DELFT UNIVERSITY OF TECHNOLOGY

REPORT 11-07

Evaluation of the interface-capturing algorithm of OpenFoam for the simulation of incompressible immiscible two-phase flow

F.Raees, D.R. van der Heul and C.Vuik

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2011

Evaluation of the interface-capturing algorithm of OpenFOAM for the simulation of incompressible immiscible two-phase flow

F.Raees, D.R. van der Heul and C.Vuik

September 19, 2011

Abstract

The Mass-Conserving Level-Set method combines the efficiency of a Level-Set algorithm with the mass conserving properties of the Volume Of Fluid method. It avoids the work intensive interface construction of the former method and imposes a mass-conserving correction to the distance function of the latter. The interface capturing algorithm implemented in OpenFOAM uses a *compressive* convection scheme for the evolution of the VOF colour function, as opposed to an interface reconstruction algorithm. Therefore, it can be assumed to match the efficiency of the MCLS method. Further analysis of the accuracy of the algorithm is required to make a fair comparison. In this report the accuracy will be evaluated for the simulation of incompressible, immiscible two-phase flow in two and three spatial dimensions. Three representative test cases are considered: The advection of a spherical bubble for an imposed, constant velocity field (2D), a rising (buoyant) bubble in a quiescent fluid (2D and 3D) and a stationary bubble in a stationary fluid (2D and 3D). The computed results are compared with results obtained with the Mass-Conserving Level-Set method of [8], benchmark results of [5] and other references. The compressive scheme accurately conserves mass, but shows large spurious currents for the test cases with surface tension. Additionally, the error in the predicted rise velocity of the gas bubble is large in comparison with that of the MCLS method.

1 Introduction:

OpenFOAM [1] is an Open Source application to solve a user defined mathematical model specified as a system of partial differential equations with appropriate initial/boundary conditions, without having to deal with many different aspects of the discretisation involved. Currently, many pre-defined models are available ranging from a simple linear transport equation to the turbulent Navier-Stokes equations. It's applicability is not limited to fluid dynamics, but includes e.g. solid mechanics, electromagnetic and even financial models.

An important aspect of OpenFOAM to improve efficiency is that it has different executables for different physical models, where only the necessary functionality is included. Within a certain physical model, many different options are available for spatial and temporal discretisation and the iterative method to invert the resulting linear systems. This means the user can get involved in choosing the numerical methods but does not necessarily have to.

OpenFOAM has the capacity to solve the partial differential equations on a multi-dimensional domain of arbitrary geometry and provides means for control of the mesh resolution for specific portions of the domain.

A simple text-based grid generator is part of OpenFOAM, that allows the definition of the geometry in terms of rectilinear polygons and cylinder surfaces. To define a computational grid on a more complicated domain, an external grid generator can be used or the OpenFOAM grid generator SnappyHexmesh. The latter uses a description of the geometry in the form of a set of polygons and uses a combination of advancing front grid generation and a truncated hexahedral mesh.

OpenFOAM has the capacity to import externally generated meshes from e.g. Gambit [2], Fluent [3] and the Open Source grid generator Gmsh. Results generated by OpenFOAM can be visualized using the Open Source application ParaView [4], which is internally linked to OpenFOAM as ParaFoam

The most important feature of the OpenFOAM is the Open Source environment, this provides it's users an opportunity to incorporate new discretisation schemes and solution techniques.

2 Efficient simulation of two phase flow:

To simulate the evolution of the interface between two immiscible fluids three basic approaches can be followed:

- Interface tracking
- Interface fitting
- Interface capturing

Only the last approach does not impose any restrictions on the evolution of the interface shape. Interface capturing methods generally use either a volume of fluid (VOF) or a level-set (LS) function to define the interface position, where in the former case the interface is only defined approximately. The VOF is relatively labour intensive, because it requires a costly interface reconstruction in each time step. However, as opposed to the level-set method it can be formulated in a strictly mass-conserving way. In the Mass-Conserving Level-Set (MCLS) method of Van der Pijl [8], the advantages of both VOF and LS are combined, while the costly interface reconstruction step is discarded from the algorithm. Currently, the MCLS is only formulated for a Cartesian grid, but for future projects this formulation has to be extended to a general boundary conforming (curvilinear, unstructured tetrahedral) discretisation.

A relatively recent development is the use of a *compressive scheme* for the discretisation of the volume fraction function (γ) conservation equation. In this approach an artificial and supplementary velocity field (U_r) is defined in the vicinity of the interface, in such a way that the local flow steepens the gradient of the volume fraction function and the interface resolution is improved. This is incorporated in the conservation equation for γ in the following form.

$$\frac{\partial \gamma}{\partial t} + \bigtriangledown \cdot (\gamma U) + \bigtriangledown \cdot (\gamma(1-\gamma)U_r) = 0$$

The last term on the right-hand side of the above equation is known as the artificial compression term and is only active at the interface. This is the approach currently used in OpenFOAM. A number of different approaches to define the compressive velocity field are reported in literature, e.g. CICSAM [7] and the currently used Multi- dimensional Universal Limiter with Explicit Solution (MULES), which limits the flux of the variables to guarantee a bounded-solution. The mixture mass flux is formulated as a function of these limited fluxes, and is used in a segregated pressure-based solver to calculate the velocity and pressure field.

We are interested in the extension of the Mass-Conserving Level-Set method for non-Cartesian grids, because of its superior accuracy with respect to the LS method and superior efficiency with respect to the VOF method. The MCLS method uses two stages: first it computes the level set function and in the second step it corrects the level set function to achieve mass-conservation. To implement these steps on an unstructured grid presents a challenge with respect to the second step that is mass-conservation. Using an unstructured grid, it is a difficult task to define a relation between the VOF function and the level-set function for an arbitrary polygonal cell intersected by the interface . However, implementation of the level-set function is straightforward.

Beside the use of a compressive term in OpenFOAM, there is another difference between OpenFOAM and MCLS. This difference is related to the arrangement of dependent variables in a cell. OpenFOAM uses a collocated arrangement for the variables and also there is no restriction on the number of faces bounding a cell. In MCLS we have a staggered arrangement of the variables on a Cartesian grid. Compressive schemes could be strong competitors in terms of efficiency with respect to the MCLS method. However, the accuracy has to be evaluated and compared to the MCLS method. The goal of the present study is the evaluation of the accuracy of the compressive scheme as used in OpenFOAM for the prediction of immiscible incompressible flow with respect to the accuracy of the MCLS method for a number of test cases that can be computed on a Cartesian grid.

The following test cases are used in order to evaluate the accuracy of the algorithm:

1. Advection of a gas filled spherical bubble in liquid, without buoyancy: The interface translates under the influence of a uniform velocity field.

2. Rising of a (spherical) gas filled bubble in a partially liquid filled container: Under the influence of buoyancy a gas filled bubble accelerates from rest until it reaches it's terminal velocity or breaks the surface.

3. Retaining a stationary spherical bubble in liquid: Under the influence of interfacial surface tension and without the effect of buoyancy small disturbance cause fluctuation in the bubble interface..

3 Test case 1: Advection of a gas filled bubble in liquid

This test is conducted in order to establish how the interface resolution changes when a bubble is convected, but also if the convection speed of the bubble actually matches the imposed velocity and the interface retains it's shape. In this test we are also interested to observe how the compressive scheme used in OpenFOAM mass is conserved.

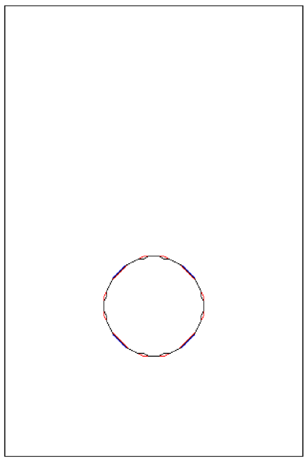
3.1 Two-dimensional computations

In this advection test we convect a spherical gas bubble of radius $0.0033m$ having centre at a height $0.01m$ from the bottom of the container. The dimensions of the container are $0.02m$ by $0.03m$. We have used the same density and the dynamic viscosity for the two fluids: density $\rho = 1000kg/m^3$ and dynamic viscosity $\mu = 0.001137kg/ms$. The surface tension is set to zero. The bubble is convected with a uniform vertical velocity field of 0.1 m/s upward. The density of the two phases is identical in order to avoid buoyancy effects.

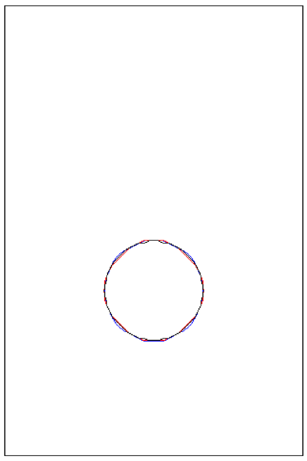
In Figure 1 we have shown the evolution of the boundary of the gas bubble using OpenFOAM at different times for the meshes 30×45, 60×90 and 120×180. This boundary is plotted as the isocontour of the 0.5 value of the volume fraction, as the exact position of the interface in undefined in a VOF model.

We tracked down the position of the center of mass of the gas bubble at different times and compared it with the exact solution, i.e. position of the center of a circle moving with the velocity equal to our advection velocity. Results for four different mesh resolutions are shown in Figure 2(a).

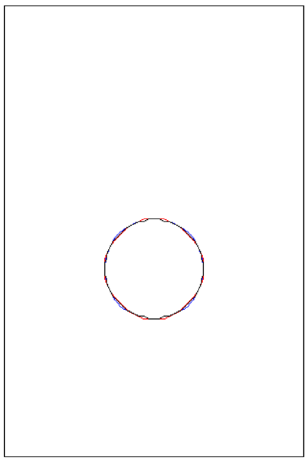
The error in the position of the interface at each time is defined as the difference of the position of the center of the circle and the center of the gas bubble. We have computed an error at different times for each mesh resolution. The errors for all different mesh resolutions are shown in Figure 2(b).



