l-1}} - z_{c^l}) \tag{9}$$

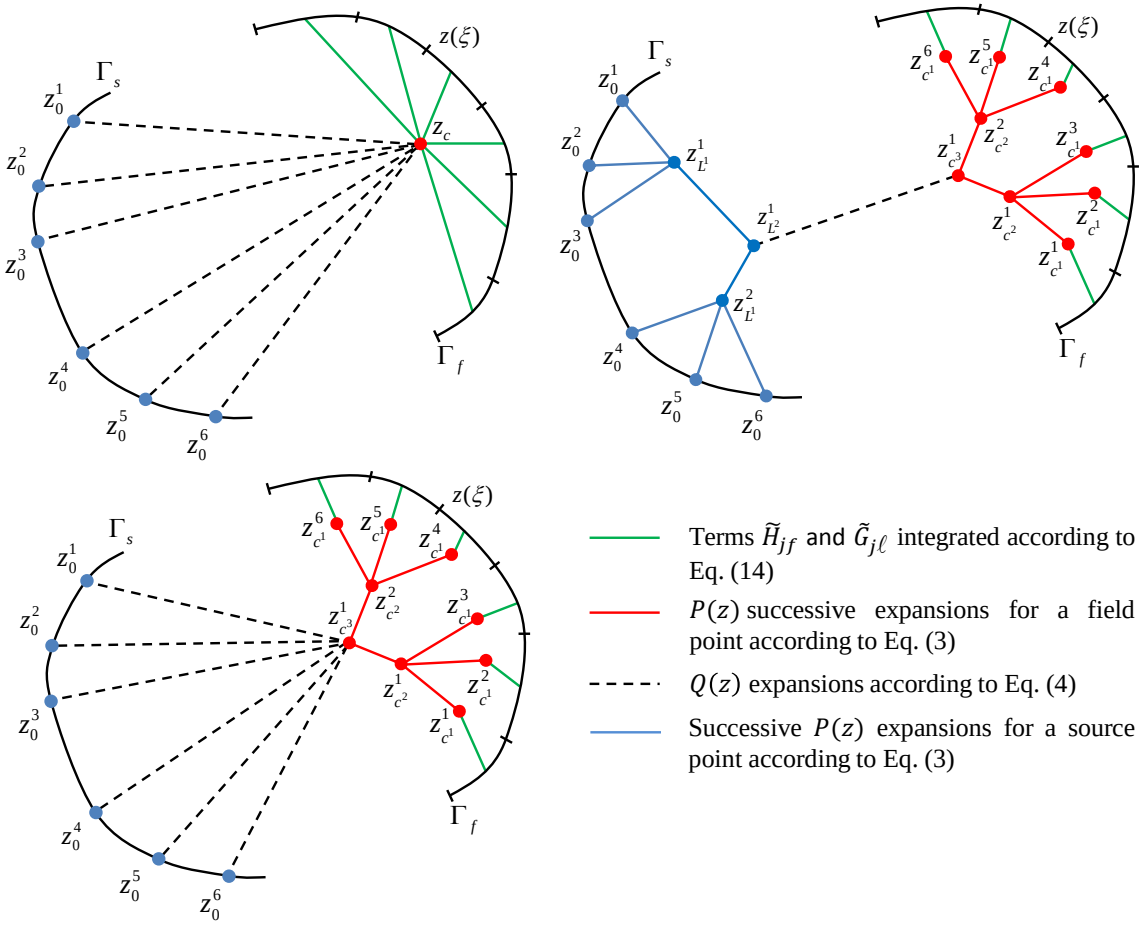


Figure 1: Schematic representations of expansions about field and source poles.

where

$$C_{ij} = \begin{cases} 1 & \text{if } i = 1 \text{ or } j = 1 \\ C_{i-1,j} + C_{i,j-1} & \text{else} \end{cases} \tag{10}$$

In Eq. (9), $P_{i+1-j}(z_{c^{l-1}} - z_{c^l})$ is defined as in Eq. (3), since it refers to two consecutive poles. On the other hand, $P_j(z - z_{c^{l-1}})$ is recursively evaluated according to the same Eq. (9) until the lowest level $P_j\,(z - z_{c^1})$ is arrived at and Eq. (3) becomes directly applicable. With this recursive approach, the binomials $P_i(z - z_{c^{n_k}})$ in Eq. (8) – and, similarly, $P_j(z_{L^{n_l}} - z_0)$ – are ultimately expressed in terms of arguments given as differences of poles on two consecutive levels.

As shown, $f(z - z_0)$ is expanded in terms of successive arrays of source poles z_{L^l} as well as field poles z_{c^k}. The indicated poles z_{L^l} and z_{c^k} are representative of arrays of poles corresponding to hierarchical expansion levels l and k. The expansion ends up with a series of products of binomials $P_i(z - z_{c^{n_k}})$ and $P_i(z_0 - z_{L^{n_l}})$, which are independent from the complexity of a problem's fundamental solution, multiplied by functions $Q_i(z_{L^{n_l}} - z_{c^{n_k}})$ that, although eventually cumbersome to obtain numerically, refer to the least number of higher-order pairs of pole arrays $z_{L^{n_l}} - z_{c^{n_k}}$. In the illustration on the upper right of Fig. 1, there is only one pair of poles $z^1_{L^2} - z^1_{c^3}$ for which, in the case, $Q_i(z^1_{L^2} - z^1_{c^3})$ needs to be evaluated.

