ABSTRACT

TALLEY, MATTHEW LOWELL. Bubble Coalescence Control Development for Level Set Interface Tracking Method. (Under the direction of Dr. Igor A. Bolotnov).

The level-set interface tracking method in connection with direct numerical simulation of turbulence is an important tool for development of improved closure laws for multiphase computational fluid dynamic models. However, the standard formulation of level-set interface tracking method is unable to accurately represent the bubble coalescence process. Physically, when two bubbles approach, a thin liquid film develops between the bubbles. If this film has sufficient time to drain, then the bubbles will coalesce. Otherwise, the bubbles will bounce off one another. However, the standard level set method will cause coalescence of any bubbles that approach close to one another. Also, since the level-set method uses a smoothed Heaviside function to transition between phase properties, this causes the coalescence process to begin sooner than experimentally observed since the thin liquid film represented by the method has somewhat mixed gas/liquid properties. In order to simulate the coalescence process more accurately, an algorithm was developed to prevent or slow the coalescence process. This algorithm locally changes the surface tension on a portion of the bubble surface when it detects that two bubbles approach each other. This local change in surface tension creates a net force that repels the bubbles. The algorithm is also capable of tracking the amount of time the surface tension has been changed to slow the coalescence process. It compares it to a coalescence time model and removes the changed surface tension if the prescribed model time has been exceeded. In order to test the capabilities of the algorithm, the following simulations were designed and performed: (i) two bubbles in laminar flow approaching one another, (ii) 32 bubbles in turbulent flow conditions, and (iii) a bubble rising towards a free surface. The first

two simulations tested the identification and prevention portion of the algorithm. The last simulation tested the time tracking portion of the algorithm. In all cases, the program was able to prevent or slow the coalescence process. An increase in computational cost from 10-25% was observed when using this algorithm. Mesh studies and another set of simulations were performed in order to verify the algorithm is performing properly and better simulate physical coalescence.

Bubble Coalescence Control Development for Level Set Interface Tracking Method

by
Matthew Lowell Talley

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Nuclear Engineering

Raleigh, North Carolina

2015

APPROVED BY:

Dr. Igor A. Bolotnov Committee Chair	Dr. Nam Dinh
Dr. Steven C. Shannon	Dr. Jack R. Edwards Jr.

BIOGRAPHY

The author graduated from Brigham Young University (BYU) with a Bachelor of Science degree in Mechanical Engineering in 2012. As an undergraduate student at BYU, started work on a research project with a professor in developing an algorithm to study the evolution of nuclear fuel composition during burn up. While attending BYU, he also took part in a Student Undergraduate Laboratory Internship at the Princeton Plasma Physics Laboratory (PPPL) in Princeton, New Jersey. He also deferred his studies for two years to serve as a missionary in Kyiv, Ukraine for the Church of Jesus Christ of Latter Day Saints. While there, he was able to develop fluency in the Ukrainian language. His research experience at BYU and PPPL lead him to pursue a Master of Science degree enroute to a Doctor of Philosophy degree both in Nuclear Engineering. As hobbies, he enjoys rock climbing, watching movies, and spending time with his family.

ACKNOWLEDGMENTS

I would first like to give my appreciation to Heavenly Father and His Son who have blessed me greatly and given me the faith and strength to become the person I am today. I would also like to deeply thank my wife McKenna for all her support, love, and patience. She has endured many long days waiting patiently for me to finish and fulfill my responsibilities for work and school. I thank my parents for all their love and support. I appreciate my father for the example he has set for me as a husband, provider, and man of God as well as his support in all my endeavors. I also have deep gratitude for my mother and the love, help, and nurturing she has provided throughout my life. She has always been there for me no matter the situation.

I want to express my most heartfelt appreciation for Dr. Bolotnov and his guidance throughout this whole experience. His knowledge and experience in computational fluid dynamics (CFD) have been instrumental in developing my foundation as a future researcher and engineer. I also want to thank Jun Fang and Matt Thomas for all their help in working through the basics of CFD and in generating insightful discussions to develop innovative solutions. Lastly I would like to express my appreciation for Eric Homer and his advice and encouragement before embarking on my graduate studies. He helped me begin the development of the necessary traits to perform quality research as an undergraduate student.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

The development of new generation of advanced nuclear reactors requires robust tools to predict thermal hydraulic behavior in complex geometries. Interface tracking approach with direct numerical simulation (DNS) of turbulence may not yet allow the prediction of large systems, but represents a valuable tool in development of closure laws for multiphase computational fluid dynamics (M-CFD) models. Massively parallel finite element based code, PHASTA [1] is a one such DNS software package that uses level set interface tracking to model the fully resolved bubbly flows. However, the standard level-set approach lacks the capability to represent the physics of bubble coalescence.

The objective of this research is to present an algorithm developed for the level-set approach that is used in PHASTA that controls bubble coalescence events. This consists of how the algorithm works to identify and simulate coalescence events in multiphase bubbly flows, 3D simulation results when using the algorithm, and the algorithm's verification and validation based on available experimental results.

Chapter 2 consists of a literature review that details previous research performed on bubble coalescence through analytical, experimental, and numerical studies. The former includes research with DNS and the level-set approach.

Chapter 3 describes the development of the algorithm to: (*i*) simulate liquid film the drainage time using a force to counteract the motion of the approaching bubbles; (*ii*) identify the coalescence event locations; (*iii*) model how long the force should be active for each event to give the bubbles the opportunity to move away from (bounce off) one another.

Chapter 4 presents several simulations performed using the coalescence control algorithm for different purposes based on certain conditions. This includes preventing all coalescence events and using the application time portion of the algorithm to allow some events to occur.

Chapter 5 covers the verification and validation of the coalescence control algorithm based on other experimental and simulation results.

Chapter 6 provides a discussion of the results and covers the necessary future work to continue to develop the algorithm.

CHAPTER 2 LITERATURE REVIEW

2.1 General Overview

Up until recently, the study of bubble coalescence in two-phase flow could be characterized into two major categories: surfactant/impurity effects on coalescence and the mechanics behind the coalescence process [2]. However, with the continuing development of computing power, a third category could be added. This third category consists of applying the analytical models of the first two categories to numerical analysis of bubble coalescence.

The analytical analysis of bubble coalescence mainly focuses on the drainage of the thin liquid film that develops between two approaching bubbles. It was determined in early analytical studies of bubble coalescence to break the process into two separate regimes: (1) an initial drainage regime where the film drains to a thickness such that rupture can occur and (2) a final drainage regime where molecular forces become significant and rupture occurs [3], [4], [5], [6]. An extensive review of the concepts used in early analytical models is covered by Lee and Hodgson [4] as they described the types of forces present near the film, the flow

of the liquid film, and the mobility of the interface under varying surfactant effects (See Figure 1). As the research progressed, models were developed to calculate the film thickness of the bubbles at the different drainage regimes based on physical properties [3], [5], [6]. These film thickness calculations also lead the authors to develop models for the time required for coalescence to occur. Prince and Blanch [7] furthered the development of coalesce time models by taking into account bubble collisions caused by turbulence, buoyance, laminar shear as well as the efficiency of these collisions. Their model was developed with distilled water as an assumption but was found to be inadequate when applied to systems with a significant amount of solute because of the large decrease in the coalescence rate. This was attributed to the effects of turbulence on surface mobility, the dynamics of bubble collisions, and the solute concentration at the gas-liquid interface of the coalescing bubbles.

Many analytical models for coalescence were also developed in conjunction with experimental studies and measurements. As mentioned previously, many of the studies focused on one of two categories. For a more comprehensive review on surfactant effects on bubble coalescence, one is referred to Marrucci and Nicodemo [8], Lessard and Zieminski [9], Cain and Lee [10], Kim and Kook [11], Craig *et al.* [12], and Danov *et al.* [13]. For the mechanics of bubbles coalescence, the areas of study frequently focused on the dynamics of the liquid flow and the properties of the liquid in which the bubbles are immersed. It was found that for bubbles vertically aligned, coalescence would occur even if the calculated infinite rise velocity suggested that it was impossible [14].

