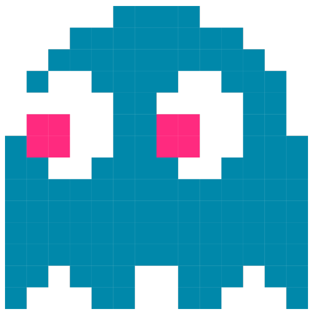
GHOST

C. Thieulot

July 5, 2018



Contents

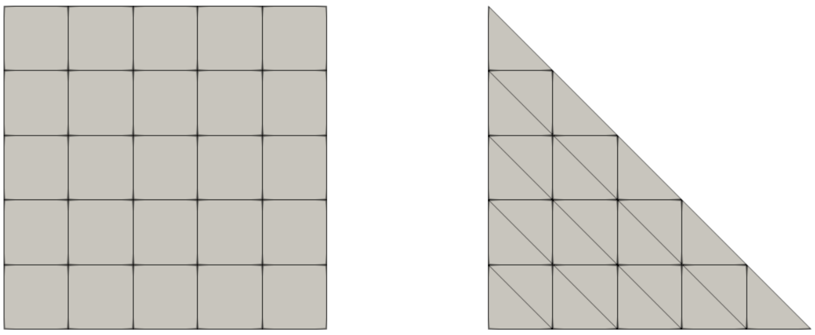


Figure 1: Reference square and triangles meshes at level 5

Please contact me if you need a new feature or if you wish to contribute to GHOST at c.thieulot@uu.nl

1 Introduction

The open source code library GHOST allows three different types of hollow sphere meshes to be built , i.e. meshes bounded by two concentric spheres:

- The cubed sphere ('HS06'), composed of 6 blocks which are themselves subdivided into $N_b \times N_b$ quadrilateral shaped cells [22, 21, 1, 12]. Four types of cubed spheres meshes have been proposed: the conformal, elliptic, gnomonic and spring types [20]. However only gnomonic meshes are considered here: these are obtained by inscribing a cube within a sphere and expanding to the surface of the sphere. The cubed sphere has recently been used in large-scale mantle convection simulation in conjunction with Adaptive Mesh Refinement [2, 12].
- The CitcomS mesh ('HS12') composed of 12 blocks also subdivided into $N_b \times N_b$ quadrilateral shaped cells [28, 23, 27, 3]. Note that ASPECT [17, 16], a relatively new code aimed at superseeding CitcomS can generate and use this type of mesh [24] but is not limited to it.
- The icosahedral mesh ('HS20') composed of 20 triangular blocks [5, 4] subdivided into triangles, which is used in the TERRA code [10, 11, 9, 13].

Given the regularity and symmetry of these meshes determining the location of the mesh nodes in space is a relatively straightforward task. Building the mesh connectivity in an efficient manner is where the difficulty lies.

The approach to building all three meshes is identical:

1. A reference square or triangle is populated with cells, as shown in Fig. (1) parametrised by a level l: the square is subdivided into $l \times l$ quadrilaterals while the triangle is subdivided into l^2 triangles.
2. This reference square or triangle is then replicated *nblock* times (6, 12 or 20) and mapped onto a portion of a unit sphere. The blocks are such that their union covers a full sphere but they cannot overlap except at the edges, see Fig. (2).
3. All block meshes are then merged together to generate a shell mesh.
4. Shell meshes are replicated *nlayer+1* times outwards with increasing radii.
5. The *nlayer* shells are then merged together to form a hollow sphere mesh, as shown in Fig. (3).

In Table (**??**) the number of nodes and cells for a variety of resolutions for all three mesh types is reported. Looking at the CitcomS literature of the past 20 years, we find that the mesh data presented in this table cover the various resolutions used, e.g. 12×48^3 [18, 3], 12×64^3 [7] 12×96^3 [8], 12×128^3 [6, 25, 26]. Note that in the case of the HS06 and HS12 meshes the mesh nodes are mapped out to the 6 or 12 blocks following either an equidistant or equiangle approach as shown in Fig. (**??**) (see [20] for details on both approaches).

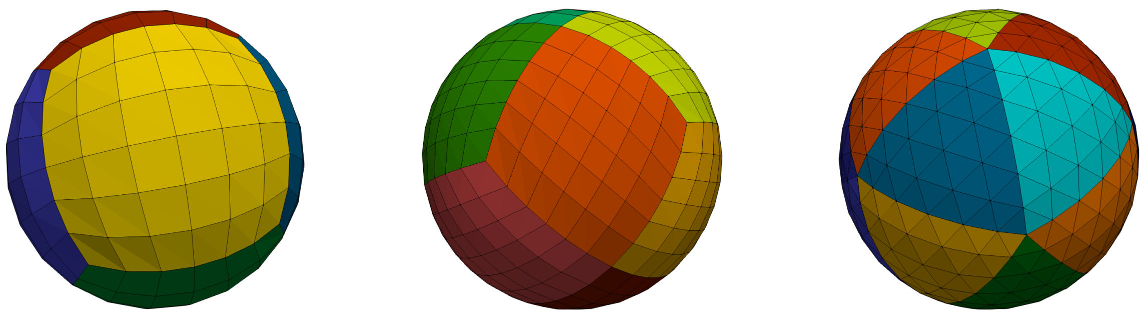


Figure 2: From left to right: HS06, HS12 and HS20 shells coloured by block number.

