Efficient Evaluation of Influence Coefficients in Three-Dimensional Extended Boundary-Node Method for Potential Problems*)

Taku ITOH, Ayumu SAITOH[1)], Atsushi KAMITANI[2)] and Hiroaki NAKAMURA[3)]

Seikei University, 3-3-1, Kichijoji-Kitamachi, Musashino, Tokyo 180-8633, Japan
[1)]*University of Hyogo, 2167, Shosha, Himeji, Hyogo 671-2280, Japan*
[2)]*Yamagata University, 4-3-16, Johnan, Yonezawa, Yamagata 992-8510, Japan*
[3)]*National Institute for Fusion Science, 322-6 Oroshi-cho, Toki, Gifu 509-5292, Japan*

(Received 21 December 2010 / Accepted 14 February 2011)

For the purpose of speed-up of the three-dimensional eXtended Boundary-Node Method (X-BNM), an efficient algorithm for evaluating influence coefficients has been developed. The algorithm can be easily implemented into the X-BNM without using any integration cells. By applying the resulting X-BNM to the Laplace problem, the performance of the algorithm is numerically investigated. The numerical experiments show that, by using the algorithm, computational costs for evaluating influence coefficients in the X-BNM are reduced considerably. Especially for a large-sized problem, the algorithm is efficiently performed, and the computational costs of the X-BNM are close to those of the Boundary-Element Method (BEM). In addition, for the problem, the X-BNM shows almost the same accuracy as that of the BEM.

Keywords: boundary-node method, boundary-element method, influence coefficient, integration cell, implicit function

DOI: 10.1585/pfr.6.2401106

1. Introduction

Potential problems often appear in various fields such as plasma physics and fusion science. The Boundary-Element Method (BEM) has been applied to these problems and has produced many attractive results; however, a boundary surface must be divided into a set of boundary elements before the BEM is applied to the problems.

Alternatively, Chati *et al.* have proposed the Boundary-Node Method (BNM) [1] as a numerical method for solving three-dimensional (3D) potential problems. In contrast to the BEM, the BNM requires only nodes on a boundary surface. However, the surface must be divided into a set of integration cells to evaluate surface integrals such as influence coefficients. Namely, the BNM still has a concept of boundary elements partly.

To remove integration cells completely, the 3D BNM has been recently reformulated. The reformulated method is called the eXtended Boundary-Node Method (X-BNM) [2]. Although a concept of boundary elements is no longer included in the X-BNM, the computational costs of the X-BNM are larger than those of the BEM. The purpose of this study is to develop an efficient algorithm for evaluation of influence coefficients and to incorporate it to the 3D X-BNM.

2. Influence Coefficients

As a typical potential problem, we consider a 3D Laplace problem. If the X-BNM is applied to the problem, the influence coefficients, G_{ij} and H_{ij} $(i, j = 1, 2, \ldots, N)$, can be written as

$$G_{ij} \equiv \int_{S_j} w^*(\boldsymbol{x}, \boldsymbol{x}_i)\phi_j(\boldsymbol{x})\, \mathrm{d}S(\boldsymbol{x}), \tag{1}$$

$$H_{ij} \equiv \int_{S_j} \frac{\partial w^*}{\partial n}(\boldsymbol{x}, \boldsymbol{x}_i)\phi_j(\boldsymbol{x})\, \mathrm{d}S(\boldsymbol{x}) + \frac{\phi_j(\boldsymbol{x}_i)\Omega_i}{4\pi}, \tag{2}$$

where $w^*(\boldsymbol{x}, \boldsymbol{x}_i) \equiv (4\pi|\boldsymbol{x} - \boldsymbol{x}_i|)^{-1}$, and S_j is a part of the boundary surface ∂V contained in a sphere of radius R and center $\boldsymbol{x}_j$. In addition, $\phi_j(\boldsymbol{x})$ denotes a shape function corresponding to the jth boundary node $\boldsymbol{x}_j$ $(j = 1, 2, \ldots, N)$. Here, N is the number of boundary nodes, and Ω_i is a solid angle on $\boldsymbol{x}_i$. In the X-BNM, the shape functions are determined by the Moving Least-Squares (MLS) approximation [1]. Throughout the present study, the MLS approximation with $m = 1$ is employed. Here, m is the order of a complete nominal basis for the MLS approximation.