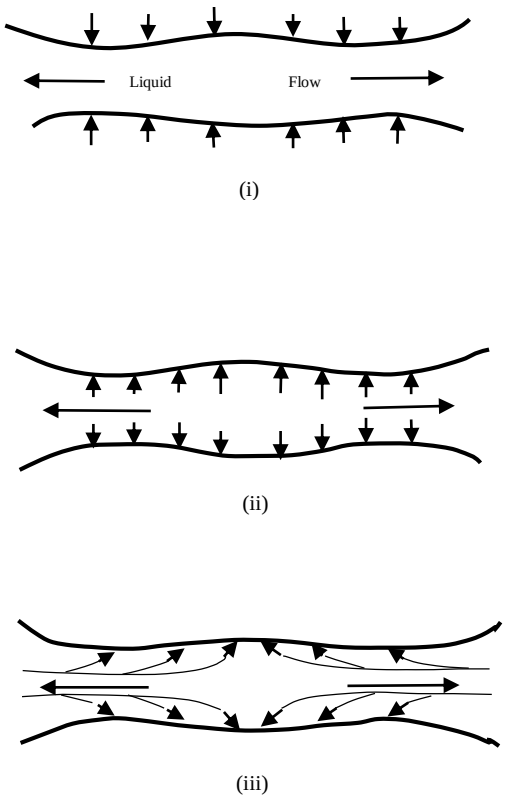


Figure 1: Figure reproduced from Lee and Hodgson [4]. Interface mobility with soluble surfactant: expansion determined by mass transfer. (i) Normal diffusion from outside film. (ii) Normal diffusion from inside film. (iii) Radial diffusion following depletion of film.

For bubbles rising in a stagnant liquid, it was determined that the only forces active on the second bubble were the buoyance force and the inertia drag force. The second bubble could approach the first bubble to coalesce because the inertia drag force would be reduced once the second bubble entered the wake of the first bubble [14], [15]. It was also found by de Nevers and Wu [15] that it was necessary to add a small amount of sodium ethyl xanthate to the distilled water in order to reduce surface tension and promote coalescence. This result may also be explained in conjunction with the work of Kirkpatrick and Lockett. They observed that when bubbles approach one another with a velocity greater than $12\,cm/s$, the bubbles would

come to rest before the film thickness would rupture [16] (See Figure 2). This allows the stored strain energy in the film to act on the bubbles and push them away from one another resulting in the bubbles bouncing off one another.

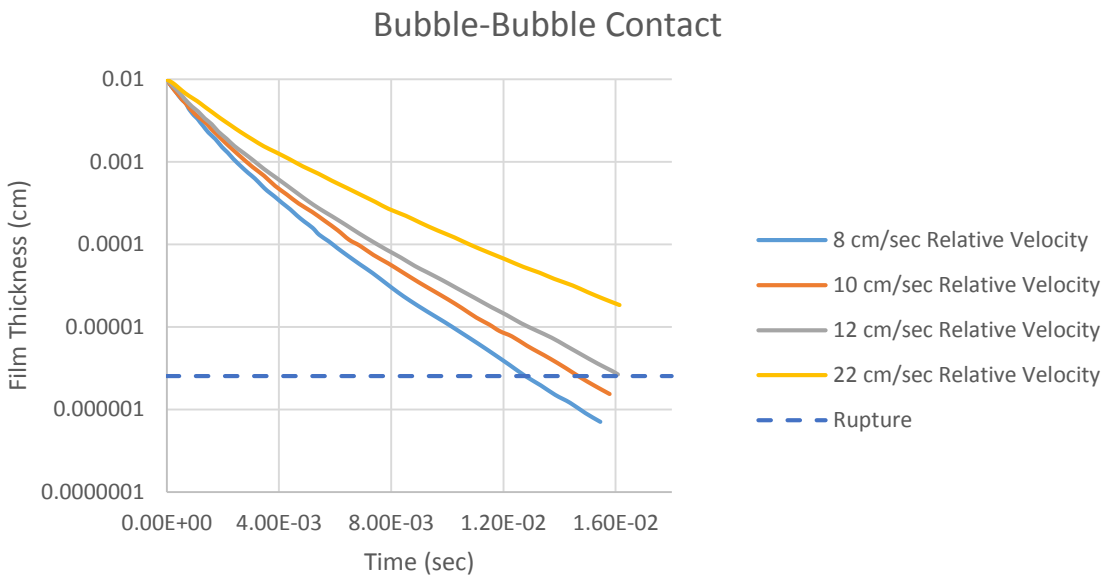


Figure 2: Graph of film thickness vs. time reproduced from data reported by Kirkpatrick and Lockett [16]. The above plot shows that only approach velocities less than 12 cm/s allow enough time for the liquid film to drain and the interface to rupture.

As these results were obtained, more focus on coalescence was turned to measuring the property effects of the liquid in conjunction with the flow characteristics. These studies generally consisted of using non-dimensional numbers to find relations between coalescence and property effects [2], [17], [18]. Stewart [17] observed that there is a transition region for coalescence mechanics with the Morton (Mo) number. He found that when $10^{-6} < Mo < 10^{-4}$ that the in-line collision coalescence ceases and that coalescence only occurs after the

collision when the bubbles begin to "dance" with one another. Sanada *et al.* [2] in their 2005 study observed that an increasing Weber (We) number lead to a bubble bouncing off a free surface which in effect lengthens the coalescence time. A later study in 2009 by Sanada *et al.* [18] showed that the critical Reynolds (Re) number over which bubbles bounced decreased with an increase in Mo. They also showed that irrespective of the Mo value, a critical We value for bouncing was approximately equal to 2.0. It is important to note that for both Re and We in the 2009 study that a vertical rise velocity was used to calculate the non-dimensional numbers. Unlike the previous experiments, Kang *et al.* [19] performed an experiment that did not use non-dimensional numbers. They measured the effects of nonuniform temperature distribution on bubble coalescence. They found that because of thermocapillary forces from the nonuniform temperature, it caused one of the bubbles to glide over the surface of the other bubble which resulted in a coalescence probability distribution based on coalescence angle. It was observed that the highest probability of coalescence occurred between $20° - 40°$ angle. The angle between the temperature gradient and line connecting the bubble centers (See Figure 3).

As the mechanics of coalescence became better understood, some research shifted focus to modeling coalescence rates and void fraction over the length of a tube with bubbly flow [20], [21]. Kamp *et al.* [22] took the work further by making the models more robust through two steps. The first step took an expression for collision frequency and coalescence probability of equal bubbles during turbulence-driven, high Re collisions [23] and altered it to be applicable for unequal bubbles and to account for interactions between the bubbles and flow as well as between bubble-bubble. In the second step, the coalescence rate is used in the

transport equation to find source terms which can be calculated in a CFD code. This results in a more robust CFD code to predict evolution of bubble sizes. Mattson and Mahesh [24] took the expression of coalescence time scale from the previous model by Kamp *et al.* [22] and used it to develop a probability of coalescence. They then validated it using the result form Colin *et al.* [20] and find excellent agreement between the bubble size distributions. For a more comprehensive review of CFD studies on bubble coalescence, one is referred to Olmos *et al.* [25], Sommerfeld *et al.* [26], and van den Hengel *et al.* [27]. However, since CFD codes use approximations, they are unable to account for all of the fluid and bubble behavior and require DNS studies [24], [28].

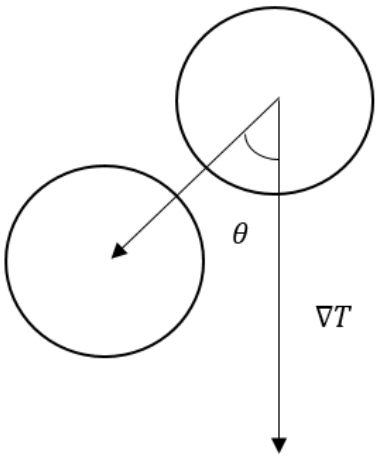


Figure 3: The picture shows how the coalescence angle is defined.

