University of Leeds

**SCHOOL OF COMPUTING**

**RESEARCH REPORT SERIES**

Report 2008.??

# Data Structures for Visualising Semi-Structured Adaptive Mesh Refinement Data

by

**M E Hubbard**

April 2008

**Abstract**

A variety of algorithms exist for approximating partial differential equations on adapted meshes which retain some partial structure (*e.g.* via overlaying structured sub-meshes – quadrilateral in two dimensions, hexahedral in three). The resulting data is commonly stored at the centres of the computational cells, which means that when this data is being visualised there is no explicit connectivity between the points at which the information is supplied. This report presents a simple method for creating this connectivity so that the data can be visualised using the full range of techniques available in standard packages, without resorting to crude "fixes".

## 1 Introduction

A commonly used tool in the approximation of partial differential equations is adaptive meshing. This process can take many forms but one of the most popular is based on overlaying regions of a structured mesh with a series of finer structured meshes (refined in a uniform, but possibly anisotropic, manner) to give a hierarchy of embedded Cartesian meshes of differing resolutions on which the mathematical equations can be approximated. An example of such a method is the adaptive mesh refinement (AMR) technique originated by Berger and Oliger [2], updated by Berger and Colella [1], and developed further by Quirk [3], amongst others. Figure 1 illustrates a typical structure of a two-dimensional mesh created using one of these approaches.

This AMR method is designed so that the meshes are, to all intents and purposes, overlaid and not connected. The majority of the communication which takes place between mesh cells is carried out through their edges, but information about the solution is typically stored at the *centres* of the mesh cells. This makes it very awkward to accurately visualise the solution data, because there is no global connectivity to allow the data to be easily interpolated. This report describes a simple approach to creating this connectivity in a unique manner, in two or three dimensions, so that the data is stored in a format which will facilitate the use of the full capabilities of standard visualisation tools.

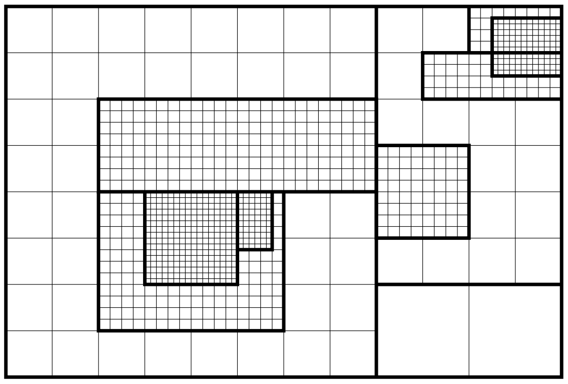

Figure 1: An example two-dimensional AMR mesh: the data is stored at the centres of the quadrilateral cells (they don't have to be squares). The thicker lines represent the boundaries of the individual meshes.

## 2 Alternative Data Structures

There are many different approaches which can be used to convert the type of AMR grid deployed here to a structure which can be used to visualise the data. A few of these are listed below, although all of them are, to some degree, unsatisfactory.

**1.** Reconstruct the data in each cell using the surrounding information.

- Standard techniques will generally give a discontinuous representation of the data between cells. The simplest approach would assume that the solution is piecewise constant and hence assign the cell-centre value to the whole cell and use this as part of the visualisation process. This covers the whole domain but isn't particularly accurate and the discontinuities across the cell boundaries can interfere with understanding. Higher order representations are possible, but can be expensive and wouldn't necessarily provide a clearer visualisation of the data.

**2.** Project the cell-centre data on to the underlying coarse mesh and interpolate the result using the inherent structure in the new dataset.

- This covers the whole domain but will inevitably compromise the accuracy of the visualisation, possibly to a point where it is no longer informative, since information will only be shown at the coarsest resolution available.

**3.** Assign the average of the surrounding cell-centre data to each mesh node.

- The nodal connectivity is simple to obtain, though the issue of hanging nodes (ones which occur on an edge/face of a neighbouring cell rather than at a vertex, and which appear at the interfaces between coarse and fine meshes) means that the shapes of the nodal areas/volumes, over which the interpolation for the visualisation would be carried out, would not be known in advance. It also requires appropriate weightings to be applied within the averaging procedure and smooths the data, losing precision. Again, the whole domain is covered.

