Ohno and Kageyama
Earth, Planets and Space (2021) 73:158
https://doi.org/10.1186/s40623-021-01485-2

Earth, Planets and Space

TECHNICAL REPORT

Open Access

In-situ visualization library for Yin-Yang grid simulations

Nobuaki Ohno[1*] and Akira Kageyama[2]

Abstract

The visualization of computer simulations is currently undergoing a transition from post-hoc to in-situ visualization in which visualization processes are applied, while the simulation is running. The selection of an appropriate method or tool is essential to efficiently perform in-situ visualization in parallelized large-scale computer simulations that run on supercomputers. Although some generic in-situ visualization libraries are available, they are overengineered for certain geophysical simulations. In this study, we focus on spherical simulations using the Yin-Yang grid. Computer simulations that use the Yin-Yang grid are gaining popularity in geophysics. We propose an in-situ visualization method dedicated to the Yin-Yang grid simulations and demonstrate its effectiveness through sample simulations.

Keywords: Scientific visualization, In-situ visualization, Yin-Yang grid, Computational fluid dynamics

Introduction

In large-scale geophysical simulations, visualization is commonly applied to numerical data saved in a disk system after performing simulation. Such a visualization style, called post-hoc visualization, would be difficult to apply in the near future. The size of the numerical data to be transferred and stored for the post-hoc visualization becomes unrealistically large as the scale of the simulation increases. Even for a simulation with a moderate grid size, say 1024^3, the post-hoc visualization requires 8 GB of disk space to save just one scalar field (in the double precision floating point numbers), for just one simulation time step. It is not uncommon today to carry out simulations with a grid size of the order of 2048^3 or 4096^3. The required disk space for saving one scalar in those cases is 64 or 512 GB, respectively. If one tries to analyze multiple scalar fields or vector fields for multiple time steps, the output data would flood the disk system.

Another style of visualization, called in-situ visualization, is garnering attention among simulation researchers (Ma et al. 2007; Ross et al. 2008; Ma 2009). It is a method to apply visualization, while the simulation is running. The data to be stored after the simulation with in-situ visualization are two-dimensional images, rather than three-dimensional floating-point numbers. In addition to the dimensional reduction, one can apply compression algorithms based on arithmetic coding to further reduce the size of the image data.

In-situ visualization plays important roles in large-scale simulations. Major application programs for general-purpose visualization are now equipped with in-situ libraries or tools, such as Catalyst (Ayachit et al. 2015) for ParaView, and libsim (Whitlock et al. 2011) for VisIt. ParaView and VisIt have rich set of visualization capabilities. These applications have been developed based on the Visualization Tool Kit (VTK) (Schroeder et al. 2006). The OpenGL graphics library is used for hardware-accelerated rendering in the VTK.

The OpenGL functions can be executed in a computing environment without graphics processing units (GPUs) using the Off-Screen Mesa (OSMesa) library. Because various libraries, such as OpenGL, VTK, and OSMesa, are involved, it becomes difficult, if not impossible, to optimize them on newly developed supercomputers with special accelerators including vector processors.

*Correspondence: ohno@sim.u-hyogo.ac.jp
[1] Graduate School of Information Science, University of Hyogo, Kobe 651-2197, Japan
Full list of author information is available at the end of the article

One of the authors (N. O.) has developed a library with minimal in-situ visualization capabilities (Ohno and Ohtani 2014). The library, VISMO, is written in the Fortran 2003 language. All the visualization methods are implemented using the language with no additional libraries. VISMO has been used in various simulations, especially fluid turbulence simulations (Miura et al. 2019; Miura 2019; Yoshida et al. 2019; Kageyama et al. 2020).

The original VISMO adopts simulation data defined on a Cartesian coordinate system, whereas spherical geometry is essential in geoscientific simulations. We developed a grid system, Yin-Yang grid, for spherical simulations (Kageyama and Sato 2004). It is a type of overset grid system that comprises two congruent component grids, Yin and Yang, which are combined to cover a spherical surface with partial overlaps. The boundary values on each component are set through mutual interpolations according to the standard overset methodology (Chesshire and Henshaw 1990).

The Yin-Yang grid was applied to geodynamo simulation (Kageyama et al. 2008; Miyagoshi et al. 2010), which was the original motivation for this new grid system, mantle-convection simulation (Yoshida and Kageyama 2004; Kameyama et al. 2008; Tackley 2008), global circulation models of the atmosphere (Peng et al. 2006; Baba et al. 2010; Qaddouri and Lee 2011; Li et al. 2013; De Grandpré et al. 2016), solar-dynamo simulations (Hotta et al. 2014; Mabuchi et al. 2015), and astrophysical simulations (Wongwathanarat et al. 2010; Raymer 2012).

We note the speciality of the visualization of spherical geometry and the importance of Yin-Yang grid. When a target field of physical quantity is distributed spaciously in all (x, y, and z) directions, one could map the data onto the cartesian coordinate system before applying visualization in the cartesian grid. However, such a cartesian mapping is impractical for data defined in a spherical shell, which is the geometry commonly used in the earth sciences. For example, shown in Fig. 1 are varied thermal convection in spherical shells between two concentric spheres with radii r_i and r_o with different shell depths $d = r_o - r_i$.

If we simply map a sphere with radius r into a cube to visualize the spherical shell data, the volume is almost doubled even if we could ignore the inner spherical void. (Volume ratio of a cube and its inscribed sphere is about 1.91.) If we accept the doubled volume cost, we would have to discretize the cube with much higher spatial resolution than that in the sphere, to describe the smooth spherical surface and physical variables in the spherical boundary layers. If we keep the same grid resolution in the cartesian map, it would cause a serious visual degradation in the spherical boundary layers. Because the degradation is caused by the geometrical difference between the cube and the sphere, we can avoid it if we use the spherical coordinate system with the latitude–longitude grid. However, the latitude–longitude grid has the problem of the grid concentration near the north and south poles; unreasonably high grid density near the poles reduces visualization efficiency of any kind. The Yin-Yang grid is free from the grid concentration.

We developed a visualization tool on the Yin-Yang grid about a decade ago (Ohno and Kageyama 2009). This tool, Yin-Yang visualizer, was for the post-hoc visualization produced through Yin-Yang grid simulations.