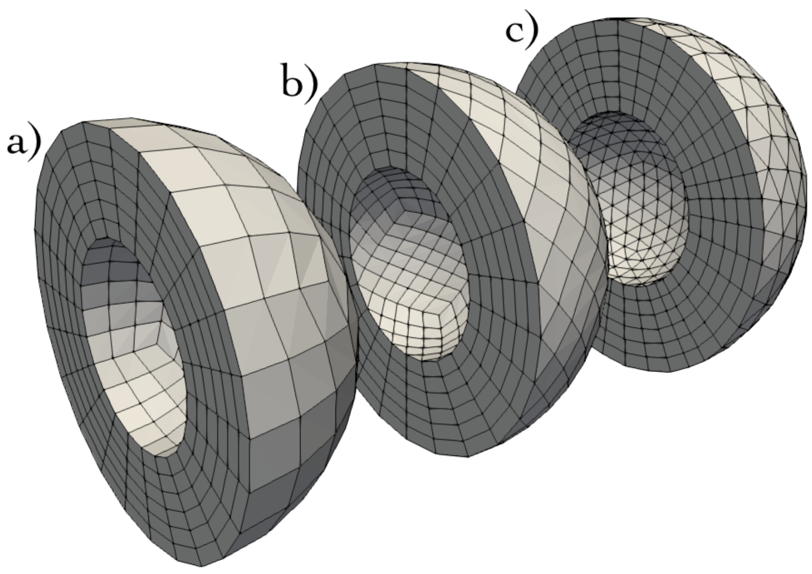
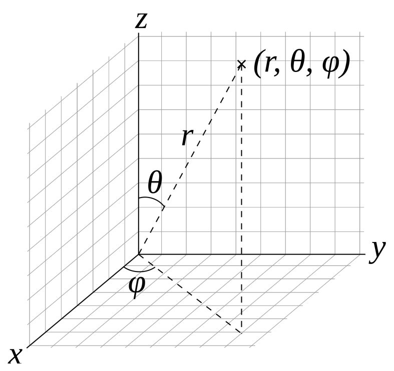


Figure 3: a) HS06 mesh composed of 6 blocks containing each 6^3 cells; b) HS12 mesh composed of 12 blocks containing each 6^3 cells; e) HS20 mesh composed of 20 blocks containing each 6^3 cells.

2 Conventions & notations

On the following figure are represented the three cartesian axis, a point and its spherical coordinates r, θ, ϕ:



Spherical coordinates as commonly used in physics: polar angle θ, and azimuthal angle ϕ.

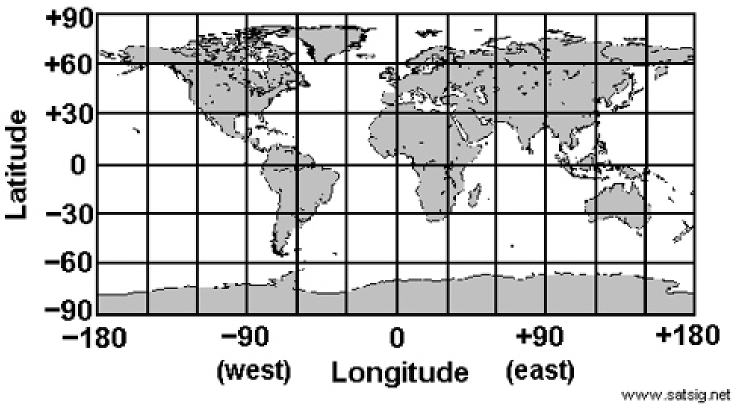
In this case $\theta \in [0 : \pi]$ and $\phi \in]-\pi : \pi]$ and we have the following relationships:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

$$\theta = acos(z/r) \quad (2)$$
$$\phi = atan(y/x) \quad (3)$$

The inverse tangent used to compute ϕ must be suitably defined, taking into account the correct quadrant of (x, y), which is why the atan2 intrinsic function is used.