The definition of the functions $P_i(z)$ and $Q_i(z)$ follows closely the definitions (3.12) of I and O by Liu (2009), although without the factorials that are present there. Observe the mnemonic appeal of P (for polynomials, actually binomials) and Q [for quotients, as in the case of the expansion of a logarithm – the simplest fundamental solution conceivable (Dumont & Peixoto, 2014; Peixoto *et al.*, 2015)]. Although these latter functions may be computationally intensive to evaluate, they are only needed for the array of poles represented by z_{L^l} interacting with the array of poles represented by z_{c^k}. Then, the evaluation of $Q_i(z_{L^{n_l}} - z_{c^{n_k}})$ ends up, depending on the numerical implementation, orders of magnitude less intensive than the direct evaluation of $f(z - z_0)$ for all source and field points – and this is the basis of the fast multipole method.

The hierarchical expansions described above are next implemented in a boundary element code. It has turned out that the hierarchical development of source poles is not of advantage, at least as implemented by the authors (Novelino, 2015), so that the scheme shown on the lower left of Fig. 1 is the actual procedure to be assessed in the numerical examples.

3 Numerical implementation

For a potential problem, the double-layer potential matrix $\mathbf{H}$ and the single-layer potential matrix $\mathbf{G}$ of the conventional boundary element method are given for the compatibility equation

$$\mathbf{Hd} = \mathbf{Gt} \tag{11}$$

(domain sources not considered, for simplicity) of the nodal potentials $\mathbf{d}$ and the boundary nodal flux attributes $\mathbf{t}$ in terms of the boundary integrals (Novelino, 2015)

$$\begin{aligned} H_{sf} &= \int_\Gamma q^*_{ks}(z - z_0)\eta_k(z)\; u_f(z)\, d\Gamma(z), \\ G_{s\ell} &= \int_\Gamma u^*_s(z - z_0) t_\ell(z)\, d\Gamma(z), \end{aligned} \tag{12}$$

where $q^*_{ks}(z - z_0)$ and $u^*_s(z - z_0)$ are the complex expressions of the flux and potential fundamental solutions of the potential problem – which have global support – and $\Gamma(z)$ is the integration boundary. The subscript s refers to a given $source$ node (at which the unit point source of the singular fundamental solution is applied) and the subscripts f (which stands for $field$) and ℓ (also a field reference) indicate the nodes to which the potential-interpolation function $u_f(z(\xi))$ and the flux-interpolation function $t_\ell(z(\xi))$ are referred. In the above equation and in the following, repeated indices mean summation, for $k = 1 \cdots D$, where $D = 2$ or 3, depending on whether one is dealing with a 2D or a 3D potential problem. The Cartesian components of the unity outward vector to $\Gamma(z)$ are denoted by $\eta_k(z)$ and $u_f(z)$ formally comes from the piecewise (with local support) interpolation of the potential $u(z)$ along the boundary: $u(z) = u_f(z)d_f$, where d_f are the nodal potentials. In an usual isoparametric representation, displacement and geometry are represented identically for each one of their Cartesian components, so that $u_f(z) \equiv u_f(z(\xi))$ are actually approximated by polynomial shape functions $N_f(\xi)$, for a given boundary segment, with $\xi \in (0, 1)$ or $\xi \in (-1, 1)$, depending on the preferred parametric representation. For $z \to z_0$, strong and weak singularities must be taken care of in the evaluation of $\mathbf{H}$ and $\mathbf{G}$. However, this issue can be disregarded in the present developments, which are actually only concerned with what occurs when z and z_0 are very far

from each other. The Jacobian used in the definition of $\eta_j(z)$ cancels out with the Jacobian of $\mathrm{d}\Gamma(z) = |J(z)|\,\mathrm{d}\xi$, in terms of the parametric variable ξ. Then, expanding $q^*_{ks}(z - z_0)$ according to Eq. (8) ends up with the evaluation of a polynomial integral corresponding to the first of Eqs. (12).

For the single-layer potential matrix $\mathbf{G}$, it is proposed that the usual interpolation polynomials t_ℓ of boundary normal flux in Eq. (12) be replaced according to

$$t_\ell \leftarrow t_\ell |J|_{(\text{at } \ell)}/|J| \tag{13}$$

where $|J|_{(\text{at } \ell)}$ is the value of the Jacobian at the field point characterized by the subscript ℓ (Dumont, 2010; Dumont & Aguilar, 2012). Nothing changes formally in the developments of the BEM, except that the numerical integration of the matrix $\mathbf{G}$ becomes much easier and actually more consistent as compared to proposed implementations given in the technical literature (Brebbia *et al.*, 1984). In fact, $|J|$ cancels out in the product $t_\ell \mathrm{d}\Gamma$ in Eq. (12) for t_ℓ defined as suggested, and the integrand of $\mathbf{G}$, in the frame of a fast multipole implementation, also becomes a polynomial independently from the assumed kernel u^*_s. In a practical implementation, the functions $t_\ell(z(\xi))$ are replaced with the same shape functions $N_\ell(\xi)$ used to represent potentials, although the context differs conceptually, as $\mathbf{G}$ in general becomes a rectangular matrix (for elements other than constant elements, ℓ spans a larger number of nodes than f).

