Multiphase Flow Solver: A Comparative study of level set re-distancing methods

Nadish Saini (UID: 200040846)
Department of Nuclear Engineering, North Carolina State University, Raleigh, NC 27606, USA

ABSTRACT

In the present work a multiphase flow solver has been developed based on the one-fluid incompressible flow formulation. The density variations across the interfaces are accounted for in the pressure Poisson solver. Further, three different level set re-distancing methods are analyzed and compared using standard advection test cases found in literature including vortex in a box and Zalesak's rotating disc. The flow solver is also analyzed using two different cases viz., bubble braking at free surface and dam break. The code developed for the project is C++ based and has been parallelized using OpenMP libraries. The results obtained from the code show that the solver is capable of successfully modelling two-phase flow with considerable accuracy with regards to mass conservation. The code can be further built upon to study flows in complex geometries and to analyze the underlying physics involved in multiphase flows.

Keywords

multiphase flow, level set, re-distancing, Navier-Stokes, pressure Poisson

CONTENTS

1 Introduction

In the present work a two-phase flow solver has been developed using the level set method to model the interface. In the level set method, introduced by Sussman and Fatemi (1999), the interface is defined as the zero level set of an implicit function, $\varphi = 0$. Here, φ represents the signed distance function of any point in the domain from an interface. The interface is convected by the equation:

$$\frac{\partial \varphi}{\partial t} + \underline{u}.\nabla\varphi = 0 \tag{1}$$

The above equation provides a good approximation to the distance field in the vicinity of the interface (Bolotnov et al., 2011), however, the solution in the regions relatively further from the interface is not expected to be correct from (1) since the convective flow field velocity varies throughout the domain. This requires that the signed distance field be corrected after the advection step since the surface tension forces are a function of curvature which in turn depends on the distance function. Therefore, for accurate coupling between the two phases it is imperative that the distance field is accurate. Furthermore, since the advection of the level set is governed by a hyperbolic equation, accurate movement of interface is subject to accraute gradients of distance function, atleast, near the interface. Failure to do so would result in apparent mass conservation issues in the involved fluids.

There are several re-distancing methods available in the open literature including the hyperbolic PDE based method (Sussman and Fatemi, 1999), the fast marching method (Sethian, 1996), the fast iterative method (Fu et al., 2011), the fast sweeping method (Zhao, 2005), direct re-distancing methods, etc., each having its own advantages and disadvantages. Three of these re-distancing methods are studied in this work with a focus on computation times and mass conservation issues associated with them. The methods are compared using two famous advection test cases. Finally, the two-phase flow solver is applied to two multiphase flow problems widely studied in the literature.

1.1 Objectives

The objective of the present work is two-fold:

1. To study and compare three different re-distancing methods for level set method. Applied to two 'popular' advection cases:
   (a) Vortex in a box
   (b) Zalesak's disc
2. To develop two-phase incompressible Navier-Stokes (variable density) solver Applied to the following test cases:
   (a) Bubble breakup at free surface
   (b) Dam break

2 Multiphase Flow

2.1 One-fluid incompressible flow solver

The governing equations for all the phases involved in a multiphase simulation are the Navier-Stokes equation, which in incompressible form are:

$$\text{Continuity :} \qquad \nabla\underline{u} = 0 \tag{2}$$

$$\text{Momentum :} \quad \frac{\partial \underline{u}}{\partial t} + \nabla(\underline{u}\underline{u}) = -\frac{1}{\rho}\nabla p + \nu\nabla^2\underline{u} + \frac{\underline{F}}{\rho} \tag{3}$$

The multiphase flow is solved as a one-fluid model with variable properties in the different phases involved. Since the properties are changing in the domain, we need to account for the density variations across the interface. Note that the Navier-Stokes are still solved as for an incompressible fluid i.e., pressure remains a non-thermodynamic variable and we do not use equation of state to close the system of equations.
In the present work finite volume formulation is used for discretisation of the Navier-Stokes equation. The momentum equation is decomposed in accordance with the pressure projection method. A predicted velocity is calculated using only the convective and diffusive terms:

$$\Delta V \frac{\underline{u}^* - \underline{u}^n}{\Delta t} = -\int_\Gamma \underline{u}(\underline{u}.\underline{n})\, d\Gamma + \int_\Omega \nu \nabla^2 \underline{u}\, d\Omega \tag{4}$$

The velocity is then projected onto the next time step using pressure and force terms:

$$\Delta V \frac{\underline{u}^{n+1} - \underline{u}^*}{\Delta t} = -\int_\Omega \frac{\nabla p^{n+1}}{\rho} d\Omega - \int_\Omega \frac{F_{surf} + F_{body}}{\rho} d\Omega \tag{5}$$

In (5) note that the surface and body forces are independent of the velocity terms. The pressure at time step $n+1$ is calculated by solving the pressure Poisson equation which is obtained by taking the gradient of (5):

$$\int_\Gamma \left(\frac{\nabla p}{\rho}\right).\underline{n}\, d\Gamma = \frac{1}{\Delta t}\int_\Omega \nabla \underline{u}^* - \int_\Omega \frac{\nabla(F_{surf} + F_{body})}{\rho} d\Omega \tag{6}$$

To maintain incompressibility of the fluid, the velocity at time step n+1 is enforced to be divergence free in accordance with the continuity equation i.e.,

$$\nabla \underline{u}^{n+1} = 0 \tag{7}$$

which is incorporated in equation (6).

2.2 Level Set method

As introduced earlier, the level set method models the interface as the zero level set of a smooth distance function φ. The 'smoothness' is artificially introduced to avoid discontinuities in the Navier-Stokes solver which otherwise may have lead to the solver braking down at the interface. The smeared out heavyside function is defined as (Osher and Fedkiw, 2006):

$$H(\varphi) = \begin{cases} 0 & \varphi < -\varepsilon \\ \frac{1}{2} + \frac{\varphi}{2\varepsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\varphi}{\varepsilon}\right) & -\varepsilon \le \varphi \le \varepsilon \\ 1 & \varepsilon < \varphi \end{cases} \tag{8}$$

where ε is the interface half thickness. The fluid properties are consequently smeared out as:

$$\rho(\varphi) = \rho_1 H(\varphi) + \rho_2(1 - H(\varphi)) \tag{9}$$

$$\mu(\varphi) = \mu_1 H(\varphi) + \mu_2(1 - H(\varphi)) \tag{10}$$

2.3 Continuous Surface Tension Force

Since the fluid properties are smeared out across the interface, a continuous description of the surface tension force can be adopted in equation (5). The continuum surface tension (CST) model was given by Brackbill et al. (1992):

$$F_{surf} = \sigma\varkappa(\varphi)\delta^{scaling}(\varphi)\nabla(\varphi) \tag{11}$$

where σ is the surface tension force coefficient; $\delta^{scaling} = 2H(\varphi)\delta(\varphi)$ (Yu et al., 2016), where;

$$\delta(\varphi) = \begin{cases} 0 & \varphi < -\varepsilon \\ \frac{1}{2\varepsilon} + \frac{1}{2\varepsilon}\cos\left(\frac{\pi\varphi}{\varepsilon}\right) & -\varepsilon \leq \varphi \leq \varepsilon \\ 0 & \varepsilon < \varphi \end{cases} \tag{12}$$

and $\varkappa$ is the curvature defined by:

$$\varkappa(\varphi) = -\nabla.\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right) \tag{13}$$

3 Level Set Re-distancing

3.1 Direct re-distancing

The distance function, in the simplest form, can be given by the following equation (2D):

$$d(x,y) = \underset{i=1..n}{min}\left(\sqrt{(x-x_i)^2 + (y-y_i)^2}\right) \tag{14}$$

where, (x_i, y_i) are the coordinates of any point i on the interface. Direct re-distancing is essentially a brute force method of evaluating the distance function for any point in the domain. The interface location can be interpolated from the value of level set values at the points lying on the smeared interface. Naturally the larger the number of seed interface points the more accurate will be the calculated distance field. Note that (14) gives an exact solution for the distance function. Thus numerical inaccuracies in the $d(\varphi)$ can only occur due to inaccuracies in interface interpolation and round-off errors. For the present simulations a linear interpolation is used for calculating interface seed points.

