Physics-based urban earthquake simulation enhanced by 10.7 BlnDOF × 30 K time-step unstructured FE non-linear seismic wave simulation

Tsuyoshi Ichimura[1], Kohei Fujita[2,3], Seizo Tanaka[1], Muneo Hori[1,2]
Maddegedara Lalith[1] Yoshihisa Shizawa[4] and Hiroshi Kobayashi[4]

[1]Earthquake Research Institute and Department of Civil Engineering, The University of Tokyo,
[2]RIKEN Advanced Institute for Computational Science, [3]Japan Society for the Promotion of Science,
[4]Research Organization for Information Science and Technology

***Abstract*—With the aim of dramatically improving the reliability of urban earthquake response analyses, we developed an unstructured 3-D finite-element-based MPI-OpenMP hybrid seismic wave amplification simulation code, GAMERA. On the K computer, GAMERA was able to achieve a size-up efficiency of 87.1% up to the full K computer. Next, we applied GAMERA to a physics-based urban earthquake response analysis for Tokyo. Using 294,912 CPU cores of the K computer for 11 h, 32 min, we analyzed the 3-D non-linear ground motion of a 10.7 BlnDOF problem with 30 K time steps. Finally, we analyzed the stochastic response of 13,275 building structures in the domain considering uncertainty in structural parameters using 3 h, 56 min of 80,000 CPU cores of the K computer. Although a large amount of computer resources is needed presently, such analyses can change the quality of disaster estimations and are expected to become standard in the future.**

***Keywords*—Time-to-solution, scalability, urban earthquake simulation, unstructured FEM**

I. Introduction

A. Overview and significance of the problem

A large-scale earthquake is a menace to modern civilization, because it can literally destroy a poorly prepared urban area. A recent harrowing example is the 2011 Tohoku Earthquake, which has produced long-lasting damage to the eastern part of Japan. An earthquake of a similar scale is likely to hit one or more 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 the spatial distribution of possible earthquake damage. High spatial resolution is essential for such an estimate because the crustal-scale phenomenon of an earthquake results in damage to buildings and structures. A physics-based urban area earthquake simulation of the phenomenon is a unique solution to realizing such a reliable estimate.

B. State of the art

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; and 3) buildings and structures are shaken by the resulting ground motion; see Fig. 1. The *physics* of these processes are quite simple because they can be described completely as solutions of linear or non-linear solid wave equations. However, the physics has largely been ignored because solving the wave equation requires a tremendous amount of computation. Empirical equations such as attenuation relations and vulnerability curves is usually used for earthquake disaster estimations [1], [2], [3].

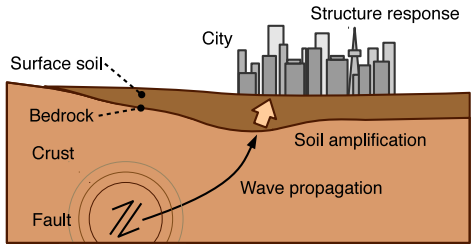


Fig. 1. Schematic view of earthquake disaster process.

Despite the simple physics of the three processes, constructing a model for analysis is not an easy task. Modern observation technologies in the earth sciences field are addressing this problem. Physics-based seismic wave propagation simulation has become a 'hot' topic since an accurate model was constructed for a fault and a shallow crust. This simulation required a large amount of computation, and some achievements using supercomputers have been presented at the Supercomputing Conferences (SCs); for example, see [4], [5], [6], [7].

The physics-based seismic response simulation of man-made structures, which is directly related to any estimate of an earthquake disaster, is also being realized [8], [9]; an analysis model is automatically constructed for every building and structure using urban area digital information (a core data source is the Google system), and an urban area seismic response is then computed as an assembly of these responses. A good example is the Tokyo simulation [10], in which non-linear responses of 253,405 buildings were computed for 100 earthquake scenarios using 16,000 CPU cores of the K computer for 4,088 s.

A seismic wave amplification simulation is the last piece in the physics-based urban area earthquake simulation; this simulation connects the seismic wave propagation simulation in the crust to the seismic response simulation of buildings and structures. Seismic wave amplification processes are complex because they are affected by non-linear material properties of the surface soil as well as the topographical effects of the surface and ground layer configuration. The degree of the

978-1-4799-5500-8/14 $31.00 © 2014 IEEE
DOI 10.1109/SC.2014.7

amplification changes from place to place, and it sometimes increases by as much as 10 times for particular components of a seismic wave, a condition that is likely fatal for structures [11]. The computation cost required for solving a non-linear wave equation in a domain of irregular geometry is much larger than that of the wave propagation simulations presented at past SCs. This is because 1) non-linearity 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 a physics-based seismic wave amplification simulation is a challenge even from the viewpoint of computer science. A target domain is $2,000 \times 2,000 \times 100$ 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 finite-element method (FEM) must be developed for this simulation, and a sophisticated model construction method is needed.

We should mention the current state of physics-based seismic wave amplification simulation. One trial that used a voxel FEM [12], [13] used 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 non-linear soil constitutive models. Another trial [14] 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 paper, we present the most updated trial aimed at realizing a physics-based seismic wave amplification simulation. In Section 2, we explain the development of a FEM code that is able to analyze a non-linear model of 10 BlnDOF for a 30 K time step and the development of an unstructured element model for the irregularly configured domain. In Section 3, we show the computation performance of the developed code on the K computer [15]. We also examine the performance on an Intel CPU machine to assess the portability of the code developed. In Section 4, we show an example of a physics-based seismic wave amplification simulation. A 10.7 BlnDOF model of Tokyo was constructed and the seismic wave amplification process was computed for a 30 K time step. The results of the simulation were then inputted to the physics-based seismic structural response simulation of 13,275 buildings. Concluding remarks are provided in Section 5.

II. Key ideas and Innovations

A. Target Equation

Ground structures are modeled as layered media, which often have complicated geometries because the thickness changes on the order of $10^{-1\sim1}$m in a $10^{0\sim1}$-m region. FEM with unstructured low-order solid elements is suitable for analyzing the response of such ground structures and satisfies traction-free boundary conditions. Here, we use a non-linear dynamic FEM with 3-D unstructured low-order solid elements for the physics-based seismic wave amplification simulation. The equations to be solved are as follows:

$$\begin{aligned}(\frac{4}{dt^2}\mathbf{M}+\frac{2}{dt}\mathbf{C}^n+\mathbf{K}^n)\ \delta\mathbf{u}^n = \\ \mathbf{F}^n-\mathbf{Q}^{n-1}+\mathbf{C}^n\mathbf{v}^{n-1}+\mathbf{M}(\mathbf{a}^{n-1}+\frac{4}{dt}\mathbf{v}^{n-1}),\end{aligned} \tag{1}$$

with

$$\begin{cases}\mathbf{Q}^n=\mathbf{Q}^{n-1}+\mathbf{K}^n\delta\mathbf{u}^n,\\ \mathbf{u}^n=\mathbf{u}^{n-1}+\delta\mathbf{u}^n,\\ \mathbf{v}^n=-\mathbf{v}^{n-1}+\frac{2}{dt}\delta\mathbf{u}^n,\\ \mathbf{a}^n=-\mathbf{a}^{n-1}-\frac{4}{dt}\mathbf{v}^{n-1}+\frac{4}{dt^2}\delta\mathbf{u}^n,\end{cases} \tag{2}$$

