Parallel computation in combination of finite and boundary elements

N. Kamiya\textsuperscript{a}, H. Iwase\textsuperscript{a} and E. Kita\textsuperscript{b}
\textsuperscript{a}School of Informatics and Sciences,
\textsuperscript{b}Graduate School of Engineering
Nagoya University, Nagoya 464-01, Japan

Abstract

An algorithm for the parallel computing of the boundary-element and finite-element combination method is presented in this paper. By introducing domain decomposition of an entire domain into the boundary-element and finite-element subdomains, each analysis is performed independently and in parallel. Renewal iterative scheme for the parallel computing is the Schwarz method which was adopted to the domain decomposition parallel scheme in the boundary-element analysis. A cluster parallel computing system by workstations connected by LAN is constructed and employed aiming at efficient analysis. Convergence and accuracy of solutions on internal virtual boundaries are shown through some numerical examples.

Introduction

The boundary-element and finite-element combination method is well-known as an effective analysis tool, which makes the most use of their individual merits. Conventional scheme employs an entire unified equation for the whole domain by combining the discretized equations for the boundary-element and finite-element subdomains. The entire equation is constructed by two different concepts; the equivalent boundary-element and the equivalent finite-element \cite{1-4}. Although the latter is conceived more convenient than the former, especially for use with finite-element existing package program, their shortcoming is that the algorithm for constructing the entire equation is highly complicated when compared with that for each single domain.

In order to overcome the stated inconvenience and to increase computational efficiency, we consider here the parallel computing for the boundary-element
and finite-element combination method. Zdenek et al. [5] treated a similar problem by the energy-type formulation for the both subdomains. Their formulation for the boundary-element subdomain is based on the so-called Galerkin principle, which is far from the popular collocation-type formulation. Alternatively, in what follows, we employ the displacement-type formulation for the finite-element subdomain and the collocation-type formulation for the boundary-element subdomain as conventional. Parallel computation for each subdomain and successive renewal of the variables on the interface of the both subdomains are performed to reach final convergence. Since several renewal schemes for the domain decomposition finite-element or boundary-element computation are known, we shall employ them here. Two-dimensional problems governed by the Laplace equation are adopted for verification of the schemes. Parallel computation is carried out on a few workstations connected by LAN (Local Area Network) i.e., under *cluster computing* circumstance.

**Conventional Combination Methods**

We briefly review the conventional boundary-element and finite-element combination methods for the most basic two-subdomains decomposition. The governing differential equation for the entire domain is the following Laplace equation:

\[ \nabla^2 u = 0 \quad ( \text{in } \Omega ) \tag{1} \]

where \( \nabla^2 \) denotes the two-dimensional Laplace operator. The boundary conditions on the real boundary are

\[ u = \bar{u} \quad ( \text{on } C_1 ) \tag{2} \]

\[ q \left( \equiv \frac{\partial u}{\partial n} \right) = \bar{q} \quad ( \text{on } C_2 ) \tag{3} \]

where \( u, q \) are the potential and flux, respectively, \( n \) the outward unit normal on the boundary, and the overscored values describe the specified ones on the boundary.

The entire domain (Fig. 1(a)) is decomposed into two subdomains; one is discretized by the boundary-elements, \( \Omega_1 \), and the other is discretized by the finite-elements, \( \Omega_2 \) (Fig. 1(b)). For the subdomain \( \Omega_1 \), the Laplace equation (1) is transformed into the corresponding integral equation, which is discretized by the boundary-elements, to yield the following system of equations for selected boundary collocation points:

\[ H u = G q \tag{4} \]

where \( u, q \) are the row vectors composed of boundary values of \( u, q \), respectively, and \( H, G \) their coefficient matrices, obtained by integrating product of the fundamental solution and interpolation function over the boundary.
For the subdomain $\Omega_2$, the displacement-type finite-element formulation reduces the following equation:

$$\mathbf{Ku} = \mathbf{p}$$  \hspace{1cm} (5)

where $\mathbf{K}$, $\mathbf{p}$ and $\mathbf{u}$ are the coefficient matrix (rigidity matrix), the row vector in terms of the equivalent nodal flux and the unknown potential vector, respectively.

On the interface $\Gamma_{12}$ (virtual boundary inside the entire domain) between $\Omega_1$ and $\Omega_2$, the following equilibrium and compatibility conditions are satisfied:

$$q_1 = -q_2 \equiv q^f, \quad u_1 = u_2 \equiv u^f$$  \hspace{1cm} (6)

where the suffixes 1, 2 and $f$ designate the respective values corresponding to the subdomains $\Omega_1$ and $\Omega_2$, and the interface $\Gamma_{12}$.