In the X-BNM, a boundary surface is assumed as an implicit surface $f(\boldsymbol{x}) = 0$ [3] and the shape function is assumed to have a support of radius R. Under the assumptions, G_{ij} and the 1st term of H_{ij} can be written in the form,

$$I = \int_S F\, \mathrm{d}S\ . \tag{3}$$

Here, S denotes a part of the implicit surface Π contained in a sphere of radius R and center $\boldsymbol{y}$. Different coordinates are used for the numerical integration of (3), depending on whether S contains a singularity $\boldsymbol{z}$ of $F(\boldsymbol{x})$ or not.

For the case where S does not contain any singularity

author's e-mail: taku@st.seikei.ac.jp

*) This article is based on the presentation at the 20th International Toki Conference (ITC20).

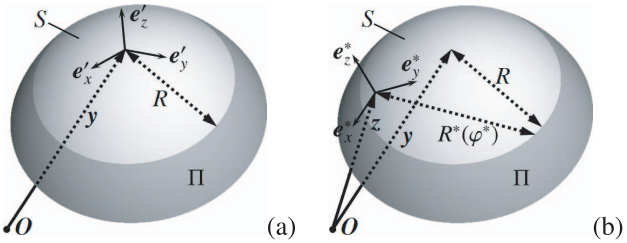

Fig. 1 Local Cartesian coordinate systems. (a) For the case where S does not contain any singularity of $F(\boldsymbol{x})$. (b) For the case where S contains a singularity $\boldsymbol{z}$ of $F(\boldsymbol{x})$.

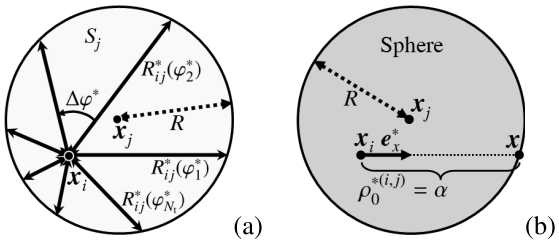

Fig. 2 (a) Relation between φ^*_m and $R^*_{ij}(\varphi^*_m)$ on S_j. (b) Distance α between $\boldsymbol{x}_i$ and a sphere of radius R and center $\boldsymbol{x}_j$ along the positive direction of $\boldsymbol{e}^*_x$.

of $F(\boldsymbol{x})$, we use the 3D polar coordinate (ρ, θ, φ) whose origin coincides with the sphere center $\boldsymbol{y}$. In addition, we employ a local Cartesian coordinate system $\langle \boldsymbol{y} : \boldsymbol{e}'_x, \boldsymbol{e}'_y, \boldsymbol{e}'_z \rangle$ shown in Fig. 1 (a). In this system, $\boldsymbol{e}'_z$ is first defined as $\boldsymbol{e}'_z \equiv \nabla f(\boldsymbol{y})/|\nabla f(\boldsymbol{y})|$. After that, $\boldsymbol{e}'_x$ is generated by Schmidt's orthogonalization. Finally, $\boldsymbol{e}'_y$ is defined as $\boldsymbol{e}'_y \equiv \boldsymbol{e}'_z \times \boldsymbol{e}'_x$. By using the system, an arbitrary point $\boldsymbol{x}$ is expressed by $\boldsymbol{x} = \boldsymbol{y} + \rho(\sin\theta\cos\varphi\,\boldsymbol{e}'_x + \sin\theta\sin\varphi\,\boldsymbol{e}'_y + \cos\theta\,\boldsymbol{e}'_z) \equiv \boldsymbol{g}(\rho, \theta, \varphi)$. Note that, on S, θ is a function of ρ and φ, i.e., $\theta = \theta(\rho, \varphi)$. This can be easily proved by using the implicit function theorem. Therefore the vector equation of S is given by $\boldsymbol{x} = \boldsymbol{g}(\rho, \theta(\rho, \varphi), \varphi)$ $(0 \le \rho \le R, 0 \le \varphi < 2\pi)$. By using the vector equation, (3) can be rewritten as follows:

$$I = \int_0^{2\pi} \mathrm{d}\varphi \int_0^R \mathrm{d}\rho G(\rho, \varphi), \tag{4}$$

