ISOADVECTOR: A NEW GEOMETRIC VOF METHOD FOR ARBITRARY MESHES

JOHAN ROENBY[1], HENRIK BREDMOSE[2], HRVOJE JASAK[3]
[1]*DHI, Department of Ports and Offshore Technology, Denmark, jro@dhigroup.com*
[2]*Technical University of Denmark, Department of Wind Energy, Denmark, hbre@dtu.dk*
[3]*University of Zagreb, Department of Mechanical Engineering and Naval Architecture, Croatia hrvoje.jasak@fsb.hr*

Keywords: *geometric VOF, arbitrary unstructured mesh, free surface, multiphase flow, interFoam, sharp interface*

Introduction

We have developed a geometric VOF method for solving the sharp interface advection problem on unstructured meshes consisting of arbitrary polyhedral cells. The method, dubbed IsoAdvector, performs very well in terms of interface sharpness, volume conservation, shape preservation, boundedness of the volume fraction data, and insensitivity to the underlying mesh type and quality. IsoAdvector is explicit of nature limiting its usage to Courant numbers below 1. However, in contrast to MULES and other algebraic VOF methods we have tested, IsoAdvector performs well even with relatively large time steps close to this limit. Just as important, calculation times are similar to MULES, making the IsoAdvector method practically applicable in interFoam type solvers. The IsoAdvector employs an isosurface based interface reconstruction step. The advection step is based on simple geometric considerations of a plane passing a general polygonal face. In the following, we describe these steps in more detail, and present the results of two test cases.

Isosurface based interface reconstruction from volume fraction data

In most existing geometric VOF methods the local interface orientation is estimated using the numerically calculated gradient of the volume fraction data [1-4]. Since the volume fraction data represents the cell volume integration of a step function - whose gradient is a Dirac delta function - we believe that one should be very cautious with such numerical gradient estimates. Indeed, this is avoided with the numerical isosurface calculation employed in IsoAdvector for obtaining the interface orientation and position inside a cell. Since the isosurface calculation involves an interpolation of the cell centred volume fraction data to the cell vertices, the effective stencil for the reconstruction step consists of all surrounding cells with which a cell shares a vertex. Thus, the isosurface model for the fluid distribution inside a cell captures very well the general idea that, if, say, all neighbour cells to the left of a cell are filled with fluid, and all cells to the right are empty, then the fluid content inside our cell is most likely at the left side of the cell. The isosurface concept has the additional advantages that the procedure for numerical isosurface calculations is simple and fast and is by construction applicable to general unstructured meshes. In the IsoAdvector method, we calculate the isosurface inside all *surface cells*, where cell i is defined as a surface cell if its volume fraction, α_i, satisfies $\epsilon < \alpha_i < 1-\epsilon$, for some chosen tolerance (we usually set $\epsilon = 10^{-8}$). The numerically calculated isosurface inside a cell is a polygon cutting it into two subcells: One which is fully submerged in fluid, and another which is empty. We will refer to such a cell cutting polygon as an *isoface*.

Interface advection by fluxing fluid through mesh faces

The advection step of geometric VOF methods is often based on the concept of flux polyhedra [1-4]. The idea here is to first estimate the volume swept by the submerged part of a mesh face during a time step, and then find the intersections of the resulting flux polyhedrons with the mesh cells. The main problem with this approach is that it involves geometric calculations (volume intersections) that are both very complex to implement and computationally expensive. We have therefore developed a simpler approach, which we will briefly describe here: The polygonal perimenter of an isoface consists of straight line segments cutting (some of) the cell faces in two. We will refer to such a face cutting line as a *face-interface intersection line* (FIIL). For a surface cell, the reconstruction step thus gives us its isoface and the FIIL's of its faces at the beginning of a time step. To calculate the total volume of fluid transported through one such face during the time step, we must estimate the time evolution of this FIIL *within* the time step. This we do by estimating the motion of the isoface of the *upwind* cell of the face (because this is the cell from which the face receives fluid during the time step). In the current IsoAdvector implementation the isoface motion is estimated by interpolating the velocity data to the isoface's face centre, $\boldsymbol{x}_I$, and then finding the velocity component, U_I, normal to the isoface's face normal, $\boldsymbol{n}_I$. At a later time, t, we then approximate the isoface with the plane passing through the point $\boldsymbol{x}_I + U_I t \boldsymbol{n}_I$ and having normal $\boldsymbol{n}_I$. It is then straightforward to find the FIIL (and the corresponding submerged face area) at any time, t. With these assumptions, we can derive an analytical expression for the total volume of fluid passing a general polygonal face during a time step. It is a remarkable feature of the method that the resulting expression is exact in the case of a planar interface advected in a spatially and temporally constant velocity field.

Optional bounding procedure

We have tested the IsoAdvector method with a number of simple test cases in both 2D and 3D on structured and unstructured meshes. The method is flux consistent and it has been verified that it preserves volume to within machine

precision. We have also found that it is very good at preserving the shapes of volumes advected in a uniform velocity field, and at keeping the interface sharp (within one cell width). While the bounding properties are also good, the method can be stressed, e.g. by using very "ugly" cells and large time steps, to cause some degree of unboundedness. Therefore, an optional bounding procedure is introduced in the algorithm at the end of each time step. For cells going out of bound, this bounding procedure makes a small correction to the fluid exchanged with its neighbour cells such as to regain boundedness, while still preserving the volume of fluid.