In geography one uses latitude and longitude, represented hereunder:



- Latitude $\in [-90:90]$, or $\in [-\pi/2:\pi/2]$
- Longitude $\in]-180:180]$, or $\in]-\pi:\pi]$

3 Running the program

The code can be compiled by typing *make*. The default compiler is *gfortran* but it can be changed by editing the *Makefile*. Running *make* generates the *ghost* executable which can then be run as follows in a terminal:

```
> ./ghost
```

If you wish to know which options are available, simply do

```
> ./ghost --help
```

Options are as follows:

- `-level X` where X is an integer number strictly larger than 2.
- `-nlayer X` where X is an integer number strictly larger than 2.
- `-mtype X` where X is equal to 6,12,20
- `-equiangular X` where X is 0 (false) or 1 (true)

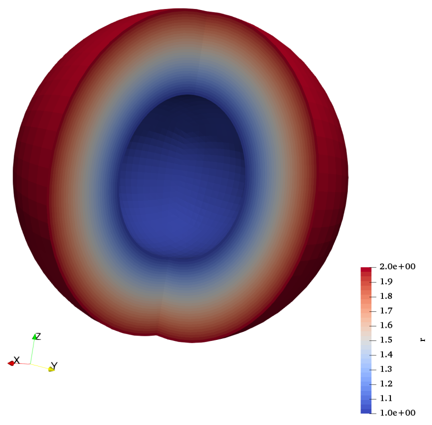
For instance, if you wish to generate a HS06 mesh of block level 16 with 20 layers in the radial direction with equiangular spacing:

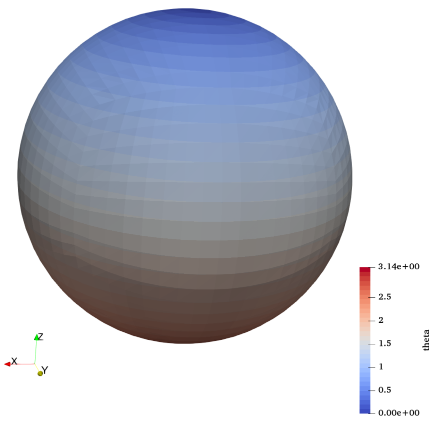
```
./ghost -level 16 -mtype 6 -nlayer 20 -equiangular 1
```

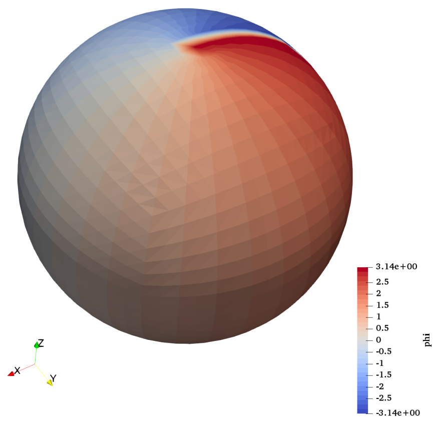
Note that a clean option exists to remove all compilation files and output files:

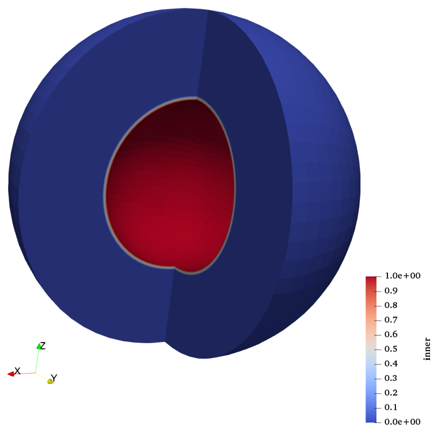
```
> make clean
```

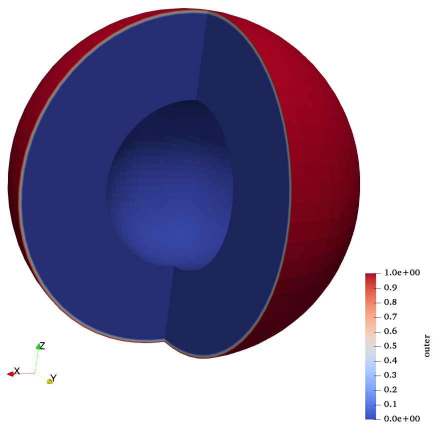
You can then open the file *hollow_sphere.vtu* in Paraview:

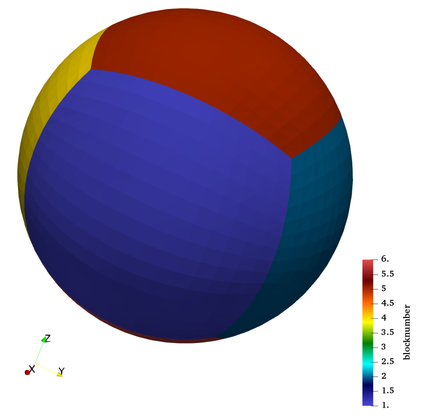


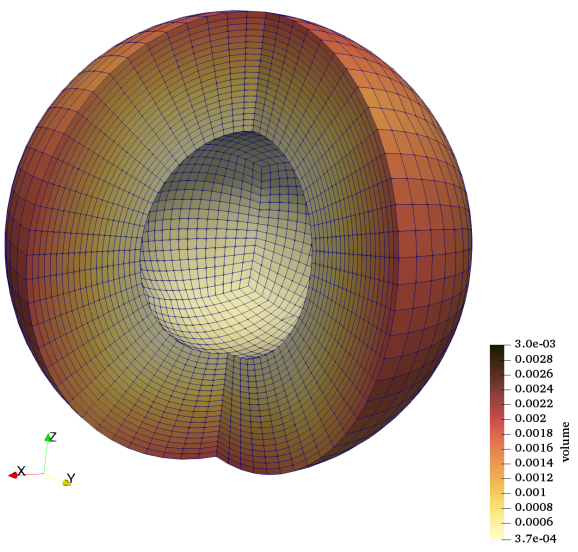


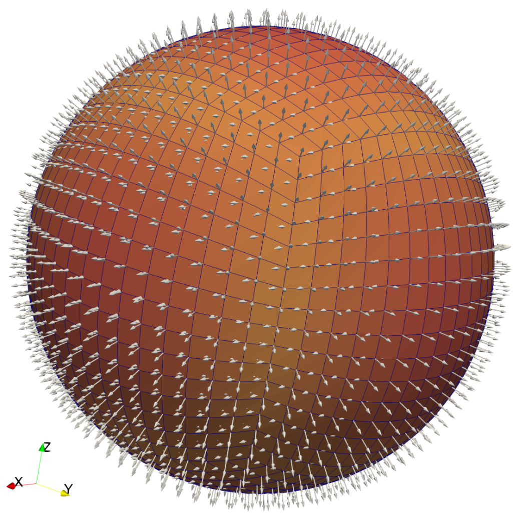












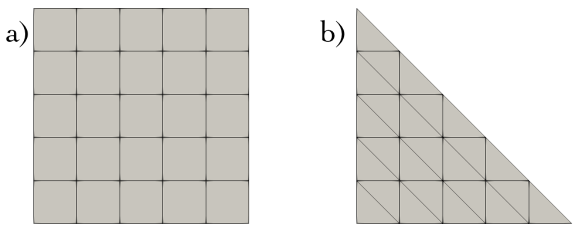
4 Subroutines

4.1 `allocate_block_memory`

This subroutine assigns to every block *i* a number of node points *block(i)%np* and a number of cells *block(i)%ncell* The number of vertices per cell is also stored in *block(i)%nv*. It then loops over all blocks and allocates the required arrays to store the mesh nodes position in Cartesian and spherical coordinates and the connectivity.

4.2 `block_node_layout`

This subroutine generates the position of the mesh node points for each block as well as their connectivity (i.e. the list of nodes making each element.



a) block layout for mtype=3,4; b) block layout for mtype=5

In the case that the block is a quadrilateral, it is assumed that it is made of $l \times l$ quadrilateral cells. In the case that the block is a triangle, it is made of l^2 triangular cells.

4.3 `build_hollow_sphere`

This subroutine first makes a copy of the current shell into *shell_temp* It then computes the number of mesh nodes and cells for the hollow sphere and allocate all arrays accordingly. It then loops over layers, places the temporary shell at the required radius, and appends it to the current hollow sphere mesh from the inside out.

4.4 `check_parameters`

This routine performs a basic check on the nblock, level and nlayer parameter values.

4.5 `hexahedron_volume`

This function computes the volume of any hexahedron, following Eq.(12) of [15].

4.6 `compute_gravity_at_point`

This subroutine receives as argument the coordinates `x,y,z` of a point and returns the gravity vector components `gx,gy,gz` at this location by means of

$$\boldsymbol{g}(\boldsymbol{r}') = \int_\Omega \mathcal{G} \frac{\rho(\boldsymbol{r}) - \rho_0}{|\boldsymbol{r}' - \boldsymbol{r}|^3} (\boldsymbol{r}' - \boldsymbol{r}) d\Omega$$

where ρ_0 =`refdensgrav`. The gravitational potential is computed as follows:

$$U(\boldsymbol{r}') = - \int_\Omega \mathcal{G} \frac{\rho(\boldsymbol{r}) - \rho_0}{|\boldsymbol{r}' - \boldsymbol{r}|} d\Omega$$

4.7 `compute_gravity_on_line`

This subroutine computes on a line parametrised by $0 \leq r \leq 2R_2$, $\theta = 13^\circ$ and $\phi = 17^\circ$. This line is discretised over npts points. At each point is the subroutine compute_gravity_at_point called.