where $G(\rho, \varphi) \equiv \rho F(\boldsymbol{g}(\rho, \theta, \varphi)) \left\{ [(\theta_\rho\, \rho)^2 + 1] \sin^2\theta + \theta_\varphi^2 \right\}^{1/2}$. Note that $\theta(\rho, \varphi)$ is determined by solving the following nonlinear equation:

$$f(\boldsymbol{g}(\rho, \theta, \varphi)) = 0. \tag{5}$$

For the case where S contains $\boldsymbol{z}$, a slightly different coordinate is employed. For this case, we use the 3D polar coordinate $(\rho^*, \theta^*, \varphi^*)$ whose origin is $\boldsymbol{z}$. In addition, we employ a local Cartesian coordinate system $\langle \boldsymbol{z} : \boldsymbol{e}^*_x, \boldsymbol{e}^*_y, \boldsymbol{e}^*_z \rangle$ shown in Fig. 1 (b). Here, $\boldsymbol{e}^*_x, \boldsymbol{e}^*_y$ and $\boldsymbol{e}^*_z$ are defined in the same manner as $\boldsymbol{e}'_x, \boldsymbol{e}'_y$ and $\boldsymbol{e}'_z$, respectively. By using the system, an arbitrary point $\boldsymbol{x}$ is expressed by $\boldsymbol{x} = \boldsymbol{z} + \rho^*(\sin\theta^*\cos\varphi^*\boldsymbol{e}^*_x + \sin\theta^*\sin\varphi^*\boldsymbol{e}^*_y + \cos\theta^*\boldsymbol{e}^*_z) \equiv \boldsymbol{g}^*(\rho^*, \theta^*, \varphi^*)$. Therefore the vector equation is given by $\boldsymbol{x} = \boldsymbol{g}(\rho^*, \theta^*(\rho^*, \varphi^*), \varphi^*)$ $(0 \le \rho^* \le R^*(\varphi^*), 0 \le \varphi^* < 2\pi)$. By using the vector equation, (3) can be rewritten as follows:

$$I = \int_0^{2\pi} \mathrm{d}\varphi^* \int_0^{R^*(\varphi^*)} \mathrm{d}\rho^* G^*(\rho^*, \varphi^*), \tag{6}$$

where $G^*(\rho^*, \varphi^*)$ is defined in the same manner as $G(\rho, \varphi)$. Note that the equation $\rho^* = R^*(\varphi^*)$ representing the edge of S is determined by solving the nonlinear system:

$$\sigma^*_1(\rho^*, \theta^*) \equiv f(\boldsymbol{g}(\rho^*, \theta^*, \varphi^*)) = 0, \tag{7}$$

$$\sigma^*_2(\rho^*, \theta^*) \equiv |\boldsymbol{g}(\rho^*, \theta^*, \varphi^*) - \boldsymbol{y}|^2 - R^2 = 0. \tag{8}$$

3. Efficient Evaluation of G_{ij} and H_{ij}

3.1 Nonlinear equation (5)

In evaluating influence coefficients G_{ij} and H_{ij}, the nonlinear equation (5) is solved many times to determine $\theta(\rho, \varphi)$ on each integration point. For numerical evaluation of G_{ij} and H_{ij}, the trapezoid formula and the Gauss-Legendre quadrature are applied to the φ- and ρ-directions, respectively. Throughout this paper, N_t and N_g denote the number of integration points for the trapezoid formula and that for the Gauss-Legendre quadrature, respectively. In evaluating G_{ij} and H_{ij}, the nonlinear equation (5) is normally solved $N^2 N_\mathrm{t} N_\mathrm{g}$ times. This is because $N_\mathrm{t} N_\mathrm{g}$ integration points are required for evaluating each set of G_{ij} and H_{ij}. It must be noted here that, in evaluating G_{ij} and H_{ij} $(i = 1, 2, \ldots, N)$, integration points do not change at all. Therefore, the nonlinear equation has only to be solved $NN_\mathrm{t}N_\mathrm{g}$ times before starting evaluation of G_{ij} and H_{ij}. Namely, the computational costs can be reduced considerably.

