

Simulation of Seismic Wave Propagation and Amplification

Muneo Hori, Tsuyoshi Ichimura, and Kohei Fujita

Abstract Large scale simulation of ground motion is a vital tool in engineering seismology and earthquake engineering. Need for high performance computing cannot be underestimated, because increasing the spatial resolution results in computing higher frequency components of ground motion which influence structure response. This chapter explains improvement of discretization for large scale simulation of ground motion simulation based on finite element method. Better mathematical treatment is needed for ground motion simulation since it solves a four-dimensional linear or non-linear wave equation so that mass matrix becomes diagonal. Diagonalization of the mass matrix is achieved by utilizing a set of orthogonal and discontinuous basis functions. While it sounds odd, the use of discontinuous basis functions provides us a larger capability of modeling. As examples of large scale simulation of ground motion, the use of K computer, the best supercomputer in Japan in the year of 2015, is explained. A non-linear finite element method is developed in K computer, so that a model of 10,000,000,000 degree-of-freedom is analyzed with 100,000 time steps in less than a half day. Numerical computation techniques and parallel computation enhancement are explained to realize this simulation, which leads to high scalability of the developed finite element method. As an illustrative example, Tokyo Metropolis is used as a target, and a K computer simulation of ground motion is presented.

M. Hori (✉) • T. Ichimura
Earthquake Research Institute, The University of Tokyo, Tokyo, Japan
e-mail: hori@eri.u-tokyo.ac.jp; ichimura@eri.u-tokyo.ac.jp

K. Fujita
Advanced Institute for Computational Science, RIKEN,
7-1-26 Minatoshima Minamimachi, Chuo-ku, Hyogo 650-0047, Japan
e-mail: kohei.fujita@riken.jp

1 Improvement of Discretization

Earthquake ground motion is strongly influenced by the three-dimensional (3D) underground structure. Therefore, it is a standard practice to employ 3D wave propagation simulation that uses a model of the underground structure. Improving the accuracy of the simulation is a key challenge in seismology research, and various studies have attempted to verify 3D elasto-dynamic codes to improve and confirm the accuracies (e.g., [1]). Because of the simplicity and flexibility, the two most popular techniques for ground motion simulation are the finite difference method (FDM) (e.g., [2–5]) and the finite element method (FEM) (e.g., [6–11]); see also [12] for a brief history of these simulations. For a large and highly heterogeneous target structure, the structured-grid discretization of FDM is attractive. On the other hand, advancements in computing have led to greater acceptance of FEM, even though its computational cost is large; it is capable to analyze a complex underground model more readily than FDM. Moreover, the variable mesh of FEM can be tailored to local wavelengths and treat irregular free surface conditions accurately.

Dynamic explicit FEM is used in engineering fields, and it is often applied to earthquake ground motion simulation owing to its efficient numerical performance (e.g., [6–9]). Explicit FEM that is used in seismology research simply diagonalizes a consistent mass matrix and derives a diagonal lumped mass matrix. This treatment produces non-negligible errors; Belytschko et al. [13] noted “procedures for diagonalizing the mass matrix are quite ad hoc, and there is little theory for these procedures.” For the sake of clarity, we explain the approximations made in determining an element matrix. In a conventional displacement FEM, the element mass matrix is determined using $\rho (b^n b^m)_e$, where ρ is the density, b^n is a basis function for the displacement at node n , and $(\cdot)_e$ implies the integration of (\cdot) over an element e , i.e.,

$$(\cdot)_e = \int_e (\cdot) dv.$$

Unless $(b^n b^m)_e$ satisfies $(b^n b^m)_e = 0$ for $n \neq m$, the global element mass matrix will not be diagonal. Some methods approximate the global element matrix by *lumping* it, or convert it into a diagonal matrix (e.g., [13–15]). This approximation is important in reducing the computational cost of applying an explicit time integration scheme. Several researchers have presented detailed discussions of the differences in numerical performance between lumped and consistent mass matrices (e.g., [16]). Because of its efficiency, the lumping approximation is taken for granted in earthquake motion simulation for actual 3D underground structure. However, the accuracy is lost, at least to a certain degree, if the lumping approximation is adapted.

Special finite elements which use orthogonal continuous polynomial basis functions have been proposed. For such elements, an explicit method can be naturally derived with the aid of orthogonality; for example, see [17–19] for recent works in

seismology research and fluid dynamics. Komatitsch and Vilotte [17] and Dumbser and Kaser [18] demonstrated the usefulness of consistent lumped-mass matrices of higher-order elements, using a set of higher-degree polynomials. Although higher-order elements are used to compute these responses with high degree of accuracy, there is also a need for lower-order elements to compute the responses of highly heterogeneous bodies. Thus, Matsumoto and Takada [19] proposed a 4-node tetrahedron with 1 bubble node, using a set of complex lower-degree polynomials. The excellent numerical performance of this element demonstrated the usefulness of the lower-order elements with consistent lumped mass matrices. Given this usefulness, standard linear elements (4-node tetrahedral and 8-node hexahedral elements) should also be developed. We propose such standard linear elements for computing earthquake ground motion in a highly heterogeneous body.

In this chapter, we present standard linear elements using *orthogonal discontinuous* basis functions, from which the consistent and diagonalized mass matrices are naturally derived. The numerical characteristics of the mass matrix are categorized as extensions of simple standard finite elements. In Sect. 1.1, we present a formulation of a consistent and diagonalized mass matrix that does not require approximation. We present the detailed configurations for 4-node tetrahedral and 8-node hexahedral elements. The relationship between the proposed finite elements and conventional standard finite elements is investigated. In Sect. 1.2, we compare the solutions obtained from a conventional explicit FEM with analytical solutions in layered media to verify the numerical dispersion caused by the lumping approximation. Comparison of the solutions obtained by using the proposed finite elements with the analytical solutions demonstrates the usefulness of the technique. Examples are also presented to illustrate the effectiveness of the proposed method in earthquake ground motion modeling in the actual 3D crust structure.

1.1 Methodology

The major difference between the present new formulation and the conventional one is the use of orthogonal discontinuous basis functions in discretizing a displacement function. Because it is difficult to satisfy both orthogonality and smoothness, the basis functions we employ are orthogonal but discontinuous. FEM that is obtained by the present new formulation is considered an extension of a mixed FEM [14] with orthogonal basis functions and an FEM with a particle discretization scheme [20].

We start from Hellinger–Reissner’s functional, i.e.,

$$J(\mathbf{u}, \boldsymbol{\sigma}) = \int_V \int_T -\frac{1}{2} \rho \dot{\mathbf{u}} \cdot \dot{\mathbf{u}} - \frac{1}{2} \boldsymbol{\sigma} : \mathbf{d} : \boldsymbol{\sigma} + \boldsymbol{\sigma} : (\nabla \otimes \mathbf{u}) - \mathbf{u} \cdot \mathbf{f} \, dv dT. \quad (1)$$

Here, \mathbf{u} and $\boldsymbol{\sigma}$ are displacement and stress, ρ and \mathbf{d} are the density and the compliance tensor (the inverse of the elasticity tensor \mathbf{c}), \mathbf{f} is a body force, and

$\dot{(\cdot)}$ and $\nabla(\cdot)$ are temporal and spatial differential operators for (\cdot) . Various finite elements have been proposed using this functional (e.g., a *mixed* method in [14]).

Next, we discretize both the displacement and stress functions in J . To this end, we consider an element e , in which ϕ^α and ψ^β are employed as basis functions for \mathbf{u} and $\boldsymbol{\sigma}$, i.e.,

$$\mathbf{u} = \sum_{\alpha} \mathbf{u}^{\alpha} \phi^{\alpha} \quad \text{and} \quad \boldsymbol{\sigma} = \sum_{\beta} \mathbf{s}^{\beta} \psi^{\beta}.$$

Here, \mathbf{u}^{α} and \mathbf{s}^{β} are unknown coefficients of discretization. Substituting these equations into Eq. (1) and using stationary conditions in place of \mathbf{u}^{α} and \mathbf{s}^{β} , we derive the following matrix equation for \mathbf{u}^{α} and \mathbf{s}^{β} :

$$\mathbf{K}\mathbf{u} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{F}, \quad (2)$$

where \mathbf{u} is a vector obtained by assembling \mathbf{u}^{α} , and \mathbf{K} and \mathbf{M} are the assembly of the element stiffness and mass matrices, \mathbf{K}_e and \mathbf{M}_e , respectively. These are computed as

$$\mathbf{K}_e = (\psi^{\beta} \nabla \phi^{\alpha})_e^T \cdot (\psi^{\beta} \psi^{\beta'})_e^{-1} \mathbf{c} \cdot (\psi^{\beta'} \nabla \phi^{\alpha'})_e$$

and

$$\mathbf{M}_e = \rho(\phi^{\beta} \phi^{\beta'})_e.$$

The superscripts in \mathbf{K}_e denote the transpose (T) and inverse (-1). Finally, \mathbf{F} is a vector for the coefficients of \mathbf{f} when $\{\phi^{\alpha}\}$ are used for the discretization.

While Eq. (2) holds for any choice of the basis functions, we restrict our attention to the special case that the set of $\{\phi^{\alpha}\}$ is orthogonal and the $\{\psi^{\beta}\}$ is not orthogonal but smooth. We use the Heaviside step function, H , to form the orthogonal basis functions. As will be shown later, even though the basis functions made from H are not continuous, we can form a suitable element stiffness matrix with an element mass matrix that is diagonal.

We next study the cases of a 4-node tetrahedron and an 8-node hexahedron element, denoted by TET4D and HEX8D, respectively. The configuration of TET4D and HEX8D is displayed in Fig. 1, where (x, y, z) is the global coordinate and (r_1, r_2, r_3) is a local coordinate. The numbers in *italics* indicate the node. For simplicity, we denote by TET4 and HEX8 an ordinary 4-node tetrahedron element and an 8-node hexahedron element, respectively. TET4 has a linear basis function of displacement, while HEX8 uses tri-linear functions.

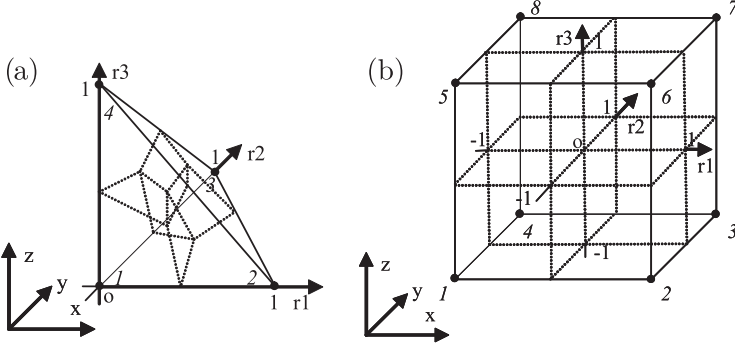


Fig. 1 Configuration of elements. *Dotted lines* indicate interfaces between Voronoi blocks. (a) 4-node tetrahedra element, (b) 8-node hexahedra element

Table 1 Table of $\bar{p}l_1$, $\bar{p}l_2$ and $\bar{p}l_3$ for 4-node tetrahedra element, where $pl_1 = r_1 + r_2 + 2r_3 - 1$, $pl_2 = 2r_1 + r_2 + r_3 - 1$, $pl_3 = r_1 + 2r_2 + r_3 - 1$, $pl_4 = r_1 - r_3$, $pl_5 = r_1 - r_2$, and $pl_6 = r_2 - r_3$

α	$\bar{p}l_1$	$\bar{p}l_2$	$\bar{p}l_3$
1	$-pl_1$	$-pl_2$	$-pl_3$
2	pl_2	pl_4	pl_5
3	pl_3	$-pl_5$	pl_6
4	pl_1	$-pl_4$	$-pl_6$

1.1.1 4-Node Tetrahedron Element

The basis functions of TET4D are as follows: four piece-wise constant functions in each Voronoi block are used for $\{\phi^\alpha\}$ and one function for $\{\psi^\beta\}$, i.e.,

$$\begin{cases} \phi^\alpha = H(\bar{p}l_1)H(\bar{p}l_2)H(\bar{p}l_3) & (\alpha = 1, 2, 3, 4), \\ \psi^1 = 1, \end{cases}$$

where H is the Heaviside function and $\bar{p}l_i$ is defined in Table 1. A Voronoi block is defined as the region of space closer to its node than to any other node. Thus, $\{\phi^\alpha\}$ is orthogonal, but discontinuous.

For TET4D, \mathbf{K}_e is identical to that of TET4. However, \mathbf{M}_e becomes diagonal, and the diagonal terms are $\rho/4(1)_e$. Thus, \mathbf{M}_e is identical to the lumped element mass matrix that is obtained by approximation for TET4.

1.1.2 8-Node Hexahedron Element

The basis functions of HEX8D are chosen as follows: eight piece-wise constant functions in each Voronoi block are used for $\{\phi^\alpha\}$, and eight tri-linear functions for $\{\psi^\beta\}$, i.e.,

$$\begin{cases} \phi^\alpha = H(\bar{r}_1 r_1) H(\bar{r}_2 r_2) H(\bar{r}_3 r_3) \quad (\alpha = 1, 2, \dots, 8), \\ \psi^1 = 1, \quad \psi^2 = r_1, \quad \psi^3 = r_2, \quad \psi^4 = r_3, \\ \psi^5 = r_1 r_2, \quad \psi^6 = r_1 r_3, \quad \psi^7 = r_2 r_3, \end{cases} \quad (3)$$

see Table 2 for \bar{r}_i . By definition, the \mathbf{M}_e of HEX8D becomes diagonal, with terms $\rho/8 (1)_e$. This \mathbf{M}_e is identical to the lumped element mass matrix that is used for HEX8D. Unlike TET4D, however, the \mathbf{K}_e of HEX8D is not identical to that of HEX8. The cost of computing \mathbf{K}_e for HEX8D, which involves the inversion of $(\psi^\beta \psi^{\beta'})_e$, is also higher than that of computing \mathbf{K}_e for HEX8. This disadvantage can be overcome through the use of structured elements, such as voxels or hybrids of voxels and unstructured tetrahedrons; see, e.g., [7, 8, 10, 11].

Let us now clarify the numerical performance of HEX8D, which is somewhat better than that of HEX8, as will be shown later. If we use $\psi = 1$ like TET4, the resulting stiffness matrix is identical to HEX8 with one point integration. Consequently, it is essential to use functions with order greater than 1 for ψ^β to avoid hourglass modes. In this context, we use Eq. (3) to avoid hourglass modes and derive appropriate hourglass control terms. \mathbf{K}_e of HEX8D for a voxel the size of which is ds can be decomposed as

$$\mathbf{K}_e = \sum_{i=1}^7 \gamma_i (\mathbf{h}_i)^T \mathbf{c} \mathbf{h}_i,$$

see Table 3 for γ_i and \mathbf{h}_i . $(\gamma_1 (\mathbf{h}_1)^T \mathbf{c} \mathbf{h}_1)$ of \mathbf{K}_e arises from the constant term in ψ^β and is identical to HEX8 with one point integration. The other terms, $\sum_{i=2}^4 \gamma_i (\mathbf{h}_i)^T \mathbf{c} \mathbf{h}_i$ and $\sum_{i=5}^7 \gamma_i (\mathbf{h}_i)^T \mathbf{c} \mathbf{h}_i$, are considered to be stabilization terms against hourglass mode I in Fig. 2a and mode II in Fig. 2b. On the basis of this

Table 2 Table of \bar{r}_1 , \bar{r}_2 , and \bar{r}_3 for 8-node hexahedra element

α or β	\bar{r}_1	\bar{r}_2	\bar{r}_3
1	-1	-1	-1
2	1	-1	-1
3	1	1	-1
4	-1	1	-1
5	-1	-1	1
6	1	-1	1
7	1	1	1
8	-1	1	1

Table 3 Coefficients and vectors for element stiffness matrix of voxel element whose size is ds

$$\begin{cases} \gamma_1 = \frac{1}{ds}, \\ \gamma_2 = \gamma_3 = \gamma_4 = \frac{3}{64ds}, \\ \gamma_5 = \gamma_6 = \gamma_7 = \frac{9}{256ds}, \end{cases} \quad (\mathbf{h}_1)^T = \begin{pmatrix} -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 \\ 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 \\ 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 \end{pmatrix},$$

(continued)

Table 3 (continued)

[illegible]

(continued)

Table 3 (continued)

$(\mathbf{h}_5)^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix},$	$(\mathbf{h}_6)^T = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix},$	$(\mathbf{h}_7)^T = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$
---	---	---

clarification, HEX8D is categorized as a sort of HEX8 with one point integration and hourglass control along conventional lines (e.g., [21, 22]). Note that these hourglass control terms can be rigorously derived in our formulation.

