doi: 10.32131/jes.4.2

*Journal of the Earth Simulator*, Volume 4, December 2005, 2–10

# ACuTEMan: A multigrid-based mantle convection simulation code and its optimization to the Earth Simulator

Masanori Kameyama

*The Earth Simulator Center, Japan Agency for Marine-Earth Science and Technology,*
*3173-25 Showa-machi, Kanazawa, Yokohama, 236-0001, Japan*
*Email: kameyama@jamstec.go.jp*

(Received April 27, 2005; Revised manuscript accepted June 6, 2005)

**Abstract** We report the current status of our numerical simulation code named "ACuTEMan" for large-scale mantle convection problems on the Earth Simulator. The ACuTEMan code comes out from a newly-developed "ACuTE" algorithm, which solves the flow field of mantle convection in combination with the multigrid method. This algorithm iteratively solves the steady-state equations for conservation of mass and momentum for highly viscous and incompressible fluids, and is proved to be suitable for mantle convection problems with excellent vector and parallel capability. In addition, a careful adaptation of the multigrid procedure enabled us to obtain a moderate parallel efficiency using up to 64 processor nodes of the Earth Simulator. In this paper we present the optimization and the performance of our method applied to mantle convection problems in a three-dimensional rectangular domain, with special emphasis on the enhancement of parallel efficiency of multigrid procedures by an "agglomeration" technique.

**Keywords**: Mantle Convection, Variable Viscosity, Multigrid Method, Parallelization, Agglomeration

## 1. Introduction

The convective motion in the Earth's mantle is the ultimate origin of the geological and geophysical phenomena observed at the Earth's surface, such as seismicity, volcanism and plate tectonics [1, 2]. A major tool for understanding the mantle convection is numerical analysis. Many numerical codes have been developed since the pioneering works in 1960's and 1970's [3, 4], and they have been playing an important role in the study of mantle convection (see [5] for a review). Now that the Earth Simulator is open to the studies on solid earth sciences, further progress is expected in numerical models of mantle convection in order to deepen our understanding of the dynamics of the Earth's interior.

In reality, however, numerical researchers of mantle convection seem to be suffering from the difficulties in adapting their codes to the Earth Simulator. The difficulties mainly lie in the solution procedures of flow fields of mantle convection. Because of the extremely high viscosity of mantle materials ($\sim 10^{22}$ Pa s) [1, 5], the flow in the mantle is described by a steady-state Stokes flow balancing among the buoyancy force, pressure gradient and viscous resistance. In addition, their viscosity varies by several orders of magnitude depending on temperature, pressure, and stress [6, 7]. Taken together with the assumption of incompressibility, one needs to solve ill-conditioned elliptic differential equations for velocity and pressure at every timestep. In order to promote the mantle convection studies in the era of Earth Simulator, it is very important to develop efficient numerical techniques that can deal with the steady-state flow of highly viscous and incompressible fluids with a strongly variable viscosity.

To overcome these difficulties, we developed a new simulation code named "ACuTEMan" for mantle convection problems on the Earth Simulator. A major innovation in our code is a solution algorithm for the flow fields of mantle convection. This algorithm makes good use of the multigrid method [8, 9, 10] together with a new smoothing algorithm named "ACuTE" [11]. The ACuTE algorithm is proved to be suitable for mantle convection problems with an excellent vector and parallel capability. Moreover, through a further optimization of multigrid procedure, we obtained a moderate overall parallel efficiency with our code. In the following part of this paper, we will introduce the solution techniques employed in our code. In particular, we will describe the solution algorithm of flow fields in detail, with special emphasis on the optimization of multigrid method.

## 2. Overview of "ACuTEMan"

Our "ACuTEMan" code is basically designed for thermal convection of a highly viscous and incompressible

fluid with a strongly variable Newtonian viscosity in a three-dimensional Cartesian geometry. The nondimensional forms of the fundamental equations are (for example [12, 13]);

$$\frac{\partial T}{\partial t} + \boldsymbol{v}\cdot\nabla T = \nabla^2 T + q, \tag{1}$$

$$0 = \nabla p + \nabla\cdot\left[\eta\left(\nabla\boldsymbol{v} + {}^t\nabla\boldsymbol{v}\right)\right] + Ra\,T\,\hat{z}, \tag{2}$$

$$0 = \nabla\cdot\boldsymbol{v}, \tag{3}$$

where superscript $t$ means transpose, $\boldsymbol{v}$ is fluid velocity, $p$ pressure, $\eta$ viscosity, $T$ temperature, $Ra$ the Rayleigh number, $\hat{z}$ the unit vector in $z$-direction, and $q$ is the internal heating rate. We assumed that $z$-axis is the vertical axis pointing upward. Boussinesq approximation is employed in the energy equation (1) and, hence, the effects of adiabatic and viscous heating are ignored. The temporal evolution of thermal convection is calculated separately in two parts. In the first part, we solve the energy equation (1), and update the temperature field. In the second part, we solve the equation of motion (2) together with the equation of continuity (3), and update the velocity $\boldsymbol{v}$ and pressure $p$ fields.

In the following subsections, we will describe the solution methods employed in our code in turn. In particular, we will discuss in detail the algorithm and its implementation of solution method for flow field ($\boldsymbol{v}$ and $p$), where we made a distinct innovation for efficient computations on the Earth Simulator.

### 2.1 Solver for Temperature Field

This part updates the temperature field $T$ through a temporal integration of the energy equation (1). Equation (1) is discretized by a first-order Euler method in time. An upwind scheme, called power-law scheme [14], is used to evaluate the contributions of heat transport by advection and conduction. The velocity field $\boldsymbol{v}$ at old time is used in evaluating the advection term. The discretized equation for $T$ can be solved in a straightforward manner by, for example, a fully explicit, a fully implicit scheme, or their combination (Crank-Nicholson scheme). In the previous paper [11], we had already shown the results calculated by both of a fully explicit scheme and a fully implicit scheme, and verified the reliability of our numerical results.