3.2 Hyperbolic PDE

The hyperbolic PDE based re-distancing was described by Sussman and Fatemi (1999). A pseudo time term, τ, is introduced such that the steady state solution of the following hyperbolic partial differential equation (PDE) gives the exact solution to the distance function.

$$\frac{\partial d}{\partial\tau} = sign(\varphi)(1 - |\nabla d|) \tag{15}$$

This may alternately be expressed as a transport equation (Bolotnov et al., 2011)

$$\frac{\partial d}{\partial\tau} + \underline{w}.\nabla d = sign(\varphi) \tag{16}$$

$$\underline{w} = sign(\varphi)\frac{\nabla d}{|\nabla d|} \tag{17}$$

Since the above equation is similar to the advection equation for level set, (1), the same solver can be used for re-distancing, resulting in a compact, re-usable code structure.
Note the the steady state solution to (15) will be independent of time term and therefore will be elliptic nature. Since a hyperbolic solver is used to arrive at an elliptic equation, un-physical results may be encountered in some critical flow scenarios such as bubble growth off of a wall, etc,. Equation (15) is usually not solved to steady state. This is because for successful coupling between the two phases we are only interested in accurate calculation of curvature for surface force terms. This can be usually achieved in 10-20 iterations for (15) given a suitable time step is chosen.

Sussman et al. (1998) introduced a further contraint to (15) to enforce distance function near the interface:

$$\frac{\partial d}{\partial t} = \text{sign}(\varphi)(1 - |\nabla d|) + \lambda_{ij} f(\varphi) \tag{18}$$

$$\lambda_{ij} = \frac{-\int_{\Omega_{ij}} H'(\varphi) L(\varphi d)}{\int_{\Omega_{ij}} H'(\varphi) f(\varphi)} \tag{19}$$

where:

$$f(\varphi) = H'(\varphi)|\nabla(\varphi)| \tag{20}$$

This constraint effectively prevents the distance field from being distorted in the vicinity of the interface during the re-distancing operation. The code in the present work includes this constraint in conjunction with hyperbolic PDE based re-distancing.

3.3 Fast Marching Method

The Fast Marching Method (FMM) was introduced by Sethian (1996). It is based on Dijkstra's method (Skiena, 1990) of computing minimum cost paths in a discrete domain. The fundamental idea behind the method is to construct an upwind solution starting from the interface such that the distance value is only determined by neighbour nodes that are closer to the interface. The algorithm essentially is a solver for Eikonal equation:

$$\|\nabla\varphi\| = 1 \tag{21}$$

which is the steady state result for equation (15). The domain is divided into 3 zones of accepted, trial and far-away nodes as described in figure 1. The upwind nodes are labelled as accepted and downwind nodes as far-away. The nodes currently being considered for updating distance function are labelled trial.

The discretisation of equation (21) in 2D for a cartesian domain yields:

$$\left(\max\left(D_{ij}^{-x}(\varphi), D_{ij}^{+x}(\varphi), 0\right)^2 + \max\left(D_{ij}^{-y}(\varphi), D_{ij}^{+y}(\varphi), 0\right)^2 \right)^{1/2} = 1 \tag{22}$$

where:

$$D_{ij}^{+x}(\varphi) = \varphi_{i+1,j} - \varphi_{i,j} \tag{23}$$

$$D_{ij}^{-x}(\varphi) = \varphi_{i,j} - \varphi_{i-1,j} \tag{24}$$

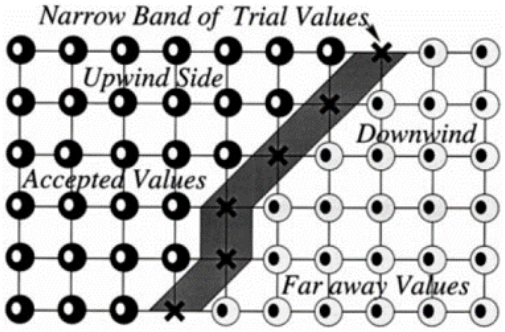


Fig. 1: Fast marching method algorithm

4 Results and Discussion

4.1 Code description

The code developed for the current project is written in C++ and parallelized using OpenMP libraries. The code has been designed to run on UNIX based systems.
The flow solver is based on a staggered grid representation with pressure and level set values stored at the physical cell centers and the velocities stored at cell edges. Owing to the staggered grid formulation the code is currently designed only for cartesian meshes.
Time discretisation for all differential equations is forward Euler explicit. The spatial discretisation for convective terms in the Navier-Stokes equation is based on linear interpolation for calculating fluxes at the cell edges. For diffusive terms the discretisation is second order central difference. For the level set convective terms the spatial discretisation is based on second order essentially non-oscillatory (ENO) schemes (Prosperetti and Tryggvason, 2009).
Point-Jacobi iterative method is used for the pressure Poisson solver with a tolerance of 10^{-2}.

4.2 Advection Cases

In order to fairly compare different re-distancing methods the cases discussed in this section do not involve the solution of Navier-Stokes equation. The velocity fields are artificially introduced and only the level set advection equation and re-distance equations are solved. Since the advection equation is common to all cases, the differences in re-distancing methods are thus brought forward.

4.2.1 Vortex in a box

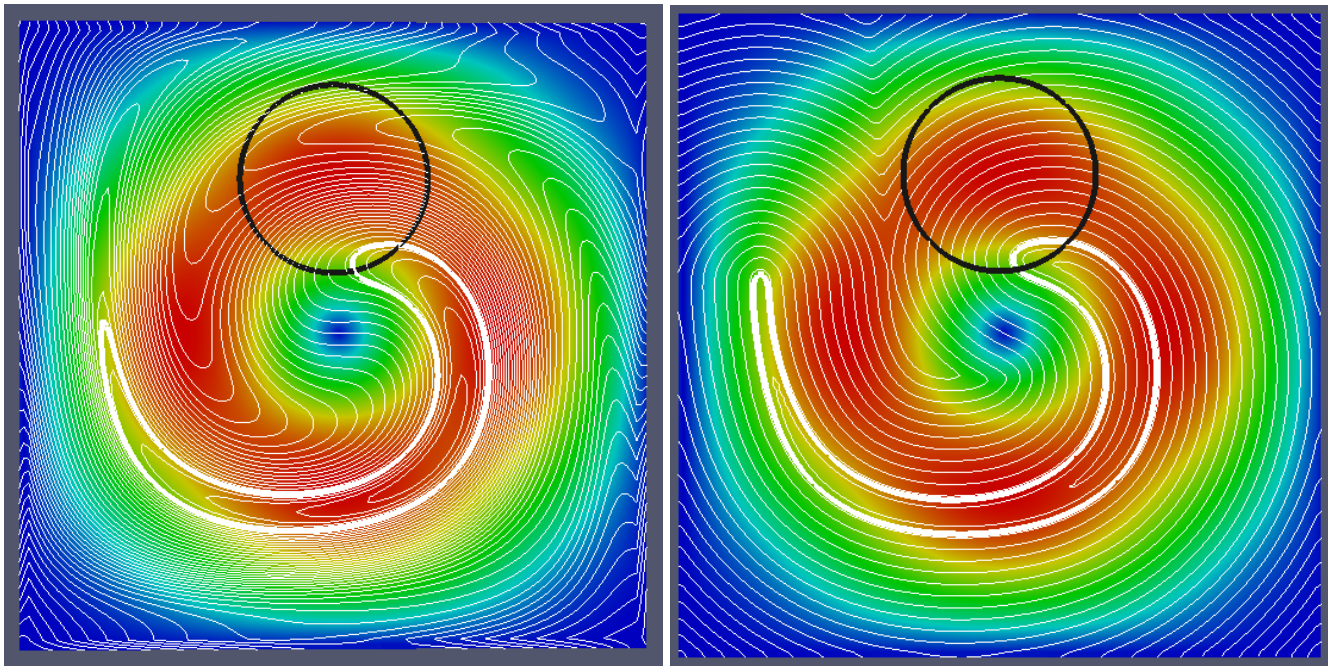
The artificially specified velocity field is given by the following equations:

$$u = -\sin^2\pi x \sin(2\pi y) \tag{25}$$
$$v = -\sin^2\pi y \sin(2\pi x) \tag{26}$$

The initial solution is given by:

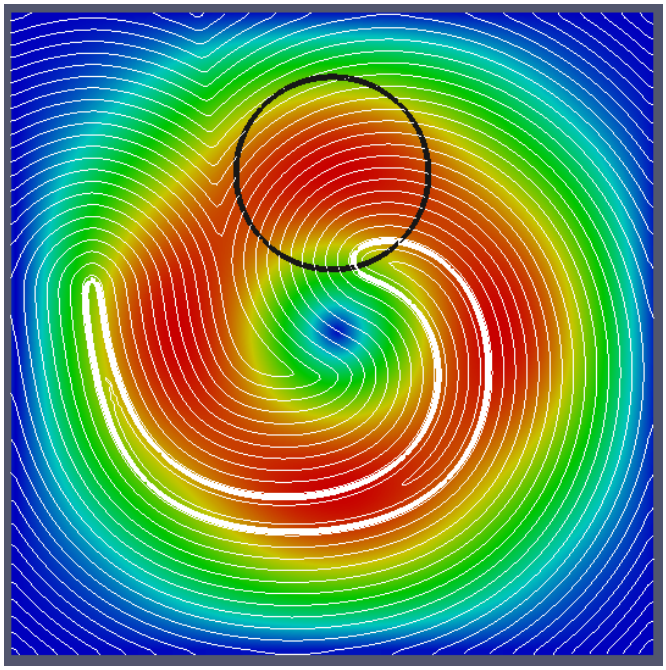
$$(x-0.5)^2 + (y-0.75)^2 = 0.0225 \tag{27}$$

The velocity field was reversed at t=T=1 such that the solution should ideally coincide with original solution at t=T=2. The grid for the current simulations was uniform wuth 128 X 128 cells. The CFL for the entire simulation was maintained at 0.01.