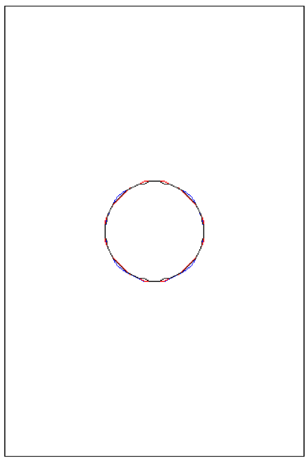
(a) t=0 seconds



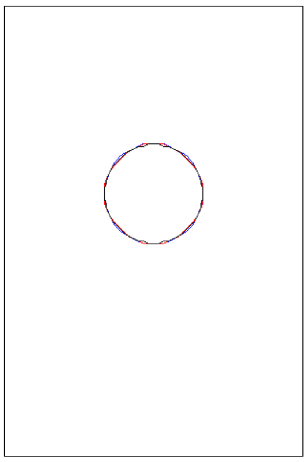
(b) t=0.01 seconds



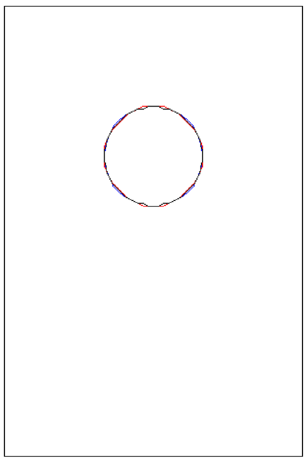
(c) t=0.025 seconds



(d) t=0.05 seconds

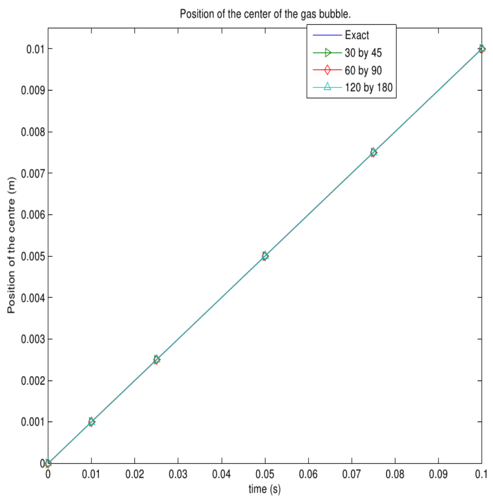


(e) t=0.075 seconds

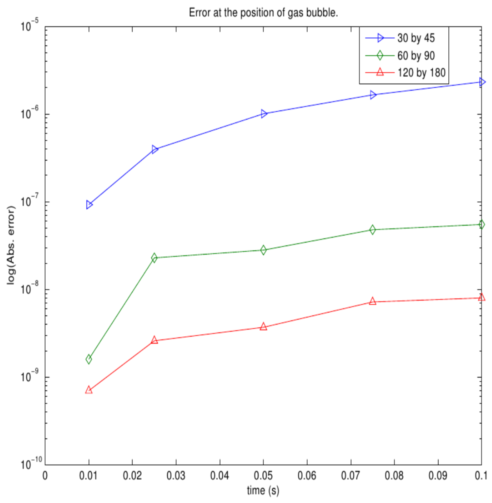


(f) t=0.1 seconds

Figure 1: Rising of gas bubble in liquid due to advection. Mesh size 30×45 : *blue*, 60×90 : *red and* 120×180 : *black*.



(a) Position of the center of the bubble.



(b) Absolute error in the position of the bubble.

Figure 2: Position and absolute error in the position of the center of gas bubble.

We observe from this test a monotonic convergence in the absolute error with increase of mesh resolution.

4 Test case 2: Rising of gas bubble in a liquid

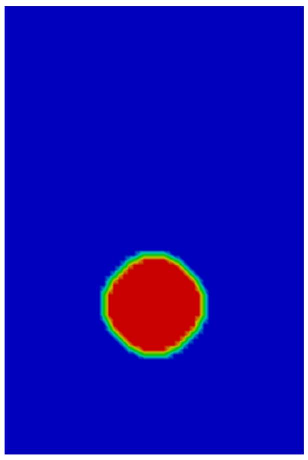
In the second test case we have considered the rising of a gas bubble in a liquid under the influence of buoyancy. This buoyancy is caused by the difference of the density of the two phases. In this case we have computed the pressure variation across the interface and the rise velocity in two and three dimensions.

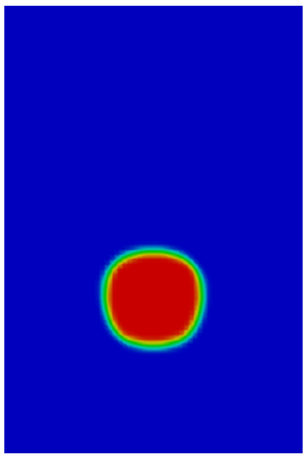
4.1 Two-dimensional computations

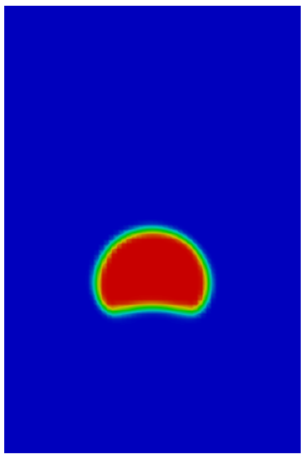
We have defined a gas bubble of radius $0.0033m$ having center at a height $0.01m$ from the bottom of the container. The dimensions of the container are $0.02m$ by $0.03m$. Material properties of the two fluids are: for the liquid density $\rho_l = 1000kg/m^3$, dynamic viscosity $\mu_l = 0.001137kg/ms$ and for the gas density $\rho_g = 1.226kg/m^3$, dynamic viscosity $\mu_g = 0.0000178kg/ms$. The gravitational acceleration is equal to $g = 9.8m/s^2$ and a surface tension $\sigma = 0.0728kg/s^2$ is used. Initially, the bubble is at rest and we have set homogeneous Dirichlet conditions for the velocity at the boundaries. Results are obtained for three different mesh resolutions. The sizes of the meshes are 30×45, 40×60 and 60×90. We consider the effect of surface tension in this case.

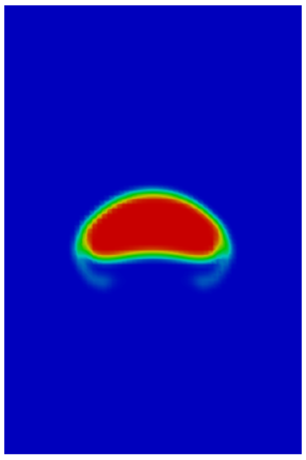
In Figure 3 the results of the rising gas bubble at a mesh resolution of 60×90 are presented at different times. These times are chosen such that a clear comparison can be made with the results presented in [8].

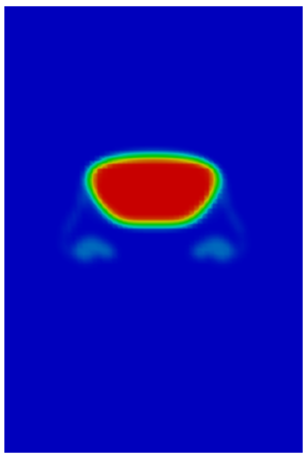
In Figure 4 the pressure distribution along a vertical cut at the centre of the domain is presented. The jump in the pressure across the interface must be balanced by the surface tension. It is observed that there are oscillations in the pressure across the interface at the beginning of the bubble translation. This is due to the fact that our initial condition is not a solution to the equation without gravity i.e. the sum of the pressure in the bubble and the surface tension does not match the hydrostatic pressure. However, after some time a smooth variation is observed. In Figure 4 results are shown upto t=0.013 seconds, after which we have a monotonic variation in pressure across the interface till the end of simulation. We have used a mesh of size 40×60 to present these results.


(a) t=0 seconds


(b) t=0.01 seconds


(c) t=0.025 seconds


(d) t=0.05 seconds


(e) t=0.075 seconds

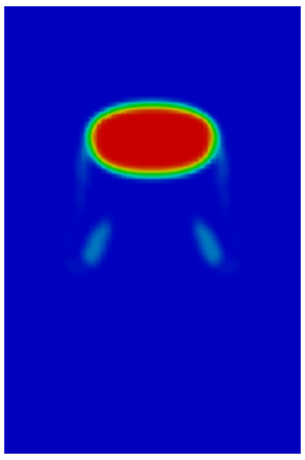
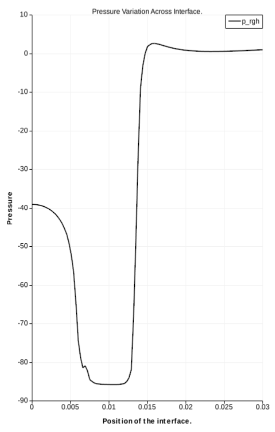
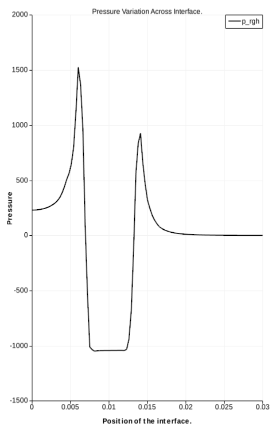

(f) t=0.1 seconds

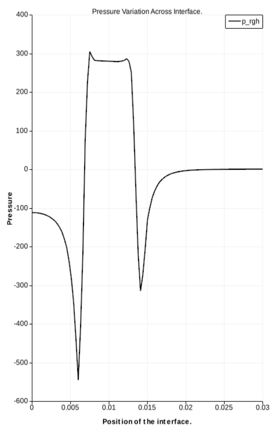
Figure 3: Rising of gas bubble in a liquid due to buoyancy.



