7
Conclusões e sugestões

7.1.
Conclusões

No capítulo 5 foi apresentada uma nova estratégia para implementação do FMM. Esta se diferencia da apresentada na literatura por empregar o FMM para uma solução genérica, tal como proposto por Dumont e Peixoto (2014) e apresentado no Capítulo 4. Desta forma, embora o algoritmo tenha sido desenvolvido para um problema de potencial, este pode ser adaptado para problemas com características vetoriais, como um de elasticidade. Outra contribuição é a substituição das funções de forma usuais pelas apresentadas na equação (2.19), conforme proposto por Dumont (2010), o que faz com que as integrações se tornem inteiramente polinomiais, até mesmo para elementos de alta ordem, o que permite que estas sejam integradas analiticamente.

O algoritmo desenvolvido também se diferencia dos apresentados na literatura (Bapat e Liu, 2010) por utilizar distâncias topológicas, ao invés de distâncias geométricas. Isto é feito através do algoritmo desenvolvido por Dumont (2012) que toma proveito da discretização da malha para gerar a estrutura hierárquica de adjacências. Conforme apresentado no Capítulo 5, a distância entre os elementos é avaliada pela adjacência entre estes em um dado nível.

A partir dos resultados apresentados no Capítulo 6, observou-se que o número de filhos associados a um polo influencia positivamente na redução do erro, pelo fato de que quanto maior o número de elementos associados a um polo maior será a quantidade de elementos adjacentes, logo haverá uma maior quantidade de elementos avaliados por integração direta.

Com relação ao número de termos da expansão, verificou-se que o acréscimo destes influencia significativamente na redução do erro, sem que haja uma redução significativa na eficiência do algoritmo.

A implementação proposta se mostrou vantajosa em termos de eficiência computacional quando comparada ao CBEM, conforme foi visto nos resultados apresentados no Capítulo 6.

7.2. Sugestões para trabalhos futuros

Os seguintes tópicos podem ser abordados, em complementação a presente dissertação:

- Implementação da técnica fast multipole ao Método Expedito dos elementos de contorno (EBEM) no intuito de acelerar o processo computacional.
- Desenvolvimento da estratégia apresentada do GFMBEM para o caso 3D.
- Desenvolver uma técnica unificada para a determinação da distância entre elementos que leve em consideração o critério topológico desenvolvido no presente trabalho com uma estrutura hierárquica que permita a avaliação de geometrias bastante irregulares.

8
Referências Bibliográficas

Bapat, M.S. e Liu, Y.J. 2010. A new adaptive algorithm for the fast multipole boundary element method. *CMES.* 2010, Vol. 58, pp. 161-183.

Beatson, R. e Greengard, L. 1997. A short course on fast multiple methods. *Wavelets, multilevel methods and elliptic PDEs 1.* 1997, pp. 1-37.

Brebbia, C. A. 1978. *The boundary element method for engineers.* Londres : Pentech Press, 1978.

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

Dumont. 2012. Unified algorithm for the generation of refined 2D surface meshes. *Comunicação interna.* 2012.

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

Dumont, N. A. e Peixoto, H. F. C. 2014. Application of the Hybrid Boundary Element Method to Large-scale Problems Using a Fast Multipole Technique. *14th Pan-American Congress of Applied Mechanics.* 2014.

Dumont, N. A. 1987. The hybrid boundary element method. *Southampton : Computational Mechanics Publications, Springer-Verlag.* 1987, pp. 125-138.

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

Dumont, N.A. 2010. The boundary element method revisited. *Boundary Elements and Other Mesh Reduction Methods XXXII, ed C. A. Brebbia. Southampton:WITPress.* 2010, pp. 277-238.

Greengard, L. e Rokhlin, V. 1987. A fast algorithm for particle simulations. *Journal of Computational Physics.* 1987, Vol. 73, pp. 325-348.

Greengard, L. 1994. Fast Algorithms for Classical Physics. *Science.* Agosto de 1994, Vol. 265.

Kurz, S., Rain, O. e Rjasanow, S. 2009. Application of the adaptive cross approximation technique for the coupled BE-FE solution of symmetric electromagnetic problems. *Computational mechanics 32.4-6,* 2009, pp. 423-429.

Liu, Y. 2009. *Fast multipole boundary element method, theory and applications in engineering.* New York : Cambridge University Press, 2009.

Liu, Y.J. e Nishimura, N. 2006. The fast multipole boundary element method for potential problems: A tutorial. *Engineering Analysis with Boundary Elements.* 2006, Vol. 30, pp. 371–381.

Liu, Y.J., et al. 2011. Recent Advances and Emerging Applications of the Boundary Element Method. *Applied Mechanics Reviews.* 2011, Vol. 64, Issue 3, 030802 (38 pp).

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

Oliveira, M.F.F.D. 2004. Métodos de Elementos de Contorno Convencional, Híbridos e Simplificados. *Dissertação de Mestrado - Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro.* 2004.

