Heuristic Optimization with CPU-GPU Heterogeneous Wave Computing for Estimating Three-dimensional Inner Structure

Takuma Yamaguchi[1], Tsuyoshi Ichimura[1], Kohei Fujita[1], Muneo Hori[2], and Lalith Wijerathne[1]

[1] Earthquake Research Institute and Department of Civil Engineering, The University of Tokyo, Bunkyo, Tokyo, Japan
`{yamaguchi, ichimura, fujita, lalith}@eri.u-tokyo.ac.jp`
[2] Japan Agency for Marine-Earth Science and Technology, Yokosuka, Kanagawa, Japan
`horimune@jamstec.go.jp`

Abstract. To increase the reliability of numerical simulations, it is important to use more reliable models. This study proposes a method to generate a finite element model that can reproduce observational data in a target domain. Our proposed method searches parameters to determine finite element models by combining simulated annealing and finite element wave propagation analyses. In the optimization, we utilize heterogeneous computer resources. The finite element solver, which is the computationally expensive portion, is computed rapidly using GPU computation. Simultaneously, we generate finite element models using CPU computation to overlap the computation time of model generation. We estimate the inner soil structure as an application example. The soil structure is reproduced from the observed time history of velocity on the ground surface using our developed optimizer.

Keywords: heuristic optimization, CPU-GPU collaborative computing, CUDA, Finite Element Analysis, Conjugate Gradient Method

1 Introduction

Numerical simulations with large number of degrees of freedom are becoming feasible due to the development of computation environments. Accordingly, more reliable models are required to obtain more reliable results for the target domain with complex structures. This approach has been discussed in various fields including biomedicine [2, 9], and it is also important for the numerical simulation of earthquake disasters. It is rational that we allocate resource and take countermeasures after detecting an area with potentially substantial damage. We can apply numerical simulation for estimating damages. The authors of [10] found that the geometry of the target domain significantly affects the distribution of displacement on the ground surface and strain in underground structures. To undertake well-suited countermeasures, three-dimensional unstructured finite element analysis is preferred, as it considers complex geometry. This analysis results

in problems with large degrees of freedom because it targets large domains with high resolution. The computation mentioned above has become more attainable due to the development of the computation environment and analysis methods for CPU-based large-scale systems [5]. However, inner soil structure is not available with high resolution, which hampers the generation of finite element models. On the ground surface, [13] is used as elevation data for Japan. With the advance of sensing technology, it is possible to observe earthquake waves on many points on the ground surface. It is desirable that we generate a finite element model which can reproduce observational data on the ground surface and conduct analyses using an estimated model. On the other hand, it is difficult to measure the underground structure with high accuracy and resolution.
One of realistic ways to address this issue is introduction of an optimization method using observation data on the ground surface for a micro earthquake. If we can generate many finite element models and conduct wave propagation analyses for each model, it is possible to select a model which can reproduce available observation data most closely. Using optimized models will increase the reliability of the analyses. There are some gradient-based methods for optimization as [12] proposed for three-dimensional crustal structure optimization. These methods have the advantage that the number of trials is small; however, they may be difficult to escape from a local solution if control parameters have low sensitivity to an error function. Thus, this study focuses on heuristic methods such as simulated annealing so that we can reach the global optimal solution robustly. The expected optimization requires many forward analyses, and the challenge is an increase in the computation cost for many analyses with large number of degrees of freedom.
We use GPUs in this paper. Its hardware and development environment are rapidly evolving [8]. The computation time can be reduced by using parallel computation with many GPU cores. However, it is known that GPU computation requires the consideration of memory access and communication cost to attain better performance. This paper proposes an algorithm that combines very fast simulated annealing and wave propagation analyses and repeats generation of finite element models and the computation of the solver for estimation of inner soil structure. Some computations in our optimizer are not suitable for GPU computation. Thus, computer resources are allocated so that we can benefit further from the introduction of GPU computation. A finite element solver appropriate for GPU computation is proposed to reduce the computation time in the solver, which is the most computationally expensive part. At the same time, generation of finite element models, which requires serial operations, is computed on CPUs so that computation time for model generation can be overlapped. We confirm that the inner soil structure has a large effect on the results and that our proposed method can estimate the soil structure with sufficient accuracy for damage estimation. Our proposed optimizer is proposed in the following section. Section 3 describes the estimation of soil structure using our developed optimizer. Section 4 describes our conclusions and future prospects.