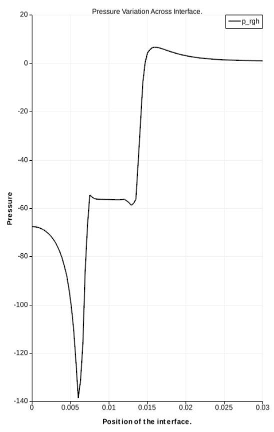
(a) t=0.0018 seconds



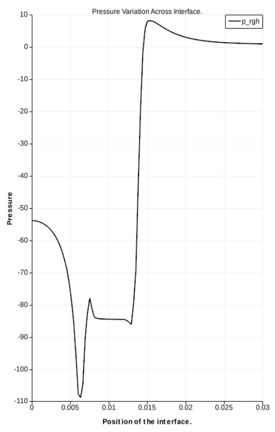
(b) t=0.003 seconds



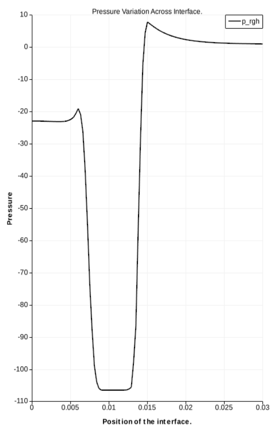
(c) t=0.004 seconds



(d) t=0.008 seconds

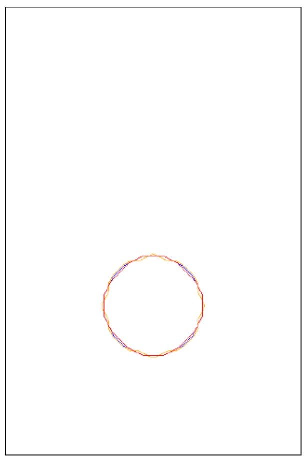


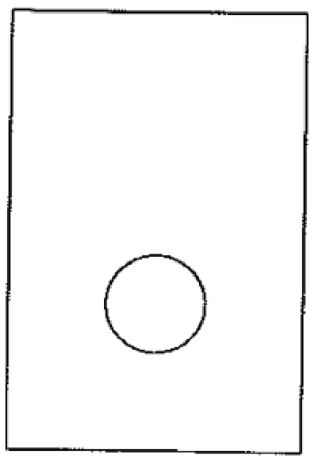
(e) t=0.01 seconds

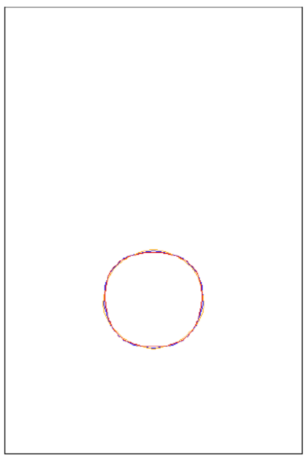


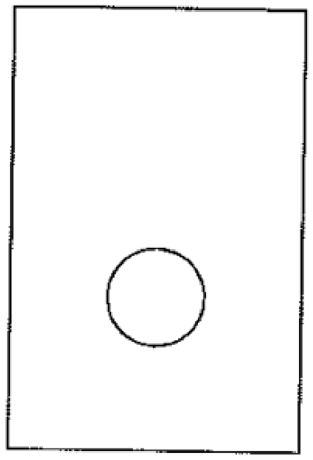
(f) t=0.013 seconds

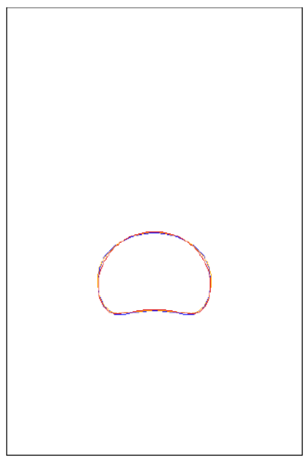
Figure 4: Pressure variation across the interface along vertical line.


(a) t=0 seconds


(b) t=0 seconds


(c) t=0.01 seconds


(d) t=0.01 seconds


(e) t=0.025 seconds

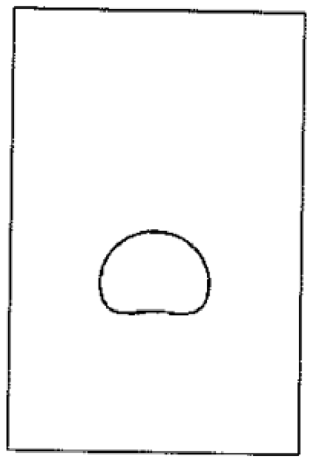
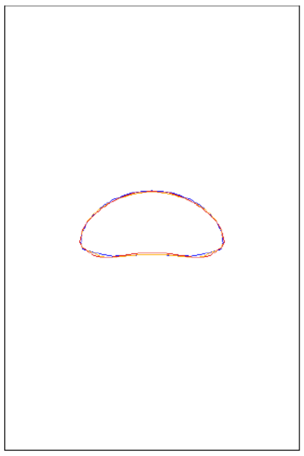
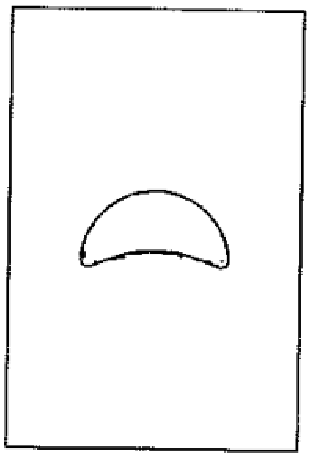

(f) t=0.025 seconds

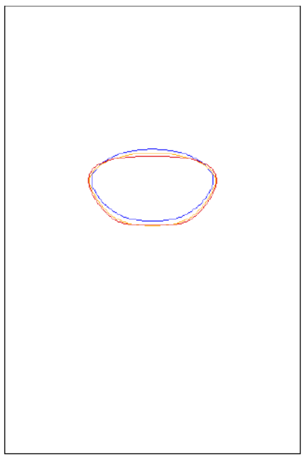
Figure 5: Rising of gas bubble in liquid at times 0 to 0.025 seconds, Using OpenFOAM(left) and MCLS(right). Meshes 30×45 : *blue*, 40×60 : *orange and* 60×90 : *red*.



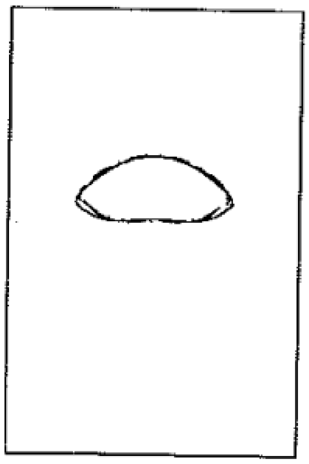
(a) t=0.05 seconds



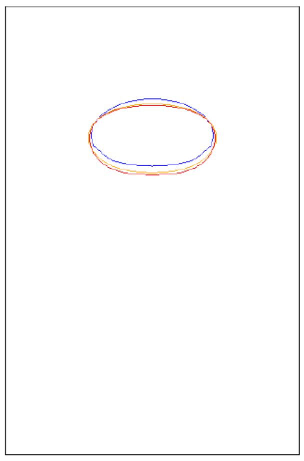
(b) t=0.05 seconds



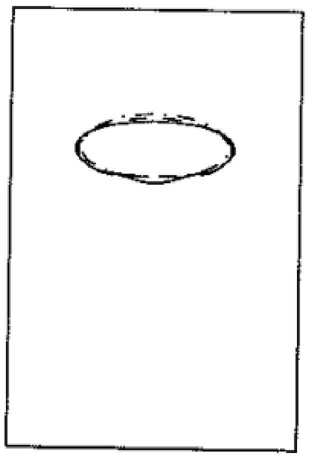
(c) t=0.075 seconds



(d) t=0.075 seconds



(e) t=0.1 seconds



(f) t=0.1 seconds

Figure 6: Rising of gas bubble in liquid at times 0.05 to 0.1 Seconds,Using Open-FOAM(left) and MCLS(right). Meshes 30×45 : *blue*, 40×60 : *orange and* 60×90 : *red.*

In Figure 5 and 6 results of the Mass-Conserving Level-Set method and the results of OpenFOAM at the mesh resolutions 30×45, $40\times$ and 60×90 are presented. Close comparison of the results obtained with OpenFOAM with the results of the MCLS method shows that up to time $t = 0.025$ seconds the predicted bubble shape is identical for all mesh resolutions. On later times there is a significant difference between the OpenFOAM results and the MCLS results in the downward pointing part of the bubble, i.e. the region of maximum interface curvature and therefore maximum surface tension. OpenFOAM does not accurately resolve the interface in this region. As illustrated in Figures 6(a) and 6(b) even further grid refinement will not change this behavior.

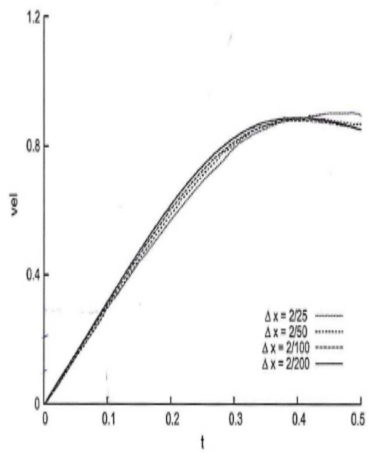
4.2 Rise velocity of two-dimensional bubble

In order to evaluate the accuracy of the prediction of the rise velocity using OpenFOAM for a two dimensional gas bubble, we have considered the results presented in [5, 6]. The numerical method in [6] uses an anti-diffusion method for obtaining a sharp interface and the other reference is a collection of benchmark solutions of two dimensional bubble flow. In this reference three different schemes are used to predict the rise velocity and other benchmark quantities. All the schemes of [5] predict nearly identical results. We have characterized the rise velocity of the bubble by the velocity of the center of gravity. The computational setup and the approach for computation of the rise velocity using ParaView as post-processor for OpenFOAM are discussed in Appendix F.

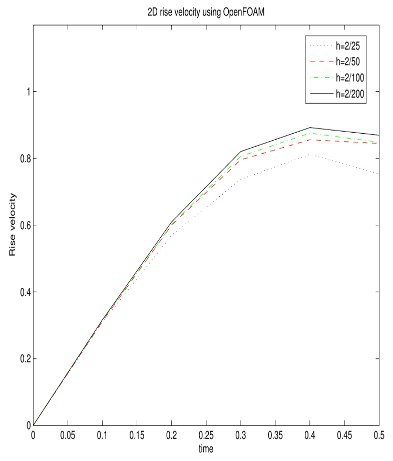
First we consider the case presented in [6]. The parameters for this case are mentioned in dimensionless form. Equivalent parameters for OpenFOAM in dimensional form are: gas bubble of radius 0.0025m centered at $(0.005m, 0.005m)$ from the bottom of the container. The dimensions of the container are $0.01m \times 0.02m$. Material properties of the two fluids are: $\rho_l = 1000kg/m^3$, $\mu_l = 1.0e - 06kg/ms$ and $\rho_g = 1.3kg/m^3$, $\mu_g = 1.6e - 05kg/ms$. The gravitational acceleration is set to $g = 9.8m/s^2$ and a value of surface tension $\sigma = 0.073kg/s^2$ is used. Furthermore, we have used four different mesh resolutions: 25×50, 50×100, 100×200 and 200×400.

In Figure 7(a) results for the anti-diffusion method [6] are presented for the grids mentioned above. In Figure 7(b) results for the rise velocity as predicted by OpenFOAM are presented. In comparison of these two results we observed that OpenFOAM generates approximately the same result as the anti-diffusion method predicts for all meshes till time t=0.2 secs. At later times a difference in the rise velocity is observed and this increases with refining of the mesh.