For solving the nonlinear equation (5), we employ the Newton method. Let ρ_n $(n = 1, 2, \ldots, N_\mathrm{g})$ be integration points of Gauss-Legendre quadrature for the ρ-direction on S. In addition, let φ_m $(m = 1, 2, \ldots, N_\mathrm{t})$ be $(m-1)\Delta\varphi$, where $\Delta\varphi = 2\pi/N_\mathrm{t}$. To determine $\theta(\rho_1, \varphi_m)$, we set $\pi/2$ as an initial solution of the Newton method. After determining $\theta(\rho_1, \varphi_m)$, we set $\theta(\rho_{n-1}, \varphi_m)$ as an initial solution for determining $\theta(\rho_n, \varphi_m)$ $(n = 2, 3, \ldots, N_\mathrm{g})$.

3.2 Nonlinear system, (7) and (8)

For solving the nonlinear system, (7) and (8), we also employ the Newton method. Let φ^*_m $(m = 1, 2, \ldots, N_\mathrm{t})$ be $(m-1)\Delta\varphi^*$, where $\Delta\varphi^* = 2\pi/N_\mathrm{t}$. In addition, let $\rho_m^{*(i,j)} = R^*_{ij}(\varphi^*_m)$ $(m = 1, 2, \ldots, N_\mathrm{t})$ be the mth length between $\boldsymbol{x}_i$ and an edge of S_j (see Fig. 2 (a)). To obtain initial solutions $(\rho_0^{*(i,j)}, \theta_0^{*(i,j)})$ of the Newton method for determination of $\rho_1^{*(i,j)} = R^*_{ij}(\varphi^*_1)$, we first consider equations: $\boldsymbol{x} = \boldsymbol{x}_i + \alpha\,\boldsymbol{e}^*_x$ and $|\boldsymbol{x} - \boldsymbol{x}_j|^2 = R^2$. The former equation is obtained by substituting $\rho^* = \alpha$, $\theta^* = \pi/2$, $\varphi^* = \varphi^*_1 = 0$ and $\boldsymbol{z} = \boldsymbol{x}_i$ into $\boldsymbol{x} = \boldsymbol{g}^*(\rho^*, \theta^*, \varphi^*)$. The later equation is obtained by substituting $\boldsymbol{y} = \boldsymbol{x}_j$ into (8). From these equations, we obtain a quadratic equation: $\alpha^2 + 2\,\boldsymbol{e}^*_x \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)\,\alpha + |\boldsymbol{x}_i - \boldsymbol{x}_j|^2 - R^2 = 0$. The quadratic equation has solutions in case $D \equiv b^2 - c \ge 0$ is satisfied, where $b = \boldsymbol{e}^*_x \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)$ and $c = |\boldsymbol{x}_i - \boldsymbol{x}_j|^2 - R^2$. The solutions $\alpha = -b \pm \sqrt{D}$ indicate distances between $\boldsymbol{x}_i$ and a sphere of radius R and center $\boldsymbol{x}_j$ along $\boldsymbol{e}^*_x$ (see Fig. 2 (b)). Note that $D \ge 0$ is always satisfied since $\boldsymbol{x}_i \in S_j$, that is, $\boldsymbol{x}_i$ is inside the sphere. Here, we adopt $\rho_0^{*(i,j)} = -b + \sqrt{D}$, and $\theta_0^{*(i,j)}$ is determined by solving

a nonlinear equation, $f(\boldsymbol{g}(\rho_0^{*(i,j)}, \theta_0^{*(i,j)}, \varphi_1^*)) = 0$. This is solved by the Newton method with an initial solution $\pi/2$.

By using the initial solutions $(\rho_0^{*(i,j)}, \theta_0^{*(i,j)})$, $\rho_1^{*(i,j)} = R_{ij}^*(\varphi_1^*)$ is determined by solving (7) and (8). To determine $R_{ij}^*(\varphi_m^*)$ $(m = 2, 3, \ldots, N_\mathrm{t})$, we set $(\rho_{m-1}^{*(i,j)}, \theta_{m-1}^{*(i,j)})$ as initial solutions, respectively. Here, by solving (7) and (8), $\theta_m^{*(i,j)}$ $(m = 1, 2, \ldots, N_\mathrm{t})$ are incidentally determined with $\rho_m^{*(i,j)}$ $(m = 1, 2, \ldots, N_\mathrm{t})$, respectively. Note that solutions $(\rho_m^{*(i,j)}, \theta_m^{*(i,j)})$ of the nonlinear system, (7) and (8), may not converge to appropriate ranges that are $0 \le \rho_m^{*(i,j)} \le 2R$ and $0 \le \theta_m^{*(i,j)} \le \pi$. In this case, $\rho_\mathrm{temp}^* = R_{ij}^*(\varphi_m^* - \Delta\varphi^*/2)$ and θ_temp^* are determined with initial solutions $(\rho_{m-1}^{*(i,j)}, \theta_{m-1}^{*(i,j)})$. The solutions $(\rho_\mathrm{temp}^*, \theta_\mathrm{temp}^*)$ are used as initial solutions for redetermining $(\rho_m^{*(i,j)}, \theta_m^{*(i,j)})$.