As proposed, let $z = N_f(\xi) z_f$ be the complex geometric representation of a boundary segment, for a 2D implementation, expressed in terms of the nodal points z_f and the shape functions $N_f(\xi)$, which are also used to interpolate potentials as well as normal fluxes, as described above. Then, the only integrations that need to be carried out for a given boundary segment Γ_{seg} in the evaluation of the matrices of Eq. (12) in the framework of the fast multipole developments represented by Eq. (8) are for terms $\tilde{H}_{jf}$ and $\tilde{G}_{j\ell}$ that are parts of $\mathbf{H}$ and $\mathbf{G}$, defined similarly to Eq. (5) as

$$\begin{aligned}
\tilde{H}_{jf} &= \int_{\Gamma_{seg}} P_j\,(z - z_c)\, z' N_f(\xi)\, d\Gamma = \int_0^1 P_j\,(N_k(\xi) z_k - z_c)\, N'_l(\xi) z_l N_f(\xi)\, d\xi \\
\tilde{G}_{j\ell} &= \int_{\Gamma_{seg}} P_j\,(z - z_c)\, N_\ell(\xi)\, d\Gamma = \int_0^1 P_j\,(N_k(\xi) z_k - z_c)\, N_\ell(\xi)\, d\xi,
\end{aligned} \tag{14}$$

where, recalling, repeated indices mean summation, in this case covering all shape functions $N_k(\xi)$ of a boundary segment representation, and a prime $(')$ means derivative with respect to ξ. Since the Jacobian inherent to $\mathrm{d}\Gamma$ in the above integrals cancels out and the integrands are polynomials, it is irrelevant whether $\xi \in (0, 1)$, as indicated, or $\xi \in (-1, 1)$ is used as the normalized interval. In the first row of the above equation, the term $N'_l(\xi) z_l$ comes from the complex representation of $\eta_k(z)$ along the boundary segment. The subscript j refers to the number of terms in the adopted expansion of Eq. (9), whereas f and ℓ refer to each of the boundary element nodes. In the present implementation for 2D problems (Novelino, 2015), either linear, quadratic or cubic boundary elements are represented in the same code. Constant elements are considered in a separate implementation. The main feature of the proposed implementation of the FMM is that the arrays $\tilde{H}_{jf}$ and $\tilde{G}_{j\ell}$ given above become ultimately expressed as polynomials of the differences of the nodal coordinates of the boundary segment to the most immediate pole,

$$\Delta_i = z_i - z_c, \text{ with } i = 1 \ldots o_e + 1, \tag{15}$$

where $o_e = 1, 2, 3$ for linear, quadratic or cubic boundary elements. As an illustration for quadratic elements (3 nodes),

$$\tilde{H}_{jf} = \sum_{k=1}^{j+1} \sum_{l=1}^{k} A_{fjkl} \Delta_1^{j-k+1} \Delta_2^{k-l} \Delta_3^{l-1}, \quad \tilde{G}_{j\ell} = \sum_{k=1}^{j+1} \sum_{l=1}^{k} B_{\ell jkl} \Delta_1^{j-k+1} \Delta_2^{k-l} \Delta_3^{l-1}, \qquad (16)$$

where the polynomial coefficients $A_{fjkl} \equiv A_{fjlk}$ and $B_{\ell jkl} \equiv B_{\ell jlk}$ can in principle be previously evaluated for each node f or ℓ and each series term j they refer to (Dumont & Peixoto, 2014; Peixoto, 2014; Novelino, 2015) and stored. The coefficients $B_{\ell jkl}$ for up to 3 terms, for instance, are given by Dumont & Peixoto (2014).

An alternative, simpler procedure has turned out to be the symbolic evaluation of the polynomial terms of Eq. (14) using $\mathrm{Maple}^{\mathrm{TM}}$ for as many terms as desired, and the subsequent storage as pre-assigned polynomial functions of Δ_i in the implemented C++ code (Novelino, 2015). There is also the possibility of evaluating the indicated integrals in Eq. (14) exactly by using Gauss-Legendre quadrature, which may become a faster procedure. However, the code would only be efficient by using different numbers of integration points for different terms of the series expansion. This is being currently assessed.

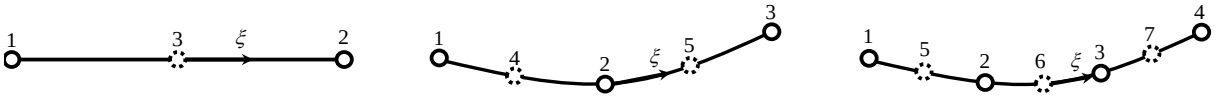
The expression of the array $Q(z)$ in Eq. (4) is given in the technical literature for the most common fundamental solutions (Liu, 2009) and need not be reproduced here. However, as an illustration, the Westergaard stress function for a rotated, elliptic semicrack (Dumont & Mamani, 2011) may be developed as

$$Q(Z) = \frac{-1}{2\pi} \left[\begin{array}{ccc} \frac{\pi Z}{2} + 1 - (1 - Z^2)F & FZ + \frac{1}{Z} + \frac{\pi}{2} & \frac{F - 1/Z^2}{1 - Z^2} \quad \frac{3ZF - (5Z^2 - 2)/Z^3}{(1 - Z^2)^2} \end{array} \right], \qquad (17)$$

where $F = \ln\left(-\left(1 + \sqrt{1 - Z^2}\right)/Z\right)/\sqrt{1 - Z^2}$ and $Z = z\mathrm{e}^{-\theta\sqrt{-1}}/a$ for the semicrack length a and inclination θ.