DNS simulations use numerical techniques to solve the time dependent Navier-Stokes equations in three dimensions [29]. For a review of multiphase DNS simulations, one is

referred to Crowe *et. al.* [30]. For DNS simulations, many different methods have been proposed to track the bubble interfaces including the Front-Tracking (FT) method, the Volume-of-Fluid (VOF) method, the Lattice Boltzmann (LBM) method, and the Level-Set (LS) method. The majority of the mention methods inherently incorporate coalescence of bubbles without any issue. The FT method however uses two separate grids for the two different phases and is unable to represent coalescence without a subgrid model. The VOF method can simulate coalescence but it is difficult to calculate the curvature from the front using volume fractions. Van Sint Annaland *et al.* [31] and Sussman *et al.* [32] provide a more comprehensive review of each of the above mentioned interface tracking methods. For more specific review of the FT, VOF, and LBM, one is referred to the following: Unverdi and Tryggvason [33], van Sint Annaland *et al.* [31], and Dabiri *et al.* [29] for FT, van Sint Annaland *et al.* [34] and Passandideh-Fard *et al.* [35] for VOF, and Takada *et al.* [36] and Inamuro *et al.* [37] for LBM.

The LS method uses a distance field to track the bubbles. The bubble-fluid interface is identified by a distance field value of zero. The method also uses a smoothed Heaviside function to shift between gaseous and liquid properties across the interface. For more information on the LS method, one is referred to two papers by Sussman *et al.*, one published in 1994 [38] and one published in 1999 [32], and by Osher and Fedkiw [39]. The LS method was further tested and validated using experimental data from Kirkpatrick and Lockett [2]. A bubble and free surface were modeled in a two dimensional simulation with a grid resolution of 150 x 150 or 40 points per diameter. In order to prevent coalescence, the two interfaces were modeled using two independent distance functions. The bubble bounced with the free surface

in the simulation just as seen in the experiment (See Figure 4). It was also determined that the We number causing bouncing is constant and that the coalescence time increases with increasing We number. Yang *et al.* [40] continued development with the LS method by combining it with the VOF method to make a hybrid called the adaptive coupled level-set/volume-of-fluid (ACLSVOF) method. They found that ACLSVOF took advantage of the strengths of both the LS and VOF methods by making the surface tension calculations easier and more accurate while also keeping the mass conserved accurately. Yu and Fan [41] also studied the accuracy of the LS method. They performed simulations for a bubble rising in an infinite liquid by means of the buoyancy force in a three dimensional environment. They investigated the bubble shapes based on the Re and found good agreement between their simulations and experimental results. They also studied the shape of bubbles during coalescence with the LS method and noted a deviation in the shape of the second bubble compared with a single bubble case. They mentioned that no successful method has been developed to portray bouncing and coalescence based on the dynamic conditions of when they collide and that whether the bubbles will coalesce or bounce must be decided before the simulation is run.

2.2 Level-Set Method Overview

In order to understand how the coalescence control algorithm detects coalescence events, a more in depth understanding of the LS method is required. As previously mentioned, one may find more information on the LS method in two papers by Sussman *et al.*, one published in 1994 [38] and one published in 1999 [32], and by Osher and Fedkiw [39]. For this deeper

look into the LS method, the following section has been taken from a paper written by Bolotnov *et al.* [42] to provide the necessary background.

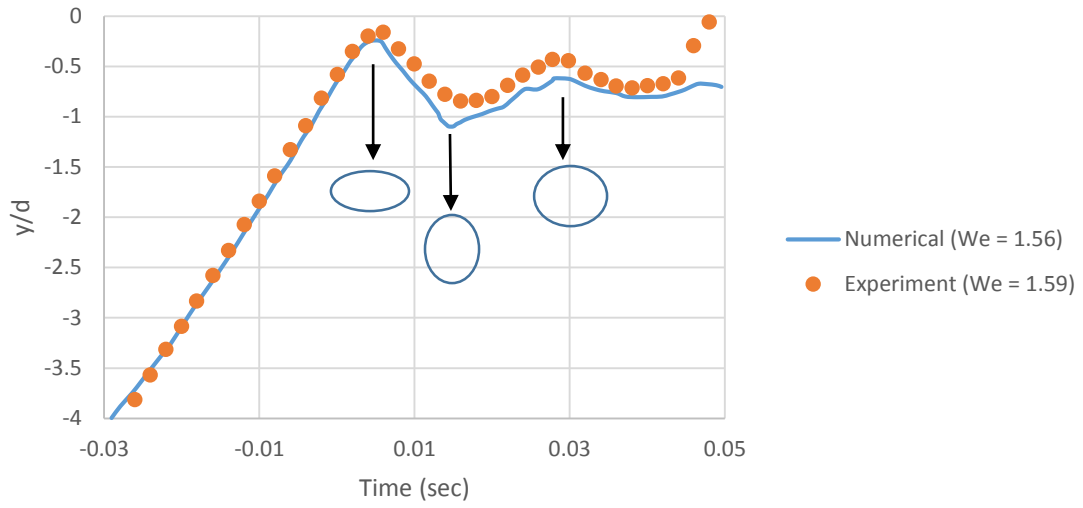


Figure 4: Plot reproduced from data reported by Sanada et. al [2]. The above plot is a comparison of experimental results and calculations. It shows that their simulation was able to closely model what was observed by Kirkpatrick and Lockett [16].

The level set method of Sussman [43], [44], [32] and Sethian [45] involves modeling the interface as the zero-level set of a smooth function, φ, where φ is often called the first scalar and it represents the signed distance from the interface. Hence, the interface is defined by $\varphi = 0$. The scalar, φ, is convected within a moving fluid according to,

$$\frac{D\varphi}{Dt} = \frac{\partial \varphi}{\partial t} + \underline{u} \cdot \nabla \varphi = 0 \qquad (1)$$

where $\underline{u}$ is the flow velocity vector. Phase-1, the liquid phase, is indicated by a positive level set, $\varphi > 0$, and phase-2, the gas, by a negative level set, $\varphi < 0$. Since evaluating the jump in

physical properties using a step change across the interface leads to poor computational results, the properties near an interface were defined using a smoothed Heaviside kernel function (See Figure 5), H_ε, given by [32]:

$$H_\varepsilon(\varphi) = \begin{cases} 0 & ,\varphi < -\varepsilon \\ \frac{1}{2}\left[1 + \frac{\varphi}{\varepsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\varphi}{\varepsilon}\right)\right] & ,|\varphi| < \varepsilon \\ 1 & ,\varphi > \varepsilon \end{cases} \quad (2)$$

where ε is the interface half-thickness.