2 Methodology

For estimating the inner structure of the target domain, this study proposes a method to conduct many wave propagation analyses and accept the inner structure with its maximum likelihood. In this study, optimization targets the estimation of boundary surfaces of the domain that has different material properties. For simplicity and for the purposes of this study, we have assumed that the target domain has a stratified structure and that target parameters for optimization are an elevation of the boundary surface on control points which are located at regular intervals in the x and y directions. A boundary surface is generated in the target domain by interpolating elevation on control points using linear functions. Figure 1 depicts the scheme for optimization. In this scheme,

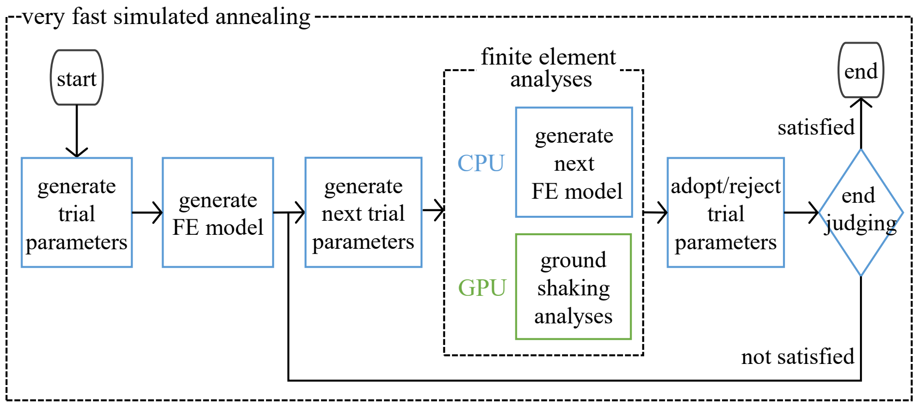


Fig. 1: Rough scheme for our proposed optimizer for an estimation of soil structure.

we conduct finite element analyses for evaluation of parameters many times in very fast simulated annealing. Our optimizer is designed so that the generation of a finite element model and the finite element solver, which account for the large proportion of the whole computation time, can be computed at the same time by CPUs and GPUs, respectively. We describe the details for each part of our optimizer in the following parts.

2.1 Very Fast Simulated Annealing

Very fast simulated annealing, which is a heuristic optimization method for problems with many control parameters [7], is applied. Simulated annealing has a parameter that corresponds to temperature, and the temperature decreases as the number of trials increases. We search and evaluate parameters in the following manner. First, trial parameters are selected randomly based on current parameters. The search parameter domain is wider when the temperature is higher. The evaluation value of trial parameters and that of previous ones are compared, and if the evaluation value is improved, parameters are always updated. Even if the evaluation value is worse, parameters are updated with a high degree of probability while the temperature is high. By repeating this procedure, this method

can move out of the local optimal solution and find the global optimal solution robustly. To evaluate the parameters, finite element analysis is conducted. We assume that we have many observation points on the surface of the target domain. Our error function is defined by the time histories of displacement in the analyses and observation data on observation points. The actual error function is defined in Section 3.
In very fast simulated annealing, temperature at the k-th trial is defined using initial temperature T_0 and the number of control points D as $T_k = T_0 \exp(-ck^{\frac{1}{D}})$, where parameter c is defined by T_0, D, lowest temperature T_f, and the number of trials k_f as $T_f = T_0 \exp(-m)$, $k_f = \exp n$, and $c = m \exp(-\frac{n}{D})$. The initial temperature, lowest temperature, and number of iterations depend on problems. A certain number of iterations are conducted for this simulation, though we can stop searching by other conditions, including acceptance frequency of new solutions. Also, we don't use re-annealing method mentioned in [7] because the cost for computing sensitivity of each parameter to the error function increases as D increases.

