COBEM2015

ABCM Associação Brasileira de Engenharia e Ciências Mecânicas

23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

CURVATURE ESTIMATION ON FLUID INTERFACES BY POINT-CLOUD SAMPLING

Bruno de Barros Mendes Kassar
Angela Ourivio Nieckele
PUC-Rio: Departamento de Engenharia Mecânica , Rua Marquês de São Vicente 225, Gávea, Rio de Janeiro - RJ - Brazil
brunokassar@gmail.com, nieckele@puc-rio.br

Abstract. *Volume of Fluid is a widely employed method in the simulation of multiphase flows. However, one of its drawbacks is its lack of accuracy on interfacial curvature estimates. Many methods of curvature estimation have been presented in the literature during the years and this work sheds a new light on the problem. The interfaces are locally described by a set of points and normals and the curvature is extracted from this set. The methodology was developed in OpenFOAM® . An increase in accuracy of the normals and curvatures is observed, what improves the computation of interfacial tension.*

Keywords: *Volume-of-fluid , Curvature, Point-cloud, OpenFOAM®*

1. INTRODUCTION

Multiphase flows are often found in nature and industry. Ocean waves, rain, clouds, sandstorms are some examples of multiphase flows in nature, just to name a few. In industry, a wide variety of examples can be found, such as fluidized beds, refrigeration, pipeline transport of oil and gas, flow of slurries or pulverized particles, pressurized water nuclear reactors, boiling water and so on (Ishii and Hibiki, 2011).

The main difficulty in the prediction of multiphase flows is to keep track of the fluid phases along the domain. Depending on several parameters like geometry, fluid properties and flow regime, the phases can be arranged in different configurations. Multiphase flows can be categorized by the geometric arrangements of its phases in *disperse flows* and *separate flows*. Disperse flows consist of finite particles, drops or bubbles spread in a continuous phase. On the other hand, separate flows are identified by different continuous phases separated by interfaces (Brennen, 2005). Combination of these two configurations can also exist.

This work focuses on two-phase separate flows, in which the interfaces are well resolved by mesh resolution. Volume of Fluid Method – VOF – is employed to keep track of the phases throughout the domain. It basically consists on keeping track of a scalar field α that represents the volume fraction of one of the phases. At each time step this volume fraction field is advected along with the velocity field.

At the interface, the pressure jump due to the surface tension must be imposed. In VOF, surface tension is most commonly taken into account by using the CSF method – *Continuous Surface Force Method* – proposed by Brackbill *et al.* (1992). In this method, the curvature of the interface is computed by taking the divergence of the normal field in the vicinities of the interface. The accuracy on the prediction of a two-phase flow can depend strongly on the surface tension determination, and for this reason several works found in the literature have focused on this issue.

Accurate computation of surface tension requires correct estimation of surface normals and curvatures. While VOF has good mass conservation properties and produces fairly sharp interfaces, what is well desired, it is not accurate at computing surface normals and curvatures directly from its volume fraction field α. It is mainly due to abrupt changes in α at regions along the interface.

A group of researchers tackle the problem of curvature accuracy using the fact that the Level-Set field γ (signed distance from interface) is smooth throughout the whole domain and, thus, accurate for normal and curvature computation. Thus, Sussman and Puckett (2000) coupled the Level-Set method with VOF in what they called CLSVOF –*Coupled Level Set/Volume-of-Fluid method* – where the γ values at each time step are derived from γ and α values from the previous time step. The curvatures and normals are obtained directly via finite differences on the γ field. The normals are computed by $\hat{n} = \nabla\gamma/|\nabla\gamma|$ and the curvatures by taking the divergence of this normal field, $\kappa = \nabla \cdot \hat{n}$. In this way, CLSVOF benefits from VOF's ability to conserve mass and Level Set's accuracy on normal and curvature computation.

Renardy and Renardy (2002) addressed the problem of accurate curvature computation with VOF by representing the interface as piecewise quadratic equations fitted on the VOF field. The algorithm is named PROST – *Parabolic Reconstruction Of Surface Tension*. It consists on fitting quadratic equations for every cell crossed by the interface, taking

the neighboring cells into consideration. It is relevant to notice that this work uses orthogonal structured meshes. In 3D, the interface is described by piecewise paraboloids for each interface cell and the curvature κ is computed from the paraboloid equations.

In order to obtain second-order accurate curvatures, Sussman (2003) computed a curvature based on the volume fraction field, instead of using γ, that would not give second-order accuracy. This method is based on reconstructing a "height" function directly from the volume fractions, using a stencil of neighboring cells as described in Helmsen and Colella (1997). In Sussman (2003), the orientation of the interface is determined by a normal computed based on the level set function as $\hat{n} = \nabla\gamma/|\nabla\gamma|$. Still, it is assumed, without loss or generality, that the surfaces are more horizontal than vertical (for vertical interfaces, an analogous procedure is taken). For each cell (i, j) containing the interface, a 3×7 block of neighboring cells is used to compute the "height" as a result of the summation over the α field along the three vertical sets of 7 cells, for $i-1$,i and $i+1$. This gives the "height" value at $(i-1, j)$, (i, j) and $(i+1, j)$. With these values, the first and second derivatives in the horizontal direction are also determined for cell (i, j). The curvature arises from this second-order accurate height function.