Peixoto, H.F.C. 2014. Um Estudo do Método Fast Multipole Aplicado a Problemas de Elementos de Contorno. *Dissertação de Mestrado - Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro.* 2014.

Peixoto, H.F.C., Novelino, L.S. e 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. 2015.

Trevelyan, J. 1994. *Boundary element for engineers: theory and applications.* Southampton : Computational Mechanics Publications, 1994.

9
Apêndice 1

Neste apêndice será apresentado o algoritmo unificado para o refinamento hierárquico de um contorno bidimensional discretizado em elementos lineares, quadráticos e cúbicos (Dumont, 2012). Um algoritmo correspondente a este para problemas tridimensionais pode ser obtido em Dumont e Aguilar (2011).

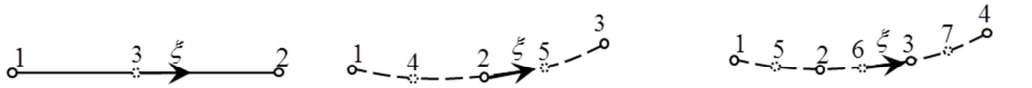
9.1.
A Unified algorithm for hierarchical mesh refinement

The following unified algorithm refines a given mesh of either linear, quadratic or cubic elements, which are characterized as of type $te = T\_t_e[o_e]$, where

$$T\_t_e = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} \tag{8.1}$$

in terms of the element number oe. This is the most basic information to be input. Then, te is the number of nodes of the element to be split in the mesh refinement. In Equation (8.1), the entries correspond to linear, quadratic or cubic elements, although it can be easily generalized to higher-order elements. As implemented, only one element type can appear in a given mesh.

Figure 28 shows the schemes of the three different elements considered in the present algorithm, as taken out of a general mesh corresponding to a given level of refinement.



a) Linear element (*te*=2, *oe*=1) b) Quadratic element (*te*=3, *oe*=2) c) Cubic element (*te*=4, *oe*=3)

Figure 28 - Scheme of three different elements that are split each into two sub-elements.

In the subdivision procedure the elements are split each into two sub-elements. The nodes of each parent element are locally numbered $1\ldots te$. There are

oe new generated nodes numbered sequentially according to the array $T_{new} = T\_T_{new}[oe]$, where

$$T\_T_{new} = \left[\left[3\right] \quad \left[4 \quad 5\right] \quad \left[5 \quad 6 \quad 7\right]\right] \tag{8.2}$$

The coordinates of the new nodes are given by $C_{new} = T\_C_{new}[oe]$, where

$$T\_C_{new} = \left[\left[0\right] \quad \left[-1/2 \quad 1/2\right] \quad \left[-2/3 \quad 0 \quad 2/3\right]\right] \tag{8.3}$$

in natural curvilinear coordinate ξ, as represented in Figure 28, which spans the interval $[-1,1]$.

The interpolation functions of the reference elements are given as $N = T\_N[oe]$, where

$$T\_N\left[1,2,3\right] = \left[\left[\frac{1-\xi}{2},\frac{1+\xi}{2}\right],\left[\frac{\xi(\xi-1)}{2},1-\xi^2,\frac{\xi(\xi+1)}{2}\right],\left[\frac{(1-\xi)(9\xi^2-1)}{16},\frac{9(3\xi-1)(\xi^2-1)}{16},\frac{9(3\xi+1)(1-\xi^2)}{16},\frac{(1+\xi)(9\xi^2-1)}{16}\right]\right] \tag{8.4}$$

As outlined in the following, the procedure consists in subdividing each element (the parent element) of a basic mesh-refinement level, thus creating new nodes. An amount of $N_{new} = T\_N_{new}[oe]$ nodes,

$$T\_N_{new} = \left[1 \quad 1 \quad 2\right] \tag{8.5}$$

is generated in the splitting procedure. These nodes are numbered as $Numb_{parent\_new} = T\_Numb_{parent\_new}[oe]$, where

$$T\_Numb_{parent\_new}\left[1 \quad 2 \quad 3\right] = \left[\left[1 \quad 1\right] \quad \left[1 \quad 2\right] \quad \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}\right] \tag{8.6}$$

in the parent element, with each column referring to one of the generated elements, as obtained from Figure 1. In the generated elements, these nodes are referred to in terms of $Numb_{child\_new} = T\_Numb_{child\_new}[oe]$, where

$$T\_Numb_{child\_new}\left[1 \quad 2 \quad 3\right] = \left[\left[2 \quad 1\right] \quad \left[2 \quad 2\right] \quad \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}\right] \tag{8.7}$$