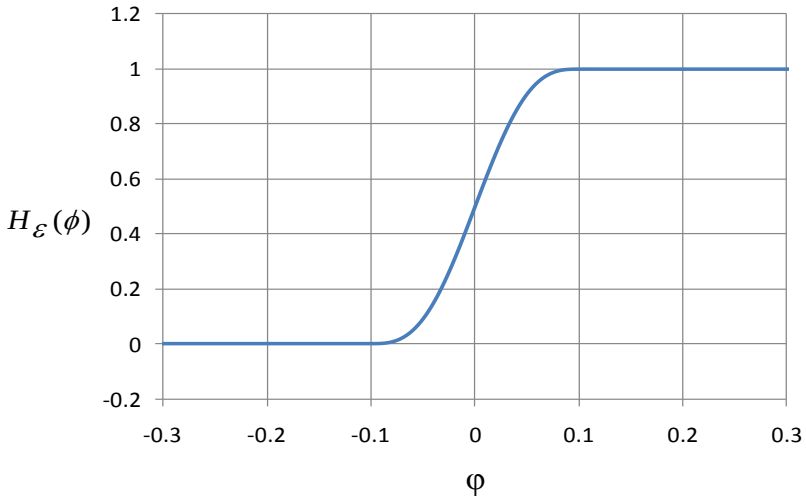


Figure 5: A plot of the smoothed Heaviside function that represents the transition of gas to liquid fluid properties

The fluid properties are then defined as:

$$\rho(\varphi) = \rho_1 H_\varepsilon(\varphi) + \rho_2(1 - H_\varepsilon(\varphi)) \quad (3)$$

$$\mu(\varphi) = \mu_1 H_\varepsilon(\varphi) + \mu_2(1 - H_\varepsilon(\varphi)) \quad (4)$$

The surface tension transition between the fluids is defined by:

$$\gamma\kappa\delta(n)\boldsymbol{n} = \gamma\kappa(\varphi)\frac{dH_\varepsilon(\varphi)}{d\varphi}\nabla(\varphi) \qquad (5)$$

Although the solution may be reasonably good in the immediate vicinity of the interface, the distance field may not be correct throughout the domain since the varying fluid velocities throughout the flow field distort the level set contours. Thus, the level set was corrected with a re-distancing operation by solving the following PDE [44]:

$$\frac{\partial d}{\partial \tau} = S(\varphi)\left[1-|\nabla d|\right] \qquad (6)$$

where d is a scalar that represents the corrected distance field and τ is the pseudo time over which the PDE is solved to steady-state. This may be alternately expressed as the following transport equation:

$$\frac{\partial d}{\partial \tau} + \underline{w}\cdot\nabla d = S(\varphi) \qquad (7)$$

The so-called second scalar, d, is originally assigned the level set field, φ, and is convected with a pseudo velocity, $\underline{w}$, where,

$$\underline{w} = S(\varphi)\frac{\nabla d}{|\nabla d|} \qquad (8)$$

and $S(\varphi)$ is defined as:

$$S(\varphi) = \begin{cases} -1 & ,\varphi < -\varepsilon_d \\ \left[\dfrac{\varphi}{\varepsilon_d} + \dfrac{1}{\pi}\sin\left(\dfrac{\pi\varphi}{\varepsilon_d}\right)\right] & ,|\varphi| < \varepsilon_d \\ 1 & ,\varphi > \varepsilon_d \end{cases} \qquad (9)$$

where ε_d is the distance field interface half-thickness which, in general, may be different from ε used in Eq. (2). Note that the zeroth level set, or interface, $\varphi = 0$, does not move since its convecting velocity, $\underline{w}$, is zero. Solving the second scalar to steady-state restores the distance field to $\nabla d = \pm 1$ but does not alter the location of the interface. The first scalar, φ, is then updated using the steady solution of the second scalar, d.

Sussman et al. [32] and Sussman and Fatemi [44] proposed an additional constraint to be applied during the re-distancing step to help ensure the interface ($\varphi = 0$) does not move. It has been found in the present work that imposing this constraint also improves the convergence of the re-distancing step. The essence of the constraint is to preserve the original volume (i.e., mass) of each phase during the re-distance step.

The re-distancing step of the LS method is of vital importance to the coalescence control algorithm. It is necessary that the LS distance field contours return to the proper step after the advection of the velocity field since the curvature of these contours is used in identifying coalescence events.

CHAPTER 3 ALGORITHM DEVELOPMENT

The previous work of Sanada *et al.* [2] shows that it was possible to simulate the thin liquid film viscosity effects in a 2D simulation without any grid dependencies. It was demonstrated that the film did not have time to drain before the bubble was repelled/bounced as observed by Kirkpatrick and Lockett [16]. This simulation used a resolution of 150 x 150 elements which meant that there were 20 elements per bubble radius. The bubble diameter ranged from 1.6-2.0 *mm*. The simulation domain used in computation used symmetry so that

the calculations were only performed on half of the domain. If the largest bubble diameter were used and the same resolution was applied to a small 3D domain of 20mm x 10mm x 10mm used in PHASTA simulations, the resolution would be 400 x 200 x 200 (16M elements) which is much finer than the finest resolution used in PHASTA of 160 x 80 x 80 (1M elements) for a 5 *mm* diameter bubble. This much finer mesh for the small 3D domain is feasible but this domain contains only two bubbles.

The goal is to prevent coalescence or model the coalescence dynamics for hundreds of bubbles. If the two bubble domain is taken as a standard, it would need to be 50 times larger to allow for 100 bubbles resulting in a 1000mm x 10mm x 10mm domain. Using the resolution from Sanada *et al.*, this would create a resolution of 20000 x 200 x 200 (800M elements) for one hundred 5 *mm* diameter bubbles. The computational cost would be much too high for a simulation of this size. This means another method is required to provide the physics of film drainage in the simulation of bubble coalescence.

3.1 Repulsive Force

The method chosen to allow the use of a rougher resolution was to apply a force acting in opposite directions to each bubble interface in order to prevent the coalescence event or slow the process. This force represents the liquid film drainage process between the approaching bubbles which cannot be modeled directly due to the high resolution requirements. Since multiple factors affect the liquid film drainage (e.g. velocity, bubble approach angle) it is necessary that this force will allow for coalescence events under certain flow conditions.

To apply this force, it was decided to locally increase the surface tension within a designated area of each bubble. The increase in surface tension was chosen because it was the most computationally affordable method to prevent coalescence compared with the other options (e.g. artificially increasing viscosity between the bubbles which requires much larger mesh resolution). Since the surface tension is not adjusted uniformly, this results in net force acting on each bubble in the direction opposite to the coalescence path (See Figure 6, Figure 7, and Figure 8).

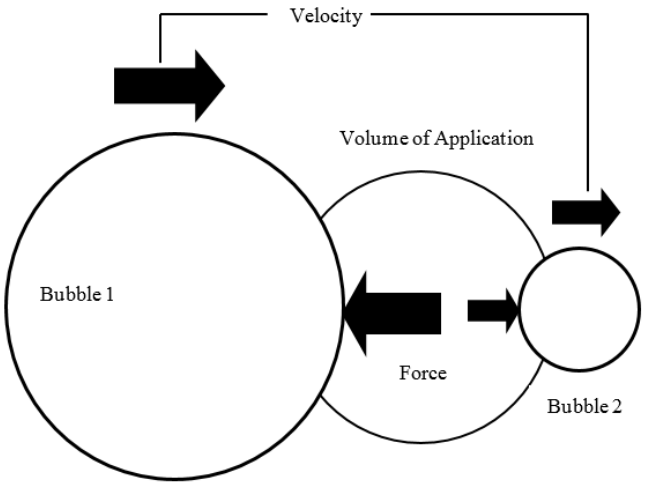


Figure 6: Schematic of how the coalescence control is implemented and restricts coalescence.

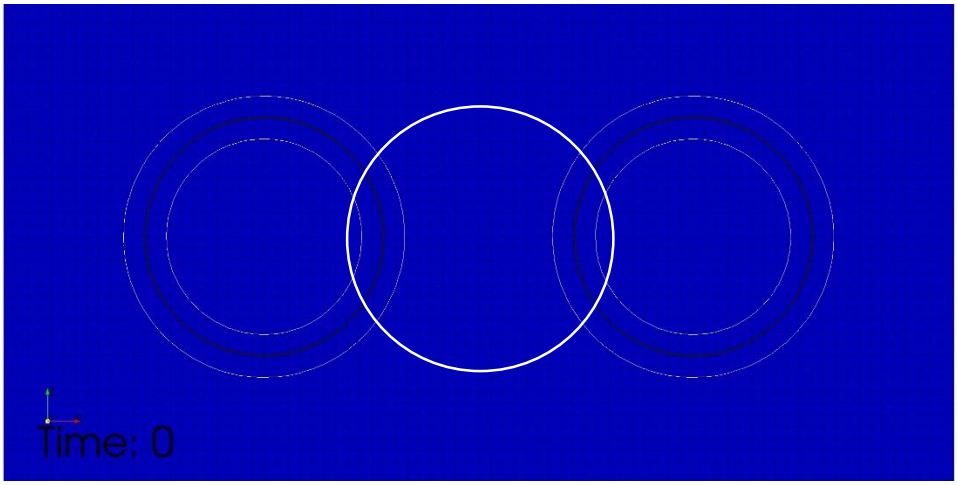


Figure 7: An initial time step with the coalescence control application volume shown as the white circle and the black circles as the bubble interfaces

