Journal of Computational Interdisciplinary Sciences (2013) 4(2): 1-6

Printed version ISSN 1983-8409 / Online version ISSN 2177-8833
http://epacis.net
doi: 10.6062/jcis.2013.04.02.0072

Relation between accuracy and computational time for boundary element method applied to Laplace equation

Olavo H. Menin[1] and Vanessa Rolnik[2]

Manuscript received on March 19, 2013 / accepted on March 28, 2013

ABSTRACT

The aim of this paper is to report a relation observed between accuracy and computational time to the boundary element method (BEM) applied to solve the Laplace equation. For this purpose, the BEM was implemented in C language and the program was tested for boundary value problems with known analytical solution. Moreover, the results show that the program is reliable, accurate and fast.

Keywords: boundary value problem, elliptic partial differential equations, numerical methods, computational cost.

1 INTRODUCTION

A boundary value problem (BVP) is composed of a partial differential equation (PDE) which governs the dependent variable inside the domain and appropriated boundary conditions. In the physical sciences and engineering there are several phenomena for which the mathematical modeling results in a BVP. For example, the problem of calculating the temperature in the internal points of an object from information about temperature or heat flux on its surface, the problem of determining the deformations and stresses inside a concrete beam which is subject to external loads and the problem of electrical impedance tomography in which the goal is to reconstruct an internal image of a body from measures of the electrical potential or current flow on its surface.

There are analytical methods to calculate solutions of BVPs. However, they only solve problems with both simple geometries of the domain and boundary conditions. In most practical situations, on the other hand, to find an analytical solution for a BVP is a task quite difficult or even impossible with the currently known techniques. To simplify the model by making approaches and considerations for determining an analytical solution is not always a good alternative since these simplifications may deviate the model from the real phenomenon and generate unreliable results.

The most used alternative is to resort to numerical methods. The available computational algorithms to solve BVPs are able to assess the dependent variable inside the domain with high precision and with increasing speed due to the great development of computers in recent decades. The main numerical methods are the Finite Difference Method (FDM), the Finite Element Method (FEM) and the Boundary Element Method (BEM). The first two are more traditional, and the FEM is the most widely used today.

Under development for few decades, BEM appears as a promising alternative technique for solving boundary value problems. An attractive feature of this method is the need to discretize only the boundary of the domain [1, 2], while FDM and FEM require discretization of the whole domain. Furthermore BEM is relatively easy to be implemented and can be naturally applied for problems involving complicated geometries and general boundary conditions. Recent papers have applied BEM, for ex-

Correspondence to: Olavo H. Menin – E-mail: olavohmenin.ifsp@gmail.com
[1]Instituto Federal de Educacão Ciência e Tecnlogia de São Paulo, Sertãozinho, SP, Brazil.
[2]Universidade de São Paulo, Ribeirão Preto, SP, Brazil.

ample, to reconstruct the boundary condition coefficients in transient heat conduction [3], determination of the convection coefficient [4] or for location inclusions through electrical impedance tomography [5, 6, 7] as well as in more theoretical study or in the improvement of the method, as to deal with singular integrals [8].

The size of the linear system depends on the number of points or elements generated by the discretization. Since BEM reduces the problem in one dimension, the linear systems generated are considerably lower than in FDM or FEM. In turn, the size of the linear system influences the computational time to solve the problem. Moreover, the numerical methods and their computational implementation are affected by errors of different sources. For example, in the BEM, the possible sources of errors are the discretization of PDE, approximation of the boundary into elements, approximation of the boundary functions, numerical integration, numerical solution of the linear system [9] as well as rounding errors caused by finite precision of computers. Therefore, it is essential to the reliability of the numerical solution of a BVP to analyze the actual error and the total time employed.

We have implemented BEM in C language and have performed a study on two key aspects of the program: accuracy and computational time. The goal of this study is to report a relation between these two aspects. To show the results, we chose two BVPs with known analytical solutions, compounded by Laplace equation into square domains and boundary conditions of Dirichlet and Newmann types.