On the other hand, $N_{old} = T\_N_{old}[oe]$ (pre-existing) nodes of the parent element, where

$$T\_N_{old} = \left[1 \quad 2 \quad 2\right] \tag{8.8}$$

are inherited by the new elements. These nodes are numbered as $Numb_{parent\_old} = T\_Numb_{parent\_old}[oe]$, where

$$T\_Numb_{parent\_old}\begin{bmatrix}1 & 2 & 3\end{bmatrix} = \left[\begin{bmatrix}1 & 2\end{bmatrix} \quad \begin{bmatrix}1 & 2\\ 2 & 3\end{bmatrix} \quad \begin{bmatrix}1 & 3\\ 2 & 4\end{bmatrix}\right] \tag{8.9}$$

in the parent element, with each column referring to one of the generated elements, as obtained from Figure 1. In the generated elements, these nodes are referred to in terms of $Numb_{child\_old} = T\_Numb_{child\_old}[oe]$, where

$$T\_Numb_{child\_old}\begin{bmatrix}1 & 2 & 3\end{bmatrix} = \left[\begin{bmatrix}1 & 2\end{bmatrix} \quad \begin{bmatrix}1 & 1\\ 3 & 3\end{bmatrix} \quad \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right] \tag{8.10}$$

9.1.1. Input data

- *oe*: either 1, 2 or 3, which defines the element type (te=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 (for example, $nv=1$ indicates that the structure will be refined once).
- Tables $Xgl[\ ]$ and $Ygl[\ ]$ with *nne* node coordinate entries $[x, y]$, 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 te global node numbers of the elements. Arrays $inc[k][\ ]$ for $k=2...nv+1$, will be generated as a result of the mesh refinement.

9.1.2. Output data

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

- $nek[k=1...nv+1]$ Number of elements at each one of the $nv+1$ levels.
- $nglk[k=1...nv+1]$ Number of nodes at each one of the $nv+1$ levels.

- Tables $Xgl[\ ]$ and $Ygl[\ ]$ with $nglk[nv+1]$ node coordinate entries, which correspond to the input values plus the coordinates of the generated nodes.
- A table $inc[k=1...nv+1][\ ]$ with $nv+1$ levels of arrays of local-to-global node incidences. The second entry are with $nek[k]$, $k=1...nv+1$, arrays, each one with te global node numbers of the elements.

The algorithm generates $nv+1$ levels of mesh refinement, 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. And the number of nodes is not known in advance.

9.1.3. Initial definitions

- Initial number of elements (on level $nv=0$): $nek[1]=nee$
- Initial number of nodes (on level $nv=0$): $nglk[1]=nne$
- Element type te, according to Eq. (8.1)
- Array T_{new} with the local numbering of the new nodes, according to Eq.(8.2)
- Array C_{new} of natural coordinates ξ of the new nodes, according to Eq.(8.3)
- Shape functions $N(\xi)$, according to Eq.(8.4)
- N_{old} according to Eq. (8.8)
- N_{new} according to Eq. (8.5)

9.1.4. Execution of the algorithm

1. Loop for the refinement levels, counting $k\ from\ 1\ to\ nv$

- Define the number of elements of the next level: $nek[k+1]=2\times nek[k]$
- Initialize the counter of the number of nodes of the next level: $nglk[k+1]=nglk[k]$ (start counting the total number of nodes for level $k+1$)

1.1 Loop for the elements to be split into 2 elements, counting $i\ from\ 1\ to\ nek[k]$

Evaluate the coordinates of the generated new nodes $T_{new}[1]\ldots T_{new}[N_{new}]$, which are all internal.

1.1.1 Loop for the new nodes, counting i_{new} $from\ 1\ to\ o_e$

$nglk[k+1] = nglk[k+1]+1$ (add 1 to the total number of nodes)

$gn_{new}[i_{new}] = nglk[k+1]$ (temporary array with the global numbering, to be used in loop 1.1.2.2)

$$Xgl[nglk[k+1]] = \sum_{j}^{t_e} N[j](C_{new}[i_{new}]).Xgl[Inc[k][i,j]]$$

$$Ygl[nglk[k+1]] = \sum_{j}^{t_e} N[j](C_{new}[i_{new}]).Ygl[Inc[k][i,j]]$$

End of loop 1.1.1 with variable i_{new}**.**

1.1.2 Loop to assign the incidences of the generated elements, counting ie $from\ 1\ to\ 2$

Generate the nodal incidence table for the already existing nodes:

1.1.2.1 Loop for the old nodes, counting i_{old} $from\ 1\ to\ N_{old}$

$$Inc[k+1][2i-2+ie, Numb_{child\_old}[i_{old}, ie]] = Inc[k][i, Numb_{parent\_old}[i_{old}, ie]]$$

End of loop 1.1.2.1 with variable i_{old} .

Generate the nodal incidence table for the new nodes:

1.1.2.2 Loop for the old nodes, counting i_{new} $from\ 1\ to\ oe$

$$Inc[k+1][2i-2+ie, Numb_{child\_new}[i_{new}, ie]] = gn_{new}[Numb_{parent\_new}[i_{old}, ie]]$$

End of loop 1.1.2.2 with variable i_{new}.

End of loop 1.1.2 with variable ie for the generated elements.