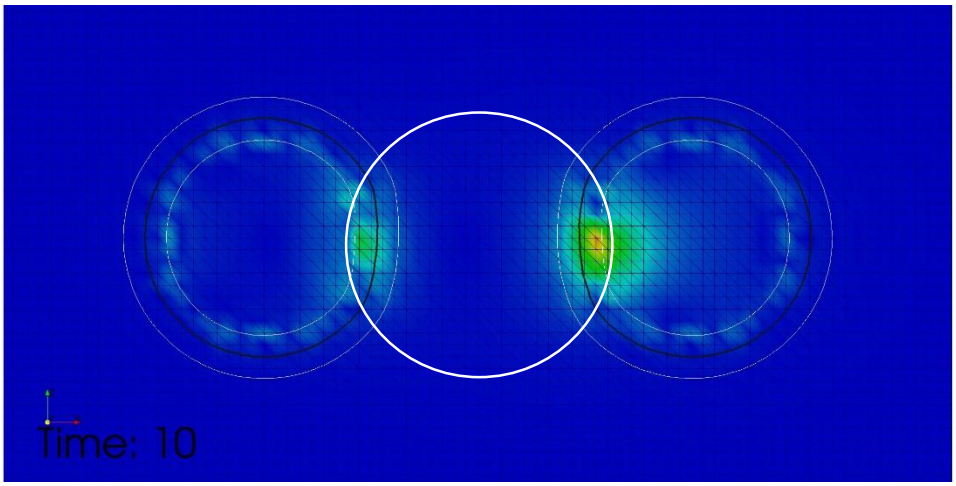


Figure 8: An later time step after surface tension changed with the coalescence control application volume shown as the white circle and the black circles as the bubble interfaces

The surface tension was changed within a sphere centered at the event location found by an identification process which is described later. The sphere uses a radius of five times the level set interface half thickness. The shape of a sphere was chosen to automatically change

the magnitude of the force based on the distance between the bubbles. This scales the force acting on the bubbles based on the surface area that is subtended by the sphere. This seems to leave the least amount of impact on the accuracy of the simulation since the properties values are only changed in a local area around the event. Also, the change only lasts as long as the bubbles are close to one another.

This force method allows for more cost efficient computation but comes with other drawbacks. One issue is that the force may not accurately represent the distance in which coalescence event take place. The initial thin liquid film distance is estimated to be 0.1 *mm* by Kirkpatrick and Lockett [16] and the final film thickness is estimated to be $1\ x\ 10^{-5}\ mm$ by Kim and Lee [11]. In comparison to the estimated initial thickness, five times the interface half thickness depends on the resolution and is only half the film distance. For a moderate resolution and 5 *mm* diameter bubble, it is equal to 4.5 *mm* which is much larger than 0.1 *mm*. This means that the slowing down of a coalescence event would begin much sooner than the physical phenomena that is being modeled. Some work has been completed in adjusting the algorithm to decrease this distance (See Section 5.2). Another problem with the model is that the algorithm adjusts the shape of the bubbles. When the surface tension is increased in the sphere, the bubble interfaces within the sphere begin to flatten. This may be physical when the liquid film is very small, however this begins to occur once the bubbles enter the force application volume. This can be an issue if it is assumed that the bubbles in the simulation are perfectly spherical and the purpose of the simulation is to observe the effects of spherical bubbles on the multi-phase flow.

3.2 Location Identification

One of the importance aspects of this algorithm is to identify the location of a coalescence event. It is necessary to know the event location to be able to prevent or simulate the coalescence properly. One simple method to find the location is to use the distance field contours generated by the LS method in the PHASTA code. By using the distance field contours, it makes it possible to identify not just one coalescence event but multiple events simultaneously. However, it is beneficial to initially consider one coalescence event because it explains some of the basic principles necessary to detect multiple coalescence events at one time.

3.2.1 Single Event

Since PHASTA uses the LS method to identify the bubble interface, there are level-set contours throughout the whole domain. In the case of more than one bubble, multiple sets of contours are spread throughout the whole domain and they will intersect with one another (See Figure 9). At these intersections, the curvature of the contours drastically changes in magnitude and sign. By choosing some of the different contours close to the bubble interface and limiting curvature values, it is possible to record the coordinates of the intersection points in an array. They are also tagged to identify the element positions in the array and that a set of coordinates has been found. These coordinates roughly fill the area between two concentric circles centered in between the approaching bubbles (See Figure 10).

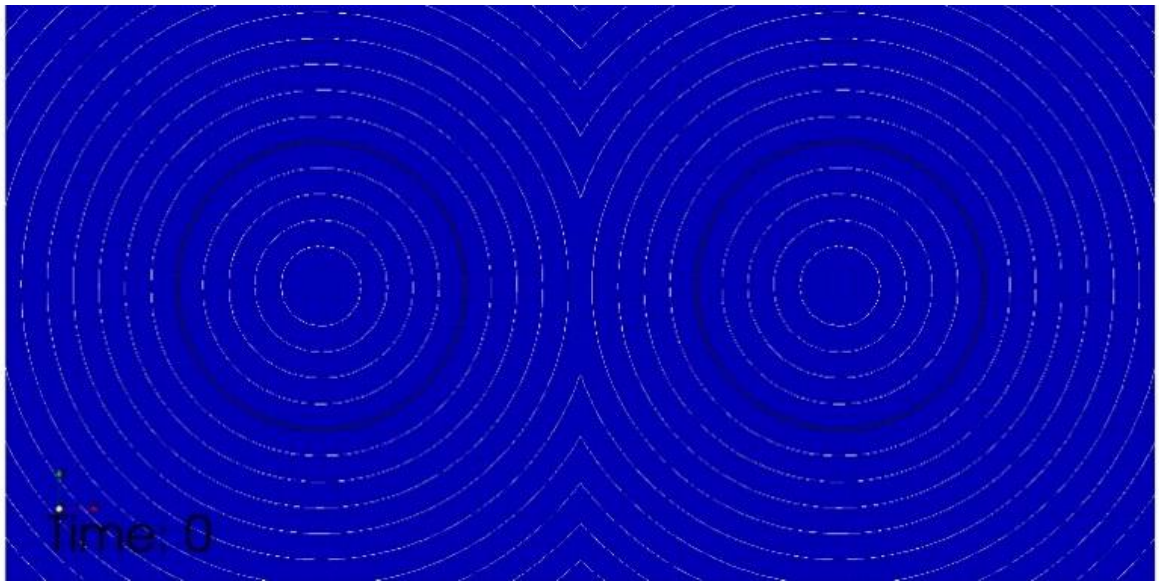


Figure 9: The distance field of the bubbles shown as the white contour lines with the black circles representing the bubble interface