2 BOUNDARY ELEMENT METHOD

The BVPs that we have chosen as examples to be solved numerically by BEM consist of a domain $\Omega \subseteq \mathbb{R}^2$ closed by the boundary $\partial\Omega$. The dependent variable ϕ is governed by the Laplace equation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0, \qquad \text{for } (x, y) \in \Omega, \qquad (1)$$

and of boundary conditions of Dirichlet and Neumann types

$$\phi(x, y) = \phi, \qquad \text{for } (x, y) \in \partial\Omega_1, \qquad (2)$$

$$\frac{\partial \phi(x, y)}{\partial n} = J, \qquad \text{for } (x, y) \in \partial\Omega_2, \qquad (3)$$

with $\partial\Omega = \partial\Omega_1 \cup \partial\Omega_2$ and $\partial\Omega_1 \cap \partial\Omega_2 = \emptyset$.

The procedure to solve the BVP given by Eqs. (1)-(3) using BEM is based on book of Ang [10]. First, one must to discretize the boundary $\partial\Omega$ in a prefixed number N of elements in such a way that the final extremity of the last element are made to coincide with the initial extremity of the first element, such that

$$\partial\Omega \approx \partial\Omega^{(1)} \cup \partial\Omega^{(2)} \cup \ldots \cup \partial\Omega^{(N)}.$$

By convention, the elements are numbered following the counter clockwise direction, so that the normal unitary vector, $\vec{n}$, always points to the external domain Ω. Figure 1 shows the discretization of the external boundary $\partial\Omega$ using 6 elements. The points which represents the extremity of each element are numbered by P_i, $i = 1, \ldots, 7$.

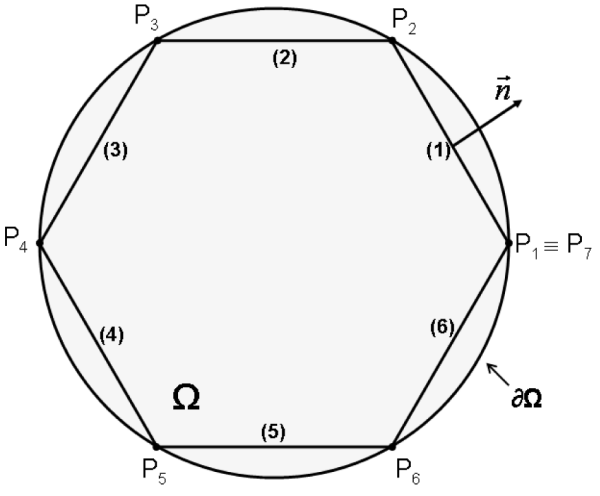

Figure 1 – Circular domain Ω with boundary $\partial\Omega$ discretized in 6 straight lines elements.

The adopted boundary elements are straight lines and of constant type. In this case the values of ϕ and J are considered constant along of each element and assume the values $\overline{\phi}$ and $\overline{J}$, respectively, at the midpoint $(\overline{x}^{(k)}, \overline{y}^{(k)})$ of element k. Then, the boundary conditions, Eqs. (2) and (3), in the discrete form become

$$\overline{\phi}^{(k)} = \phi\left(\overline{x}^{(k)}, \overline{y}^{(k)}\right) \qquad \text{for } k = 1, 2, \ldots, N$$

$$\overline{J}^{(k)} = \frac{\partial \phi}{\partial n}\left(\overline{x}^{(k)}, \overline{y}^{(k)}\right) \qquad \text{for } k = 1, 2, \ldots, N$$

Following, one must to discretize the integral equations on the boundary. This procedure yields two equation, one for the points (ξ, η) inside the domain Ω,

$$\phi(\xi, \eta) = \sum_{k=1}^{N} \left[\overline{\phi}^{(k)} F_2^{(k)}(\xi, \eta) - \overline{J}^{(k)} F_1^{(k)}(\xi, \eta)\right], \qquad \text{for } (\xi, \eta) \in \Omega, \quad (4)$$

and another for the points (ξ, η) on the boundary $\partial\Omega$,

$$\frac{1}{2}\overline{\phi}(\xi,\eta) = \sum_{k=1}^{N}\left[\overline{\phi}^{(k)} F_2^{(k)}(\xi,\eta) - \overline{J}^{(k)} F_1^{(k)}(\xi,\eta)\right], \quad \text{for } (\xi,\eta) \in \partial\Omega, \quad (5)$$

where $F_1^{(k)}(\xi,\eta)$ and $F_2^{(k)}(\xi,\eta)$ are given by

$$F_1^{(k)}(\xi,\eta) = \int_{\partial\Omega^{(k)}} \Phi(x, y; \xi, \eta) ds(x, y), \quad (6)$$

$$F_2^{(k)}(\xi,\eta) = \int_{\partial\Omega^{(k)}} \frac{\partial}{\partial n}\left[\Phi(x, y; \xi, \eta)\right] ds(x, y), \quad (7)$$

and $\Phi(x, y; \xi, \eta) = (1/4\pi)\ln[(x-\xi)^2 + (y-\eta)^2]$ is called *fundamental solution*.