### 2.2 Multigrid Solver for Flow Field

This part solves the equations (2) and (3) simultaneously for the velocity $\boldsymbol{v}$ and pressure $p$ fields at new time using the new temperature $T$ updated by (1). Here we assume, for simplicity, the viscosity $\eta$ at new time is already known by some means and kept unchanged while the velocity and pressure are computed. These assumptions yield a set of linear elliptic equations for velocity and pressure, which is solved by a multigrid method.

The multigrid is known as the optimal method for a numerical solution of large-scale elliptic equations. The basic idea of the multigrid is to solve the elliptic problem using a hierarchy of grids with different numbers of grid points (see Figure 1). Suppose an iterative procedure for solving the discretized problem on the finest grid. Assuming that the approximate solutions are already calculated on coarser grids, the iterative procedure on the finest grid can be started from an initial guess provided by an interpolation of the approximates on the coarser grids. Since the initial guess defined thus is most likely to be a good approximation, the computational costs of the iterative procedure is expected to be significantly saved. However, in most of practical multigrid methods, the computations on different grids are intended to reduce (or smooth) the components of errors in the approximate solutions with different spatial wavelengths. The smoothing calculations on finer grids are responsible for reducing the components with shorter wavelengths, while those on

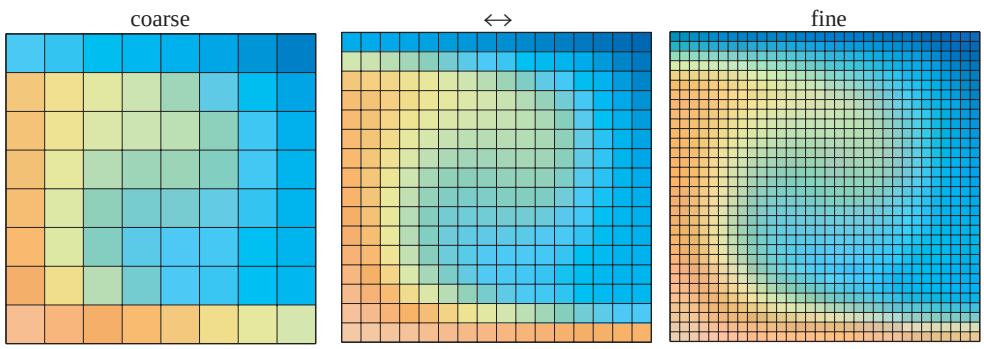


**Fig. 1** A schematic diagram of the principle of the multigrid method. By providing an initial guess from the interpolation of approximates on coarser grids, the computational cost for the iterative procedure for solving the discretized problem on a particular grid can be significantly saved.

coarser grids are for the components with longer wavelengths. By appropriately combining the approximates on different grids, the desired solution is obtained on the finest grid. A more detailed description of the multigrid methods can be found in, for example, [8, 9, 10].

The multigrid procedures are, in general, composed of multigrid components (including smoothers, inter-grid transfers, and derivation of coarse-grid equations) and their hierarchical combination into multigrid cycles over the entire grid levels. In the rest of this subsection, we will introduce them in turn. In particular, we will discuss in detail the smoothing operator and the multigrid cycles, where we made special adaptations to the Earth Simulator.

#### 2.2.1 Smoothing algorithm "ACuTE"

In this subsection, we briey introduce a newly-developed smoothing algorithm named "ACuTE". The name comes from an acronym of "Artificial Compressibility using local Time Evolution". The detail of the algorithm can be found in [11].

The ACuTE algorithm comes from an extension of the artificial compressibility method [15], which is often used to solve steady-state flow of incompressible fluid with high Reynolds number, to the problems of highly viscous fluid with variable viscosity. In this algorithm, instead of directly handling (2) and (3), we consider the following set of "pseudo-evolutionary" equations for $\boldsymbol{v}$ and $p$:

$$M\frac{\partial \boldsymbol{v}}{\partial \tilde{t}} = -\nabla p + \nabla\cdot\left[\eta\left(\nabla\boldsymbol{v} + {}^{t}\nabla\boldsymbol{v}\right)\right] + RaT\hat{z}\,, \qquad (4)$$

$$-K\frac{\partial p}{\partial \tilde{t}} = \nabla\cdot\boldsymbol{v}. \qquad (5)$$

Here $\tilde{t}$ is analogous to time, and $M$ and $K$ are positive parameters analogous to density and compressibility, respectively. The pseudo-time integration is repeated until a steady solution is achieved. It is obvious that the steady solution of (4) and (5) satisfies (2) and (3), and that it does not depend on the choice of $M$ or $K$. In addition, we employ the local time-stepping technique in order to accelerate the convergence to a steady state in the presence of spatial variations in $\eta$. In the present algorithm, this technique is taken into account by varying $M$ and $K$ depending on $\eta$. As had been demonstrated in the earlier paper [11], the convergence becomes sufficiently fast even for the cases with variable viscosity, by assuming $M \propto \eta$ and $K \propto \eta^{-1}$. Indeed, as will be shown later (see (6)), the convergence behavior of the ACuTE algorithm is mainly controlled by the distribution of viscosity $\eta$. This also implies that the convergence behavior is not, in principle, affected by other parameters such as the Rayleigh number $Ra$ or the internal heating rate $q$.