**4.** Interpolate the cell-centre data on each structured mesh independently, ignoring the parts which are overlaid by finer meshes.

- This would cover the whole domain except for gaps at interfaces between meshes.

**5.** Triangulate/tetrahedralise a set of vertices defined by the cell-centres and use the resulting mesh for visualisation.

- This provides a connectivity and ensures that it is the original data that is being visualised. However, not only is this expensive in terms of time, it will also result in a mesh which has a considerably larger number of cells than the original. This is unnecessarily inefficient.

The major obstacle to producing a useful data structure is the lack of any explicit connectivity across interfaces between different levels of mesh resolution in the AMR grid. However, this can be overcome.

## 2.1 A Semi-Structured Approach

The algorithm presented here produces a single *unstructured* quadrilateral or hexahedral mesh, where the nodes are the original cell-centres. The lack of structure arises from the fact that mesh edges are allowed to be degenerate, *i.e.* have zero length, but this does not matter as long as the visualisation tool can cope with interpolation over cells whose vertices are allowed to be coincident.

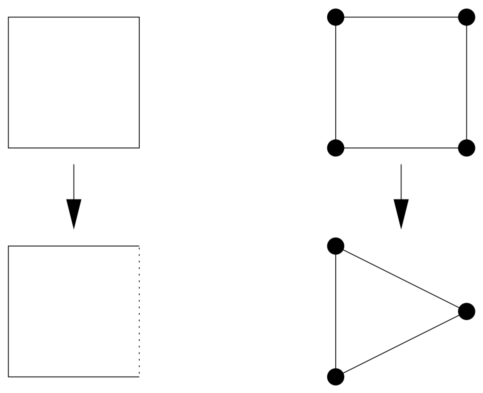

Figure 2: Possible cell shapes in two dimensions: with degenerate edges highlighted by dashed lines (left), and showing the actual cell shape (right).

The first step involves constructing a global numbering of the cells of the adapted mesh. Once this has been carried out, the algorithm can then be described as follows.

Consider a background mesh which covers the whole computational domain at the finest resolution required. For each *internal* node of that fine, structured, Cartesian mesh, five steps are executed.

**(i)** Find the *fine mesh* cells (8 in 3D, 4 in 2D) with that node as a vertex.

**(ii)** For each of these cells, find the finest cell of the adapted mesh which contains it.

- This simply requires interrogation of the mesh structure and is straightforward because the AMR data structure means that the position of every cell of the adapted mesh can easily be found in terms of a global cell index for a structured Cartesian mesh covering the whole domain, whatever the level of refinement.

**(iii)** Consider these cell-centres as vertices of a cell (hexahedral in 3D, quadrilateral in 2D) in a new, global, unstructured mesh.

**(iv)** Check this new cell for degeneracy, *i.e.* zero “volume”. This can be done by comparing it against the possible non-degenerate cases, illustrated for two dimensions in Figure 2 and for three dimensions in Figures 3 and