3.3 Algorithm

In the previous subsections, some techniques for speed-up of the X-BNM are presented. Specifically, an efficient algorithm for evaluating G_{ij} and H_{ij} is summarized as follows:

1. After integration points ρ_n $(n = 1, 2, \ldots, N_\mathrm{g})$ in $0 \le \rho \le R$ are determined, $\theta_{m,n}^{(j)} = \theta_j(\rho_n, \varphi_m)$ $(m = 1, 2, \ldots, N_\mathrm{t}, n = 1, 2, \ldots, N_\mathrm{g})$ are determined on S_j $(j = 1, 2, \ldots, N)$ by solving (5). Namely, the following C-like pseudo code is executed $NN_\mathrm{t}N_\mathrm{g}$ times. Note that $\theta_{m,0}^{(j)} = \pi/2$ $(j = 1, 2, \ldots, N, m = 1, 2, \ldots, N_\mathrm{t})$.

```
θ^(j)_{m,n} = Newton1D(ρ_n, θ^(j)_{m,n-1}, φ_m);
Newton1D(ρ_c, θ_ini, φ_c){
  θ is determined by solving (5) with an initial solution
  θ_ini on the assumption ρ = ρ_c and φ = φ_c;
  return θ;
}
```

2. G_{ij} and H_{ij} $(i, j = 1, 2, \ldots, N)$ are evaluated. A C-like pseudo code for evaluation of G_{ij} and H_{ij} is as follows:

```
for(j = 1; j ≤ N; ++j){
  for(i = 1; i ≤ N; ++i){
    if(x_i ∉ S_j){
      G_ij and H_ij are evaluated by (4) with θ^(j)_{m,n};
    }
    else if(x_i ∈ S_j){
      b = e*_x · (x_i − x_j); c = |x_i − x_j|^2 − R^2; D = b^2 − c;
      Δφ* = 2π/N_t; ρ_0^{*(i,j)} = −b + √D;
      θ_0^{*(i,j)} = Newton1D(ρ_0^{*(i,j)}, π/2, φ_1^*);
      for(m = 1; m ≤ N_t; ++m){
        (ρ_m^{*(i,j)}, θ_m^{*(i,j)}) = Newton2D(ρ_{m-1}^{*(i,j)}, θ_{m-1}^{*(i,j)}, φ_m^*, Δφ*);
        R_ij^*(φ_m^*) = ρ_m^{*(i,j)};
        Integration points ρ_{m,n}^{*(i,j)} (n = 1, 2, ..., N_g)
        in 0 ≤ ρ* ≤ R_ij^*(φ_m^*) are determined;
        θ_{m,0}^{*(i,j)} = π/2;
        for(n = 1; n ≤ N_g; ++n){
          θ_{m,n}^{*(i,j)} = Newton1D(ρ_{m,n}^{*(i,j)}, θ_{m,n-1}^{*(i,j)}, φ_m^*);
        }
      }
      G_ij and H_ij are evaluated by (6) with θ_{m,n}^{*(i,j)};
    }
  }
}
Newton2D(ρ*_ini, θ*_ini, φ*_c, Δφ*){
  (ρ*, θ*) are determined by solving (7) and (8) with
  initial solutions (ρ*_ini, θ*_ini) on the assumption φ* = φ*_c;
  if((ρ* < 0 || ρ* > 2R) || (θ* < 0 || θ* > π)){
    φ̂* = φ*_c − Δφ*/2;
    (ρ*, θ*) = Newton2D(ρ*_ini, θ*_ini, φ̂*, Δφ*/2);
  }
  return (ρ*, θ*);
}
```