We note that the ACuTE algorithm is suitable for large-scale numerical simulations on massively vector-parallel supercomputers such as the Earth Simulator. The pseudo-temporal evolution of (4) and (5) can be naturally vectorized and parallelized. In particular, its vectorization and parallelization are quite straightforward if these equations are discretized by a fully explicit scheme in the direction of pseudo-time $\tilde{t}$. In addition, this algorithm can be regarded as a kind of iterative solvers of linear simultaneous equations. This implies that this algorithm can be directly used as a smoother of multigrid method in place of basic iterative methods (such as Jacobi or Gauss-Seidel methods).

We also note that we must carefully choose the costs of smoothing calculations by (4) and (5), in particular when the spatial variation in viscosity is involved. To see this more clearly, we consider the Courant-Friedrichs-Lewy (CFL) criterion for (4) and (5). From these equations we obtain the pseudo-evolutionary equation of $\nabla\cdot\boldsymbol{v}$ as,

$$\begin{aligned}\frac{\partial^2}{\partial \tilde{t}^2}(\nabla\cdot\boldsymbol{v}) = {} & \frac{1}{KM}\nabla^2(\nabla\cdot\boldsymbol{v}) + \frac{2\eta}{M}\frac{\partial}{\partial\tilde{t}}\left[\nabla^2(\nabla\cdot\boldsymbol{v})\right] \\ & + \frac{1}{KM}\nabla\cdot\left[(\nabla\cdot\boldsymbol{v})\nabla(\ln\eta)\right] \\ & + \frac{\eta}{M}\nabla\cdot\left[\nabla(\ln\eta)\cdot\frac{\partial}{\partial\tilde{t}}(\nabla\boldsymbol{v} + {}^{t}\nabla\boldsymbol{v})\right]. \qquad (6)\end{aligned}$$

(In (6) we assumed $M \propto \eta$ and $K \propto \eta^{-1}$.) The first term in the right-hand side of (6) represents the effect of "pseudo-sound wave" which propagates at a uniform rate $1/\sqrt{KM}$, and the second term represents the effect of viscous damping at a uniform diffusion coefficient $2\eta/M$. The effect of spatial variation in $\eta$ is included in the third and fourth terms, and it acts as an additional "advection velocity" of $\nabla\cdot\boldsymbol{v}$. Therefore, the CFL criterion demands smaller increment of pseudo-time $\Delta\tilde{t}$ in proportion to the viscosity variation. In other words, the smoothing effect of single pseudo-temporal integration becomes less significant as the spatial variation in $\eta$ is stronger. A simple remedy is to increase the number of pseudo-temporal integrations according to the magnitude of the spatial variation in $\eta$. In the present version of ACuTEMan, the number of pseudo-time integrations $N_s$ for each smoothing step on the grid level $\ell$ is taken to be,

$$N_s(\ell) = \left[\ln(r_\eta) + 1\right] \times 8 \times 2^{N_{\mathrm{grid}}-\ell} \quad (1 \le \ell \le N_{\mathrm{grid}}), \qquad (7)$$

where $r_\eta$ is the magnitude of viscosity variation in the entire domain. As had been demonstrated in the earlier paper [11], we obtained a significant improvement in the robustness of our method against the viscosity variation $r_\eta$.

#### 2.2.2 Multigrid method and its optimization for vector-parallel environments

One of the difficulties in optimizing the multigrid pro-

grams comes from the fact that the computations on coarser grids become less efficient. As the grid becomes coarser, the size of the discretized problems becomes smaller. The reduction of problem sizes on coarse grids, which is the heart of multigrid methods, degenerates the computational efficiency from two aspects. First is that the amounts of arithmetic computations become smaller as the grid becomes coarser. This may severely spoil the capability of vector operations, since the reduction of problem sizes results in shorter vector lengths on coarser grids. Second is that the relation between computation and communication becomes worse than on fine grids. As the problem size becomes smaller, the relative communication overhead (as compared to the time for arithmetic computations) increases, and may finally dominate the arithmetic work on the very coarse grids. This can result in a significant loss of efficiency for the overall parallel computation.

Another difficulty may arise when performing parallel computations with a large number of processor elements (PEs). If we proceed coarser and coarser grids during a multigrid cycle, smaller and smaller numbers of meshes are assigned to each PE. Then at a certain coarse grid level, the number of meshes becomes smaller than that of PEs involved in parallel computations. In such a situation, the domain decomposition strategy becomes incompatible with the grid coarsening for very coarse grid levels. Even if the calculations are parallelized by some means, the relative communication overhead on these grids cannot be affordable.

The above consideration suggests that the key to optimizing the parallelized multigrid is to enhance the computational efficiency particularly on coarse grid levels. In the following, we will introduce our strategies to reduce the time spent in the execution on coarse grids.

(1) Tips for construction of Multigrid cycles

We construct the multigrid cycle as summarized in Figure 2. In our method, the V-cycle is used with slight modification. The benefit of V-cycle is that it visits coarser grids at fewer times than other multigrid cycles (such as W-, F-cycles) do. In addition, we employed an "adaptive" strategy in determining which of adjacent grid levels is visited. We measure how much the residuals are reduced after every pre-smoothing step during the V-cycle (green circles in Figure 2). If the residuals are sufficiently reduced at a certain grid level, we go up to the finer grid level (thick green arrows in Figure 2), instead of going down to the coarser grid level. By this configuration of multigrid cycle, the calculations on coarse grids can be saved as much as possible.