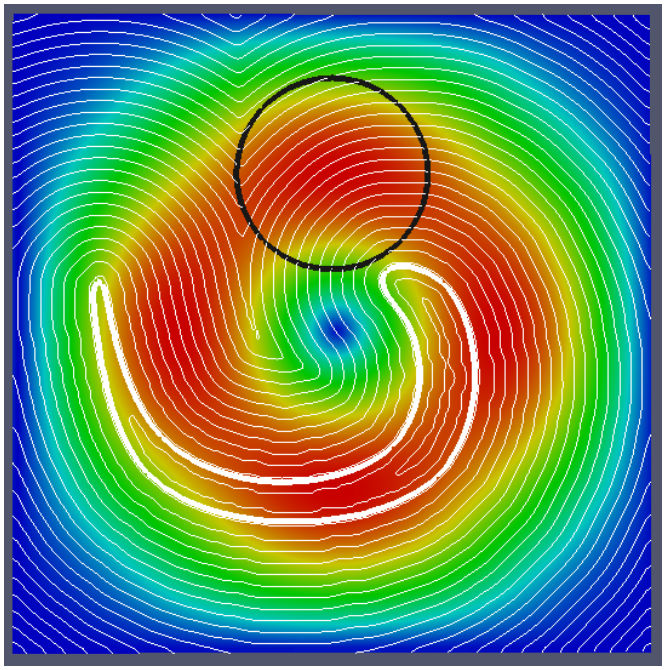


(a) Advection only

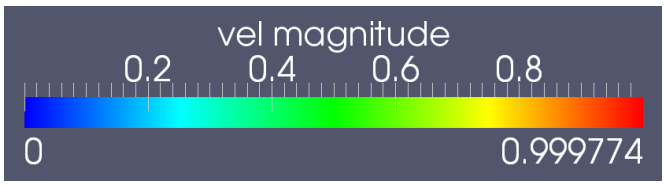
(b) Hyperbolic PDE



(c) Fast March

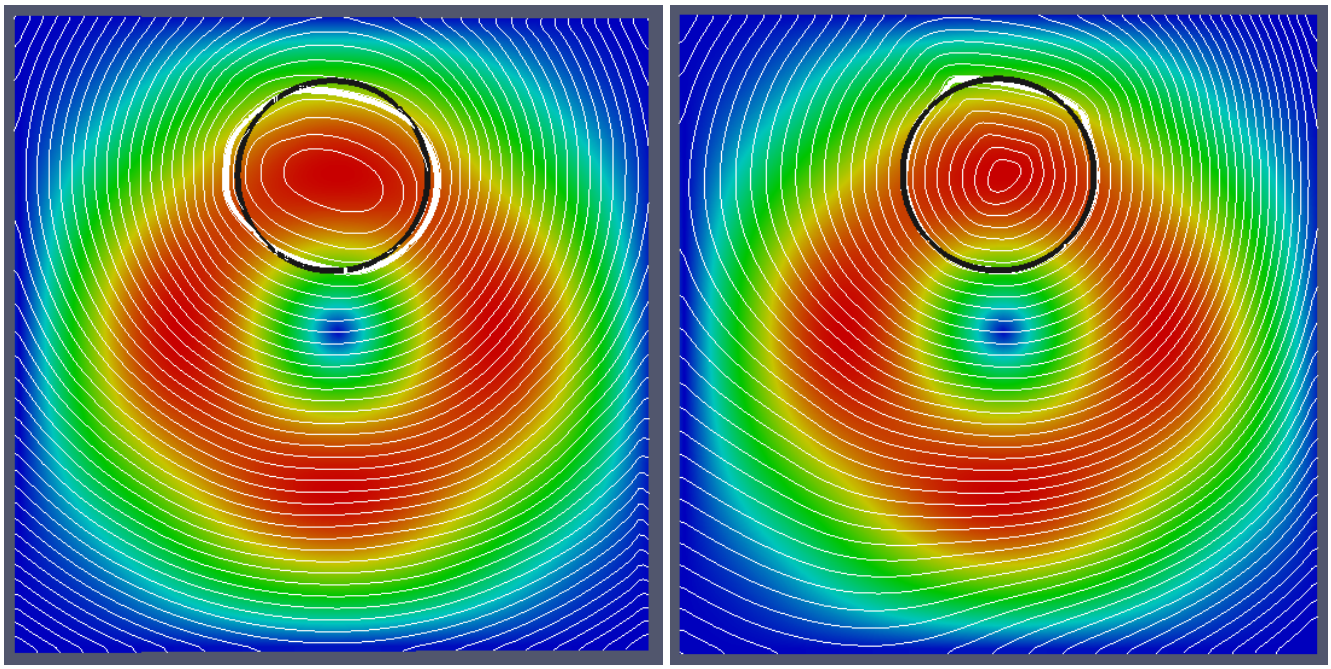


(d) Direct re-distancing



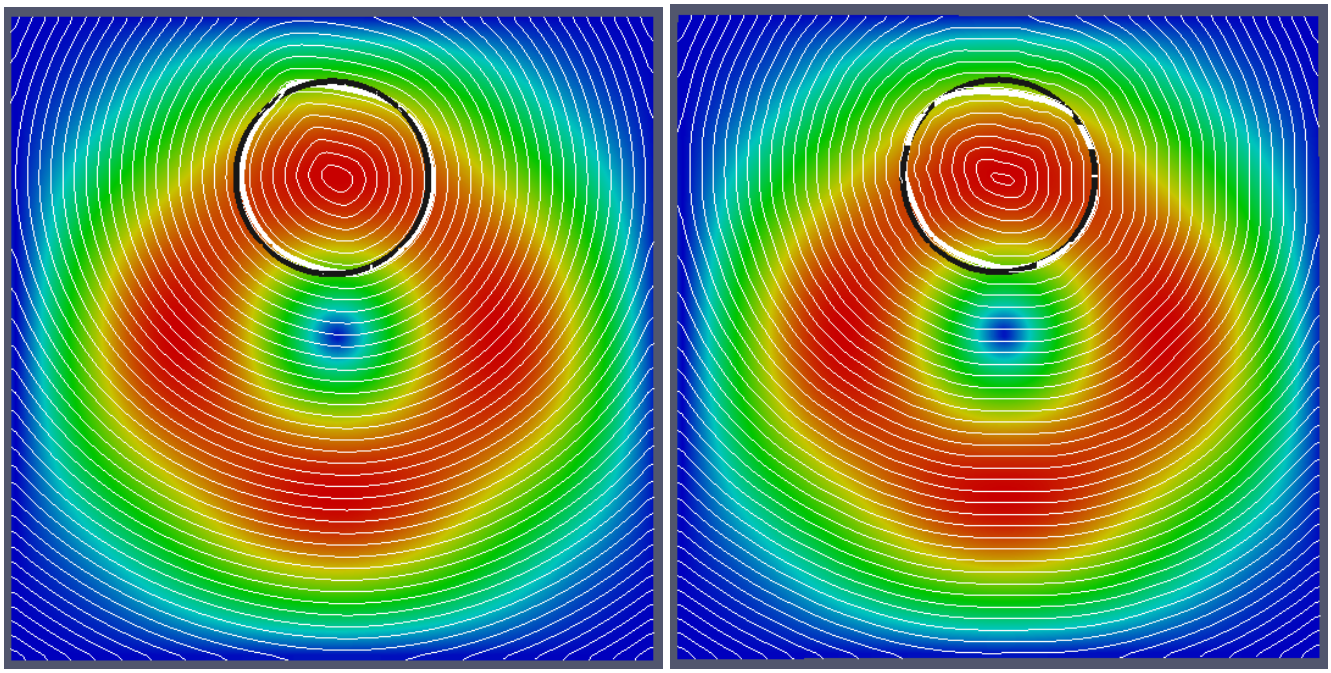
(e) Legend

Fig. 2: Results from different re-distancing method at t=T=1



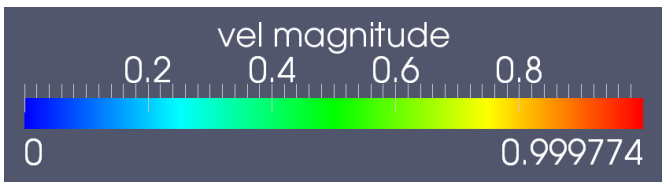
(a) Advection only

(b) Hyperbolic PDE



(c) Fast March

(d) Direct re-distancing



(e) Legend

Fig. 3: Results from different re-distancing method at t=T=2 as compared to exact solution (in black)

Figures 2 and 3 show the results obtained from different re-distancing methods and without any re-distancing for t=1 and t=2 respectively. It is evident from figure 3a that the discretisation of advection equation itself is significantly diffusive. Thus mass errors are expected to be large as evident in figure 4. It is seen that the hyperbolic PDE based re-distancing is closest to the advection case in terms of mass conservation followed by fast march method and direct re-distancing. Errors in direct re-distancing method can be attributed to poor interpolation of interface from level set field and smaller number of seed points. The mass errors in advection case must be minimum in order to properly compare the re-distancing methods. Thus higher order discretisation schemes are required for advection terms.
In terms of computation times, the advection case took 2351 s; fast march method took 7450 s; hyperbolic PDE took 14711 s and direct re-distancing was the most expensive at 29765 s. All times reported here are in clock seconds.

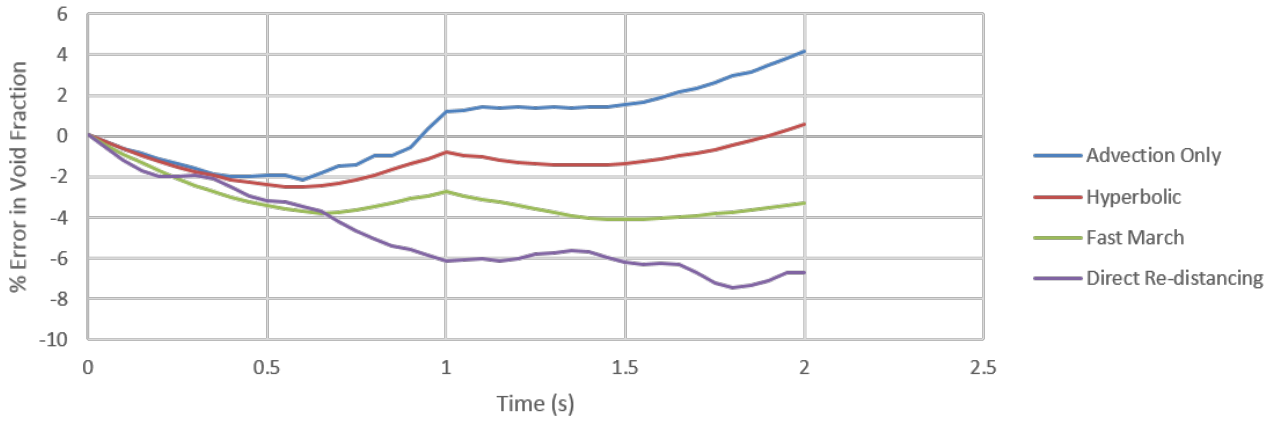


Fig. 4: Percentage error in void fraction for different cases

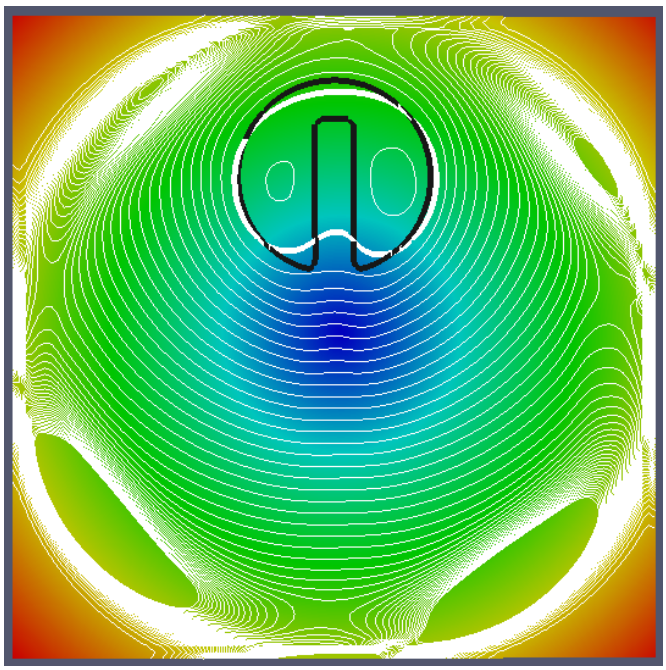
4.2.2 Zalesak's Rotating Disc

In this problem a slotted disk with a radius of 15 and a slot width of 6 is located at 50,75 coordinates. The domain itself is 100 X 100 with 100 X 100 mesh cells. The prescribed velocity field is:

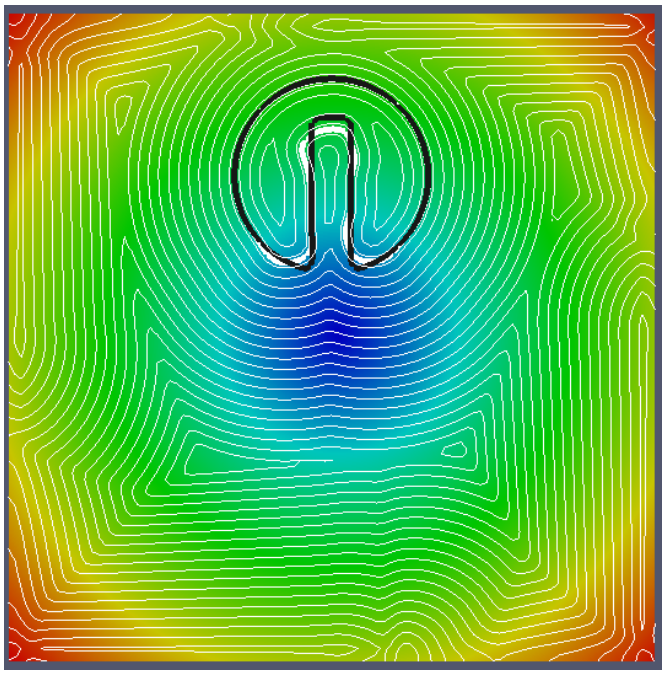
$$u = \frac{\pi(50 - y)}{314} \tag{28}$$

$$v = \frac{\pi(x - 50)}{314} \tag{29}$$

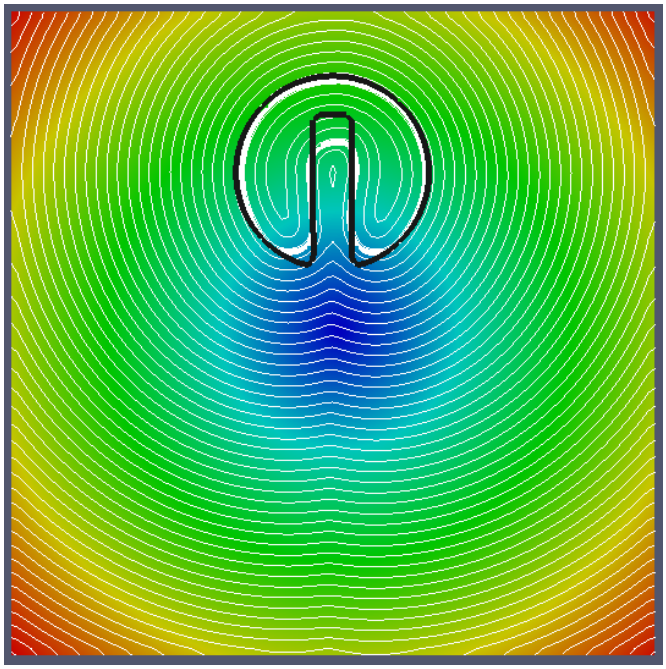
Time step for the simulation is 0.25 Δx which amounts to a CFL of 0.5.



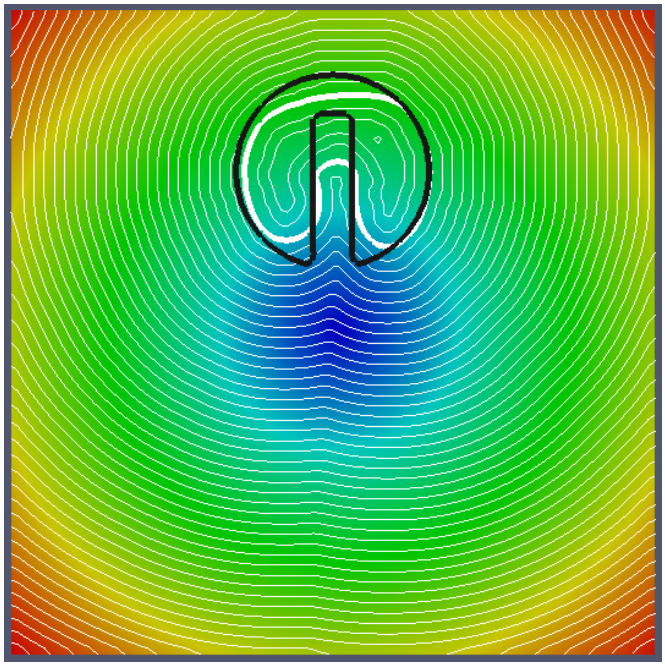
(a) Advection only



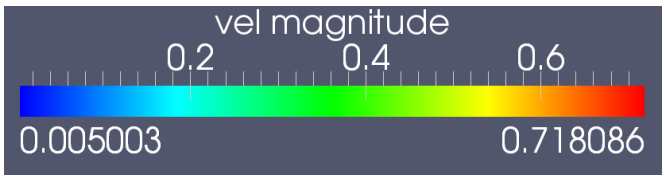
(b) Hyperbolic PDE



(c) Fast March



(d) Direct re-distancing



(e) Legend

Fig. 5: Results with different methods as compared to exact solution (in black)

Again, it is clearly evident from figure 5a that the advection schemes used are very diffusive. Since re-distancing operations corrects the gradients around the interface, the solution obtained with re-distancing is much better as compared to the advection case. Figure 6 gives the evolution of mass fraction for different cases. The computational times for advection, hyperbolic PDE, fast march and direct re-distancing are 224 s, 1335 s, 3612 s and 5248 s respectively.

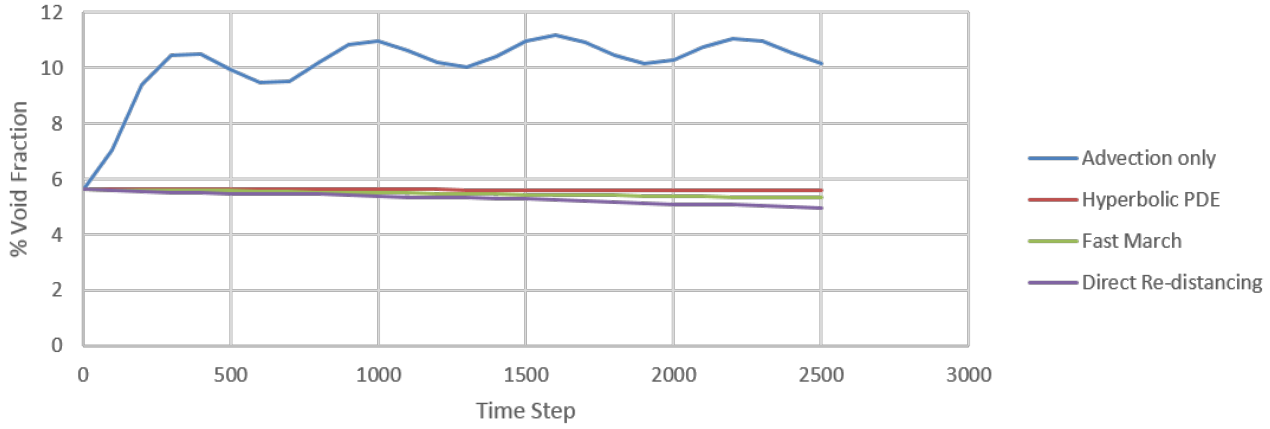


Fig. 6: Percentage void fraction for different cases

4.3 Two-phase flow cases

The test cases presented in this section demonstrate the capabilities of the code developed for modelling two-phase flow.

