ATMOSPHERIC SCIENCE LETTERS
Atmos. Sci. Let. **13**: 29–35 (2012)
Published online 22 August 2011 in Wiley Online Library
(wileyonlinelibrary.com) **DOI:** 10.1002/asl.358

Royal Meteorological Society

Non-hydrostatic atmospheric cut cell model on a block-structured mesh

Hiroe Yamazaki* and Takehiko Satomura
Division of Earth and Planetary Sciences, Graduate School of Science, Kyoto University, Sakyo, Kyoto 606-8502, Japan

**Correspondence to:
H. Yamazaki, Division of Earth and Planetary Sciences, Graduate School of Science, Kyoto University, Kitashirakawa-Oiwake, Sakyo, Kyoto 606-8502, Japan.
E-mail: yamazaki_h@kugi.kyoto-u.ac.jp*

Received: 19 May 2011
Revised: 19 July 2011
Accepted: 1 August 2011

Abstract

A block-structured Cartesian mesh approach based on the Building-Cube Method is implemented into a 2D non-hydrostatic atmospheric cut cell model to obtain high near-ground resolution using Cartesian coordinates. A simple flux-matching algorithm is introduced that ensures mass conservation across varying grid resolutions in a subcycling time integration. Results of simple diffusion and advection problems show that the method produces sufficiently accurate results with high computational efficiency. The developed model successfully reproduces a flow over a semicircular mountain on a locally refined mesh around the mountain. The result agrees well with that using a uniformly fine mesh. Copyright © 2011 Royal Meteorological Society

Keywords: cut cell method; non-hydrostatic atmospheric model; Cartesian coordinates; mesh refinement; topographic effects

1. Introduction

Recently, Cartesian coordinates are drawing attention as an attractive choice for high-resolution atmospheric models that need to handle steep slopes in mountainous areas. They have the advantage of avoiding the errors because of the slantwise orientation of grid lines in models using conventional terrain-following coordinates (Thompson *et al*., 1985). A disadvantage of Cartesian coordinates is that high vertical resolution near the ground is expensive to attain over a wide range of topographic height (Walko and Avissar, 2008). Horizontal grid intervals must also be closely spaced at steep slopes to achieve high near-ground resolution.

A block-structured Cartesian mesh approach named Building-Cube Method (BCM) developed by Nakahashi (2003) provides an efficient way of locally refining a Cartesian grid at regions requiring higher resolution. In this method, a flow field is divided into a number of subdomains called ‘cubes’, and each cube has the same number of cells (Figure 1). The local computational resolution is therefore determined by the cube size, so that locally refined Cartesian grids at arbitrary boundaries can be obtained by refining the size of cubes near the boundaries.

The two-tiered data structure of cubes and cells provides several attractive features of the BCM. The use of a uniform Cartesian mesh in each cube allows the direct use of any existing code for Cartesian coordinates. Moreover, the same number of cells among all cubes provides an advantage for parallel computing since the load balance of each cube is equivalent. For example, Kim *et al*. (2007) investigated the parallel efficiency of the BCM using OpenMP. They reported that the performance of the parallelization was very close to the ideal one when 32 central processing units (CPUs) were used for the computations. The BCM has been applied to several problems of computational fluid dynamics including flow around an airfoil (e.g. Nakahashi *et al*., 2006; Kim *et al*., 2007) and around a circular cylinder (e.g. Takahashi *et al*., 2009). A similar approach to the BCM designed for spherical coordinates was proposed by Oehmke and Stout (2001) and Oehmke (2004), and applied to advection problems on the sphere by Jablonowski *et al*. (2006, 2009).

In this study, the BCM is modified and introduced to a 2D non-hydrostatic atmospheric model based on a Cartesian cut cell grid, named ‘Sayaca-2D’ (Yamazaki and Satomura, 2010). The main modifications lie in the time-stepping procedure and flux calculation at fine-coarse cube boundaries. Because the original BCM (Nakahashi, 2003) uses the same time step at all cubes, the time step must be small to stabilize the numerical integration of the finest cube and is applied to all cubes: overhead is added to the coarse cubes. Our method provides larger time steps at the coarse cubes and the solvers are subcycled at the fine cubes, like the method reported by Berger and Colella (1989), for example. Furthermore, a simple flux-matching procedure for the BCM data structure is derived and introduced at fine-coarse cube boundaries to ensure global mass conservation of the model. This Conserved Building-Cube Method (CBCM) enables Sayaca-2D to perform computationally efficient Cartesian-grid simulations with both high near-ground resolution and reasonable conservation characteristics.

