A FAST MULTIPOLE BOUNDARY ELEMENT CODE WRITTEN IN JULIA LANGUAGE

Emerson Bastos

Éder Lima de Albuquerque

emersonbas@gmail.com

eder@unb.br

Departamento de Engenharia Mecânica, Faculdade de Tecnologia, Universidade de Brasília.

Campus Darcy Ribeiro, Asa Norte, Brasília, DF. CEP: 29075-910

Lucas Silveira Campus

zaz1588@gmail.com

Universidade Federal do Espírito Santo

Av. Fernando Ferrari, 514. Goiabeiras, Vitória, ES. CEP: 29075-910

Abstract. *This paper presents a comparison of two fast multipole boundary element method implementations, in Julia and Fortran language, with application to a heat conduction problem. Fortran is a well known compiled language while Julia is a newborn pre-compiled language that offers many resources for the programmer. In this work, firstly, the fast multipole boundary element method using complex variables and Taylor series in the expansion of fundamental solutions is presented. Since influence matrices are not explicitly obtained, it is necessary the use of an iterative method to solve the linear system. The generalized minimum residue method (GMRES) was chosen based on previous works. The hierarchical data structure and the implemented algorithm are described. The two implementations of the fast multipole boundary element method are applied to a heat conduction problem and their performances are compared.*

Keywords: *Fast Multipole Method, Boundary Element Method, Julia Language, Fortran.*

1 INTRODUÇÃO

Julia é uma linguagem de programação dinâmica de alto nível e desempenho para a computação numérica. Ela apresenta vários atrativos como um compilador muito sofisticado, execução paralela distribuída, precisão numérica, e uma extensa biblioteca de funções matemáticas. A biblioteca base de Julia integra código fonte aberto escrito em C e bibliotecas Fortran para álgebra linear, geração de números randômicos, processamento de sinais e sequencial. Além disso, existe uma ampla comunidade de desenvolvedores contribuindo com Julia, fornecendo continuamente uma grande quantidade de pacotes externos, tornando seu desempenho muito equivalente, se não superior, a linguagens comerciais no mercado.

Com o objetivo de avaliar e comparar a linguagem Julia com o Fortran, implementamos, nessas duas linguagens, uma formulação do Boundary Element Methods (BEM) (ou Método dos Elementos de Contorno). Sabe-se que o BEM oferece certa facilidade na modelagem de muitos problemas, mas que sua eficiência tem sido um grande entrave para a análise de modelos em grande escala, daí limitando-se a resolver problemas com alguns milhares de graus de liberdade. Uma alternativa para suprir essa deficiência é a combinação entre o Fast Multipole Method (FMM) e o BEM, combinação referida como FMBEM, que possibilita a resolução de problemas com milhões de graus de liberdade reduzindo, de forma considerável, o custo computacional das operações de $O(N^2)$ para $O(N)$ ou $O(NlogN)$, conforme Nishimura et. al. (2002).

A ideia básica do FMM é tradução das iterações nó-a-nó (ou elemento-a-elemento) a célula-a-célula, onde essas células possuem uma estrutura hierárquica tipo árvore em que as menores (folhas) contém um número prescrito de elementos de contorno. O agrupamento desses elementos nas células e a expansão das soluções fundamentais em séries multipólos fazem com que ocorra a redução do custo computacional. Também é utilizado um solver iterativo como o Generalized Minimal Residual Method (GMRES), que dispensa o armazenamento da matriz dos coeficientes na memória do computador. Dessa forma, o FMM tem ampliado a aplicação do BEM a problemas de larga escala, antes, prejudicada pela falta de eficiência no processo de solução até então.

A proposta deste artigo é fazer uma comparação quanto ao custo computacional (tempo de execução) de duas implementações do FMBEM, sendo uma em linguagem Fortran e outra em linguagem Julia. O algoritmo implementado encontra-se diponível em Yijun Liu (2009). À medida do possível, considerando a característica de cada linguagem, o algoritmo implementado foi o mesmo nas duas linguagens, sendo as mesmas funções, as mesmas variáveis, etc..