2.2 Finite Element Analyses

In the scheme, we must conduct more than 10^3 finite element analyses; thus, it is essential to conduct these analyses in a realistic timeframe. We target linear wave propagation analyses. Our governing equation is $\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla \cdot \sigma(\mathbf{u}) = \mathbf{f}$ on Ω, where $\mathbf{u}$ and $\mathbf{f}$ are displacement and force vector, ρ is density, and σ is strain, respectively. By using Newmark-β method with β=1/4 and δ=1/2 for time integration and discretizing the governing equation in space with finite element method, we can obtain the target equation $\left(\frac{4}{dt^2}\mathbf{M} + \frac{2}{dt}\mathbf{C} + \mathbf{K}\right)\mathbf{u}_n = \mathbf{f}_n + \mathbf{C}\mathbf{v}_{n-1} + \mathbf{M}\left(\mathbf{a}_{n-1} + \frac{4}{dt}\mathbf{v}_{n-1}\right)$, where $\mathbf{v}$ and $\mathbf{a}$ are velocity and acceleration vector, and $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are mass, damping, and stiffness matrix, respectively. In addition, dt is the time increment, and n is the number of time steps. For the damping matrix $\mathbf{C}$, we use Rayleigh damping and compute it by linear combination as $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$. Coefficients α and β are set so that $\int_{f_{\min}}^{f_{\max}}(h - \frac{1}{2}(\frac{\alpha}{2\pi f} + 2\pi f \beta))^2 df$ is minimized, where $f_{\max}$, $f_{\min}$, and h are maximum targeting frequency, minimum targeting frequency, and damping ratio. Vectors $\mathbf{v}_n$ and $\mathbf{a}_n$ can be described as $\mathbf{v}_n = -\mathbf{v}_{n-1} + \frac{2}{dt}(\mathbf{u}_n - \mathbf{u}_{n-1})$, $\mathbf{a}_n = -\mathbf{a}_{n-1} - \frac{4}{dt}\mathbf{v}_{n-1} + \frac{4}{dt^2}(\mathbf{u}_n - \mathbf{u}_{n-1})$. We obtain displacement vector $\mathbf{u}_n$ by solving the equation above and updating vectors $\mathbf{v}_n$ and $\mathbf{a}_n$. Computation in the finite element solver and generation of finite element models are most computationally expensive parts in our optimizer.

Finite Element Solver We developed a solver based on our MPI-parallel solver using conjugate gradient method [5]. The original solver has been developed for CPU-based supercomputers. The solver combines a conjugate gradient method with adaptive preconditioning, geometric multigrid method, and mixed precision arithmetic to reduce the amount of arithmetic counts and data transfer size. In

the solver, sparse matrix vector multiplication is computed by the Element-by-Element (EbE) method. It computes element matrix on-the-fly and reduces the memory access cost. Specifically, the multiplication $\mathbf{y} = \mathbf{A}\mathbf{x}$ is computed as $\mathbf{y} = \sum_{i=1}^{ne} (\mathbf{Q}^{(i)T}(\mathbf{A}^{(i)}(\mathbf{Q}^{(i)}\mathbf{x})))$, where ne is the number of elements in the domain, $\mathbf{Q}^{(i)}$ is a mapping matrix from local node numbers in the i-th element to the global node numbers, and $\mathbf{A}^{(i)}$ is the i-th element matrix and satisfies $\mathbf{A} = \sum_{i=1}^{ne} \mathbf{Q}^{(i)T}\mathbf{A}^{(i)}\mathbf{Q}^{(i)}$. In this problem, $\mathbf{A}^{(i)} = \frac{4}{dt^2}\mathbf{M}^{(i)} + \frac{2}{dt}\mathbf{C}^{(i)} + \mathbf{K}^{(i)}$. The entire part of the solver is implemented in the multiple GPUs using CUDA. To exhibit higher performance using GPUs, we have to reduce the operations that are not suitable for GPU computation; thus, we modify the algorithm of the solver.
When we compute EbE kernel in GPU, we assign one thread for one element and each element adds temporal results per element into the global vector. This summation can be operated without data race conditions using atomic operation; however, many random accesses to the global vector in this scheme may decrease the performance of the kernel. To improve the performance of this part, we use shared memory as a buffer and reduce the number of data accesses to global memory. These methods are extension of the finite element solver for crustal deformation computation by [14].
In addition, we overlap computation and communication as described in [11]. In the domain of each MPI process, some elements are adjacent to domains of other MPI processes and require point-to-point communications, and others do not require these communications. First we compute elements that require data transfer among other GPUs. Next we communicate with other GPUs while we are computing elements that do not require data transfer. By following this procedure, it is possible to overlap MPI point-to-point communication in the solver.
In the conjugate gradient solver, the coefficients are derived from the result of inner product calculations so that orthogonal residual vector and A-orthogonal searching vector can be generated to those in the previous iteration, respectively. When multiple GPUs are used with MPI, calculations of these coefficients require data transfer and synchronization among MPI processes such as MPI_Allreduce. Thus, they become relatively time-consuming taking into account that other computations including vector operations and sparse matrix vector multiplication are accelerated by GPUs. In our solver, we employ the method described in [1]. This algorithm requires one MPI_Allreduce per iteration, which halves the number of MPI_Allreduce per iteration in the original conjugate gradient method. The amount of vector operation increases in this scheme. However, the reduction of calculations of coefficients is more effective for GPU-based systems.