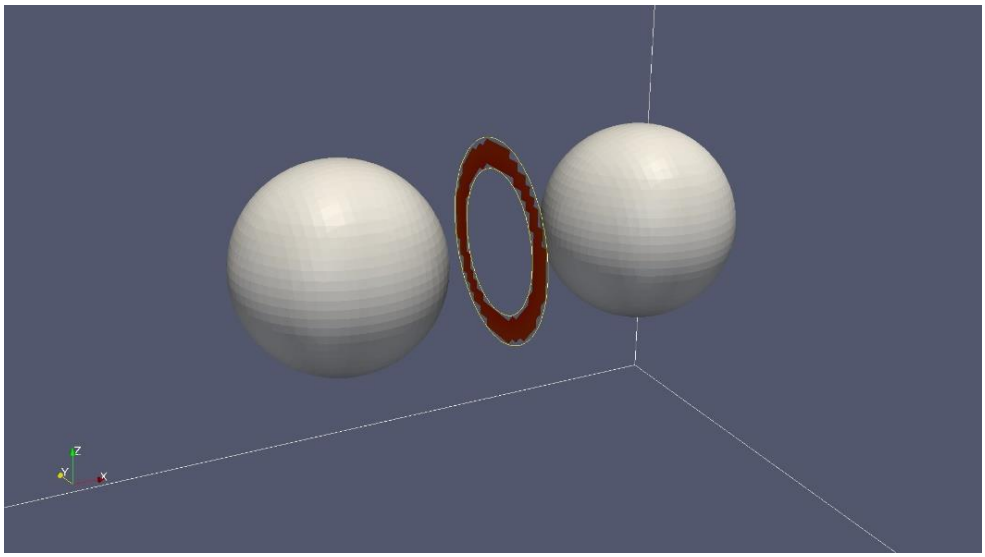


Figure 10: Location of high curvature where coordinates are used to generate average coalescence event coordinates

The current algorithm uses jumps in curvature between the first and sixth distance field contour. The limiting curvature values used to determine the coordinates has been determined empirically. A set of simple two bubble simulations, similar to the setup above, were used (See Figure 10). Each simulation in the set used a different resolution with 5 *mm* bubbles. The curvature values between the first and sixth distance field contour were recorded. A curvature value close to each jump in curvature was then taken and plotted against the number of elements across the bubble diameter (See Figure 11). A linear fit was then used to obtain an equation to calculate the limiting curvature value. As a buffer, this limiting curvature value was also multiplied by 1.45 to make sure intersecting contour coordinates were the only points tagged.

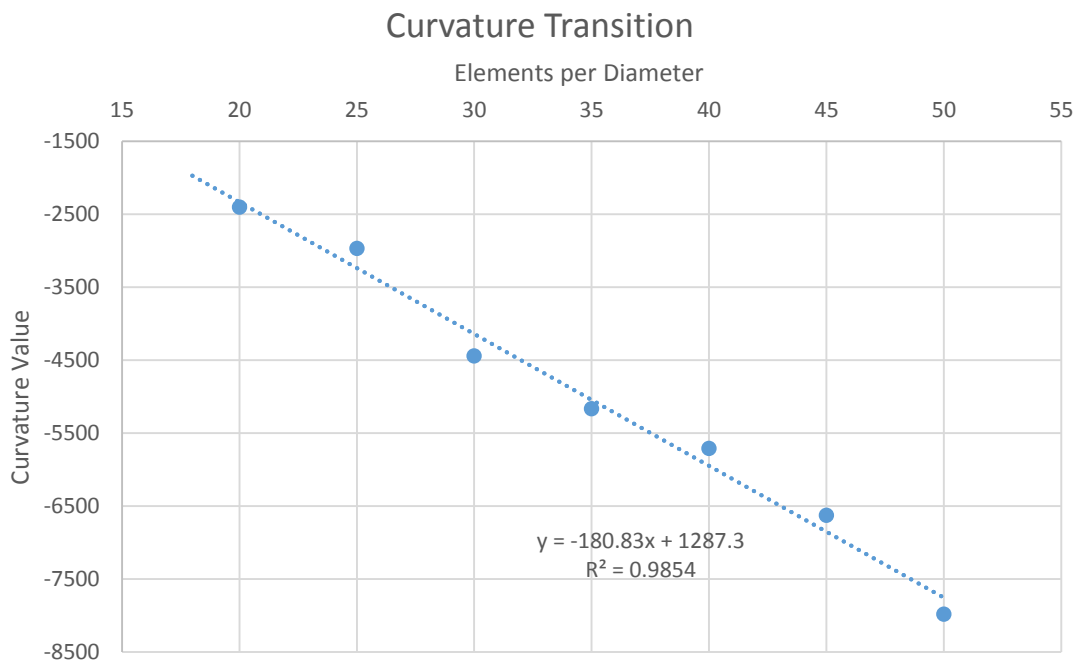


Figure 11: Plot of chosen curvature values plotted against the number of elements across a 5 mm bubble diameter. A linear fit was applied with the resulting equation and square of the correlation coefficient.

Since it is possible to run PHASTA in parallel, multiple processors will contain an array with the recorded coordinates. These coordinate arrays from each processor are consolidated into three separate arrays distinguished by axis direction. The tag array from each processor is also consolidated into a single array. Each element of the coordinate arrays are summed and averaged by using the summation of the elements in the tag array. The average coordinates are the center coordinates for the two concentric circles generated by the intersecting contours. Since it is assumed there is only one coalescence event, the center coordinates identify the location of the event and can be used as the midpoint between the two approaching bubbles. A summary of the process can be seen in the following block diagram (See Figure 12).

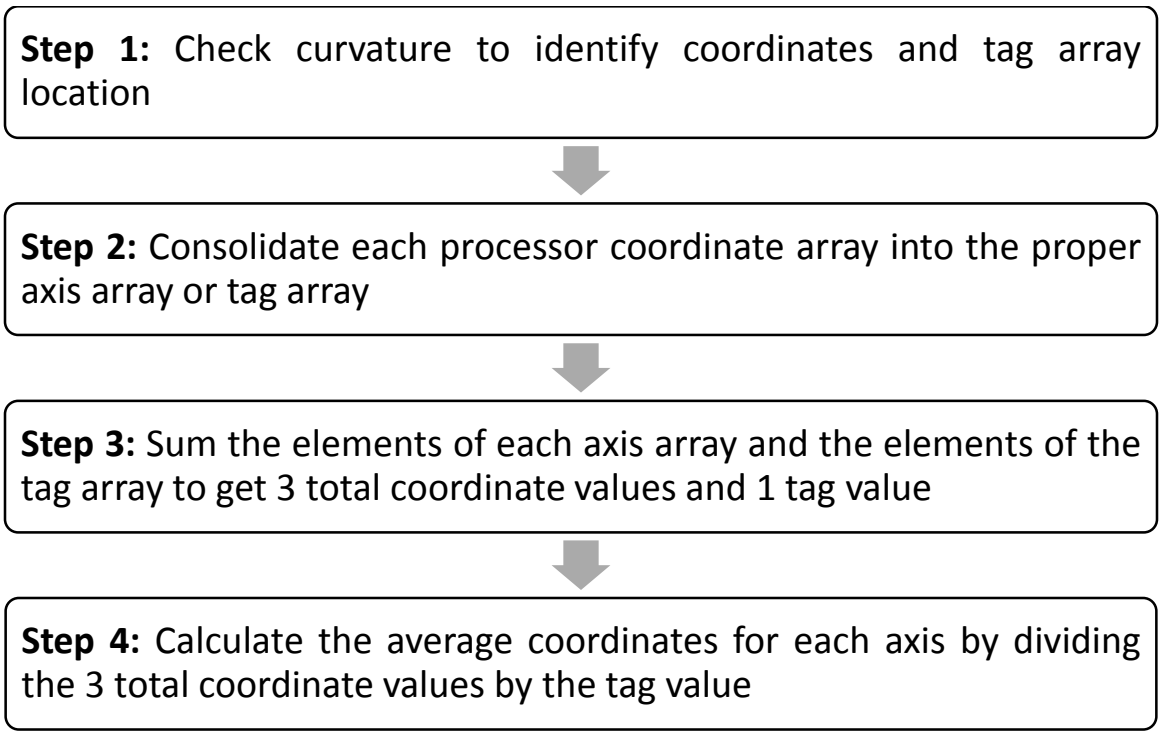


Figure 12: Summary of process to identify a coalescence event and calculate the average coordinates for the event.

3.2.2 Multiple Events

The previously described method works in the case of just one coalescence event but fails to identify multiple events occurring simultaneously. To identify more than one location at a time, an extension of the previously described method is required. A summary of the following process can be seen in the block diagram of Figure 13. To determine if there are multiple events occurring simultaneously, the average vector distance from the average coordinates to the coalescence events coordinates is calculated. If this distance is larger than the contour diameter distance, then it signifies there are multiple coalescence events in the domain.