In the computational procedure Eq. (5) must be calculated N times for $(\xi, \eta) = (\overline{x}^{(m)}, \overline{y}^{(m)})$, $m = 1, 2, \ldots, N$. Whereas that the boundary conditions (Eqs. 2 and 3) provide the value of either $\overline{\phi}$ or $\overline{J}$ on each element, but not both, Eq. (5) produces a linear system of N equations with N unknowns.

Therefore the solution on the boundary is calculated first by solving the linear system and then, knowing the values of $\overline{\phi}$ and $\overline{J}$ in each boundary element, Eq. (4) can be used to calculate the value of ϕ in the internal points of the domain.

It must be noted that the Eq. (5) constructs a linear system which depends only on the coordinates of the boundary elements and the types of the boundary condition. An important consequence is that the size of the linear system is determined by the number of elements used to discretized the boundary and does not depend on the number of internal points where the solution will be calculated. Thus, the internal grid can be considerably fine and the method can reach a solution with high accuracy by solving a relatively small linear system. Another characteristic is that the linear system is solved only once and does not need to be solved again during the calculation of ϕ in the points of the internal grid, independently of the number of these points.

3 NUMERICAL TESTS AND RESULTS

A program has been implemented in C language. The Gaussian elimination with partial pivoting solves the linear system generated by Eq. (5). The integrals in Eq. (6)-(7) are generally solved numerically using, for instance, the quadrature of Gauss. However, this work uses an analytical solution for the integrals, given in Ref. [10]. The tests were performed on a Intel Core™ 2 Duo, 2.5 GHz and 2 GB RAM.

This section shows the results and discussions around two aspects of the program, accuracy and computational time, and reports a relation observed between these two aspects. Despite the program is versatile admitting boundary conditions of Dirichlet, Neumann or mixed types, rectangular and circular domains or any boundary geometry, the results are presented taking two BVPs with unitary square domain. The first one, BVP I, has Dirichlet type condition on the whole boundary and the second one, BVP II, has Neumann type condition on two sides and Dirichlet elsewhere of the boundary, as are described below and shown in Figure 2.

BVP I

$$\begin{cases} \nabla^2\phi = 0 & \text{for } \Omega = (0,1)\times(0,1) \\ \phi(x,0) = x^2 \text{ and } \phi(x,1) = x^2 - 1 & \text{for } x \in [0,1] \\ \phi(0,y) = -y^2 \text{ and } \phi(1,y) = 1 - y^2 & \text{for } y \in [0,1] \end{cases}$$

The analytical solution of BVP I is

$$\phi(x, y) = x^2 - y^2$$

BVP II

$$\begin{cases} \nabla^2\phi = 0 & \text{for } \Omega = (0,1)\times(0,1) \\ \dfrac{\partial\phi}{\partial n}(x,0) = 0 \text{ and } \dfrac{\partial\phi}{\partial n}(x,1) = 0 & \text{for } x \in [0,1] \\ \phi(0,y) = 0 \text{ and } \phi(1,y) = \cos(\pi y) & \text{for } y \in [0,1] \end{cases}$$

The analytical solution of BVP II is

$$\phi(x, y) = \frac{\operatorname{senh}(\pi x)\cos(\pi y)}{\operatorname{senh}(\pi)}$$

To compare the analytical and numerical solutions we have chosen 361 internal points of each domain (BVP I and BVP II) uniformly distributed in a 19 × 19 cartesian grid with spacing $\delta = 0.05$. The comparison was carried out through two different errors defined by

$$R_{max} = \| \phi_{exact} - \phi_{num} \| = \max_{\substack{i=1..19 \\ j=1..19}} \left\{ | \phi_{exact}(x_i, y_j) - \phi_{num}(x_i, y_j) | \right\}, \quad (8)$$

$$R_{medium} = \frac{1}{361} \times \sum_{i,j=1}^{19} | \phi_{exact}(x_i, y_j) - \phi_{num}(x_i, y_j) | . \quad (9)$$

The program calculated the medium (R_{medium}) and maximum (R_{max}) errors to values of the number N of boundary elements varying between 20 and 320 elements, for both BVPs. The