Generation of the Finite Element Model We automatically generate finite element model using the method by [3]. Its procedure utilizes OpenMP parallelization, where each thread has temporal array required to compute connectivity of elements and numbering of nodes. This enables us to compute model generation with up to 10^2 OpenMP threads on CPU; however, we cannot apply GPU computation for this part as GPU requires more than 10,000 threads for

Table 1: Material properties in target domain. V_p, V_s, and ρ are primary and secondary wave velocity, and density, respectively. h is the damping ratio used in the linear wave field calculation, $h_{\max}$ is maximum damping ratio, and γ is the reference strain used in the non-linear wave analyses.

	V_p(m/s)	V_s(m/s)	ρ (kg/m^3)	h	$h_{\max}$	γ
soft layer	700	100	1500	0.001	0.23	0.007
hard layer	2100	700	2100	0.001	0.001	-

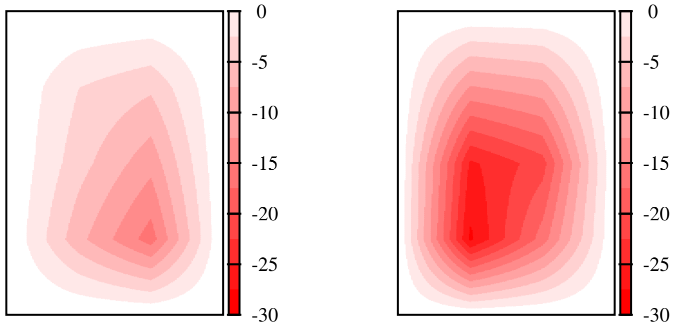
efficient computation and memory consumption greatly increases. Generation of finite element models can account larger proportion of the whole elapsed time, which is not negligible compared to the computation time in the finite element solver. Therefore, we design our optimizer so that it is possible to generate a finite element model for the next trial on CPUs while wave propagation analysis is computed on GPUs. All of the main computation in the solver can be computed in GPUs, so we can assign only one core of CPUs for each GPU and this has little effect on the performance of the solver. Other cores in CPUs are assigned for the generation of finite element models. Program of the model generation is created separately from that of the finite element solver and we executed them asynchronously using a shell-script. Output files are shared in the file system and controlled so that they are updated in correct timing. By allocating heterogeneous computer resource as mentioned above, it is possible to overlap model generation with GPU computation.

3 Application Example

We use our developed optimizer to estimate soil structure. Our target domain has two layers, and we define their boundary surface. We use IBM POWER System AC922 for computation, which has two POWER9 CPUs (16 cores, 2.6 GHz) and four NVIDIA Volta V100 GPUs. We assign one CPU core to each GPU for finite element analysis with MPI and we use the remaining 28 CPU cores for the model generation with OpenMP.
The target domain is 300 m $\times$ 400 m $\times$ 75 m. In this problem, material properties of the soil structure are deterministic. These properties are described in Table 1. In the model generation using [3], tetrahedral elements are generated based on a background octree-based structured grid, and according to the previous study [4], its resolution ds must satisfy the condition $ds < \frac{V_s}{10 f_{max}}$ in soft layers. The frequency components below 2.5 Hz are dominant in the strain response analysis, so we set $f_{max} = 2.5$ Hz. Thus, we set $ds = 2.5$ m so that the condition above is satisfied. Elevation data at the surface are available. They are flat and we set them as $z = 0$ m. Control points are located at regular intervals in the x and y directions. We simplify the problem and notate the elevation of the hard layer on points $(x, y) = (100i, 100j)$(i=0-3, j=0-4) as α_{ij} in metric units. The points $x = 0$, $x = 300$, $y = 0$, and $y = 400$ are the edges of the domain, and we assume $\alpha_{ij} = 0$ for these points. The parameters for optimization are α_{ij}(i=1-2, j=1-3). Initial parameters and reference parameters, which

Table 2: Parameters. The units of α are meters.

	α_{11}	α_{21}	α_{12}	α_{22}	α_{13}	α_{23}
initial model	-9.190	-16.260	-6.490	-11.660	-4.980	-7.050
reference model	-28.030	-16.260	-25.550	-21.140	-12.790	-11.090



(a) Initial model **(b)** Reference model

Fig. 2: Distribution of elevation (m) of the hard layer.