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, $\mathbf{M}$ is the consistent mass matrix, $\mathbf{C}$ is the damping matrix, $\mathbf{K}$ is the stiffness matrix, dt is the time increment, and n is the time step. We used Rayleigh damping, where the element damping matrix $\mathbf{C}_e^n$ was 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., $minimize[\int_{f_{min}}^{f_{max}}(h^n-\frac{1}{2}(\frac{\alpha}{2\pi f}+2\pi f\beta))^2df]$, where f_{max} and f_{min} are the maximum and minimum 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, and therefore, satisfying the Courant condition when using explicit time-integration methods leads to small time increments and considerable computational expense. To resolve this, we used the Newmark-β method for time integration (with β=1/4, δ=1/2). Because complicated non-linear constitutive relations are often used in non-linear ground motion analysis and the implicit scheme sometimes cannot reach an accurate solution, an explicit scheme was selected in this paper to compute the response robustly. Semi-infinite absorbing boundary conditions were used for the bottom and side boundaries of the simulation domain. In this paper, the modified Ramberg-Osgood model [16] and Masing rule [17] were used for the non-linear constitutive relations describing the soil. The unknowns in Eq. (1) 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: 1) Read boundary conditions; 2) 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; 3) Compute $\mathbf{K}^n$ and $\mathbf{C}^n$ using the stiffness and damping ratio for the n-th time step; 4) Compute $\delta\mathbf{u}^n$ by solving Eq. (1), and update the values in Eq. (2) using $\delta\mathbf{u}^n$; 5) Output results; Because most of the computation cost in this procedure is due to solving Eq. (1), the key point for achieving efficient analysis is the development of a suitable solver, considering the characteristics of the non-linear dynamic finite-element method. A method for constructing a FEM model with a large number of unstructured low-order elements was also developed to model the complex ground structure accurately.

B. Key ideas and Innovations in the Solver

In this physics-based seismic wave amplification simulation, we 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. From the point of 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 developed a fast, iterative solver, considering the characteristics of the matrix equation. There has been much research on fast, iterative solvers for solving responses of solid FEMs with unstructured low-order elements, and some research has been reported at SCs (e.g., [18]). These researchers sought to reduce the computation time using a preconditioner with a high computation cost to improve the poor convergency of the target problem. Others have reported similar research (e.g., [19], [20]). For example, [19] presents a fast solver based on a balancing domain decomposition method [21]-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 [22]. Although a sophisticated preconditioner with smart program tuning can reduce the computation cost and resolve the poor convergency of the target problem, our problem needed a much faster solver. Although the non-linear dynamic finite-element method 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 convergency of the current matrix equation is worse than the convergency of a matrix equation coming from an 'ordinary' problem with lower resolution, even though its DOF is same. Thus, we need a fast iterative solver that has a smart preconditioner with high scalability under a many-compute-nodes environment with lower computation cost. 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 can be seen that a moderate preconditioner with less computation cost and high scalability in a many-compute-nodes environment is more suitable than is a smart preconditioner with higher computation cost and only moderate scalability. Although one candidate for such a solver is the 'preconditioned conjugate gradient' (PCG) method [23] with the block Jacobi method, the number of iterations tend to increase when solving matrices with poor characteristics. Indeed, we examined this in our numerical experiments, and the number of iterations often increased markedly. So, because the convergency rate of PCG is slow, development of a much faster iterative solver for our problem was very desirable.

Thus, we developed the code for a huge DOF and large time-step, physics-based seismic wave amplification simulation on a huge number of scalar computers based on a fast and scalable iterative solver using a moderate preconditioner with lower computation cost. We must pay attention to the amount of computation in seeking to achieve fast computation, and we must also pay attention to reducing the frequency of communication and the amount of synchronization necessary between compute nodes. Considering the characteristics of a non-linear dynamic finite-element method, we developed the code with the following key ideas a)∼e). The procedure for computing $\mathbf{u}^n$ is shown in Algorithm 1 and 2.

a) Predictor: The initial solution of the iterative solver was 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 [24]. 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;

b) Adaptive conjugate gradient method [25]: 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;

c) Multigrid (MG) method [23]: FEMmodel_c is set as though it is equivalent to the FEMmodel, but its resolution is coarser. In the inner loop, a rough solution is first computed using FEMmodel_c, and then, using this solution as an initial solution, a rough solution is computed using the FEMmodel with fewer iterations. Although additional computation related to FEMmodel_c is required, the computation cost and the communication cost are much smaller than using only the FEMmodel because the DOF and nodes on the interfaces of the FEMmodel_c are much smaller than those for the FEMmodel. As a result, the computation cost of the inner loop is reduced by the MG method;

d) Mixed-precision (MP) arithmetic: Although double-precision arithmetic is needed in the outer loop, single-precision arithmetic can be used for the inner loops as only a rough estimation is needed for the preconditioner. With the DOF of FEMmodel_c being smaller than that of the FEMmodel, and halving the required memory size by using single precision, we can store the matrix of FEMmodel_c in the CRS format. Also, a reduction in communication size can be expected; the size of adjacent node communication for a FEMmodel with single precision is reduced to half that with double precision, and the size of adjacent node communication for a FEMmodel_c with single precision is reduced to 1/8 of that for a FEMmodel with double precision. Thus, faster computation and memory access can also be expected;

e) Element-by-element (EBE) method [26]: It is difficult to store $\mathbf{K}$ of FEMmodel for solving Eq. (1) because the DOF computed by one compute node is large. Instead of storing $\mathbf{K}$, computation of $\mathbf{Kx}$ is computed by the EBE method. In this paper, EBE was applied to compute the second-order tetrahedral elements of the FEMmodel. With the advantage of displacement-stress mixed FEM [27] and the orthogonal-basis function, we successfully transformed $\mathbf{Kx}$ into $\sum_i \mathbf{N}_{\mathbf{b}i}{}^T(\mathbf{D}(\mathbf{N}_{\mathbf{b}i}\mathbf{x}))$, where $\mathbf{D}$ and $\mathbf{N}_{\mathbf{b}i}$ are the elastic tensor and equivalent mode, respectively. Sophisticated reformulation and careful tuning can achieve a further reduction in the number of operations and 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