4.8 `compute_hollow_sphere_volume`

This routine loops over all cells and adds up their volumes. Each cell volume is obtaines by means of a Gauss quadrature. Measurements are written in *fort.777*.

4.9 compute_normals_hollow_sphere

This routine computes the normal vector to the inner and outer boundaries of the 3D mesh at the nodes. The vector is an average of several normal vectors computed on the cells containg the node in question.

4.10 compute_PREM_density

This routine gets a radius $r < 6371$km and returns the density as given by the PREM model [14].

4.11 compute_r_theta_phi_hollow_sphere

This subroutine computes r, θ and ϕ from x, y, z for all nodes of the hollow_sphere mesh with

$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\theta = \cos^{-1}(z/r)$$
$$\phi = \tan_2^{-1}(y/x)$$

4.12 compute_r_theta_phi_shell

This subroutine computes r, θ and ϕ from x, y, z for all nodes of the shell mesh with

$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\theta = \cos^{-1}(z/r)$$
$$\phi = \tan_2^{-1}(y/x)$$

4.13 compute_shell_area

This routine computes the total shell area by looping over all shell cells and adding up their areas. Quadrilaterals are broken down in 4 triangles as explained in the paper.

4.14 compute_shell_distributions

This routine computes the average edge length of a shell and its standard deviation. Results are generated in the *fort.889* file.

4.15 compute_triangle_area

`https://en.wikipedia.org/wiki/Heron%27s_formula` Heron's formula states that the area of a triangle whose sides have lengths a, b, and c is

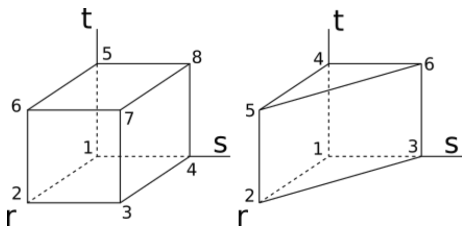
$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where s is the semiperimeter of the triangle; that is,

$$s = \frac{a+b+c}{2}$$

4.16 generate_quadrature_points

This routine computes the location of all quadrature points [1] of the mesh. If the mesh is based on hexahedra (HS06,HS12) each cell counts 2x2x2 quadrature points each with a weight of 1.



[1] https://en.wikipedia.org/wiki/Gaussian_quadrature

If the mesh is based on triangular prisms (HS20) it has 2x3 quadrature points, (2 in the radial direction and 3 in the triangular plane) and the weights are 1/6. All q-points are generated in the reference cell and then mapped out to real cells by means of a linear mapping. For hexahedral elements the linear shape functions are:

$$\begin{aligned}
N_1(r,s,t) &= (1-r)(1-s)(1-t)/8\\
N_2(r,s,t) &= (1+r)(1-s)(1-t)/8\\
N_3(r,s,t) &= (1+r)(1+s)(1-t)/8\\
N_4(r,s,t) &= (1-r)(1+s)(1-t)/8\\
N_5(r,s,t) &= (1-r)(1-s)(1+t)/8\\
N_6(r,s,t) &= (1+r)(1-s)(1+t)/8\\
N_7(r,s,t) &= (1+r)(1+s)(1+t)/8\\
N_8(r,s,t) &= (1-r)(1+s)(1+t)/8
\end{aligned}$$

For the triangular prism elements the linear shape functions are:

$$\begin{aligned}
N_1(r,s,t) &= (1-r-s)(1-t)/2\\
N_2(r,s,t) &= r(1-t)/2\\
N_3(r,s,t) &= s(1-t)/2\\
N_4(r,s,t) &= (1-r-s)(1+t)/2\\
N_5(r,s,t) &= r(1+t)/2\\
N_6(r,s,t) &= s(1+t)/2
\end{aligned}$$

4.17 `map_blocks`

4.17.1 HS06 - cubed sphere

See section 3.1 of [20], or [22, 19]

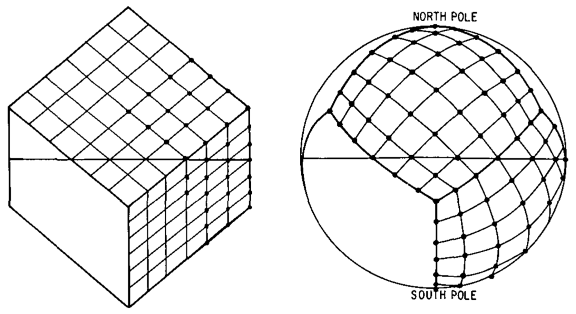
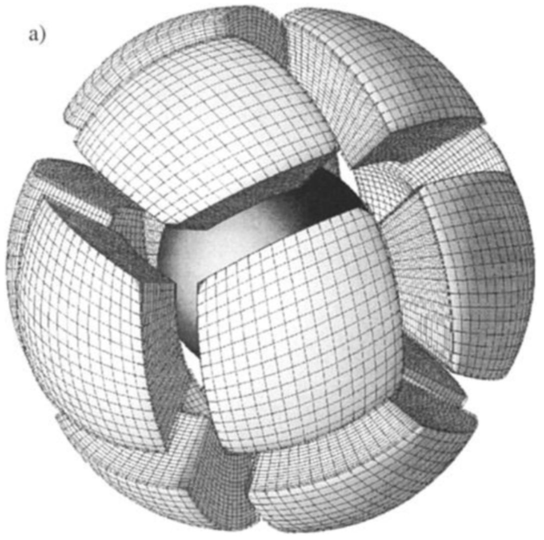