In order to investigate further we have considered a second reference [5]. As mentioned earlier it is a benchmark case and tested by three different groups, so the comparison of the results obtained from OpenFOAM and the results for the benchmark case will provide good understanding of the prediction capacity of OpenFOAM for the rise velocity. We have used the same approach for the computation of the rise velocity as mentioned in Appendix F for the first reference. We consider only one test case namely Case 1 mentioned in reference [5] and compared it with the results obtained by group TP2D (Transport Phenomena in 2D). All the groups mentioned in reference [5] generated the same result for the rise velocity



(a) Results of anti-diffusion interface steeping.



(b) Results of OpenFOAM.

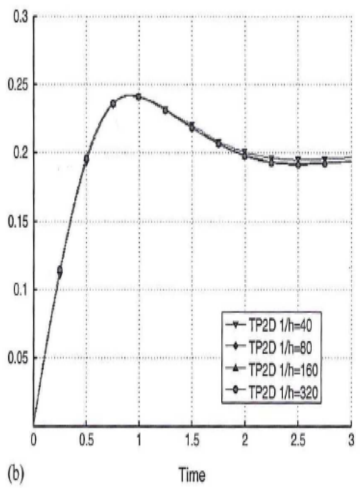
Figure 7: Rise velocities for two dimensional rising of gas bubble.

using their schemes, so comparison with result of one of the groups is sufficient to analyse the accuracy of OpenFOAM for the rise velocity.

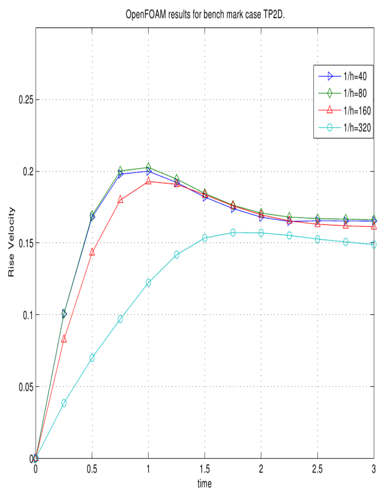
This benchmark case is presented in dimensionless form, We have used the following material properties: density for liquid and gas are $\rho_l = 1000kg/m^3$ and $\rho_g = 100kg/m^3$ respectively, kinematic viscosity for liquid and gas are $\nu_l = 0.01m^2/s$ and $\nu_g = 0.01m^2/s$ respectively. Gravitational value $0.98m/s^2$ and surface tension $24.5kg/s^2$. Further, we have used a $1m \times 2m$ container and a gas bubble of radius 0.25m centered at (0.5m,0.5m). We have used four different mesh sizes: 40×80, 80×160, 160×320 and 320×640. In Figure 8(a) results obtained by the TP2D group for case one are presented and in Figure 8(b) we have presented the results obtained by using OpenFOAM for all mesh sizes. Comparison indicates that OpenFOAM under-predicts the magnitude of the rise velocity for all the mesh sizes. Also, the underprediction of rise velocity increases with the increase of mesh resolution. A significant difference is observed for the mesh 320×640.

4.3 Rise velocity of three-dimensional bubble

The under prediction of the magnitude of the rise velocity for the two-dimensional cases by OpenFOAM, motivates us to investigate this also in three spatial dimensions. In order to do this we have considered a three-dimensional test case defined in [8]. In this case we have also considered the effect of a free surface. A no slip condition is imposed at the boundary of the container. We have considered a square shaped container having dimensions 0.01m by 0.01m by 0.01m, containing a gas bubble of radius 0.000125m, centered at (0.005m, 0.0025m, 0.005m) and the free surface lying at height of 0.0075m above the bottom of the container. Material properties of the fluids are: liquid density $\rho_l = 1000kg/m^3$, liquid dynamic viscosity $\mu_l = 1.137e-3kg/ms$ and gas density $\rho_g = 1.226kg/m^3$, gas dynamic



(a) Results of TP2D.



(b) Results of OpenFOAM.

Figure 8: Rise velocities for two-dimensional rising of gas bubble.

viscosity $\mu_g = 1.78e-5kg/ms$. The values $g = 9.8m/s^2$ and $\sigma = 0.0728kg/s^2$ are used for the gravitational acceleration and the surface tension, respectively. The mesh size is 64^3. Details of the computational setup can be found in Appendix F.

In Figure 10 the rise velocity obtained by using the MCLS method and results obtained for the rise velocity using OpenFOAM are presented.

Comparing both, it is observed that OpenFOAM is predicting a higher rise velocity. Due to this the bubble reaches the free surface earlier as compared to the MCLS. When the bubble reaches the free surface the computed rise velocity decreases. We have considered another three-dimensional experiment for the prediction of rise velocity mentioned in [6]. The method used in this reference provided a reasonable approximation for the rise velocity in the two-dimensional computation. Now we are interested to extend this comparison for the three-dimensional case. Material properties presented in [6] are all dimensionless. Equivalent dimensional parameters are: liquid density $\rho_l = 1387kg/m^3$, liquid dynamic viscosity $\mu_l = 2.8kg/ms$ and gas density $\rho_g = 1.226kg/m^3$, gas dynamic viscosity $\mu_g = 1.78e-5kg/ms$. The gravitational acceleration value $g = 9.8m/s^2$ and the surface tension $\sigma = 0.08kg/s^2$ are used. Further, we have used a container of dimensions 0.1305m by 0.0.2610m by 0.1305m, containing a gas bubble of radius 0.0131m, centered at (0.0653m,0.0261m,0.0653m) from the bottom of the container.

In Figure 11 the rise velocity obtained by using the anti-diffusion method of [6] and results obtained for the rise velocity using OpenFOAM are presented.

Figure 11 reflects that OpenFOAM under predicts the rise velocity as compared to the anti-diffusion method of reference [6]. OpenFOAM used the compressive scheme in order to prevent diffusion at the interface and the anti-diffusion scheme of [6] is designed to minimize diffusion. Due to the similarity of the methods, it is expected that they will

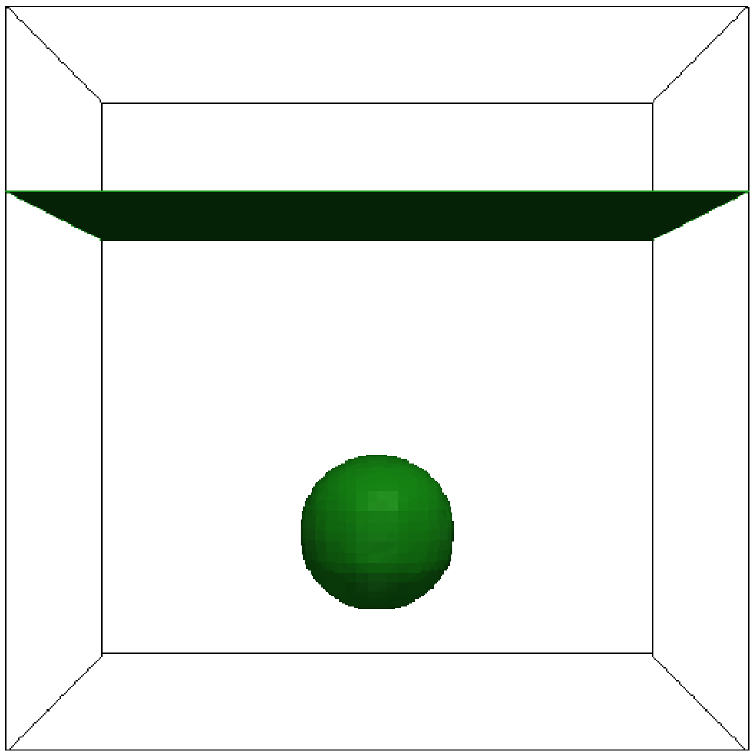


Figure 9: Setup for rising bubble with free surface, using 64^3 mesh grid.

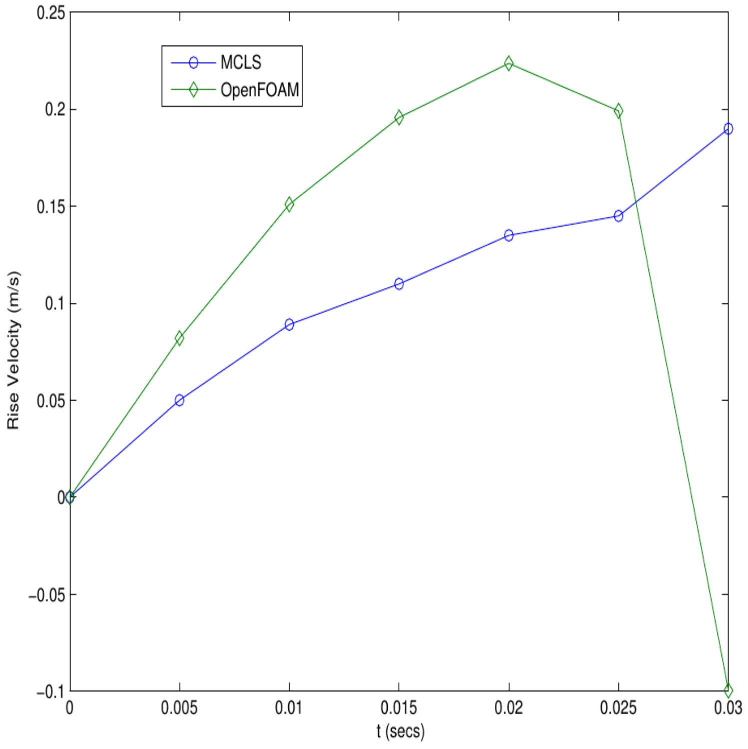


Figure 10: Rise velocity comparison between the MCLS method and OpenFOAM with free surface, 64^3 mesh grid.