Cummins *et al.* (2005) presented a new method to compute curvature in a VOF framework based on what they called RDF – *Reconstructed Distance Function*. Inspired in the Level Set method, they proposed to define a distance function from the reconstructed interface (reconstructed by *Piecewise Linear Interface Calculation* – PLIC – proposed by Rider and Kothe (1998)) to be used for curvature computation. This RDF function Φ is computed for all interface cells and for cells around it up to a normal distance of $3\Delta x$ from the interface, where Δx is the cell size. The value Φ_{ij} of the RDF for each considered cell (i, j) is a weighted average of the Euclidean distances from the center of the cell (i, j) to the many neighboring PLIC segments that represent the interface. This work compares the accuracy and robustness of curvature estimates based on the proposed RDF with two other methods: a convolved VOF (CV) function (Williams, 2000) and the already discussed height function (HF) (Sussman and Puckett, 2000; Helmsen and Colella, 1997). The convolved VOF smoothes the α field by applying a kernel to it and the curvature is computed based on the convolved α. Results obtained using height function were superior compared to both convolved VOF and RDF. However, when interfacial length scale is not well resolved by the grid, CV and RDF methods exhibit greater robustness over the HF approach.

Sussman and Ohta (2009) continued the research on curvature estimation based on a height function. They derived the height function from the Level-Set field instead of from the VOF field, as in previous work (Sussman, 2003). They found it not only easier to implement, but also more accurate. The authors also presented an important consideration regarding the curvature discretization for surface tension purposes. They have stated that the curvature should represent an approximation to the curvature *on the interface*, like the height function approximation does (height function reconstructed from either γ or α). The curvature computed directly from the level set field, i.e. , from the formula $\kappa = \nabla \cdot (\nabla\gamma/|\nabla\gamma|)$ gives an estimate at the cell center instead of on the surface itself.

This work brings a new approach to the curvature estimate in order to increase accuracy. It represents the interfaces as surfaces sampled by point clouds with normals and extracts the curvatures from this cloud. This approach showed an increase in accuracy on the estimations of curvature compared to standard OpenFOAM® computations.

2. GOVERNING EQUATIONS

Isothermal fluid flows are governed by the equations of Conservation of Mass and Linear Momentum. The conservation of mass, for incompressible fluid, considering no mass transfer, is given by

$$\nabla \cdot \overrightarrow{V} = 0 \qquad \text{; where } \overrightarrow{V} \text{ is the velocity vector} \tag{1}$$

For multiphase flows, to account for the pressure jump due to surface traction, the surface tension σ is added to the linear momentum equation. As this effect is concentrated at the interface, it is represented by a δ function. Considering the surface tension, the linear momentum equation for a Newtonian fluid can be written as

$$\frac{\partial(\rho\overrightarrow{V})}{\partial t} + \overrightarrow{\nabla} \cdot (\rho\overrightarrow{V}\overrightarrow{V}) = -\overrightarrow{\nabla}p + \overrightarrow{f} + \overrightarrow{\nabla} \cdot [\mu(\nabla\overrightarrow{V} + \nabla^T\overrightarrow{V})] + \sigma\,\kappa\,\delta(n)\,\hat{n} \tag{2}$$

where κ is the curvature, $\hat{n}$ is the unit normal vector of the interface and n is a normal coordinate, with $n = 0$ at the interface. $\overrightarrow{f}$ is a body force, such as gravity ($\overrightarrow{f} = \rho\overrightarrow{g}$), p is the pressure and ρ and μ are respectively the fluid's density and dynamic viscosity defined as $\rho = \rho_1\alpha + \rho_2(1-\alpha)$ and $\mu = \mu_1\alpha + \mu_2(1-\alpha)$ where the subscripts 1 and 2 refer to phases 1 and 2 respectively.

3. SURFACE CURVATURES

Curvature measures the bending of a surface or a curve. In curves, the curvature is defined by the modulus of the derivative of the tangent vector $\overrightarrow{t}$ with respect to the arc length s, such as $\kappa_{\text{curve}}(s) = \left|\frac{d\overrightarrow{t}(s)}{ds}\right|$ .Thus, the curvature is uniquely defined at each point of a curve. It also does not depend on the parametrization direction. On the other hand, in

surfaces, at a given position, there are infinite directions that the tangent vector can point to. In this way, any point has infinite curvatures associated to it depending on the direction taken into consideration. Two of these directions, however, indicate the maximum and minimum bending of the surface. These are called *principal directions* and the curvatures associated to them are the *principal curvatures*, usually represented by κ_1 and κ_2 (Kreyszig, 1991). In this work, we are interested in the *mean curvature* H of the surface that separates two fluid components, the interface surface. It is given by $H = \frac{\kappa_1+\kappa_2}{2}$.