In this paper, we propose a simple, easy-to-use, and versatile method of incorporating in-situ visualization

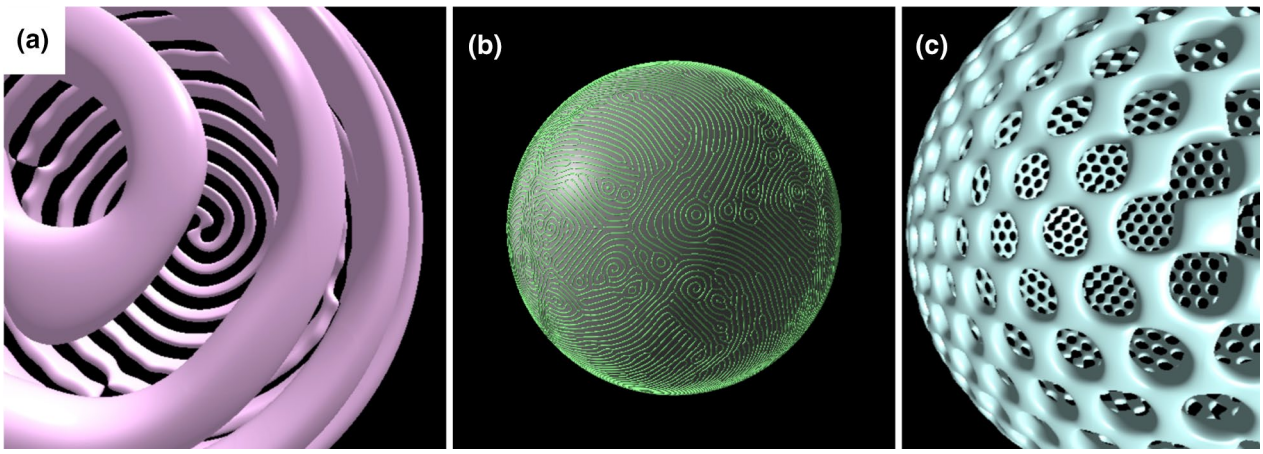

Fig. 1 Varied convection patterns visualized by VISMO-YY in spherical shells with different depths $d = r_o - r_i$; **a** spiral roll ($d = 0.1$); **b** quasi-straight rolls with dislocations ($d = 0.01$); and **c** quasi-regular polyhedron ($d = 0.05$)

into simulation programs based on the Yin-Yang grid. We have developed a new library and call it VISMO-YY.

Visualization strategy

Basic algorithm

We intentionally focus in this work only on the following minimum visualization algorithms; isosurface rendering, slicer rendering, volume rendering, arrow glyphs, and streamlines. Another key point in our visualization strategy is that all the visualization algorithms are implemented based on the same technique, i.e., ray-casting (Levoy 1990). It is shown in our in previous paper (Ohno and Kageyama 2009) that the ray-casting algorithm is suitable for visualization on CPUs on supercomputer systems. We take the same approach to the in-situ visualization with no graphics hardware (GPUs), or software rendering, in the development of VISMO-YY.

The visualization method that we propose in this paper is intrinsically parallelized. It is supposed to be applied to a parallel simulation in which the simulation domain is decomposed into multiple subdomains. The visualization algorithms implemented in VISMO-YY are parallelized based on processes (Message Passing Interface, MPI) and threads (OpenMP). Each MPI process allocated to the local simulation subdomain calls subroutines of VISMO-YY during the simulation. The visualization is applied to render a subimage for the allocated local subdomain to each MPI process. OpenMP is used for thread-based parallelization to generate the subimages, which are composited to a final image before proceeding to the next simulation step.

In the ray-casting algorithm, the line of sight is tracked from the point of view to each pixel direction of the screen. The physical variables to be visualized are integrated along the ray in the simulation space. Collisions of the ray with surfaces, such as pre-defined materials or computationally generated objects, are also detected along the ray. Shading is applied to the surfaces to provide three-dimensional appearances.

Because VISMO-YY is a flexible library, we can also use it for the post-hoc visualization. Before we describe the performance of VISMO-YY as an in-situ visualization library, here we describe the ray-casting procedures for each visualization algorithm by applying the library to a test data set in the post-hoc visualization style. The test data set is produced by a thermal–convection simulation in a rotating spherical shell using the Yin-Yang grid with eight subdomains (four for Yin and four for Yang). The spherical shell is assumed to be rotating with a constant angular velocity along the z-axis. The data size for each domain is about 21 MB. The total size for all domains is 168 MB.

Volume rendering

The first visualization method described is volume rendering, which visualizes a scalar field as semi-transparent objects. The volume rendering with the ray-casting is straightforward when the simulation domain is not split: A ray that is running through a pixel of image is cast from a viewing point toward the target scalar field. The scalar values sampled with fixed intervals are transformed into colors and opacities using a lookup table called transfer function, and the colors are integrated along the ray with opacities as the weight.

We need some ingenuity to integrate the ray when the simulation domain is divided into multiple subdomains for parallel simulation, because a single ray may span multiple subdomains. Here we divide the integral of a ray into pieces. In other words, each single ray is divided into multiple sub-rays for every subdomains on the ray. The sub-rays for a pixel are combined into the single ray, taking the summing order into consideration, to obtain the final image. This procedure of parallel ray-casing is performed for all the pixels for an image with OpenMP.

Figure 2a shows an example of visualization by this parallel volume rendering that is implemented as a subroutine of VISMO-YY. The semi-transparent orange columnar objects in Fig. 2a represent the volume rendering of the axial vorticity ω_z in the test data set. Here, $\omega_z = \boldsymbol{\omega} \cdot \boldsymbol{e}_z$, with $\boldsymbol{\omega} = \nabla \times \boldsymbol{v}$, where $\boldsymbol{v}$, $\boldsymbol{\omega}$, and $\boldsymbol{e}_z$ denote the flow velocity, vorticity, and unit vector along the z-direction.

Isosurface rendering

For the isosurface rendering, a surface in the simulation space for a specified level of a scalar field is constructed through the ray-casting. The scalar-field values on consecutive sampling points on a ray are compared with the isosurface level. If the level is between two adjoining values, the ray segment connecting the two points crosses the isosurface. The convection columns, which are visualized through the orange volume rendering of ω_z in Fig. 2a, are presented by the isosurface rendering in Fig. 2b.

Note that the isosurfaces shown in Fig. 2b are shaded. All the visualized objects in VISMO-YY are basically shaded in the same procedure. The surface of an object can be light or dark for a uniform color. A part of the object appears bright when light emitted from a source strikes the object at that part and is then reflected to the viewpoint. They are non-recursive shading; the reflection of a ray from an object is calculated only once for the first reflection. We adopt the standard reflection models in the computer graphics, i.e., the ambient, diffuse, and specular reflections. To calculate the reflections, the parameters of the light sources, parameters for the

materials, and normal vectors of each object are required. Among these, the parameter set of the light sources and materials is specified by the user of VISMO-YY through a configuration file, which is described in the next section. A normal vector is calculated by vector gradient of the target scalar field. In the volume rendering, normal vectors obtained by the same procedure are used for the shading. The objects in other visualization methods, such as arrow glyphs and stream tubes, are calculated on the basis of their geometric shapes. As the shading is a parallel process, it is separately performed for each pixel through thread parallelization using OpenMP.