4 A unified algorithm for hierarchical mesh refinement

Figure 2 shows the schemes of three different elements considered in the present algorithm, as taken out of a general 2D mesh corresponding to a given level of refinement. The algorithm is also implemented for constant elements. A corresponding algorithm has also been implemented for 3D problems (Dumont & Aguilar, 2011). The input data for the hierarchical mesh refinement



a) linear element b) quadratic element c) cubic element

Figure 2: Schemes for splitting a general element into two sub-elements.

are:

- o_e = either 1, 2 or 3, which defines the element type (t_e = 2, 3 or 4).
- nee: Initial number of elements of the initial level.
- nne: Number of nodes of the initial level.

- nv: Number of additional levels of mesh refinement ($nv = 1$, for example, indicates that the structure will be refined once).
- Table $Coord\,[\,]$, with nee node coordinate entries in the format $[x, y]$, which are the initial nodes of the mesh structure to be refined. This table is successively expanded, as new nodes are added during the mesh refinement.
- Table $Inc\,[k]\,[\,]$, where $k = 1$ refers to the initial, first level local-to-global nodal incidence of the input mesh and the second entry are nee arrays, each one with t_e global node numbers of the elements. Arrays $Inc\,[k]\,[\,]$, for $k = 2 \ldots nv + 1$, will be generated as a result of the mesh refinement.

The output data are a generalization of the input data, which now refer to $nv + 1$ levels of refinement:

- $nel\,[k = 1 \ldots nv + 1]$Number of elements on each one of the $nv + 1$ levels.
- $nnl\,[k = 1 \ldots nv + 1]$Number of nodes on each one of the $nv + 1$ levels.
- Table $Coord\,[\,]$with $nnl[nv + 1]$ node coordinate entries, which correspond to the input values plus the coordinates of the generated nodes.
- A table $Inc\,[k = 1 \ldots nv + 1]\,[\,]$with $nv + 1$ levels of arrays of local-to-global node incidences. The second entry are $nel\,[k = 1 \ldots nv + 1]$ arrays, each one with t_e global node numbers of the elements.

The algorithm, which is not shown in the present short paper, generates $nv + 1$ levels of refined meshes, including the initial one, which is referred to as level 1. The number of elements on any level is two times the number of elements of the preceding level. However, the number of nodes is not known in advance and is evaluated inside the procedure.

5 A unified algorithm for pole expansions

In the conventional BEM, as given in Eq. (11), the fundamental solutions $q^*_{js}(z - z_0)$ and $u^*_s(z - z_0)$ shown in Eq. (12), which have global support, can in principle be expanded according to Eqs. (8) and (9) about arbitrary (but feasible, depending on the mesh discretization) numbers of source poles z_{L^l} as well as of field poles z_{c^k}. In the proposed FMM algorithm, a pole expansion is combined with the mesh refinement introduced (but not detailed) in the previous Section, in such a way that the element (be it linear, quadratic or cubic) dictates the expansion. The scheme is applicable to constant elements, although in a separate, simpler, code, and this is shown in Fig. 3 to illustrate that a pole expansion can be undertaken either at each mesh split (a), at every two mesh splits (b), or at every four mesh splits (c) and so on, generating in the illustration either two, four or eight children poles.

The complete code combines the algorithm of Section 4 with the compact, self-recurring expansions of Eqs. (8) and (9) and is too convoluted to be shown here (Novelino, 2015). Given an initial, rough, mesh configuration, as illustrated in Fig. 4, field as well as source pole expansions can be undertaken along with the process of dyadic refinement of the mesh, as illustrated in Fig. 2, so that, for known values of $\mathbf{d}$ and $\mathbf{t}$ in Eq. (11), line multiplications (for instance, the multiplication $\mathbf{H}_s\mathbf{d}$ of a given line s of matrix $\mathbf{H}$ corresponding to a source node) are carried out in a faster way that requires usually only $\log(N)$ instead of N operations, where N is the vector dimension of $\mathbf{d}$. It is worth noticing that, since the outermost loop in the implemented

algorithm refers to the boundary elements to which the field poles are attached, all line multiplications $\mathbf{H}_s\mathbf{d}$, $s = 1 \ldots N$, are evaluated concomitantly. Moreover, the predicted number of operations $\log(N)$ for each source point $i = 1 \ldots N$ is arrived at rather experimentally than theoretically, in the present developments, and happens to coincide with the literature results. In fact, this has been corroborated for all 2D numerical experiments carried out so far and as illustrated in Sections 7 and 8.