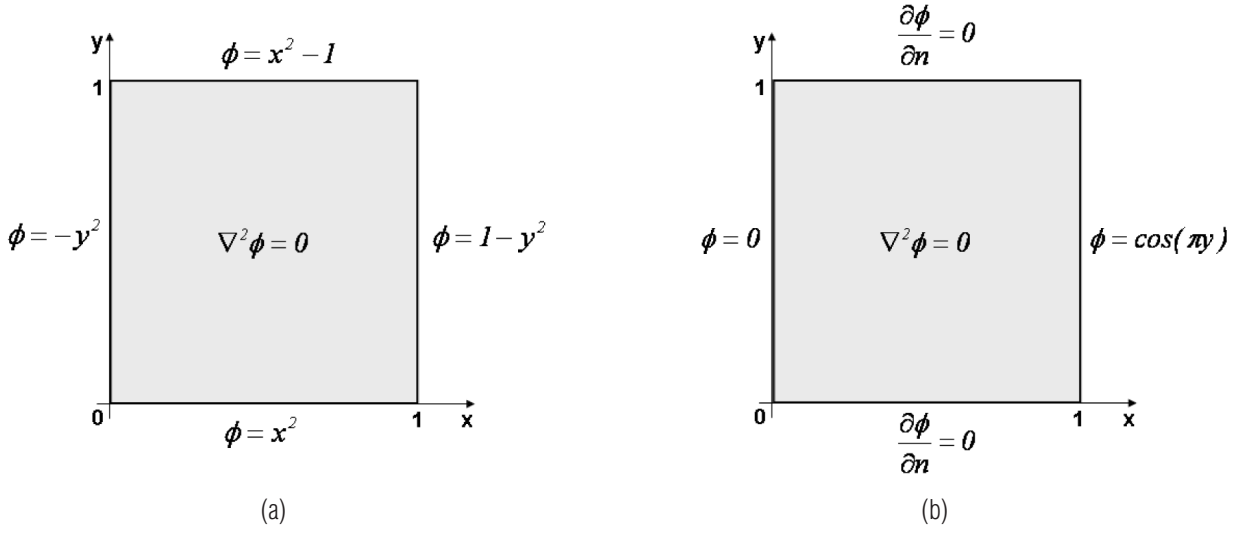

Figure 2 – Domain and boundary conditions of (a) BVP I and (b) BVP II.

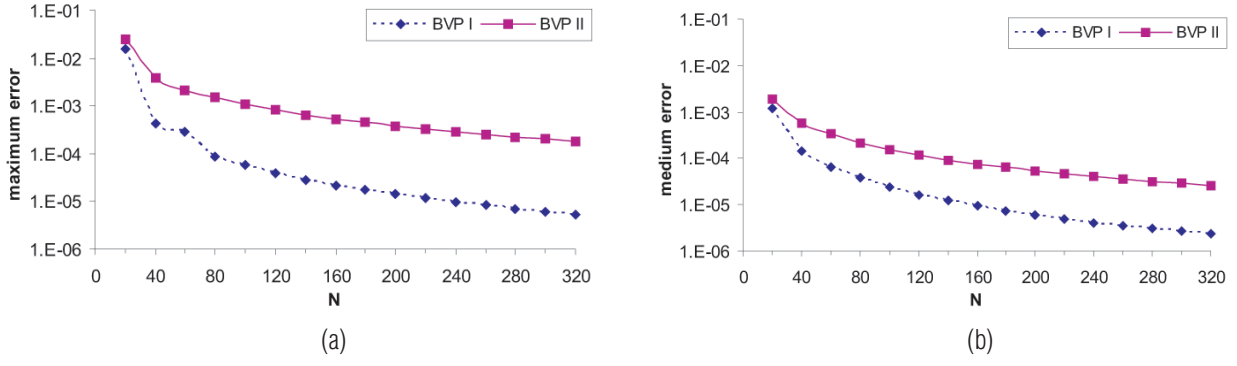

Figure 3 – (a) Maximum error and (b) medium error *versus* number of boundary elements.

results can be seen in Figure 3(a) and Figure 3(b), where the error axes are in logarithmic scale. The graphics show that both errors decrease significantly when the number of boundary elements increases. More precisely, the rate of decrease of error is high when the values of N are small and smoother for larger values of N.