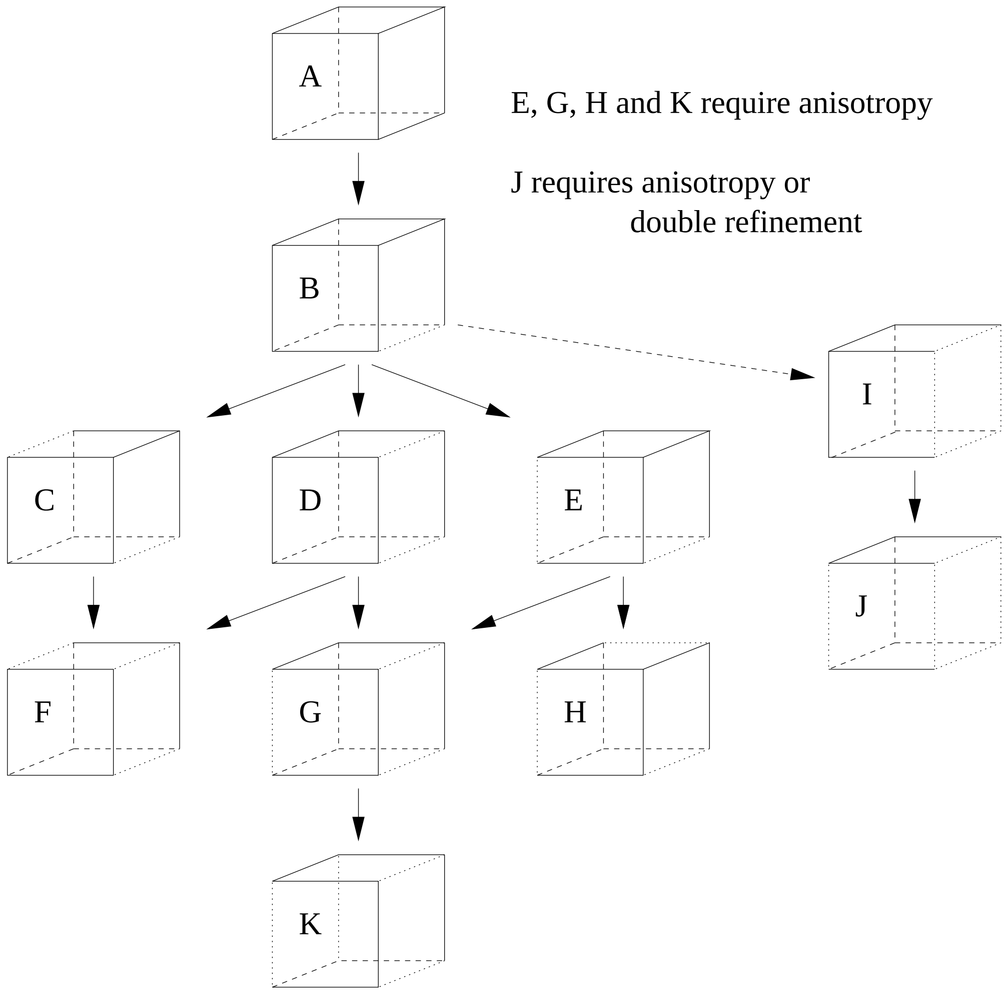
A
E, G, H and K require anisotropy
J requires anisotropy or double refinement
B
I
C
D
E
J
F
G
H
K

Figure 3: Possible cell shapes in three dimensions with degenerate edges of hexahedra highlighted: hidden edges are indicated by dashed lines, degenerate edges by dotted lines. Arrows indicate that the second topology can be obtained from the first through the degeneration of one edge.

4: a degenerate edge is defined to be one for which both vertices appear in the same cell of the *adapted* mesh (and are therefore associated with the same cell-centre data value). However, a simpler way to check (with appropriate simplifications for two dimensions) is to

**a.** assign integer coordinates $(\pm 1, \pm 1, \pm 1)$ appropriately to each vertex of the cell, forming a representative cube;

**b.** search through the edges of that cell and, for any edge which has both vertices in the same cell of the adapted mesh, *i.e.* a degenerate edge, collapse it so that the coordinate for that direction is 0 for both vertices;

**c.** for the resulting cell, calculate

$$V = \left(\sum_i dx\right) \times \left(\sum_j dy\right) \times \left(\sum_k dz\right) ;$$

**d.** if $V = 0$ then the cell is degenerate and can be ignored, otherwise it should be included in the new mesh.

A simple example of a degenerate cell is one whose vertices are all contained in the same cell of the adapted mesh: the representative cube collapses to a point and can be ignored (case **d** in Figure 5).

**(v)** If the cell is degenerate then ignore it; otherwise include it in the connectivity of the new mesh as a hexahedron/quadrilateral (which may have coincident vertices).

The bulk of the work is involved in steps **(ii)** and **(iv)**.