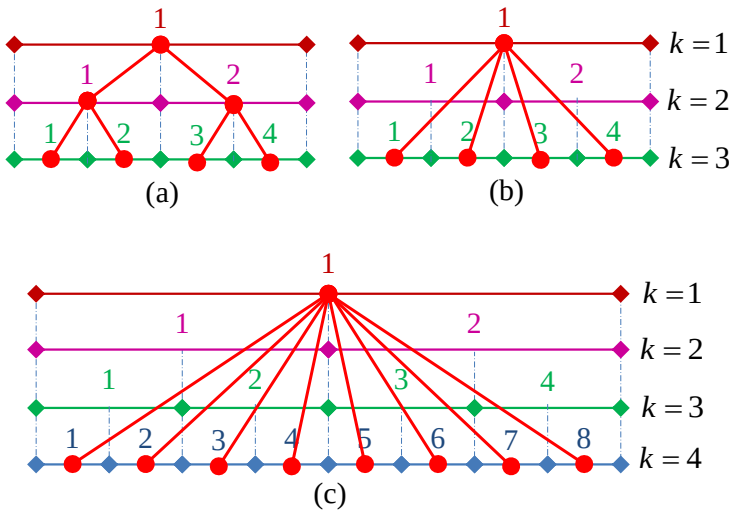


Figure 3: Schematic pole expansions using numbers of children poles $n_c = 2, 4$ or 8 (constant elements).

Basic entries of the developed FMM algorithm are, for a given mesh and a given boundary element type, the numbers of field and source poles as well as the numbers of children poles to be used at each expansion, according to Fig. 3. The code developed so far by the authors is still a long way from being considered an accomplished work. Our goal is to rethink and carefully assess every step towards the development of a complete FMM procedure applicable to generally curved elements and for general underlying fundamental solutions.

The complete analysis of a problem in the frame of a FMM procedure presupposes the implementation of a direct solver (GMRES, for instance) and the use of a robust preconditioner. In order to efficiently assess the performance of the proposed FMM algorithm, the complete solution of a problem has been postponed to our last step of developments (when most possibly a well-proven solver – such as the one given by Liu (2009) – will be adopted). Moreover, the geometric distance between source and field points is at the moment replaced by the topological distance obtained in the mesh refinement introduced in Section 4 and used as illustrated in Fig. 3. This is not correct for domains that are too irregular. There is already a work in progress to insert a given problem's domain in a mesh of hierarchical squares, making use of developments done for a 3D BEM implementation (Dumont & Aguilar, 2011). This concept is not far from the developments of the technical literature (Liu, 2009), only that we work with Boolean algebra rather than with geometric distances to evaluate how far two poles are from each other. A possibly more advantageous approach would arise from the combination of geometrical and topological distances.

Our initial assessments have shown that, for the solution of Eq. (11), the expansion of the source poles does not contribute to the efficiency of the proposed FMM algorithm (Novelino,

2015). As a result, the following numerical assessments only consider expansion of the field poles.

6 Brief outline of the implemented FM algorithm

Let $k_0 = 1$ be the first refinement level for the hierarchical calculations. Referring to the definitions given in Section 4, the following algorithm is implemented, as a brief outline from the developments by Novelino (2015).

FMM algorithm For $i_e = 1, \cdots, nel[k_0]$:

- Create the first adjacency structure for the macroelement i_e by calling the routine

  Routine_adjacencies_0(k_0, i_e)

- Call the recursive procedure

  Routine_adjacencies(k_0, i_e),

  which creates the adjacency structure on all subsequent levels and carries out all necessary geometric calculations, which now include the fast-multipole developments.

  After returning from the above procedure, which is recursive for successive levels $k = 2, \cdots, nv + 1$, all geometric data related to the boundary elements obtained from the successive splitting of the macroelement i_e will be made available. This includes the construction of the arrays of finite parts as well as of discontinuous terms of the matrix $\mathbf{H}$.

- Call the recursive procedure that generates the data related to the source point and leads to all necessary numerical evaluations. This procedure checks for adjacency of i_{e_s} (element to which a source node is attached) and i_e on successive refinement levels. If there is no adjacency on a certain level, all evaluations are carried out using the fast-multipole technique. If the finest mesh refinement is reached, evaluations are carried out conventionally. This is done in the frame of the following routine, which is recursive and keeps calling itself for increasingly refined levels:

  For $i_{e\_s} = 1, \cdots, nel[k_0]$: *call Routine_source*(k_0, i_{e_s}, i_e)

- Conclude the evaluations for the diagonal coefficients of $\mathbf{H}$.

7 A basic assessment for constant elements

Although the proposed algorithm is for general curved elements, the simplicity of the constant element enables the assessment of the basic features of the algorithms introduced in Sections 3 – 6. The problem to be considered is the simplest case of linear potential in a square domain, as shown on the left of Fig. 4, whose edges are progressively discretized with up to $2^{24} = 16,777,216$ constant elements, as indicated in the horizontal axis of the graph on the left of Fig. 5. This graph gives the execution time for the evaluation of Eq. (11) for the prescribed analytical values of potentials $\mathbf{d}$ and normal gradients $\mathbf{t}$, as implemented in a C++ code and running on a desktop computer (i7-4770 CPU 3.4GHz, 16GB RAM in Windows® 7). For the sake of comparison, the execution time required for evaluating Eq. (11) in the frame of the conventional BEM is also plotted and shown to be proportional to N^2 (dash-dot line). The

plotted execution times of Eq. (11) are for the field poles successively expanded with number of children $n_c = 2$ (first illustration of Fig. 3) and the number n of terms in the series of Eq. (8) equal to 5, 9 or 13. The time results are in all cases quite the same and shown to be proportional to $N \log N$ (dash line). The highest computational time for the fast multipole implementations was not larger than 15 minutes.