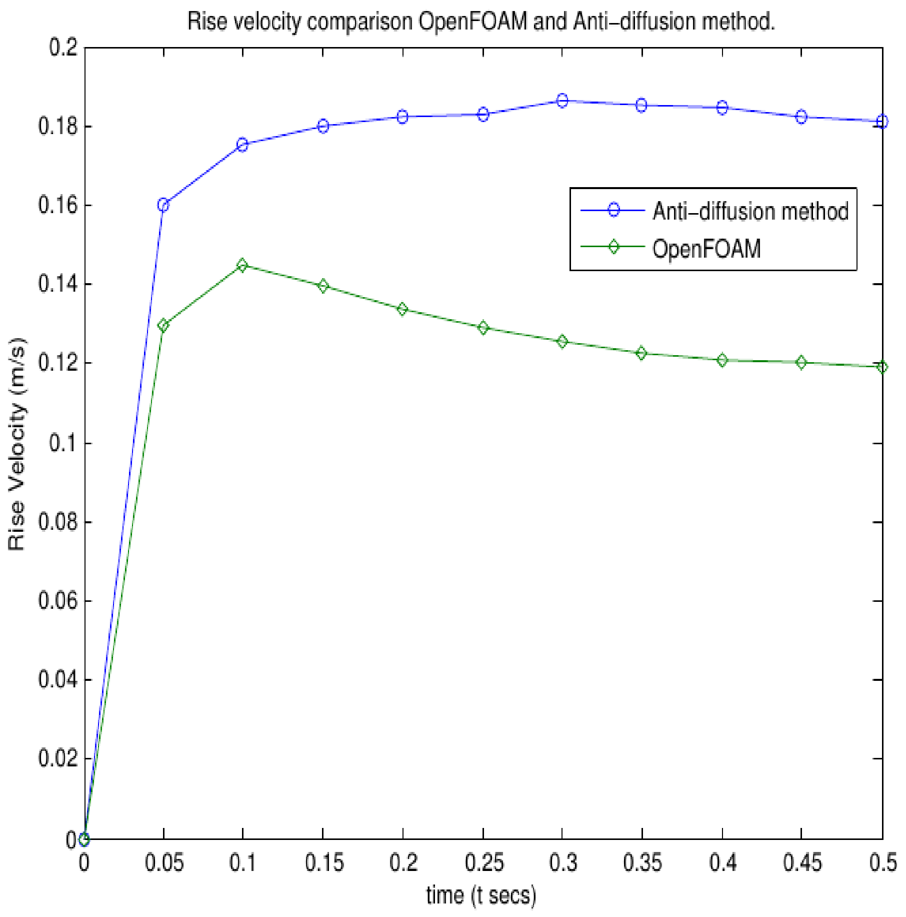


Figure 11: Rise velocity comparison for anti-diffusion method and OpenFOAM, for the mesh size $50 \times 100 \times 50$.

generate results which have a reasonable agreement, as we observed in the two-dimensional case. But it is observed that OpenFOAM now under predicts the rise velocity for the three-dimensional case.

5 Test case 3: Stationary gas bubble in a liquid

This test is conducted in order to analyse the spurious currents at the interface due to numerical diffusion and the lack of boundedness of the volume fraction in a cell. The spurious currents cause the interface to oscillate. This problem can be addressed by using upwind and downwind shifting of the volume fraction in order to achieve boundedness in the discretization scheme. Right now we are interested in the magnitude of the induced spurious currents due to the compressive scheme of OpenFOAM. In order to achieve this we made a setup in which we assume a stationary bubble ($\rho_l/\rho_g = 1$ and no advection velocity).

5.1 Two-dimensional computation

We have used a fixed mesh resolution of 60×90 and performed this experiment with different values of the surface tension in order to observe changes in the shape of the interface. The velocity near the interface in this case is only generated by the oscillation of the interface. This is also known as a parasitic current. In order to observe the effect of surface tension, we have used three different values of surface tension. These values are $\sigma = 0.03kg/s^2$, $\sigma = 0.07kg/s^2$ and $\sigma = 0.11kg/s^2$. The surface tension affects the magnitude of the pressure jump across the interface.

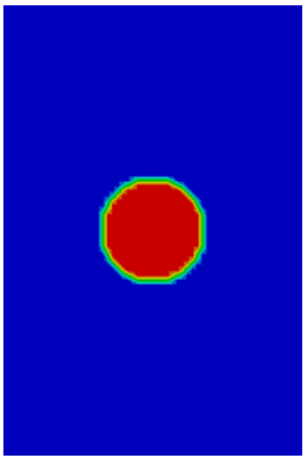
In Figure 12 we have shown the result obtained for the surface tension $\sigma = 0.07kg/s^2$. In Figure 13 plots of the isocontour of the volume fraction at 0.5 of the gas bubble are shown for all the values of the surface tension under consideration. We have used the same mesh resolution of 60×90 for all the cases.

In Figure 14 velocity variations due to parasitic currents at the interface are shown for three different values of the surface tension and at three different times of simulation. No large deformation in the shape of the interface for different values of surface tension is observed. However, there exist wiggles at the interface. Due to these wiggles the magnitude of the velocity around the interface increases with the increase of surface tension.

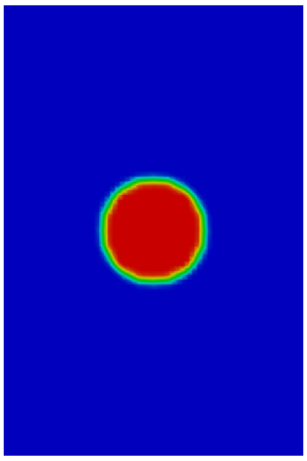
5.2 Three dimensional computation

In order to further investigate the parasitic currents generated by the compressive scheme of OpenFOAM in a quantitative manner and compare it with the results of the MCLS method, we have conducted the three dimensional Laplace test presented in [8]. For this test we have considered a sphere with radius $0.25m$ placed in the center of a cubic domain with dimensions $1m \times 1m \times 1m$. The flow is initially at rest. Since the initial condition satisfies the steady state problem, all velocities are due to parasitic currents. The value of gravitational acceleration and the material constant are $g = 0m/s^2$, $\sigma = 0.01kg/s^2$ respectively. $\rho_l/\rho_g = 1$ and the viscosity for both the fluids is set to $0.1kg/ms$. In order to understand the effect of mesh resolution on the magnitude of the parasitic current we have considered three different mesh sizes: 32^3, 64^3 and 96^3.

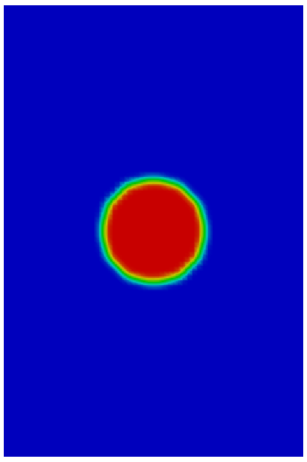
For quantitative analysis we have considered the maximum value of the parasitic current. The magnitude of the parasitic current *increases* with the *increase* of the mesh resolution. For a proper discretization of the VOF function the magnitude of these currents has to decreased, as the mesh size increases. The computational setup for determination of the maximum velocity due to parasitic currents using ParaView as post-processor is given in Appendix E. In Figure 15 the experimental setup is shown for the Laplace test for the grid size 64^3. The sphere represents the 0.5 isocontour of the volume fraction function.



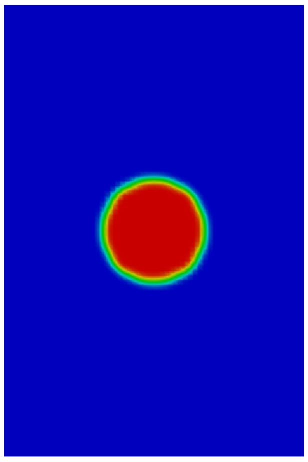
(a) t=0 seconds



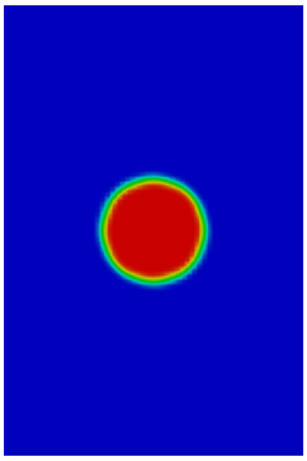
(b) t=0.01 seconds



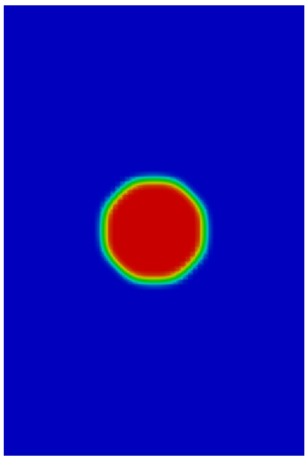
(c) t=0.025 seconds



(d) t=0.05 seconds

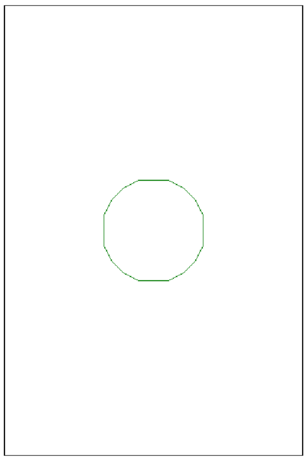


(e) t=0.075 seconds

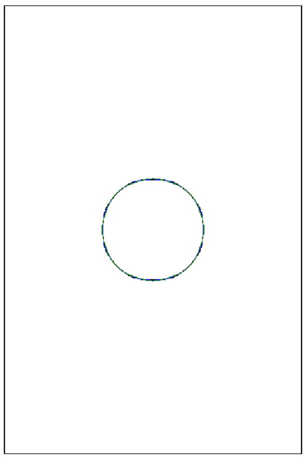


(f) t=0.1 seconds

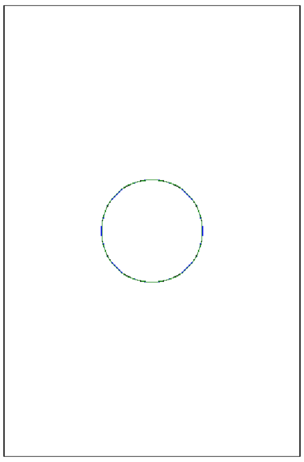
Figure 12: Stationary gas bubble in liquid with $\sigma = 0.07 kg/s^2$.