The program also calculated the computational time employed in each previous tests. It was measured the time of run of the program by setting the number of internal points evaluated in 361 and varying N between 20 and 320. The results are shown in Figure 4(a). Another test was carried out setting the number of boundary elements in $N = 100$, $N = 200$ and $N = 300$ and measuring the time of run varying the number of internal points evaluated between 10 000 and 100 000, according to Figure 4(b). The values for computational time were almost the same for BVP I and BVP II. Thus there is only one graphic for each test in Figure 4.

As expected, results show that the computational time increases with the number N of boundary elements. However, it remains below 0.3 second even for a large number of N ($N = 320$). The computational time also increases with the number of internal points evaluated and the relation between them is linear with a constant rate depending on the number of boundary elements, as shown in Table 1.

Table 1 – Rate of increase of the computational time per internal point evaluated and number of boundary elements.

N	rate of increase (second per internal point)
100	8.5×10^{-5}
200	1.7×10^{-4}
300	2.5×10^{-4}

Observing that the maximum error decreases with the number N of boundary elements while the computational time increases with N, we analyzed the behavior of the multiplication of the maximum error and the computational time (error $\times$ time). For

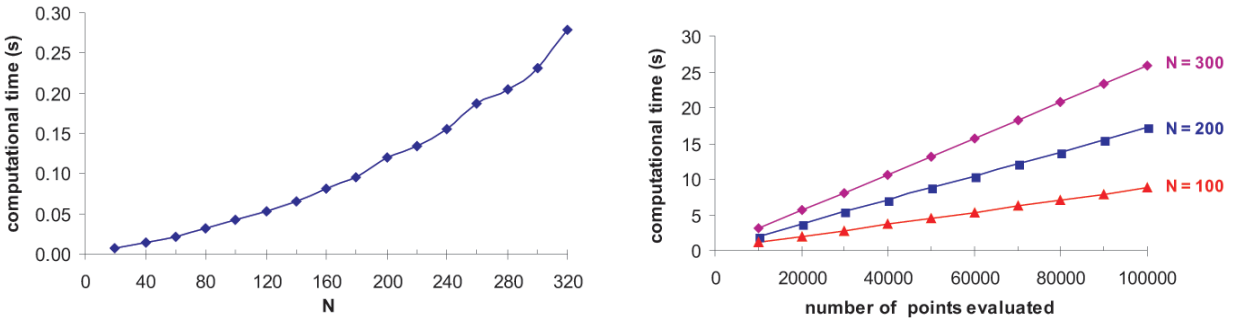

Figure 4 – Computational time *versus* number of boundary elements (a) and computational time *versus* number of points evaluated (b).

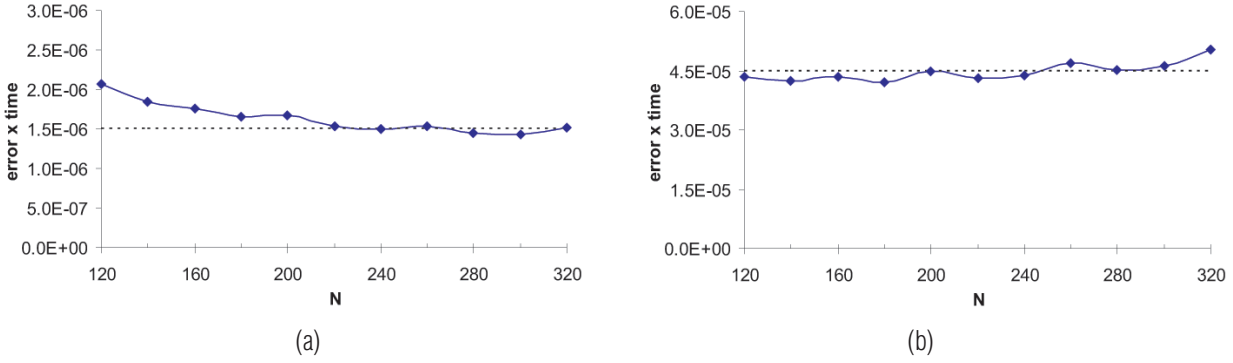

Figure 5 – Maximum error *versus* computational time considering $N > 120$ for (a) BVP I and (b) BVP II.