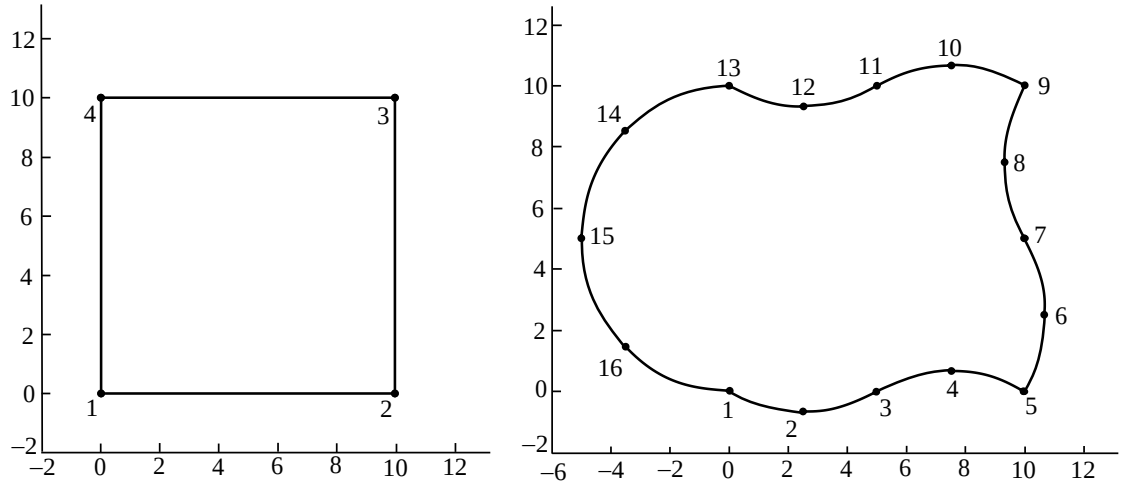


Figure 4: Regular square and deformed quadrilateral domains used for the numerical assessments of Sections 7 and 8.

The graph on the right of Fig. 5 shows the Euclidean error norm $\varepsilon = |\mathbf{Hd} - \mathbf{Gt}|/|\mathbf{Gt}|$ for evaluations carried out combining different $n_c = 2, 4, 8$ (Fig. 3) and different numbers $n = 5, 7, 9, 11, 13$ of expansion terms, also comparing with results of the CBEM. The accuracy tends to increase with n_c mainly because the number of adjacent elements to a given element – for which no series expansion is carried out – also increases with n_c. Although the accuracy always increases with n, there is also always a threshold for the mesh refinement.

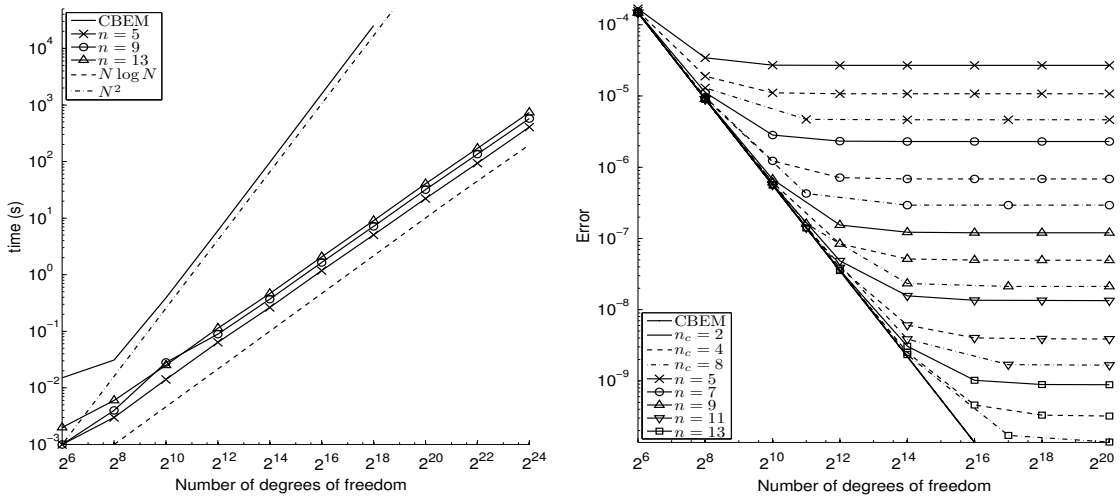


Figure 5: Execution times (left) for the evaluation of Eq. (11) for a linear potential problem over a square, and accuracy results (right) for different numbers of children poles and expansion terms.

8 Some results for curved elements

The deformed quadrilateral on the right of Fig. 4 is built up with eight quadratic macroelements, which then undergo successive splitting (always approximating the original geometry) in terms of either constant, linear of quadratic elements with up to $2^{20} = 1,048,576$ degrees of freedom for a potential problem analysis. This irregularly-shaped domain is submitted to a potential field $x^2 - y^2$, with corresponding nodal potentials $\mathbf{d}$ and normal gradients $\mathbf{t}$ evaluated for the numerical analysis using Eq. (11) in terms of FMM expansions. Owing to the polynomial characteristics of the applied potential together with the large domain size and the irregular boundary, the proposed problem actually poses a good challenge for the numerical simulation.

Figure 6 shows the computational times required for the different element types and different cases of either 5, 9 or 13 expansion terms. The graphs are for numbers of children n_c, as given in Fig. 3, equal to either two (left) or four (right). The running times are all proportional to $N \log N$. The expansions with $n_c = 4$ required in general 50% more time than for $n_c = 2$. The most time expensive case of all did not require more than 15 minutes to run. For comparison, running times for the evaluations in terms of the conventional BEM (solid circles) are also shown for up to $2^{15} = 32,768$ degrees of freedom, which are all proportional to N^2.