When the original domain is decomposed into the finite-element and boundary-element subdomains, Eqs. (4) and (5) are solved simultaneously with the boundary conditions (2), (3) and (6). For the purpose, the following two schemes are well-populated:

The Equivalent Finite-Elements: The boundary-element discretized subdomain is regarded as a finite-element and Eq. (4) is transformed to the equation of the finite-element-type. By left-multiplying Eq. (4) by the inverse of $\mathbf{G}$, we obtain

$$\mathbf{G}^{-1}\mathbf{Hu} = \mathbf{q}$$  \hspace{1cm} (7)

Further left-multiplying the transformation matrix $\mathbf{M}$ to change the dimension of $\mathbf{q}$ to that of $\mathbf{p}$, we get

$$\mathbf{MG}^{-1}\mathbf{Hu} = \mathbf{Mq}$$  \hspace{1cm} (8)

Now, we introduce new variables
226 High Performance Computing

\[ K' = MG^{-1}H, \quad p' = Mq \]  \hspace{1cm} (9)

and then Eq. (8) becomes

\[ K'u = p' \]  \hspace{1cm} (10)

which is the finite-element-type equation.

Using \( K \) in Eq. (5) and \( K' \) in Eq. (10), we can construct a so-called rigidity matrix for the entire domain for the finite-element analysis. It should be mentioned that, in this combination method, the inverse of \( G \) is required and \( K' \) is a fully-populated and asymmetric matrix, which reduces the well-known and outstanding advantages of the finite-element coefficient matrix; sparseness and symmetry.

**The Equivalent Boundary-Elements:** Equation (4) for the boundary-element subdomain is divided into those for the real boundary and the interface (fictitious boundary) as

\[ \begin{bmatrix} G'_f & G' \end{bmatrix} \begin{bmatrix} q'_f \\ q_i \end{bmatrix} = \begin{bmatrix} H'_f & H_i \end{bmatrix} \begin{bmatrix} u'_f \\ u_i \end{bmatrix} \]  \hspace{1cm} (11)

Similarly, Eq. (5) for the finite-element subdomain is divided as

\[ \begin{bmatrix} K_2 & K'_f \end{bmatrix} \begin{bmatrix} u'_2 \\ u'_f \end{bmatrix} = \begin{bmatrix} M_2 & M'_f \end{bmatrix} \begin{bmatrix} q'_2 \\ q'_f \end{bmatrix} \]  \hspace{1cm} (12)

These two equations are combined, by employing the equilibrium and compatibility conditions, Eq. (6), on the interface to reach

\[ \begin{bmatrix} H_i & H'_f & -G'_f & 0 \\ 0 & K'_f & M'_f & K_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u'_f \\ u'_2 \\ q'_f \end{bmatrix} = \begin{bmatrix} G_1 & 0 \\ 0 & M_2 \end{bmatrix} \begin{bmatrix} q_1 \\ q'_2 \end{bmatrix} \]  \hspace{1cm} (13)

Equation (13) is to be solved in terms of unknown potential and flux with help of Eqs. (2) and (3) on the real boundary. In this combination scheme, the resulting equation (13) differs for each distinct selection of the interface between the boundary-element and finite-element subdomains and distinct numbers of subdomains. Therefore, we must construct Eq. (13) for each problem, which is generally inconvenient for use of the scheme as blackbox.

As mentioned above, the conventional finite-element and boundary-element combination methods encounter inevitable inconvenience in efficient practical application.
Combination Method Using Parallel Processing

The parallel computing algorithm considered here is basically identical to that employed for sole finite-element or boundary-element domain decomposition method. Computations for the finite-element and boundary-element subdomains are performed independently and are available in parallel, to fit the values on the interface using successive renewal iterations. As the renewal schemes for the interface variables, several are known such as the Uzawa scheme [6], the Schwarz Neumann-Neumann scheme (referred to Method (1) in what follows) and the Schwarz Dirichlet-Neumann scheme (Method (2)) [7, 8]. In this study we shall employ two schemes due to Schwarz, which were applied previously to the boundary-element parallel computing. For the definition of the fictitious boundary, we adopt the non-overlapping boundary because of the characteristics of the sole boundary-element discretization in the boundary element subdomain.

Equilibrium and Compatibility Conditions on the Interface: We suppose that, on the interface, the boundary-element is taken to coincide with a side of the finite-element. In what follows, the boundary-element is the simplest element, constant element and the finite-element is a linear triangular element, for which the middle point on the boundary-element is the collocation point and represents constant magnitudes of the potential and flux on the element, while the flux is constant and the potential varies linearly on the finite-element. Consequently, the equilibrium condition on the interface is able to be formulated easily. Needless to say, higher order boundary- and finite-elements are also available without any essential modification. Denoting the flux on the boundary-element $m$ as $q^m_{BEM}$ and that on the finite-element $e$ as $q^e_{FEM}$, the equilibrium condition is written as

$$q^m_{BEM} = -q^e_{FEM} \quad (14)$$

Similarly, the compatibility condition is (Fig. 2)

$$u^m_{BEM} = u^e_{FEM} \left( \frac{u^i_{FEM} + u^j_{FEM}}{2} \right) \quad (15)$$

![Figure 2 Relation of potential between FEM and BEM on virtual internal boundary.](image)
where $u_{\text{FEM}}^e$ is defined as a mean potential value on the nodes $i$ and $j$ of the finite-element $e$.