Slicer rendering

In slicer rendering, we extract a slice out of a scalar field for visualization. A slice plane $ax + by + cz + d = 0$ is analytically specified by the four coefficients a, b, c, and d in the configuration file. An example is depicted as a slanted disk in Fig. 2b. In this method, we first calculate the intersection point of a ray and slice plane in the data region. Subsequently, we assign a color to the ray on the basis of the physical value of the target scalar field at the cross-sectional point and a transfer function. The scalar field and transfer function are specified in a configuration file by the user. Repeating this procedure for every ray allocated to each pixel, we obtain a subimage for the MPI process. Even if there is more than one slice plane on a line of sight, the hidden-surface treatment is appropriately performed; only the frontmost section is shown.

In Fig. 2b, the green part in the disk corresponds to the region with almost no axial vorticity ω_z. The red (blue) color in the disk indicates positive (negative) ω_z on the disk.

Streamline rendering

Streamline rendering is used to visualize vector fields. It could also be called streamtube, because each streamline is in fact a tubed curves with a radius. The shades of the tube enhance the visibility of the streamline. The streamline is obtained through the numerical integration of the target vector field using the standard 4th-order Runge–Kutta method.

Sample streamlines are observed as colored coils in both panels in Fig. 2. It is moderately tricky to trace a streamline of a vector field defined in separated subdomains. VISMO-YY traces the streamlines across multiple subdomains. When a streamline leaves a specific subdomain and enters an adjacent subdomain, VISMO-YY invokes MPI to send and receive the position of the tracking point of the streamline. The tube of a streamline is constructed using a sequence of elemental cylinders and spheres, which are rendered using the ray-casting algorithm.

Arrow glyphs

Another visualization method for vector fields is the arrow glyphs, in which arrows are placed on a plane specified by the configuration file. A sample of the arrow-glyph visualization is shown in Fig. 2a, in which the glyphs are placed on an oblique plane that cuts the spherical shell. In VISMO-YY, one can specify multiple planes for the glyphs. The arrows comprise pairs of cylinders and cones, which are rendered using the ray-casting algorithm.

Program design

We implemented the above visualization methods in VISMO-YY, which is a library provided as a Fortran module for the users. It is easy to use VISMO-YY for in-situ visualization in a simulation. All you have to do is to add a few lines to the simulation program and to specify the parameters related to the visualization in plain text in the configuration file.

In the simulation code, the user writes use vismo at the beginning of the main program or the module in which the in-situ visualization is supposed to be applied. The user calls subroutines of VISMO-YY in the program/module to set data arrays and coordinates. VISMO-YY supports double and single precision data. The collocated data defined on the Yin-Yang grid points are assumed. Staggered grid data is not supported. An example of the subroutine calls with descriptive comments is presented below

.

```
use vismo
........
! Initialization of VISMO-YY
call vsmInit(mpi_comm_world, myrank, psize, "configfile.vsm")

! Setting arrays and its grid size
call vsmAddScalar(scalar, RDSIZE, THSIZE, PHSIZE)
call vsmAddVector(vectx, vecty, vectz, RDSIZE, THSIZE, PHSIZE)

! Initialization of coordinates. dp means double precision
rad(1) = 0.5_dp; rad(2) = 1.0_dp      ! rad(1): min radius, rad(2):max radius
rad(3) = (rad(2) - rad(1))/(RDSIZE-1) ! rad(3): dr
call vsmInitCoords(vismo__Yin, rad) ! vismo__Yin or vismo__Yang must be specified.

! Setting coordinates
! by specifying global and local grid sizes, the positions of corners, and the
    spacing.
call vsmSetUniCoord(n, gridcorner, dx)
call vsmSetLocalUniCoord(ln, lx0, rln, localgridcorner, dx)

! Preparation for visualization
call vsmPrepareVis
........
! <begin of computation loop>
! Visualization
call vsmVisualization(timestep) ! timestep is an integer number.
! <end of computation loop>
........
! Finalization of VISMO-YY
call vsmFinalize
```

The subroutines whose name starts with "vsm" are those of VISMO-YY. In this example, nine subroutines of VISMO-YY are called. If there are more variables to be visualized, vsmAddScalar or vsmAddVector should be called for them.

In the configuration file, the user specifies visualization methods and their parameters (such as isosurface and its level), light sources, cameras, etc. The master process of VISMO-YY loads the file in the beginning and scatters the information to all the other processes to conduct the in-situ visualization in accordance with it. One can specify the parameters via multiple files when the parameters are long. An example of configuration file of VISMO-YY is presented below. The user writes the value for the (capitalized) keyword after the words (or in between the corresponding word pairs). For example, the image size (width and height) is specified after IMAGESIZ

```
IMAGESIZE 512 512

BGCOLOR 0.8 0.8 0.8

LIGHT
 PARALLEL
 POSITION 10.0 20.0 5.0
 AMB 0.35 0.35 0.35
 DIF 0.75 0.75 0.75
 SPEC 0.5 0.5 0.5
END_LIGHT

CAMERA
 POSITION 1.7 1.7 1.7
 FRONT -0.57735 -0.57735 -0.57735
 UP 0 0 1
 FOVY 45.0
 NEARFAR 0.01 10
END_CAMERA

SCAL_COLORMAP
 SCALNUM 1
 EQUALLY_SPACED
 0.00000 0.06200 0.00000 1.00000 0.00000
 0.00004 0.04607 0.00000 1.00000 0.00000
 0.00008 0.03014 0.00000 1.00000 0.00000
 0.00012 0.01421 0.00000 1.00000 0.00000
 0.00016 0.00000 0.00172 1.00000 0.00000
 0.00020 0.00000 0.01765 1.00000 0.00000
 ........

 0.01000 1.00000 0.00000 0.00000 0.00000
END_SCAL_COLORMAP

VISUALIZATION

ISOSURFACE
 SCALNUM 1
 LEVEL 0.005
 AMB 0.80 1.0 0.60
 DIF 0.85 1.00 0.55
 SPEC 0.80 0.9 0.60
 SHIN 80
END_ISOSURFACE

SLICE
 SCALNUM 1
 EQ 0.0 0.0 1.0 0.0
 AMB 0.4
 DIF 0.9
 SPEC 0.4
 SHIN 10.0
END_SLICE

LIGHT 1
INNERSPHERE

END_VISUALIZATION
```