Figure 5 shows the four configurations which result in degenerate cells. Note that it is impossible to have exactly three vertices of a face of the cell in the same cell of the adaptive mesh: this will always force the fourth vertex to be in the same cell, so all the other configurations of degenerate edges can be ignored. They automatically become one of configurations I or J in Figures 3/4 or **b**, **c** or **d** in Figure 5.

**Remark**: It is worth noting here that the algorithm described above was constructed and implemented before noticing that exactly the same effect could be obtained by storing the information in a tree-based data structure

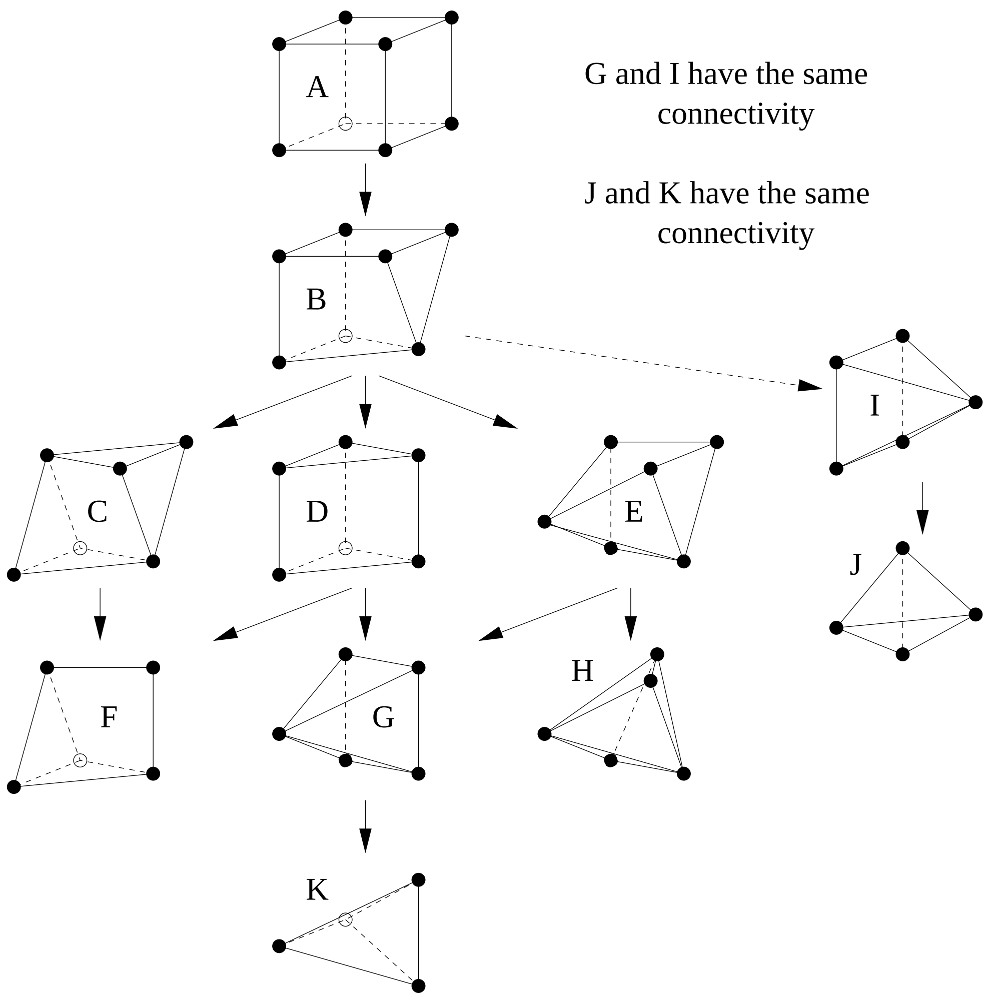
A
G and I have the same connectivity
J and K have the same connectivity
B
I
C
D
E
J
F
G
H
K

Figure 4: Possible cell shapes in three dimensions with degenerate edges collapsed: hidden edges are indicated by dashed lines and hidden vertices by empty circles (other vertices by filled circles). Arrows indicate that the second topology can be obtained from the first through the degeneration of one edge. Note that the vertices of the quadrilateral faces are not necessarily coplanar.