Figure 1.—A cubic representation of the earth. A cubic grid is shown together with the corresponding spherical grid which fits into the cubic splitting of the sphere, in the exact disposition that was used in the actual computations.

4.17.2 HS12 - CitcomS mesh



See [28, 27].

4.17.3 HS20 - Icosahedron

Let $t = (1+\sqrt{5})/2$, then the vertices of the icosahedron are

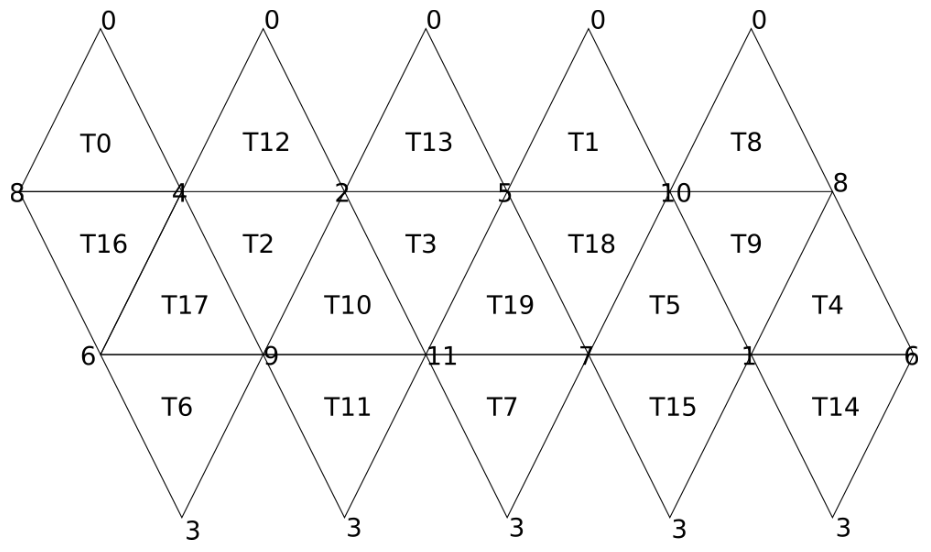
$$
\begin{aligned}
\boldsymbol{x}_0 &= \frac{1}{\sqrt{1+t^2}}(t,1,0) & \boldsymbol{x}_6 &= \frac{1}{\sqrt{1+t^2}}(-1,0,t) \\
\boldsymbol{x}_1 &= \frac{1}{\sqrt{1+t^2}}(-t,1,0) & \boldsymbol{x}_7 &= \frac{1}{\sqrt{1+t^2}}(-1,0,-t) \\
\boldsymbol{x}_2 &= \frac{1}{\sqrt{1+t^2}}(t,-1,0) & \boldsymbol{x}_8 &= \frac{1}{\sqrt{1+t^2}}(0,t,1) \\
\boldsymbol{x}_3 &= \frac{1}{\sqrt{1+t^2}}(-t,-1,0) & \boldsymbol{x}_9 &= \frac{1}{\sqrt{1+t^2}}(0,-t,1) \\
\boldsymbol{x}_4 &= \frac{1}{\sqrt{1+t^2}}(1,0,t) & \boldsymbol{x}_{10} &= \frac{1}{\sqrt{1+t^2}}(0,t,-1) \\
\boldsymbol{x}_5 &= \frac{1}{\sqrt{1+t^2}}(1,0,-t) & \boldsymbol{x}_{11} &= \frac{1}{\sqrt{1+t^2}}(0,-t,-1)
\end{aligned}
$$

It is obvious that all points lie on a sphere of unit radius.