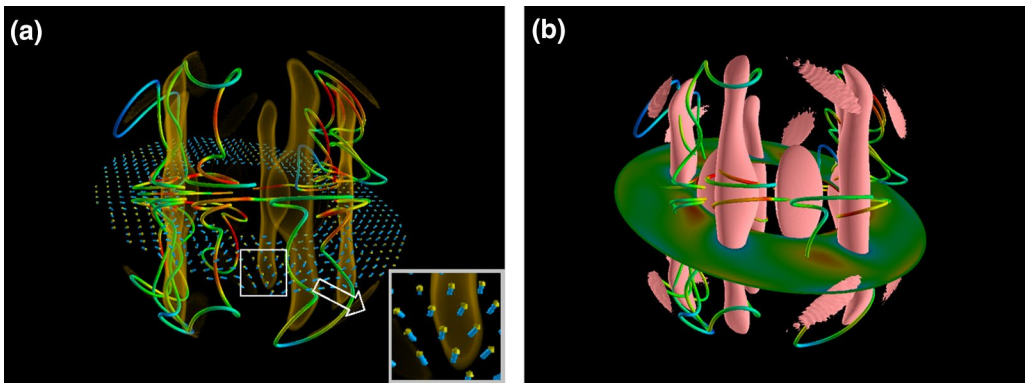

Fig. 2 Sample (post-hoc) visualization by VISMO-YY applied to numerical data defined on the Yin-Yang grid; **a** Axial vorticity of the thermal convection in a rotating spherical shell is visualized by the volume rendering (orange). Flow velocity is visualized by stream tubes (colored coils) and arrow glyphs. **b** Axial vorticity is visualized by isosurface and slicer rendering. Flow velocity is visualized by stream tubes

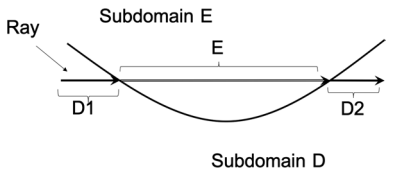

Fig. 3 Illustration of a ray passing through subdomains D and E. The rays for local domains are divided

Most parameters in the configuration file have default values. If you are not familiar with the lighting model in computer graphics (CG), you can skip specifying the material and light properties in the configuration file, such as ambient (AMB), diffusive (DIF), and specular (SPEC) light reflection. For example, without specifying its reflection properties, the LIGHT 1 is a gray directional light.

The configuration file for VISMO-YY has to be prepared in advance. It is theoretically possible to change visualization parameters, such as the background color, levels of isosurfaces, and so on, while a simulation job is running. We will implement the dynamic control of visualization parameters in the future.

A common problem in the in-situ visualization is the loss of interactivity. One has to specify the visualization parameters, such as isosurface levels, camera position, and viewing direction, in advance. To retain the interactivity in the in-situ visualization, there are

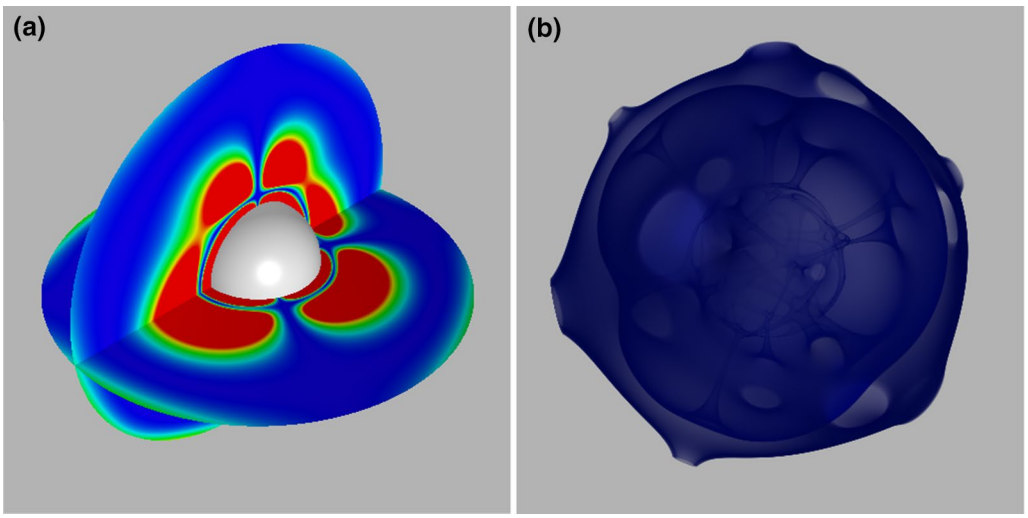

Fig. 4 Slice planes and volume rendering; **a** Slice planes visualization of the enstrophy density. **b** Volume Rendering of the enstrophy density. Those figures are obtained by the in-situ visualization of thermal convection in a deep shell

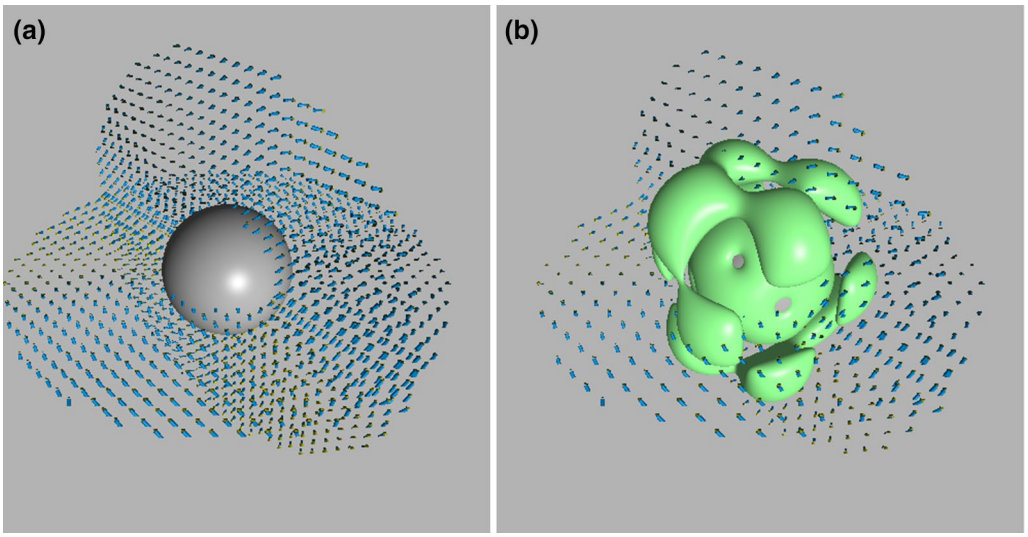

Fig. 5 Arrow glyph and isosurface visualization; **a** Arrow glyph visualization of convection velocity on two perpendicular planes. **b** Combined image of the isosurface visualization of the enstrophy density and the arrow glyph visualization of the flow velocity

various proposals, such as particle-based in-situ visualization (Kawamura et al. 2016) and proxy image method (Tikhonova et al. 2010a, b; Ye et al. 2013). For supercomputer system with GPUs, libraries for interactive steering, such as ISAAC are available (Matthes et al. 2016). We proposed interactive viewing method of in-situ generated visualization videos, "4-Dimensional Street View (4DSV)" (Kageyama and Yamada 2014; Kageyama and Sakamoto 2020). The key idea of 4DSV is to generate myriads of images with various visualization parameters