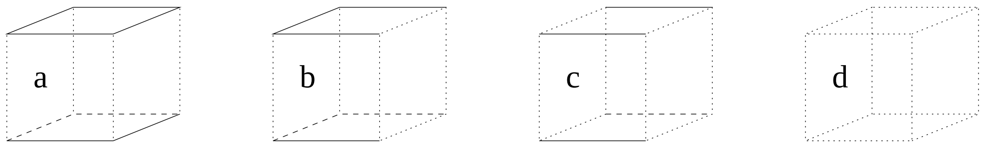

Figure 5: Possible degenerate cell shapes in three dimensions with degenerate edges of hexahedra highlighted: hidden edges are indicated by dashed lines, degenerate edges by dotted lines.

(quadtree- or octree-based if a refinement factor of 2 is used isotropically at each stage). The coarse mesh itself acts as the root node of the tree, with each of its cells forming a branch. Each adapted cell at a given level of refinement is then linked to the cell in the level above that it overlays (the AMR algorithm guarantees that this will always exist). Given that each coarse mesh cell knows its neighbours and that each refined cell knows its position within the cell it overlays, this tree structure can easily be searched to find neighbouring cells at different levels of refinement, thus replacing the need to have the explicit connectivity sought after above. Such a structure would also provide far greater flexibility, since it would be simple to ignore specific levels of detail without having to recreate the connectivity from scratch. This is currently under investigation but has yet to be implemented, so it is not clear that it would be fast enough to be a viable option for visualising large datasets. An initial attempt to assess the viability of such a data structure was made in [7].

## 2.2 Two Dimensions

The two-dimensional case is simple because only two legitimate cell topologies exist in the new mesh, as shown in Figure 2. Either all 4 vertices are in different cells or a neighbouring pair of vertices are in the same cell. Every other combination is either impossible (if two diagonally opposite vertices are in the same cell then the other two must be as well) or results in a degenerate cell which need not be included in the mesh. Figure 6 shows an example of the results obtained when the algorithm has been applied in two space dimensions. Note the apparently triangular cells at the interfaces between different grid levels.