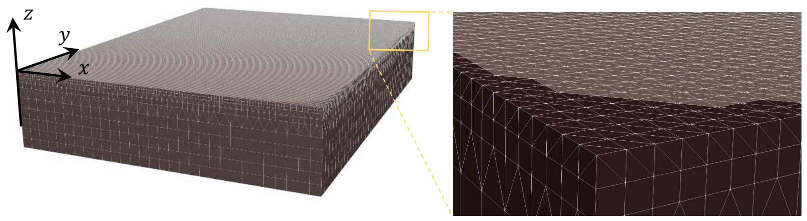


Fig. 3: One of finite element models in the analysis.

are true, are shown in Table 2. We assume the information from the boring survey are available at points $(x, y) = (50, 50), (150, 350), (200, 100)$. Elevations at these points are -7.01 m, -5.97 m, and -16.3 m, respectively, and these elevations are interpolated to make the initial boundary surface. The distributions of the boundary surface for initial and reference models are described as Fig. 2. An unstructured mesh with approximately 3,000,000 degrees of freedom is generated by using the method by [3]. Figure 3 shows one of the FE models in the analysis. Input waves for wave propagation analyses can be obtained by pulling back observed waves on the ground surface. In this paper, we assume that input waves are generated by micro earthquakes, and linear analysis can be applied. It is then possible to use Ricker wave as our input wave. We derive amplification functions from observation data and pulled back waves. Using these functions, it is possible to estimate observation data when we input Ricker wave. These operations reduce time steps for wave propagation analyses and the entire computation time. Ricker waves, represented as $(1 - 2\pi^2 f_c^2 (t - t_c)^2)\exp(-\pi^2 f_c^2 (t - t_c)^2)$, are input as x and y components of velocity at the bottom of models. t is time in second, f_c is central frequency and t_c is central time and they can be set independently. For this application example, the target frequency is as much as 2.5 Hz and we set the period of each analysis to 2.56 seconds. Considering these settings, we set $(f_c, t_c) = (0.8, 1.2)$. Time increment of the analysis must be small enough to converge the results in the time integration. We set it to 0.01 second, which is the same as the setting in [6]; thus each wave propagation analysis requires computation for 256 times steps. We set two cases for observation points. In case 1, we allocate 35 observation points defined as $(x, y) = (50i, 50j)$ (i=1-5, j=1-7) and in case 2, observation points are $(x, y) = (-50 + 100i, -50 + 100j)$ (i=1-3, j=1-4) and the number of points is 12. We use an error function as follows; $Error = \frac{1}{np}\sum_{i=1}^{np}\sum_{j=1}^{3}\int_0^{f_{\max}} |F[v_{ij}] - F[\bar{v_{ij}}]|df$, where np is the number of observation points, v is the velocity of the observation data, and $f_{\max}$ is the maximum targeting frequency, which is 2.5 Hz in our paper. v is the time history of x, y, and z components of velocity on each observation point. Values with a over-line corresponds to the observation data. In addition, $F[\ ]$ corresponds to the discrete Fourier transformation. In other words, this error function is the total sum of absolute values of difference for frequency components on observation points. These settings mentioned above are the same as settings in [6].

In our proposed method, we generate finite element models for the next trial and conduct wave propagation analysis at the same time. In simulated annealing, we generate next trial parameters after current trial parameters are adopted or rejected. It is desirable that we generate two models in cases that trial parameters are adopted and rejected while we are conducting wave propagation analysis; however, generation of finite element model twice takes more time than the computation in our finite element solver. Parameters in these problem settings are thought to be rejected with high probability. Thus, we generate a finite element model with prediction that trial parameters will be rejected. When trial parameters are adopted, we regenerate next finite element models for updated parameters. This regeneration has a small effect on the whole computation time. The breakdown of computation costs in our optimizer is described later in the performance evaluation part. The number of control points in very fast simulated annealing $D = 6$. Also, we set the number of trials $k_f = 1500$ and $c = 4.2975$. This c satisfies that parameters which increase the value of the error function by ΔE are adopted with the probability of 80% at the initial temperature and parameters which increase the value of the error function by $\Delta E \times 10^{-5}$ are adopted with the probability of 0.1% at the lowest temperature, where ΔE is the value of error function obtained in the initial model. The history of error function is described in Figure 4 and parameters are estimated as Table 3. Optimization of both case 1 and 2 adopted trial parameters 51 times in 1,500 trials. Trial parameters are rejected with the probability of more than 90% and we find that the generation of finite element model is mostly overlapped by the computation in the solver. Compared to case 2, case 1 with more observation points estimated the soil structure more accurately. Our previous study [6] used a multigrid stochastic search algorithm and optimized the same parameters in