4.3.1 Bubble Breakup on free surface

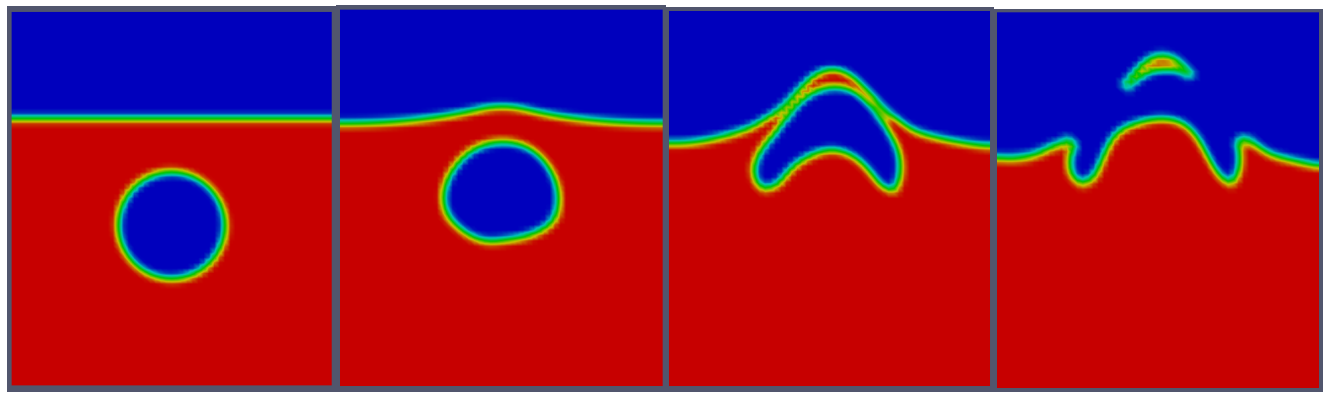
Bubble breaking at a free surface is a widely studied test case to test the veracity of multiphase flow codes. The simulation presented here follows the parameters as used by Yu et al. (2016):

$$\rho_f = 1000 \text{ kg/m}^3 \tag{30}$$
$$\rho_g = 500 \text{ kg/m}^3 \tag{31}$$
$$\mu_f = 1 \text{ Pas} \tag{32}$$
$$\mu_g = 0.5 \text{ Pas} \tag{33}$$
$$\sigma = 1 \text{ N/m} \tag{34}$$
$$g = 9.81 \text{m/s}^2 \tag{35}$$

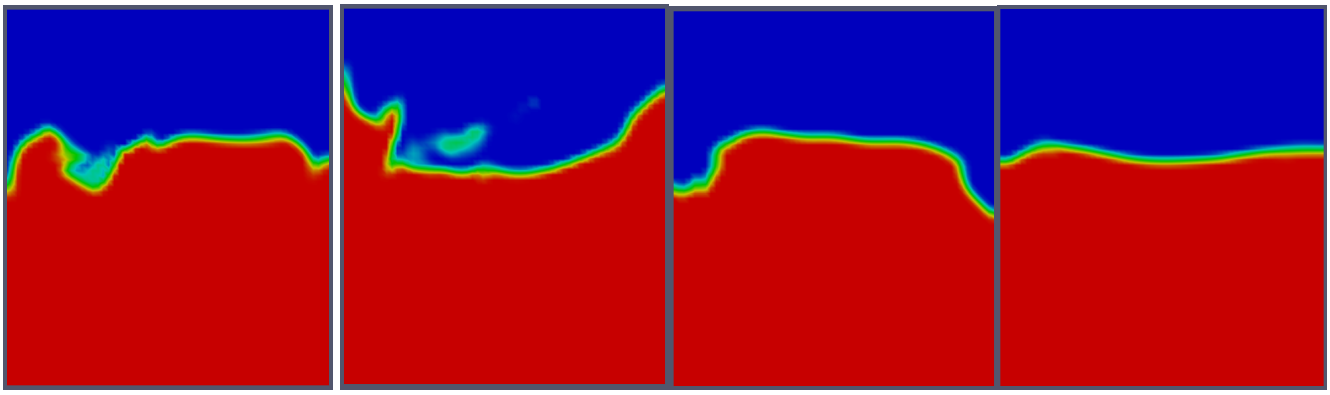
A bubble submerged in fluid is initialized at the position (1.5,1.5). The diameter of the bubble is 1 and the initial height of the fluid is 2.5. An interface thickness of 1.2ε is chosen for the simulation. The mesh has 60 X 70 isotropic elements. The pressure and velocities are initialized to 0. The flow field evolves only under the effect of gravitational forces. For re-distancing of the level set field hyperbolic PDE based method is employed.

Figure 7 shows the density/viscosity contour plots obtained at different time steps for the present case. Figure 8 shows the pressure contour and level set contours for the corresponding time steps. As expected after the simulation is run long enough the mass inside the bubble escapes the heavier fluid and the

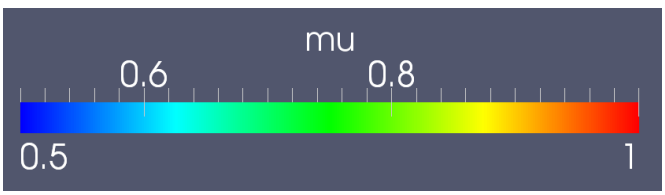
heavier fluid settles down to a horizontal level. The initial void fraction was 37.85% and the final void fraction at t=20s was 38.61%. Thus the mass was conserved pretty good by the solver.



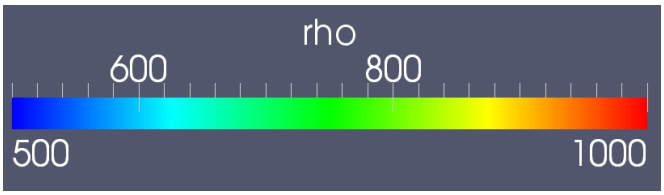
(a) t = 0s (b) t = 1s (c) t = 2s (d) t = 3s