The triangles are as follows:

$$
\begin{aligned}
T_0 &= [0,8,4] & T_{10} &= [2,9,11] \\
T_1 &= [0,5,10] & T_{11} &= [3,11,9] \\
T_2 &= [2,4,9] & T_{12} &= [4,2,0] \\
T_3 &= [2,11,5] & T_{13} &= [5,0,2] \\
T_4 &= [1,6,8] & T_{14} &= [6,1,3] \\
T_5 &= [1,10,7] & T_{15} &= [7,3,1] \\
T_6 &= [3,9,6] & T_{16} &= [8,6,4] \\
T_7 &= [3,7,11] & T_{17} &= [9,4,6] \\
T_8 &= [0,10,8] & T_{18} &= [10,5,7] \\
T_9 &= [1,8,10] & T_{19} &= [11,7,5]
\end{aligned}
$$

Graphically, the mesh has the following structure:



4.18 `merge_blocks`

This subroutine loops over all the (shell) blocks and merges them together to arrive at a full shell. In order to do so it needs to remove point duplicates and also merges the connectivity arrays of all the blocks. All shell blocks (x,y,z coordinates and hull information) are first merged into temporary arrays (tempx, tempy,tempz,sides) of size nblock*nnp. A logical array doubble of the same size is then set to true when two points of two different blocks share the same position. At the same time another integer array pointto is used: if node m is colocated with point n then pointto(m)=n (for n<m). The 'real' total number of nodes of the assembled shell can then be computed while the total number of cells simply is the number of blocks multiplied by the number of cells each block contains. All arrays pertaining to this assembled shell can then be allocated. Finally the doubble and pointto arrays are used to construct a single set of coordinates and a connectivity array for an assembled shell without duplicates.

4.19 `output_blocks_vtu`

If the *generate_vtu_output* flag is true, it generates a vtu file for each block in the *OUT* folder.

4.20 `output_hollow_sphere_vtu`

If the *generate_vtu_output* flag is true, it generates the *hollow_sphere.vtu* file in the *OUT* folder.

4.21 `output_shell_ascii`

If the *generate_ascii_output* flag is set to true it generates two ascii files in *OUT/ASCII*. The first is shell_xyz.ascii and contains the x,y,z position of all points of the shell. The second is shell_icon.ascii and contains the connectivity array.

4.22 `output_shell_lat_lon`

If the *generate_ascii_output* flag is set to true this subroutine exports θ and ϕ values of all points on the shell in the file *shell_theta_phi.dat* in *OUT*.

4.23 `output_shell_vtu`

If *generate_vtu_output* is true a vtu file of the shell is created in *OUT*.

4.24 `project_on_sphere`

This subroutine gets a radius r value as argument and the current coordinates of a point. It returns the new Cartesian coordinates of the point on a sphere of radius r with the same θ, ϕ.

4.25 `read_command_line_options`

This subroutine reads in the command line arguments

4.26 `read_s40rts`

This routine reads in the S40RTS data which should be located in the DATA folder. Please go to the webiste of the S40RTS author http://jritsema.earth.lsa.umich.edu//Research.html and download the S40RTS.sph file. The routine then loops over all nodes of the mesh and assigns them a $\delta \ln V_s$ value, and consequently a $\delta\rho$ value. Note that this routine is triggered if the *s40rts* flag is set to true.

4.27 `write_positions`

This routine is part of vtu_tools.f90

4.28 `write_field_dp`

This routine is part of vtu_tools.f90

4.29 `write_field_int`

This routine is part of vtu_tools.f90

4.30 `write_icon`

This routine is part of vtu_tools.f90

4.31 `write_offsets`

This routine is part of vtu_tools.f90

4.32 `write_types`

This routine is part of vtu_tools.f90

References

[1] *Three-dimensional spherical shell convection at infinite Prandtl number using the cubed sphere method*, Proceedings Second MIT Conference on Compurational Fluid and Solid Mechanics June 1720, 2003. Elsevier, 2003.

[2] L. Alisic, M. Gurnis, G. Stadler, C. Burstedde, and O. Ghattas. Multi-scale dynamics and rheology of mantle flow with plates. *J. Geophys. Res.*, 117:doi:10.1029/2012JB009234, 2012.

[3] P.A. Arrial, N. Flyer, G.B. Wright, and L.H. Kellogg. On the sensitivity of 3-D thermal convection codes to numerical discretization: a model intercomparison. *Geosci. Model Dev.*, 7:2065–2076, 2014.

[4] J.R. Baumgardner. Three-Dimensional treatment of convective flow in the Earth's mantle. *Journal of Statistical Physics*, 39(5/6):501, 1985.

[5] J.R. Baumgardner and P.O. Frederickson. Isocahedral discretisation of the two-sphere. *SIAM J. Numer Anal.*, 22(6):1107–1115, 1985.

[6] T.W. Becker. On the effect of temperature and strain-rate dependent viscosity on global mantle flow, net rotation, and plate-driving forces. *Geophy. J. Int.*, 167:943–957, 2006.

[7] A.L. Bull, M. Domeier, and T.H. Torsvik. The effect of plate motion history on the longevity of deep mantle heterogeneities. *Earth Planet. Sci. Lett.*, 401:172–182, 2014.