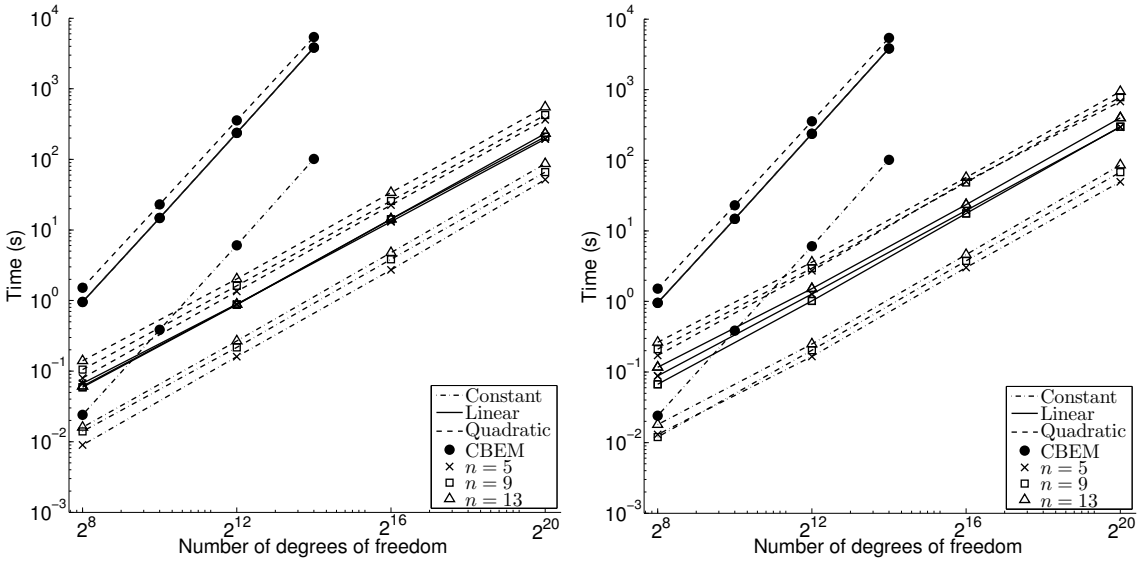


Figure 6: Execution times for the deformed quadrilateral of Fig. 4 with $n_c = 2$ (left) and $n_c = 4$ (right) children pole expansions.

A convergence analysis using the same Euclidean norm of the previous example is shown in Fig. 7 for all cases discussed with regard to Fig. 6 and also using the results from the conventional BEM as target values. The convergence pattern is the same of the previous example, although not as regular. Although both discretization and expansion errors have shown to be small from the very beginning, there is also an accuracy threshold indicating that it is not worth refining the mesh any further. The results with $n = 5$ expansion terms do not improve with mesh refinement, and the results deteriorate – independently from whether $n_c = 2$ or $n_c = 4$ – from constant to linear to quadratic elements most probably due to the boundary-layer effect. The results with $n = 9$ expansion terms show a significant improvement. However, at least for

the present numerical example, the results using quadratic elements and $n_c = 2$ have turned out to be very accurate at a relatively low cost for $n = 13$ expansion terms.

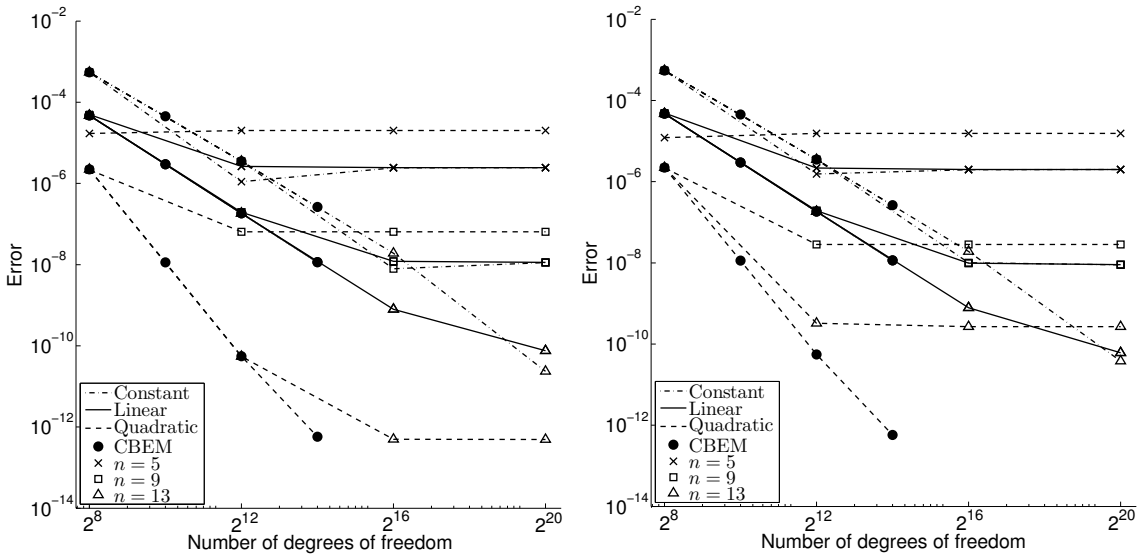


Figure 7: Accuracy results for the deformed quadrilateral of Fig. 4 with $n_c = 2$ (left) and $n_c = 4$ (right) children pole expansions.