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

read boundary condition of n-th time step
$\mathbf{f} \Leftarrow \mathbf{F}^n - \mathbf{Q}^{n-1} + \mathbf{C}^n\mathbf{v}^{n-1} + \mathbf{M}(\mathbf{a}^{n-1} + \frac{4}{dt}\mathbf{v}^{n-1})$
$\mathbf{x} \Leftarrow \frac{dt}{24}(-9\mathbf{v}^{n-4} + 37\mathbf{v}^{n-3} - 59\mathbf{v}^{n-2} + 55\mathbf{v}^{n-1})$
$\bar{\mathbf{B}} \Leftarrow diag[\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n]$
$\bar{\mathbf{B}}_c \Leftarrow diag[\frac{4}{dt^2}\mathbf{M}_c + \frac{2}{dt}\mathbf{C}_c^n + \mathbf{K}_c^n]$
$\bar{\mathbf{A}}_c \Leftarrow \frac{4}{dt^2}\mathbf{M}_c + \frac{2}{dt}\mathbf{C}_c^n + \mathbf{K}_c^n$ (stored on memory with CRS format)
$\mathbf{r} \Leftarrow \mathbf{f} - (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n)\mathbf{x}$
$\beta \Leftarrow 0$
$i \Leftarrow 1$
(*outer loop start*)
while $\|\mathbf{r}\|_2/\|\mathbf{f}\|_2 \geq \epsilon$ **do**
  (*inner loop start*)
  $\bar{\mathbf{r}} \Leftarrow \mathbf{r}$
  $\bar{\mathbf{z}} \Leftarrow \mathbf{B}^{-1}\mathbf{r}$
  $\bar{\mathbf{r}}_c \Leftarrow \bar{\mathbf{P}}^T\bar{\mathbf{r}}$
  $\bar{\mathbf{z}}_c \Leftarrow \bar{\mathbf{P}}^T\bar{\mathbf{z}}$
  $\bar{\mathbf{z}}_c \Leftarrow \bar{\mathbf{A}}_c^{-1}\bar{\mathbf{r}}_c$ (*solved on FEMmodel$_c$ by algorithm 2 with ϵ_c^{in} and initial solution $\bar{\mathbf{z}}_c$*)
  $\bar{\mathbf{z}} \Leftarrow \bar{\mathbf{P}}\bar{\mathbf{z}}_c$
  $\bar{\mathbf{z}} \Leftarrow (\frac{4}{dt^2}\bar{\mathbf{M}} + \frac{2}{dt}\bar{\mathbf{C}}^n + \bar{\mathbf{K}}^n)^{-1}\bar{\mathbf{r}}$ (*solved on FEMmodel by algorithm 2 with ϵ^{in} and initial solution $\bar{\mathbf{z}}$*)
  $\mathbf{z} \Leftarrow \bar{\mathbf{z}}$
  (*inner loop end*)
  if $i > 1$ **then**
    $\beta \Leftarrow (\mathbf{z}, \mathbf{q})/\rho$
  end if
  $\mathbf{p} \Leftarrow \mathbf{z} + \beta\mathbf{p}$
  $\mathbf{q} \Leftarrow (\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C}^n + \mathbf{K}^n)\mathbf{p}$
  $\rho \Leftarrow (\mathbf{z}, \mathbf{r})$
  $\alpha \Leftarrow \rho/(\mathbf{p}, \mathbf{q})$
  $\mathbf{q} \Leftarrow -\alpha\mathbf{q}$
  $\mathbf{r} \Leftarrow \mathbf{r} + \mathbf{q}$
  $\mathbf{x} \Leftarrow \mathbf{x} + \alpha\mathbf{p}$
  $i \Leftarrow i + 1$
end while
(*outer loop end*)
$\delta\mathbf{u}^n \Leftarrow \mathbf{x}$
$\mathbf{Q}^n \Leftarrow \mathbf{Q}^{n-1} + \mathbf{K}^n\delta\mathbf{u}^n$
$\mathbf{u}^n \Leftarrow \mathbf{u}^{n-1} + \delta\mathbf{u}^n$
$\mathbf{v}^n \Leftarrow -\mathbf{v}^{n-1} + \frac{2}{dt}\delta\mathbf{u}^n$
$\mathbf{a}^n \Leftarrow -\mathbf{a}^{n-1} - \frac{4}{dt}\mathbf{v}^{n-1} + \frac{4}{dt^2}\delta\mathbf{u}^n$
output results of n-th time step
modify material properties considering nonlinearity

is very high, this speed-up can reduce computation time significantly;

Hereafter, we call this code 'GAMERA'[1] (the physics-based seismic wave amplification simulator, enhanced by a multiGrid method, Adaptive conjugate gradient method, Mixed precision arithmetic, Element-by-element 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

[1] an imaginary Japanese Earth guardian, supported by prayers for peace.

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
$\mathbf{z} \Leftarrow \mathbf{x}$

cost of the preconditioner itself is also reduced. As a result, the total computation time of GAMERA can be reduced. In actual computations, we compute $\mathbf{u}^n$ with FEMmodel^i and FEMmodel$_c{}^i$ using Algorithm 1 and 2 by the i-th MPI process; FEMmodel^i and FEMmodel$_c{}^i$ are submodels of FEMmodel and FEMmodel$_c$, as shown in section II-C. 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 each FEMmodel^i and FEMmodel$_c{}^i$ is computed by OpenMP parallelization, the whole process of GAMERA is computed by MPI-OpenMP hybrid parallelization.

C. Key ideas and Innovations in Model Construction

We present the construction method of the 10 Bln-order-DOF 3-D FEM model for GAMERA using 3-D layered ground structure data. 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 computational cost of constructing such a large 3-D FEM model is expected to be considerable, and thus a fully automated meshing algorithm that can generate elements of high quality is desirable. Although it is common to perform manual tuning to avoid elements with poor quality, doing so with large models can become time consuming. For this reason, various methods have been developed for generating FEM models automatically. We used the method of [28], which was modified as the huge-DOF 3-D FEM model can be constructed. 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

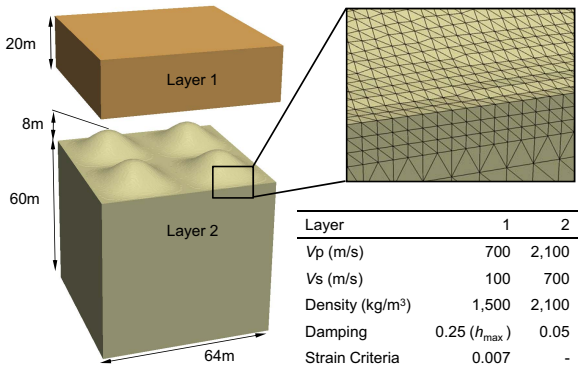
the cubic elements to linear-tetrahedral elements. Next, the linear-tetrahedral elements are converted in to second-order tetrahedral elements. We use the model with linear-tetrahedral elements as the FEMmodel$_c$ and the model with second-order tetrahedral elements as the FEMmodel. We divide FEMmodel and FEMmodel$_c$ so that the number of elements in each domain becomes uniform, using METIS 5.1.0 [29] with suitable options on a shared-memory computer. These i-th divided models are defined as FEMmodel^i and FEMmodel$_c{}^i$, respectively. Although there has been research related to the efficient generation of FEMmodel^i on distributed memory computers, we used this generation method because load imbalance for FEMmodel^i and FEMmodel$_c{}^i$ can be reduced significantly, even for a complicated ground structure model. Such reduced load imbalance contributes to high scalability with Algorithm 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.