Another difference of this study from Nakahashi (2003) is in the surface boundary representation. Although Nakahashi (2003) uses a staircase representation of boundaries, here we use a cut cell

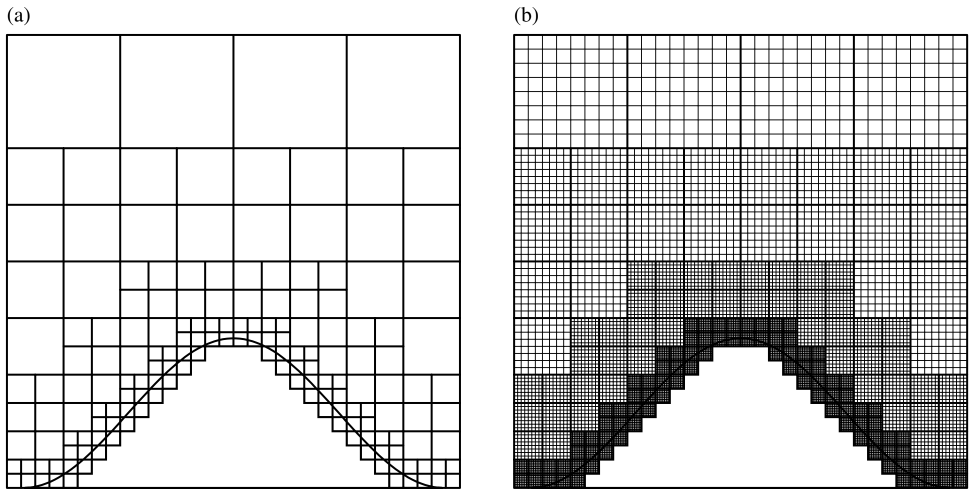

Figure 1. Schematic of (a) cubes and (b) cells around a cosine-shaped mountain for 2D computation. In this case, all cubes have 8×8 cells.

method recently developed for and implemented in Sayaca-2D. The cut cell method is based on a finite-volume discretization and uses a cell-merging approach (Yamazaki and Satomura, 2010). Note that it differs from the immersed boundary method based on a finite-difference discretization that was used with the BCM in some early studies, for example, in the study by Kamatsuchi (2007).

Section 2 provides descriptions of the basic CBCM design principles, including mesh generation, time-stepping procedure, parallelization and flux-matching algorithm. In Section 3, we show the superiority of CBCM through numerical examples of simple diffusion and advection problems. Then the performance of Sayaca-2D with CBCM is examined by comparing its flow result over a mountain to that using Sayaca-2D with a uniformly fine mesh throughout the entire domain.

2. Numerical method

2.1. Overall procedure

Following the BCM (Nakahashi, 2003), the computational mesh of CBCM is generated by two steps: cube generation and cell generation in each cube. First, cubes are generated in a manner similar to the quadtree method (Berger, 1986) where the size of a cube becomes small with getting closer to the terrain surface (Figure 1(a)). The size differences among cubes are adjusted to guarantee a uniform 2 : 1 mesh resolution at fine-coarse cube boundaries. The cubes that are located completely inside of the topography are removed and not used for the computation.

After cube generation, a Cartesian mesh of equal spacing and equal number of cells in each cube is generated (Figure 1(b)). For cells in the cubes that cross the topography, intersections to the terrain surface are checked. Although the BCM simply determines whether each cell is inside or outside of a staircase boundary, here we identify cells cut by the topography and determine the intersections of the terrain surface with the faces of the cells. Then, small cut cells are merged with neighboring cells to avoid severe restrictions on time steps by the Courant–Friedrichs–Lewy condition. Details of this cell-merging procedure are found in the work of Yamazaki and Satomura (2010). To prevent cell merging between cells that belong to different sizes of cubes, the size of the cubes near the terrain surface is readjusted, if necessary, so that both merged and non-merged cut cells are inside of the finest region.

All flow calculations are performed in each cube independently. Information is transferred between adjacent cubes through ghost cells that are added beyond the boundary of each cube, which allows merging of cells that belong to the different finest cubes. Note that all ghost regions are at the same resolution as the inner domain of the cube. Although the first-order interpolation is used as in the BCM for exchanging information between different sizes of cubes, a flux-matching algorithm introduced in the following subsection ensures conservation across varying grid resolutions.

Time step for cubes at each refinement level is chosen as:

$$\Delta t(l) = 2^{-l} \Delta t_{\mathrm{g}} \qquad (1)$$

Here the refinement level l ranges from 0 (coarsest) to $l_{\max}$ (finest) and the global time step $\Delta t_{\mathrm{g}} \equiv \Delta t(0)$. We start with the integration of cells in the cubes at level $l_{\max}$, and the cells at level l are updated with two subcycling steps while those in cubes at level $l - 1$ are advanced in one time step. No temporal interpolation of the coarse boundary data during the subcycling steps is applied in CBCM. To prevent the errors due to time suspending at coarse boundaries from contaminating the results, two more cells beyond the cube boundary are predicted at each fine cube and are used in the calculation during the subcycling steps.