If it has been determined that multiple events are occurring simultaneously, it is necessary to sort and tag the coalescence events specific to each event. For multiple coalescence events, the average coordinates found from all of the intersecting contour coordinates is located at a point between all of the different coalescence events. This location is not necessarily centered between all the events because the averaging process used in steps 3 and 4 do not use any weights based on the number of coordinates recorded from each event. By knowing this location and the coordinates from each of the coalescence events, it possible to generate vectors that start at this averaged location and extend to each coalescence event coordinate.

To identify each coalescence event separately, two different calculated criteria are used to consolidate the coordinates to their proper coalescence event. The first criteria used is the distance of each vector originating from the averaged location and the coalescence event coordinates which was previously calculated. For each coalescence event, the vector with the largest magnitude is found. For a coordinate to be considered part of the specific event belonging to the maximum length vector, it must be located within a distance equal to or less

than the diameter of the farthest chosen level set contour for event identification. The distance between an event coordinate and the coordinate of the maximum length vector can be calculated by using vector addition and then calculating the magnitude of the resulting vector. If the resulting vector is less than or equal to the diameter of the farthest chosen contour for the event identification, then there is a high probability that the coordinate belongs to the specific coalescence event.

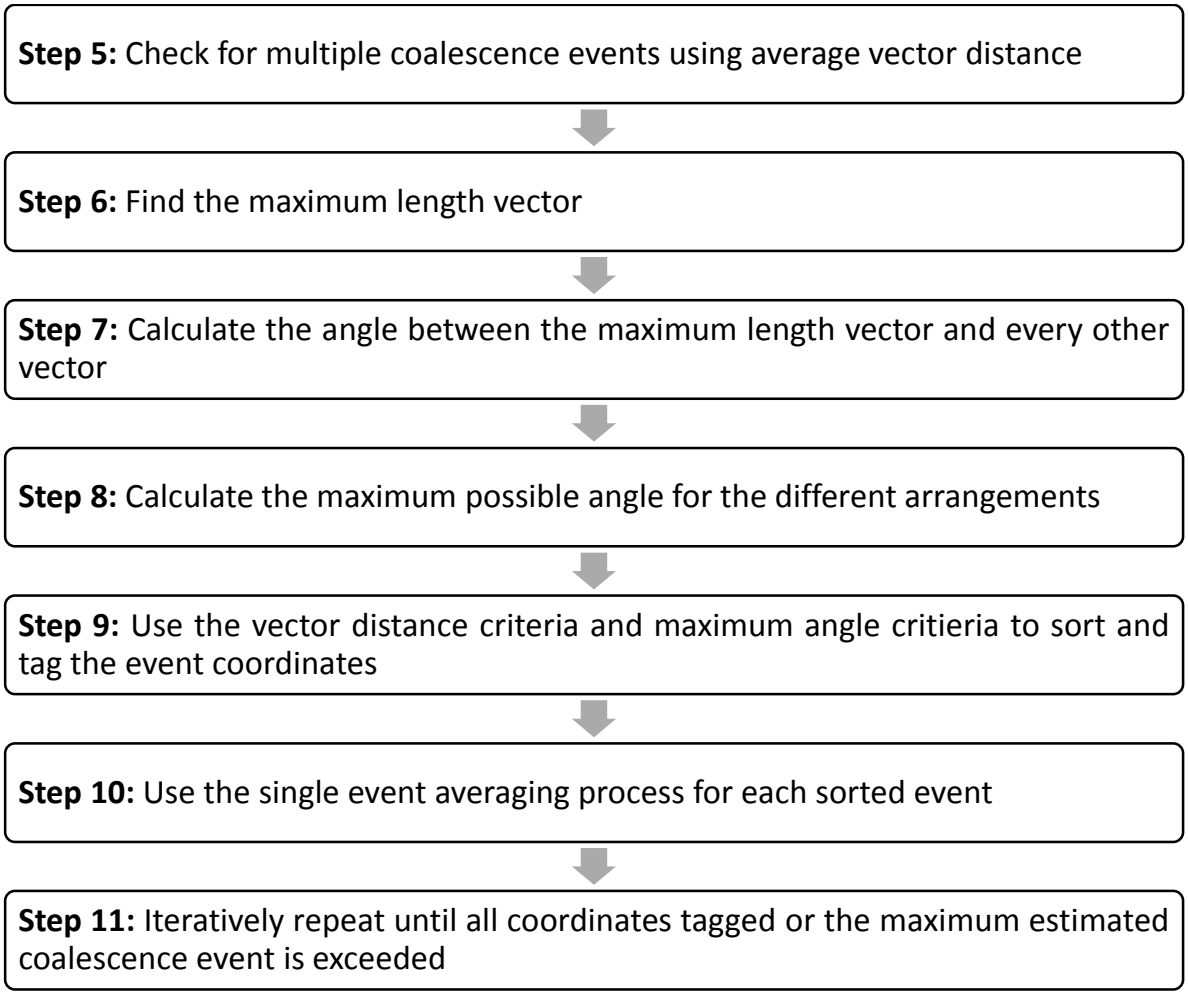


Figure 13: Summary of process to find the average coordinates for each specific coalescence event during multiple coalescence events.

Since it is not possible to assuredly determine that a coordinate belongs to a specific coalescence event using the vector comparison, the angle between each vector and the maximum length vector for a specific coalescence event is used as the second criteria. By using the farthest chosen contour diameter from which coordinates are tagged, it is possible to calculate the maximum angle that would exist between the longest vector for a specific coalescence event and another vector resulting in a triangle with the contour diameter and maximum length vector. A geometric representation of this scenario can be seen in Figure 14.

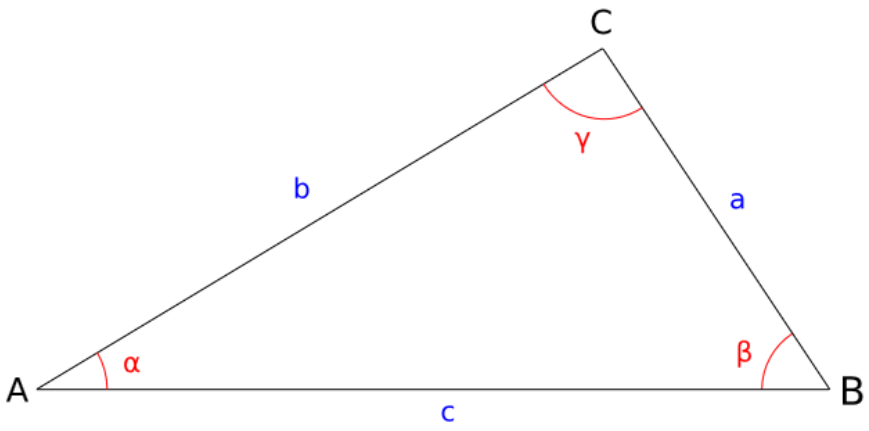


Figure 14: Triangle formed by three vectors during multi-event coalescence identification.

Since the average location is not centered between all the coalescence events, there are three separate arrangements of which side of the above triangle represents the maximum length vector, the contour diameter, and the other vector finishing the triangle. The law of cosines is

used to determine the maximum angle at which the vectors can be positioned for each arrangement (See Eqs. (10), (17)).