1.1.3 Discontinuous Basis Function

The discretization scheme presented in the above may seem odd, because a discretized function is neither continuous nor differentiable and does not belong to the ordinary H1 Sobolev space for the problem. However, a solution obtained in terms of $\{\phi^\alpha\}$ is the *closest* to a solution in H1; the closeness is in the following sense: for a given function $f(\mathbf{x})$, the discretization scheme determines the coefficients by minimizing

$$E(f^1, f^2, \dots) = \int \left(f - \sum_{\alpha} f^{\alpha} \phi^{\alpha} \right)^2 dx.$$

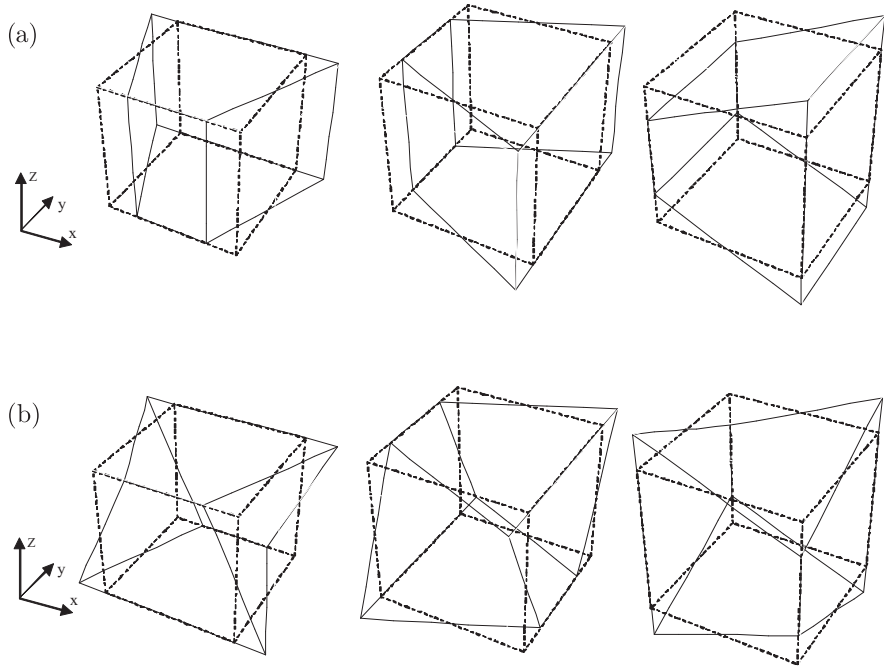


Fig. 2 Configuration of hourglass modes. *Solid and dotted line* indicates voxel and hourglass mode, respectively. **(a)** Hourglass modes I, **(b)** hourglass modes II

A key issue is that a function closest to the solution (which is continuous and differentiable) can be found in a space of continuous and differentiable functions.

Raised is a question regarding how close a function obtained by the present method (i.e., the numerical solution) is to the exact solution. It has not yet been proved in general that the above E vanishes at the limit as the number of ϕ^α increases infinitely. However, it has been rigorously demonstrated that the numerical solution coincides with a solution of an ordinary FEM with linear elements. In the numerical experiments presented here, it is shown that the numerical solution is quantitatively close to the exact solution and the discontinuity inherent to the numerical solution does not provide fatal error.

1.2 Numerical Experiment

We perform simulation of earthquake ground motion which is excited by a single point source in horizontally layered media and in an actual 3D underground structure, in order to test the numerical performance of the proposed elements. The

conventional FEM and the proposed orthogonal FEM are designated by CFEM and OFEM, respectively, and the configurations of the two FEM's are summarized as follows:

- CFEM: unstructured TET4 and structured HEX8 (cubic). A simple lump approximation is applied to the element mass matrix.
- OFEM: unstructured TET4D and structured HEX8D (cubic).

Although the difference between CFEM and OFEM is the element stiffness matrix of the cubic element, we expect that the numerical dispersion of OFEM is less than that of CFEM because the stiffness matrix of HEX8D is consistent with a lumped mass matrix. Additionally, because the element stiffness matrices of TET4D are identical to those of TET4 and the stiffness matrices of structured HEX8D are stored in memory, the computational costs of OFEM are identical to those of the CFEM. We should point out that all numerical techniques applied to CFEM (e.g., parallel computation) are applicable to OFEM.

The target matrix equation is

$$\mathbf{K}\mathbf{u} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{F},$$

where \mathbf{C} is Rayleigh's damping, the assembly of $\alpha\mathbf{M}_e + \beta\mathbf{K}_e$ with α and β being determined as [8]. Unstructured 4-node tetra and structured 8-node cubic elements are automatically generated for 3D models using [11], as long as more than 10 elements are used for one wavelength [6–8, 11]. We use the same mesh model in both CFEM and OFEM. The point source is implemented as an equivalent body force. To remove the outgoing waves, absorbing boundary conditions are applied to all surfaces except the top [23]. A two-step Adams-Bashforth method is applied for the time integration, and frequency components outside of the target frequency domain are filtered out from the solution.

1.2.1 Horizontal Layered Problem

Table 4 summarizes the problem setup, and Figs. 3 and 4 present a 3D model and a close-up view of the generated mesh. Observation points, denoted by obs_i ($i = 1, 2, \sim 14$), are set on the surface at coordinates $(x, y) = (-37800 + 1800i, -37800 + 1800i)$.

Figure 5 compares a CFEM solution and an analytic solution [24] obtained by computing Green's function (GF). There is close agreement between the two solutions, although small differences are seen at observation points located far from the source. To estimate the total difference, we used the quantitative misfit criteria, em and pm , which indicate the misfit in terms of the envelope and phase, respectively [25]. Table 5 presents misfit criteria em and pm between the two solutions. As can be seen, they increased as the waves propagated.

Figure 6 shows wave profiles of an OFEM solution and the analytic solution. As can be seen, the agreement is also close in this case. Table 6 presents misfit criteria

Table 4 Horizontally layered half space problem setting

Time duration	40.96 s
Target frequency	≤ 1.0 Hz
Domain size	$-86.4 \text{ km} \leq x \leq 86.4 \text{ km}$ $-86.4 \text{ km} \leq y \leq 86.4 \text{ km}$ $-72 \text{ km} \leq z \leq 0 \text{ km}$
Depth of layer	3.6 km
Layer	$\rho = 2500 \text{ kg/m}^3$ $V_p = 3900 \text{ m/s}$ $V_s = 2250 \text{ m/s}$ $Q = 500$
Half space	$\rho = 3000 \text{ kg/m}^3$ $V_p = 7800 \text{ m/s}$ $V_s = 4500 \text{ m/s}$ $Q = 500$
Excitation	$M_0 (2 t^2/T_0^2) \quad 0 \leq t \leq T_0/2$ $M_0 (1-2 (t-T_0)^2/T_0^2) \quad T_0/2 \leq t \leq T_0$ $M_0 \quad T_0 \leq t$
Strike, dip, rake	$30^\circ, 40^\circ, 0^\circ$
Magnitude (M_0)	$1.0 \times 10^{18} \text{ Nm}$
Rise time (T_0)	2 s
Source location	$(-35887.5 \text{ m}, -35887.5 \text{ m}, -1687.5 \text{ m})$

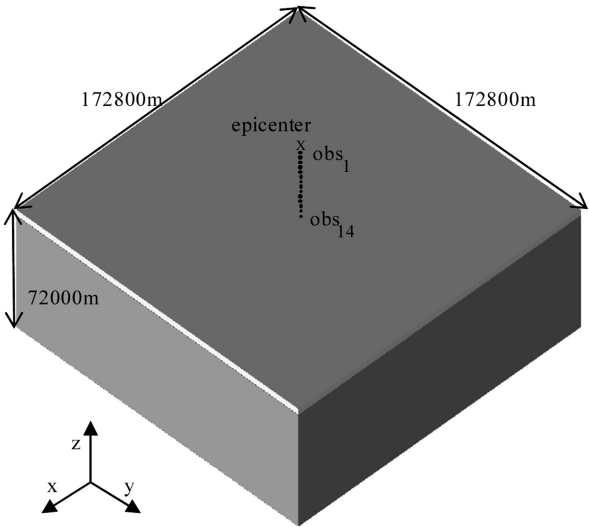


Fig. 3 3D model of horizontally layered half space problem

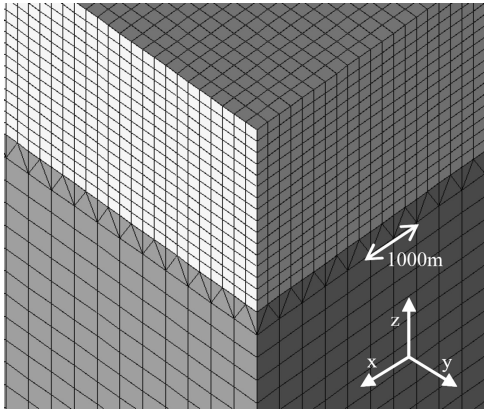


Fig. 4 Mesh configuration of horizontally layered half space problem. Number of nodes, tetrahedron elements, and hexahedron elements are 33,617,834, 12,681,216, and 30,965,760

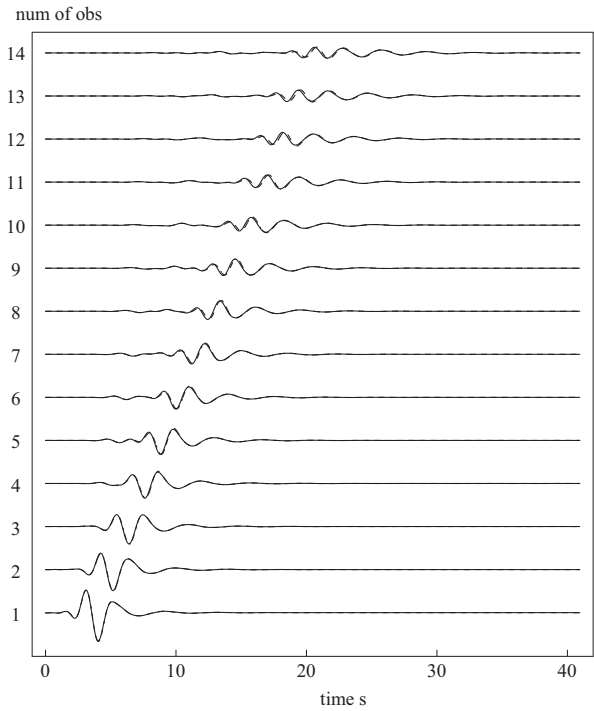


Fig. 5 Comparison of wave velocity profiles in z direction at each observation point, where *dotted* and *solid line* indicates CFEM and GF solutions

between the OFEM solution and the GF solution. It is seen that agreement of OFEM and GF is closer, compared with that of CFEM and GF. The misfits barely increased as the waves propagated.

Table 5 Misfits % of CFEM and GF solutions in each direction at obs_i for horizontally layered half space problem

obs_i	em			pm		
i	x	y	z	x	y	z
1	1.34	1.45	1.71	0.71	0.56	0.95
2	2.62	1.07	2.37	1.51	0.45	1.71
3	2.75	4.21	3.05	2.11	1.64	2.46
4	2.79	5.45	4.29	2.05	3.84	3.62
5	3.31	5.58	5.12	2.26	5.21	4.74
6	4.69	6.66	5.66	3.32	4.07	5.16
7	4.68	7.03	6.17	3.30	5.87	5.82
8	3.82	8.12	7.56	2.23	7.65	7.36
9	6.58	8.57	8.21	4.16	8.36	8.32
10	8.12	7.51	8.96	7.65	7.24	8.66
11	7.41	7.72	10.20	8.18	7.36	10.61
12	10.93	8.98	13.62	8.93	8.48	13.65
13	10.20	9.27	17.77	11.54	7.64	15.92
14	10.60	7.68	37.47	12.30	4.15	16.69

Target frequency domain is [0.05 Hz, 1.0 Hz]

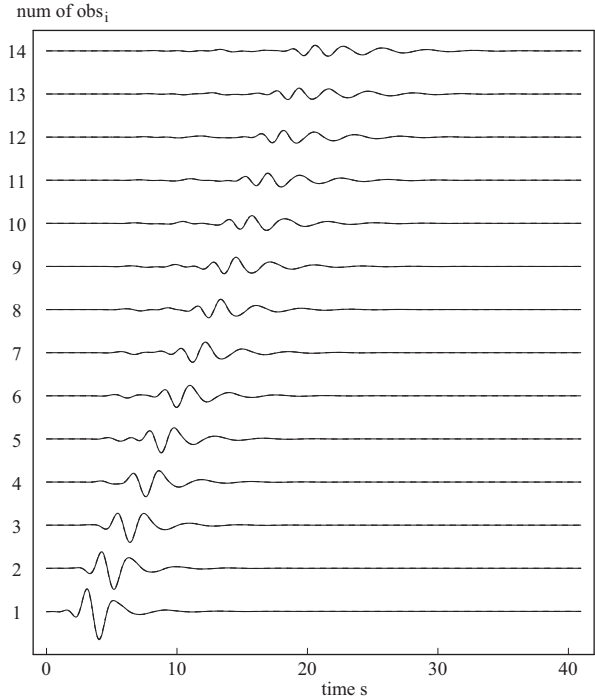


Fig. 6 Comparison of wave velocity profiles in z direction at each observation point, where *dotted* and *solid line* indicates OFEM and GF solutions

Table 6 Misfits % of OFEM and GF solutions in each direction at obs_i for horizontally layered half space problem

obs_i	em			pm		
i	x	y	z	x	y	z
1	1.94	1.03	2.54	0.67	1.24	1.25
2	1.43	2.63	2.22	0.76	1.34	1.45
3	1.21	2.94	1.92	1.31	1.46	1.50
4	1.49	2.40	2.29	1.48	1.03	1.79
5	2.01	2.61	2.13	1.64	1.43	1.71
6	2.09	4.27	1.68	1.61	1.19	2.00
7	1.42	3.24	1.85	1.73	0.95	2.16
8	3.17	2.35	2.12	1.86	1.44	2.31
9	3.17	2.76	1.58	1.77	2.10	2.41
10	2.44	4.01	1.68	1.64	2.35	2.73
11	3.71	3.20	1.83	1.87	2.07	3.03
12	4.91	2.49	2.48	1.66	2.25	3.25
13	4.78	2.65	2.03	1.95	2.58	3.50
14	3.68	3.51	3.65	2.18	2.52	4.34

Target frequency domain is [0.05 Hz, 1.0 Hz]

Figure 7 shows wave profiles at obs_1 and obs_{14} . At obs_1 , which is located near the source, the difference among CFEM, OFEM, and GF is small, although some errors are caused by the implementation of the point source as an equivalent body force. At obs_{14} , which is located far from the source, the difference between OFEM and GF is small. However, the difference between CFEM and GF is large. There is surely no significant difference in the first part of the waveform. However, the difference grows in the latter part and the waveforms break. This is because the numerical dispersion of CFEM increases as the waves propagate. These results demonstrate the usefulness of OFEM as well as the effect of approximation made for the diagonalization of a mass matrix in CFEM on the numerical accuracy.

1.2.2 Actual 3D Underground Structure

Next, we simulate earthquake ground motion from a single point source in an actual 3D underground structure. This analysis is conducted using both CFEM and OFEM to examine whether any differences in accuracy arise. Table 7 summarizes the problem setup, and Figs. 8 and 9 show a 3D model and a close-up view of the generated mesh. The 3D model is constructed based on [26]. Observation points obs_i ($i = 1, 2, \sim 13$) are set on the top surface at coordinates $(x, y) = (2000i, 2000i)$.