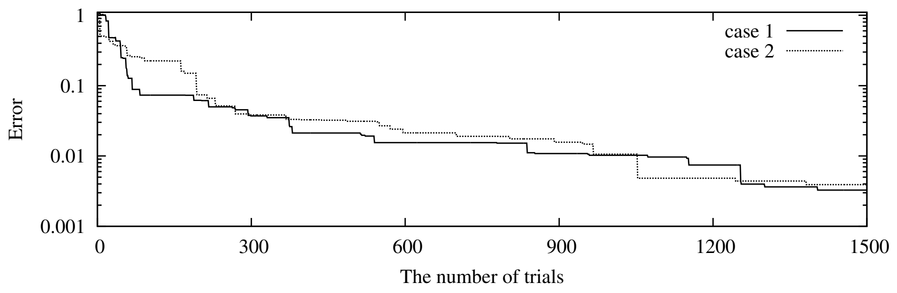


Fig. 4: Time history of error function. Each value is normalized by the error of the initial model.

Table 3: Parameters obtained by the optimizer for each case. The units of α are meters. *RSS* is the residual sum of squares based on the reference model and defined as $\sum_{ij}(\alpha_{ij}/\bar{\alpha}_{ij}-1)^2$, where $\bar{\alpha}_{ij}$ are parameters of the reference model.

	α_{11}	α_{21}	α_{12}	α_{22}	α_{13}	α_{23}	*RSS*
reference	-28.030	-16.260	-25.550	-21.140	-12.790	-11.090	-
case 1	-28.007	-16.290	-25.611	-21.092	-12.770	-11.100	1.6×10^{-5}
case 2	-28.119	-16.133	-25.484	-21.182	-12.849	-11.086	1.2×10^{-4}

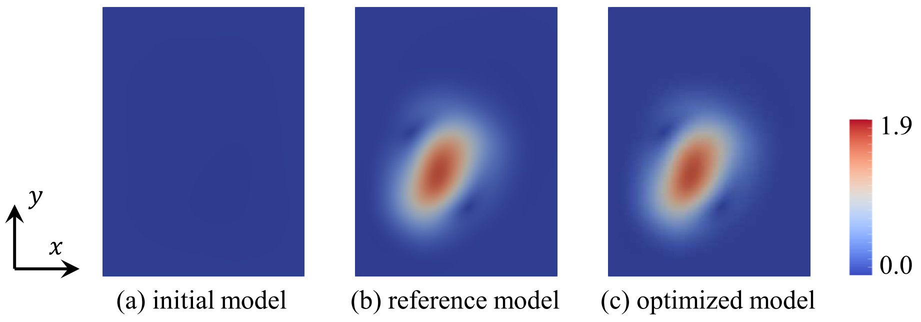


Fig. 5: Norm distribution of displacement (m) on the ground surface at $t = 2.20s$ in the linear ground shaking analysis.

meters. The numbers of iteration were 3,000 in case 1 and 1,300 in case 2; thereby we found that parameters were efficiently optimized with higher accuracy as the number of trials by the very fast simulated annealing was 1,500. For confirmation of the optimized model, we conduct wave propagation analysis with parameters obtained in case 1. Figure 5 is the distribution of the displacement on the ground surface at time $t = 2.20$ s and Fig. 6 is the time history of the velocity on point $(x, y, z) = (150, 200, 0)$. Judging from these figures, we can confirm that the results by optimized model and reference model are consistent.

Here we evaluate the performance of the computation in our optimization. The elapsed time for our solver part is about 18 s per trial. In [6], wave propagation

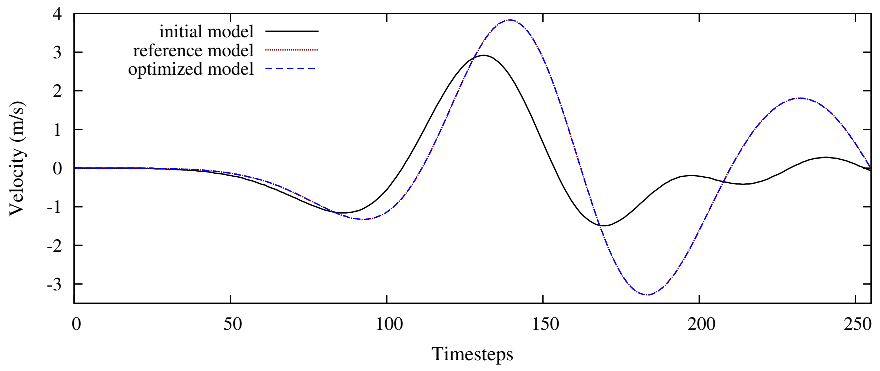
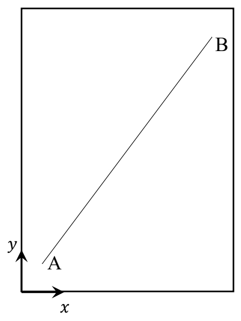