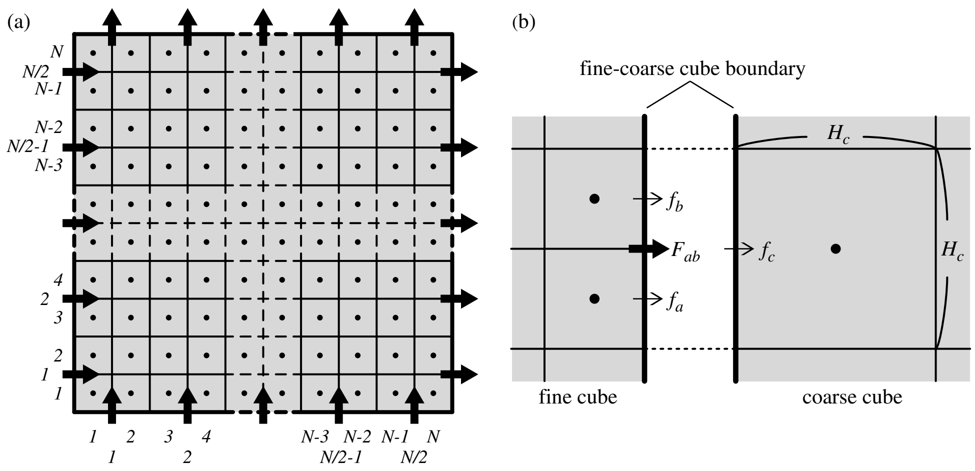

Figure 2. Cube boundary flux and flux-matching algorithm. The advected quantity is defined at the cell centers. Solid and thick lines describe the boundaries of cells and cubes, respectively. Solid and thick arrows indicate the fluxes at cell boundaries and cube boundaries, respectively.

The parallelization of CBCM is straightforward. Load balancing is achieved by distributing an equal number of cubes at each refinement level for each processor. To keep the load balance in the presence of topography at the maximum refinement level, we compute solutions for all cells in the finest cubes, and then set the values at cells that are located bellow the terrain surface to zero. The overhead required to calculate values of the underground cells are sufficiently low because cells in completely underground cubes are excluded from computation (Figure 1). For efficient parallelization, a sufficient number of cubes are needed to avoid inequality of distributed numbers of cubes among processors as demonstrated by Takahashi *et al.* (2009).

2.2. Flux-matching algorithm

To achieve conservation on a hierarchically refined grid, flux matching at fine-coarse grid interfaces is imperative (e.g. Berger and Colella, 1989; Jablonowski *et al.*, 2006). With a subcycling time-stepping scheme, at a fine-coarse grid interface the numerical flux on the coarse grid must equal the fluxes on the fine grid accumulated during the subcycling steps.

This process can be easily incorporated into the BCM framework by introducing the cube boundary flux as shown in Figure 2(a). Here we assume that the advected quantity is defined at the cell centers. If a cube has $N \times N$ cells, $N/2$ cube boundary fluxes are defined on each side of the cube boundary. These fluxes are used to correct the coarse cell flux to match the accumulated fine cell fluxes at fine-coarse cube boundaries.

For example, consider the case shown in Figure 2(b). This figure shows the locations of the fine cell fluxes f_a and f_b, coarse cell flux f_c and cube boundary flux F_{ab} per unit time and length at a fine-coarse cube boundary. In the BCM, fluxes at cube boundaries are just calculated using ghost cell information in each cube, and f_c is not consistent with the accumulated fluxes f_a and f_b. Here we calculate F_{ab} to satisfy the following equation:

$$\int_{n\Delta t(l_c)}^{(n+1)\Delta t(l_c)} F_{ab}\mathrm{d}t = \frac{1}{H_c}\left\{\frac{H_c}{2}\left(\int_{n\Delta t(l_c)}^{(n+1/2)\Delta t(l_c)} f_a\mathrm{d}t + \int_{(n+1/2)\Delta t(l_c)}^{(n+1)\Delta t(l_c)} f_a\mathrm{d}t\right) + \frac{H_c}{2}\left(\int_{n\Delta t(l_c)}^{(n+1/2)\Delta t(l_c)} f_b\mathrm{d}t + \int_{(n+1/2)\Delta t(l_c)}^{(n+1)\Delta t(l_c)} f_b\mathrm{d}t\right)\right\} \tag{2}$$

where l_c and H_c are the refinement level and cell length of the coarse cube, respectively. By evaluating the integral as

$$\int_{n\Delta t(l_c)}^{(n+1)\Delta t(l_c)} \psi\mathrm{d}t = \psi^n\Delta t(l_c) \tag{3}$$

where $\psi^n \equiv \psi(t)$ at $t = n\Delta t(l_c)$, the cube boundary flux is calculated during the subcycling steps as

$$F_{ab}^n = \frac{1}{\Delta t(l_c)}\frac{1}{H_c}\left\{\frac{H_c}{2}\left(f_a^n\frac{\Delta t(l_c)}{2} + f_a^{n+1/2}\frac{\Delta t(l_c)}{2}\right) + \frac{H_c}{2}\left(f_b^n\frac{\Delta t(l_c)}{2} + f_b^{n+1/2}\frac{\Delta t(l_c)}{2}\right)\right\} = \frac{1}{4}(f_a^n + f_b^n + f_a^{n+1/2} + f_b^{n+1/2}) \tag{4}$$

and then overrides the coarse cell flux f_c. This algorithm ensures mass conservation on the condition that the density is defined at the cell centers. The run-time overhead is as low as about 5% of a typical run time in the heat diffusion experiment in the following section.