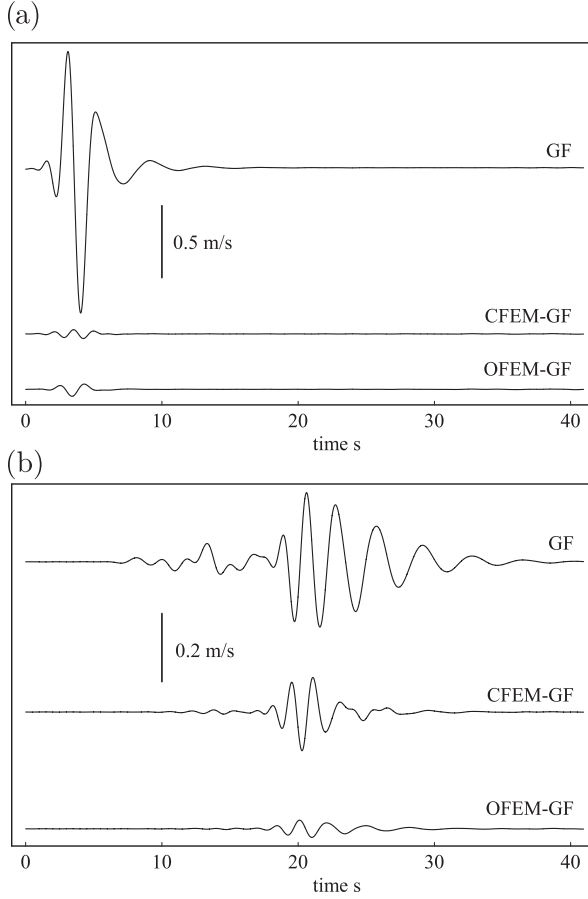


Fig. 7 Comparison of wave velocity profiles. **(a)** Wave profiles in z direction at obs_1 , **(b)** wave profiles in z direction at obs_{14}

The misfit criteria (em and pm) between a CFEM solution and an OFEM solution at each observation point are summarized in Table 8. As the distance of the observation point from the source point increases, the difference tends to become larger. Because the wave path of the present underground structure is more complicated than in the horizontal layered media, the distribution of misfit criteria is not simple. However, it is seen that the difference increases as the waves propagated. Additionally, because the wave path is not simple, the difference increased locally at points close to the source. Moreover, the EM and PM maximum values are both about 10. Because this value is in the range of the numerical verification problem, significant differences in the accuracy occur in the earthquake ground motion simulation in an actual 3D crust. Wave profiles at obs_1 and obs_{13} are shown in Fig. 10. Although there is no large difference in

Table 7 Actual 3D crust structure problem setting

Time duration	40.96 s
Target frequency	≤ 0.5 Hz
Domain size	$-28.8 \text{ km} \leq x \leq 28.8 \text{ km}$
	$-28.8 \text{ km} \leq y \leq 28.8 \text{ km}$
	$-30 \text{ km} \leq z \leq 0 \text{ km}$
First layer	$\rho = 2000 \text{ kg/m}^3$
	$V_p = 2000 \text{ m/s}$
	$V_s = 1000 \text{ m/s}$
	$Q = 500$
Second layer	$\rho = 2000 \text{ kg/m}^3$
	$V_p = 2500 \text{ m/s}$
	$V_s = 1200 \text{ m/s}$
	$Q = 600$
Third layer	$\rho = 2300 \text{ kg/m}^3$
	$V_p = 3900 \text{ m/s}$
	$V_s = 2000 \text{ m/s}$
	$Q = 1000$
Fourth layer	$\rho = 2500 \text{ kg/m}^3$
	$V_p = 4500 \text{ m/s}$
	$V_s = 2500 \text{ m/s}$
	$Q = 1250$
Fifth layer	$\rho = 2600 \text{ kg/m}^3$
	$V_p = 5000 \text{ m/s}$
	$V_s = 3000 \text{ m/s}$
	$Q = 1500$
Excitation	$M_0 (2 t^2/T_0^2) \quad 0 \leq t \leq T_0/2$
	$M_0 (1-2 (t-T_0)^2/T_0^2) \quad T_0/2 \leq t \leq T_0$
	$M_0 \quad T_0 \leq t$
Strike, dip, rake	$30^\circ, 40^\circ, 0^\circ$
Magnitude (M_0)	$1.0 \times 10^{18} \text{ Nm}$
Rise time (T_0)	2 s
Source location	(200 m, 200 m, -5000 m)

the first part of the wave profile and at the observation point located near the source, the difference increases due to numerical dispersion as the wave propagates. The accuracy of the analysis result differs by approximately 20 % of the velocity amplitude.

These results support the usefulness of OFEM in earthquake ground motion simulations. Recent research [27] has clarified the effects of surface geometry on seismic motion. Because FEM readily analyzes a body of complex structures, it is often used for numerical simulations of long-period earthquake motion. However, as demonstrated here, if the observation points are far from the source point in terms of

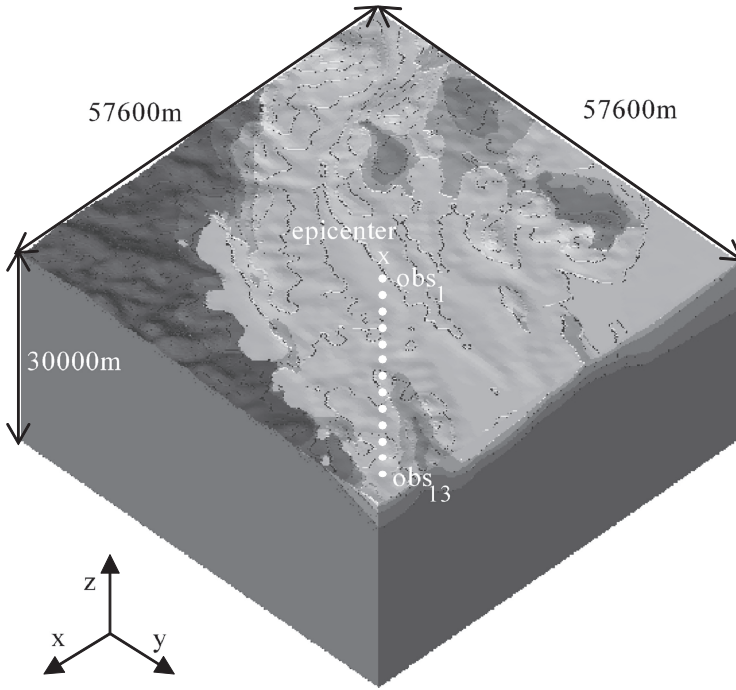


Fig. 8 3D model of actual crust structure problem

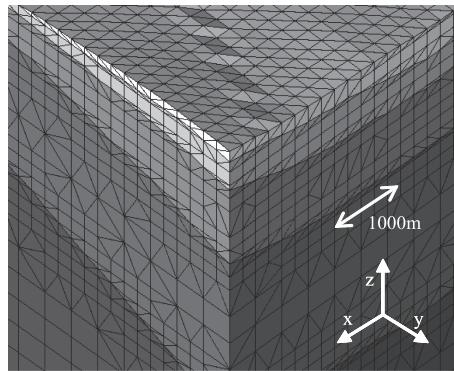


Fig. 9 Mesh configuration of actual crust structure problem. Number of nodes, tetrahedron elements, and hexahedron elements are 3,122,675, 4,643,138, and 2,257,905

wavelength, large errors in the envelope and phase may occur. Numerical simulation of OFEM is more accurate compared with SFEM, and it is expected that OFEM will contribute to estimate the long period earthquake ground motion.

Table 8 Misfits % of CFEM and OFEM solutions in each direction at obs_i for actual 3D crust structure problem

obs_i	em			pm		
i	x	y	z	x	y	z
1	2.32	5.37	3.95	1.43	2.89	2.22
2	3.42	4.94	3.69	2.27	3.28	2.65
3	7.32	3.98	4.25	3.73	3.26	3.53
4	6.57	3.67	7.80	5.34	3.95	4.49
5	3.71	4.61	6.73	4.25	4.80	7.47
6	4.62	4.83	8.99	4.39	4.96	6.93
7	7.26	9.23	6.93	6.87	6.68	7.10
8	8.41	9.61	8.50	6.99	8.39	10.21
9	6.01	7.91	6.96	6.18	8.02	6.62
10	7.17	9.83	10.84	6.30	9.14	8.74
11	8.04	7.70	7.48	9.90	8.20	8.86
12	7.97	6.85	7.94	9.44	6.67	12.04
13	11.13	7.53	6.54	9.83	6.78	6.64

Target frequency domain is [0.05 Hz, 0.5 Hz]

1.3 Summary

This chapter develops an FEM with a diagonalized consistent mass matrix, using discontinuous orthogonal basis functions. The presented FEM formulation reduces numerical dispersion by generating a lumped element mass matrix without making any approximation. Furthermore, we present a detailed formulation of 4-node tetrahedra and 8-node hexahedra and verify their numerical performance for wave modeling through experiments on earthquake ground motion. The results of the numerical experiments reveal large differences between the conventional FEM solution and the analytical solution at points far from the point source because of the effect of approximation on diagonalizing the mass matrix. Moreover, the numerical experiments indicate that higher accuracy is obtained from the proposed FEM than from the conventional FEM. Earthquake ground motion simulations in an actual 3D underground structure are conducted as an example of the proposed method. The results show significant differences in accuracy between the conventional and proposed FEM's. Because recent research has shown that earthquake ground motion is significantly affected by surface geometry, explicit FEM that can handle complex surface geometry is of greater importance. Thus, the proposed FEM is expected to contribute to the field of earthquake ground motion simulation.

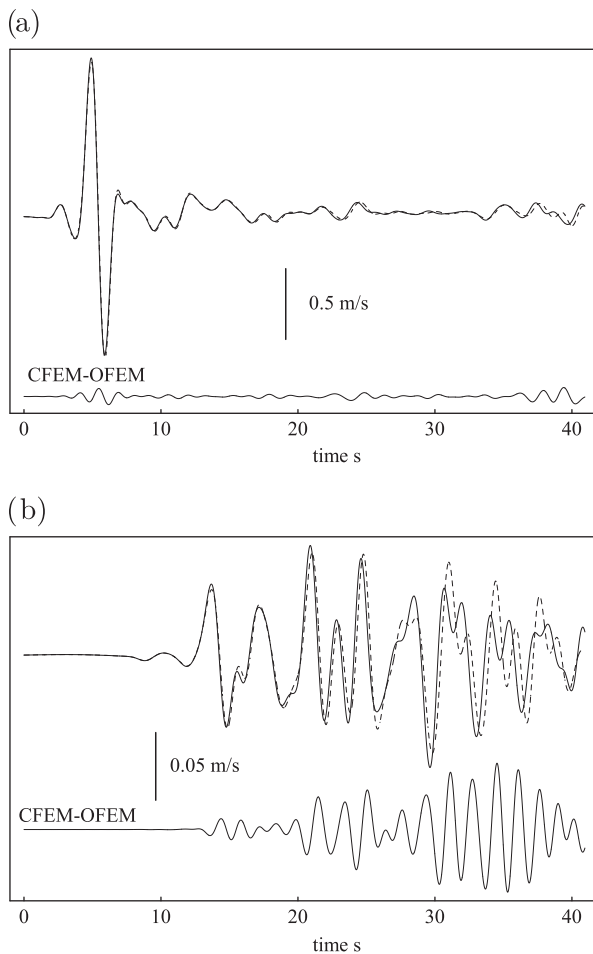


Fig. 10 Comparison of wave velocity profiles where *solid* and *dotted line* indicates OFEM and CFEM solutions. CFEM-OFEM is the difference between CFEM and OFEM solutions. **(a)** wave profiles in x direction at obs₁, **(b)** wave profiles in x direction at obs₁₃

2 Ground Motion Simulation

A large-scale earthquake is a menace to modern civilization, since it has a potential of literally destroying a densely populated urban area. A recent harrowing example is 2011 Great East Japan Earthquake, which has produced long-lasting damage to the eastern part of Japan. An earthquake of a similar scale is likely to hit another mega cities, such as San Francisco, Seattle, Istanbul, or Tokyo, in the near future. The first step in mitigating and reducing earthquake disasters is a reliable estimate of possible earthquake disaster. High spatial resolution is essential for such an estimate

because the disaster is accumulation of damage to each building or structure in the area while the earthquake itself is a crustal-scale phenomenon. An urban area simulation of earthquake hazard and disaster is a unique solution to realizing such a reliable estimate.

An earthquake and the disaster it induces consist of the following three processes:

1. seismic waves generated by the fault propagate in the crust;
2. the seismic waves are amplified near soft ground-surface layers;
3. buildings and structures are shaken by the resulting ground motion;

see Fig. 11. The *physics* of these processes is quite simple. This is because the processes are described completely as solutions of linear or nonlinear solid wave equations. However, the physics has largely been ignored because solving the wave equation for all the elements of the processes requires a tremendous amount of computation. Empirical equations such as attenuation relations and vulnerability curves are usually used for earthquake hazard and disaster estimations [28–30].

Despite the simple physics of the three processes, constructing a model for analysis is not an easy task. Modern observation technologies in Earth Sciences are addressing this problem. However, seismic wave propagation simulation that solves the wave equation has become a hot topic since an accurate model is constructed for a fault and a shallow crust. This simulation requires a large amount of computation, and some achievements using high performance computing techniques have been presented even at the supercomputing conferences (SCs); for example, see [31–34].

The structural seismic response simulation of man-made structures, which is directly related to estimate of an earthquake disaster, is also being realized [35, 36]; an analysis model is automatically constructed for every building and structure using urban area digital information (which is stored in a form of Geographical Information System). Urban area seismic response is then computed as an assembly of structural seismic responses of all the buildings and structures. A good example is the Tokyo simulation [37], in which nonlinear responses of 253,405 buildings are

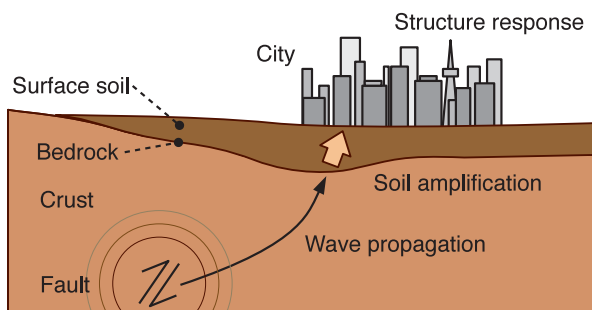


Fig. 11 Schematic view of earthquake disaster process

computed for 100 earthquake scenarios using 16,000 CPU cores of the K computer for 4088 s.

A missing item that links the seismic wave propagation simulation and the urban area seismic response simulation is the seismic wave amplification simulation; high impedance contrast between shallow soft soil and deep hard rock makes the degree of the amplification be larger than 10 times for ground motion components of 1–10 Hz. Seismic wave amplification processes are complicated due to the nonlinear material properties of the surface soil as well as the topographical effects of the surface and ground layer configuration. This results in a wide range of degree of the amplification which changes from place to place. The amplification process could lead to a condition that is likely fatal for structures [38]. The computation cost required for solving a nonlinear wave equation in a domain of irregular geometry is much larger than that of the seismic wave propagation simulation. This is because (1) nonlinearity of soil reduces stiffness, decreasing the wavelength (if stiffness becomes 25 %, the wavelength is reduced to 50 %) and (2) deformation is concentrated at irregular parts of the surface and layers (hills or valleys). These two issues contribute to the need to increase the spatial resolution.

Realization of the seismic wave amplification simulation is a challenge even from the viewpoint of computer science. A target domain is $2000 \times 2000 \times 100$ m, and the spatial resolution needed is of the order of 0.1 m. An earthquake duration of 30 s and a temporal resolution of 0.001 s are required. The resulting degrees of freedom (DOF) are on the order of 10 BlnDOF (or 10×10^9 DOF), and the resulting time step is 30 K (or 30×10^3). Moreover, an unstructured element model must be constructed to account for the irregular geometry of the domain. A most efficient FEM must be developed for this simulation, and a sophisticated model construction method is needed.