(e) t = 6s (f) t = 8s (g) t = 10s (h) t = 20s

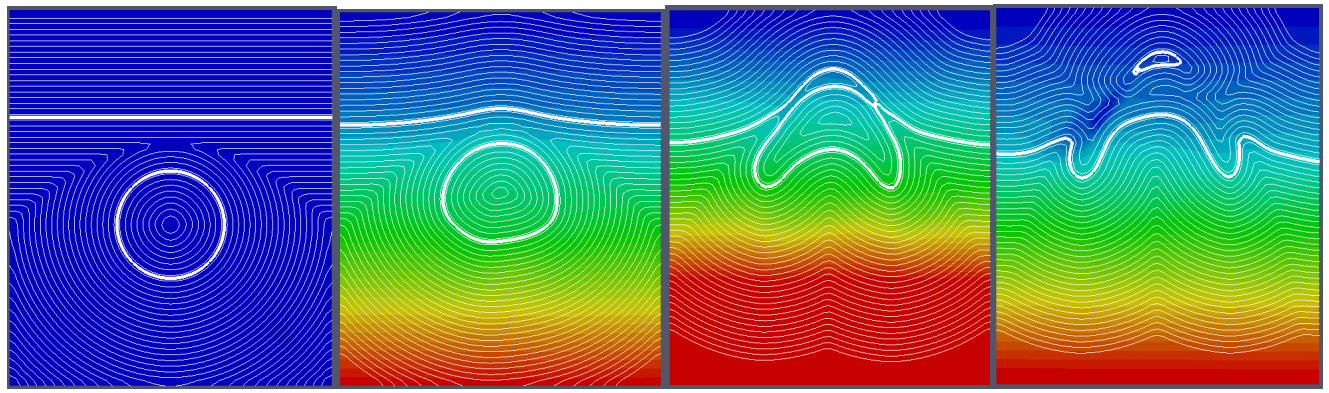


(i) Viscosity scale

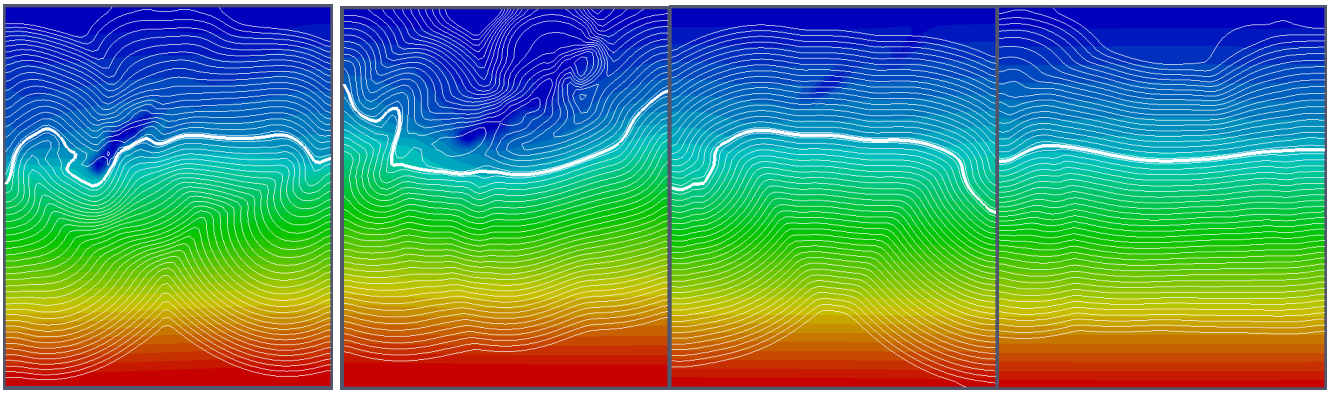


(j) Density scale

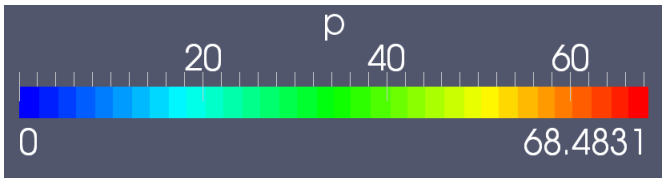
Fig. 7: Results for bubble break simulation at different time steps



(a) t = 0s (b) t = 1s (c) t = 2s (d) t = 3s



(e) t = 6s (f) t = 8s (g) t = 10s (h) t = 20s



(i) Viscosity scale

Fig. 8: Pressure contour plot (colored) and level set contours (white lines) at different time steps.

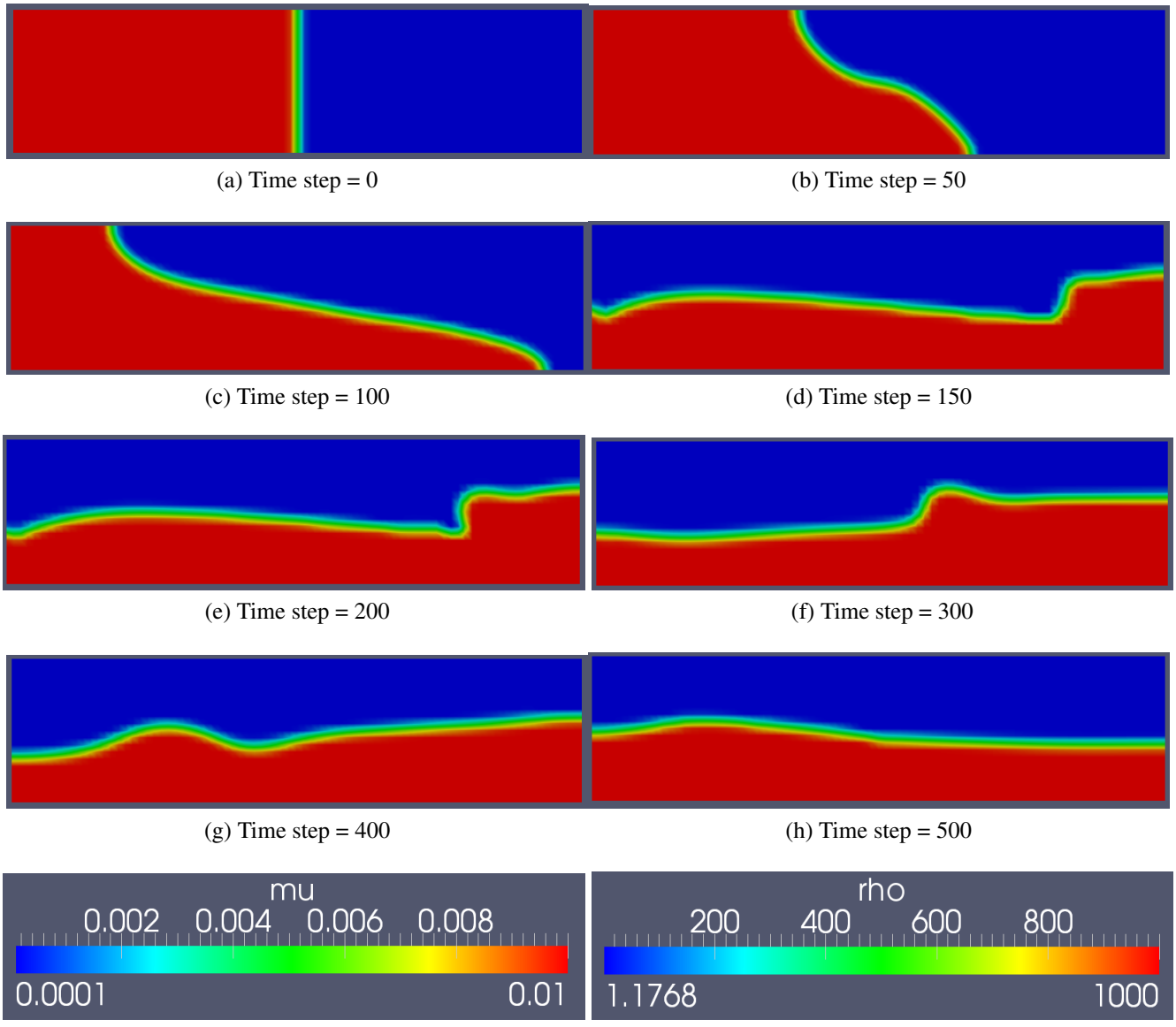
4.3.2 Dam break problem

For the dam break problem the domain was initialized such that half of the fluid is water and half of it air. All properties chosen correspond to real water/air properties at 25°C:

$$\rho_f = 1000\ \text{kg/m}^3 \tag{36}$$
$$\rho_g = 1.1768\ \text{kg/m}^3 \tag{37}$$
$$\mu_f = 10^{-3}\ \text{Pas} \tag{38}$$
$$\mu_g = 10^{-5}\ \text{Pas} \tag{39}$$
$$\sigma = 0.07\ \text{N/m} \tag{40}$$
$$g = 9.81\text{m/s}^2 \tag{41}$$

Pressure and velocities are initialized to 0. The pressure at the top wall was fixed to 0. Slip velocity was assigned at all walls and a zero gradient was specified for pressure at the remaining walls. For initial

iterations the interface was not allowed to advect so that the pressure may develop in the domain, which ultimately drives the flow. For this particular simulation the re-distancing solver was fast marching method. Figure 9 and 10 show the result plots for this case.