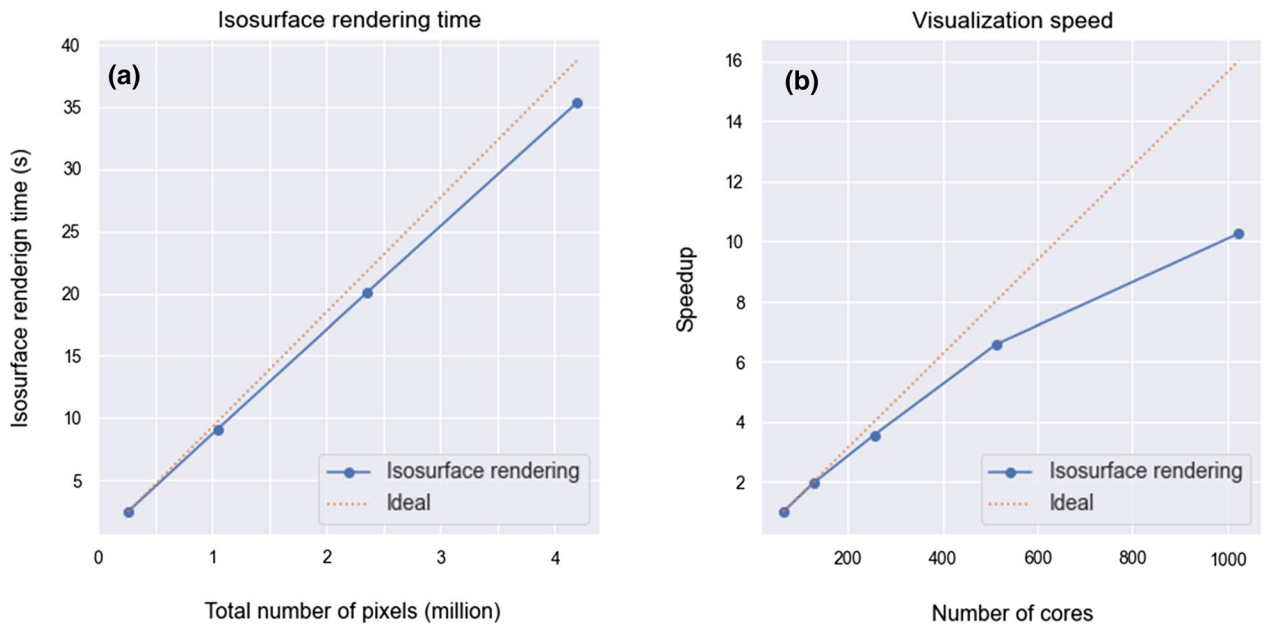

Fig. 6 Performance tests of VISMO-YY; **a** Time for isosurface visualization by VISMO-YY as a function of total number of pixels. The time linearly depends on the total pixel number, because the underlying algorithm is the ray-casting. Time for volume rendering also linearly depends on the total pixel number. Other visualization methods are less burdensome. **b** Strong scaling of VISMO-YY for isosurface visualization when the simulation size is moderate. It has a good parallelization performance up to 1024 cores with this grid size

and interactively explore the output "video space". We have already incorporated VISMO, the original cartesian version, to the 4DSV visualization of simulations in rectangular geometry (Kageyama et al. 2020; Kageyama and Sakamoto 2020). It is straightforward to apply VISMO-YY to Yin-Yang grid simulation and interactively change the visualization parameters in the framework of 4DSV.

Here, we briefly mention the technical issues associated with parallel rendering. Parallel visualization is relatively easy when multiple objects to be rendered are completely opaque. Different MPI processes can independently render the opaque objects in their own subdomains. The depth, or the distance from the viewpoint to the opaque object, which is also called the z-value, is incorporated into the rendered subimage. After all the MPI processes finish the rendering, the subimages are gathered and superimposed into a single image based on the z-value. Only the object closest to the viewpoint appears in each pixel of the image and other objects are hidden.

When translucent objects exist, however, the order of superposition matters. The volume rendering in which translucent color distribution is integrated along the ray belongs to this case. We need special care for the integration when we deal with curved coordinate system, such as Yin-Yang grid. Suppose a ray that goes through a subdomain (say D in Fig. 3) which is managed by a certain MPI process. In the curved coordinate system, the ray may re-enter D (possibly for multiple times, depending on the configuration) after leaving from D. Between the leaving and re-entering of D, the ray may run through another subdomain (say E) that is managed by another MPI process. In this case, we need a trick to obtain a consistent volume rendering along the ray. Notably, we cannot simply superimpose the two partial integrals in the subdomains D and E in each MPI process. Instead, we attain partial integrals in subdomain D (say D1 and D2), and superimpose the values in the order of D1 + E + D2 or its reverse, i.e., D2 + E + D1. In short, the divided rays for the local domain are divided further in the volume rendering in Yin-Yang grid data. This technique is similar to the one adopted for the volume rendering in unstructured grid system (Max et al. 1990).

The in-situ visualization images generated by VISMO-YY are stored one after another on the disk system on the supercomputer system, while the simulation program is running. The image format can be specified as PNG, PPM or BMP.

Applications: thermal convection in spherical shells

In this section, we explain the application of VISMO-YY for the intended purpose, i.e., in-situ visualization, and its performance for practical simulations using the Yin-Yang grid.

Simulation model

The target simulation, to which in-situ visualization is integrated, is again the thermal convection in a spherical shell, but with no rotation this time. We solve the thermal convection of a fluid confined in a spherical shell between two concentric spheres; outer sphere of radius $r = r_o$ and inner sphere $r = r_i$. The viscosity and thermal