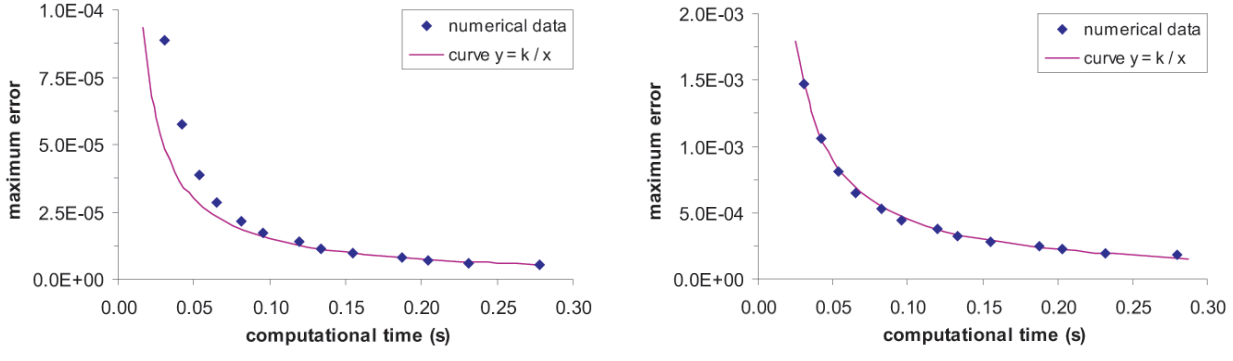

Figure 6 – Maximum error *versus* computational time for (a) BVP I and (b) BVP II and the equilateral hyperbola $y = k/x$. For BVP I, $k = 1.5\times10^{-6}$, and for BVP II, $k = 4.5\times10^{-5}$.

both BVPs the multiplication has tended to assume a constant value for $N > 180$ as shown in Figure 5. For the BVP I the values of error $\times$ time tends to the value of 1.5×10^{-6} while for the BVP II tends to 4.5×10^{-5}.

These results indicate that there is an inverse proportion between maximum error and computational time. This relation can be observed by plotting the values of maximum error (y) *versus* computational time (x) and the equilateral hyperbola $y = k/x$, so that $k = 1.5 \times 10^{-6}$ for BVP I and $k = 4.5 \times 10^{-5}$ for BVP II, according to Figure 6.

4 CONCLUSIONS

The results show that the BEM is extremely fast to solve the Laplace equation. In fact, tests taken less than 0.3 second to

evaluate 361 internal points even using a large number of boundary elements ($N = 320$). Also, BEM is accurate since the maximum error was below 10^{-5} for 320 boundary elements.

It was verified that for a fixed number of boundary elements there is a linear relation between the computational time and number of internal points evaluated. Moreover, the slope of this linear relation depends of the number of boundary elements fixed.

Regarding to the maximum error and the computational time, they are related by an approximately inverse proportion. In fact, the multiplication of the maximum error and the computational time tends to assume a constant value. Choosing this value to the constant k in equilateral hyperbola $y = k/x$, the curve fits approximately with the numerical data. Nevertheless, the value of k can vary since it depends on several factors such as type and value of the boundary conditions, form of integration and resolution of linear system and even the configuration of the computer used.

ACKNOWLEDGMENTS

The authors would like to acknowledge *FAPESP*, Grant No. 08/01284-4, for the financial support given to this research.

REFERENCES

[1] BREBBIA CA. 1978. The boundary element method for engineers, Pentech Press, Plymonth.

[2] BREBBIA CA & DOMINGUEZ J. 1989. Boundary elements: an introductory course, WIT Press, Southampton.

[3] ONYANGO TTM, INGHAM DB & LESNIC D. 2009. Applied Mathematics and Computation, 207: 569–575.

[4] MARTIN TJ & DULIKRAVICH GS. 1998. Transactions of the ASME, 120: 328–334.

[5] MENIN OH & ROLNIK V. 2011. International Journal of Modern Physics C, 22: 825–839.

[6] MENIN OH, ROLNIK V & MARTINEZ AS. 2013. Revista Brasileira de Ensino de Física, 34: 2304.

[7] XU Y, DONG F & TAN C. 2010. Engineering Analysis with Boundary Elements, 34: 876–883.

[8] RIBEIRO GO, RIBEIRO TSA, JORGE AB & CRUSE TA. 2009. J. of the Braz. Soc. of Mech. Sci. & Eng., XXXXI: 261–268.

[9] KIRKUP SM & HENWOOD DJ. 1994. Appl. Math. Modelling, 18: 32–38.

[10] ANG WT. 2007. A beginner's course in boundary element methods, Universal Publishers, Boca Raton.