(2) Tips for domain decomposition

The basic strategy for parallelization is, as well as in many applications, based on the domain decomposition

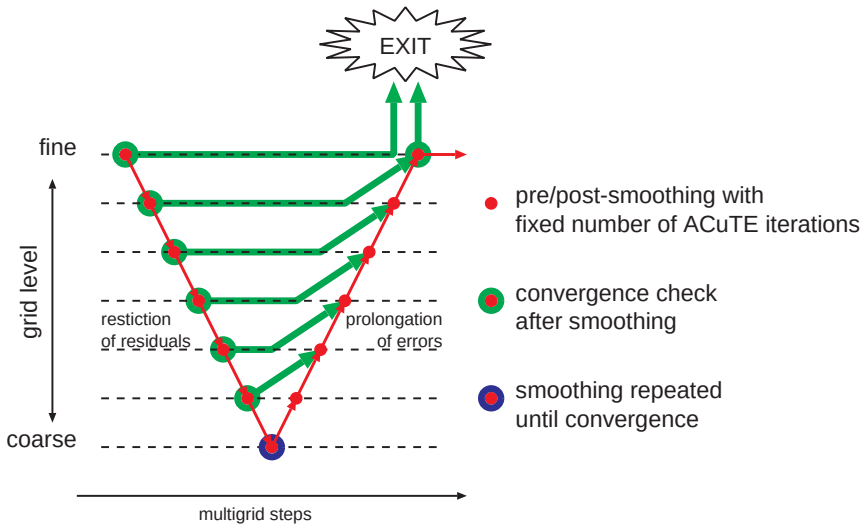


**Fig. 2** A schematic diagram of the multigrid cycle employed in this study. Shown in red are the course of calculations in a standard V-cycle of multigrid iterations. The red dots indicate the pre- and post-smoothing steps where a fixed number of ACuTE iterations are carried out. The red downward arrows indicate the restriction of residuals, followed by the visit to a coarser grid level. The red upward arrows indicate the prolongation of errors, followed by the visit to a finer grid level. The blue circle indicates that the smoothing steps are repeated at the coarsest grid level until a convergence is achieved. The green circles indicate that a convergence check is performed after the smoothing steps. The thick green arrows indicate the courses of calculations taken when the residual becomes sufficiently small. In these cases we skip the visits to coarser grids, and either move back to a finer grid level or stop the multigrid iterations.

method with partial overlaps. By applying this strategy to all of the grid levels involved, the multigrid components (such as smoothers) can be parallelized in a straightforward manner. In the present method, the widths of the overlap regions are taken to be 1 for all grid levels, which are sufficient for the spatial discretization with second-order accuracy. Here we also note that the geometrical domains at all grid levels should be divided in the same manner. In other words, each PE is responsible for the computations on the same subdomain on a sequence of grid levels. By assigning the same geometric positions of decomposed subdomains for all grid levels, we can save communications during inter-grid transfers.

In addition, we adjusted the manner of domain decomposition on very coarse grids where the number of computational meshes becomes small compared to that of PEs involved in parallel computations. Figure 3 summarizes our strategy. Suppose a model problem which is solved using $N_{\mathrm{grid}}$ grid levels and $N_{\mathrm{PE}}$ processor elements (PEs). Let us first divide the $N_{\mathrm{grid}}$ grid levels into two groups, one with grid level $\ell \geq \ell_{\mathrm{thres}}$ and the other with $\ell < \ell_{\mathrm{thres}}$ (see blue dashed line in Figure 3), depending on whether or not the number of meshes $N_{\mathrm{mesh}}(\ell)$ on grid level $\ell$ is sufficiently larger than $N_{\mathrm{PE}}$ ($N_{\mathrm{PE}} = 512$, $N_{\mathrm{grid}} = 8$ and $\ell_{\mathrm{thres}} = 4$ in the present case). Next, we apply different patterns of domain decomposition to the different groups of grid levels. On fine grid levels $\ell \geq \ell_{\mathrm{thres}}$ (namely $N_{\mathrm{mesh}}(\ell) \gg N_{\mathrm{PE}}$), the entire computational domain is decomposed into $N_{\mathrm{PE}}$ subdomains. According to the standard parallelization strategy, the computations on these grid levels are carried out in parallel, by assigning each subdomain to different PEs. On coarse grid levels where the above condition is not met ($\ell < \ell_{\mathrm{thres}}$), in contrast, we give up decomposing the entire meshes into subdomains. Instead of using all of the available PEs in parallel, the computations in the entire domain are carried out by one PE (in this case the PE with rank 0 is chosen), leaving other PEs idle. This strategy is sometimes called "agglomeration" technique [10]. The idea of this approach is to "agglomerate" the computational meshes to new "process units" and to redistribute these new units to a subset of the active PEs.

A major benefit of the agglomeration is that this strategy improves the overall patterns of communications. As had been already pointed out, the main cause of communication overhead was the frequent communications during the smoothing calculations on coarse grid levels. In the "agglomerated" case, in contrast, significant portion of communication overhead can be removed, since no communication is required on grid levels where only one PE is involved in computations. Of course, the agglomeration itself requires a large amount of communication associated with the inter-grid transfers between $\ell = \ell_{\mathrm{thres}}$ and $\ell_{\mathrm{thres}} - 1$ (thick green arrows in Figure 3). The data