The seismic wave amplification simulation is another hot topic for Earth Science, or more specifically, engineering seismology. One trial that used a voxel FEM [39, 40] employed the octree-base domain decomposition with a minimum element size of 5 m, but it was not able to accurately model the irregular geometry and implemented only relatively simple nonlinear soil constitutive models. Another trial [41] used an unstructured grid model, but the spatial resolution was limited; the minimum element size was 4 m, and the related temporal resolution was insufficient for the seismic response simulation of man-made structures.

In this chapter, we present the most updated trial aimed at realizing the seismic wave amplification simulation that numerically solves the nonlinear wave equation. In Sect. 2.1, we explain the development of an FEM code that is able to analyze a nonlinear model of 10 BlnDOF for a 30 K time step and the development of an unstructured element model for the irregularly configured domain. In Sect. 2.2, we show the computation performance of the developed code on the K computer [42]. We also examine the performance on an Intel CPU machine to assess the portability of the code developed. In Sect. 2.3, we show an example of the seismic wave amplification simulation. A 10.7 BlnDOF model of Tokyo is constructed and the seismic wave amplification process is computed for a 30 K time step. The results of the simulation are then inputted to the urban area seismic response simulation that

is an assembly of the structural seismic response simulations of 13,275 buildings. Concluding remarks are provided in Sect. 2.4.

2.1 Key Ideas and Innovations

2.1.1 Target Equation

Underground structure is modeled as a layered medium, each layer of which often has complicated geometries because the thickness changes on the order of $10^{-1} \sim 1$ m in a $10^0 \sim 1$ m region. FEM with unstructured low-order solid elements is suitable for analyzing the response of such a medium together with satisfying traction-free boundary condition on the surface. Here, we use a nonlinear dynamic FEM with three-dimensional (3D) unstructured low-order solid elements for the seismic wave amplification simulation. The equations to be solved are as follows:

$$\left(\frac{4}{dt^2} \mathbf{M} + \frac{2}{dt} \mathbf{C}^n + \mathbf{K}^n \right) \delta \mathbf{u}^n = \mathbf{F}^n - \mathbf{Q}^{n-1} + \mathbf{C}^n \mathbf{v}^{n-1} + \mathbf{M} \left(\mathbf{a}^{n-1} + \frac{4}{dt} \mathbf{v}^{n-1} \right), \quad (4)$$

with

$$\begin{aligned} \mathbf{Q}^n &= \mathbf{Q}^{n-1} + \mathbf{K}^n \delta \mathbf{u}^n, \\ \mathbf{u}^n &= \mathbf{u}^{n-1} + \delta \mathbf{u}^n, \\ \mathbf{v}^n &= -\mathbf{v}^{n-1} + \frac{2}{dt} \delta \mathbf{u}^n, \\ \mathbf{a}^n &= -\mathbf{a}^{n-1} - \frac{4}{dt} \mathbf{v}^{n-1} + \frac{4}{dt^2} \delta \mathbf{u}^n, \end{aligned} \quad (5)$$

where $\delta \mathbf{u}$, \mathbf{u} , \mathbf{v} , \mathbf{a} , and \mathbf{F} are vectors describing incremental displacement, displacement, velocity, acceleration, and external force vectors, respectively, \mathbf{M} , \mathbf{C} , and \mathbf{K} are the consistent mass matrix, the damping matrix, and the stiffness matrix, respectively, dt is the time increment, and n is the time step. We use Rayleigh damping, where the element damping matrix \mathbf{C}_e^n is calculated using the consistent element mass matrix \mathbf{M}_e and element stiffness matrix \mathbf{K}_e^n as follows:

$$\mathbf{C}_e^n = \alpha \mathbf{M}_e + \beta \mathbf{K}_e^n.$$

The coefficients α and β were determined by solving the least-squares equation, i.e.,

$$\text{minimize} \left[\int_{f_{\min}}^{f_{\max}} \left(h^n - \frac{1}{2} \left(\frac{\alpha}{2\pi f} + 2\pi f \beta \right) \right)^2 df \right],$$

where f_{\max} and f_{\min} are the maximum and minimum values of the target frequencies, and h^n is the damping ratio at time step n . The damping matrix is calculated at each time step because the stiffness and damping ratio change with time.

Small elements are generated locally when modeling complex geometry with solid elements. Satisfying the Courant condition in using an explicit time-integration method leads to small time increments, which results in considerable computational expense. To resolve this, we used the Newmark- β method for time integration (with $\beta = 1/4$, $\delta = 1/2$). Because complicated nonlinear constitutive relations are often used in analyzing nonlinear ground motion and there are cases that the implicit scheme does not reach an accurate solution, an explicit scheme is selected, in order to carry out the time integration robustly. Semi-infinite absorbing boundary conditions are used for the bottom and side boundaries of the simulation domain.

The modified Ramberg–Osgood model [43] and Masing rule [44] are used for the nonlinear constitutive relations of the surface soil; while more sophisticated constitutive relations are proposed, these models are standard for dynamic analysis of soil.

The unknowns in Eq. (4) are $\delta \mathbf{u}^n$. We solve for $\delta \mathbf{u}^n$ at each time step to obtain the history of \mathbf{u} . The calculation proceeds as follows:

- Read boundary conditions;
- Compute the stiffness and damping ratio for the n -th time step using the Ramberg–Osgood model and Masing rule with the strain obtained at the $n - 1$ -th time step;
- Compute \mathbf{K}^n and \mathbf{C}^n using the stiffness and damping ratio for the n -th time step;
- Compute $\delta \mathbf{u}^n$ by solving Eq. (4), and update the values in Eq. (5) using $\delta \mathbf{u}^n$;
- Output results;

Because most of the computation cost in this procedure is due to solving Eq. (4), the key point for achieving efficient analysis is the development of a suitable solver, considering the characteristics of the nonlinear dynamic FEM. A method for constructing such an FEM code with a large number of unstructured low-order elements was also developed to model the complex ground structure accurately.

2.1.2 Key Ideas and Innovations in the Solver

In the seismic wave amplification simulation, we must compute $\delta \mathbf{u}^n$ with a 30 K order time step using a high-resolution 10 Bln-order-DOF FEM model (FEMmodel) that can ensure the convergence of \mathbf{u}^n in terms of acceleration response and the convergence of structural response in an urban area. In short, this problem results in solving a 10 Bln-order-DOF matrix equation with a 30 K-order time step, where the components are changed at every time step.

In view of limitations on computer resources, a fast solver is very desirable. For example, we must solve this problem with a 30 K time step in 12 h on the K computer. This requires that only 1.44 s be used to solve the response for one time step. Because a large-DOF problem must be solved with relatively small computer memory, we develop a fast iterative solver, considering the characteristics of the matrix equation. There has been much research on fast iterative solvers which are tuned for FEM with unstructured low-order elements; there are some researches

which have been reported even at SCs (e.g., [45]). These researchers seek to reduce the computation time using a preconditioner with a high computation cost, in order to improve the poor convergence of the target problem. Others have reported similar research (e.g., [46, 47]). For example, Ogino et al. [46] presents a fast solver based on a balancing domain decomposition method [48]-type preconditioner, which could solve a 140 M-DOF problem in 577 s (3.88 TFLOPS, 24.26 % of peak) with 3.05 TB memory and 2048 CPU on the Earth Simulator [49].

Although a sophisticated preconditioner with smart program tuning can reduce the computation cost and resolve the poor convergence of the target problem, our problem needs a much faster solver. Although the nonlinear dynamic FEM has difficulties in solving a matrix equation repeatedly where the components are changing at every time step, there is a clue in that the characteristics of the matrix are not so bad because of the effect of inertia terms. However, the convergence of the current matrix equation is worse than the convergence of a matrix equation which comes from an “ordinary” problem with lower resolution, even though the same DOF is used for the problem. Thus, we need a smart preconditioner with high scalability under a many-compute-nodes environment and with lower computation cost for a fast iterative solver. Additionally, less use of computer memory is required for the preconditioner because each compute node manages a large number of DOF. As a result, it is concluded that a moderate preconditioner with less computation cost and high scalability in a many-compute-nodes environment is more suitable than a smart preconditioner with higher computation cost and only moderate scalability.

One candidate for such a fast iterative solver is the “preconditioned conjugate gradient” (PCG) method [50] with the block Jacobi method. However, the number of iterations tends to increase when solving matrices with poor characteristics. Indeed, we examined this in our numerical experiments, and the number of iterations often increased markedly. The slow convergence rate of PCG should not be overlooked in developing a faster iterative solver for our problem. In this viewpoint, it is a wise choice to employ a moderate preconditioner with lower computation cost, which is explained in the above. We must pay attention to the amount of computation in seeking to achieve fast computation, as well as to reducing the frequency of communication and the amount of synchronization necessary between compute nodes. Considering the characteristics of a nonlinear dynamic FEM, we develop an FEM code with the following four key ideas:

Predictor

The initial solution of the iterative solver is predicted as $\frac{dt}{24}(-9\mathbf{v}^{n-4} + 37\mathbf{v}^{n-3} - 59\mathbf{v}^{n-2} + 55\mathbf{v}^{n-1})$ by the Adams–Bashforth method [51]. Using this method with a small time increment leads to a good initial solution with a relative error of $10^{-4\sim-3}$, which reduces the number of CG iterations.

Adaptive Conjugate Gradient Method [52]

A preconditioner in a conventional conjugate-gradient method uses the inverse of a matrix, the characteristics of which are similar to the matrix of the target problem. In the adaptive conjugate-gradient method, the matrix equation is solved roughly using a conjugate-gradient iteration instead of the inverse matrix. The conventional gradient iteration and the gradient iteration for a preconditioner are called the inner and outer loops, respectively.

Multigrid (MG) Method [50]

The MG method constructs a coarse FEM model from the original model; for simplicity, the coarse model is denoted by M_c , and the original one by M . M_c is equivalent to M but the resolution of M_c is coarser. In the inner loop, a rough solution is first computed for M_c . Using this solution as an initial solution, a rough solution is computed for M with fewer iterations. Although additional computation of M_c is required, the computation cost and the communication cost become much smaller than those needed for M only, because the DOF and the nodes on the interfaces of M_c are much smaller than those of M . As a result, the computation cost of the inner loop is reduced by the MG method.

Mixed-Precision (MP) Arithmetic

Although double-precision arithmetic is needed in the outer loop, single-precision arithmetic is used for the inner loops since only estimation of low precision is needed for the preconditioner. With the DOF of M_c being much smaller than that of M and with halving the required memory size by using single precision, we can store the matrix of M_c in the CRS format. Also, a reduction in communication size is expected; the size of adjacent node communication for M with single precision is reduced to half that with double precision, and the size of adjacent node communication for M_c with single precision is reduced to 1/8 of that for M with double precision. Thus, faster computation and memory access are also expected.

Element-by-Element Method [53]

It is difficult to store \mathbf{K} of M for solving Eq. (4) because the DOF computed by one compute node is large. Instead of storing \mathbf{K} , computing $\mathbf{K}\mathbf{x}$ is made according to the element-by-element (EBE) method; EBE is applied to compute the second-order tetrahedral elements of M . With the advantage of displacement-stress mixed FEM [54] and the orthogonal-basis function, we successfully transformed $\mathbf{K}\mathbf{x}$ into

$$\sum_i \mathbf{N}_{b_i}^T (\mathbf{D}(\mathbf{N}_{b_i} \mathbf{x})),$$

where \mathbf{D} and \mathbf{N}_{bi} are the elastic tensor and equivalent mode, respectively. Sophisticated reformulation and careful tuning achieve a further reduction in the number of operations and lead to higher peak performance ratios (e.g., 21.68 % for EBE with single precision) on the K computer. Because the frequency of EBE with single precision is very high, this speed-up can reduce computation time significantly.

Hereafter, we call this code “GAMERA”¹ (the seismic wave amplification simulator, enhanced by a multiGrid method, Adaptive conjugate gradient method, Mixed precision arithmetic, EBE method, and pRedictor by Adams–Bashforth method). Although the computation cost of one outer loop is almost the same as that for one loop of PCG, the number of outer iterations is reduced markedly by the sophisticated preconditioner. Moreover, the computation cost of the preconditioner itself is also reduced. As a result, the total computation time of GAMERA can be reduced.

In actual computations, we decompose \mathbf{M} into a set of \mathbf{M}^i for the i -th MPI process. Then, we compute \mathbf{u}^n with each pair of \mathbf{M}^i and \mathbf{M}_c^i using Algorithms 1 and 2; \mathbf{M}^i and \mathbf{M}_c^i are regarded as s submodel of \mathbf{M} and \mathbf{M}_c , respectively, as will be shown in the next subsection. Adjacent node communication (MPI_Isend, MPI_Irecv, MPI_Waitall) after the matrix vector product and collective communication (MPI_Allreduce) after the inner product are conducted for synchronization. Because \mathbf{M}^i and \mathbf{M}_c^i are computed by OpenMP parallelization, the whole process of GAMERA is computed by MPI-OpenMP hybrid parallelization.

2.1.3 Key Ideas and Innovations in Model Construction

We present the construction method of a 10 Bln-order-DOF 3D FEM model for GAMERA using data of 3D layered ground structure. Because we seek to evaluate strain and stress in complex ground structures at high resolution, we use a large number of unstructured, second-order tetrahedral elements. The amount of work that is needed for constructing such a large 3D FEM model is considerably high. Thus, a fully automated meshing algorithm that is capable to generate elements of high quality is desirable. Although it is common to perform manual tuning to avoid elements with poor quality, manual tuning for the present model is unrealistic. For this reason, we combine various techniques to develop an automated model construction. We use the method of [55] as the base, which is developed for a 3D FEM model.

In this model generation, a structured grid is used to break up the simulation domain into a number of cells, and elements are constructed for each cell with small aspect ratios. This leads to robust mesh of high-quality elements in full automation. Because the generation of elements in each cell is performed individually, the problem is suitable for parallelization, leading to a reduction in the time for meshing. We generate linear-tetrahedral elements and cubic elements and then convert the

¹An imaginary Japanese Earth guardian, supported by prayers for peace.