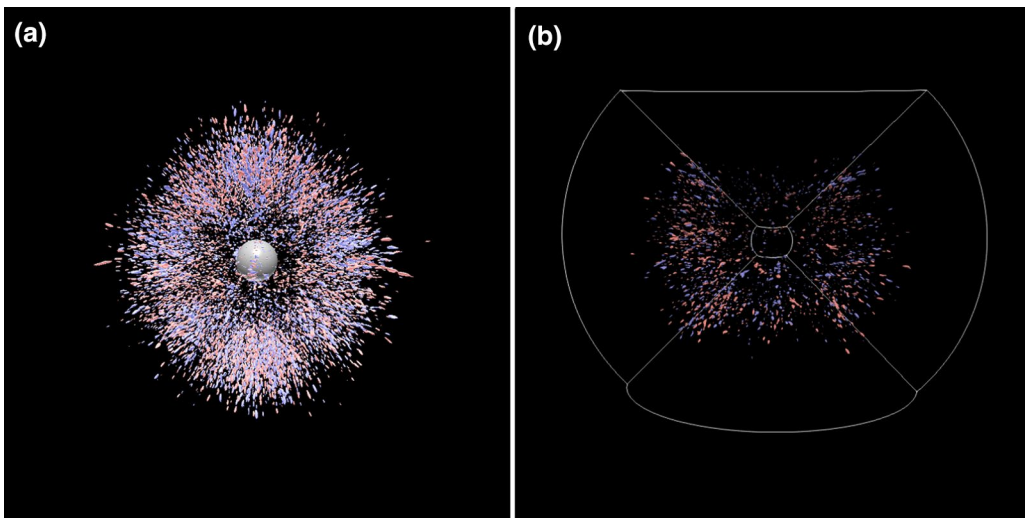

Fig. 7 Isosurfaces of the radial component of convection flow in a deep spherical shell: **a** Full spherical in-situ visualization by VISMO-YY. **b** Partial, post-hoc, visualization by ParaView with a workstation (32 GB main memory). Because of the memory limit, we could not load the whole spherical data

diffusivity of the fluid are constant; their ratio (Prandtl number) is unity. We assume a central gravity force and fixed temperatures T_o and T_i on the boundaries at $r = r_o$ and $r = r_i$. Because $T_o < T_i$, thermal convection sets in if the Rayleigh number Ra of the system is greater than the critical value.

Figure 4 shows an in-situ slicer rendering and volume rendering applied to convection simulation with the Yin-Yang grid. Herein the convection layer is relatively deep; $r_i = 0.3$ and $r_o = 1.0$. The total grid size is $N_r \times N_\theta \times N_\phi \times 2 = 201 \times 202 \times 606 \times 2$ with N_r, N_θ, and N_ϕ are grid sizes in radial, latitudinal, and longitudinal directions, respectively. The final factor 2 is for Yin- and Yang-components. We show in Fig. 4a an in-situ visualization of slicer rendering. We can place multiple slices in VISMO-YY.

The inner spherical surface, depicted as the gray ball in Fig. 4a, can be switched on and off using the configuration file. VISMO-YY can show the inner and/or outer spheres with surface/mesh.

Figure 4b shows an in-situ visualization of the volume rendering of enstrophy density. The semitransparent appearance of the volume rendering helps us observe the three-dimensional distribution of the scalar field.

One aspect we did not mention thus far regarding the features implemented in VISMO-YY is the background color. It is sometimes useful to change the background, which is black by default, to lighter colors. In Fig. 4, we set the background as gray in the configuration file. The gray background helps to comprehend the three-dimensional structure of the convection cell visualized through the bluish volume rendering.

Figure 5a shows convection-flow visualization by the arrow glyphs. In VISMO-YY, the arrow glyphs are placed on a plane specified in the configuration file. In this case, two perpendicular planes are set: one for the equatorial plane and the other for the meridian plane. The density of the arrows and size of each glyph can be controlled using the configuration file. The planes of arrow glyphs reveal part of the distribution of the flow.

Applying multiple visualization methods at once is straightforward with VISMO-YY. By specifying the methods with their parameters in the configuration file, we can automatically obtain multiple in-situ visualizations, resulting in superimposed images with appropriate ordering as well as hiding. Figure 5b depicts an example of such a visualization in which the isosurface rendering of enstrophy density and the arrow glyph rendering of flow velocity are combined. The pale green objects represent the isosurfaces. The density of the arrow glyphs in Fig. 5b is lower than that in Fig. 5a.

Visualization performance

The coupled simulation code that generated Figs. 4 and 5 were executed on SGI Standard-Depth Server C2112-4GP3 (CPU: Intel Xeon E5-2650v3 2.3GHz (10 cores) ×2, Memory: 128GB / node) using 512 CPU cores of 32 nodes. The total number of MPI processes was 256 and 2 OpenMP threads per MPI process. We calculated 50, 000 steps. VISMO-YY visualized the data, for every 1, 000 steps, and output 4 kinds of visualization images with the resolution 512×512 pixels.

The times required for simulation and visualization were 73,223 s and 283 s, respectively. Only about 0.38 % of the total time is used for the visualization. The low time cost proves that VISMO-YY can be a practical library for in-situ visualization.

The cost depends on the image resolution. In general, the rendering time of VISMO-YY linearly depends on the total number of pixels of images. To confirm this, we show a plot in Fig. 6a, as a function of the total number of pixels, the rendering time for isosurface visualization with the same data used to make Fig. 2. Figure 6a verifies the linear dependence on the total number of pixels. According to this estimate, even if we apply VISMO-YY visualizations with the pixel size of 2048×2048, it would take about $0.38 \times 16 \sim 6\%$ of the total simulation time to generate visualization images. Another critical indicator of VISMO-YY's performance is parallel scaling. We measured strong scaling of VISMO-YY for the same convection simulation with a relatively small size of total computational grid $N_r \times N_\vartheta \times N_\varphi \times 2 = 401 \times 402 \times 1206 \times 2$. We applied in-situ visualization of v_r isosurfaces. Figure 6b summarises results for different number of cores from 64 to 1024. Fixing the number of MPI processes as 64, we increased OpenMP threads from 1 to 16. It took 0.056% of the total simulation time in the case of 1024 cores. This graph shows that VISMO-YY has a practically good parallelization performance with 1024 cores even with this moderate computational grid size. For more extensive simulations, VISMO-YY would linearly scale to more parallelization numbers.

The test simulations as well as in-situ visualization results presented so far are relatively simple ones. Here we demonstrate the feasibility of VISMO-YY in much larger simulation, The target simulation is the same problem (thermal convection in a spherical shell), but with larger scale with higher parallelization. The spherical shell is deep; $r_i = 0.1$ and $r_o = 1.0$ that leads to higher Rayleigh number. The grid size is $N_r \times N_\theta \times N_\phi \times 2 = 1023 \times 1002 \times 3006 \times 2$. The number of MPI processes is 15, 360. The computation

was performed on Plasma Simulator (SX-Aurora TSUBASA A412-8) at National Institute for Fusion Science, Japan. Figure 7a shows a snapshot of isosurface rendering of radial velocity $v_r = \pm c$ by VISMO-YY. The isosurface level c is set as a half of the maximum value of $|v_r|$ at the snapshot time. The positive flow $v_r = +c$ is rendered as pink surfaces and negative flow $v_r = -c$ light blue. The image size of the in-situ visualization is 1024×1024 pixels. Among 1800 steps of numerical integration of the convection, we produced one in-situ visualization image. The time for the 1800 integration was 299 s, while the in-situ visualization with VISMO-YY was 57 s. We also performed the simulation with the same grid size ($N_r \times N_\vartheta \times N_\varphi \times 2 = 1023 \times 1002 \times 3006 \times 2$) on HPE Apollo 2000 Gen10 (Xeon Gold 6248 2.5 GHz 40 cores per node) for 1800 steps. The number of MPI processes is 512, with 4 OpenMP threads per MPI process. It took about 12000 s for the simulation, while it took only 3.0 s using this computer system to generate the in-situ visualization of isosurfaces of v_r with the same image resolution (1024×1024). The performance ratio (57 s/3.0 s) indicates that there should be plenty of room for optimization of VISMO-YY for vector-type computer systems.