2 O MÉTODO DOS ELEMENTOS DE CONTORNO

Basicamente, o BEM consiste na formulação de uma equação integral que elimina a discretização do domínio, considerando apenas a discretização do contorno com a qual é feita o estudo de toda a região. Essa discretização consiste na divisão do contorno em segmentos que são chamados de elementos de contorno nos quais as condições de contorno são aplicadas.

São apresentadas, a seguir, a formulação da equação integral de contorno e a discretização de uma malha usando o MEC convencional (BEM) para um um problema de potencial 2D, Fig. 1, em regime estacionário governado pela equação de Laplace.

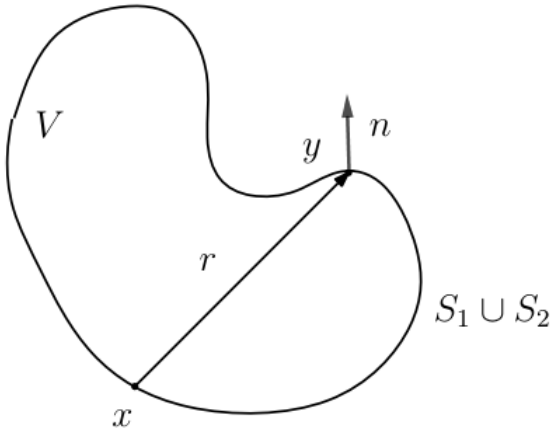

Figure 1: Domínio V e Contorno S

2.1 Equação Integral de Contorno

Seja u o campo potencial no domínio V, tal que

$$\bigtriangledown^2 u(x) = 0, \qquad \forall \in V; \tag{1}$$

com as seguintes condições de contorno

$$\begin{aligned} u(x) &= \overline{u}(x), \qquad \forall x \in S_1; \\ q(x) \equiv \frac{\partial u}{\partial n}(x) &= \overline{q}(x) \qquad \forall x \in S_2; \end{aligned} \tag{2}$$

onde $S = S_1 \cup S_2$ é o contorno do domínio V, n o vetor normal, x o ponto fonte, y o ponto campo e as variáveis com barra são os valores conhecidos no contorno.

A solução para o problema de valor de contorno descrito pela Eq. (2) pode ser escrita como:

$$u(x) = \int_S [u^*(x,y)q(y) - q^*(x,y)u(y)]dS(y), \qquad \forall x \in S, \tag{3}$$

onde $u^*(x,y)$ é a função de Green, Brebbia (1992), dada por

$$u^*(x,y) = \frac{1}{2\pi} log\left(\frac{1}{r}\right) \tag{4}$$

para um problema 2D, e

$$u^*(x,y) = \frac{\partial u^*(x,y)}{\partial n(y)} = \frac{1}{2\pi r}\frac{\partial r}{\partial n}, \tag{5}$$

em que r é a distância entre o ponto fonte x e o ponto campo y, conforme Fig. 2.

Fazendo $x \to S$, é obtida a formulação clássica do BEM denominada equação integral de contorno, dada por:

$$C(x)u(x) = \int_S [u^*(x,y)q(y) - q^*(x,y)u(y)]dS(y), \qquad \forall x \in S, \tag{6}$$

conforme Kane (1994).

2.2 Discretização

Supondo que as partes $S_1, S_2, S_3, \cdots, S_N$ do contorno S sejam aproximadas por segmentos retilíneos com um nó no centro de cada elemento (elementos constantes) é obtida a seguinte discretização da equação integral de contorno a partir da Eq. (6) para o nó i Fig. 2:

$$\frac{1}{2}u_i = \sum_{j=1}^{N} G_{ij}q_j - \sum_{j=1}^{N} H_{ij}u_j, \qquad i = 1, 2, 3, \cdots, N; \tag{7}$$

onde $C(x) = \frac{1}{2}$ por se tratar de um contorno suave, u_j e q_j $(j = 1, 2, 3, \cdots, N)$ são valores nodais de u e q em Γ_j, respectivamente, e os coeficientes são expressos por:

$$G_{ij} = \int_{\Gamma_j} u^*(x, y)dS(y) \quad \text{e} \quad H_{ij} = \int_{\Gamma_j} q^*(x, y)dS(y), \quad i, j = 1, 2, 3, \cdots, N; \tag{8}$$