Our FEM models were 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 [30]. For the generation of the 27 BlnDOF, 6.7 Bln-element FEM model used in Section III, most of the cost was spent on the generation of the FEMmodel$_c$, conversion of FEMmodel$_c$ to FEMmodel, and division into FEMmodel^i. Generation of FEMmodel$_c$ took 3 h 43 min with 160 OpenMP threads and 698 GB computer memory, conversion of FEMmodel$_c$ to FEMmodel took 4 h 53 min with 1,455 GB computer memory, and division into FEMmodel^i with 82,944 divisions took 13 h 59 min with 1,708 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 FEMmodel and FEMmodel$_c$. The throughput of the file system used (NL-SAS, 7200 rpm HDD) is 100-200MB/s using the HDF5 library, while the total file size of the models is 1,649 GB. It is expected that construction time could be reduced by using faster file systems with parallel I/O. However, because we conduct many non-linear 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 a FEM model with less load imbalance is much more desirable, even if the construction time is relatively long.

III. Performance measurement

A. System and environment where performance was measured

In this study, we measured the performance of GAMERA on the K computer, a massively parallel supercomputer at RIKEN, Advanced Institute for Computational Science [15]. The K computer consists of 82,944 compute nodes, each with single SPARC64$^{\mathrm{TM}}$ 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 FLOP $\times$ 2 GHz $\times$ 8 CPU cores $=$ 128 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 [31]. Each node can communicate in four directions simultaneously, with 5 GB/s throughput in each direction. An OpenMPI 1.4.3 [32]-based MPI library optimized for the Tofu network was used, following the MPI 2.1 standard [33].



Layer	1	2
Vp (m/s)	700	2,100
Vs (m/s)	100	700
Density (kg/m^3)	1,500	2,100
Damping	0.25 ($h_{\max}$)	0.05
Strain Criteria	0.007	-

Fig. 2. 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.

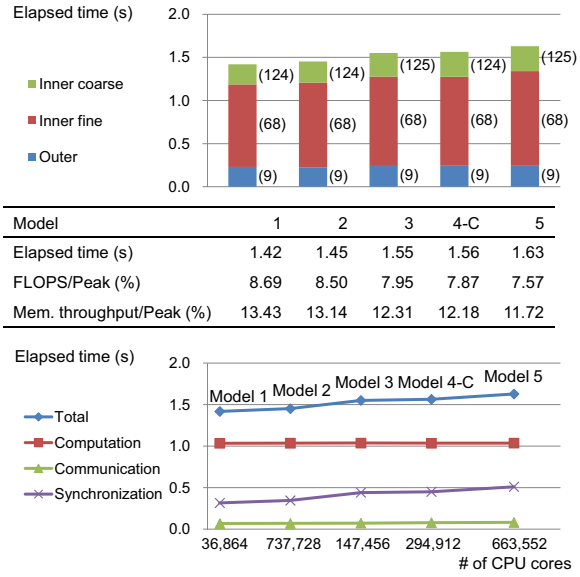
B. Problem setting

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 multigrid preconditioner disabled. PCGE is GAMERA whose solver is replaced with PCG using the 3 $\times$ 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 $\epsilon = 10^{-8}$, $\epsilon^{in} = 0.25$, $\epsilon_c^{in} = 0.5$ for the tolerance of the linear solvers. We use one compute node (one MPI process) for each FEMmodel^i and FEMmodel$_c{}^i$ 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. 2, the first layer is a soft layer that behaves non-linearly 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 a 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 non-linearity. 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 [34]. Table I shows the model sizes and number of partitions used

TABLE I. CONFIGURATION OF MODELS USED FOR PERFORMANCE MEASUREMENT. σ_n, σ_e INDICATES THE MAXIMUM IMBALANCE OF THE NUMBER OF NODES AND ELEMENTS. $\sigma_n = (\max n - \text{mean } n)/\text{mean } n$, $\sigma_e = (\max e - \text{mean } e)/\text{mean } e$.

Model	# of MPI domains	# of CPU cores	Total DOF	Mean DOF per MPI domain	σ_n (%)	σ_e (%)	# of PCGE iterations
1	4,608	36,864	1,505,755,135	326,767	4.13	0.86	355
2	9,216	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	9,216	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



Model	1	2	3	4-C	5
Elapsed time (s)	1.42	1.45	1.55	1.56	1.63
FLOPS/Peak (%)	8.69	8.50	7.95	7.87	7.57
Mem. throughput/Peak (%)	13.43	13.14	12.31	12.18	11.72

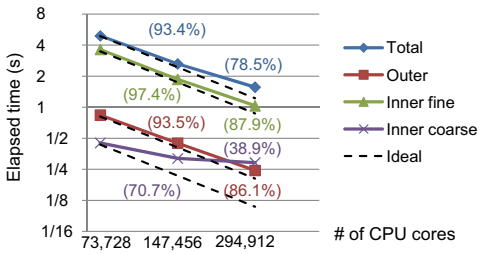
Fig. 3. Elapsed time and its breakdown (inner fine, inner coarse, outer loop) for the ninth time step of the weak-scaling problem set. Numbers in brackets indicate the number of elapsed loops required for convergence of the linear solver. The breakdown of the elapsed time for computation, communication, and synchronization is also shown.

for measuring size-up efficiency (weak scaling) and speed-up efficiency (strong scaling).

C. Performance results

Although GAMERA is best suited for systems with a fast single-precision floating-point computing ability, we measured performance using the K computer, as the time-to-solution for solving large-scale problems is important, and thus the scalability for solving such large-scale problems becomes important. We measure performance using the ninth time step because it was not influenced by the fluctuations in the initial part of the wave and became a poor convergency problem for all of the models. 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 could see that the number of loops elapsed for convergence of the PCGE solver was the same for all of the models (see Table I), and thus we could see that this set of models could be used for measuring the size-up efficiency of the codes. We first fixed the problem size per compute node to 326 K DOF and increased the problem size and number of CPU cores to measure the size-up efficiency of the code. Fig. 3 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 loops elapsed for convergence of the linear solver. We can see that the numbers of loops elapsed were almost the same for all the models; thus, the model set is suitable for measuring size-up efficiency of the code. Looking at the elapsed time, we can see that the increase in elapsed time is small and that the code scales well to a large number of CPU cores. The smallest model (model 1, 1.5 BlnDOF) was computed in 1.42 s using 36,864 CPU cores (4,608 compute nodes × 8 CPU cores, 8.69% of peak performance, 51.3 TFLOPS), and the largest model (model 5, 27 BlnDOF) was computed in 1.63 s using the whole K computer, with 663,552 CPU cores (82,944 compute nodes × 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 elapsed time in terms of computation, communication, and synchronization[2]. Although 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, the increase in communication time was, in fact, only 0.081 s (model 5) - 0.067 s (model 1) = 0.014 s. Such a small increase in communication time is due to the decrease in communication size for peer-to-peer communication and hardware enhancements in the K computer's interconnect for all-reduce operations [31]. However, we can see that synchronization time has increased from 0.316 s (model 1) to 0.511 s (model 5). Looking at the imbalance of the models shown in Table I, we can see that the imbalance in the number of elements and nodes increased with respect to model size. The amount of EBE and CRS computation 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

[2]Defined as:
communication = AVG{MPI_Isend + MPI_Irecv + MPI_Allreduce(total) - MPI_Allreduce(wait)} + MIN{MPI_Waitall},
synchronization = AVG{MPI_Allreduce(wait) + MPI_Waitall} - MIN{MPI_Waitall},
computation = AVG{total} - communication - synchronization.
AVG and MIN indicates the average and minimum values of all the compute nodes, respectively. MPI_Barrier is not used.