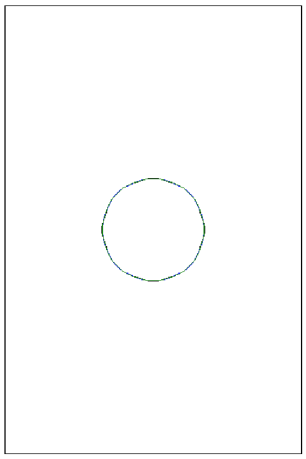
(a) t=0 seconds



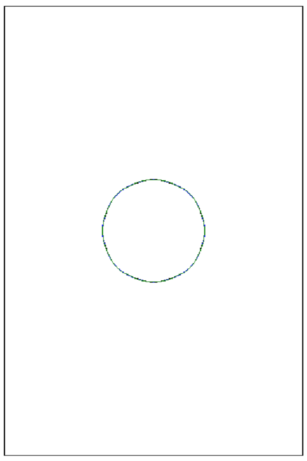
(b) t=0.01 seconds



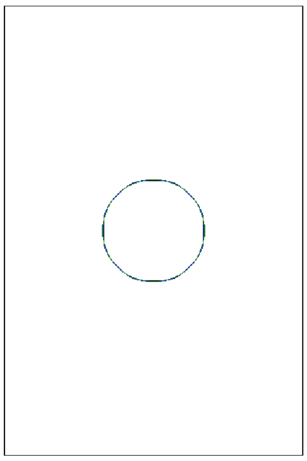
(c) t=0.025 seconds



(d) t=0.05 seconds

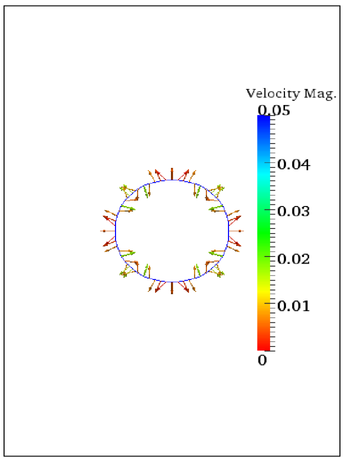


(e) t=0.075 seconds

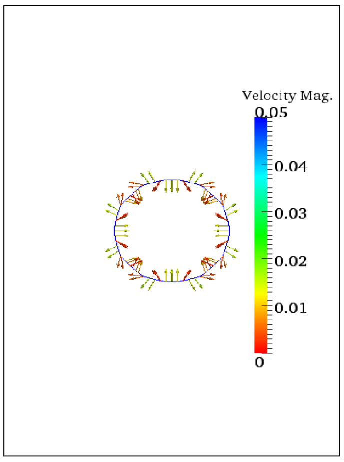


(f) t=0.1 seconds

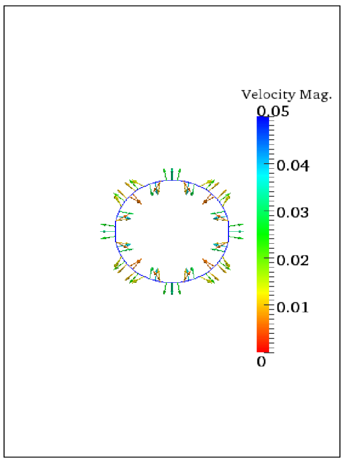
Figure 13: Stationary gas bubble in liquid with $\sigma = 0.03kg/s^2$ (green color), $\sigma = 0.07kg/s^2$ (blue color) and $\sigma = 0.11kg/s^2$ (black color).



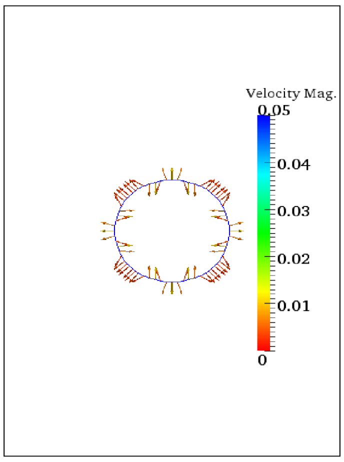
(a) t=0.025sec, $\sigma = 0.03kg/s^2$



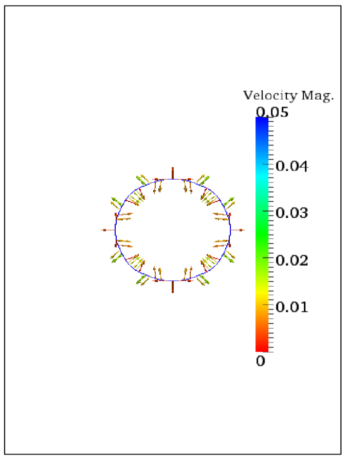
(b) t=0.025sec, $\sigma = 0.07kg/s^2$



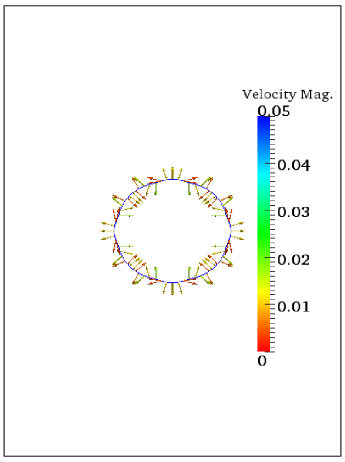
(c) t=0.025sec, $\sigma = 0.11kg/s^2$



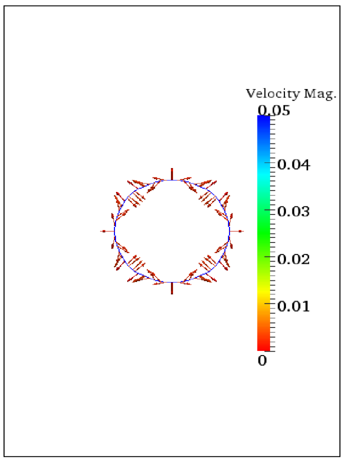
(d) t=0.05sec, $\sigma = 0.03kg/s^2$



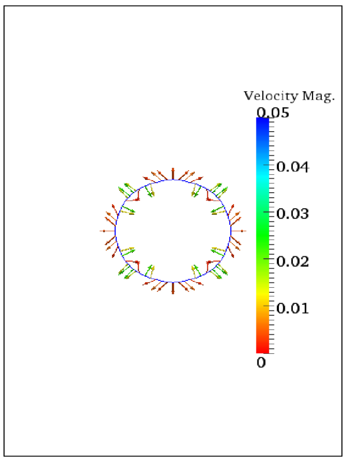
(e) t=0.05sec, $\sigma = 0.07kg/s^2$



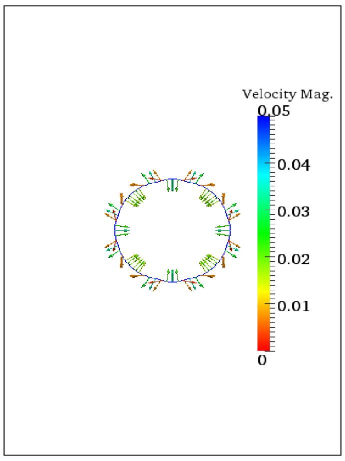
(f) t=0.05sec, $\sigma = 0.11kg/s^2$



(g) t=0.075sec, $\sigma = 0.03kg/s^2$



(h) t=0.075sec, $\sigma = 0.07kg/s^2$



(i) t=0.075sec, $\sigma = 0.11kg/s^2$

Figure 14: Magnitude of the velocity variation across the interface of stationary gas bubble at three different surface tensions.

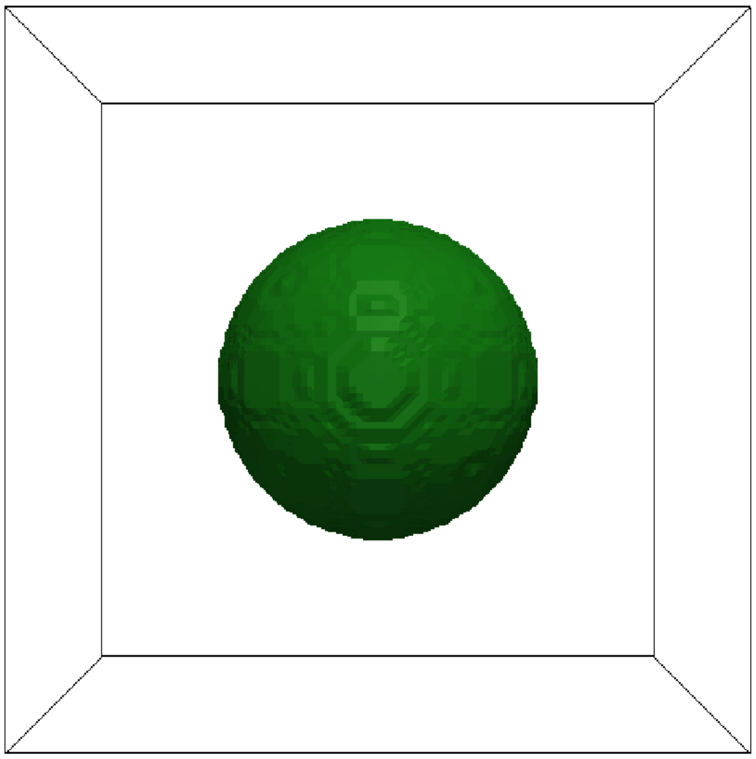
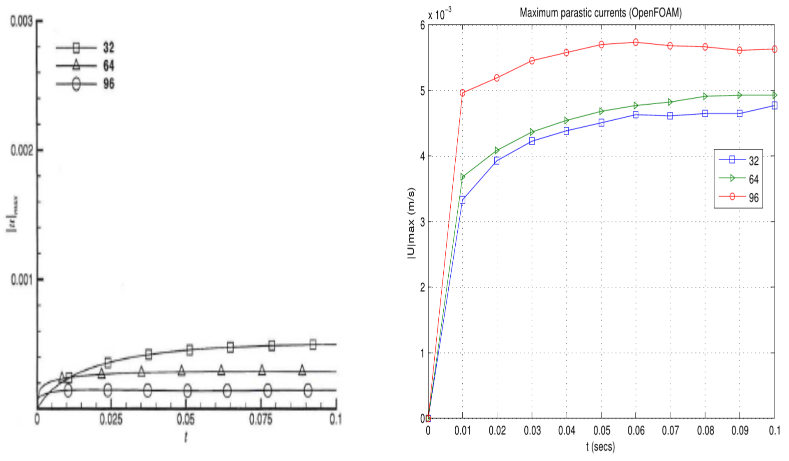


Figure 15: Setup for parasitic currents for 96^3 mesh grid.



(a) Maximum parasitic currents for the MCLS method.

(b) Maximum parasitic currents for OpenFOAM .

Figure 16: Comparison of maximum parasitic currents for the MCLS method and OpenFOAM.

Figure 16(a) shows the time evolution of the maximum value of the parasitic current computed with the MCLS method and Figure 16(b) with OpenFOAM at three different mesh sizes. The magnitude of the parasitic currents generated by the scheme used in OpenFOAM is one order of magnitude larger than for the MCLS scheme and also it is observed that there is no convergence in the maximum value of the parasitic currents with respect to time for different mesh size.

Figure 17 shows the maximum magnitude of the currents at the time $t = 0.1$ seconds for the three different grids. It is observed that for the MCLS method the maximum value of the parasitic current decreases as compared to OpenFOAM, for the increase in the mesh resolution. This test revealed that the compressive scheme used in OpenFOAM for the computation of volume of fraction needs further improvement so that it will generate smaller parasitic currents at the interface.