The momentum equation for multiphase flows is similar to the one for single phase, but augmented with the interfacial tension term, that depends on the curvature κ, as displayed in Eq. 2. For both 2D and 3D flows, the curvature κ is twice the mean curvature H, so $\kappa = \kappa_1 + \kappa_2$. Notice that for 2D flows, $\kappa_2 = 0$.

4. INTERFACE REPRESENTATION

The cell-centered α value measures the volume fraction of fluid 1 in a given cell. It ranges from 0 to 1, where 1 means the cell is fully occupied by fluid 1 and 0 means it is fully occupied by fluid 2. In this way, one may consider the interface as the isosurface for which $\alpha = 0.5$. Is is a well know issue that the curvatures and normals computed by the α field suffer from loss of accuracy once the α field varies abruptly across the interface. In this way, this work proposes another approach to extract the interfacial curvature. It samples the interface surface by a set of points and normals and the local curvature is computed based on this point cloud.

4.1 Sampling Points Estimation

The idea is to cut the mesh edges by the isosurface represented by $\alpha(\overrightarrow{x}, t) = 0.5$. This is the same as to find points on the mesh that satisfy $\alpha = 0.5$.

As the α field is stored at cell centers, it is first projected onto the mesh vertexes as follows. Let's call α projected on the vertex points by α_p. The value of α_{P_i} at point P_i is computed by the inverse distance interpolation of the N_i cells neighboring vertex P_i.

$$\alpha_{P_i} = \frac{\sum_{j=1}^{N_i} w_{ij}\alpha_j}{\sum_{j=1}^{N_i} w_{ij}} \qquad ; \qquad w_{ij} = \frac{1}{|\overrightarrow{x_{P_i}} - \overrightarrow{x_{C_j}}|} \tag{3}$$

where w_{ij} is the weighting factor of the j^{th} adjacent cell to vertex P_i. The vectors $\overrightarrow{x_{P_i}}$ and $\overrightarrow{x_{C_j}}$ are the position vectors of vertex P_i and cell center C_j, respectively, as shown in Fig. 1.

After the projection step is performed, the cutting process takes place. For each edge, the values of α_P on its vertexes ($\alpha_{P_{SV}}$ and $\alpha_{P_{EV}}$) are checked (Fig. 1). If the values are such that $\alpha_{P_{SV}} \geq 0.5 \geq \alpha_{P_{EV}}$ or $\alpha_{P_{SV}} \leq 0.5 \leq \alpha_{P_{EV}}$, there is an interception between the surface and the edge. In this case, we call the edge as *interfacial edge*. Given this condition, the point $\overrightarrow{x_e}$ where the interface intercepts the edge is computed by linear interpolation.

$$\overrightarrow{x_e} = \overrightarrow{x_{SV}} + (\overrightarrow{x_{EV}} - \overrightarrow{x_{SV}}) \frac{0.5-\alpha_{P_{SV}}}{\alpha_{P_{EV}}-\alpha_{P_{SV}}} \tag{4}$$

To avoid division by zero, in case $\alpha_{P_{EV}} = \alpha_{P_{SV}}$, the point $\overrightarrow{x_e}$ is set to the edge midpoint. Notice that it only happens in cases where both values are equal do 0.5 within a tolerance.

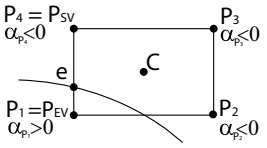


Figure 1. Cell being cut by an interface in 2D. Points e, SV and EV of an *interfacial edge* are highlighted.

Along with the cutting procedure, a first estimate of the interfacial normal vectors is also performed in a similar fashion. This estimate is based on the gradient of the α field. First, the cell-centered field $\overrightarrow{\nabla}\alpha$ is projected onto the mesh vertexes by the same inverse distance interpolation, giving rise to the field $(\overrightarrow{\nabla}\alpha)_P$. Then a linear interpolation of the $(\overrightarrow{\nabla}\alpha)_P$ values at SV and EV is performed for every *interfacial edge*. The normal $\hat{n}_e$ at a given edge is, thus, given by

$$\hat{n}_e = \frac{\vec{N}}{|\vec{N}|} \qquad ; \qquad \vec{N} = (\overrightarrow{\nabla}\alpha)_{P_{SV}} + \left[(\overrightarrow{\nabla}\alpha)_{P_{EV}} - (\overrightarrow{\nabla}\alpha)_{P_{SV}}\right] \frac{0.5-\alpha_{P_{SV}}}{\alpha_{P_{EV}}-\alpha_{P_{SV}}} \tag{5}$$

Again, to avoid division by zero, in case $\alpha_{P_{EV}} = \alpha_{P_{SV}}$, the normal is the normalized average between $(\overrightarrow{\nabla}\alpha)_{P_{EV}}$ and $(\overrightarrow{\nabla}\alpha)_{P_{SV}}$. Figure 2 shows in 2D the resulting points and normals lying on the interceptions between the interface and the edges.