3. Results

In this section, we present results of three numerical experiments in two dimensions using CBCM. First, a

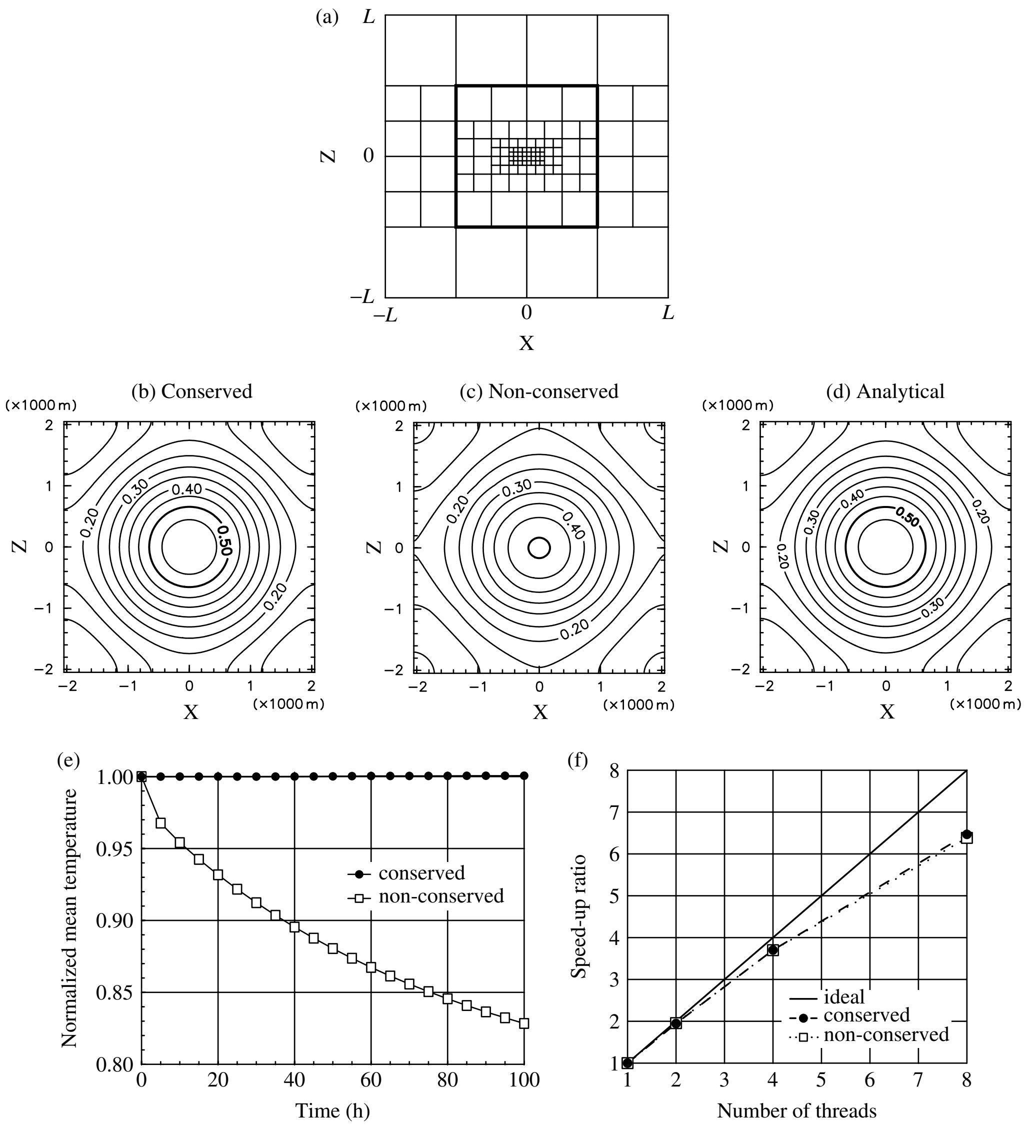

Figure 3. Settings and results of the heat diffusion experiment. (a) 2D computational domain. Solid lines are the cube boundaries. Thick line indicates the boundary of the initially heated region. (b) Simulated temperature field with the flux-matching algorithm and (c) that without the flux-matching algorithm. (d) Temperature field calculated from the analytical solution. The calculation time and contour interval in (b), (c) and (d) are 100 h and 0.05 K, respectively. (e) Time change of the mean temperature in the results in (b) and (c) normalized by that in the initial state. (f) Comparison of speed-up ratio with increasing the number of threads.