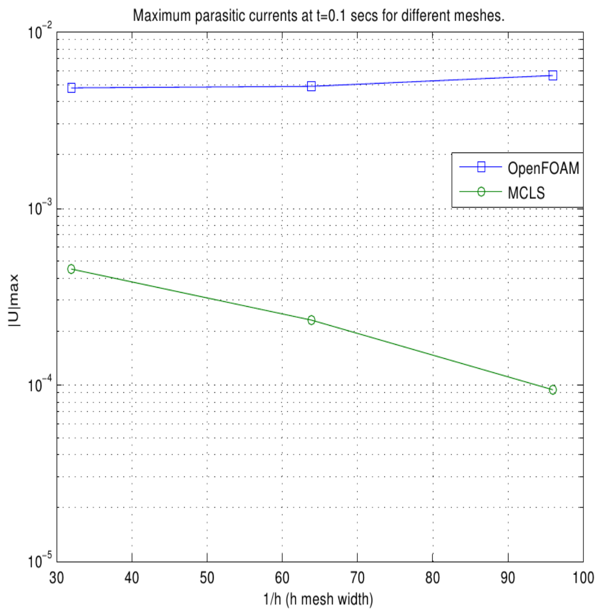


Figure 17: Maximum parasitic currents for all meshes at time t=0.1 seconds.

6 Conclusion

OpenFOAM uses a non conventional approach to the Volume of Fluid method, in the sense that it uses a compressive scheme to accurately predict the interface between the two phases, as opposed to a geometrical reconstruction that is commonly used. This means the algorithm is far less costly to apply, but if the accuracy of conventional PLIC or SLIC methods can be matched remains to be investigated.

Therefore, the basic purpose of this study is to evaluate the accuracy of the compressive scheme, used in OpenFOAM for discretisation of the Volume of Fluid color function. In order to achieve this we have performed different tests using OpenFOAM and compared the outcome with a number of different results from literature. Our study revealed that OpenFOAM will give accurate results for predicting the position of the interface on relatively high resolution meshes. This will increase the time of computation, but the basic scheme itself is not very computationally intensive, balancing off the efficiency.

We obtain good agreement of results for the rise velocity of a buoyancy driven flow in the two-dimensional case in comparison with results obtained with a similar *anti-diffusion* scheme from literature but for the three-dimensional case it is under predicting the rise velocity. Compared to the MCLS method OpenFOAM *over* predicts the rise velocity. The validity of the schemes used in OpenFOAM for three-dimensional setup demands more study. Relatively strong spurious currents are observed at the interface. The magnitude of the currents is one order of magnitude larger than is observed for the MCLS and the magnitude increases with mesh resolution. This clearly indicates that the compressive term of OpenFOAM demands more attention and needs to be further developed in order to meet the accuracy of the MCLS.

A Settings for OpenFOAM

In OpenFOAM simulation for the fluid flows are done by providing the information about domain, initial and boundary values for velocity and pressure, fluid fraction in a cell and other related informations in the form of files and directories. These directories consist of three sub directories 0 sub directory, constant sub directory and system sub directory. Each of these directories contains different information about solution procedure. We will explain each sub directory with reference to our case in the following section.

A.1 0 sub directory

In this sub-directory, all the initial values are given. All the patches that have been named in the polyMeshDict should be assigned a value here. This directory contains the three dictionaries alpha1, U and p_rgh . The Alpha1 dictionary sets the information about the position of the two fluids in the domain. U and p_rgh set the value of velocity and pressure through out the domain. Each of the dictionaries with reference to our case is explained as follows.

A.1.1 alpha1 dictionary for fluid settings

Using this dictionary we can set the fluids position in the domain. We have two fluids one is liquid which is labelled 1 and other is gas which is labelled as 0. In our case we do not have interaction of the interface and the wall of the container so we imposed a zero gradient condition in order to set static contact angle to 90 degrees. Alpha1 represents the volume fraction and it is dimensionless.

```
FoamFile  version 2.0;
format ascii;
class volScalarField;
location "0";
object alpha1;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions [0 0 0 0 0 0 0];
internal Field uniform 0;
boundary Field
wallLeft
type zeroGradient;
wallRight
type zeroGradient;
wallBottom
type zeroGradient;
```

```
atmosphere
type inletOutlet;
inletValue uniform 1;
value uniform 1;
frontAndBack
type empty;
// ************************************************************ //
```

A.1.2 U dictionary for velocity settings

We have set the boundary condition for the velocity at the walls to zero and for the advection case we assigned the vertical velocity of magnitued 0.1 m/s.

```
FoamFile
version 2.0;
format ascii;
class volVectorField;
location "0";
object U;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions [0 1 -1 0 0 0 0];
internal Field uniform (0 0 0);
boundary Field
wallLeft
type fixedValue;
value uniform (0 0 0);
wallRight
type fixedValue;
value uniform (0 0 0);
wallBottom
type fixedValue;
value uniform (0 0 0);
atmosphere
type pressureInletOutletVelocity;
value uniform (0 0 0);
frontAndBack
type fixedValue;
value uniform (0 0 0);
```

```
// ************************************************************ //
```

A.1.3 *p_rgh* dictionary for pressure settings

We have set the *buoyant pressure* boundary condition for the pressure field. This calculates the normal gradient from the local density gradient.

```
FoamFile
version 2.0;
format ascii;
class volScalarField;
location "0";
object p_rgh;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions [1 -1 -2 0 0 0 0];
internal Field uniform 0;
boundary Field
wallLeft
type buoyantPressure;
gradient uniform 0;
rho rho;
value uniform 10;
wallRight
type buoyantPressure;
gradient uniform 0;
rho rho;
value uniform 10;  wallBottom  type buoyantPressure;
gradient uniform 0;
rho rho;
value uniform 10;
atmosphere
type totalPressure;
rho rho;
psi none;
gamma 0;
p0 uniform 1;
value uniform 1;
frontAndBack
```

```
type empty;
// ************************************************************ //
```

B Constant subdirectory

The folder contains information about the properties of the two fluids, information about mesh construction and gravitational acceleration. It has a file known as transportProperties. This file contains information about two fluids properties like density, viscosity and etc. This directory contain an other sub directory polyMesh. This directory contains information about mesh generation. Each of these files with reference to our case will be explained below.

B.1 transportProperties dictionary

In this dictionary we provide all the fluid properties that are needed. Phase 1 represents liquid and phase 2 represents gas. We provide the value of kinematic viscosity which is obtained by dividing the dynamic viscosity by the density of the fluid. Additionally, the value of the surface tension is specified.

```
FoamFile
version 2.0;
format ascii;
class dictionary;
location "constant";
object transportProperties;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * // phase1
transport Model Newtonian;
nu nu [ 0 2 -1 0 0 0 0 ] 1.137e-06;
rho rho [ 1 -3 0 0 0 0 0 ] 1000;
CrossPowerLawCoeffs
nu0 nu0 [ 0 2 -1 0 0 0 0 ] 1.137e-06;
nuInf nuInf [ 0 2 -1 0 0 0 0 ] 1.137e-06;
m m [ 0 0 1 0 0 0 0 ] 1;
n n [ 0 0 0 0 0 0 0 ] 0;
BirdCarreauCoeffs
nu0 nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
nuInf nuInf [ 0 2 -1 0 0 0 0 ] 1.137e-06;
k k [ 0 0 1 0 0 0 0 ] 99.6;
n n [ 0 0 0 0 0 0 0 ] 0.1003;
```

```
phase2
transportModel Newtonian;
nu nu [ 0 2 -1 0 0 0 0 ] 1.4519e-05;
rho rho [ 1 -3 0 0 0 0 0 ] 1.226;
CrossPowerLawCoeffs
nu0 nu0 [ 0 2 -1 0 0 0 0 ] 1.137e-06;
nuInf nuInf [ 0 2 -1 0 0 0 0 ] 1.137e-06;
m m [ 0 0 1 0 0 0 0 ] 1;
n n [ 0 0 0 0 0 0 0 ] 0;
BirdCarreauCoeffs
nu0 nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
nuInf nuInf [ 0 2 -1 0 0 0 0 ] 1.137e-06;
k k [ 0 0 1 0 0 0 0 ] 99.6;
n n [ 0 0 0 0 0 0 0 ] 0.1003;
sigma sigma [ 1 0 -2 0 0 0 0 ] 0.0728;
// ************************************************************ //
```

B.2 enviromentalProperties dictionary

In this dictionary the gravitational acceleration is defined. We have specified $g = 9.8\frac{m}{s^2}$.

```
FoamFile
version 2.0;
format ascii;
class uniformDimensionedVectorField;
location "constant";
object g;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions [0 1 -2 0 0 0 0];
value ( 0 -9.81 0 );
// ************************************************************ //
```

For the case of advection and the stationary bubble case we neglect the effect of gravity by setting it equal to zero.

B.3 blockMesh dictionary

All the geometry of the problem is handled by the dictionary. The first line defines the conversion scale to meter. OpenFOAM uses S.I system and we need to provide the physical quantities as per this system or we can define a conversion factor in this first line. After that Cartesian coordinates of the vertices of the blocks are specified and then connectivity of the vertices.

```
FoamFile
version 2.0;
format ascii;
class dictionary;
object blockMeshDict;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 0.001;
vertices
( (-10 0 0)
(0 0 0)
(10 0 0)
(10 30 0)
(0 30 0)
(-10 30 0)
(-10 0 0.000001)
(0 0 0.000001)
(10 0 0.000001)
(10 30 0.000001)
(0 30 0.000001)
(-10 30 0.000001)
);
blocks ( hex (1 4 5 0 7 10 11 6) (45 15 1) edgeGrading ( 1 1 1)
hex (2 3 4 1 8 9 10 7) (45 15 1) edgeGrading ( 1 1 1) )
edges
(
);
patches
(
wall wallLeft
(
(0 5 11 6)
)
wall wallRight
(
(8 9 3 2)
)
wall wallBottom
( (1 0 6 7)
(2 1 7 8)
)
patch atmosphere
( (4 10 11 5)
(3 9 10 4)
)
empty frontAndBack
( (1 4 5 0)
(7 10 11 6)
(1 2 3 4)
(7 8 9 10)
)
);
mergePatchPairs
(
);
// ************************************************************ //
```

C System sub directory

In this directory parameters for the solution procedure for the momentum equation and transport equation are defined. This directory also contains information about time stepping and saving of data. All these operation will done by using the following four dictionaries.