Algorithm 1 Details of GAMERA. Matrix vector product $(\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n)(\cdot)$ is computed by EBE. $\text{diag}[\cdot]$, (\cdot) , and ϵ indicate 3×3 block Jacobi of $[\cdot]$, single-precision variable and tolerance for relative error. $(\cdot)_c$ indicates calculation related to FEMmodel_c, and the other is the related calculation of FEMmodel. $\bar{\mathbf{P}}$ is a mapping matrix, from FEMmodel_cⁱ to FEMmodelⁱ, which is defined by interpolating the displacement in each element of FEMmodel_cⁱ

```

1: read boundary condition of n-th time step
2:  $\mathbf{f} \leftarrow \mathbf{F}^n - \mathbf{Q}^{n-1} + \mathbf{C}^n \mathbf{v}^{n-1} + \mathbf{M}(\mathbf{a}^{n-1} + \frac{4}{dt} \mathbf{v}^{n-1})$ 
3:  $\mathbf{x} \leftarrow \frac{dt}{24}(-9\mathbf{v}^{n-4} + 37\mathbf{v}^{n-3} - 59\mathbf{v}^{n-2} + 55\mathbf{v}^{n-1})$ 
4:  $\bar{\mathbf{B}} \leftarrow \text{diag}[\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n]$ 
5:  $\bar{\mathbf{B}}_c \leftarrow \text{diag}[\frac{4}{dt^2}\mathbf{M}_c + \frac{2}{dt}\mathbf{C}_c^n + \mathbf{K}_c^n]$ 
6:  $\bar{\mathbf{A}}_c \leftarrow \frac{4}{dt^2}\mathbf{M}_c + \frac{2}{dt}\mathbf{C}_c^n + \mathbf{K}_c^n$  (stored on memory with CRS format)
7:  $\mathbf{r} \leftarrow \mathbf{f} - (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n)\mathbf{x}$ 
8:  $\beta \leftarrow 0$ 
9:  $i \leftarrow 1$ 
10: (*outer loop start*)
11: while  $\|\mathbf{r}\|_2 / \|\mathbf{f}\|_2 \geq \epsilon$  do
12:   (*inner loop start*)
13:    $\bar{\mathbf{r}} \leftarrow \mathbf{r}$ 
14:    $\bar{\mathbf{z}} \leftarrow \mathbf{B}^{-1}\mathbf{r}$ 
15:    $\bar{\mathbf{r}}_c \leftarrow \bar{\mathbf{P}}^T \bar{\mathbf{r}}$ 
16:    $\bar{\mathbf{z}}_c \leftarrow \bar{\mathbf{P}} \bar{\mathbf{z}}$ 
17:    $\bar{\mathbf{z}}_c \leftarrow \bar{\mathbf{A}}_c^{-1} \bar{\mathbf{r}}_c$  (*solved on FEMmodelc by Algorithm 2 with  $\epsilon_c^{in}$  and initial solution  $\bar{\mathbf{z}}_c^*$ *)
18:    $\bar{\mathbf{z}} \leftarrow (\frac{4}{dt^2}\bar{\mathbf{M}} + \frac{2}{dt}\bar{\mathbf{C}}^n + \bar{\mathbf{K}}^n)^{-1} \bar{\mathbf{r}}$  (*solved on FEMmodel by Algorithm 2 with  $\epsilon^{in}$  and initial solution  $\bar{\mathbf{z}}^*$ *)
19:    $\mathbf{z} \leftarrow \bar{\mathbf{z}}$ 
20:   (*inner loop end*)
21:   if  $i > 1$  then
22:      $\beta \leftarrow (\mathbf{z}, \mathbf{q}) / \rho$ 
23:   end if
24:    $\mathbf{p} \leftarrow \mathbf{z} + \beta \mathbf{p}$ 
25:    $\mathbf{q} \leftarrow (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n)\mathbf{p}$ 
26:    $\rho \leftarrow (\mathbf{z}, \mathbf{r})$ 
27:    $\alpha \leftarrow \rho / (\mathbf{p}, \mathbf{q})$ 
28:    $\mathbf{q} \leftarrow -\alpha \mathbf{q}$ 
29:    $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{q}$ 
30:    $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{p}$ 
31:    $i \leftarrow i + 1$ 
32: end while
33: (*outer loop end*)
34:  $\delta \mathbf{u}^n \leftarrow \mathbf{x}$ 
35:  $\mathbf{Q}^n \leftarrow \mathbf{Q}^{n-1} + \mathbf{K}^n \delta \mathbf{u}^n$ 
36:  $\mathbf{u}^n \leftarrow \mathbf{u}^{n-1} + \delta \mathbf{u}^n$ 
37:  $\mathbf{v}^n \leftarrow -\mathbf{v}^{n-1} + \frac{2}{dt} \delta \mathbf{u}^n$ 
38:  $\mathbf{a}^n \leftarrow -\mathbf{a}^{n-1} - \frac{4}{dt} \mathbf{v}^{n-1} + \frac{4}{dt^2} \delta \mathbf{u}^n$ 
39: output results of n-th time step
40: modify material properties considering nonlinearity

```

Algorithm 2 PCG with a preconditioner. This is used for roughly solving $\mathbf{z} = (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n)^{-1}\mathbf{r}$ and $\mathbf{z}_c = \mathbf{A}_c^{-1}\mathbf{r}_c$ in Algorithm 1. (or) is used for estimation of $\mathbf{z}_c = \mathbf{A}_c^{-1}\mathbf{r}_c$. \mathbf{B} and ϵ^{in} are a block Jacobi matrix and the tolerance for the relative error. All the calculations are conducted using single-precision variables

```

 $\mathbf{x} \leftarrow \mathbf{z}$ 
 $\mathbf{e} \leftarrow \mathbf{r} - (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C} + \mathbf{K})\mathbf{x}$  (or  $\mathbf{A}_c\mathbf{x}$ )
 $\beta \leftarrow 0$ 
 $i \leftarrow 1$ 
while ( $\|\mathbf{e}\|_2 / \|\mathbf{r}\|_2 \geq \epsilon^{in}$  (or  $\epsilon_c^{in}$ )) do
   $\mathbf{z} \leftarrow \mathbf{B}^{-1}\mathbf{e}$  (or  $\mathbf{B}_c^{-1}\mathbf{e}$ )
   $\rho_a \leftarrow (\mathbf{z}, \mathbf{e})$ 
  if  $i > 1$  then
     $\beta \leftarrow \rho_a / \rho_b$ 
  end if
   $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$ 
   $\mathbf{q} \leftarrow (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C} + \mathbf{K})\mathbf{p}$  (or  $\mathbf{A}_c\mathbf{p}$ )
   $\alpha \leftarrow \rho_a / (\mathbf{p}, \mathbf{q})$ 
   $\rho_b \leftarrow \rho_a$ 
   $\mathbf{e} \leftarrow \mathbf{e} - \alpha\mathbf{q}$ 
   $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$ 
   $i \leftarrow i + 1$ 
end while
 $\mathbf{z} \leftarrow \mathbf{x}$ 

```

cubic elements to linear-tetrahedral elements. Next, the linear-tetrahedral elements are converted into second-order tetrahedral elements. We use the model with linear-tetrahedral elements as the \mathbf{M}_c and the model with second-order tetrahedral elements as \mathbf{M} . We divide \mathbf{M} and \mathbf{M}_c , so that the number of elements in each domain becomes uniform, using METIS 5.1.0 [56] with suitable options on a shared-memory computer; recall that the i -th submodels are denoted by \mathbf{M}^i and \mathbf{M}_c^i , respectively. Although there has been research related to the efficient generation of \mathbf{M}^i from \mathbf{M} on distributed memory computers, we use this generation method because load imbalance for \mathbf{M}^i and \mathbf{M}_c^i can be reduced significantly, even for a complicated ground structure model. Such reduced load imbalance contributes to high scalability with Algorithms 1 and 2, and the advantage in reducing the analysis time with high scalability is much greater than the disadvantage of increased model generation time.

FEM models for GAMERA are generated on a virtual shared memory machine: 20 computers (Dell R720xd, each with dual Intel Xeon E5-2670 CPUs and DDR3 384 GB memory) are connected with InfiniBand FDR and shared virtually by vSMP_Foundation [57]. For the generation of the 27 BlnDOF, 6.7 Bln-element FEM model, most of the cost was spent on the generation of \mathbf{M}_c , the conversion of \mathbf{M}_c to \mathbf{M} , and the division of \mathbf{M} into $\{\mathbf{M}^i\}$. The generation of \mathbf{M}_c takes 3 h 43 min with 160 OpenMP threads and 698 GB computer memory, the conversion of \mathbf{M}_c to \mathbf{M} takes 4 h 53 min with 1455 GB computer memory, and the division of \mathbf{M} into $\{\mathbf{M}^i\}$ for $i = 1, 2, \dots, 82,944$ takes 13 h 59 min with 1708 GB computer memory.

Because recent increases in computer memory have largely resolved the problem of the large computer memory required, only the construction time remains an issue. Almost half of the construction time is due to file I/O while the other half consists of meshing, calling the METIS library, and mapping between M and M_c . The throughput of the file system used (NL-SAS, 7200 rpm HDD) is 100–200 MB/s using the HDF5 library, while the total file size of the models is 1649 GB. It is expected that construction time could be reduced by using faster file systems with parallel I/O. However, because we conduct many nonlinear seismic wave amplification simulations using one FEM model to estimate the response against many earthquake scenarios, this model construction time is in fact a relatively minor problem. Indeed, construction of an FEM model with less load imbalance is much more desirable, even if the construction time is relatively long.

2.2 Performance Measurement

In this study, we measured the performance of GAMERA on the K computer, a massively parallel supercomputer at RIKEN, Advanced Institute for Computational Science [42]. The K computer consists of 82,944 compute nodes, each with single SPARC64™ VIIIfx CPUs with 6 MB L2 shared cache. Each of the eight cores of the SPARC CPU has two sets of twin fused multiply-and-add (FMA) units, which leads to four multiplications and four additions in one clock cycle. The number of operations per clock cycle is the same for single and double precision floats. The operating clock speed of the CPU is 2.0 GHz, leading to a peak performance of $8 \text{ FLOP} \times 2 \text{ GHz} \times 8 \text{ CPU cores} = 128 \text{ GFLOPS}$ per CPU. Each node has 16 GB of DDR3-SDRAM memory, with peak memory bandwidth of 64 GB/s. Tofu, a six-dimensional interconnection network, is used for communication between nodes [58]. Each node can communicate in four directions simultaneously, with 5 GB/s throughput in each direction. An OpenMPI 1.4.3 [59]-based MPI library optimized for the Tofu network was used, following the MPI 2.1 standard [60].

2.2.1 Problem Setting

We show the effectivity of GAMERA by comparing four codes: GAMERA, GAMERA (DP, MG), GAMERA (DP, SG), and PCGE. GAMERA (DP, MG) is GAMERA with the mixed-precision arithmetics disabled. GAMERA (DP, SG) is GAMERA with both the mixed-precision arithmetics and the multi-grid preconditioner disabled. PCGE is GAMERA whose solver is replaced with PCG using the 3×3 block Jacobi method for the preconditioner and the EBE method for matrix–vector products. The same tuning applied to GAMERA is applied to all codes. We used a relative error of $\epsilon = 10^{-8}$, $\epsilon^{\text{in}} = 0.25$, $\epsilon_c^{\text{in}} = 0.5$ for the tolerance of the three linear solvers, i.e., the outer loop solver, the inner loop

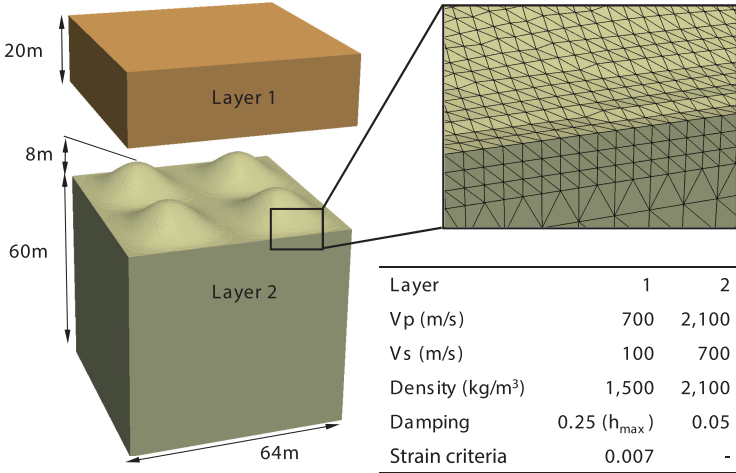


Fig. 12 Ground model for performance measurement. The 94,407,951-DOF model shown is duplicated in the x and y directions for making large-scale models with periodical geometry

solver and the inner loop solver for the coarse model. We use one compute node (one MPI process) for each M^i and M_c^i and parallelized each MPI domain using OpenMP. In the case of the K computer, $8m$ CPU cores are used for m compute nodes.

For performance measurement, we uniformly input a wave from the bottom of a two-layered ground that mimicked actual ground structures. The geometry of the ground was made periodical, so that it was suitable for measuring size-up efficiency. As shown in Fig. 12, the first layer is a soft layer that behaves nonlinearly under large earthquakes, and the second layer is a harder layer that mimics the bedrock layer. The interface between the two layers has bumps that mimic local changes in the ground structure. We duplicate this problem in the x and y directions when enlarging the problem size. We constructed an FEM model of this problem with an element size small enough (0.44 m) to obtain convergence in ground acceleration, even when the soils soften due to nonlinearity. In this case, we input a wave with 100 time steps of $dt = 0.002$ s, which was the main part of the Kobe earthquake wave [61]. Table 9 shows the model sizes and number of partitions used for measuring size-up efficiency (weak scaling) and speed-up efficiency (strong scaling).

2.2.2 Performance Results (1)

Although GAMERA is best suited for systems with a fast single-precision floating-point computing ability, we measure its performance using the K computer. The time-to-solution is important for solving large-scale problems, and thus the

Table 9 Configuration of models used for performance measurement

Model	# of MPI domains	# of CPU cores	Total DOF	Mean DOF per MPI domain	σ_n (%)	σ_e (%)	# of PCGE iterations
1	4608	36,864	1,505,755,135	326,767	4.13	0.86	355
2	9216	73,728	3,010,699,983	326,681	4.37	0.70	355
3	18,432	147,456	6,019,818,087	326,596	4.41	0.68	355
4-A	9216	73,728	12,038,067,615	1,306,213	2.67	0.46	355
4-B	18,432	147,456	12,038,067,615	653,107	3.40	0.55	355
4-C	36,864	294,912	12,038,067,615	326,553	4.55	0.68	355
5	82,944	663,552	27,082,129,647	326,511	4.51	0.75	355

σ_n, σ_e indicates the maximum imbalance of the number of nodes and elements. $\sigma_n = (\max n - \text{mean } n) / \text{mean } n$, $\sigma_e = (\max e - \text{mean } e) / \text{mean } e$

scalability for solving the large-scale problems becomes essential. We measure performance using the ninth time step because this step is not influenced by the fluctuations in the initial part of the wave and a poor convergence problem arises at this step.

Although problem characteristics change according to the DOF of the problem, the models used here have almost the same convergence characteristics due to the periodical geometry. In fact, we confirm that the number of loops elapsed for convergence of the PCGE solver is the same for all of the models, as shown in Table 9, and thus we use this set of the FEM models to measure the size-up efficiency of the codes. We first fix the problem size per compute node to 326 K DOF and increase the problem size and the number of CPU cores to measure the size-up efficiency of the code.

Figure 13 shows the elapsed time and its breakdown for solving the ninth time step. Here, “inner coarse,” “inner fine,” and “outer” indicate line 17 of Algorithm 1, line 19 of Algorithm 1, and the outer loop, respectively. The numbers in brackets in the figure indicate the number of the loops elapsed for convergence of the linear solver. It is seen that the numbers of loops elapsed are almost the same for all the models. Also, it is seen that the increase in the elapsed time is small. Therefore, we conclude that the code scales well to a large number of CPU cores. The smallest model (model 1, 1.5 BlnDOF) is computed in 1.42 s using 36,864 CPU cores (4608 compute nodes \times 8 CPU cores, 8.69 % of peak performance, 51.3 TFLOPS), and the largest model (model 5, 27 BlnDOF) is computed in 1.63 s using the whole K computer, with 663,552 CPU cores (82,944 compute nodes \times 8 CPU cores, 7.57 % of peak performance, 0.804 PFLOPS), indicating a size-up efficiency of 87.1 %.