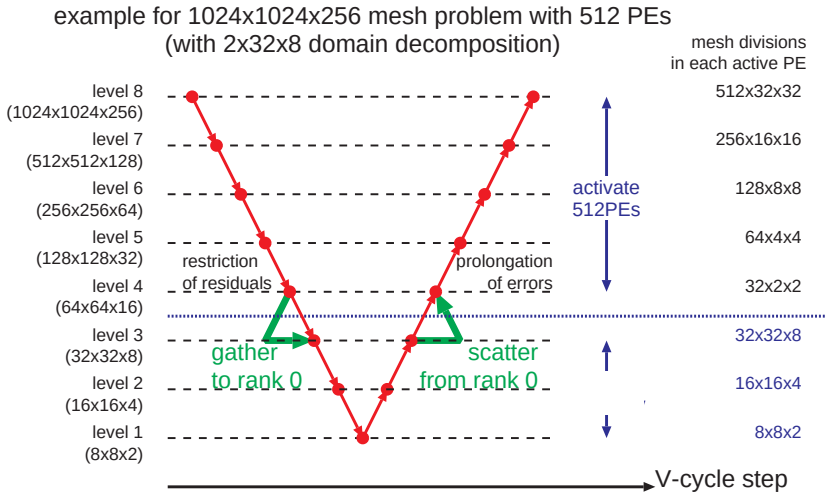


**Fig. 3** A schematic diagram of the course of multigrid iterations together with the pattern of domain decompositions employed in this study. The red dots and red arrows indicate the course of calculations in a standard V-cycle of multigrid method. The dotted blue line indicates a threshold grid level which divides the entire grid levels into two groups. The grid levels finer than the threshold contain a sufficient number of meshes compared to that of available processor elements (PEs) for parallel computations, while the coarser levels do not. The calculations on the finer grid levels are performed in parallel using all available PEs, while the calculations on the coarser levels are done using only one PE with rank 0. The thick green arrows indicate the communications between all PEs and the PE with rank 0, which take place during the transitions of grids across the threshold level.

must be gathered from all PEs to one PE when going down from grid level $\ell = \ell_{\text{thres}}$ and, similarly, must be scattered from one PE to all PEs when going up to $\ell = \ell_{\text{thres}}$. However, the increase in communication costs by agglomeration is most likely to be compensated by the decrease coming from the removal of frequent communications during the smoothing calculations for $\ell < \ell_{\text{thres}}$.

Another benefit of the agglomeration is that it can improve the computational efficiency during the calculations on $\ell < \ell_{\text{thres}}$. This is somewhat counterintuitive, since the reduction in the number of active PEs nominally increases the execution time for the same amount arithmetic works. However, this strategy can work particularly on vector architectures. In the agglomerated case a larger number of meshes are mapped to active PEs than in non-agglomerated case, which results in better vectorization because of larger vector lengths. The increase in computational time by agglomeration is most likely to be compensated as a whole by the decrease coming from the enhanced vector capability for $\ell < \ell_{\text{thres}}$.

#### 2.2.3 Notes for Other Multigrid Components

Other multigrid components are constructed according to a "standard" strategy of multigrid. A linear interpolation is used in both fine-to-coarse (restriction) and coarse-to-fine (prolongation) operations. The discretized equations on coarser grids are derived by directly discretizing the differential equations (2) and (3) on the particular coarser grids. The values of viscosity on coarser grids, which are necessary to derive the coarse-grid equations, are calculated by an appropriate interpolation from those on the finer grids by one grid level.

Here we note that the "standard" treatment described above may be inappropriate when viscosity varies strongly in space. More precise treatments, such as operator-dependent transfer and Galerkin approximation [9, 10], are expected to improve the robustness against the viscosity variation [16].

### 2.3 Performance of ACuTEMan: Impact of Adjustment in Multigrid

We measured a performance of our simulation code on the Earth Simulator. A model problem is a thermal convection of an isoviscous fluid with Rayleigh number $Ra = 10^7$ in a three-dimensional rectangular domain of aspect ratio $4 \times 4 \times 1$. The computational domain is divided into equally-spaced $1024 \times 1024 \times 256$ mesh divisions, and decomposed into $2 \times 32 \times 8$ subdomains in $x$-, $y$- and $z$-directions, respectively. The calculation is parallelized by a so-called "flat MPI" approach, and is carried out using 64 processor nodes (i.e. 512 PEs) of the Earth Simulator. In order to calculate the temporal evolution of thermal convection, we solve the energy equation (1) together with (2) and (3). We performed one thousand temporal integrations, starting from a purely conductive state plus small sinusoidal temperature perturbation. At each timestep, the equations for $\boldsymbol{v}$ and $p$ are solved by the multigrid method summarized in Figure 3, using $N_{\text{thres}} = 8$ grid levels and $\ell_{\text{thres}} = 4$. The mesh spacing is successively doubled as the grid level $\ell$ decreases, and

```
MPI Program Information:
========================
Note: It is measured from MPI_Init till MPI_Finalize.
      [U,R] specifies the Universe and the Process Rank in the Universe.

Global Data of 512 processes:          Min [U,R]            Max [U,R]         Average
=============================

Real   Time (sec)           :     2167.564 [0,79]      2168.180 [0,184]     2168.026
User   Time (sec)           :     2148.951 [0,398]     2161.454 [0,219]     2156.579
System Time (sec)           :        3.240 [0,231]        7.921 [0,280]        3.950
Vector Time (sec)           :     1886.794 [0,70]      1999.746 [0,0]       1907.307
Instruction Count           : 172586180925 [0,56]  204325670446 [0,0]   175903786565
Vector Instruction Count    :  45725028280 [0,1]    59885749447 [0,0]    46845002132
Vector Element Count        : 8972700070279 [0,295] 11284593905563 [0,0]   9115733866864
FLOP Count                  : 3141552699120 [0,1]   4245768380296 [0,0]   3222272951859
MOPS                        :     4219.400 [0,231]     5290.575 [0,0]       4286.788
MFLOPS                      :     1454.453 [0,1]       1965.394 [0,0]       1494.161
Average Vector Length       :      188.435 [0,0]        199.675 [0,1]        194.601
Vector Operation Ratio (%)  :       98.584 [0,124]       98.736 [0,0]         98.604
Memory size used (MB)       :      979.454 [0,1]       1043.454 [0,2]       1011.461


Overall Data:
=============

Real   Time (sec)           :     2168.180
User   Time (sec)           :  1104168.311
System Time (sec)           :     2022.255
Vector Time (sec)           :   976541.389
GOPS   (rel. to User Time)  :     2194.835
GFLOPS (rel. to User Time)  :      765.010
Memory size used (GB)       :      505.730
```

**Fig. 4** A summary of the overall performance of the ACuTEMan code. The values are reported by MPIPROGINF available on the Earth Simulator.

the mesh spacing is 1/2 on the coarsest grid level $\ell=1$. Figure 4 summarizes the overall performance of our code. In this case we achieved 765 Gflops using 512 PEs, and 18% of peak performance. We also found that our code has a moderate vectorization capability, with average vector length 194.6 and vector operation ratio 98.6%.