End of loop 1.1 with variable i for the elements split into two new ones.

End of loop 1 with variable k for the mesh refinement level.

10
Apêndice 2

Neste apêndice apresenta-se o algoritmo unificado para as expansões do GFMBEM, estes são executados após o refinamento da malha segundo o algoritmo, desenvolvido por Dumont (2012), apresentado no apêndice anterior. Esta implementação foi realizada a partir do algoritmo desenvolvido por Dumont (2012) em linguagem Maple® o qual é responsável pela criação da estrutura de adjacências descrita no Capítulo 5, e aplicada à técnica de 'fast multipole' pela autora. O algoritmo foi desenvolvido em linguagem C^{++} e está disponível para consulta.

10.1.A unified algorithm for pole expansions

The following procedures are executed inside the macroelements loop, counting *ie from* 1 *to nek*$[k=1]$ and all the others procedures are called inside it. The combinations of the following main procedures with the refinement procedure presents in the former appendix, output the vectors **Gt** and **Hd**.

10.2.
Unified algorithm for the generation of refined boundary meshes – use of a hierarchical concept

The following algorithm describes the construction of the element adjacency structure for general 2D or 3D problems. This concept shall replace the concept of node adjacencies, as applied up to here.

10.2.1.
Input data

The input are the results from the mesh generation algorithms for a 2D and the number of partitions nv, which is the number of element subdivisions, which is always equal to 2 or 4, for 2D or 3D problems.

10.2.2. Output data

- $el\_adj[k][ie][ia]$ which is the element adjacency
- $n\_adj[k][ie]$ number of adjacent elements

where k is the refinement level (equal to 1, in the initializing algorithm), ie is the reference element and ia is one of the adjacent elements. By definition in the following algorithm, an element is adjacent to itself.

10.2.3. Algorithm for the first level $(k = 1)$

Initialize the counter of adjacent elements: $count = 0$

Store the global numbering of the reference element: $elsplit[1] = ie$

1 Loop for the possible adjacent elements, counting ia *from* 1 *to* $nel[1]$

Set the initial condition for breaking the search for an adjacent element: $breakCond = false$

1.1 Loop for the nodes of the reference element, counting i_n *from* 1 *to te while* $breakCond = false$

1.1.1 Loop for the nodes of the adjacency candidate element, counting i_{na} *from* 1 *to te while* $breakCond = false$

if $inc[1][ie, i_n] = inc[1][ia, i_{na}]$ *then* (adjacency found)

$count = count + 1$

$elAdj[1][count] = ia$

$breakCond = true$ (exit loops i_{na} and i_n)

end if

End of loop 1.1.1 (i_{na})

End of loop 1.1 (i_n)

(This is the return point from the above if loop in the case of adjacency found)

End of loop 1 (ia)

$n_{adj}[1][ie] = count$ (Total number of adjacent elements to element ie)

The algorithm just described produces a global adjacency structure. The following algorithm describes the construction of the element adjacency structure for the subsequent levels ($k > 1$). However, they refer to the np partitioned elements derived from a parent element, starting from the first, coarsest mesh, and in the frame of successive mesh refinement.

10.2.4. Algorithm for the next levels $(k > 1)$

This routine evaluates the element adjacencies on a given refinement level once the adjacencies of the parent element are known.

10.2.4.1. Input data

- latest evaluated level k
- parente element globally numbered ie
- corresponding adjacencies $elAdj[k][\ ]$
- number $n_{adj}[k][\ ]$ of adjacent elements of the parent element ie (on level k)

10.2.4.2. Output data

The procedure consists in splitting the given parent element ie of level k into np elements, also generating the necessary adjacency information. For $iepr = 1..np$, evaluate

- Global numbering of the split elements $iep = (ie-1)np + iepr$, stored in $elsplit[k+1]$
- Number of adjacent elements $n_{adj}[k+1]$
- Adjacency structure $elAdj[k+1][\ ]$, which is a set with $n_{adj}[k+1]$ elements.

10.2.4.3. Execution of the algorithm

1. **Loop for the np split elements, counting** *iepr from* 1 *to np*

Evaluate the global numbering of the split element: $iep = (ie-1)np + iepr$

Store the global numbering of the split element: $elsplit[k+1] = iep$

Initialize the counter of adjacent elements: $count = 0$

1.1. Loop for the adjacent elements on level k, counting ia *from* 1 *to* $n_{adj}[k]$

1.1.1. Loop for the split elements (level $k+1$) originated from the adjacent elements on level k, counting $iapr$ *from* 1 *to np*

Evaluate the global numbering of the adjacent split element $iap = (elAdj[k][ia]-1)np + iapr$

Set the initial condition for breaking the search for an adjacent element: $breakCond = false$

1.1.1.1. Loop for the nodes of the reference split element, counting i_n *from* 1 *to te while* $breakCond = false$

1.1.1.1.1 Loop for the nodes of the adjacency candidate element, counting i_{na} *from* 1 *to te while* $breakCond = false$

if $inc[k+1][iep, i_n] = inc[k+1][iap, i_{na}]$ *then* (adjacency found)