Renewal of the Boundary Conditions on the Interface: Renewal of the interface variables is carried out using the equilibrium and compatibility conditions. In Method 1, the flux values (Neumann data) are assumed in advance both on the interface of the finite-element and boundary-element subdomains,

$$q_{\text{BEM}} = \bar{q}_{\text{BEM}}, \quad q_{\text{FEM}} = \bar{q}_{\text{FEM}}$$  \hspace{1cm} (16)

and the potential values are determined. For the renewal iteration the following equations are employed:

$$q_{\text{BEM}}^{s+1} = q_{\text{BEM}}^s + \beta (u_{\text{FEM}}^s - u_{\text{BEM}}^s)$$  \hspace{1cm} (17)

$$q_{\text{FEM}}^{s+1} = -q_{\text{BEM}}^s$$  \hspace{1cm} (18)

where superfix $s$ denotes the iteration number. Along with convergence of the potential on the interface, the flux thereon converges. On the other hand, in Method 2, the assumed value on the boundary of the boundary-element subdomain is the potential (Dirichlet data), while that of the finite-element domain is the flux (Neumann data),

$$u_{\text{BEM}} = \bar{u}_{\text{BEM}}, \quad q_{\text{FEM}} = \bar{q}_{\text{FEM}}$$  \hspace{1cm} (19)

and the unknown values are determined. The renewal schemes are

$$u_{\text{BEM}}^{s+1} = u_{\text{BEM}}^s + \gamma (u_{\text{FEM}}^s - u_{\text{BEM}}^s)$$  \hspace{1cm} (20)

$$q_{\text{FEM}}^{s+1} = -q_{\text{BEM}}^s$$  \hspace{1cm} (21)

Figure 3 Combination of BEM and FEM in rectangular domain.
The constants $\beta$ and $\gamma$ appearing in Eqs. (17) and (29) are specified empirically with some trial and error.

**Examination of the Combination Algorithm for the Parallel Computing**

Some simple examples are considered for examination and comparison of the boundary-element and finite-element combination parallel algorithms. The first example (Ex. 1) is shown in Fig. 3. A rectangular entire domain is divided into two identical square subdomains by the fictitious interface $\Gamma_{12}$; one subdomain $\Omega_1$ is discretized by the boundary-elements and the other $\Omega_2$ by the finite-elements. The boundary of $\Omega_1$ is discretized by 30 equal-length constant boundary-elements and the subdomain $\Omega_2$ is discretized by 70 linear finite-elements of equal size. The constants $\beta$ and $\gamma$ for the renewal are determined empirically as 0.03 and 0.3 for Methods 1 and 2, respectively. Initial guess of the values on the fictitious boundary is taken as 0. The convergency on selected few points is shown in Figs. 3 and 4, correspondingly to Methods 1 and 2.
Good convergence is found for every point but tendency of convergence is different for Methods 1 and 2.

![Graph showing convergence](image)

Figure 5 Convergence of solutions in Method (2).

**Computational Efficiency of Parallel Algorithm**

Computational efficiency of the cluster parallel computing is examined in comparison with the serial computing with one workstation. For the parallel processing, three workstations (Fujitsu S-4/EC) construct the cluster system; one is the host machine which controls the whole computation process and works for the renewal of the boundary conditions on the interface; the other two are submachines which compute the finite-element and boundary-element computations, respectively.

The required computation time and its efficiency is affected by the numbers of boundary-elements and finite-elements. Comparison is performed between Example 1 and the case of finer meshes (Ex. 4, 60 boundary-elements and 200 finite-elements, Fig. 6). Method 2 is employed for the renewal iteration and the resulting computational efficiency is estimated by the following equation:
\[ \eta(\%) = \left(1 - \frac{\text{Parallel computing time by three workstations}}{\text{Computing time by one workstation}}\right) \times 100 \]  

(22)

For 30 renewal iterations, the detail of required computer times for some essential operations are shown in Fig. 6 for both Exs. 1 and 4. As can be seen in this figure, efficiency in Ex. 4 increased by 23 % while not in Ex. 1. When the size of equations for boundary-element and finite-element analyses is small, the efficiency is decreased by additional operations such as communication time, waiting time for synchronization between workstations. For more elements in the boundary-element and finite-element discretizations, the computer time for computation becomes larger, and therefore the additional time for the cluster system is negligible compared with computation time. This fact suggests that for large-scale computation, the efficiency of the proposed scheme will be improved undoubtedly. The mentioned tendency in the efficiency was already indicated in the sole finite-element or boundary-element domain decomposition parallel computing. When more subdomains are employed, the communication between workstations plays a key role affecting the total efficiency of the system.

Conclusions

The previously developed parallel computing scheme with domain decomposition was applied to the boundary-element and finite-element combination analysis. Some sample examples in the potential analysis indicated good convergence of solution. Using this algorithm, the difficulties in the conventional combination analysis scheme were removed pronouncedly, and the unified algorithm became available, which will exploit further possibility to more complicated and large-scale problems. Practical and further applications are under study.

References


5. Zdenek, D., and Josef, M., Symmetric FE-BE coupling with iteration on
232 High Performance Computing


---

**Figure 6** Times for serial and parallel computing