To demonstrate the impact of our multigrid implementation in detail, we show in Figure 5 the plots of the execution time $T_{\mathrm{smooth}}(\ell)$ per one smoothing step against grid level $\ell$. The values are separately plotted for $\ell \geq \ell_{\mathrm{thres}}$ and $\ell < \ell_{\mathrm{thres}}$ (the value for $\ell=1$ is not shown, because the grid level $\ell=1$ was not visited at all in this calculation). Also shown by a thick black line in the figure is a theoretical relation of $T_{\mathrm{smooth}}$ with $\ell$, estimated solely from the changes in problem sizes $(T_{\mathrm{smooth}} \propto 8^{\ell})$. By comparing the actual relations between $T_{\mathrm{smooth}}$ and $\ell$ with the theoretical one, we can estimate the influence of communication overhead. Indeed, the slope of the plot of $T_{\mathrm{smooth}}$ against $\ell$ is close to that of theoretical relation for $\ell < \ell_{\mathrm{thres}}$ where no communication takes place during the smoothing steps. For $\ell \geq \ell_{\mathrm{thres}}$, in contrast, the slope significantly deviates from the theoretical one as $\ell$ decreases. This implies that the influence of communication overhead becomes severer as the amount of arithmetic work decreases. In particular, the influence is quite severe for $\ell = \ell_{\mathrm{thres}}$.

From the comparison of $T_{\mathrm{smooth}}$ between $\ell = \ell_{\mathrm{thres}}(=4)$ and $\ell_{\mathrm{thres}}-1(=3)$, we can see the impact of the agglomeration technique on the efficiency of smoothing calculations on coarse grid levels. Let us first estimate a theoretical difference in $T_{\mathrm{smooth}}$ between $\ell=4$ and 3. The amount of arithmetic works for $\ell=3$ is 8 times smaller than that for $\ell=4$ owing to the reduction of problem sizes. On the other hand, the number of active PEs for $\ell=3$ is 512 times smaller than that for $\ell=4$. Taken together, $T_{\mathrm{smooth}}$ for $\ell=3$ is expected to be 64 times larger than that for $\ell=4$. Surprisingly, however, Figure 5 shows that $T_{\mathrm{smooth}}$ increases only slightly by changing from $\ell=4$ to 3. This discrepancy mainly comes from the difference in the influences of communication overheads between $\ell=4$ and 3. The smoothing steps for $\ell=3$ are free from communications, while those for $\ell=4$ are significantly influenced by the communication overheads due to the frequent communications among the active $N_{\mathrm{PE}}$ (=512) PEs. The decrease in execution time by the removal of communication dominates the potential increase by idling the majority of PEs.

In order to see the overall parallelization efficiency in our multigrid method, we show in Figure 6 the ratio of total execution time of smoothing steps for all grid levels to that of the entire smoothing calculations over one thousand temporal integrations. The portion of execution times spent in the smoothing steps for $\ell < \ell_{\mathrm{thres}}$ is indicated by shading. Figure 6 clearly shows that the overall parallelization efficiency is not severely spoiled by idling the

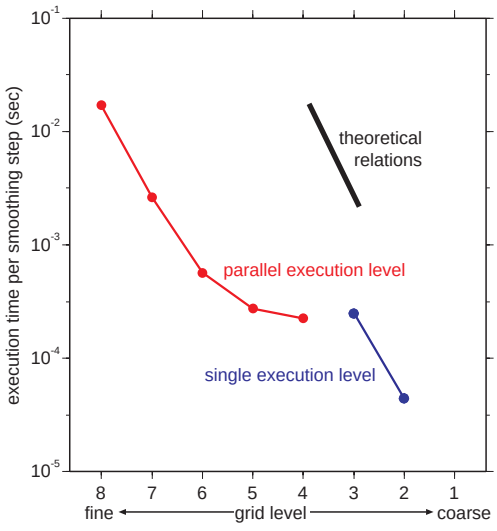


**Fig. 5** Plots of execution times of one smoothing calculation $T_{\mathrm{smooth}}$ against the grid level $\ell$ obtained for a calculation of $1024 \times 1024 \times 256$ mesh system using 512 processor elements (PEs) together with 8 grid levels and $\ell_{\mathrm{thres}}=4$. Shown in red are the values of the finer grid levels $(\ell \geq \ell_{\mathrm{thres}})$ where the smoothing calculations are carried out in parallel using all available PEs, while shown in blue are the values of $\ell < \ell_{\mathrm{thres}}$ where the calculations are done using one PE. We also show by a thick black line, for comparison, a theoretical relations between the execution time and grid level estimated simply from the changes in problem sizes $(T_{\mathrm{smooth}}(\ell) \propto 8^{\ell})$.