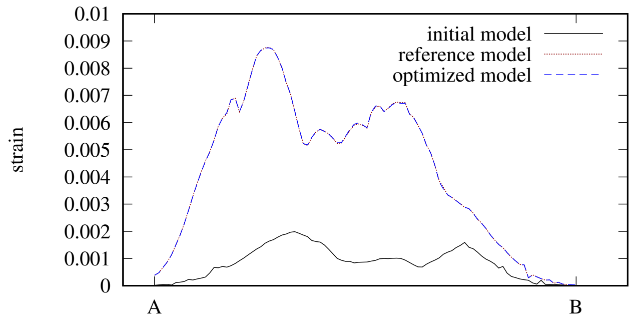
Fig. 6: x component of the velocity at $(x, y) = (150, 200)$ on the ground surface in the linear ground shaking analysis.

analysis was computed in 263 s for a finite element model with 274,041 degrees of freedom using Intel Xeon E5-4667 v3 CPU.If this CPU-based solver is used for the analysis in this paper and we assume that computation cost increases in proportion to the number of degrees of the freedoms, estimated elapsed time will be 3,000,000/274,041=10.94 times longer and 263 s $\times$ 10.94 = 2,879 s. Therefore, our GPU-based solver has achieved about 160-fold speeding up per problem size, although it is difficult to compare the performance on different systems. Here we use peak memory bandwidth to evaluate the speeding up ratio, as general finite element analyses are memory bandwidth bound computations. Intel Xeon E5-4667 v3 CPU has 68 GB/s and four NVIDIA V100 GPUs have 900 GB/s $\times$ 4 = 3,600 GB/s of memory bandwidth. We attained higher speeding up ratio than the ratio of peak memory bandwidth; this indicates that we efficiently computed on GPUs. The optimization in case 1 was computed in 13 h 32 min. The elapsed time per trial in simulated annealing was 32 s. The breakdown of computation cost was as follows: The computation part of our solver took 18 s, other part of our solver such as I/O and data transfer from CPU to GPU before computation took 7 s, model decomposition for MPI parallelization using METIS took 3 s, postprocessing to obtain the response on the ground surface took 2 s, and other computations including Fourier transformation took 2 s. Besides, it took about 10 s for the generation of each model, which is overlapped with the computation in the solver; thus, the whole elapsed time would be 13 h 32 min + 10 s $\times$ 1500 = 17 h 42 min and increase by 30% if model generation and finite element solver were computed sequentially. Thereby we confirmed that efficient allocation of computer resources is important for this optimization. Considering our previous method took 9.4 days for parameter optimization in meters using finite element model with 1/10 of degrees of freedoms, our proposed method has achieved great reduction in computational cost.

Finally, we conduct a non-linear ground shaking analysis using the optimized model. The methods are the same as [5]. We input wave observed in the 1995 Kobe Earthquake at the Kobe Local Meteorological Office and its time incre-



(a) Location



(b) Maximum axial strain of the pipeline

Fig. 7: Maximum distribution of axial strain along a buried pipeline for each model in the non-linear ground shaking analysis. The buried pipeline is located between point A $(x, y, z) = (30, 40, -1.2)$ and point B $(x, y, z) = (270, 360, -1.2)$.

ment is 0.005 s and the number of time steps is 16,384. We used the modified Ramberg-Osgood model and Masing Rule for non-linear constitutive models. We assume that a gas pipeline is buried as shown in Fig. 7 (a). Figure 7 (b) shows the maximum axial strain of the pipeline. We can confirm that the strain distributions obtained by our optimized model and initial model, which is derived from boring survey, are completely different. This analysis is used for screening of underground structures which will be damaged and its result shows that this optimization is important to assure the reliability of the result.