9 Concluding remarks

This paper significantly advances the developments by Dumont & Peixoto (2014) and Peixoto *et al.* (2015) and synthesizes some of the achievements obtained by Peixoto (2014) and Novelino (2015). Owing to space restrictions, only the basic features of the proposed developments could be presented. They are the compact expression of the expansion of a general fundamental solution about successive levels of source and field poles, as given in Eq. (8), and the pre-integrations represented in Eq. (14) and in part illustrated in Eq. (16) for quadratic elements considered in the frame of a consistent BEM formulation (Dumont, 2010). As implemented by Novelino (2015), an alternative, simpler procedure seems to be the symbolic evaluation of the polynomial terms of Eq. (14) using $\mathrm{Maple}^{\mathrm{TM}}$ for as many terms as desired, and the subsequent storage as pre-assigned polynomial functions in the implemented C++ code. The still preliminary examples for constant and linear elements as well as for high-order, curved elements attest the possibilities of analysis that are being achieved. The combination with the expedite BEM (Dumont & Aguilar, 2012) in order to further accelerate the whole computational process and the use of a mesh with hierarchical squares (Dumont & Aguilar, 2011) to evaluate pole distances in terms of Boolean algebra are some of the implementations in progress. There are some conceptual differences between the conventional and the expedite BEM, as for instance the fact that the matrix $\mathbf{H}$ is used in Eq. (11) whereas its transpose appears in the hybrid BEMs, which lead to implementation peculiarities still to be explored.

The proposed FMM must be ultimately inserted into an iterative solver. This is not the authors' concern at the moment, since the use of a GMRES solver, for instance, together with the identification of the best pre-conditioner, seems to be well established in the technical literature (Liu, 2009). Thus, the proposed systematic assessment takes apart the issues related to a

plain FMM procedure from the issues related to an iterative solver particularly with respect to execution time and may lead to a more efficient codification of the introduced algorithms.

ACKNOWLEDGEMENTS

This work was supported by the Brazilian agencies CAPES, CNPq and FAPERJ.

REFERENCES

Brebbia, C.A., Telles, J.C.F. & Wrobel, L.C., 1984. *Boundary Element Techniques*. Springer-Verlag: Berlin and New York.

Dongarra, J. & Sullivan, F, 2000. The top ten algorithms of the century. *Computing in Science and Engineering*, vol.2, n. 1, pp. 22-23.

Dumont, N.A., 2010. The boundary element method revisited. In Brebbia, C.A., ed, *Boundary Elements and Other Mesh Reduction Methods XXXII*, pp. 227-238, WITPress, Southampton.

Dumont, N.A. & Aguilar, C.A., 2011. Three-dimensional implementation of the expedite boundary element method. In *Procs. IABEM2011 - Symposium of the International Association for Boundary Element Methods*, pp. 113-118, Brescia, Italy.

Dumont, N.A. & Aguilar, C.A., 2012. The best of two worlds: the expedite boundary element method. *Engineering Structures*, vol. 43, pp. 235-244.

Dumont, N.A. & Mamani, E.Y., 2011. Generalized Westergaard Stress Functions as Fundamental Solutions. *CMES – Computer Modeling in Engineering & Sciences*, vol. 78, n. 2, pp. 109-150.

Dumont, N.A. & Peixoto, H.F.C., 2014. Application of the hybrid boundary element method to large-scale problems using a fast multipole technique. In Bustamante, R., ed, *Procs. PACAM XIV – 14th Pan-American Congress of Applied Mechanics*, pp. 6 on CD, Santiago, Chile.

Greengard, L. & Rokhlin, V., 1987. A fast algorithm for particle simulations. *Journal of Computational Physics*, vol. 135, pp. 280-292.

Liu, Y., 2009. *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, New York, USA.

Liu, Y.J., Mukherjee, S., Nishimura, N., Schanz, M., Ye, W., Sutradhar, A., Pan, E., Dumont, N.A., Frangi, A. & Saez, A., 2011. Recent advances and emerging applications of the boundary element method. *Applied Mechanics Reviews*, vol. 64, n. 3, 38 pp.

Liu, Y.J. & Nishimura, N., 2006. The fast multipole boundary element method for potential problems: a tutorial. *Engineering Analysis with Boundary Elements*, vol. 30, pp. 371-381.

Nishimura, N., 2002. Fast multipole accelerated boundary integral equation methods. *Applied Mechanics Reviews*, vol. 55, pp. 299-324.

Novelino, L.S., 2015. *A Novel Fast Multipole Technique in the Boundary Element Methods*. M.Sc. thesis, PUC-Rio, Brazil.

Peixoto, H.F.C., 2014. *Application of the Hybrid Boundary Element Method to Large-Scale Problems Using Fast Multipole Techniques*. M.Sc. thesis, PUC-Rio, Brazil.

Peixoto, H.F.C., Novelino, L.S. & Dumont, N.A., 2015. *Basics of a fast-multipole unified technique for the analysis of several classes of continuum mechanics problems with the boundary element*. In Brebbia, C.A., ed, *Boundary Elements and Other Mesh Reduction Methods XXXVII* (accepted), Southampton, 21-23 Sept.