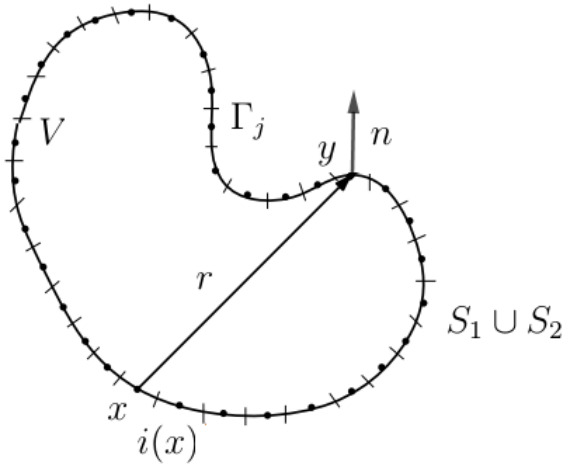

Figure 2: Discretização do Contorno por elementos constantes

Considerando o ponto fonte no nó i, é obtida, a partir da Eq. (6), a formulação discreta da equação integral de contorno e dada por:

$$\frac{1}{2} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1N} \\ H_{21} & H_{22} & \cdots & H_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ H_{N1} & H_{N2} & \cdots & H_{NN} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1N} \\ G_{21} & G_{22} & \cdots & G_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ G_{N1} & G_{N2} & \cdots & G_{NN} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix}$$

Aplicando as condições de contorno expressas pela Eq. (2) e permutando as colunas de modo que os termos desconhecidos fiquem do lado esquerdo e os demais do lado direito da equação matricial, tem-se:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

que pode ser escrita como

$$A\lambda = b, \tag{9}$$

onde A é a matriz dos coeficientes, λ é o vetor dos termos desconhecidos e b o vetor dos termos conhecidos. A construção da matriz A requer operações de $O(N^2)$, bem como o espaço para seu armazenamento. Além disso, a solução do sistema de equações (7) utilizando eliminação de Gauss requer operações de $O(N^2)$. São esses fatores que limitam a utilização do BEM convencional para problemas em larga escala.

3 O FAST MULTIPOLE BOUNDARY ELEMENT METHOD (FMBEM)

Considerando o *kernel* u^* dado pela Eq. (6)

$$\int_{S_0} u^*(x,y)q(y)dS(y), \tag{10}$$

em que S_0 é um subconjunto de S e está distante do ponto fonte x conforme a Fig. 3. Por conveniência é introduzida a notação complexa, onde

$$\begin{aligned} x &= z_0 = x_1 + ix_2 \\ y &= z = y_1 + iy_2 \end{aligned} \tag{11}$$