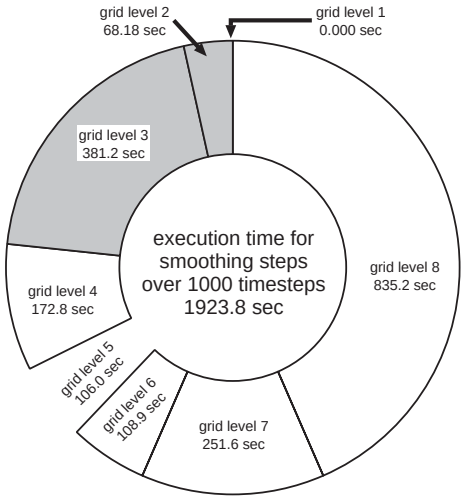


**Fig. 6** A graph showing the execution times spent in the smoothing steps for different grid levels and their ratios to the total execution times. The shaded portion indicates the execution times where the smoothing calculations are carried out using one processor element (PE) leaving other PEs idle.

majority of available PEs for $\ell < \ell_{\text{thres}}$. The smoothing steps for these grid levels take only about one-fourth of the total execution time, even though only one PE is responsible for these calculations. This is a direct consequence that these calculations are executed very efficiently owing to the removal of communication (see also Figure 5). We thus conclude that the agglomeration technique is a very effective strategy in enhancing parallelization efficiency of the multigrid method.

## 3. Discussion and concluding remarks

In this paper we introduced our numerical simulation code named "ACuTEMan" for solving mantle convection problems with strongly variable viscosity in a three-dimensional rectangular box. The essence of our code lies in the improvement of the solution method for the flow field of mantle convection, which is typically the most time-consuming part. In this part we employed a multigrid method, which is theoretically optimal for a solution method for large-scale elliptic equations, in combination with a special optimization for the Earth Simulator. We developed a smoothing algorithm named "ACuTE" for this problem [11], and demonstrated that this algorithm is suitable for vector and parallel computations. By optimizing the multigrid procedures using the agglomeration technique, we successfully obtained sufficient vector and parallel efficiency of our code using up to 64 processor nodes (i.e. 512 PEs) of the Earth Simulator. We are still continuing to develop our code toward a more "realistic simulation" of mantle dynamics, by incorporating the effects of solid-state phase transitions and the spatial variations in physical properties of mantle materials [17] (see Figure 7), and by applying our algorithm to a convection in a three-dimensional spherical geometry [18, 19].

The performance measurement described in Section 2.3 suggest that the agglomeration technique is quite efficient in optimizing the multigrid program. In this study the agglomeration technique is taken into account by activating only one processor element (PE) during the calculations on the grid level $\ell < \ell_{\text{thres}}$ while leaving other PEs idle. One must note here that an appropriate choice of $\ell_{\text{thres}}$ is crucial for the overall performance of parallelized multigrid. Suppose a case, for example, where the calculation presented in Figure 6 were performed with $\ell_{\text{thres}} = 5$. Since the problem size for $\ell = 4$ is eight times larger than that for $\ell = 3$, it would take about ($381.2 \times 8 \simeq$) 3000 seconds in the calculations for $\ell = 4$, which would have resulted in a significant increase in execution time. Unfortunately, however, it is impossible to derive a "golden rule" for the choice of $\ell_{\text{thres}}$. Since the optimal $\ell_{\text{thres}}$ is chosen to minimize the sum of the execution times for arithmetic work and those for communication, it depends, in principle, on the specific computers as well as on the smoothing algorithms. It is therefore necessary to find $\ell_{\text{thres}}$ by a trial-and-error manner for specific problems.

We believe that the solution technique for flow fields described here has a great impact on many research fields other than mantle convection studies. In the fields of earth sciences, efficient solvers of large-scale elliptic equations are required in many kinds of applications, such as incompressible fluid dynamics, potential problems, and (quasi-)static viscoelastic analysis. We dedicate this paper to Earth Simulator users who are seriously annoyed by the solution procedure of large-scale elliptic

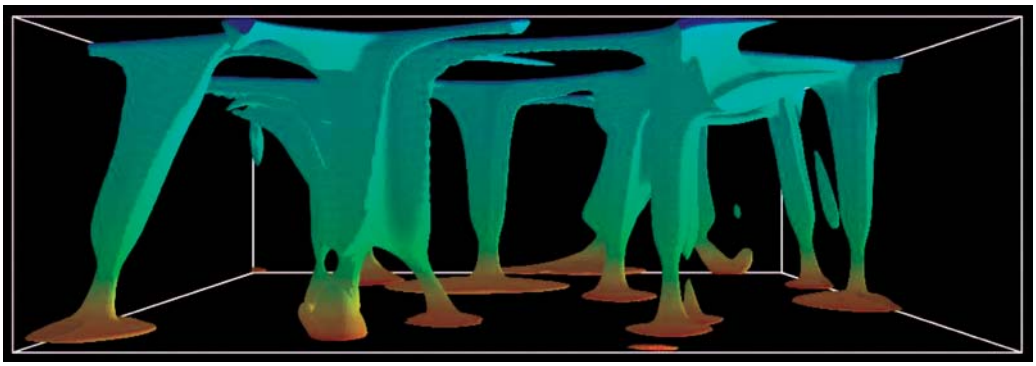

**Fig. 7** Example of numerical simulations performed by ACuTEMan. A thermal convection in a three-dimensional rectangular box of 3000 km height and aspect ratio of $6 \times 6 \times 1$ are calculated with $512 \times 512 \times 128$ mesh divisions. Temperature- and depth-dependence of viscosity and temperature-dependence of thermal diffusivity are included. An endothermic (having negative Clapeyron slope) and exothermic (positive slope) phase transitions are imposed at the depth of around 660 km and 2800 km, respectively. Shown in the figure are the isosurfaces of the deviation of nondimensional temperature from its horizontal average $\delta T = -0.05$ in a subdomain of $3 \times 3 \times 1$. One temporal integration step takes about 3.4 seconds for the calculation with 16 processor nodes (i.e. 128 PEs) of the Earth Simulator. See [17] for more details.

equations. Up to now, an excellent computational performance on the Earth Simulator has been reported only on the fields where a solution of elliptic equations is not involved [20, 21, 22, 23, 24]. We hope that our experience presented in this paper can help to broaden the horizon of a highly efficient numerical simulation of elliptic problems on the Earth Simulator.