Model	4-A	4-B	4-C
Elapsed time (s)	4.92	2.63	1.56
FLOPS/Peak (%)	9.99	9.36	7.87
Mem. Throughput /Peak (%)	19.14	16.05	12.18

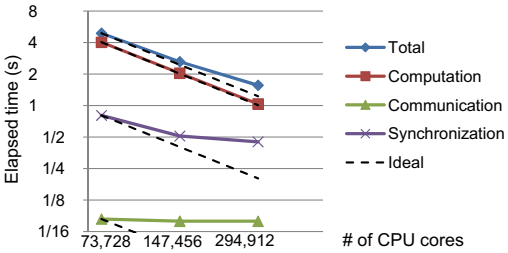
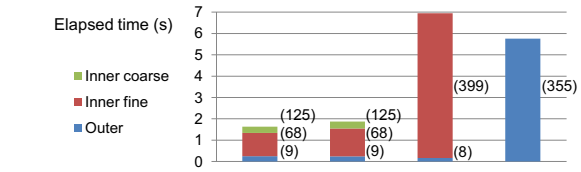


Fig. 4. Elapsed time and its breakdown (inner fine, inner coarse, outer loop) for the ninth time step of the strong-scaling problem set. Numbers in brackets show the speed-up efficiency with respect to Model 4-A (73,728 CPU cores). The breakdown of the elapsed time for computation, communication, and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

elements by tuning the parameters of METIS.

Next, we measure the speed-up efficiency of the code. Fig. 4 shows the change in elapsed time with respect to the change in the number of CPU cores. We can see that the 12 BlnDOF is solved in 4.92 s using 73,728 CPU cores (9,216 compute nodes × 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 × 8 CPU cores, 371.4 TFLOPS, 7.87% of peak), with a relatively good speed-up efficiency of 78.5%. Looking at the breakdown of elapsed time, we can see 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); it is the primary reason for the deterioration in the scalability of the whole program. As was 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. Next, we show the breakdown of 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.

Fig. 5 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 × 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.



	GAMERA (MP,MG)	GAMERA (DP,MG)	GAMERA (DP,SG)	PCGE
Elapsed time (s)	1.63	1.87	6.95	5.75
FLOPS/Peak (%)	7.57	6.60	7.88	8.11
Mem. Throughput/Peak (%)	11.72	19.40	20.40	21.66

Fig. 5. 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.

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 $\mathrm{FEMmodel_c}^i$ (line 17 of Algorithm 1) uses linear elements with a CRS format, whereas the computation of $\mathrm{FEmodel}^i$ (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 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 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.

All of the measurements above concerned the cost of the linear solver; we show the total cost of the 100 time steps in Fig. 6. 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

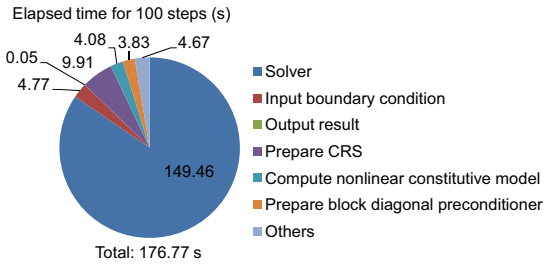
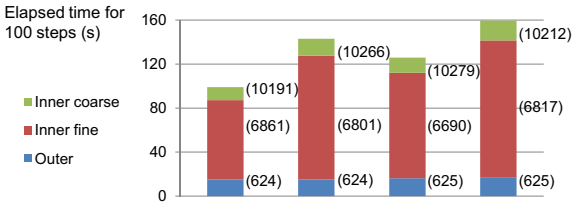
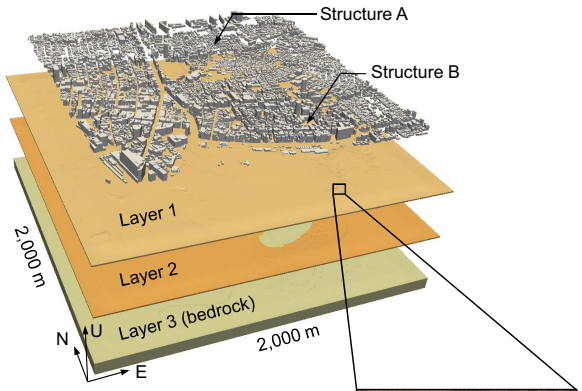


Fig. 6. Breakdown of the elapsed time used for analyzing 100 steps of Model 5 (27 BlnDOF, 663,552 CPU cores). 0.736 PFLOPS (6.93% of peak) was attained for the whole program.



Compute environment	Stampede 256 nodes (each with dual Intel Xeon E5-2680 CPUs)		K computer 256 nodes (each with single SPARC64VIIIfx CPU)	
Solver	GAMERA (MP,MG)	GAMERA (DP,MG)	GAMERA (MP,MG)	GAMERA (DP,MG)
Elapsed time (s)	99.0	142.9	125.9	159.5
Ratio between (MP,MG) and (DP,MG)	1.44	-	1.27	-

Fig. 7. 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.



Layer	1	2	3
Vp (m/s)	1,210	1,380	1,770
Vs (m/s)	150	255	490
Density (kg/m^3)	1,500	1,800	1,900
Damping	0.25 (h_{max})	0.05	0.005
Strain Criteria	0.005	-	-

Fig. 8. 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.

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. Fig. 7 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 [35], 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 I. 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 compute 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.

In the Appendix, we report performance of GAMERA$^{\mathrm{EBE4}}$, 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$^{\mathrm{EBE4}}$ which are suitable for different types of hardware in the future.