For comparison with post-hoc visualization, we have tried to perform the same isosurface visualization of v_r of the above mentioned large scale simulation data. We used ParaView for this comparison, aiming at the same visualization as Fig. 7a. The purpose of the comparison is to roughly estimate the cost for post-hoc visualization when a special graphics system is unavailable. It is possible that utilizing a more powerful hardware system will significantly improve the visualization performance of ParaView.

For this post-hoc visualization test, we used DELL Precision 5550, a mobile workstation with Intel Core i7-10850H and 32GB memory. The data size of v_r (downsized to single precision) for just one time step is $4\,\mathrm{B} \times \mathrm{N_r} \times \mathrm{N}_\theta \times \mathrm{N}_\phi \times 2 \sim 24.65\,\mathrm{GB}$. Transferring the data from Plasma Simulator to a local PC took more than 11 minutes for just one time step data. We apply the ParaView visualization on the Yin-Yang grid, using the vtkStructuredGrid format. We need to hold additional three float data for each grid position, i.e., x, y, and z coordinates. Therefore, the total data size needed for the isosurface visualization amounts to $4 \times 24.65 = 98.6$ GB, which is beyond the main memory of the workstation.

Giving up the complete realization of Fig. 7a with ParaView, we have visualized only a part of the Yin-grid data, discarding the Yang-part, as to $N_r \times N_\theta \times N_\phi = 1023 \times 1002 \times 1850$. The total data size of the extracted data with the coordinates information by vtkStructuredGrid format is about 28 GB, which is close to the practical limit of the workstation.

It took about 70 s to load the 28 GB data to ParaView, and about 40 s to render the isosurfaces shown in Fig. 7b. In total, it took about 110 s to obtain one snapshot. If we were to visualize the whole Yin-Yang data (98.6 GB), it would take about $110 \times 3 = 330$ s. The performance of 57 s by VISMO-YY is a stark contrast.

Let us note that the above test is just for one scalar field. If we were to visualize multiple scalar fields at once, or to apply vector field visualizations, such as stream lines or arrow glyphs, the memory burden on the post-hoc visualization becomes much more serious. In short, the post-hoc visualization for this scale is impractical, even if we could make use of larger main memory in the workstation. It would almost impossible to adopt the post-hoc approach to the visualization for many time steps due the storage and time costs.

Conclusion

In large-scale simulations, post-hoc visualization is being replaced by in-situ visualization, in which visualization is applied, while the simulation runs. Geophysical simulations are no exception to this trend. Special tools are required to apply in-situ visualization on a supercomputer system. Because the present supercomputer systems are composed of myriads of processors, parallel visualization is indispensable. For supercomputer systems without GPUs, the ability to make the best use of CPUs for graphics is also essential. In the in-situ visualization of geophysical simulations in large scale, additional difficulties exist stemming from the spherical geometry. The Yin-Yang grid is used in spherical simulations in several geophysical problems.

In this study, we propose an approach to the in-situ visualization of simulations with the Yin-Yang grid. The minimum required visualization methods, such as isosurfaces, volume rendering, and streamlines, are realized by a single and simple algorithm, i.e., ray-casting. The ray-casting is suitable for parallel rendering on supercomputers without GPUs. We implemented them as a library, VISMO-YY. VISMO-YY needs only Fortran compiler, not other basic libraries, such as VTK and OSMesa.

Abbreviations

VTK: Visualization Toolkit; GPU: Graphics Processing Unit; OSMesa: Off-Screen Mesa; MPI: Message Passing Interface; CG: Computer Graphics; 4DSV: 4-Dimensional Street View.

Acknowledgements

This work was supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" and "High Performance Computing

Infrastructure" in Japan (Project ID: jh190007-NAH). A part of this work was performed on "Plasma Simulator" (NEC SX-Aurora TSUBASA) of NIFS with the support and under the auspices of the NIFS Collaboration Research program (NIFS15KNSS062). Some parts of the computation in this work were done using the facilities of the Center for Cooperative Work on Data science and Computational science, University of Hyogo.

Authors' contributions

NO developed VISMO-YY. AK developed the Yin-Yang simulation code. The authors jointly integrated VISMO-YY into the simulation code. Both the authors equally contributed to prepare the manuscript, to perform the in-situ visualization, and to discuss the results. Both authors read and approved the final manuscript.

Funding

This work was supported by Grant-in-Aid for Scientific Research (KAKENHI) 16K00173 and 17H02998.

Availability of data and materials

VISMO-YY can be downloaded from N.O.'s web page (https://vizlab.sakura.ne.jp/en/vismo.en.html).

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Graduate School of Information Science, University of Hyogo, Kobe 651-2197, Japan. [2]Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan.

Received: 15 September 2020 Accepted: 20 July 2021
Published online: 05 August 2021

References

Ayachit U, Bauer A, Geveci B, O'Leary P, Moreland K, Fabian N, Mauldin J (2015) ParaView catalyst: enabling in situ data analysis and visualization. Proceedings of the first workshop on in situ 485 infrastructures for enabling extreme-scale analysis and visualization—ISAV. pp 25–29, https://doi.org/10.1145/2828612.2828624

Baba Y, Takahashi K, Sugimura T, Goto K (2010) Dynamical core of an atmospheric general circulation model on a Yin-Yang grid. Mon Weather Rev 138:3988–4005. https://doi.org/10.1175/2010MWR3375.1

Chesshire G, Henshaw WD (1990) Composite overlapping meshes for the solution of partial differential equations. J Comput Phys 90:1–64. https://doi.org/10.1016/0021-9991(90)90196-8

De Grandpré J, Tanguay M, Qaddouri A, Zerroukat M, McLinden CA (2016) Semi-Lagrangian advection of stratospheric ozone on a Yin-Yang grid system. Mon Weather Rev 144:1035–1050. https://doi.org/10.1175/MWR-D-15-0142.1

Hotta H, Rempel M, Yokoyama T (2014) High-resolution calculations of the solar global convection with the reduced speed of sound technique I the structure of the convection and the magnetic field without the rotation. Astrophys J 786:24. https://doi.org/10.1088/0004-637X/786/1/24