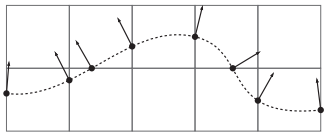
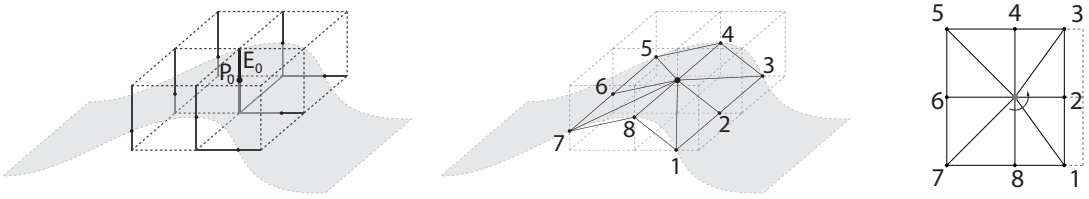


Figure 2. Surface sampled by a set of points and normals lying on the edges.

4.2 Normal Refinement

For the sake of curvature computation, it is highly desirable to obtain normals as accurate as possible. In this work, the normals are computed at each edge that crosses the interface, at the interception point $\overrightarrow{x_e}$. First a raw estimate of the normals is performed by interpolating the $\overrightarrow{\nabla}\alpha$ field as described in the previous section. Then, for each interception point, a set of neighboring points is built. In our context, the neighborhood can be efficiently obtained by traversing the mesh cells around each interfacial edge. Lets denote by $\overrightarrow{P}_0$ the interception point of a given interfacial edge E_0. Its neighboring points are related to the edges E_i neighboring E_0. If E_i is neighbor to E_0 and has an interface point $\overrightarrow{P}_i$, then $\overrightarrow{P}_i$ belongs to the neighborhood of $\overrightarrow{P}_0$. The neighborhood of E_0 is composed by all interfacial edges that belong to the neighboring cells of E_0 as seen in Fig. 3(a). Taking the point $\overrightarrow{P}_0$, its first normal estimate $\hat{n}_0$ – computed by Eq. (5) – and its neighboring points, the normal at $\overrightarrow{P}_0$ is computed as follows. First a plane centered at $\overrightarrow{P}_0$ is defined with $\hat{n}_0$ as its normal. Then, all neighboring points are projected onto this plane and sorted in a counterclockwise way around $\overrightarrow{P}_0$, as in Figs. 3(b) and 3(c). The plane and first normal estimate are needed just for this ordering. With the points sorted, a triangle fan may be constructed around $\overrightarrow{P}_0$. Its normal vectors are computed by cross products such as $(P_i - P_0) \times (P_{i+1} - P_0)$ with proper normalization. The normal at point $\overrightarrow{P}_0$ may be taken as the average normal of all triangles.



(a) Edge E_0 and its neighborhood (b) Neighborhood Triangulation (c) 2D view.

Figure 3. Triangulation around an interfacial edge.

The procedure of normal refinement is performed for every interfacial edge. At the end of this process, good estimates of points and normals describing the interface are obtained. The next step is the curvature computation itself using these points and normals.

5. CURVATURE ESTIMATION ON A POINT CLOUD

Many works can be found in the literature for curvature computations on a point cloud with or without prescribed normals. Many propose the fitting of high order surfaces to the point data, others propose triangulations of these points. Yang and Qian (2007) adjust a *moving least-squares* –MLS–surface (Levin, 2003) to the point cloud with normals and, then, compute the curvatures with closed formulas. This procedure is computationally expensive and depends on a good estimate of a scale parameter, that gives rise to further complexity. Other methods depend on mesh generation to arrange the point cloud, such as Delaunay Triangulation (de Berg *et al.*, 2008).

Seibert *et al.* (2010) defines a finite number of planes containing the normal vector of a point in the cloud. These planes sweep a $360°$ rotation around the normal. For each plane, an osculating circle is fitted to the neighboring points on the vicinity of this plane, giving rise to a value of curvature for each direction. With these values of curvatures, κ_1 and κ_2 are computed. This approach is considered computationally expensive once many fitting problems must be solved for every point in the point cloud. Besides, it depends on a high density of points on the point cloud.

The focus of this work is not on a good representation of the surface itself, but on a good representation of its curvature. Keeping this is mind, the method proposed in Cheng and Zhang (2009) was the one chosen and gave good results. It fits linear functions to describe the normal vector on the vicinity of a given point. From these functions, the curvatures may be extracted. The authors argue that minimization of the algebraic distance between the points of the cloud and a surface – in a least squares sense – does not necessarily result in the minimization of the error in the the curvature estimates. In this way, for the purpose of curvature estimation, fitting functions to normal vectors in local regions is more accurate and robust than fitting to point positions. It makes sense if one notices that curvatures are second-order derivatives of the surface and measure directly the variation of normals itself.