IV. Application Example

As an application example, GAMERA was applied to a physics-based urban earthquake simulation of a 2.0 × 2.0 km region of central Tokyo. The region consists of businesses, shops, and densely-built residential districts, with a total of 13,275 buildings. We targeted the ground structure, which had a rectangular shape of 2.0 × 2.0 km and a depth of 100 m. The ground structure was modeled with three soil layers using the 5-m grid elevation map [36] and digital geotechnical database [37]. Fig. 8 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 [38]. The Kobe earthquake wave [34] with $dt = 0.001$ s and 30,000 time steps was inputted into the bottom of the model. This simulation was carried out on the K computer using 294,912 CPU cores (36,864 compute nodes × 8 CPU cores). Relative errors of $\epsilon = 10^{-8}$, $\epsilon^{in} = 0.25$, $\epsilon_c^{in} = 0.5$ were used for the tolerances

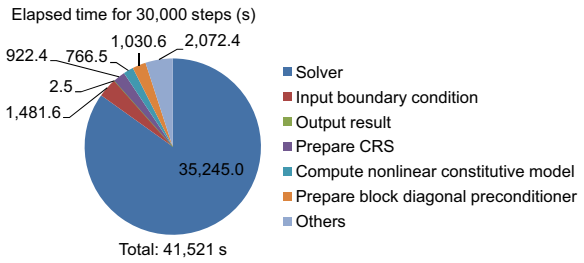


Fig. 9. Computation performance of a 10.7 BlnDOF ground model. For the first 1,000 time steps, the performance of the linear solver was 0.421 PFLOPS (8.93% of peak), and the performance of the whole program was 0.358 PFLOPS (7.59% of peak).

of the linear solvers. The FEM model was constructed with an element size small enough to obtain convergence in ground acceleration, even when the soils softened due to non-linearity. This resulted in a model with 10.7 BlnDOF and a minimum element size of 0.66 m.

Fig. 9 shows the breakdown of the elapsed time of the simulation. It took 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 time was spent solving the linear equation system (35,245 s). Thus, we can see that adequate speed-up of the solver enabled such a large-scale analysis. The performance of the whole program was also high due to the reduction in I/O cost (1,481.6 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 1,000 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.

Fig. 10(i) shows the response of ground at the surface as well as the response of structures using the computed ground motion as an input. From Fig. 10(ii), we can see a complex ground motion response and the structures shaking according to the ground motion. In typical ground motion analyses, it is difficult to ensure the convergence of acceleration due to the fatal computational costs, and thus the reliability of results is not necessarily high. However, we can obtain highly reliable results that converge in terms of acceleration response for large domains using GAMERA. Here, we used three models with different resolutions: model I with the highest resolution (10.7 BlnDOF model), model II with 1.5 times the element size of model I, and model III with 3.0 times the element size of model I. Fig. 10(iii) shows the comparison of acceleration time history at the surface (Point A). We can see 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 the responses of structures are dependent on the input acceleration at the ground surface, the responses of structures also converge with the increase in resolution; see Fig. 10(iv). From these results, we can see that high-resolution finite-element modeling of ground motion is essential for analyzing the response of structures. By comparing the ground and structure responses in Fig. 10(i), we can see that larger SI values (an index commonly used for evaluating the destructiveness of waves to buildings [39]) does not necessarily lead to large structural responses. Such a complex distribution of structure responses 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 a city [40]. Fig. 11 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. 11, 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 buildings $\times$ 10,000 samples shown here took 3 h, 56 min using 80,000 CPU cores (10,000 compute node $\times$ 8 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.

V. Concluding Remarks

In this paper, GAMERA was developed as an efficient code for a non-linear dynamic finite-element method for physics-based seismic wave amplification simulation. 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 finite-element method 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 mention here 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

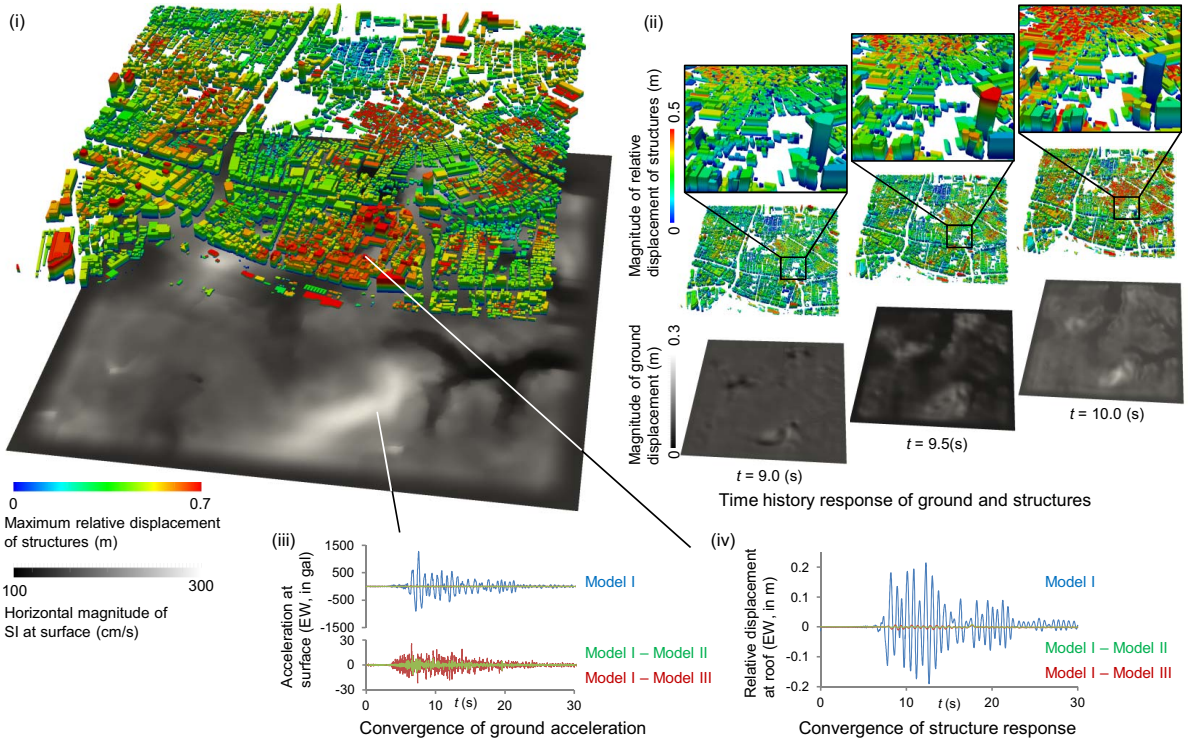


Fig. 10. Response of ground and structures. (i) Horizontal magnitude of SI at surface and maximum relative displacement of structures. (ii) Snapshots of time history response. (iii) Acceleration waveforms at surface (Point A) obtained from Model I (10.7 BlnDOF model), Model II (with element sizes 1.5 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. (iv) 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.

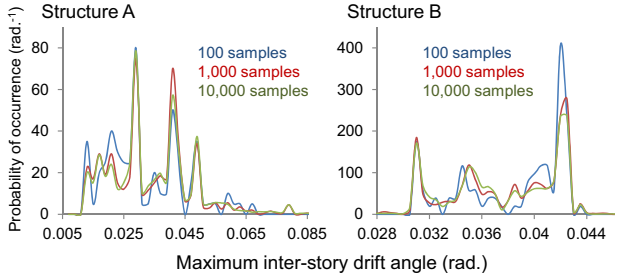


Fig. 11. Probability density of response of two two-story RC structures with stochastic structural parameters. The horizontal axis indicates the maximum inter-story drift angle (an index frequently used to evaluate damage of structures), and the vertical axis indicates the probability of occurrence. We can see that the distribution converges by increasing the number of samples from 100 to 1,000 and then to 10,000 samples.

Intel CPU was observed for a small-scale crust deformation problem [41] 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 finite-element method 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.