$count = count + 1$

$elAdj[k+1][count] = iap$

$breakCond = true$ (exit loops i_{na} and i_n)

end if

End of loop 1.1.1.1.1. (i_{na})

End of loop 1.1.1.1. (i_n)

(This is the return point from the above if loop in the case of adjacency found)

End of loop 1.1.1. ($iapr$)

End of loop 1.1. (ia)

$n_{adj}[k+1] = count$ (Total number of adjacent elements to element $iepr$)

Test if the most refined mesh has been attained

if $(k+1)=(nv+1)$ *then* the limit level has been attained

- Evaluate the contribution of the adjacent elements through CBEM (Procedure 1)
- Evaluation of the first expansion and the vectors $\widetilde{\mathbf{G}}$ and $\widetilde{\mathbf{H}}$ for the element's nodes through Procedure 2.
- Test if all brothers of the current microelement were expanded to the father pole, if some expansions are still to be made the test is closed, returning to the current, where a structure of adjacencies of the brother element will be constructed and subsequently go through all given former procedures. In case all child elements have already been expanded then Procedure 3 is executed, which is responsible for managing the expansions related to GFMM, having as a result the relative contributions of vectors **Gt** and **Hd**.

else

The current procedure is called recursively, giving as parameters the level $k=k+1$ and the element $ie=iep$.

end if

End of loop 1 $(iepr)$

10.2.5. Procedures referred to in the algorithm

10.2.5.1. input for the procedures referred to in the algorithm

- n : number of series terms.
- k_{BEM} : level from which elements are considered adjacent and are evaluated through the CBEM.
- k_{child} : difference of levels between parent and child elements.
- k_{exp} : determine the last level of field expansions.
- Initial definitions:

  Define vector fac through Preliminary Procedure 1

  Define matrix C through Preliminary Procedure 2

  Define vector $Q(Z)$ through Preliminary Procedure 3

10.2.5.2. Procedures

Procedure 1: This procedure evaluates the adjacent elements contribution through the CBEM, using numerical integration and the proposed substitution of $t_l \leftarrow t_l |J|_{(\text{at } l)} / |J|$.

1. **Loop for the adjacent elements to the field element $(iepr)$, counting** ia *from* 1 *to* $nAdj[k_{BEM}]$

   Adjacent element at level k_{BEM} : $ia\_parent = elAdj[k_{BEM}][ia]$

   1.1. Loop for the child adjacent element, counting n_{child} *from* 1 *to* $np^{(nv+1-k_{BEM})}$

   Adjacent element at level k_{BEM} :

   $$ia\_child = (ia\_parent - 1)\, np^{(nv+1-k_{BEM})} + n_{child}$$

   Execute the CBEM algorithm.

   End of loop 1.1. (n_{child})

End of loop 1. (ia)

Procedure 2: In this step the first expansion of the field element is calculated, this being in reference to the expansion from the node of the child element to the node of the parent element at the level $nv + 1 - k_{child}$. At this moment the parts $\widetilde{\mathrm{G}}$ and $\widetilde{\mathrm{H}}$ of the contributions to the vectors $\vec{\mathrm{G}}$ and $\vec{\mathrm{H}}$ will be calculated. These parts will be temporarily saved in the line $nv+1$ of the matrixes $Pmatrixg$ and $Pmatrixh$, respectively.

Parent element: $parent = elsplit[nv+1-k_{child}]$

Evaluate the coordinates of the parent's pole: $z_{c0} = 0.5(x_{pole} + y_{pole} I)$ where:

$$x_{pole} = Xgl[inc[nv+1-k_{child}][parent,1]] + Xgl[inc[nv+1-k_{child}][parent,oe+1]]$$
$$y_{pole} = Ygl[inc[nv+1-k_{child}][parent,1]] + Ygl[inc[nv+1-k_{child}][parent,oe+1]]$$

Generate the vector Δ with the distances between the each node of the field element and the pole z_{c0}:

for jl from 1 *to* $oe+1$ *do*

$\Delta[jl] = z[jl] - z_{c0}$

end do

where $z[jl] = Xgl\left[inc[nv+1][iepr, jl]\right] + Ygl\left[inc[nv+1][iepr, jl]\right] I$

1. **Loop for the nodes of the field element** $(iepr)$**, counting**

   jl from 1 *to* $oe+1$

   Local numbering of the field node: $jk = (iepr-1)oe + jl$

   for i to $(n+1)$ *do*

   $$PMatrixg[nv+1, i] = PMatrixg[nv+1, i] + IntegTableG \cdot Jac0 \cdot tvector(jk + iepr)$$

   $$PMatrixh[nv+1, i] = PMatrixh[nv+1, i] + IntegTableH \cdot dvector\left(\left[inc[nv+1][iepr, jl]\right]\right)$$

   where $Jac0$ in the nodal value of the jacobian that will be evaluated through Procedure 4 and $IntegTableG$ and $IntegTableH$ are the results of the integration (4.27) and (4.28), respectively, provided by the integration tables.

   end do $(loop\ i)$

End of loop 1. (jl)

Procedure 3: This procedure oversees the expansions between field points, besides being responsible for expansions between field and source poles.