Compared with the outer and inner fine loops, the inner coarse loop has a larger increase in elapsed time. Next, we show the breakdown of the elapsed time in

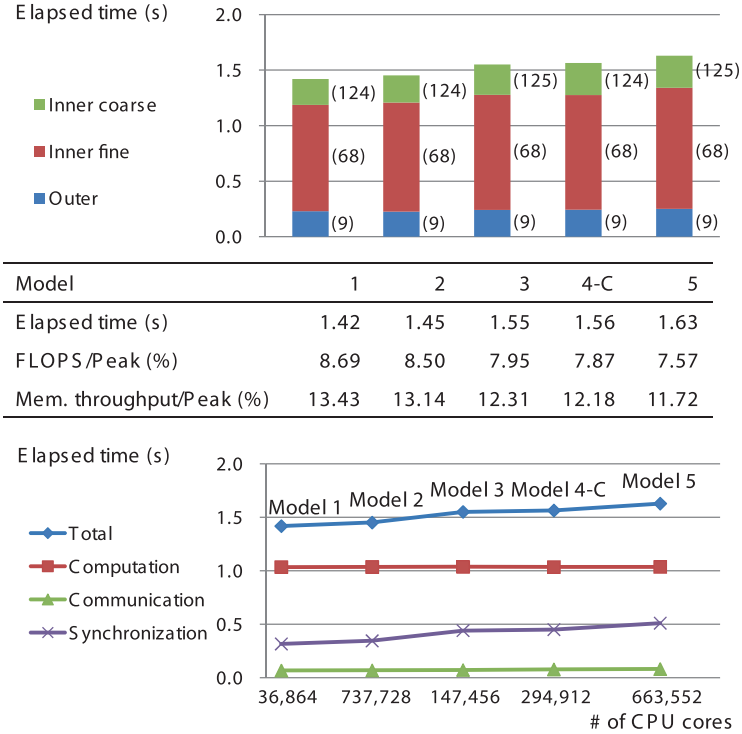


Fig. 13 Elapsed time and its breakdown (inner fine, inner coarse, outer loop) for the ninth time step of the weak-scaling problem set. *Numbers in brackets* indicate the number of elapsed loops required for convergence of the linear solver. The breakdown of the elapsed time for computation, communication, and synchronization is also shown

terms of computation, communication, and synchronization.² It may be expected that the communication time would increase with respect to model size due to the increase in the number of hops in peer-to-peer communications and the increase in the number of nodes involved in all-reduce operations. However, the increase in communication time is, in fact, only 0.014 s; the communication time of model 5 and

² Defined as:

$$\begin{aligned}
 \text{communication} &= \text{AVG}\{\text{MPI_Isend} + \text{MPI_Irecv} \\
 &\quad + \text{MPI_Allreduce}(\text{total}) - \text{MPI_Allreduce}(\text{wait})\} + \text{MIN}\{\text{MPI_Waitall}\}, \\
 \text{synchronization} &= \text{AVG}\{\text{MPI_Allreduce}(\text{wait}) + \text{MPI_Waitall}\} - \text{MIN}\{\text{MPI_Waitall}\}, \\
 \text{computation} &= \text{AVG}\{\text{total}\} - \text{communication} - \text{synchronization}.
 \end{aligned}$$

Here, AVG and MIN indicate the average and minimum values of all the compute nodes, respectively. MPI_Barrier is not used.

model 1 is 0.081 and 0.067 s. Such a small increase in the communication time is due to the decrease in the communication size for the peer-to-peer communication and the hardware enhancements in the K computer's interconnect for all-reduce operations [58]. However, we can see that synchronization time has increased from 0.316 (model 1) to 0.511 s (model 5).

The imbalance of the models are summarized in Table 9. It is seen that the imbalance in the number of elements and nodes increases with respect to the model size. The amount of EBE and CRS computations is correlated with the number of elements and nodes, respectively. We might guess that the larger imbalance in the number of nodes is the cause of worse deterioration in the performance of the inner coarse solver. Although load imbalance due to I/O or jitters of the OS is included in the synchronization time, we can expect to improve performance by decreasing the imbalance in the number of nodes and elements by tuning the parameters of METIS.

We measure the speed-up efficiency of the code. Figure 14 shows the change in elapsed time with respect to the change in the number of CPU cores. It is seen that the 12 BlnDOF is solved in 4.92 s using 73,728 CPU cores (9216 compute nodes \times 8 CPU cores, 117.8 TFLOPS, 9.99 % of peak) and that the same problem is solved in 1.56 s using 294,912 CPU cores (36,864 compute nodes \times 8 CPU cores, 371.4 TFLOPS, 7.87 % of peak), with a relatively good speed-up efficiency of 78.5 %. In view of the breakdown of the elapsed time, it is seen that the outer and inner fine loops have high scalability, above 85 %. However, the increase in the time used for solving the inner coarse loop is prominent (with 38.9 % speed-up efficiency); this is the primary reason for the deterioration in the scalability of the whole program. As is the case with the weak-scaling models, this may also be due to the large increase in the imbalance in the number of nodes with respect to the increase in the number of compute cores.

We show the breakdown of the elapsed time in terms of computation, communication, and synchronization. We can see that the time for communication and synchronization has increased, but the time for communication is still well suppressed. Figure 15 shows the effectivity of mixed-precision arithmetics and the multi-grid preconditioner for model 5 using the whole K computer with 663,552 CPU cores (82,944 compute nodes \times 8 CPU cores). Because the number of floating-point operations per clock cycle does not change for single-precision or double-precision numbers, the peak FLOPS are the same for both. However, the bytes required are halved when using single precision, which would be expected to increase performance. First, we compare GAMERA and GAMERA (DP, MG). Because the B/F of the preconditioner doubles by changing to double precision, FLOPS decreases, and the elapsed time increases to 1.15 times that of GAMERA.

Next, we compare GAMERA and GAMERA (DP, SG). The computation of M_c^i (line 17 of Algorithm 1) uses linear elements with a CRS format, whereas the computation of M^i (line 19 of Algorithm 1) uses second-order elements with EBE. Thus, the B/F of the inner coarse loop becomes larger than that of the inner fine loop, and the floating-point arithmetic performance of the inner fine loop becomes better

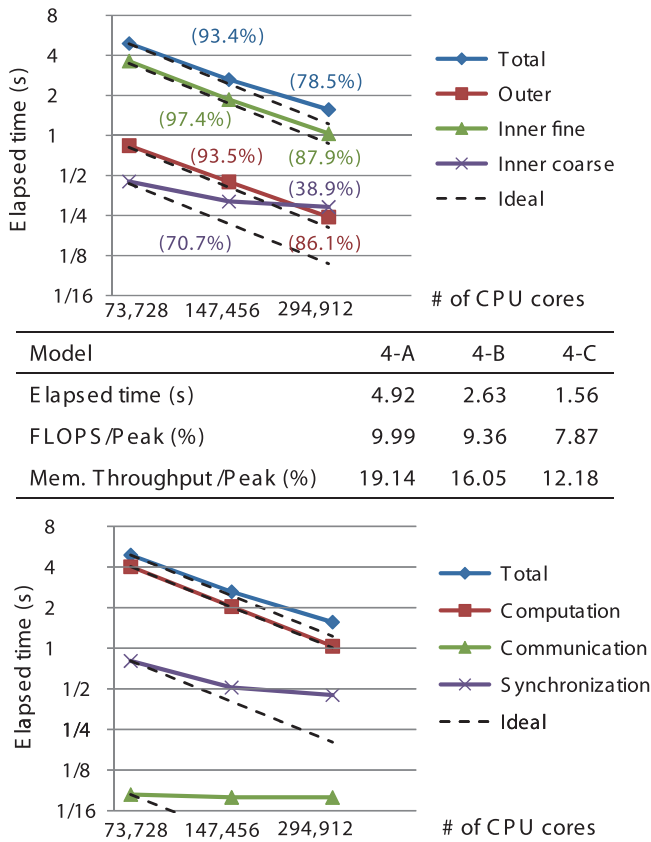


Fig. 14 Elapsed time and its breakdown (inner fine, inner coarse, outer loop) for the ninth time step of the strong-scaling problem set. *Numbers in brackets* show the speed-up efficiency with respect to Model 4-A (73,728 CPU cores). The breakdown of the elapsed time for computation, communication, and synchronization is also shown

than the inner coarse loop. Consequently, the FLOPS performance of GAMERA (DP,SG) increases by eliminating the inner coarse solver, but the number of loops needed for convergence increases, and thus the total time for solving the matrix equation increases, to 4.26 times that of GAMERA. Most of the cost for PCGE is involved in the EBE computation for second-order elements. Because we use the tuned EBE computation developed in this paper, the FLOPS and memory throughput are good, and thus it is possible to compute each loop in a short time. However, the number of loops needed for convergence becomes large and thus the time required to solve the matrix equation is 3.53 times that of GAMERA. Although the PCGE is slightly better in terms of peak performance when using the K computer, we can see that the multi-grid preconditioner is highly effective

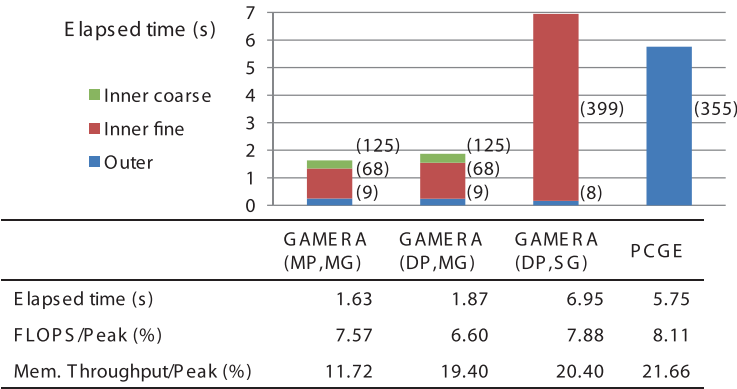


Fig. 15 Comparison of performance of GAMERA, GAMERA (DP,MG), GAMERA (DP,SG), and PCGE for solving the ninth time step of Model 5 (27 BlnDOF, 663,552 CPU cores). *Numbers in brackets indicate the number of elapsed loops required for convergence of the linear solver*

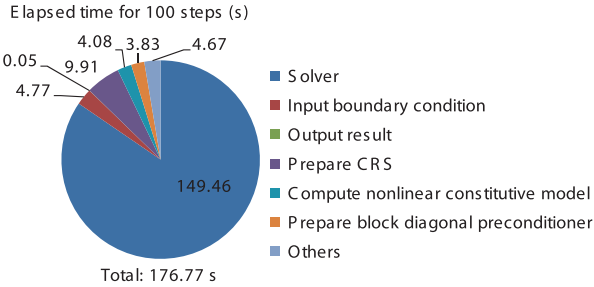


Fig. 16 Breakdown of the elapsed time used for analyzing 100 steps of Model 5 (27 BlnDOF, 663,552 CPU cores). 0.736 PFLOPS (6.93 % of peak) was attained for the whole program

and that the multi-grid method combined with mixed-precision arithmetics results in a time to solution more than 3 times faster compared with PCGE.

2.2.3 Performance Results (2)

All of the measurements above concerned the cost of the linear solver; we show the total cost of the 100 time steps in Fig. 16. In addition to the solver, the program involves preparing CRS of the coarse mesh for every time step, updating the material parameters using the strain of the last time step, and computing the block Jacobi matrix used for the preconditioner. Because the input is the same for all ranks, rank 0 reads the input wave from the file system and broadcasts it to other processors. The output of the 100-step simulation is gathered per rank and outputted in binary to reduce the number of files and increase throughput. By such measures, the time spent for I/O is reduced to 2.7 % of the whole program. Also, the other parts

take 12.7 % of the whole elapsed time. Thus, we can see that most of the time is used for the solver (84.6 %) and that the performance of the solver dominates the performance of the whole analysis. Although the performance of GAMERA was high for the 100 steps solved here, we can expect increased effectivity of GAMERA when solving poorer convergency problems, considering more complex grounds or input waves with phase differences.

Finally, we checked the potential of GAMERA for use in an environment favorable for single-precision arithmetics. The floating-point performance of the Intel Xeon E5-2680 CPU for single-precision floats is twice that for double-precision floats, and this was expected to lead to a large change in time-to-solution. Figure 17 shows the elapsed time and performance of GAMERA and GAMERA (DP,MG). Here, we use 256 compute nodes of Stampede at Texas Advanced Computing Center, The University of Texas at Austin [62], and 256 compute nodes of the K computer. Although each compute node of Stampede has an Intel Xeon Phi Coprocessor, we only used the dual Xeon E5-2680 CPUs for measurement. We set the DOF per compute node to 369 K DOF, which is similar to that of the weak scaling models in Table 9. Then, 16 OpenMP threads were used per compute node of Stampede, and eight OpenMP threads were used per compute node of the K computer. From the figure, we can see that both the inner fine and inner coarse loops used in the preconditioner were accelerated when using mixed-precision arithmetics for both of the computer systems. The acceleration due to the use of mixed-precision arithmetics was about 1.27 times for the K computer with SPARC CPUs and 1.44 times for Stampede with Intel CPUs. Thus, we can expect further speed-up of GAMERA when using hardware with faster single-precision operation capabilities.

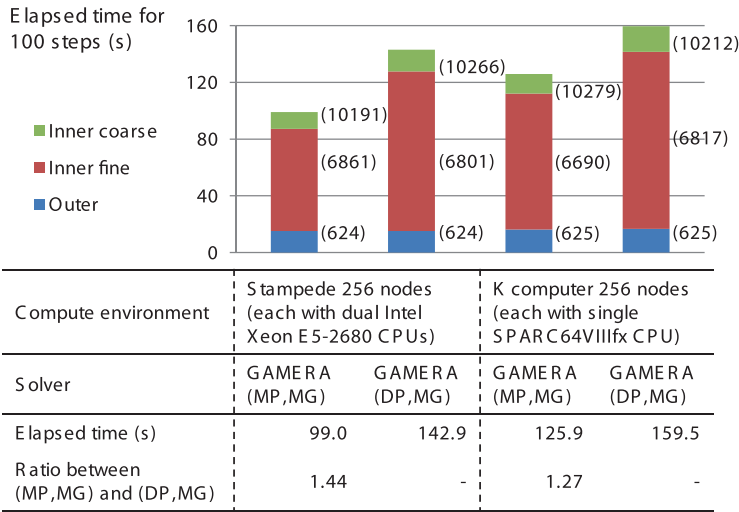


Fig. 17 Comparison of the effectiveness of multi-precision arithmetics when using Intel CPU- and SPARC CPU-based systems. *Numbers in brackets* indicate the number of elapsed loops required for convergence

In the Appendix, we report performance of GAMERA^{EBE4}, which is aimed to further reduce time-to-solution when solving larger DOF problems. Due to restrictions in computational resources, we have not been able to perform tests up to the full K computer, but we have attained favorable results compared with GAMERA, such as: elapsed time reduced by 49 % for model 4-A (73,728 CPU cores, 17.1 % of peak FLOPS) and 29 % for model 4-C (294,912 CPU cores, 11.8 % of peak FLOPS). We plan to develop both GAMERA and GAMERA^{EBE4} which are suitable for different types of hardware in the future.

2.3 Application Example

As an application example, GAMERA is used for the seismic wave amplification simulation that is input to the urban area seismic response simulation. The target area is a 2.0×2.0 km region of central Tokyo; this region consists of businesses, shops, and densely built residential districts, with a total of 13,275 buildings. The ground structure has a rectangular shape of 2.0×2.0 km and a depth of 100 m. The ground structure is modeled with three soil layers using the 5-m grid elevation map [63] and digital geotechnical database [64]. Figure 18 shows the geometry and the material properties of the ground. Material properties of each layer were set based on data at the nearest earthquake observation point [65]. The Kobe earthquake wave [61] with $dt = 0.001$ s and 30,000 time steps is inputted into the bottom of the model. This simulation is carried out on the K computer using 294,912 CPU cores (36,864 compute nodes \times 8 CPU cores). Relative errors of $\epsilon = 10^{-8}$, $\epsilon^{\text{in}} = 0.25$, $\epsilon_c^{\text{in}} = 0.5$ are used for the tolerances of the linear solvers. The FEM model is constructed with an element size small enough to obtain convergence in ground acceleration, even when the soils softened due to nonlinearity. This resulted in a model with 10.7 BlnDOF and a minimum element size of 0.66 m.

Figure 19 shows the breakdown of the elapsed time of the simulation. It takes 41,521 s to compute 30,000 time steps, including I/O, meaning that only 1.38 s was used for solving each time step. Most of the computation time is spent solving the linear equation system (35,245 s). The performance of the whole program is high due to the reduction in I/O cost (1481.6 s for inputting data and 2.5 s for outputting the results). Displacements at the surface were outputted with 0.33-m spatial resolution every 10 time steps, leading to total output size of 1.32 TB. For the first 1000 time steps, the performance of the linear solver was 0.421 PFLOPS (8.93 % of peak), and the performance of the whole program was 0.358 PFLOPS (7.59 % of peak); we can see that high performance was realized on an actual application run.