GAMERA is the last piece of the physics-based urban area earthquake simulation to make reliable estimates of earthquake hazards and disasters. There is no doubt that the estimates made by the physics-based simulation are scientifically rational compared with current empirical equations. The world should not experience earthquake damage to high rise buildings and dense transportation networks. The tragedy of the nuclear power plant accident in the 2011 Tohoku Earthquake should not be repeated in modern urban areas. The fast and large-scale computing capabilities of GAMERA can contribute to mitigating an as-yet unexperienced earthquake disaster by performing physics-based simulations that take full advantage of the power of modern supercomputers.

APPENDIX

We explain GAMERA$^{\mathrm{EBE4}}$, 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$^{\mathrm{EBE4}}$. 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.

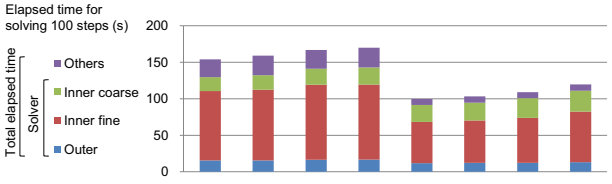
Fig. 12 shows the size-up efficiency of GAMERA and GAMERA$^{\mathrm{EBE4}}$. As expected, the inner coarse loop is slightly faster using CRS, but the inner coarse loop using EBE takes less time than the sum of preparation of CRS matrices (included in "Others") and the inner coarse loop. Compared with 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 s to 23.2 s for model 1. Combined with the improvement in OpenMP implementations of the inner fine loop, the 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$^{\mathrm{EBE4}}$ 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.

Fig. 13 shows the speed-up efficiency of the solvers of GAMERA and GAMERA$^{\mathrm{EBE4}}$. 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$^{\mathrm{EBE4}}$), 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 (64GB/s / 128GFLOPS = 0.5bytes/FLOP), which leads to fast access to memory. By changing from CRS to EBE, the total memory usage decreases from 1214.9 MB to 982.8 MB for model 4-A and 584.3 MB 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$^{\mathrm{EBE4}}$, 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.

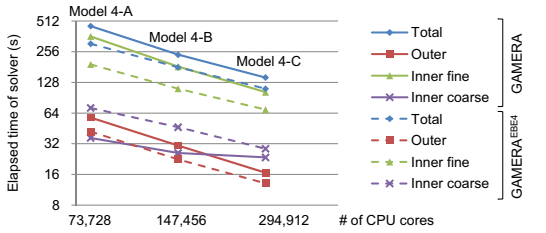
ACKNOWLEDGMENTS

We gratefully acknowledge the anonymous reviewers. Results are obtained using the K computer at the RIKEN Advanced Institute for Computational Science (Proposal number hp120308). We used the National Digital Soil Map provided by the Japanese Geotechnical Society. We acknowledge the support from Japan Society for the Promotion of Science (24-8183). We thank Prof. Westerink at University of Notre Dame for providing computer resources.



Method	GAMERA				GAMERA$^{\mathrm{EBE4}}$			
Model	1	2	3	4-C	1	2	3	4-C
Total elapsed time (s)	158.9	159.0	166.8	169.8	99.6	103.2	109.0	119.0
Elapsed time for solver (s)	129.6	132.2	141.2	143.0	91.5	94.7	100.8	111.0
FLOPS/Peak of solver (%)	9.00	8.79	8.25	8.19	14.49	14.06	13.28	11.82

Fig. 12. Size-up efficiency of GAMERA and GAMERA$^{\mathrm{EBE4}}$. GAMERA$^{\mathrm{EBE4}}$ 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.



Method	GAMERA			GAMERA$^{\mathrm{EBE4}}$		
Model	4-A	4-B	4-C	4-A	4-B	4-C
Elapsed time of solver (s)	456.6	239.4	143.0	306.3	180.0	111.0
FLOPS/Peak (%)	10.3	9.65	8.19	17.1	14.7	11.8

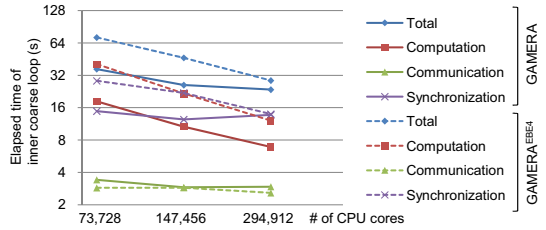


Fig. 13. Speed-up efficiency of GAMERA and GAMERA$^{\mathrm{EBE4}}$.

REFERENCES

[1] P. Somerville, N. Collins, N. Abrahamson, R. Graves, and C. Saikia, "GROUND MOTION ATTENUATION RELATIONS FOR THE CENTRAL AND EASTERN UNITED STATES," Final Report (June 30, 2001). [Online]. http://earthquake.usgs.gov/hazards/products/conterminous/2008/99HQGR0098.pdf

[2] Second report of the Nankai Trough Large Earthquake Model Committee, Cabinet Office, Government of Japan (August 28,2012). [Online]. http://www.bousai.go.jp/jishin/nankai/model/index.html

[3] Disaster assessment of Tokyo due to large earthquakes such as the Nankai Trough Eearthquake, Tokyo Metropolitan Government (May 14, 2013). [Online]. http://www.bousai.metro.tokyo.jp/taisaku/1000902/1000402.html

[4] T. Tiankai, Y. Hongfeng, L. Ramirez-Guzman, J. Bielak, O. Ghattas, M. Kwan-Liu, and D. R. O'Hallaron, "From mesh generation to scientific visualization: an end-to-end approach to parallel supercomputing," Proceedings of the 2006 ACM/IEEE conference on Supercomputing (SC '06). ACM, New York, NY, USA, 2006, Article 91. DOI=10.1145/1188455.1188551

[5] C. Yifeng Cui, K. B. Olsen, T. H. Jordan, K. Lee, J. Zhou, P. Small, D. Roten, G. Ely, D. K. Panda, A. Chourasia, J. Levesque, S. M. Day and P. Maechling, "Scalable Earthquake Simulation on Petascale Supercomputersm," Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10). IEEE Computer Society, Washington, DC, USA, 2010, pp.1-20. DOI=10.1109/SC.2010.45 http://dx.doi.org/10.1109/SC.2010.45

[6] M. Rietmann, P. Messmer, T. Nissen-Meyer, D. Peter, P. Basini, D. Komatitsch, O. Schenk, J. Tromp, L. Boschi, and D. Giardini. "Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures," Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12). IEEE Computer Society Press, Los Alamitos, CA, USA, 2012, Article 38, 11 pages.

[7] Y. Cui, E. Poyraz, K. B. Olsen, J. Zhou, K. Withers, S. Callaghan, J. Larkin, C. Guest, D. Choi, A. Chourasia, Z. Shi, S. M. Day, P. J. Maechling and T.H. Jordan. "Hysics-based seismic hazard analysis on petascale heterogeneous supercomputers," Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, (SC'13). IEEE Computer Society Press, New York, NY, USA, 2013, Article 70, 12 pages.