5.1 Curvature Estimation by Normal Fitting

For every point P_0 in the cloud, the curvature should be estimated based on its neighboring points. A local system $(\hat{u}, \hat{v}, \hat{w})$ is defined, where $\hat{w}$ is equal to the normal vector at $\overrightarrow{P}_0$, i.e. $\hat{w} = \hat{n}_0$. The vectors $\hat{u}$ and $\hat{v}$ are such that $\hat{u} \times \hat{v} = \hat{w}$. These vectors give the base for a plane with the origin at point $\overrightarrow{P}_0$. We will call it the *local frame* at point P_0. One could write functions to represent the normal vector of the surface around point P_0 in this local frame as functions of the parameters u and v. The projection of the normal vectors $\hat{n}$ of every point on the vicinity of P_0 may be projected onto the local frame to give rise to its components n_u and n_v, where $n_u = \hat{n} \cdot \hat{u}$ and $n_v = \hat{n} \cdot \hat{v}$. Notice that the projection of $\hat{n}_0$ onto the local frame gives rise to $n_{u0} = 0$ and $n_{v0} = 0$. The normal vectors in the local frame may be described by the following functions

$$n_u(u, v) = \nu_1 + w_{11}u + w_{12}v + o(u^2 + v^2) \qquad ; \qquad n_v(u, v) = \nu_2 + w_{21}u + w_{22}v + o(u^2 + v^2) \tag{6}$$

where w_{11}, w_{12}, w_{21} and w_{22} are the derivatives of the normal vector and $(\nu_1, \nu_2) = (n_{u0}(0,0), n_{v0}(0,0))$. The high order terms may be dropped resulting in linear functions to represent the normal vectors.

The objective function and constraint are given bellow. Details in the constraint may found in Cheng and Zhang (2009).

$$\begin{cases} \text{min} & \sum_i^N \left(\nu_1 + w_{11}\, u_i + w_{12}\, v_i - n_{ui}\right)^2 + \left(\nu_2 + w_{21}\, u_i + w_{22}\, v_i - n_{vi}\right)^2 \\ \text{subject to} & \nu_1\, \nu_2\, w_{11} - (1 - \nu_2^2)\, w_{12} + (1 - \nu_1^2)\, w_{21} - \nu_1\, \nu_2\, w_{22} = 0 \end{cases} \tag{7}$$

where N is the total number of points in the vicinity of the point and n_{ui} and n_{vi} are the coordinates of the normals at point i in the local system. Notice that when the normal at the desired point is well estimated, ν_1 and ν_2 are equal to zero.

This problem brings about a nonlinear system of equations, which may be relatively expensive to be solved. So, it can be simplified in two independent minimization problems. One minimization for the u component and another for the v component without constraints, such as

$$\text{min} \sum_i^N \left(\nu_1 + w_{11}\, u_i + w_{12}\, v_i - n_{ui}\right)^2 \quad \text{and} \quad \text{min} \sum_i^N \left(\nu_2 + w_{21}\, u_i + w_{22}\, v_i - n_{vi}\right)^2 \tag{8}$$

It was observed that the solution of these two minimization problems gives good approximations of the curvature. Once the values of w_{11}, w_{12}, w_{21} and w_{22} are obtained, the curvature may be computed by taking the eigenvalues of the Weingarten curvature matrix $\boldsymbol{W}$.

$$\boldsymbol{W} = -\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \tag{9}$$

The curvatures k_1 and k_2, where $k_1 \geq k_2$, are the eigenvalues of $\boldsymbol{W}$ itself.

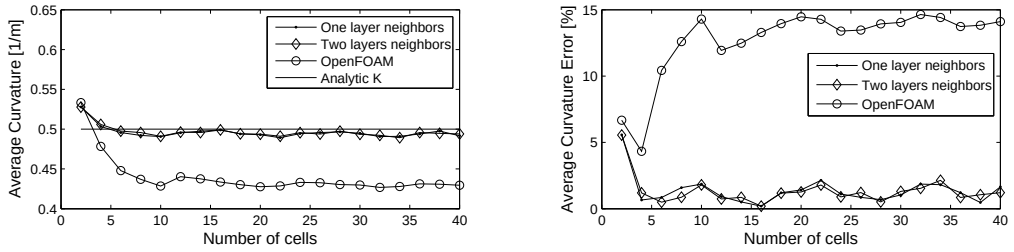
6. RESULTS

The accuracy of the curvature estimation method was studied for drops shaped as a cylinder and as an elliptic cylinder. The cylinder is such that its radius is 2m and the ellipse has major and minor axes of 2m and 1m, respectively. Every simulation in OpenFOAM® is essentially 3D, so these cases are of 2D shapes extruded in z direction. The first test was performed to compare the average curvature of the proposed method with the ones obtained with standard OpenFOAM® code. Mesh resolution was varied in order to observe its role on curvature values. Also, the cylinder was set to motion and its average curvature was obtained with the finest mesh resolution. The ellipse was placed in a rotating velocity field and its local curvatures were also determined.