The first step is to evaluate witch source element is adjacent to the field element on the current expansion level (k), this evaluation is done through the list of adjacent elements on the level ($k - k_{child}$).

Define the child element that will have its pole expanded: $child = elsplit[k]$

1. Loop of the adjacent elements on level $k - k_{child}$**, counting** ia *from* 1 *to* $n_{adj}[k - k_{child}]$

Adjacent parente element $ia_{parent} = elAdj[k - k_{child}][ia]$

1.1. Loop of child elements of the adjacent element on the level $k - k_{child}$**, counting** n_{child} *from* 1 *to* η_c *do*

child of the adjacent element: $ia_{child} = (ia_{parent} - 1)\eta_c + n_{child}$

Start of the adjacency test between the child element (ia_{child}) and the field element.

Set the initial condition for breaking the search for an adjacent element: $breakCond = false$

1.1.1. **Loop of the adjacent elements on level** k**, counting** n_{verif} *from* 1 *to* $n_{adj}[k]$ *while* $breakCond = false$

Test if immediate adjacency occurs between the elements

if $ia_{child} = el_{adj}[k][n_{verif}]$ *then* stop the search defining: $breakcond = true$

end if

End loop 1.1.1. (n_{verif}**)**

In case elements are adjacent on level $k - k_{child}$, but not on level k

if $breakCond = false$ *then*

$source = ia_{child}$

Evaluate vector $Q(Z)$ regarding elements on level k and the contributions to vectors **Gt** and **Hd**. (Procedure 3.2)

end if

End of loop 1.1. (n_{child})

End of loop 1. (ia)

The second step involves the evaluation of the expansions between the current field pole and the pole of the parent element on level $k - k_{child}$, if there are expansions to be made.

Test if there still are poles to be expanded,

if $k - k_{child} \geq k_{\exp}$ *then* Procedure 3.1 is called, where the parent element will be: $parent = elsplit\left[k - k_{child}\right]$

Next, a test is done to see if the brother of the child element has already been expanded to the parent element.

if $\left(\frac{2\ child}{\eta_c}\right) = even\ number\ then$ all the child elements have already been expanded to the father element

The current procedure is called again, giving $k = k - k_{child}$, which is the level of the current pole.

enf if

else if $\left(k - k_{child}\right) < k_{exp}$ *then* it means that all the expansions between field poles were taken care.

Since this corresponds to the last expansion pole, all non-adjacent elements on level $k - k_{child}$ are looked for, meaning all source elements for which the contributions where not evaluated.

2. Loop for the source elements on level $k - k_{child}$ **candidates for non-adjacency, counting** $el\_source\ from\ 1\ to\ nek\left[k - k_{child}\right]$

Set the initial condition for breaking the search for a non-adjacent element: $breakCond = false$

2.1. **Loop for the adjacent elements at level** $k - k_{child}$ **, counting** $N_{verif}\ from\ 1\ to\ n_{adj}\left[k - k_{child}\right]\ while\ breakCond = false$

if $el\_source = elAdj\left[k - k_{child}\right]\left[N_{verif}\right]$ *then* el_source is an adjacent element

$breakCond = true$

end if

End of loop 2.1. $\left(N_{verif}\right)$

if $breakCond = false$ *then* the elements are not adjacent

2.2. Loop for the source child elements, counting $n_{child}\ from\ 1\ to\ Nchild\ do$

Source element non-adjacent at level k:
$source = (el\_source - 1)\, Nchild + n_{child}$

Evaluate vector $Q(Z)$ relative to elements on level k and then the contributions to the vectors **Gt** and **Hd**. (Procedure 3.2)

End of loop 2.2. (n_{child})

end if

End of loop 2. (el_source)

Reinitialize Matrixes $PMatrixg$ and $PMatrixh$.

end if

Procedure 3.1: This procedure is responsible for the expansion between field poles.

Calculation of the vector $P(Z)$ of dimension $(n+1)$, relative to the expansion of poles of consecutives levels. Z being the distance between the child element pole of level k and the parent of level $k - k_{child}$.