Kageyama A, Sato T (2004) "Yin-Yang grid": an overset grid in spherical geometry. Geochem Geophys Geosyst 5:Q09005. https://doi.org/10.1029/2004GC000734

Kageyama A, Miyagoshi T, Sato T (2008) Formation of current coils in geodynamo simulations. Nature 454(7208):1106–1109. https://doi.org/10.1038/nature07227

Kageyama A, Sakamoto N (2020) 4D street view: a video-based visualization method. PeerJ Comput Sci 6:e305. https://doi.org/10.7717/peerj-cs.305

Kageyama A, Sakamoto N, Miura H, Ohno N (2020) Interactive exploration of the in-situ visualization of a magnetohydrodynamic simulation. Plasma Fusion Res 15:1401065

Kageyama A, Yamada T (2014) An approach to exascale visualization: interactive viewing of in-situ visualization. Comput Phys Commun 185(1):79–85. https://doi.org/10.1016/j.cpc.2013.08.017

Kawamura T, Noda T, Idomura Y (2016) In-situ visual exploration of multivariate volume data based on particle based volume rendering. proceedings of the second workshop on in situ infrastructures for enabling extreme-scale analysis and visualization—ISAV. 2016, pp 18–22, https://doi.org/10.1109/ISAV.2016.009

Kameyama M, Kageyama A, Sato T (2008) Multigrid-based simulation code for mantle convection in spherical shell using Yin-Yang grid. Phy Earth Planet Inter 171:19–32. https://doi.org/10.1016/j.pepi.2008.06.025

Levoy M (1990) Efficient ray tracing of volume data. ACM Trans Graph 9(3):245–261. https://doi.org/10.1145/78964.78965

Li X, Shen X, Peng X, Xiao F, Zhuang Z, Chen C (2013) An accurate multimoment constrained finite volume transport model on Yin-Yang grids. Adv Atmos Sci 30:1320–1330. https://doi.org/10.1007/s00376-013-2217-x

Ma KL (2009) In situ visualization at extreme scale: challenges and opportunities. IEEE Comput Graph Appl 29:14–19

Ma KL, Wang C, Yu H, Tikhonova A (2007) In-situ processing and visualization for ultrascale simulations. J Phys Conf Ser 78(1):1–10. https://doi.org/10.1088/1742-6596/78/1/012043

Mabuchi J, Masada Y, Kageyama A (2015) Differential rotation in magnetized and non-magnetized stars. Astrophys J 806:10. https://doi.org/10.1088/0004-637X/806/1/10

Matthes A, Huebl A, Widera R, Grottel S, Gumhold S, Bussmann M (2016) In situ, steerable, hardware-independent and data-structure agnostic visualization with ISAAC. Supercomput Front Innov 3(4):30–48

Max N, Hanrahan P, Crawfis R (1990) Area and volume coherence for efficient visualization of 3D scalar functions. Proc 1990 Workshop Vol Vis VVS 10(1145/99307):99315

Miura H (2019) Extended magnetohydrodynamic simulations of decaying, homogeneous, approximately-isotropic and incompressible turbulence. Fluids 4:46. https://doi.org/10.3390/fluids4010046

Miura H, Yang J, Gotoh T (2019) Hall magnetohydrodynamic turbulence with a magnetic Prandtl number larger than unity. Phys Rev E 100:063207. https://doi.org/10.1103/PhysRevE.100.063207

Miyagoshi T, Kageyama A, Sato T (2010) Zonal flow formation in the Earths core. Nature 463(7282):793–796. https://doi.org/10.1038/nature08754

Ohno N, Kageyama A (2009) Visualization of spherical data by Yin-Yang grid. Comput Phys Commun 180(9):1534–1538. https://doi.org/10.1016/j.cpc.2009.04.008

Ohno N, Ohtani H (2014) Development of in-situ visualization tool for PIC simulation. Plasma Fusion Res 9(SpecialIssue2):3401071. https://doi.org/10.1585/pfr.9.3401071

Peng X, Xiao F, Takahashi K (2006) Conservative constraint for a quasi-uniform overset grid on the sphere. Q J R Meteorol Soc 132:979–996. https://doi.org/10.1256/qj.05.18

Qaddouri A, Lee V (2011) The Canadian global environmental multiscale model on the Yin-Yang grid system. Q J R Meteorol Soc 137:1913–1926. https://doi.org/10.1002/qj.873

Raymer E (2012) Three-dimensional hydrodynamic simulations of accretion in short-period Algols. MNRAS 427:1702–1712. https://doi.org/10.1111/j.1365-2966.2012.22090.x

Ross RB, Peterka T, Shen HW, Hong Y, Ma KL, Yu H, Moreland K (2008) Visualization and parallel I/O at extreme scale. J Phys Conf Ser 125:012099. https://doi.org/10.1088/1742-6596/125/1/012099

Schroeder W, Martin K, Lorensen B (2006) The visualization toolkit: an object-oriented approach to 3-D graphics, 4th edn. Kitware, New York

Tackley PJ (2008) Modelling compressible mantle convection with large viscosity contrasts in a three-dimensional spherical shell using the yin-yang grid. Phys Earth Planet Inter 171:7–18. https://doi.org/10.1016/j.pepi.2008.08.005

Tikhonova A, Correa CD, Ma K-L (2010a) Explorable images for visualizing volume data. Proceedings of IEEE Pacific Visualization Symposium, pp 177–184, https://doi.org/10.1109/PACIFICVIS.2010.5429595

Tikhonova A, Correa CD, Ma K-L (2010b) Visualization by proxy: a novel framework for deferred interaction with volume data. IEEE Trans Vis Comput Graph 16(6):1551–1559. https://doi.org/10.1109/TVCG.2010.215

Whitlock B, Favre M J, Meredith S J (2011) Parallel in situ coupling of simulation with a fully featured visualization system. Eurographics Symposium on Parallel Graphics and Visualization, pp 101–109, https://doi.org/10.2312/EGPGV/EGPGV11/101-109

Wongwathanarat A, Hammer NJ, Müller E (2010) An axis-free overset grid in spherical polar coordinates for simulating 3D self-gravitating flows. Astron Astrophys 514:A48. https://doi.org/10.1051/0004-6361/200913435

Ye Y, Miller R, Ma K-L (2013) In situ pathtube visualization with explorable images. Proc EGPGV 2013:9–16. https://doi.org/10.2312/EGPGV/EGPGV13/009-016

Yoshida K, Miura H, Tsuji Y (2019) Spectrum in the strong turbulence region of Gross-Pitaevskii turbulence. J Low Temp Phys 196:211–217. https://doi.org/10.1007/s10909-019-02197-4

Yoshida M, Kageyama A (2004) 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 31:L12609. https://doi.org/10.1029/2004GL019970

Publisher's Note