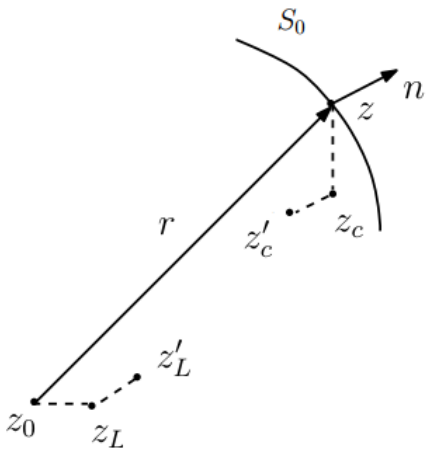

Figure 3: Pontos de Expansão para o Fast Multipole

Assim

$$u^*(x,y) = Re\{u^*(z_0,z)\}, \tag{12}$$

onde

$$u^*(z_0, z) = -\frac{1}{2\pi} log(z_0 - z), \tag{13}$$

Dessa forma, a integral dada pela Eq.(10) é a parte real da equação

$$\int_{S_0} u^*(z_0, z) q(z) dS(z) \tag{14}$$

Analogamente é dada a integral para o *kernel* q^* da pela Eq. (6),

$$\int_{S_0} q^*(z_0, z) u(z) dS(z) \tag{15}$$

$$\begin{aligned} q^*(z_0, z) &= \frac{\partial u^*}{\partial n} = (n_1 + i n_2) u^{*'} = n(z) u^{*'} \\ u^{*'} &\equiv \frac{\partial u^*}{\partial z}. \end{aligned} \tag{16}$$

Assim o *kernel* $q^*(x, y)$ é a parte real de $q^*(z_0, z)$.

3.1 Conceitos Teóricos

A seguir são apresentados conceitos importantes do FMBEM, conforme Yijun Liu (2009).

3.1.1 Expansão Multipolos (Momentos)

Dada a equação

$$u^*(z_0, z) = -\frac{1}{2\pi} log(z_0 - z). \tag{17}$$

Supondo z_c um ponto próximo ao ponto z (ponto campo) e $\| z - z_c \| \ll \| z_0 - z_c \|$ a Eq.(17) pode ser escrita como:

$$u^*(z_0, z) = -\frac{1}{2\pi} \left[log(z_0 - z_c) + log\left(1 - \frac{z - z_c}{z_0 - z_c}\right) \right]. \tag{18}$$

Fazendo

$$\xi = \frac{z - z_c}{z_0 - z_c} \tag{19}$$

e expandindo o segundo logaritmo em série de Taylor de modo a avaliar $log(1 - \xi)$, tem-se:

$$log(1 - \xi) = -\sum_{k=1}^{\infty} \frac{\xi^k}{k}, \qquad |\xi| < 1. \tag{20}$$

Assim, é obtida a equação

$$u^*(z_0, z) = \frac{1}{2\pi} \sum_{k=0}^{\infty} O_k(z_0 - z_c) I_k(z - z_c), \tag{21}$$

onde, por conveniência são introduzidas duas funções auxiliares:

$$\begin{aligned} I_k(z) &= \frac{z^k}{k!}, \quad k \geq 0; \\ O_k(z) &= \frac{(k-1)!}{z^k}, \quad k \geq 1; \end{aligned} \tag{22}$$

com $O_0(z) = -log(z)$ e derivas que satisfazem

$$\begin{aligned} I_k'(z) &= I_{k-1}(z), \quad k \geq 1; \\ O_k'(z) &= -O_{k+1}(z), \quad k \geq 0; \end{aligned} \tag{23}$$

Observando que, na Eq.(21), os pontos z_0 e z são separados devida a introdução do ponto z_c que é a base do FMM. Assim,

$$\int_{S_0} u^*(z_0, z) q(z) dS(z) = \frac{1}{2\pi} \int_{S_0} \left[\sum_{k=0}^{\infty} O_k(z_0 - z_c) I_k(z - z_c) \right] q(z) dS(z). \tag{24}$$

A Eq.(24) pode ser escrita como:

$$\int_{S_0} u^*(z_0, z) q(z) dS(z) = \frac{1}{2\pi} \sum_{k=0}^{\infty} O_k(z_0 - z_c) M_k(z_c), \tag{25}$$

que é a expansão multipólo, onde

$$M_k(z_c) = \int_{S_0} I_k(z - z_c) q(z) dS(z) \tag{26}$$

são os momentos próximos ao ponto z_c independentes da posição de z_0 e só precisam ser calculados uma única vez.