Figure 20a shows the response of ground at the surface as well as the response of structures using the computed ground motion as an input. In Fig. 20b, it is shown that ground motion responses are far from being uniform and resulting structures shaking change. In an ordinary ground motion analyses, it is difficult to ensure

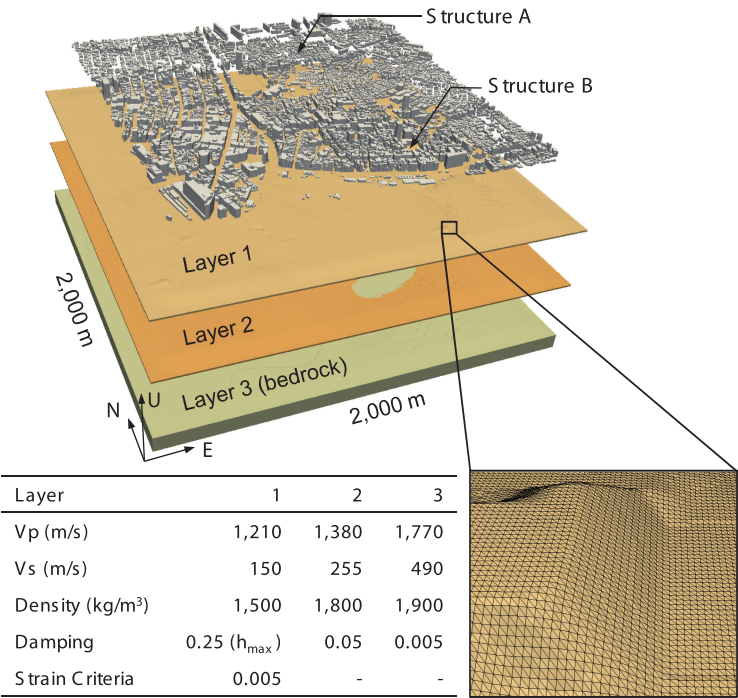


Fig. 18 Ground and structure model of the application example. A domain of 2.0×2.0 km with 13,275 structures was targeted. The three-layered ground was modeled with minimum element size of 0.66 m, leading to a 10,755,536,091-DOF model, and computed using 294,912 CPU cores (36,864 compute nodes) of the K computer

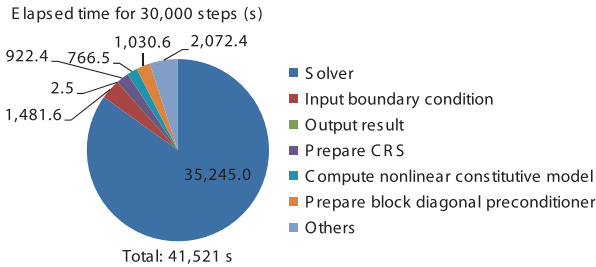


Fig. 19 Computation performance of a 10.7 BlnDOF ground model. For the first 1000 time steps, the performance of the linear solver was 0.421 PFLOPS (8.93 % of peak), and the performance of the whole program was 0.358 PFLOPS (7.59 % of peak)

the convergence of acceleration due to the fatal computational costs, and thus the reliability of results is not necessarily high. However, we are able to achieve highly reliable results that converge in terms of acceleration response for a large domain using GAMERA. Here, we use three models with different resolutions: model I with the highest resolution (10.7 BlnDOF model), model II with 1.5 times the element

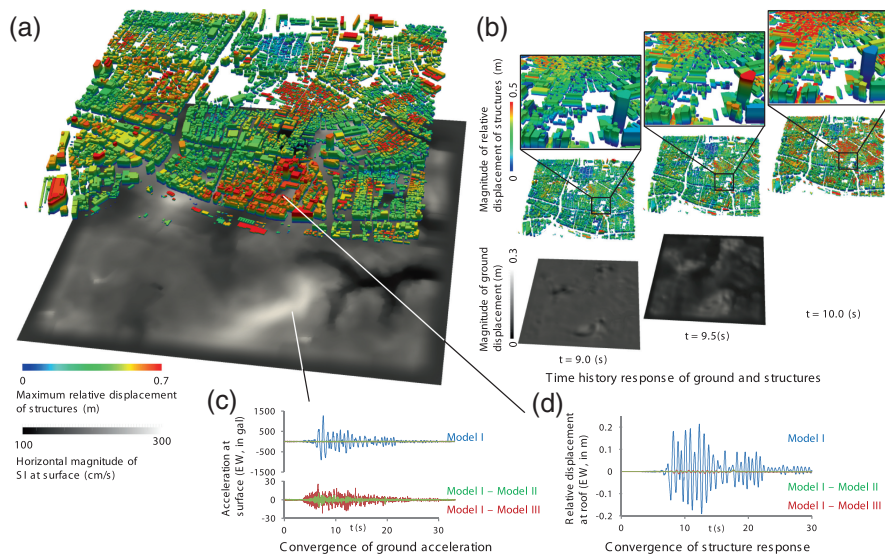


Fig. 20 Response of ground and structures. (a) Horizontal magnitude of SI at surface and maximum relative displacement of structures. (b) Snapshots of time history response. (c) Acceleration waveforms at surface (Point A) obtained from Model I (10.7 BlnDOF model), Model II (with element sizes 1.5 times larger than Model I), and Model III (with element sizes 3 times larger than Model I). We can see that the ground acceleration converges with the increase in resolution of the models. (d) Relative displacement waveforms of structure A (a two-story RC building located above Point A) obtained using acceleration waveforms computed using Models I, II, and III. We can also see the structure responses converge with respect to the increase in ground model resolution

size of model I, and model III with 3.0 times the element size of model I. Figure 20c shows the comparison of acceleration time history at the surface (Point A). It is seen that the waveform converges with respect to the increase in the discretization resolution of the FEM model and that the waveform of the finest model converges within 1 % of the amplitude of the wave. As they are dependent on the input acceleration at the ground surface, the responses of structures also converge with the increase in resolution; see Fig. 20d.

From these results, we can conclude that high-resolution finite-element modeling of ground motion is essential for analyzing the response of structures. By comparing the ground and structure responses shown in Fig. 20a, it is seen that larger SI values (an index commonly used for evaluating the destructiveness of waves to buildings [66]) do not necessarily lead to large structural responses. Such a complexity in the structure seismic response distribution in an urban area can only be realized by combining high-resolution 3-D ground motion analysis and structure response analysis.

Considering the ambiguities included in the information used for making structure models may further improve the reliability of response analyses of structures in

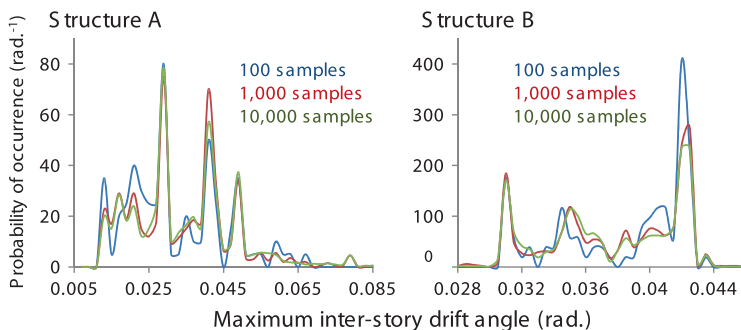


Fig. 21 Probability density of response of two two-story RC structures with stochastic structural parameters. The horizontal axis indicates the maximum inter-story drift angle (an index frequently used to evaluate damage of structures), and the vertical axis indicates the probability of occurrence. We can see that the distribution converges by increasing the number of samples from 100 to 1000 and then to 10,000 samples

a city [67]. Figure 21 shows the result of a stochastic analyses (Monte Carlo simulation) considering the ambiguity of concrete stiffness, following a normal distribution. Although we show only two structures in Fig. 21, we can simulate the stochastic response of all the buildings in the urban area. Such analyses are expected to contribute to improving the reliability of earthquake damage estimation in urban areas. On the other hand, large numbers of samples are needed for attaining convergence in the probability response distribution, and the computational costs for stochastic response analysis of structures become huge. For example, the seismic structural simulations for $13,275 \text{ buildings} \times 10,000 \text{ samples}$ shown here took 3 h, 56 min using 80,000 CPU cores ($10,000 \text{ compute node} \times 8 \text{ CPU cores}$) of the K computer. We can see that the supercomputers can also serve an important role for improving the reliability of structure response analyses as well as ground motion analyses.

2.4 Concluding Remarks

In this chapter, GAMERA is presented as an efficient code for a nonlinear dynamic FEM for the seismic wave amplification simulation that solves nonlinear wave equations. The high computation performance of GAMERA is attributable to the smart use of the computer architecture in solving a physics problem of a wave equation. In particular, the algorithm developed results in a very fast and portable iterative solver. We regard GAMERA as a next-generation dynamic FEM due to its excellent performance: 1.42-s run time for 1.5 BlnDOF with 36,864 CPU cores and 1.63-s run time for 27 BlnDOF with 663,552 CPU cores of the K computer. The scalability (size-up efficiency) was 87.1 %. This performance enabled us to perform a physics-based seismic wave amplification simulation for Tokyo. A problem of 10.7 BlnDOF and 30 K time steps was solved in 11 h, 32 min using 294,912 CPU

cores of the K computer. The average runtime was 1.38 s, which is remarkably fast in terms of time-to-solution.

We should point out the high portability of GAMERA. The architecture of the K computer makes single-precision arithmetic not much faster than double-precision arithmetic. The algorithm using MP arithmetic is still efficient for the K computer, but it is expected that the performance will be improved with other computers, such as Intel CPU- or GPU-based machines, which allow single-precision arithmetic faster than double-precision arithmetic. Indeed, the speed-up due to this algorithm on the K computer was 1.27-fold, while that on an Intel CPU machine was 1.44-fold. A crude estimate shows that the speed-up of the GAMERA would be 4.0 ($= 3.53 \times 1.44 / 1.27$) with reference to the original PCGE. Currently, the fast solver of GAMERA is being implemented on Intel CPU and GPU machines; a speed-up of 6.99 on an Intel CPU was observed for a small-scale crust deformation problem [68] that used a model of 158 M DOF.

This study has several implications for future systems. GAMERA does not cause any problem with inter-node communication on the K computer, since the algorithm for computing vectors and matrices in the FEM was tuned to minimize data transfer time among the nodes in the most stable manner. Fast inter-node connections with Torus fusion (Tofu) on the K computer also contribute to this smooth inter-node communication. We expect further improvements in inter-node communication technology, which is available for a larger class of parallel computers. In particular, reducing the latency of MPI communication is very desirable in enhancing frequent communication of small amounts of data. Additionally, frequent I/O with large sizes and large numbers of files is involved in constructing the model and visualization of results; we expect further improvements in file I/O technology on a larger class of parallel computers in handling such files.

We explain GAMERA^{EBE4}, which aims to further improve the time-to-solution for solving larger DOF problems. Since preparation of CRS matrices used for the inner coarse loop takes considerable time, we change the CRS used in the inner coarse loop to EBE in GAMERA^{EBE4}. Computation of matrix–vector products by EBE involves more computation than matrix–vector products by CRS, but it is expected that the total time including preparation of CRS matrices can be reduced by using EBE. We also improve the summation of vectors computed by OpenMP in EBE computations in the inner and outer EBE loops, and the computation of block Jacobi matrices in line 4 of Algorithm 1.

Figure 22 shows the size-up efficiency of GAMERA and GAMERA^{EBE4}. As expected, the inner coarse loop is slightly faster using CRS, but the inner coarse loop using EBE takes less time than the sum of preparation of CRS matrices (included in “Others”) and the inner coarse loop. Compared with using CRS, using EBE for matrix–vector multiplication involves 5.6 times the floating point operations, but the data used for EBE computation fits inside the 6 MB L2 cache and improves the floating point performance (31.3 % of peak FLOPS is attained for serial performance of EBE in inner coarse loop). Thus, the time used for the inner coarse loop is only increased from 19.1 to 23.2 s for model 1. Combined with the improvement in OpenMP implementations of the inner fine loop, the

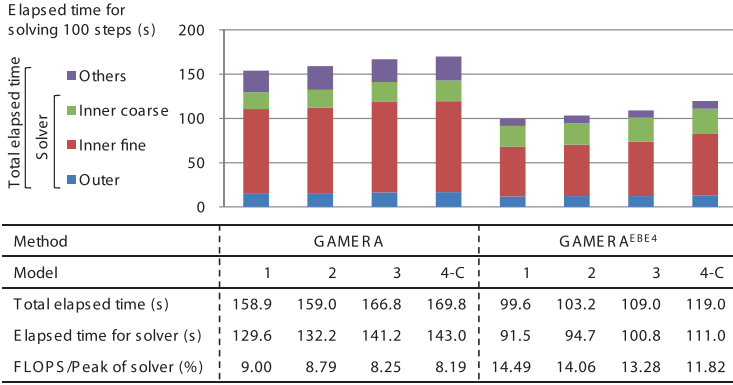


Fig. 22 Size-up efficiency of GAMERA and GAMERA^{EBE4}. GAMERA^{EBE4} uses EBE for matrix–vector multiplications in the inner coarse loop instead of CRS used in GAMERA. OpenMP implementations of EBE computations in inner/outer loops are also improved

total elapsed time for solving 100 time steps improved by 59.5 % for model 1 and 42.7 % for model 4-C. The floating point performance has also improved from 9.0 to 14.5 % for model 1, and from 8.2 to 11.8 % for model 4-C. Since the size and frequency of communication is the same for both methods, the ratio of time spent for communication and synchronization has increased in GAMERA^{EBE4} and thus the scalability has decreased from 93.5 to 83.7 % when comparing model 1 and 4-C. Since time-to-solution is greatly improved, it is possible to compute larger problems per CPU core, and thus it is expected that the drop in scalability will not be as bad when used in practice.

Figure 23 shows the speed-up efficiency of the solvers of GAMERA and GAMERA^{EBE4}. The scalability of the inner coarse loop using CRS deteriorates at a smaller number of CPU cores than the inner coarse loop using EBE. On the other hand, the scalability of the total solver drops from 79.8 % (GAMERA) to 68.9 % (GAMERA^{EBE4}), when comparing model 4-A and 4-C. The reason for the drop in scalability can also be due to the increase in ratio of communication and computation time as explained in the size-up efficiency case. From the lower figure, we can see that the time spent for synchronization using CRS does not decrease with the increase in number of CPU cores, while the time spent for synchronization using EBE is decreased by increasing the number of CPU cores. Such difference in synchronization time could be due to the variability in computation time among CPUs due to the difference in matrix structures (e.g., position of non-zero elements, node numbering, and element numbering), the effect of such matrix structures is larger for CRS than for EBE. In these measurements, inner coarse loops using CRS is faster regardless of the number of CPU cores. This can be due to the high B/F of the K computer’s SPARC CPUs (64 GB/s/128 GFLOPS = 0.5 bytes/FLOP), which leads to fast access to memory. By changing from CRS to EBE, the total memory usage

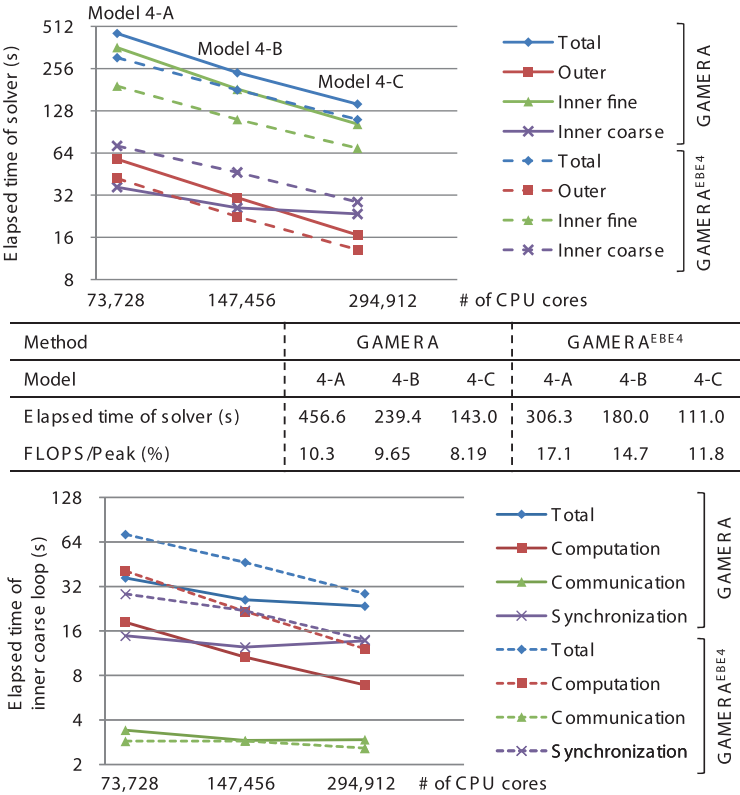


Fig. 23 Speed-up efficiency of GAMERA and GAMERA^{EBE4}

decreases from 1214.9 to 982.8 MB for model 4-A and 584.3 to 411.1 MB for model 4-C, and thus we can increase the problem size per compute node when using EBE.