6.1 Static Cylinder

For various mesh sizes, the values of curvature, error in positions and error in normals were computed. It was observed that increasing the number of neighboring points, the accuracy of the curvature slightly increases. In this way, the neighboring points used to compute curvature are the points associated to the first two layers of cells neighboring the edge that contains the interfacial point. Figure 4(a) displays a comparison between the average curvatures obtained for the cylinder employing the standard OpenFOAM® formulation, and employing one layered neighboring points and two layered neighboring points, with the present formulation. The analytic result is $\kappa = 0.5$. Figure 4(b) displays its respective errors in percentage. The errors were computed as $|\kappa - \kappa_{\text{analytic}}|/\kappa_{\text{analytic}} \times 100$. It can be observed that the errors associated to the average curvatures of the proposed method is significantly smaller than the ones associated to standard OpenFOAM® . Further, it can also be observed a larger mesh dependency with the OpenFOAM® formulation.

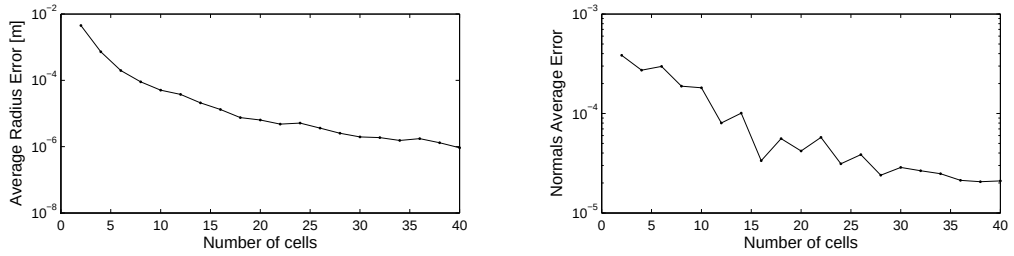
Figure 5 displays errors in the sampling points positions, Fig. 5(a) and normals, Fig. 5(b) . Once the axis of the cylinder and its radius are known, the distance d from the sampling points to the axis are computed and compared to the



(a) Average Curvature (b) Average Curvature Errors

Figure 4. Average curvatures and its errors for the 2m radius cylinder.

radius r. The errors in positions are, thus defined as the deviation of d from radius r, i.e. $error_{\text{position}} = |d - r|$. Given a sampling position P_i, an analytic normal $\hat{n}_a$ is computed as the vector connecting the closest point of the axis to the sampling point, i.e. $\hat{n}_a = (P_i - P_{\text{closest}})/|P_i - P_{\text{closest}}|$. The error in normals is the magnitude of the deviation of the computed normal $\hat{n}_i$ from the analytic normal $\hat{n}_a$, i.e. $error_{\text{normal}} = |\hat{n}_a - \hat{n}_i|$. It can be noticed that although the errors are quite small, the errors in positions and normals depend on the mesh refinement, while the average curvature values do not show a dependency on mesh discretization.



(a) Average Radius Error (b) Normals Average Error

Figure 5. Average error in the sampling points and normals for the 2m radius cylinder.

6.2 Moving Cylinder

The same cylinder of 2m radius was set to motion in a uniform velocity field $\overrightarrow{V} = [2\ 0\ 0]^T m/s$ with a mesh resolution of $0.05m$, such that the cylinder occupies a block of 40×40 cells. The average curvature at every time step was computed and is presented in Figure 6 along with with the exact curvature. It can be seen that the mean curvature oscillates around a mean value. The range between the maximum ($0.4956m^{-1}$) and minimum ($0.4848m^{-1}$) values is $0.0108m^{-1}$, what accounts for 2.2% of the mean value ($0.4907m^{-1}$).

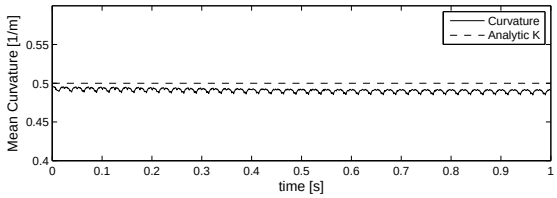
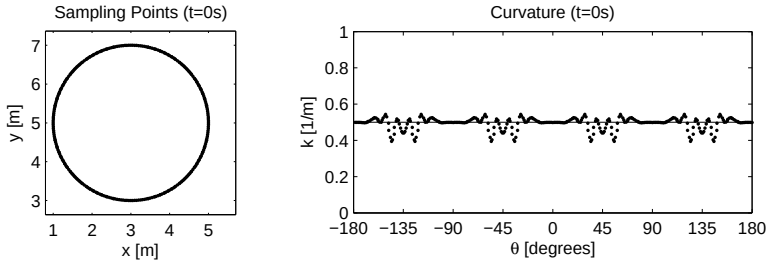