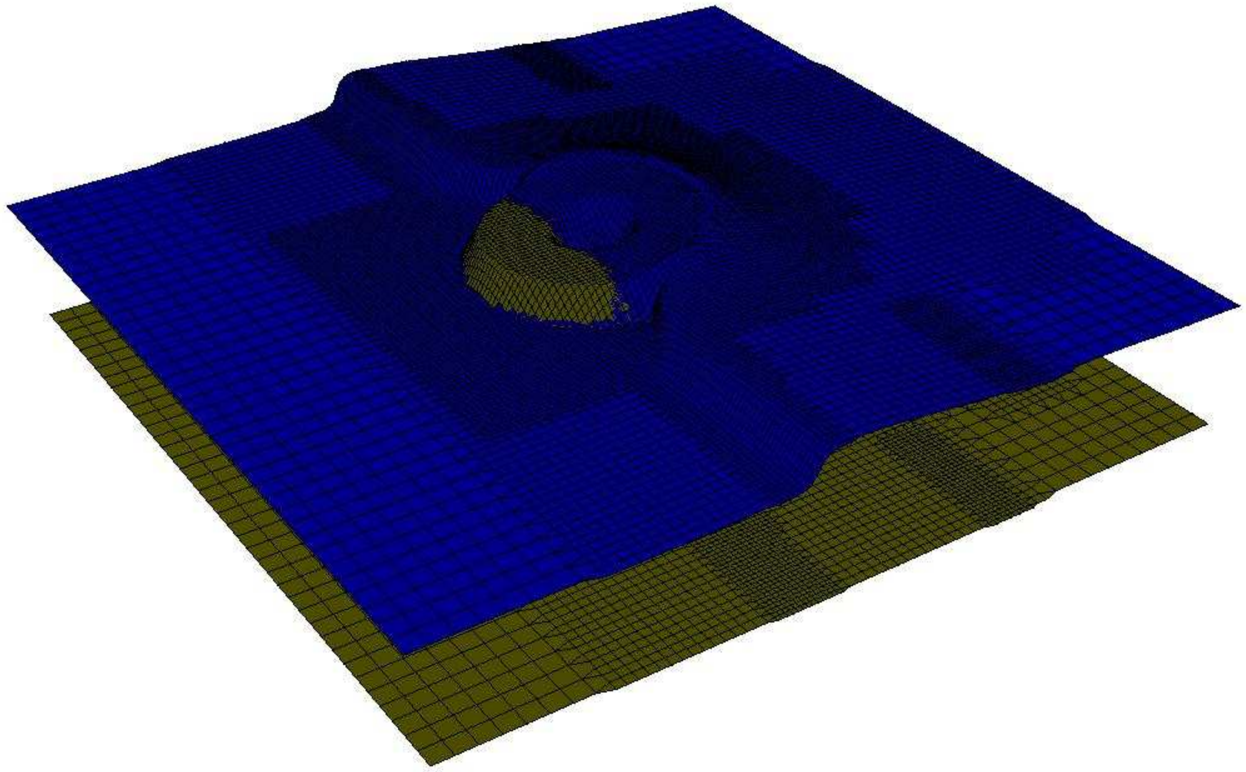

Figure 6: Example of the two-dimensional algorithm being used to visualise the flooding of a circular island by a tidal wave [5, 8]. The water surface and the bed topography are shown.

## 2.3 Three Dimensions

The three-dimensional case is more complicated because it involves a larger number of legitimate cell topologies. All of these are illustrated in Figures 3 and 4. It should be noted that, in practice, only configurations A, B, C, D, F, I and J will occur (and J requires the possibility that interfaces can occur between meshes which differ by more than one level of refinement). Configurations E, G, H and K require the possibility that meshes at the same *level* of refinement are refined anisotropically *and* variably across the domain. While anisotropy may be allowed (using different refinement factors in different directions), each level of mesh refinement invariably uses the same directional factors over the whole computational domain, otherwise the benefits of the structure are lost. Figure 7 shows an example of the results obtained when the algorithm has been applied in three space dimensions.

## 3 Summary

A simple algorithm has been presented which will convert an adaptive mesh, formed by a hierarchy of embedded structured meshes (or cells) in which the information is stored at (nominally) the cell-centres, into a single unstructured mesh which can be used for visualising the data properly.

It will not work for genuinely unstructured meshes because their connectivities is not known *a priori*. Also, if the information is stored at the cell vertices (as it commonly is), simpler methods are available for dealing with the hanging nodes. It has been applied in a variety of two- and three-dimensional situations to visualise coastal engineering and atmospheric data [4, 5, 6, 8].

It is also noted that the use of a tree data structure could achieve the same result but provide more flexibility. This is currently under investigation, to see whether it might be fast enough to be a useful approach for visualising the very large datasets which can be produced by the simulation software.

## References

[1] M.J.Berger and P.Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys.*, **82**:67–84, 1989.