4. Numerical Experiments

In this section, the performance of the X-BNM is compared with that of the BEM. To this end, both methods are applied to a 3D Laplace problem. A boundary is assumed as $f(\boldsymbol{x}) = x^2/4 + y^2/9 + z^2/16 - 1 = 0$. The boundary condition is chosen so that the analytic solution may be $u = 2\bar{r}^3 P_3^1(\cos\bar{\theta})\cos\bar{\varphi}$. Here, $(\bar{r}, \bar{\theta}, \bar{\varphi})$ is a usual 3D polar coordinates and $P_3^1(x)$ is the associated Legendre function. In addition, Dirichlet and Neumann conditions are assumed on given boundary nodes $\boldsymbol{x}_k$ with $z_k \ge 0$ and those with $z_k < 0$, respectively. For the shape functions $\phi_i(\boldsymbol{x})$ $(i = 1, 2, \ldots, N)$, radii are set so that at least four nodes may be contained inside the supports. Note that all the shape functions have the same radius.

For the BEM and the X-BNM, same nodes are employed. The nodes are uniformly placed on the boundary. For the BEM, the boundary is divided into a set of triangles that consist of these nodes. In addition, the linear elements are adopted for the BEM. For the algorithm described in Sect. 3.3, we set $N_\mathrm{t} = 11$ and $N_\mathrm{g} = 5$. The linear system of the X-BNM and that of the BEM are solved by the GMRES(k) method [4], where k is the number of restarts and we set $k = 300$.

Computations were performed on a computer equipped with dual 2.8 GHz Quad-Core Intel Xeon processors, 24 GB RAM, Mac OS X ver. 10.6 and g++ ver. 4.2.1. Note that we used only a single core of the computer for the computations.

Let us first investigate the dependence of computational time on the number N of nodes. Fig. 3 (a) shows the computational time required for evaluation of all influence coefficients of the X-BNMs and that of the BEM. In this figure, new and conventional X-BNMs denote the X-BNMs developed with and without the algorithm described in Sect. 3.3, respectively. From this figure, we see that the computational time of the new X-BNM is considerably reduced compared with that of the conventional one. However, the computational time of the new X-BNM is larger than that of the BEM. Note that, for a relatively large-sized problem, there is no obvious difference between the computational time of the new X-BNM and that of the BEM. Hence, by using the proposed algorithm, the computational

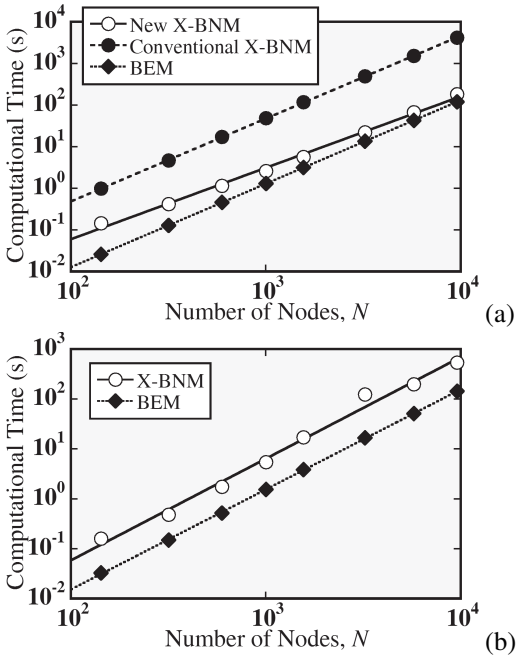

Fig. 3 Dependence of the computational time on the number N of nodes. (a) The computational time required for evaluation of all influence coefficients and (b) The total computational time.

time required for evaluation of all influence coefficients of the X-BNM is close to that of the BEM for the case where N is large. In the following, the new X-BNM is simply called the X-BNM. Fig. 3 (b) shows the total computational time of the X-BNM and that of the BEM. From this figure, we see that the total computational time of the X-BNM is about 4 times as large as that of the BEM. However, Figs. 3 (a) and 3 (b) show that, for the case where N is large, most of the difference between the total computational time of the X-BNM and that of the BEM is caused by a process for solving the linear system. Therefore the difference between the total computational costs may be reduced, if a fast solver for the linear system in the X-BNM is developed.