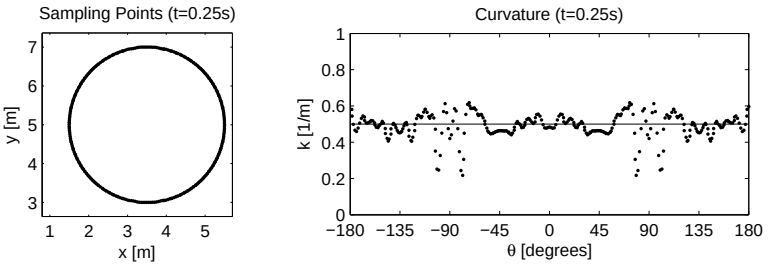
Figure 6. Mean curvatures for the moving cylinder.

Results for the sampling points at the beginning of the movement ($t = 0$s) are presented at Fig. 7(a), where an excellent representation of the cylinder can be seen. At the same time instant, local curvatures are presented in Fig. 7(b). The curvatures are plotted against the angle θ, where $\theta = \text{atan}[(y - y_c)/(x - x_c)]$. For $t = 0$, $[x_c, y_c] = [3, 5]$. The reader may notice the oscillations on curvatures around a mean value, occurring at angles multiple of 45^o. The largest curvature estimate is equivalent to the mean curvature predicted by the standard OpenFOAM® . As time evolves, the cylinder travels along the x direction. Excellent prediction of its sampling points are still obtained at $t = 0.25$s, Fig. 7(c), however, the maximum and minimum values of the curvature estimates increases. Further, the curvature oscillations occurs at different location as illustrated at Fig. 7(d). The curvature oscillations are due to perturbations in the points estimates. Figure 8 depicts the errors in the sampling points, for $t = 0$s and $t = 0.25$s, where the mentioned oscillations can be seen, and although small, they can disturb the interface prediction. The oscillations in the local error increases

the errors in its normals, and these errors propagate towards the curvatures. The authors are aware of this issue and are devising a way to mitigate the oscillations. This is the major problem with the present method.

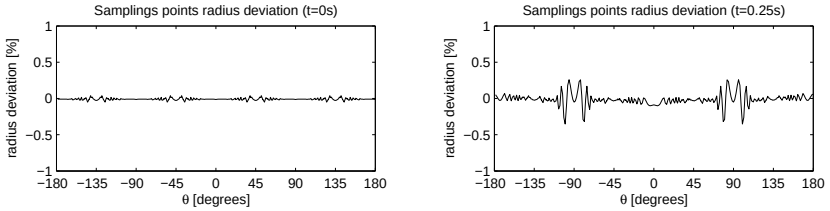


(a) Sampling points, $t = 0$s (b) Curvature, $t = 0$s



(c) Sampling points, $t = 0.25$s (d) Curvature, $t = 0.25$s

Figure 7. Sampling points and curvatures for the cylinder moving with constant velocity.



(a) $t = 0$s (b) $t = 0.25$s

Figure 8. Sampling points deviation in percentage of the radius for the moving cylinder.

6.3 Rotating Ellipse

A rotating ellipse with major and minor axes of 2m and 1m, respectively, was placed in a constant rotating velocity field $\overrightarrow{V}(\overrightarrow{x}) = \overrightarrow{\omega} \times \overrightarrow{x}$, where $\overrightarrow{\omega} = [0\ 0\ \omega]$ and the value of ω was set to $2\pi\ rad/s$, so that in 1 second, the ellipse completes a full revolution. Results of sampling points are presented in Fig. 9, where excellent result was obtained, with the ellipse shape being maintained during the movement.

Figure 10 presents the local curvatures for each sampling point for time steps of 0s, 0.125s and 0.25s around the θ angle, where $\theta = \text{atan}(y/x)$. Good agreement with the analytical results is observed in the curvature for initial time step. For subsequent time steps, the curvature is still reasonable, however, it presents a deterioration of the maximum value, and an oscillation can also be seen in the valley of the curve. The same arguments for the cylinder are valid here, where the oscillations in the sampling positions deteriorate the curvature results.

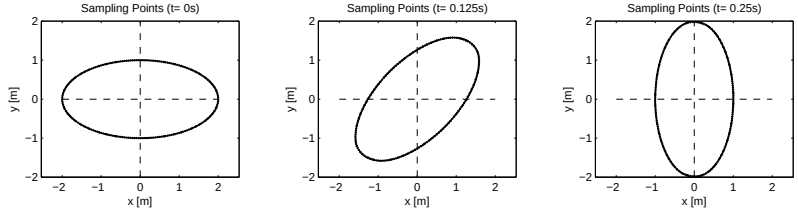


Figure 9. Sampling points for the rotating ellipse at t= 0, 0.125 and 0.25s.