simple heat diffusion problem is solved to demonstrate the conservation property and parallelization performance of the method:

$$\frac{\partial T}{\partial t} = \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2}\right) \quad \text{for} \quad -L \le x \le L,\ -L \le z \le L \tag{5}$$

where boundary conditions are

$$\frac{\partial T}{\partial x} = 0 \quad \text{at} \quad x = \pm L, \qquad \frac{\partial T}{\partial z} = 0 \quad \text{at} \quad z = \pm L \tag{6}$$

indicating that the temperature averaged over the entire domain is held constant over time. The domain consists of 112 cubes shown in Figure 3(a) and each cube has 64×64 cells. The half domain size $L =$ 2048 m and the cell length of the largest cubes $H =$ 16 m. The temperature field is initially defined as

$$T(x,z) = \begin{cases} 1 & \text{if} \quad |x| \le L/2 \quad \text{and} \quad |z| \le L/2 \\ 0 & \text{if} \quad |x| > L/2 \quad \text{or} \quad |z| > L/2 \end{cases} \tag{7}$$

and calculated for 100 h.

Figure 3(b) and (c) shows the simulated temperature fields using the method with and without the flux-matching algorithm described in Section 2.2, respectively. Referring to the analytical solution to the problem shown in Figure 3(d), the experiment with flux matching produces a sufficiently accurate solution, while the simulated amplitude without flux matching is much smaller than that in the analytical solution. Figure 3(e) shows the time change of the mean temperature normalized by that in the initial state. It clearly shows that the mean temperature remains constant to the machine precision in the result with the flux-matching algorithm, while it decreases to less than 85% of the initial value after 100 h of integration in the result without flux matching. Finally, the speed-up ratio is estimated to verify the scalability (Figure 3(f)). Here OpenMP is used to parallelize the code and the computations are performed on a Linux PC which has two 6-core AMD Opteron CPUs. The parallel efficiency is about 80% at eight processors in both results with and without the flux-matching algorithm. This indicates that the matching procedure implemented here does not alter the parallel efficiency of the BCM algorithm. Although the current parallel performance is not as good as that described in the work of Kim *et al.* (2007), this is the first trial of parallelization and the code has some room for performance improvement.

As the second experiment, an advection equation for a passive scalar ϕ,

$$\frac{\partial \phi}{\partial t} + \frac{\partial c_x \phi}{\partial x} + \frac{\partial c_z \phi}{\partial z} = 0 \quad \text{for} \quad -L \le x \le L,\ -L \le z \le L \tag{8}$$

is integrated to examine the accuracy of CBCM. Here $L = 2048$ m is used as in the first experiment. The cyclic boundary condition is applied in both x and z directions and both c_x and c_z are 1 m s^{-1}. The initial