In summary, we further improved total time-to-solution by GAMERA^{EBE4}, which uses EBE for inner coarse loop together with tuned OpenMP computation. By improving the OpenMP computation of GAMERA, we can expect similar improvements for the inner fine loop. In such case, the choice of CRS or EBE depends on the hardware characteristics; we expect that CRS will be suitable for hardware with large and fast memory, while EBE will be suitable for hardware with small memory but fast floating point computations.

References

1. Day, S.M., Bielak, J., Dreger, D., Graves, R.W., Larsen, S., Olsen, K.B., Pitarka, A.: Tests of 3D Elastodynamic Codes: Final Report for Lifelines Project 1A03, Pacific Earthquake Engineering Research Center (2005)

2. Frankel, A.: Three-dimensional simulations of ground motions in the San Bernardino Valley, California, for hypothetical earthquakes on the San Andreas fault. *Bull. Seismol. Soc. Am.* **83**, 1020–1041 (1993)
3. Graves, R.W.: Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences. *Bull. Seismol. Soc. Am.* **86**, 1091–1106 (1996)
4. Furumura, T., Koketsu, K.: Specific distribution of ground motion during the 1995 Kobe earthquake and its generation mechanism. *Geophys. Res. Lett.* **25**, 785–788 (1998)
5. Pitarka, A.: 3D elastic finite-difference modeling of seismic motion using staggered grids with non-uniform spacing. *Bull. Seismol. Soc. Am.* **89**, 54–68 (1999)
6. Bao, H., Bielak, J., Ghattas, O., Kallivokas, L.F., O'Hallaron, D.R., Shewchuk, J., Xu, J.: Large-scale simulation of elastic wave propagation in heterogeneous media on parallel computers. *Comput. Methods Appl. Mech. Eng.* **152**, 85–102 (1998)
7. Koketsu, K., Fujiwara, H., Ikegami, Y.: Finite-element simulation of seismic ground motion with a voxel mesh. *Pure Appl. Geophys.* **161**, 2463–2478 (2004)
8. Bielak, J., Ghattas, O., Kim, E.J.: Parallel octree-based finite element method for large-scale earthquake ground motion simulation. *Comput. Model. Eng. Sci.* **10**, 99–112 (2005)
9. Ma, S., Liu, P.: Modeling of the perfectly matched layer absorbing boundaries and intrinsic attenuation in explicit finite-element methods. *Bull. Seismol. Soc. Am.* **96**, 1779–1794 (2006)
10. Ichimura, T., Hori, M., Kuwamoto, H.: Earthquake motion simulation with multi-scale finite element analysis on hybrid grid. *Bull. Seismol. Soc. Am.* **97**, 1133–1143 (2007)
11. Ichimura, T., Hori, M., Bielak, J.: A hybrid multiresolution meshing technique for finite element three-dimensional earthquake ground motion modeling in basins including topography. *Geophys. J. Int.* **177**, 1221–1232 (2009)
12. Moczo, P., Kristeka, J., Galisb, M., Pazaka, P., Balazovjecha, M.: The finite-difference and finite-element modeling of seismic wave propagation and earthquake motion. *Acta Phys. Slovaca* **57**, 177–406 (2007)
13. Belytschko, T., Liu, W.K., Moran, B.: *Nonlinear Finite Elements for Continua and Structures*. Wiley, New York (2000)
14. Zienkiewicz, O.C., Taylor, R.L.: *The Finite Element Method*, 4th edn. Basic Formulation and Linear Problems, vol. 1. McGraw-Hill, London (1989)
15. Wu, S.R.: Lumped mass matrix in explicit finite element method for transient dynamics of elasticity. *Comput. Methods Appl. Mech. Eng.* **195**, 5983–5994 (2006)
16. Strang, G., Fix, G.J.: *An Analysis of the Finite Element Method*. Prentice Hall, Englewood Cliffs (1973)
17. Komatitsch, D., Vilotte, J.: The spectral element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures. *Bull. Seismol. Soc. Am.* **88**, 368–392 (1998)
18. Dumbser, M., Kaser, M.: An arbitrary high order discontinuous Galerkin method for elastic waves on unstructured meshes ii: the three-dimensional isotropic case. *Geophys. J. Int.* **167**, 319–336 (2006)
19. Matsumoto, J., Takada, N.: Two-phase flow analysis based on a phase-field model using orthogonal basis bubble function finite element method. *Int. J. Comput. Fluid Dyn.* **22**, 555–568 (2008)
20. Wijerathne, M.L.L., Oguni, K., Hori, M.: Numerical analysis of growing crack problems using particle discretization scheme. *Int. J. Numer. Methods Eng.* **80**, 46–73 (2009)
21. Flanagan, D.P., Belytschko, T.: A uniform strain hexahedron and quadrilateral with orthogonal hourglass control. *Int. J. Numer. Methods Eng.* **17**, 679–706 (1981)
22. Belytschko, T., Bindman, L.P.: Assumed strain stabilization of the eight node hexahedral element. *Comput. Methods Appl. Mech. Eng.* **105**, 225–260 (1993)
23. Lysmer, J., Kuhlemeyer, R.L.: Finite dynamic model for infinite media. *J. Eng. Mech. ASCE* **95**, 859–877 (1969)
24. Hisada, Y.: An efficient method for computing Green's functions for a layered half-space with sources and receivers at close depths. *Bull. Seismol. Soc. Am.* **84**, 1456–1472 (1994)

25. Kristeková, M., Kristek, J., Moczo, P., Day, S.M.: Misfit criteria for quantitative comparison of seismograms. *Bull. Seismol. Soc. Am.* **96**, 1836–1850 (2006)
26. Japan Nuclear Energy Safety Organization: Working report on stochastic estimation for earthquake ground motion. JNES/SAE05–048 (2005)
27. Ma, S., Archuleta, R.J., Morgan, T.: Effects of large-scale surface topography on ground motions, as demonstrated by a study of the San Gabriel Mountains, Los Angeles, California. *Bull. Seismol. Soc. Am.* **97**, 2066–2079 (2007)
28. Somerville, P., Collins, N., Abrahamson, N., Graves, R., Saikia, C.: Ground Motion Attenuation Relations for the Central and Eastern United States. Final Report, 30 June 2001 [Online]. <http://www.earthquake.usgs.gov/hazards/products/conterminous/2008/99HQGR0098.pdf>
29. Second Report of the Nankai Trough Large Earthquake Model Committee, Cabinet Office, Government of Japan, 28 August 2012 [Online]. <http://www.bousai.go.jp/jishin/nankai/model/index.html>
30. Disaster Assessment of Tokyo Due to Large Earthquakes Such as the Nankai Trough Earthquake, Tokyo Metropolitan Government, 14 May 2013 [Online]. <http://www.bousai.metro.tokyo.jp/taisaku/1000902/1000402.html>
31. Tiankai, T., Hongfeng, Y., Ramirez-Guzman, L., Bielak, J., Ghattas, O., Kwan-Liu, M., O'Hallaron, D.R.: From mesh generation to scientific visualization: an end-to-end approach to parallel supercomputing. In: *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC '06)*. ACM, New York, NY (2006), Article 91. doi:10.1145/1188455.1188551
32. Yifeng Cui, C., Olsen K.B., Jordan, T.H., Lee, K., Zhou, J., Small, P., Roten, D., Ely, G., Panda, D.K., Chourasia, A., Levesque, J., Day, S.M., Maechling, P.: Scalable earthquake simulation on petascale supercomputers. In: *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10)*, pp. 1–20. IEEE Computer Society, Washington, DC (2010). doi:10.1109/SC.2010.45. <http://www.dx.doi.org/10.1109/SC.2010.45>
33. Rietmann, M., Messmer, P., Nissen-Meyer, T., Peter, D., Basini, P., Komatitsch, D., Schenk, O., Tromp, J., Boschi, L., Giardini, D.: Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12)*, 11 pp. IEEE Computer Society Press, Los Alamitos, CA (2012), Article 38
34. Cui, Y., Poyraz, E., Olsen, K.B., Zhou, J., Withers, K., Callaghan, S., Larkin, J., Guest, C., Choi, D., Chourasia, A., Shi, Z., Day, S.M., Maechling, P.J., Jordan, T.H.: Hysics-based seismic hazard analysis on petascale heterogeneous supercomputers. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, (SC'13)*, 12 pp. IEEE Computer Society Press, New York, NY (2013), Article 70
35. Hori, M., Ichimura, T.: Current state of integrated earthquake simulation for earthquake hazard and disaster. *J. Seismol.* **12**(2), 307–321 (2008). doi:10.1007/s10950-007-9083-x
36. Wijerathne, M.L.L., Hori, M., Kabeyazawa, T., Ichimura, T.: Strengthening of parallel computation performance of integrated earthquake simulation. *J. Comput. Civil Eng.* **27**, 570–573 (2013)
37. Fujita, K., Ichimura, T., Hori, M., Wijerathne, M.L.L., Tanaka, S.: Basic study on high resolution seismic disaster estimation of cities under multiple earthquake hazard scenarios with high performance computing. *J. Jpn. Soc. Civil Eng. Ser. A2 (Appl. Mech.)* **69**(2), I_415–I_424 (2013) (in Japanese with English abstract)
38. Liang, J., Sun, S.: Site effects on seismic behavior of pipelines: a review. *J. Pressure Vessel Technol. (Am. Soc. Mech. Eng.)* **122**, 469–475 (2000)
39. Taborda, R., Bielak, J.: Large-scale earthquake simulation: computational seismology and complex engineering systems. *Comput. Sci. Eng.* **13**, 14–27 (2011)
40. Taborda, R., Bielak, J., Restrepo, D.: Earthquake ground-motion simulation including nonlinear soil effects under idealized conditions with application to two case studies. *Seismol. Res. Lett.* **83**(6), 1047–1060 (2012)
41. Ichimura, T., Fujita, K., Hori, M., Sakanoue, T., Hamanaka, R.: Three-dimensional nonlinear seismic ground response analysis of local site effects for estimating seismic behavior of buried

- pipelines. *J. Pressure Vessel Technol. (Am. Soc. Mech. Eng.)* **136**, 041702 (2014)
42. Miyazaki, H., Kusano, Y., Shinjou, N., Shoji, F., Yokokawa, M., Watanabe, T.: Overview of the K computer system. *FUJITSU Sci. Tech. J.* **48**(3), 302–309 (2012)
 43. Idriss, I.M., Singh, R.D., Dobry, R.: Nonlinear behavior of soft clays during cyclic loading. *J. Geotech. Eng. Div.* **104**, 1427–1447 (1978)
 44. Masing, G.: Eigenspannungen und verfestigung beim messing. In: *Proceedings of the 2nd International Congress of Applied Mechanics*, pp. 332–335 (1926) (in German)
 45. Akiba, H., Ohyama, T., Shibata, Y., Yuyama, K., Katai, Y., Takeuchi, R., Hoshino, T., Yoshimura, S., Noguchi, H., Gupta, M., Gunnel, J., Austel, V., Sabharwal, Y., Garg, R., Kato, S., Kawakami, T., Todokoro, S., Ikeda, J.: Large scale drop impact analysis of mobile phone using ADVN on Blue Gene/L. In: *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC '06)*, Article 46. ACM, New York, NY (2006). doi:10.1145/1188455.1188503
 46. Ogino, M., Shioya, R., Kanayama, H.: An inexact balancing preconditioner for large-scale structural analysis. *J. Comput. Sci. Technol.* **2**(1), 150–161 (2008)
 47. Kawai, H., Ogino, M., Shioya, R., Yoshimura, S.: Large-scale elast-plastic analysis using domain decomposition method optimized for multi-core CPU architecture. *Key Eng. Mater.* **462–463**, 605–610 (2011)
 48. Mandel, J.: Balancing domain decomposition. *Commun. Numer. Methods Eng.* **9**(3), 233–241 (1993)
 49. Earth Simulator (ES) (2014). <http://www.jamstec.go.jp/es/en/es1/index.html>
 50. Saad, Y.: *Iterative Methods for Sparse Linear Systems*, 2nd edn. SIAM, Philadelphia (2003)
 51. Hairer, E., Nørsett, S.P., Wanner, G.: *Solving Ordinary Differential Equations I: Non stiff Problems*, 2nd edn. Springer, Berlin (1993)
 52. Golub, G.H., Ye, Q.: Inexact conjugate gradient method with inner-outer iteration. *SIAM J. Sci. Comput.* **21**(4), 1305–1320 (1997)
 53. Winget, J.M., Hughes, T.J.R.: Solution algorithms for nonlinear transient heat conduction analysis employing element-by-element iterative strategies. *Comput. Methods Appl. Mech. Eng.* **52**, 711–815 (1985)
 54. Zienkiewicz, O.C., Taylor, R.L.: *The Finite Element Method for Solid and Structural Mechanics*, 6th edn. Elsevier, Amsterdam (2005)
 55. Ichimura, T., Hori, M., Bielak, J.: A Hybrid multiresolution meshing technique for finite element three-dimensional earthquake ground motion modeling in basins including topography. *Geophys. J. Int.* **177**, 1221–1232 (2009)
 56. METIS 5.1.0 (2014). <http://www.glaros.dtc.umn.edu/gkhome/metis/metis/overview>
 57. vSMP Foundation, ScaleMP Inc. (2014). <http://www.scalemp.com/products/vsmp-foundation/>
 58. Ajima, Y., Inoue, T., Hiramoto, S., Shimizu, T.: Tofu: interconnect for the K computer. *FUJITSU Sci. Tech. J.* **48**(3), 280–285 (2012)
 59. OpenMPI (2014). <http://www.open-mpi.org/>
 60. MPI: A Message-Passing Interface Standard, Version 2.1 (2014). <http://www.mpi-forum.org/docs/mpi21-report.pdf>
 61. Strong Ground Motion of the Southern Hyogo Prefecture Earthquake in 1995 Observed at Kobe JMA Observatory, Japan Meteorological Agency (2014). http://www.data.jma.go.jp/svd/eqev/data/kyoshin/jishin/hyogo_nanbu/dat/H1171931.csv
 62. Stampede at Texas Advanced Computing Center, The University of Texas at Austin (2014). <https://www.tacc.utexas.edu/resources/hpc/stampede-technical>
 63. 5m Mesh Digital Elevation Map, Tokyo Ward Area, Geospatial Information Authority of Japan (2014). <http://www.gsi.go.jp/MAP/CD-ROM/dem5m/index.htm>
 64. National Digital Soil Map, The Japanese Geotechnical Society (2014). <http://www.denshi-jiban.jp/>
 65. Strong-Motion Seismograph Networks (K-NET, KiK-net), National Research Institute for Earth Science and Disaster Prevention (2014). <http://www.kyoshin.bosai.go.jp/>
 66. Housner, G.W.: Spectrum intensities of strong-motion earthquakes. In: *Symposium on Earthquakes and Blast Effects on Structures*, Los Angeles, CA (1952)

67. Homma, S., Fujita, K., Ichimura, T., Hori, M., Citak, S., Hori, T.: A physics-based Monte Carlo earthquake disaster simulation accounting for uncertainty in building structure parameters. In: The International Conference on Computational Science **29**, 855–865 (2014). doi:10.1016/j.procs.2014.05.077
68. Ichimura, T., Agata, R., Hori, T., Hirahara, K., Hori, M.: Fast numerical simulation of crustal deformation using a three-dimensional high-fidelity model. *Geophys. J. Int.* **195**, 1730–1744 (2013)

High-Performance Computing for Structural Mechanics
and Earthquake/Tsunami Engineering

Yoshimura, S.; Hori, M.; Ohsaki, M. (Eds.)

2016, V, 199 p., Hardcover

ISBN: 978-3-319-21047-6