Preliminary test results

In Figure 1, we show the result of passively advecting a spherical volume of fluid in a steady, uniform velocity field across an unstructured tetrahedral mesh. The sphere has been advected 16 radii with Courant number 0.5. The left panel shows the result obtained with IsoAdvector, which spends 6907 s on the simulation. The right panel shows the result obtained with MULES in 7306 s. Both MULES and IsoAdvector perform well in terms of volume conservation, boundedness, and sharpness, but IsoAdvector is significantly better at preserving the spherical shape of the fluid region.
In Figure 2, we show the result of releasing a circular droplet of water in air on a rectangular 2D mesh. The left panel shows the result after a few seconds using an interFoam solver, where MULES has been replaced by IsoAdvector. The right panel shows the corresponding result obtained with the original interFoam solver using MULES (Note: we use OpenFOAM-2.2.0 and the results may be different with newer versions). Contours of the volume fraction data are shown in red, green and blue. As is evident, IsoAdvector is very good at preserving the shape, whereas MULES leaves behind some amount of "water vapor" in the wake of the droplet. Since the air-water density aspect ratio is 1:1000, the small circular 10^{-3} contours (green) above the droplet in the MULES calculation contain at least as much water as air in terms of mass. This has a visible effect on the flow pattern, with wider wake compared to the IsoAdvector result.
The IsoAdvector code will be published as open source together with [5] in which more test results can be found.

In the talk, we will explain and illustrate the IsoAdvector concept and give further examples of its ability to accurately advect a sharp interface across structured and unstructured 2D and 3D meshes.

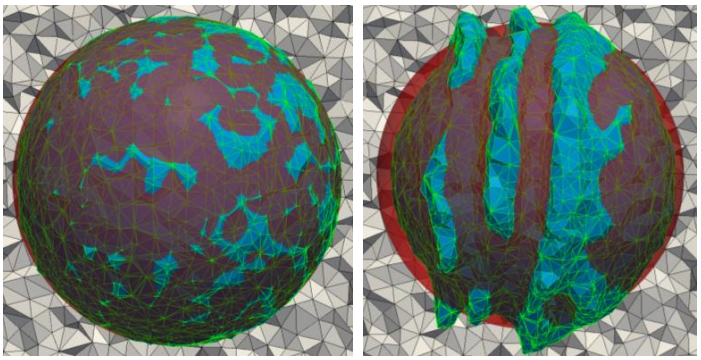

Figure 1: 0.5-isosurface (blue) of a spherical fluid volume advected 16 radii in a uniform velocity field on a tetrahedral mesh (background). Exact solution also shown (red). Left: IsoAdvector. Right: MULES.

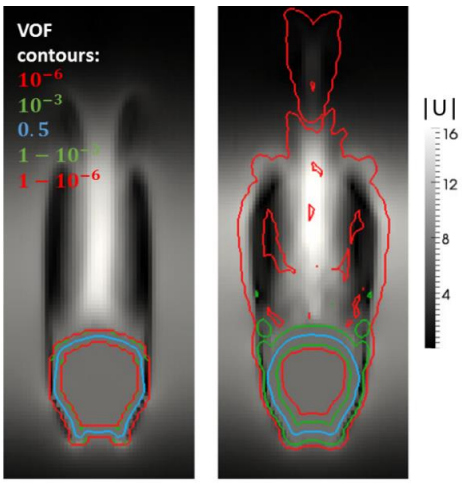

Figure 2: Volume fraction contours for a falling droplet. Left: IsoAdvector. Right: MULES.

Acknowledgements

This work was supported by The Danish Council for Independent Research via a Sapere Aude postdoc grant (DFF – 1337-00118B – FTP) and by The Danish Agency for Science, Technology and Innovation via the GTS grant to DHI.

References

[1] J. Hernández, J. López, P. Gómez, C. Zanzi, and F. Faura, "A new volume of fluid method in three dimensions—Part I: Multidimensional advection method with face-matched flux polyhedra," International Journal for Numerical Methods in Fluids, vol. 58, no. 8, pp. 897–921, 2008.

[2] J. López, C. Zanzi, P. Gómez, F. Faura, and J. Hernández, "A new volume of fluid method in three dimensions—Part II: Piecewise-planar interface reconstruction with cubic-Bézier fit," International Journal for Numerical Methods in Fluids, vol. 58, no. 8, pp. 923–944, 2008.

[3] T. Maric, H. Marschall, and D. Bothe, "voFoam - A geometrical Volume of Fluid algorithm on arbitrary unstructured meshes with local dynamic adaptive mesh refinement using OpenFOAM," arXiv:1305.3417 [physics], May 2013.

[4] L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva, "A 3-D Volume-of-Fluid advection method based on cell-vertex velocities for unstructured meshes," Computers & Fluids, vol. 94, pp. 14–29, May 2014.

[5] J. Roenby, H. Bredmose, and H. Jasak, "A Computational Method for Sharp Interface Advection," arXiv:1601.05392 [physics], Jan. 2016, (submitted).