Das Eqs.(16) e (21), tem-se

$$\int_{S_0} q^*(z_0, z) u(z) dS(z) = \frac{1}{2\pi} \sum_{k=1}^{\infty} O_k(z_0 - z_c) N_k(z_c), \tag{27}$$

onde

$$N_k(z_c) = \int_{S_0} I_{k-1}(z - z_c) u(z) dS(z) \tag{28}$$

3.1.2 Translação Momento-Para-Momento (M2M)

Supondo um ponto z_c' para o qual é movido o ponto z_c (Fig. (3)), a translação M2M para a integral do *kernel* u^* é dada como segue:

$$\begin{aligned} M_k(z_c') &= \int_{S_0} I_k(z - z_c') q(z) dS(z) \\ &= \int_{S_0} I_k((z - z_c) + (z_c - z_c')) q(z) dS(z) \end{aligned} \tag{29}$$

Aplicando a fórmula binomial

$$(a+b)^n = \sum_{m=0}^{n} \begin{pmatrix} n \\ k \end{pmatrix} a^m \, b^{n-m}$$

à equação anterior, obtém-se

$$M_k(z_c') = \sum_{l=0}^{k} I_{k-l}(z_c - z_c') M_l(z_c), \tag{30}$$

De modo análogo ao que foi feito na translação M2M, obtém-se fórmulas similares a partir da integral do *kernel* q^*.

3.1.3 Expansão Local e Translação Momento-para-Momento (M2L)

Supondo um ponto z_L próximo ao ponto fonte z_0 (Fig. (3)), com $\| z_0 - z_L \| \ll \| z_c - z_L \|$. Segue da expansão multipólo descrita na Eq.(24):

$$\begin{aligned} f(z_0) &\equiv \int_{S_0} u^*(z_0, z) q(z) dS(z) \\ &\equiv \frac{1}{2\pi} \sum_{k=0}^{\infty} O_k(z_0 - z_c) M_k(z_c) \end{aligned} \tag{31}$$

A fim de obter a equação da expansão deve-se aplicar a expansão em série de Taylor de $f(z_0)$ em torno do ponto z_L,

$$f(z_0) = \sum_{l=0}^{\infty} f^{(l)}(z_L) I_l(z_0 - z_L) \tag{32}$$

Escrevendo a Eq.(31) em z_L

$$f^{(l)}(z_L) = \frac{(-1)^l}{2\pi} \sum_{k=0}^{\infty} O_{l+k}(z_L - z_c) M_k(z_c) \tag{33}$$

e substituindo na Eq.(32), obtém-se:

$$\int_{S_0} u^*(z_0, z) q(z) dS(z) = \sum_{l=0}^{\infty} L_l(z_L) I_l(z_0 - z_L) \tag{34}$$

que é a expansão local, onde os coeficientes são dados pela seguinte translação M2L:

$$L_l(z_L) = \frac{(-1)^l}{2\pi} \sum_{k=0}^{\infty} O_{l+k}(z_L - z_c) M_k(z_c) \tag{35}$$

De modo análogo, a translação M2L é aplicada para a integral do *kernel* q^*.

3.1.4 Translação Local-para-Local (L2L)

Supondo um ponto z'_L para o qual o ponto de expansão local z_L é movido (Fig. (3)). A partir da expansão local descrita na Eq.(34), obtém-se:

$$\begin{aligned} \int_{S_0} u^*(z_0, z) q(z) dS(z) &= \sum_{l=0}^{\infty} L_l(z_L) I_l(z_0 - z_L) \\ &= \sum_{l=0}^{\infty} L_l(z_L) I_l((z_0 - z'_L) + (z'_L - z_L)) \end{aligned} \tag{36}$$