Evaluate the coordinates of the parent element: $z_{parent} = 0.5\left(x_{parent} + y_{parent} I\right)$

where:

$$x_{parent} = Xgl\left[inc[k - k_{child}][parent, 1]\right] + Xgl\left[inc[k - k_{child}][parent, oe + 1]\right]$$
$$y_{parent} = Ygl\left[inc[k - k_{child}][parent, 1]\right] + Ygl\left[inc[k - k_{child}][parent, oe + 1]\right]$$

Evaluate the coordinates of the child element: $z_{child} = 0.5\left(x_{child} + y_{child} I\right)$

where:

$$x_{child} = Xgl\left[inc[k][child, 1]\right] + Xgl\left[inc[k][child, oe + 1]\right]$$
$$y_{child} = Ygl\left[inc[k][child, 1]\right] + Ygl\left[inc[k][child, oe + 1]\right]$$

Evaluate vector $P(Z)$, where $Z = z_{child} - z_{parent}$

$Pvector[1]=1$

$Pvector[2]=Z$

for i from 3 *to* $(n+1)$ *do*

$Pvector[i]=Pvector(i-1)Z$

end do

for i to $(n+1)$ *do*

$$PMatrixg[k,i]=PMatrixg[k,i]+\sum_{j=1}^{i}C[j,i+1-j]\,PMatrixg[k+k_{child},j]Pvector[i+1-j]$$

$$PMatrixh[k,i]=PMatrixh[k,i]+\sum_{j=1}^{i}C[j,i+1-j]\,PMatrixh[k+k_{child},j]Pvector[i+1-j]$$

end do $(loop\ i)$

Reinitialize the line $k+kchild$ of matrixes $PMatrixg$ and $PMatrixh$.

Procedure 3.2: This procedure can happen in two different ways, depending if the source expansions will or not be considered.

a) Procedure 3.2 without source expansion: This procedure is responsible for the evaluation of $Q(Z)$ vector, as it was defined in Equation (4.5), and for the evaluation of the contributions for the vectors **Gt** and **Hd**.

In this procedure, variable Z is substituted in vector Q(Z) which was pre-calculated, being Z the difference of coordinates between the field pole (z_{cnk}) and the source element's node (z_0) at level $nv+1$

Evaluation of the field pole coordinates z_{cnk} :

$$\left.\begin{aligned} x_f &= Xgl\left[inc[k][child,1]\right]+Xgl\left[inc[k][child,oe+1]\right]\\ y_f &= Ygl\left[inc[k][child,1]\right]+Ygl\left[inc[k][child,oe+1]\right]\end{aligned}\right\}\rightarrow z_{cnk}=0.5\left(x_f+y_f I\right)$$

1. Loop for the source microelements, counting n_{child} *from* 1 *to* $np^{(nv+1-k)}$

Source element at level $nv+1$

$source\_child=(source-1)np^{(nv+1-k)}+n_{child}$

1.1. Loop for the source element's nodes, counting *ml from* 1 *to oe*

Evaluation of the field pole coordinates (z_c):

$$\left.\begin{aligned} x_s &= Xgl\left[inc[nv+1][source\_child, ml]\right] \\ y_s &= Ygl\left[inc[nv+1][source\_child, ml]\right] \end{aligned}\right\} \rightarrow z_0 = (x_s + y_s I)$$

for i from 1 *to* $(n+1)$ *do*

$$Q[i] = Q\left(z_{c^{nk}} - z_0\right)[i]$$

end do

Evaluation of the contributions for vectors

$$\vec{\mathrm{G}}[m] = \vec{\mathrm{G}}[m] + \mathrm{Re}\left(\sum_{i=1}^{n+1} fac[i]\, PMatrixg[k + k_{child}, i]\, Q[i]\right)$$

$$\vec{\mathrm{H}}[m] = \vec{\mathrm{H}}[m] + \mathrm{Im}\left(\sum_{j=2}^{n+2} fac[i-1]\, PMatrixh[k + k_{child}, i-1]\, Q[i]\right)$$

where $m = inc[nv+1][source\_child, ml]$

End of loop 1.1. (ml)

End of loop 1. (n_{child})

b) Procedure 3.2 with source expansion: This procedure is responsible for the evaluation of vectors $Q(Z)$, as defined in Equation (4.7), and for the evaluation of the contributions for the vectors **Gt** and **Hd.**

In this procedure, variable Z is substituted in vector Q(Z) which was pre-calculated, being Z the difference of coordinates between the field pole ($z_{c^{nk}}$) and the source pole ($z_{L^{nl}}$) at level $nv+1$

Evaluation of the field pole coordinates $z_{c^{nk}}$:

$$\left.\begin{aligned} x_f &= Xgl\left[inc[k][child,1]\right] + Xgl\left[inc[k][child, oe+1]\right] \\ y_f &= Ygl\left[inc[k][child,1]\right] + Ygl\left[inc[k][child, oe+1]\right] \end{aligned}\right\} \rightarrow z_{c^{nk}} = 0.5\left(x_f + y_f I\right)$$

Evaluation of the field pole coordinates $z_{L^{nl}}$:

$$\left.\begin{aligned} x_s &= Xgl\left[inc[k][source,1]\right] + Xgl\left[inc[k][source, oe+1]\right] \\ y_s &= Ygl\left[inc[k][source,1]\right] + Ygl\left[inc[k][source, oe+1]\right] \end{aligned}\right\} \rightarrow z_{L^{nl}} = 0.5(x_s + y_s I)$$