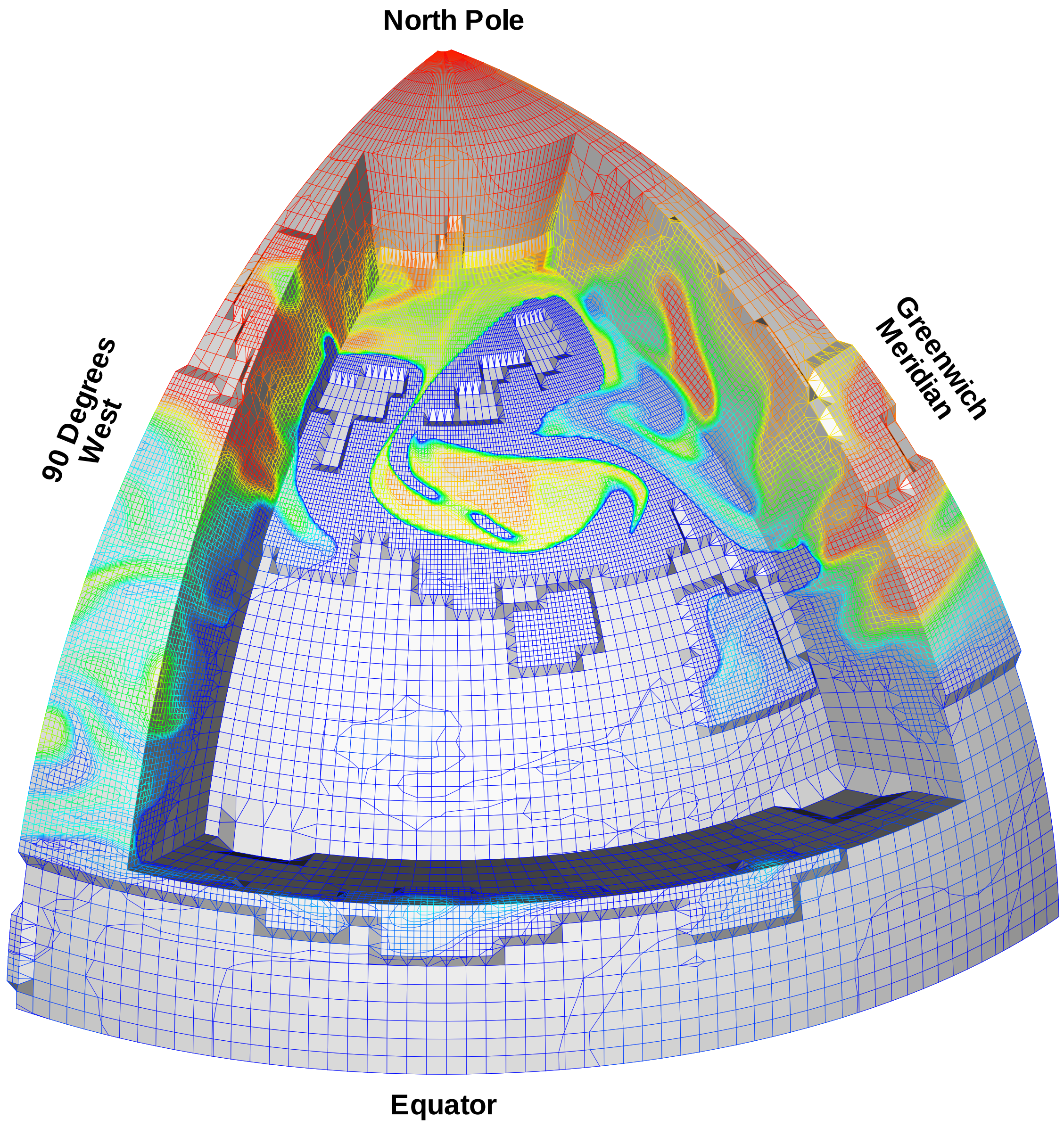

Figure 7: Example of the three-dimensional algorithm being used to visualise the advection of potential vorticity in the atmosphere over one quadrant of the northern hemisphere [6].

[2] M.J.Berger and J.Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, *J. Comput. Phys.*, **53**:482–512, 1984.

[3] J.J.Quirk, An adaptive algorithm for computational shock hydrodynamics, PhD thesis, College of Aeronautics, Cranfield Institute of Technology, 1991.

[4] M.E.Hubbard, Adaptive mesh refinement for three-dimensional off-line tracer advection over the sphere, *Int. J. Numer. Methods Fluids*, **40**:369–377, 2002.

[5] M.E.Hubbard and N.Dodd, A 2d numerical model of wave run-up and overtopping, *Coastal Engineering*, **47**(1):1–26, 2002.

[6] M.E.Hubbard and N.Nikiforakis, A three-dimensional, adaptive, Godunov-type model for global atmospheric flows, *Mon. Weather Rev.*, **131**(8):1848–1864, 2003.

[7] D.E.Tate, Visualisation of water wave simulations, Final Year Project Report, School of Computing, University of Leeds, 2007.

[8] A.T.Wilde, Visualisation of tsunami simulations, Final Year Project Report, School of Computing, University of Leeds, 2006.