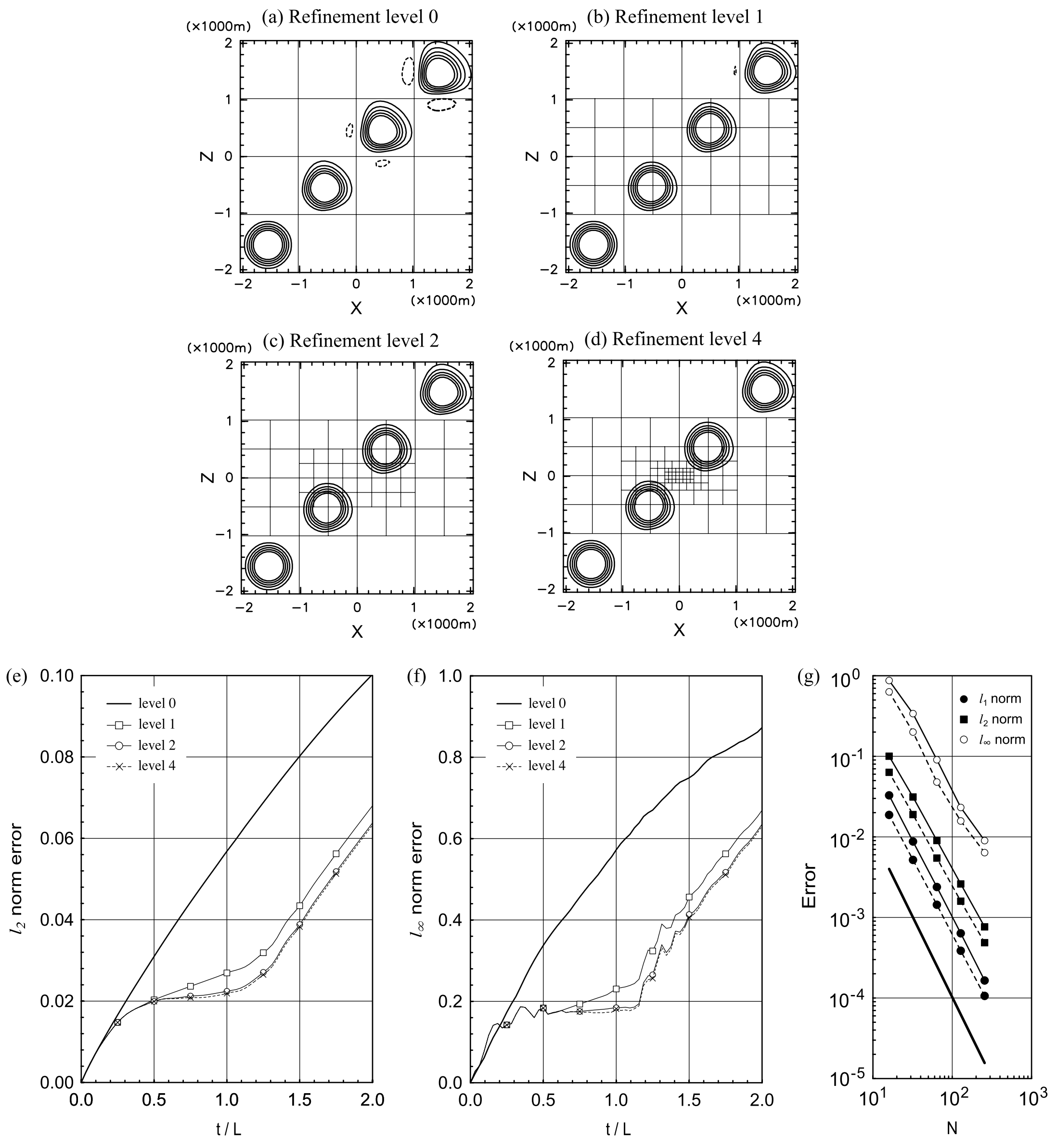

Figure 4. Results of the advection experiment. (a)–(d) Simulated cosine bell transported over cubes with refinement levels 0, 1, 2 and 4, respectively. Solid lines are the cube boundaries and each cube has 16 × 16 cells. Snapshots are taken at $t = 0$, $L/2$, L and $3L/2$ from the bottom left to the top right of each figure, respectively. Thick lines and thick-dashed lines indicate positive and negative values of the cosine bell, respectively. The contour interval is 0.2. (e) Time traces of the l_2 and (f) l_∞ norms of the errors of the cosine bell for different refinement levels. (g) Variations of l_1, l_2 and l_∞ norms of the errors with the number of cells on a side of each cube. Solid lines and dashed lines indicate the variation of errors for cases (a) and (d), respectively. Thick line corresponds to the second-order accurate convergence.

distribution of ϕ is defined as

$$\phi(x,z) = \begin{cases} 1+\cos(\pi r/R) & \text{if} \quad r < R \\ 0 & \text{if} \quad r \geq R \end{cases} \tag{9}$$

with radius $R = L/2$ and

$$r = \sqrt{(x-x_c)^2 + (z-z_c)^2} \tag{10}$$

denotes the distance between a position (x, z) and the initial center of the cosine bell $(x_c, z_c) = (-3L/4, -3L/4)$. Thus the cosine bell is advected diagonally across the domain and reaches its initial position after $2L$ seconds. The solution is compared to the analytical solution.

Figure 4(a)–(d) shows four snapshots of the cosine bell at $t = 0$, $L/2$, L and $3L/2$ from the bottom left to the top right of each figure with refinement level 0, 1, 2 and 4, respectively. Here every test case uses 16×16 cells in each cube. Although the shape of the cosine bell is largely distorted in Figure 4(a) due to the relatively coarse resolution, the distortion is successfully reduced when implementing refined regions. Moreover, no visible distortions of the shape of the cosine bell are observed as it passes over fine-coarse cube boundaries. To quantitatively examine the errors due to the abrupt resolution changes, time traces of the l_2 and l_∞ norms of the errors are calculated and illustrated as shown in Figure 4(e) and (f), respectively. Even though the time trace of the l_∞ norm shows little spikes when the cosine bell is transported over fine-coarse cube boundaries, these results confirm that the benefit due to the higher resolution within the finer cubes exceeds the errors due to the abrupt resolution changes. Figure 4(g) shows the variations of l_1, l_2 and l_∞ norms of the errors with the number of cells on a side of each cube in the cases of Figure 4(a) and (d). These errors are calculated in the results after one revolution. While the convergence rate for l_∞ norms in the case with four refinement levels drops slightly below that with no refinement, approximately second-order convergence rates are found for l_1 and l_2 norms in both cases. This result demonstrates that first- and second-order interpolations at the interface between different sizes of cubes conserve the global second-order spatial accuracy of CBCM. In addition, the run time of the experiment in the case of Figure 4(d) is as low as about 72% of that required by non-subcycling calculation.