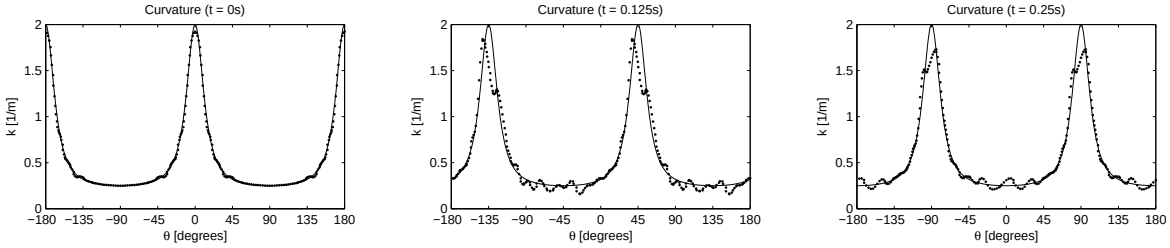


Figure 10. Local curvatures for the rotating ellipse at t= 0, 0.125 and 0.25s. Solid lines are the analytical results.

7. CONCLUSIONS

As discussed in the previous section, the proposed method for curvature computation in the VOF framework of OpenFOAM® improves the accuracy on the mean curvature with respect to the standard formulation of OpenFOAM® . However, locally the errors increase with time as the interface moves. The main reason for this is associated to the oscillations on the sampling points around the interface. Further investigation is required in order to alleviate the oscillations in curvature estimates.

8. ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support provided by CNPq and CAPES.

9. REFERENCES

Brackbill, J., Kothe, D. and Zemach, C., 1992. "A continuum method for modeling surface tension". *Journal of Computational Physics*, Vol. 100, No. 2, pp. 335 – 354. ISSN 0021-9991.

Brennen, C.E., 2005. *Fundamentals of Multiphase Flow*. Cambridge University Press, Cambridge. ISBN 0521848040.

Cheng, Z. and Zhang, X., 2009. "Estimating differential quantities from point cloud based on a linear fitting of normal vectors". *Science in China Series F: Information Sciences*, Vol. 52, No. 3, pp. 431–444.

Cummins, S.J., Francois, M.M. and Kothe, D.B., 2005. "Estimating curvature from volume fractions". *Computers & Structures*, Vol. 83, pp. 425 – 434. ISSN 0045-7949. Frontier of Multi-Phase Flow Analysis and Fluid-Structure Frontier of Multi-Phase Flow Analysis and Fluid-Structure.

de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M., 2008. *Computational Geometry: Algorithms and Applications*. Springer-Verlag Berlin Heidelberg, 3rd edition. ISBN 9783540779735.

Helmsen, J. and Colella, P., 1997. "Non-convex profile evolution in two dimensions using volume of fluids". Technical report, Lawrence Livermore National Lab., CA (United States).

Ishii, M. and Hibiki, T., 2011. *Thermo-Fluid Dynamics of Two-Phase Flow*. Springer, 2nd edition. ISBN 9781441979858.

Kreyszig, E., 1991. *Differential Geometry*. Dover Publications, USA, 1st edition. ISBN 0486667219.

Levin, D., 2003. "Mesh-independent surface interpolation". *Geometric Modeling for Scientific Visualization*, Vol. 3.

Renardy, Y. and Renardy, M., 2002. "Prost: A parabolic reconstruction of surface tension for the volume-of-fluid method". *Journal of Computational Physics*, Vol. 183, No. 2, pp. 400 – 421. ISSN 0021-9991.

Rider, W.J. and Kothe, D.B., 1998. "Reconstructing volume tracking". *Journal of Computational Physics*, Vol. 141, No. 2, pp. 112 – 152. ISSN 0021-9991.

Seibert, H., Hildenbrand, D., Becker, M. and Kuijper, A., 2010. "Estimation of curvatures in point sets based on geometric algebra". In *VISIGRAPP 2010. Proceedings*. pp. 12–19.

Sussman, M., 2003. "A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles". *Journal of Computational Physics*, Vol. 187, No. 1, pp. 110 – 136. ISSN 0021-9991.

Sussman, M. and Ohta, M., 2009. "A Stable and Efficient Method for Treating Surface Tension in Incompressible Two-Phase Flow". *SIAM Journal on Scientific Computing*, Vol. 31, No. 4, pp. 2447–2471. ISSN 1064-8275.

Sussman, M. and Puckett, E.G., 2000. "A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows". *Journal of Computational Physics*, Vol. 162, No. 2, pp. 301 – 337. ISSN 0021-9991.

Williams, M.W., 2000. *Numerical methods for tracking interfaces with surface tension in 3-D mold-filling processes*. PhD thesis, University of California, Davis.

Yang, P. and Qian, X., 2007. "Direct Computing of Surface Curvatures for Point-Set Surfaces." In *SPBG'07*. pp. 29–36.

10. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.