4 Conclusion

To increase the reliability of numerical simulations, it is essential to use more reliable models. Our proposed optimizer searches for a finite element model that can reproduce observation data by combining very fast simulated annealing and finite element analyses. As an application example, we estimated soil structure using observation data with 1,500 wave propagation analyses with a finite element model with 3,000,000 degrees of freedom. The finite element solver, which accounted for the large proportion of the whole computation time, was accelerated by utilizing the GPU computation. Compared to the previous study, the elapsed time per problem size was decreased by 1/160. Generation of a finite element model was difficult to computed on GPUs. We designed our algorithm so that the computation in model generation on CPUs was overlapped by the computation in the solver on GPUs and enhanced the effect of GPU acceleration. For future prospects, more trials will be required for larger problem size or parameter searching in higher resolution, as the convergence of simulated annealing gets worse. To reduce the computation time, we must attain more speedup ratio for the solver or design a faster algorithm of our optimizer.

References

1. AT Chronopoulos and Charles William Gear. s-step iterative methods for symmetric linear systems. *Journal of Computational and Applied Mathematics*, 25(2):153–168, 1989.
2. R Clement, J Schneider, H-J Brambs, A Wunderlich, Martin Geiger, and Franz Günter Sander. Quasi-automatic 3D finite element model generation for individual single-rooted teeth and periodontal ligament. *Computer methods and programs in biomedicine*, 73(2):135–144, 2004.
3. Tsuyoshi Ichimura, Ryoichiro Agata, Takane Hori, Kazuro Hirahara, Chihiro Hashimoto, Muneo Hori, and Yukitoshi Fukahata. An elastic/viscoelastic finite element analysis method for crustal deformation using a 3-D island-scale high-fidelity model. *Geophysical Journal International*, 206(1):114–129, 2016.
4. Tsuyoshi Ichimura, Kohei Fujita, Muneo Hori, Takashi Sakanoue, and Ryo Hamanaka. Three-dimensional nonlinear seismic ground response analysis of local site effects for estimating seismic behavior of buried pipelines. *Journal of Pressure Vessel Technology*, 136(4):041702, 2014.
5. Tsuyoshi Ichimura, Kohei Fujita, Pher Errol Balde Quinay, Lalith Maddegedara, Muneo Hori, Seizo Tanaka, Yoshihisa Shizawa, Hiroshi Kobayashi, and Kazuo Minami. Implicit nonlinear wave simulation with 1.08 T dof and 0.270 T unstructured finite elements to enhance comprehensive earthquake simulation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 4. ACM, 2015.
6. Tsuyoshi Ichimura, Kohei Fujita, Atsushi Yoshiyuki, Pher Errol Quinay, Muneo Hori, and Takashi Sakanoue. Performance enhancement of three-dimensional soil structure model via optimization for estimating seismic behavior of buried pipelines. *Journal of Earthquake and Tsunami*, 11(05):1750019, 2017.
7. Lester Ingber. Very fast simulated re-annealing. *Mathematical and computer modelling*, 12(8):967–973, 1989.
8. David Kirk et al. NVIDIA CUDA software and GPU parallel computing architecture. In *ISMM*, volume 7, pages 103–104, 2007.
9. Rolf M Koch, Markus H Gross, Friedrich R Carls, Daniel F von Büren, George Fankhauser, and Yoav IH Parish. Simulating facial surgery using finite element models. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 421–428. ACM, 1996.
10. Jianwen Liang and Shaoping Sun. Site effects on seismic behavior of pipelines: a review. *Journal of pressure vessel technology*, 122(4):469–475, 2000.
11. Paulius Micikevicius. 3D finite difference computation on GPUs using CUDA. In *Proceedings of 2nd workshop on general purpose processing on graphics processing units*, pages 79–84. ACM, 2009.
12. Pher Errol B Quinay, Tsuyoshi Ichimura, and Muneo Hori. Waveform inversion for modeling three-dimensional crust structure with topographic effects. *Bulletin of the Seismological Society of America*, 102(3):1018–1029, 2012.
13. Geospatial Information Authority of Japan Tokyo ward area. 5m mesh digital elevation map. `http://www.gsi.go.jp/MAP/CD-ROM/dem5m/index.htm`.
14. Takuma Yamaguchi, Kohei Fujita, Tsuyoshi Ichimura, Anne Glerum, Ylona van Dinther, Takane Hori, Olaf Schenk, Muneo Hori, and Lalith Wijerathne. Viscoelastic crustal deformation computation method with reduced random memory accesses for GPU-based computers. In *International Conference on Computational Science*, pages 31–43. Springer, 2018.