Finally, a mountain wave experiment is performed by Sayaca-2D with CBCM to demonstrate the performance of the method with cut cells in atmospheric applications. Here a semicircular mountain of radius $r = 1$ km is located at 15 km from the left end of

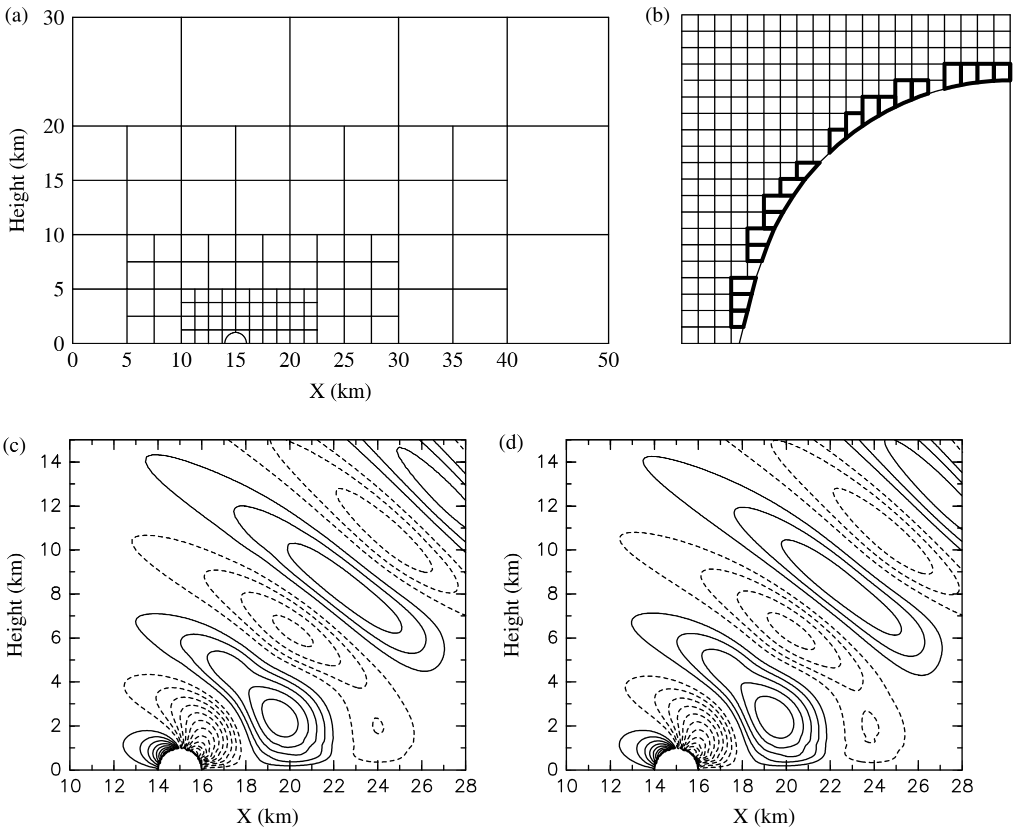

Figure 5. Settings and results of the mountain wave experiment. (a) Cube boundaries and (b) the cell boundaries in a cube that intersects with the mountain surface. Thick lines in (b) indicate the boundaries of the merged cells. (c) Vertical velocity reproduced by Sayaca-2D with CBCM using four different cell length: 500, 250, 125 and 62.5 m. (d) Vertical velocity reproduced by Sayaca-2D with a uniformly fine mesh of 62.5 m. The integration time and contour interval in (c) and (d) are 1 h and 1 m s^{-1}, respectively. Solid and dashed lines in (c) and (d) indicate positive and negative values, respectively.

the lower boundary. A constant horizontal velocity, $U = 10$ m s^{-1}, and the constant Brunt–Väisälä frequency, $N = 0.01$ s^{-1}, are initially imposed on the entire domain. The lower and lateral boundary conditions are free-slip and cyclic, respectively. The width of the domain is set to 200 km to be large enough to prevent cyclic lateral boundaries from contaminating the simulated results. The height of the domain is 30 km and a sponge layer (Klemp and Lilly, 1978) is placed higher than 22.5 km to avoid reflecting the gravity wave at the rigid top boundary.

The cube configuration used in this experiment is shown in Figure 5(a). The total number of cubes is 144 with 20×20 cells in each cube. The minimum cell length near the mountain is 62.5 m and the maximum cell length in the far field is 500 m. The rest of the domain not shown in Figure 5(a) is filled with the largest cubes. Figure 5(b) shows an enlarged view of the Cartesian mesh near the mountain, showing the cut cell representation of terrain surface with cell merging.