[8] M. Hori, T. Ichimura. "Current state of integrated earthquake simulation for earthquake hazard and disaster," Journal of Seismology, 2008, vol.12, No.2, pp. 307-321, DOI: 10.1007/s10950-007-9083-x.

[9] M. L. L. Wijerathne, M. Hori, T. Kabeyazawa and T. Ichimura. 2013. "Strengthening of parallel computation performance of integrated earthquake simulation," Journal of Computing in Civil Engineering, 2013, pp. 570-573.

[10] K. Fujita, T. Ichimura, M. Hori, M. L. L. Wijerathne and S. Tanaka. "Basic study on high resolution seismic disaster estimation of cities under multiple earthquake hazard scenarios with high performance computing," Journal of Japan Society of Civil Engineers, 2013, Ser. A2 (Applied Mechanics (AM)), Vol.69, No.2, pp. I_415-I_424 (in Japanese with English abstract).

[11] J. Liang and S. Sun. "Site effects on seismic behavior of pipelines: a review," Journal of Pressure Vessel Technology, 2000, American Society of Mechanical Engineers, 122, pp. 469–475.

[12] R. Taborda and J. Bielak. "Large-scale earthquake simulation: computational seismology and complex engineering systems," Computing in Science & Engineering, 2011, 13, pp. 14-27.

[13] R. Taborda, J. Bielak, D. Restrepo. "Earthquake ground-motion simulation including nonlinear soil effects under idealized conditions with application to two case studies," Seismological Research Letters, 2012; 83(6), pp. 1047-1060.

[14] T. Ichimura, K. Fujita, M. Hori, T. Sakanoue and R. Hamanaka. "Three-dimensional nonlinear seismic ground response analysis of local site effects for estimating seismic behavior of buried pipelines," Journal of Pressure Vessel Technology, 2014, American Society of Mechanical Engineers, **136**, 041702.

[15] H. Miyazaki, Y. Kusano, N. Shinjou, F. Shoji, M. Yokokawa and T. Watanabe. "Overview of the K computer system," FUJITSU Sci. Tech. J., 2012, Vol. 48, No. 3, pp.302-309.

[16] I.M. Idriss, R.D. Singh and R. Dobry. "Nonlinear behavior of soft clays during cyclic loading," Journal of the Geotechnical Engineering Division, 1978, **104**, pp. 1427–1447.

[17] G. Masing. "Eigenspannungen und verfestigung beim messing," *Proceedings of the 2nd International Congress of Applied Mechanics*, 1926, pp. 332–335 (in German).

[18] H. Akiba, T. Ohyama, Y. Shibata, K. Yuyama, Y. Katai, R. Takeuchi, T. Hoshino, S. Yoshimura, H. Noguchi, M. Gupta, J. Gunnels, V. Austel, Y. Sabharwal, R. Garg, S. Kato, T. Kawakami, S. Todokoro, J. Ikeda. "Large scale drop impact analysis of mobile phone using ADVC on Blue Gene/L," In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (SC '06), 2006, ACM, New York, NY, USA, Article 46. DOI=10.1145/1188455.1188503

[19] M. Ogino, R. Shioya and H. Kanayama. "An inexact balancing preconditioner for large-scale structural analysis," *Journal of Conputational Science and Technology*, 2008, **2(1)**, pp. 150–161.

[20] H. Kawai, M. Ogino, R. Shioya and S. Yoshimura. "Large-scale elast-plastic analysis using domain decomposition method optimized for multi-core CPU architecture," *Key Engineering Materials*, 2011, **462-463**, pp. 605–610.

[21] J. Mandel, "Balancing domain decomposition," Communications in numerical methods in engineering, 1993, Volume 9, Issue 3, pp.233-241.

[22] Earth Simulator (ES), http://www.jamstec.go.jp/es/en/es1/index.html

[23] Saad, Y., 2003, Iterative methods for sparse linear systems (2nd ed.), *SIAM*.

[24] Hairer, E., Nørsett, S. P., Wanner, G., 1993, Solving ordinary differential eauations I: Non stiff problems (2nd ed.), *Springer*.

[25] G. H. Golub, Q. Ye Inexact conjugate gradient method with inner-outer iteration. *SIAM., Journal on Scientific Computing* 1997; **21**(4):1305-1320.

[26] J. M. Winget, T. J. R. Hughes "Solution algorithms for nonlinear transient heat conduction analysis employing element-by- element iterative strategies," *Computer Methods in Applied Mechanics and Engineering* 1985; **52**:711-815.

[27] O.C. Zienkiewicz, R. L. Taylor. The Finite Element Method for Solid and Structural Mechanics (6th ed.), 2005, *Elsevier*.

[28] T. Ichimura, M. Hori, J. Bielak. "A Hybrid multiresolution meshing technique for finite element three-dimensional earthquake ground motion modeling in basins including topography," Geophysical Journal International, 2009, **177**, pp. 1221–1232.

[29] METIS 5.1.0, [Online].
http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[30] vSMP Foundation, ScaleMP, Inc., [Online].
http://www.scalemp.com/products/vsmp-foundation/

[31] Y. Ajima, T. Inoue, S. Hiramoto and T. Shimizu. "Tofu: interconnect for the K computer," FUJITSU Sci. Tech. J., 2012, Vol.48, No.3, pp.280-285.

[32] OpenMPI, [Online].
http://www.open-mpi.org/

[33] MPI: A Message-Passing Interface Standard, Version 2.1, [Online].
http://www.mpi-forum.org/docs/mpi21-report.pdf

[34] Strong ground motion of The Southern Hyogo prefecture earthquake in 1995 observed at Kobe JMA observatory, Japan Meteorological Agency, [Online].
http://www.data.jma.go.jp/svd/eqev/data/kyoshin/jishin/hyogo_nanbu/dat/H1171931.csv

[35] Stampede at Texas Advanced Computing Center, The University of Texas at Austin, [Online].
https://www.tacc.utexas.edu/resources/hpc/stampede-technical

[36] 5m mesh digital elevation map, Tokyo ward area, Geospatial Information Authority of Japan, [Online].
http://www.gsi.go.jp/MAP/CD-ROM/dem5m/index.htm

[37] National Digital Soil Map, The Japanese Geotechincal Society, [Online].
http://www.denshi-jiban.jp/

[38] Strong-motion seismograph networks (K-NET, KiK-net), National Research Institute for Earth Science and Disaster Prevention, [Online].
http://www.kyoshin.bosai.go.jp/

[39] G.W. Housner. "Spectrum intensities of strong-motion earthquakes," Symposium on earthquakes and blast effects on structures, 1952, Los Angeles, CA.

[40] S. Homma, K. Fujita, T. Ichimura, M. Hori, S. Citak and T. Hori. "A physics-based Monte Carlo earthquake disaster simulation accounting for uncertainty in building structure parameters," The International Conference on Computational Science, 2014 (in press).

[41] T. Ichimura, R. Agata, T. Hori, K. Hirahara, M. Hori. "Fast numerical simulation of crustal deformation using a three-dimensional high-fidelity model," *Geophysical Journal International*, 2013, **195**, pp.1730–1744.