Evaluation of vector $Q(Z)$ as defined in Equation (4.7):

for i from 1 *to* $2(n+1)$ *do*

$Q[i] = Q\left(z_{c^{nk}} - z_{L^{nl}}\right)[i]$

end do

1. Loop for the source microelements, counting n_child *from* 1 *to* $np^{(nv+1-k)}$

Source element at level $nv+1$:

$source\_child = (source-1)np^{(nv+1-k)} + n\_child$

1.2. Loop for the source element's nodes, counting *ml from* 1 *to oe*

$$\left.\begin{aligned} x_{sc} &= Xgl\left[inc[nv+1][source\_child, ml]\right] \\ y_{sc} &= Ygl\left[inc[nv+1][source\_child, ml]\right] \end{aligned}\right\} \rightarrow z_0 = \left(x_{sc} + y_{sc} I\right)$$

Evaluate the $P(Z)$ vector related to the source expansion:

$Pvector[1] = 1$

$Pvector[2] = Z$

for i from 3 *to* $(n+1)$ *do*

$Pvector[i] = Pvector(i-1)Z$

end do

where $Z = \left(z_{L^{nl}} - z_0\right)$

Evaluate of the vector $Q(Z)$ related to the source expansion, as defined in Equation (4.7):

for i from 1 *to* $(n+2)$ *do*

$$Q_i\left(z_{c^{n_k}} - z_o\right) = \sum_{j=1}^{n+1} fac_j P_j\left(z_{L^{nl}} - z_o\right) Q_{i+j-1}\left(z_{c^{n_k}} - z_{L^{nl}}\right)$$

end do

Evaluation of the contributions for vectors **Gt** and **Hd**

$$\vec{G}[m] = \vec{G}[m] + \operatorname{Re}\left(\sum_{i=1}^{n+1} \frac{1}{fac_i} PMatrixg\left[k + k_{child}, i\right] \sum_{j=1}^{n+1} fac[j]\, PvectorSource[j]\, Q[i+j-1]\left(z_{c^{n_k}} - z_o\right)\right)$$

$$\vec{\mathrm{H}}[m] = \vec{\mathrm{H}}[m] +$$
$$\mathrm{Im}\left(\sum_{i=2}^{n+2} fac[i-1]\, PMatrixh[k+k_{child}, i-1] \sum_{j=1}^{n+1} fac[i]\, PvectorSource[i]\, Q[i+j-1]\left(z_{c^{n_k}} - z_o \right) \right)$$

where $m = inc[nv+1][source\_child, ml]$

End of loop 1.1. (ml)

End of loop 1. (n_child)

Procedure 4: Evaluation of the nodal value of the Jacobian

$$\left.\begin{aligned} dx(\xi) &= \sum_{i=1}^{oe+1} N'(\xi)[oe][i]\, Xgl\left[inc[nv+1][j,i]\right] \\ dy(\xi) &= \sum_{i=1}^{oe+1} N'(\xi)[oe][i]\, Ygl\left[inc[nv+1][j,i]\right] \end{aligned}\right\} \rightarrow J(\xi) = \sqrt{dx(\xi)^2 + dy(\xi)^2}$$

Where j refers to the field microelement under consideration, N' is the table with the shape functions derivatives, as presented in Equation (4.30), and the parametric variable ξ is replaced by the natural coordinates given by the following table, resulting $\xi = TabJac[oe][i]$.

$$TabJac[1 \quad 2 \quad 3] = \left[[0 \quad 1] \quad [0 \quad 1/2 \quad 1] \quad [0 \quad 1/3 \quad 2/3 \quad 1] \right]$$

10.2.6. Preliminary procedures for the GFMM

This group of procedures are evaluated before any other procedure, since it only need as input the number of series term n, and they will be executed only once.

Preliminary Procedure 1: Evaluate fac vector

$fac[1] = 1$
$fac[2] = 1$
for i from 3 *to n do*
  $fac[i] = fac(i-1)/i$
end do

Preliminary Procedure 2: Evaluate Matrix C

for i from 1 *to* $(n+1)$ *do*
  $C[i,1]=1$
  $C[1,i]=1$
  for j from 1 *to* $(n+1)$ *do*
    $C[i,j]=C[i-1,j]+C[i,j-1]$
  end do $(\text{loop } j)$
end do $(\text{loop } i)$

Preliminary Procedure 3: Pre-evaluation of the derivatives vector $Q(Z)$, as defined in Equation (4.5).

$Q[1]=f(Z)$
for i from 1 *to Qdim do*
  $Q[i]=\dfrac{\partial f^{(i-1)}(Z)}{\partial Z^{(i-1)}}$
end do

Where $Qdim=\begin{cases}(n+2) & \text{for procedure 3.2 without source expansion}\\ 2(n+1) & \text{for procedure 3.2 with source expansion}\end{cases}$