Aplicando a fórmula binomial e a relação

$$\sum_{l=0}^{\infty} \sum_{m=0}^{\infty} = \sum_{m=0}^{\infty} \sum_{l=0}^{\infty} \tag{37}$$

obtém-se

$$\int_{S_0} u^*(z_0, z) q(z) dS(z) = \sum_{l=0}^{\infty} L_l(z'_L) I_l(z_0 - z'_L) \tag{38}$$

cujos coeficientes são dados a partir da translação L2L:

$$L_l(z'_L) = \sum_{m=l}^{p} I_{m-l}(z'_L - z_L) L_m(z_L) \tag{39}$$

Fazendo $k = m - l$, a Eq.(39) pode ser escrita como:

$$L_l(z'_L) = \sum_{k=0}^{p-l} I_k(z'_L - z_L) L_{l+k}(z_L). \tag{40}$$

3.2 Algoritmo

Os principais procedimentos no FMBEM, conforme Dondero (2007), Greengard (et. al. 1987), Greengard (et. al. 2009), são descritos como segue:
Passo 1: Discretização
Discretizar o contorno S de modo análogo ao BEM convencional, usando elementos constantes conforme Fig. (2).
Passo 2: Determinar a estrutura hierárquica (árvore) da malha.
Essa estrurura está organizada em células da seguinte forma: Primeiro constrói-se um quadrado circunscrito à malha, essa célula é do nvel 0. Depois o quadrado é dividido em 4 quadrados iguais em que as células que contém elementos de contorno, isto é, aquelas que contém nós, são as células do nível 1. Mais uma vez dividi-se cada quadrado em 4 quadrados iguais e as células que possuem nós são do nível 2 e assim dividi-se sucessivamente até que seja alcançada a quantidade mínima de elementos de contorno por célula.
Passo 3: Cálculo dos momentos multipolo
O cálculo dos momentos multipolo é dado pelas Eqs. (26) e (28), considerando o ponto z_c como o centro das células. Para as células do nível l que não são folhas, os momentos multipolo são calculados somando os momentos multipolo de suas células filhas aplicando a translação $M2M$, Eq. (30), que acontece desde os momentos do centro da célula filha ao centro da célula pai.
Passo 4: Cálculo dos coeficientes das expansões locais
Deve-se calcular a expansão local associada a uma célula C que é dada pela soma das contribuições dos elementos de contorno nas células da lista de interações de C (1), com as contribuições de todos os elementos de contorno das células que não são adjacentes à célula pai de C (2). A lista de interações de uma célula C em nível l é composta por todas as células bem separadas que pertencem à célula C, isto é, composta por todas as células do nível l que não possuem vértice comum. O cálculo das contribuições (1) é feito por meio da translação $M2L$ e o cálculo das contribuições (2) se faz transladando a expansão local desde o interior da célula pai de C ao centro de C, utilizando a translação $L2L$. Este processo se repete a partir de $l = 2$ com incremento 1 até chegar às folhas.
Passo 5: Avaliação das integrais de contorno
Calcular o lado direito da Eq. (6). As contribuições dos elementos que pertencem à célula que contém o ponto fonte z_0 e as células que são adjacentes a C são calculadas pelo BEM convencional. A contribuição de todo o resto é dada pela expansão local Eq. (31) que, por sua vez, utiliza os coeficientes calculados no Passo 4.
Passo 6: Iterações da solução
Atualizar o vetor de valores desconhecidos do sistema dado pela Eq. (26) e continuar o Passo 3 para o produto matriz-vetor a fim de que a solução convirja dentro de uma tolerância dada.

4 RESULTADOS NUMÉRICOS

Implementamos o FMBEM em um problema que consiste em uma placa quadrada com uma quantidade crescente de furos regularmente distribuídos nas direções horizontal e vertical.

A Fig. (4), a seguir, representa um caso com 16 furos. O tamanho dos furos é modificado de modo que a razão entre a área total dos furos e a área da placa seja constante e igual a 12,47% de acordo com Liu e Nishimura (2006).