## Acknowledgement

I thank Akira Kageyama and Tetsuya Sato for fruitful discussion and encouragement, and Yoshinori Tsuda and Ken'ichi Itakura for technical assistance. I also thank Prof. Takesi Yukutake for careful and thoughtful review. All of the calculations presented in this paper were done by the Earth Simulator at Japan Agency for Marine-Earth Science and Technology, as a part of the "Dynamics of Core-Mantle Coupled System" project.

(This article is reviewed by Dr. Takesi Yukutake.)

## References

[1] D. L. Turcotte and G. Schubert, *Geodynamics: Applications of continuum physics to geological problems*, 450 pp., John Wiley, New York, 1982.

[2] F. D. Stacey, *Physics of the Earth*, Third edition, 513 pp., Brookfield Press, Brisbane, Australia, 1992.

[3] D. L. Turcotte and E. R. Oxburgh, Finite amplitude convection cells and continental drift, *J. Fluid Mech.*, vol. **28**, pp. 29 – 42, 1967.

[4] D. P. McKenzie, J. Roberts, and N. O. Weiss, Convection in the Earth's mantle, *Tectonophysics*, vol. **19**, pp. 89 – 103, 1973.

[5] G. Schubert, D. L. Turcotte, and P. Olson, *Mantle convection in the Earth and planets*, 940 pp., Cambridge University Press, Cambridge, 2001.

[6] J. Weertman, The creep strength of the Earth's mantle, *Rev. Geophys.*, vol. **8**, pp. 145 – 168, 1970.

[7] S.-I. Karato and P. Wu, Rheology of the upper mantle: A synthesis, *Science*, vol. **260**, pp. 771 – 778, 1993.

[8] A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Math. Computation*, vol. **31**, pp. 333 – 390, 1977.

[9] P. Wesseling, *An introduction to multigrid methods*, 284 pp., Wiley, New York, 1992.

[10] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, 631 pp., Academic Press, San Diego, 2001.

[11] M. Kameyama, A. Kageyama, and T. Sato, Multigrid iterative algorithm using pseudo-compressibility for three-dimensional mantle convection with strongly variable viscosity, *J. Comput. Phys.*, vol. **206**, pp. 162 – 181, 2005.

[12] H. Schmeling and W. R. Jacoby, On modelling the lithosphere in mantle convection with nonlinear rheology, *J. Geophys.*, vol. **50**, pp. 89 – 100, 1981.

[13] M. Ogawa, G. Schubert, and A. Zebib, Numerical simulations of three-dimensional thermal convection in a fluid with strongly temperature-dependent viscosity, *J. Fluid Mech.*, vol. **233**, pp. 299 – 328, 1991.

[14] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, 197 pp., Hemisphere,Washington DC, 1980.

[15] A. J. Chorin, A numerical method for solving incompressible viscous flow problems, *J. Comp. Phys.*, vol. **2**, pp. 12 – 26, 1967.

[16] W.-S. Yang and J. R. Baumgardner. A matrix-dependent transfer multigrid method for strongly variable viscosity infinite Prandtl number thermal convection. *Geophys. Astrophys. Fluid Dyn.*, vol. **92**, pp. 151 – 195, 2000.

[17] M. Kameyama, D. A. Yuen, and A. Kageyama, High-resolution 3-D numerical studies on the interplay between variable thermal conductivity and post-perovskite phase transition, *Eos Trans. AGU*, vol. **85**, no. 47, Fall Meet. Suppl., Abstract MR23A – 0194, 2004.

[18] A. Kageyama and T. Sato, "Yin-Yang grid": an overset grid in spherical geometry, *Geochem. Geophys. Geosyst.*, vol. **5**, Q09005, doi:10.1029/2004GC000734, 2004.

[19] M. Yoshida and A. Kageyama, Application of the Yin-Yang grid to a thermal convection of a Boussinesq fluid with infinite Prandtl number in a three-dimensional spherical shell, *Geophys. Res. Lett.*, vol. **31**, L12609, doi:10.1029/2004GL019970, 2004.

[20] S. Shingu, H. Takahara, H. Fuchigami, M. Yamada, Y. Tsuda, W. Ohfuchi, Y. Sasaki, K. Kobayashi, T. Hagiwara, S.-I. Habata, M. Yokokawa, H. Itoh, and K. Ohtsuka, A 26.58 Tflops global atmospheric simulation with the spectral transform method on the Earth Simulator, In *Proceedings of the ACM/IEEE Supercomputing SC2002 conference*, 2002.

[21] H. Sakagami, H. Murai, Y. Seo, and M. Yokokawa, 14.9 TFLOPS three-dimensional fluid simulation for fusion science with HPF on the Earth Simulator, In *Proceedings of the ACM/IEEE Supercomputing SC2002 conference*, 2002.

[22] M. Yokokawa, K.-I. Itakura, A. Uno, T. Ishihara, and Y. Kaneda, 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator, In *Proceedings of the ACM/IEEE Supercomputing SC2002 conference*, 2002.

[23] D. Komatitsch, S. Tsuboi, C. Ji, and J. Tromp, A 14.6 billion degree of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the Earth Simulator, In *Proceedings of the ACM/IEEE Supercomputing SC2003 conference*, 2003.

[24] A. Kageyama, M. Kameyama, S. Fujihara, M. Yoshida, M. Hyodo, and Y. Tsuda, A 15.2 TFlops simulation of geodynamo on the Earth Simulator, In *Proceedings of the ACM/IEEE Supercomputing SC2004 conference*, 2004.