Figure 5(c) and (d) shows the vertical velocity fields over the semicircular mountain calculated by Sayaca-2D with CBCM and with a uniform mesh of 62.5 m throughout the entire domain, respectively. There are no visible distortions of mountain wave patterns in the result of Figure 5(c) associated with the change of mesh resolution. Although in the region of coarse resolution the simulated amplitude in Figure 5(c) is slightly smaller than that in Figure 5(d), Sayaca-2D with CBCM reproduces practically the same result compared to that with a uniformly fine mesh. At the same time, the number of cells with CBCM is as low as 3.75% of that with a uniformly fine mesh. These results lead to the conclusion that Sayaca-2D with CBCM reproduces a reasonably accurate flow over the mountain with a substantially low computational cost, thereby demonstrating the advantage of the CBCM in combination with cut cells for flows over complex topography including steep slopes.

4. Conclusion

To obtain high resolution near the ground with high computational efficiency in Cartesian coordinate models, a block-structured Cartesian mesh approach CBCM was developed, and was implemented to a non-hydrostatic atmospheric cut cell model Sayaca-2D.

CBCM employs a subcycled time step to minimize the computational overhead in time integration. In addition, the method ensures mass conservation at fine-coarse grid interfaces by introducing a simple flux-matching algorithm. The results of simple diffusion and advection problems showed that the method has sufficient conservation property, high computational efficiency and global second-order accuracy.

To demonstrate the performance of Sayaca-2D with CBCM, the test of flow over a semicircular mountain was performed using a locally refined mesh around the mountain. The model reproduced a smooth and accurate mountain wave comparable with the uniformly fine-grid computation.

Acknowledgements

The authors thank Prof Tetsuya Takemi and Mr Ryuji Yoshida at the Disaster Prevention Research Institute, Kyoto University, for their support and helpful comments. The first author also thanks Dr Phillip Blakely and Mr Leif Denby at the Cavendish Laboratory, University of Cambridge, as well as two anonymous reviewers for their suggestions and discussions. This study was supported in part by a Grant-in-Aid for JSPS Fellows, for JSPS Bilateral Joint Research Project, for Scientific Research B-22340137 of MEXT, Japan and for the Global COE Program GCOE-ARS of Kyoto University. Some figures were drawn by the GFD-DENNOU Library.

References

Berger MJ. 1986. Data structures for adaptive grid generation. *SIAM Journal on Scientific and Statistical Computing* **7**: 904–916.

Berger MJ, Colella P. 1989. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* **82**: 64–84.

Jablonowski C, Herzog M, Penner JE, Oehmke RC, Stout QF, van Leer B, Powell KG. 2006. Block-structured adaptive grids on the sphere: advection experiments. *Monthly Weather Review* **134**: 3691–3713.

Jablonowski C, Oehmke RC, Stout QF. 2009. Block-structured adaptive meshes and reduced grids for atmospheric general circulation models. *Philosophical Transactions of the Royal Society A* **367**: 4497–4522.

Kamatsuchi T. 2007. Turbulent flow simulation around complex geometries with Cartesian grid method. Proceedings of 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 2007, AIAA paper: 2007–1459.

Kim LS, Nakahashi K, Jeong HK, Ha MY. 2007. High-density mesh flow computations by Building-Cube Method. *Journal of Mechanical and Science Technology* **21**: 1306–1319.

Klemp JB, Lilly DK. 1978. Numerical simulation of hydrostatic mountain waves. *Journal of Atmospheric Science* **35**: 78–107.

Nakahashi K. 2003. Building-cube method for flow problems with broadband characteristic length. In *Computational Fluid Dynamics 2002*. Armfeld S, Morgan P, Srinivansan K, Eds, Springer Verlag: Berlin; 77–81.

Nakahashi K, Kitoh A, Sakurai Y. 2006. Three-dimensional flow computations around an airfoil by building-cube method. Proceedings of 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 2006, AIAA paper: 2006–1104.

Oehmke RC. 2004. High performance dynamic array structures. PhD thesis, University of Michigan, USA; 93 pp.

Oehmke RC, Stout QF. 2001. Parallel adaptive blocks on a sphere. Proceedings of 10th SIAM Conference on Parallel Processing for Scientific Computing, Portsmouth, VA, SIAM, CD-ROM.

Takahashi S, Ishida T, Nakahashi K. 2009. Parallel computation of incompressible flow using building-cube method. *Parallel Computational Fluid Dynamics 2007, Lecture Notes in Computational Science and Engineering*, **67**: 195–200.

Thompson JF, Warsi ZUA, Mastin CW. 1985. *Numerical Grid Generation: Foundations and Applications*. Elsevier Science Publishing Company: North-Holland.

Walko RL, Avissar R. 2008. The Ocean-Land-Atmosphere Model (OLAM). Part 2: formulation and tests of the nonhydrostatic dynamic core. *Monthly Weather Review* **136**: 4045–4062.

Yamazaki H, Satomura T. 2010. Nonhydrostatic atmospheric modeling using a combined Cartesian grid. *Monthly Weather Review* **138**: 3932–3945.