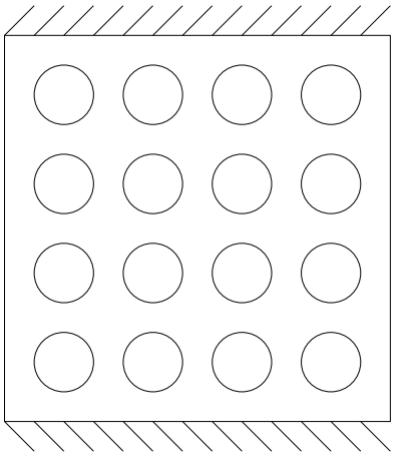

Figure 4: Placa Com Furos

O processamento foi realizado em um computador notebook Acer, com processador Intel i5 (2.3 GHz de clock), 8 G Bytes de memória RAM. O compilador Fortran usado foi o Mingw 6.3.0, e a linguagem Julia usada foi a versão 0.5.1., os resultados constam da tabela 1 a seguir.

Table 1: Tempo de processamento em segundos.

Número Total de Furos na Placa	Número Total de Elementos	Tempo de Execução (s)		Razão
		Fortran (T1)	Julia (T2)	T2/T1
16	336	0.1562	2.5294	16.2
49	864	0.4532	2.4724	5.5
100	1680	1.0	3.4341	3.4
169	2784	2.2188	4.4074	1.9
256	4176	2.9686	4.9862	1.7
361	5856	3.7968	8.8538	2.3
400	6480	4.8906	10.3828	2.1
625	10080	7.1094	16.4706	2.3
900	14480	10.0937	22.2	2.2
1225	19680	16.2813	46.6203	2.9
1600	25680	19.9531	77.8269	3.9

Os resultados demonstram que a linguagem Julia foi mais lenta que a linguagem Fortran em todos os casos. Para o caso com maior número de variáveis, a razão entre os tempos de execução do Julia e do Fortran foi $3,9$, ou seja, Julia gastou aproximadamente quatro vezes o tempo do Fortran. Acreditamos que seja viável otimizar o fluxo de operações do programa implementado em Julia tonando-o ainda mais rápido. Como se trata de um trabalho inicial, no houve tempo para um investimento maior nessa proposta de melhoria. Entretanto, este aspecto será contemplado em trabalhos futuros.

5 CONCLUSÕES

Este trabalho fez uma comparação entre o tempo de execução de um algoritmo de FMBEM, aplicado a um problema de potencial 2D, implementado em duas linguagens diferentes, Julia e Fortran. Foi analisado uma placa com furos, com o número de furos variando entre 16 e 1600. A linguagem Fortran se mostrou mais rápida que a linguagem Julia. Entretanto, considerando as facilidades inerentes à linguagem Julia, como uma linguagem compilada em tempo real, com bibliotecas gráficas eficientes e fáceis de usar, ela se mostra como uma ótima opção para ser usada na implementação de métodos computacionais. Um fator importante é que a linguagem Julia é uma linguagem de código aberto (open source), portanto acessível a todos. E vale ressaltar que uma boa perspectiva para a linguagem Julia, está na substituição de linguagens comerciais, como, por exemplo, o MATLAB.

REFERÊNCIAS

Brebbia, C.A. and Dominguez, J., 1992. Boundary Elements, An Introductory Course, McGrawHill.

Dondero, M., Cisilino, A. P. and Stavroulakis, G., 2007. Implementación de una formulación rapida de multipolos aplicados al método de los elementos de contorno. Mecânica Computacional, Vol XXVI, 468-484.

Greengard, L. and Beatson, R., 2009. A short course on fast multipole methods, University of Canterbury and New York University.

Greengard, L.F. and Rokhlin, V., 1987. A fast algoritmam for particle simulations, *J Comput Phys*, 73(2):325-48.

Kane, J.H., 1994. Boundary element analysis in engineering continuum mechanics, Prentice-Hall.

Liu, Y.J, 2009. The fasmultipole boundary element method. Cambridge.

Liu, Y.J. and Nishimura, N., 2006. The fasmultipole boundary element method for potential problems: a tutorial, Engineering Analysis with Boundary Elements, 30(5):371-381.

Nishimura, N. and Liu, Y.J., 2002. The fasmultipole accelerate boundary integral equation methods, Appl Mech Rev, 2002, Vol 55(4(July)), 299-324.