Next, we investigate accuracy of the X-BNM and that of the BEM. To this end, we define a relative error as $\varepsilon \equiv \|\boldsymbol{x}^{\mathrm{e}}_{uq} - \boldsymbol{x}^{\mathrm{n}}_{uq}\|_2 / \|\boldsymbol{x}^{\mathrm{e}}_{uq}\|_2$, where $\boldsymbol{x}^{\mathrm{e}}_{uq} \equiv [u^{\mathrm{e}}_1, u^{\mathrm{e}}_2, \ldots, u^{\mathrm{e}}_N, q^{\mathrm{e}}_1, q^{\mathrm{e}}_2, \ldots, q^{\mathrm{e}}_N]^{\mathrm{T}}$, and $\boldsymbol{x}^{\mathrm{n}}_{uq} \equiv [u^{\mathrm{n}}_1, u^{\mathrm{n}}_2, \ldots, u^{\mathrm{n}}_N, q^{\mathrm{n}}_1, q^{\mathrm{n}}_2, \ldots, q^{\mathrm{n}}_N]^{\mathrm{T}}$. Here, u^{e}_i and u^{n}_i are exact and numerical solutions of u on $\boldsymbol{x}_i$, respectively. In addition, $q \equiv \partial u/\partial n$, and q^{e}_i and q^{n}_i are defined the same as u^{e}_i and u^{n}_i, respectively. The relative errors of the X-BNM and those of the BEM are determined as a function of the number N of nodes and are depicted in Fig. 4. This figure indicates that, for a relatively large-sized problem, there is no obvious difference between accuracy of the X-BNM and that of the BEM. Therefore, without using any integration cells, the X-BNM can show almost the same accuracy as that of the BEM for the case where N is large.

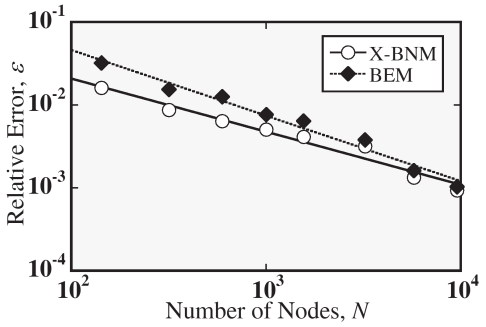

Fig. 4 Relation between the number N of nodes and the relative error ε.

5. Conclusion

For the purpose of speed-up of the X-BNM, we have developed an efficient algorithm for evaluating influence coefficients. The algorithm can be easily implemented into the X-BNM without using any integration cells. On the basis of the algorithm, a numerical code has been developed for solving a 3D Laplace problem. By means of the code, the performance of the X-BNM has been investigated numerically. Conclusions obtained in the present study are summarized as follows:

1) By using the proposed algorithm, computational costs for evaluating influence coefficients are reduced considerably. In particular, the algorithm is efficiently performed for the case where N is large.
2) Although the total computational time of the X-BNM is about 4 times as large as that of the BEM, most of the difference between the total computational time of the X-BNM and that of the BEM is caused by a process for solving the linear system for the case where N is large.
3) Without using any integration cells, the X-BNM can show almost the same accuracy as that of the BEM for the case where N is large.

In order to reduce the total computational time for the X-BNM, a fast linear-system solver has to be developed or selected. On the other hand, for the purpose of improving the accuracy of the X-BNM, shape functions of the higher-order MLS approximation should be employed. From the standpoint of practicability of the X-BNM, these problems need to be resolved in near future.

Acknowledgment

This work was supported by KAKENHI (No. 20700098 and No. 22360042) and by the NIFS Collaboration Research Program (NIFS09KDBN003).

[1] M.K. Chati and S. Mukherjee, Int. J. Numer. Methods Eng. **47**, 1523 (2000).
[2] T. Itoh, A. Saitoh, A. Kamitani and H. Nakamura, Plasma Fusion Res. **5**, S2111 (2010).
[3] J. Bloomenthal *et al.*, *Introduction to Implicit Surfaces* (Morgan Kaufmann Publishers, Inc., San Francisco, 1997).
[4] Y. Saad and M.H. Schultz, SIAM J. Sci. Stat. Comput. **7**, 856 (1986).