C.1 controlDict dictionary

All the parameters for starting time, ending time and time step as well as method of saving data is defined in this dictionary. The first line contains information about the solver of the problem. We have considered laminar two-phase flow and therefore used the interFoam solver. We have set the start time to zero seconds and end time to 0.1 seconds with the time step of 0.001 seconds. Write Interval is 0.001 seconds, that is solution will be recorded after every 0.001 seconds. We have set the adjustable time to yes with this solver automatically adjust the time in order to maintain the CFL number defined in maxCo.

```
FoamFile
version 2.0;
format ascii;
class dictionary location "system";
object controlDict;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application interFoam;
startFrom startTime;
startTime 0;
stopAt endTime;
endTime 0.1;
deltaT 0.001;
writeControl adjustableRunTime;
writeInterval 0.001;
purgeWrite 0;
writeFormat ascii;
writePrecision 6;
writeCompression uncompressed;
timeFormat general;
timePrecision 6;
runTimeModifiable yes;
adjustTimeStep yes;
maxCo 0.1;
maxAlphaCo 0.1;
maxDeltaT 1;
// ************************************************************ //
```

C.2 fvSchemes dictionary

In this dictionary all the discretization schemes are defined. In interface tracking we need to solve the momentum equation and a transport equation for VOF color function. In our case we have used Van Leer schemes for the discretization of divergence term in both the equations. The lapalcian term in momentum equation is discretised by using a central difference scheme. Time discretization is done by using the Crank Nicholson scheme.

```
FoamFile
version 2.0;
format ascii;
class dictionary;
location "system";
object fvSchemes;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes
default crankNicholson 0.9;
gradSchemes
default Gauss linear;
grad(U) Gauss linear;
grad(alpha1) Gauss linear;
divSchemes
div(rho*phi,U) Gauss limitedLinearV 1;
div(phi,alpha) Gauss vanLeer;
div(phirb,alpha) Gauss interfaceCompression;
laplacianSchemes
default Gauss linear corrected;
interpolationSchemes
default linear;
snGradSchemes
default corrected;
fluxRequired
default no;
```
$p_r gh$;
```
pcorr;
alpha1;
// ************************************************************ //
```

C.3 fvSolution dictionary

All the parameters related to the solution of linear system of equation are specified in this dictionary. All linear system are solved using the preconditioned conjugate gradient (PCG) method.

```
FoamFile
version 2.0;
format ascii;
class dictionary;
location "system";
object fvSolution;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers
pcorr
solver PCG;
preconditioner DIC;
tolerance 1e-10;
relTol 0;
```
$p_r gh$
```
solver PCG;
preconditioner DIC;
tolerance 1e-07;
relTol 0.05;
```
$p_r ghFinal$
```
solver PCG;
preconditioner DIC;
tolerance 1e-07;
relTol 0;
U
solver PBiCG;
preconditioner DILU;
tolerance 1e-06;
relTol 0;
PISO
momentumPredictor no;
nCorrectors 3;
nNonOrthogonalCorrectors 0;
```

```
nAlphaCorr 1;
nAlphaSubCycles 2;
cAlpha 1;
// ************************************************************ //
```

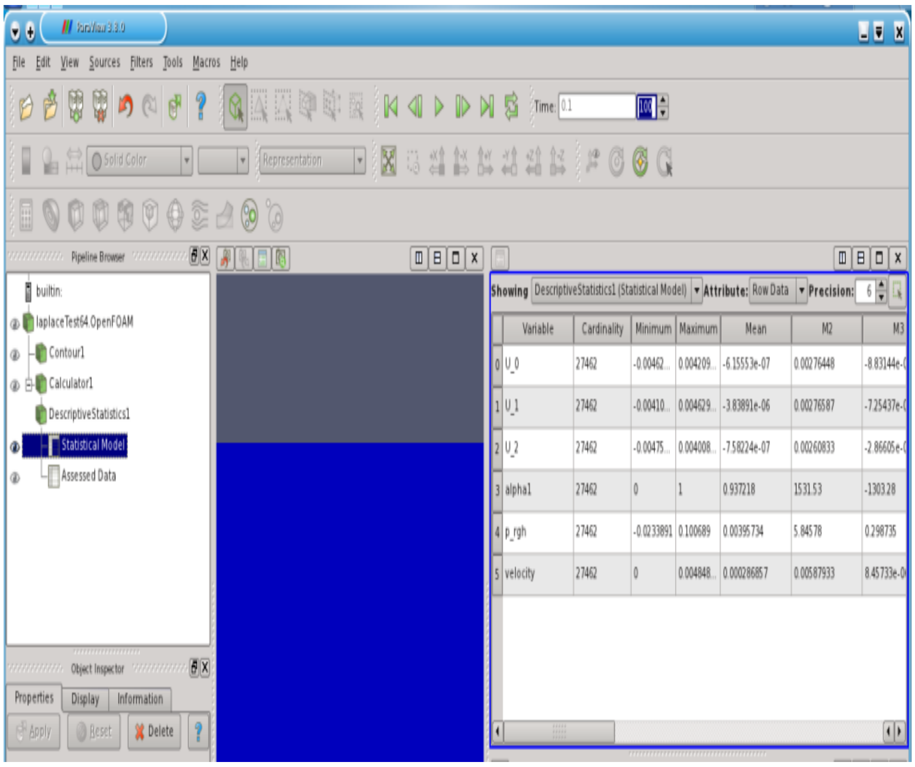
C.4 SetFields dictionary

The initial condition for the volume fraction of the two fluids need to be well defined. If we specify uniform internal fields then only one fluid will occupy the entire domain. In order to represents different fluids the utility setFieldsDict is used to divide the domain into two parts and give alpha1 value for each region. We can also defined different values into the selected portion of the domain by using setfieldDict.

```
FoamFile
version 2.0;
format ascii;
class dictionary;
location "system";
object setFieldsDict;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
defaultFieldValues
( volScalarFieldValue alpha1 1
volVectorFieldValue U (0 0 0)
volScalarFieldValue $p_rgh$ 10
);bench mark
regions
(sphereToCell
centre ( 0 0.01 0 );
radius 0.0033;
fieldValues
(
volScalarFieldValue alpha1 0
volScalarFieldValue $p_rgh$ 0
);
);
// ************************************************************ //
```

D Maximum velocity computation for 3D Laplace test, Using paraView.

For the computation of the maximum velocity, using paraView we compute the magnitude of the velocity U. This can be done by selecting tab *filters* and then selecting *calculator* in sub tab of *common*. In calculator select magnitude (mag button) and then from vectors select vector U for velocity. In this computation we apply descriptive statistics available in *filters* then *statistics* and *descriptive statistics*. Maximum velocity can be read at any step from the right hand window corresponding to velocity array. This setup is shown in the Figure 18 as a screen shot of paraView.



Showing: DescriptiveStatistics1 (Statistical Model) Attribute: Row Data Precision: 6

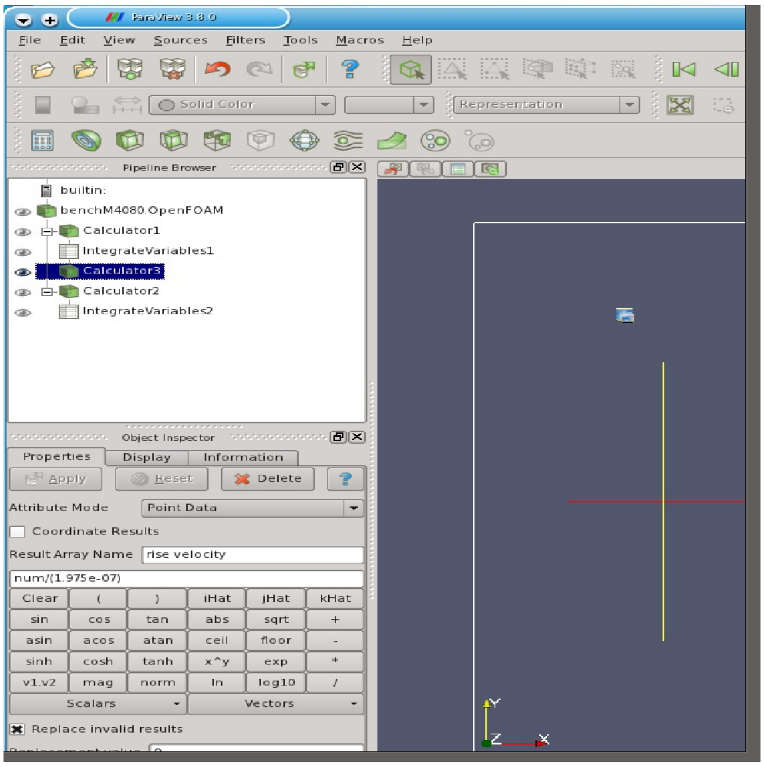
	Variable	Cardinality	Minimum	Maximum	Mean	M2	M3
0	U_0	27462	-0.00462...	0.004209...	-6.15553e-07	0.00276448	-8.83144e-
1	U_1	27462	-0.00410...	0.004629...	-3.83891e-06	0.00276587	-7.25437e-
2	U_2	27462	-0.00475...	0.004008...	-7.58224e-07	0.00260833	-2.86605e-
3	alpha1	27462	0	1	0.937218	1531.53	-1303.28
4	p_rgh	27462	-0.0233891	0.100689	0.00395734	5.84578	0.298735
5	velocity	27462	0	0.004848...	0.000286857	0.00587933	8.45733e-0

Figure 18: Setup for computation parasitic currents, using paraView.

E Rise velocity computation, Using paraView.

We have used the velocity of the center of gravity of the bubble to approximate the rise velocity of the bubble. We have used the formula mentioned in [8] at page 83. In OpenFOAM, it can be written as

$$RiseVelocity = \frac{\int(1-alpha1)U d\Omega}{\int(1-alpha1)d\Omega} \tag{1}$$



builtin:
benchM4080.OpenFOAM
Calculator1
IntegrateVariables1
Calculator3
Calculator2
IntegrateVariables2

Attribute Mode: Point Data
☐ Coordinate Results
Result Array Name: rise velocity
num/(1.975e-07)
☒ Replace invalid results

Figure 19: Setup for computation of the rise velocity, using paraView.

References

[1] `http://openfoam.com`.

[2] `http://openfoam.com/docs/user/mesh-conversion.php#x27-1600005.5`.

[3] `www.ansys.com`.

[4] `www.ParaView.org`.

[5] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. Proposal for quantitative benchmark computations of bubble dynamics. Technical report, Fakultät für Mathematik, TU Dortmund, August 2007. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 351.

[6] K. K. So, X. Y. Hu, and N. A. Adams. Anti-diffusion method for interface steepening in two-phase incompressible flow. *J. Comput. Physics*, 230(13):5155–5177, 2011.

[7] O. Ubbink. Numerical prediction of the two fluid systems with sharp interfaces. *PhD Thesis,University of London*, 1997.

[8] S.P. van der Pijl. Computation of bubbly flows with a mass-conserving level-set method. *PhD Thesis, TU Delft, The Netherlands*, 2005.