[8] A.L. Bull, A.K. McNamara, T.W. Becker, and J. Ritsema. Global scale models of the mantle flow field predicted by synthetic tomography models. *Phys. Earth. Planet. Inter.*, 182:129–138, 2010.

[9] H.-P. Bunge, M. Richards, C. Lithgow-Bertelloni, J.R. Baumgardner, S.P. Grand, and B. Romanowicz. Time scales and heterogeneous structure in geodynamic Earth models. *Science*, 280:91–95, 1998.

[10] H.-P. Bunge, M.A. Richards, and J.R. Baumgardner. Effect of depth-dependent viscosity on the planform of mantle convection. *Nature*, 379:436–438, 1996.

[11] H.-P. Bunge, M.A. Richards, and J.R. Baumgardner. A sensitivity study of three-dimensional spherical mantle convection at 10^8 Rayleigh number: Effects of depth-dependent viscosity, heating mode, and endothermic phase change . *J. Geophys. Res.*, 102(B6):11,991–12,007, 1997.

[12] C. Burstedde, G. Stadler, L. Alisic, L.C. Wilcox, E. Tan, M. Gurnis, and O. Ghattas. Large-scale adaptive mantle convection simulation. *Geophy. J. Int.*, 192:889–906, 2013.

[13] D.R. Davies, J.H. Davies, P.C. Bollada, O. Hassan, K. Morgan, and P. Nithiarasu. A hierarchical mesh refinement technique for global 3-D spherical mantle convection modelling. *Geosci. Model Dev.*, 6:1095–1107, 2013.

[14] A.M. Dziewonski and D.L. Anderson. Preliminary reference Earth model. *Phys. Earth. Planet. Inter.*, 25:297–356, 1981.

[15] J. Grandy. Efficient computation of volume of hexahedral cells. Technical Report UCRL-ID-128886, Lawrence Livermore National Laboratory, 1997.

[16] T. Heister, J. Dannberg, R. Gassmöller, and W. Bangerth. High Accuracy Mantle Convection Simulation through Modern Numerical Methods. II: Realistic Models and Problems. *Geophy. J. Int.*, 210(2):833–851, 2017.

[17] M. Kronbichler, T. Heister, and W. Bangerth. High accuracy mantle convection simulation through modern numerical methods . *Geophy. J. Int.*, 191:12–29, 2012.

[18] A.K. McNamara and S. Zhong. Thermochemical structures within a spherical mantle: Superplumes or piles? *J. Geophys. Res.*, 109(B07402), 2004.

[19] R.D. Nair, S.J. Thomas, and R.D. Loft. A Discontinuous Galerkin Transport Scheme on the Cubed Sphere. *Monthly Weather Review*, 133:814–828, 2005.

[20] W.M. Putman and S.-J. Lin. Finite-Volume transport on various cubed-sphere grids. *J. Comp. Phys.*, 227:55–78, 2007.

[21] C. Ronchi, R. Iacono, and P.S. Paolucci. The "Cubed Sphere": A New Method for the Solution of Partial Differential Equations in Spherical Geometry. *J. Comp. Phys.*, 124:93–114, 1996.

[22] R. Sadourny. Conservative Finite-Difference Approximations of the Primitive Equations on Quasi-Uniform Spherical Grids. *Monthly Weather Review*, 100(2):136–144, 1972.

[23] K. Stemmer, H. Harder, and U. Hansen. A new method to simulate convection with strongly temperature- and pressure-dependent viscosity in a spherical shell: Applications to the Earth's mantle. *Phys. Earth. Planet. Inter.*, 157:223–249, 2006.

[24] C. Thieulot. Analytical solution for viscous incompressible stokes flow in a spherical shell. *Solid Earth Discussions*, 2017:1–19, 2017.

[25] M.B. Weller and A. Lenardic. The energetics and convective vigor of mixed-mode heating: Velocity scalings and implications for the tectonics of exoplanets. *Geophys. Res. Lett.*, 43, 2016.

[26] M.B. Weller, A. Lenardic, and W.B. Moore. Scaling relationships and physics for mixed heating convection in planetary interiors: Isoviscous spherical shells. *J. Geophys. Res.*, 121, 2016.

[27] S. Zhong, A. McNamara, E. Tan, L. Moresi, and M. Gurnis. A benchmark study on mantle convection in a 3-D spherical shell using CITCOMS. *Geochem. Geophys. Geosyst.*, 9(10), 2008.

[28] S. Zhong, M.T. Zuber, L.N. Moresi, and M. Gurnis. The role of temperature-dependent viscosity and surface plates in spherical shell models of mantle convection. *J. Geophys. Res.*, 105(B5):11,063–11,082, 2000.