(a) Time step = 0

(b) Time step = 50

(c) Time step = 100

(d) Time step = 150

(e) Time step = 200

(f) Time step = 300

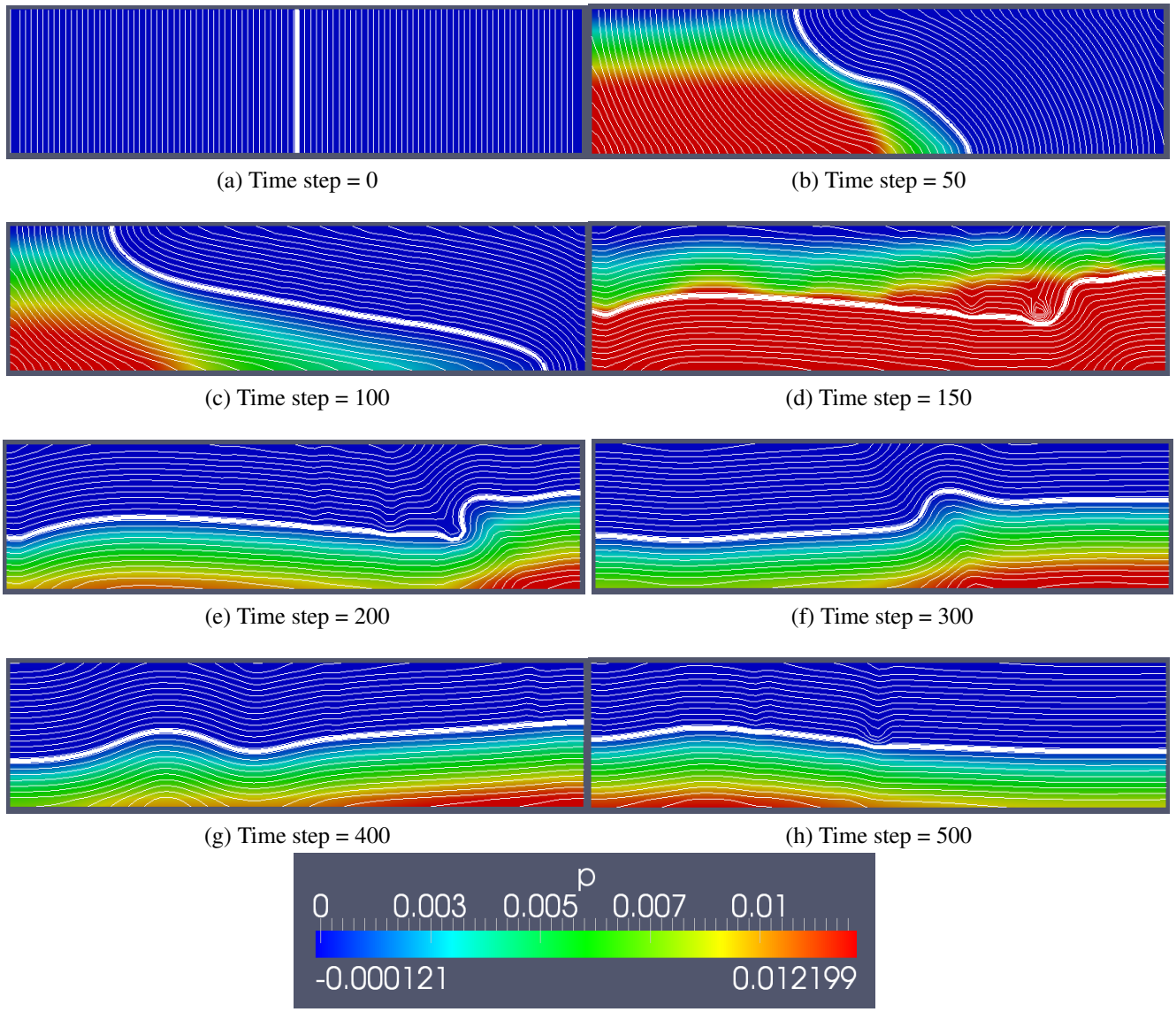
(g) Time step = 400

(h) Time step = 500

(i) Viscosity scale

(j) Density scale

Fig. 9: Results for dam break simulation at different time steps



(a) Time step = 0

(b) Time step = 50

(c) Time step = 100

(d) Time step = 150

(e) Time step = 200

(f) Time step = 300

(g) Time step = 400

(h) Time step = 500

(i) Viscosity scale

Fig. 10: Pressure contour plot (colored) and level set contours (white lines) at different time steps.

5 Conclusion and Future Work

From the advection cases it can be concluded that the discretisation schemes for level set convective terms is highly diffusive. Higher order schemes are suggested for future work on the code. Hyperbolic PDE and fast march re-distancing method do a good job at mass conservation and retain the exact solution fairly good. Errors observed in direct re-distancing are attributed to poor interpolation of the interface from level set field. Better interpolation methods must be approached for future analysis.

For the incompressible flow cases the developed code produced good results in terms of void fraction conservation. For bath cases shown the simulation proceeded smoothly till a large number of time steps establishing the robustness of the code. For future work the code can be extended to 2D/3D complicated geometries, high order advection and diffusion terms can be included and re-distancing methods can be further enhanced. To simulate more complex cases the code must also be designed for parallel processing using MPI or GPU libraries.

References

Bolotnov, I. A., Jansen, K. E., Drew, D. A., Oberai, A. A., Lahey, R. T., and Podowski, M. Z. (2011). Detached direct numerical simulations of turbulent two-phase bubbly channel flow. *International Journal of Multiphase Flow*, 37(6):647–659.

Brackbill, J., Kothe, D. B., and Zemach, C. (1992). A continuum method for modeling surface tension. *Journal of computational physics*, 100(2):335–354.

Fu, Z., Jeong, W.-K., Pan, Y., Kirby, R. M., and Whitaker, R. T. (2011). A fast iterative method for solving the eikonal equation on triangulated surfaces. *SIAM Journal on Scientific Computing*, 33(5):2468–2488.

Osher, S. and Fedkiw, R. (2006). *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media.

Prosperetti, A. and Tryggvason, G. (2009). *Computational methods for multiphase flow*. Cambridge university press.

Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595.

Skiena, S. (1990). Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227.

Sussman, M. and Fatemi, E. (1999). An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on scientific computing*, 20(4):1165–1191.

Sussman, M., Fatemi, E., Smereka, P., and Osher, S. (1998). An improved level set method for incompressible two-phase flows. *Computers & Fluids*, 27(5):663–680.

Yu, C., Ye, Z., Sheu, T. W., Lin, Y., and Zhao, X. (2016). An improved interface preserving level set method for simulating three dimensional rising bubble. *International Journal of Heat and Mass Transfer*, 103:753–772.

Zhao, H. (2005). A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627.