Arrangement 1: Sides c and a are known

The first arrangement occurs when side *c* represents the maximum length vector and side *a* represents the contour diameter. The form of the law of cosines used for this arrangement is the following:

$$a^2 = b^2 + c^2 - 2bc * \cos(\alpha) \qquad (10)$$

Since the largest possible value of α is needed as a criteria, the length of b must be determined when α is maximized. To do this, Equation (10) is solved for α and the first derivative of α is taken with respect to b (See Eqs. (11) and (12)).

$$\alpha = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2bc}\right) \qquad (11)$$

$$\frac{d\alpha}{db} = -\frac{1}{\sqrt{1 - \left(\frac{b^2 - c^2 + a^2}{2b^2c}\right)^2}}\left(\frac{b^2 - c^2 + a^2}{2b^2c}\right) \qquad (12)$$

In order to maximize α, $\frac{d\alpha}{db}$ is set equal to zero. From Equation (12), there are two possibilities that would make $\frac{d\alpha}{db}$ equal to zero:

$$-\frac{1}{\sqrt{1 - \left(\frac{b^2 - c^2 + a^2}{2b^2c}\right)^2}} = 0 \qquad (13)$$

$$\left(\frac{b^2 - c^2 + a^2}{2b^2 c}\right) = 0 \quad (14)$$

For Equation 4 to be true, b must be equal to infinity. This does not make sense for a finite triangle which means Equation 5 must be used. This equation is solved for b to calculate the length of b to maximize α:

$$b = \sqrt{c^2 - a^2} \quad (15)$$

Equation 6 is just a modified form of the Pythagorean Theorem which means that α is maximized when vectors $a, b,$ and c form a right triangle. Since the vectors form a right triangle, α can be calculated using a basic trigonometric function:

$$\alpha = \cos^{-1}\left(\frac{b}{c}\right) \quad (16)$$

Arrangement 2: Sides c and b are known

Within this arrangement, there are two possible setups. The first occurs when the maximum length vector is still acting as c but the farthest chosen contour diameter is represented by side b. For this arrangement, the angle β is maximized instead of the angle α. For this arrangement, the following form of the law of cosines is used:

$$b^2 = a^2 + c^2 - 2ac * \cos(\beta) \quad (17)$$

The same process is followed as in Arrangement 1 but $\frac{d\beta}{da}$ is found in order to maximize β. When calculated, you get a function of the same form as Equation 3. Here $\frac{d\beta}{da}$ is set equal to zero which results in the same type of possibilities as discussed in Arrangement 1. Using the second possibility, the length of side a is determined:

$$a = \sqrt{c^2 - b^2} \quad (18)$$

Again, this equation is an alternate form of the Pythagorean Theorem. This means that simple trigonometric functions can be used to solve for the maximum of angle β:

$$\beta = \sin^{-1}\left(\frac{b}{c}\right) \quad (19)$$

The second setup of this arrangement is where the contour diameter acts as side c and the maximum length vector acts as side b. In this instance, the average coordinate may be located within the sixth contour of one of the bubbles. This represents a situation where this coalescence event generates many more event coordinates than other events. In the extreme, it is possible that the average coordinates could be at the center of the coalescence event. However, it is assumed that the number of event coordinates does not significantly outweigh other events so it is assumed that the maximum angle of γ occurs when the length of a line drawn from point C in Figure 14 and perpendicular to side c is equal to the bubble radius plus one contour distance (See Figure 15).

With this assumption, it is not possible to assume that the triangle formed is a right triangle. The line drawn between point C and perpendicular to side c can be used since this forms two right triangles. This means that the angle γ and the side c can be broken into two pieces:

$$\gamma = \theta + \phi \quad (20)$$

$$c = l + m \quad (21)$$

This means that θ and ϕ can be found by the following equations:

$$\theta = \cos^{-1}\left(\frac{d}{b}\right) \quad (22)$$

$$l = b * \sin(\theta) \quad (23)$$

$$m = c - l \quad (24)$$

$$\phi = \tan^{-1}\left(\frac{m}{d}\right) \quad (25)$$

The angle γ can then be used as the maximum angle between the maximum length vector and any other vector.

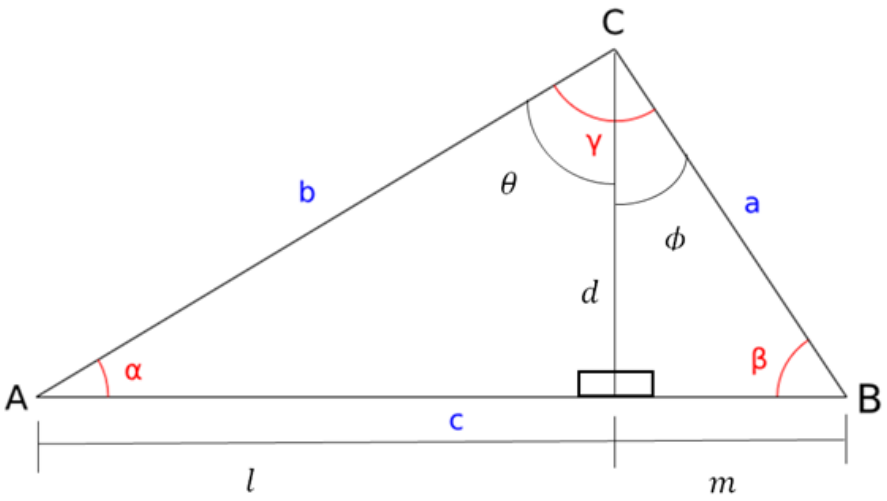


Figure 15: Schematic of triangle to calculate the angle gamma when the sides b and c are known.

Now that the maximum angles can be calculated, it is necessary to calculate the angles between the maximum length vector and every other vector in order to compare the angles as the second criteria. To calculate the angles, the definition of the dot product between two vectors is used:

$$\psi_i = \cos^{-1}\left(\frac{c \cdot v_i}{\|c\|\|v_i\|}\right) \quad (26)$$

where c is the largest length vector, v_i is any other vector, and ψ_i is the angle between the largest length vector and any other vector.

Once each of the values for the criteria have been calculated, they can be used to sort and tag the event coordinates specific to each coalescence event. This process will occur iteratively until all event coordinates have been tagged or the maximum number of estimated coalescence events has been exceeded. Once all the iterations have completed, the same averaging process used for a single coalescence event is used for each tag separately. This results in an array of center coordinates for each set of concentric circles generated by the different coalescence events and therefore identifies each event separately.

3.3 Drainage Time

Once the coalescence event has been identified and the force is implemented, it is necessary to track the amount of time the force has been active. The tracked time can then be compared to a coalescence time model [7] that estimates the total coalescence time. Once the tracked time has exceeded the estimated total coalescence time, the force is removed, signifying the liquid film has had sufficient time to drain, and the bubbles are allowed to coalesce.

The drainage time uses a model that is based on the equivalent bubble radii (r_{ij}), the density of the liquid (ρ_l), the surface tension (σ), and the initial and final film thickness (h_o, h_f):

$$t_{ij} = \sqrt{\frac{r_{ij}^3 \rho_l}{16\sigma}} \ln\left(\frac{h_o}{h_f}\right) \quad (27)$$

The equivalent radius is found by the following equation:

$$r_{ij} = \frac{1}{2}\left(\frac{1}{r_{bi}} + \frac{1}{r_{bj}}\right)^{-1} \quad (28)$$

where the indices i, j represent different bubbles. The initial film thickness is estimated as $1 * 10^{-4}\ m$ by Kirkpatrick and Lockett [16] and the final film thickness is usually taken to be $1 * 10^{-8}\ m$ [11].

To track the coalescence events, it is necessary to develop several logic loops in order to: identify new coalescence events, keep track of the continuing events that have and have not exceeded the model time limit, and determining when events have ended whether through coalescence or the bubbles moving away from (bouncing off) one another. Since each event is not assigned the same event number after each iteration, the distance between the average coordinates from the previous iteration and the current iteration are used to identify which events are the same. The distance between the average coordinates will always be quite small. The default values for the average coordinates are used to help identify when any new and ending events occur.

CHAPTER 4 SIMULATIONS

In order to test the functionality and capability of the algorithm and its cost, three different types of simulations were used and performed. Each of these simulations was performed using the PHASTA code with the coalescence control algorithm added in as a functional capability.

4.1 Two Bubble Coalescence Prevention

The first simulation consists of how the algorithm handles when there is only one coalescence event occurring in the domain. The time tracking portion of the algorithm was not tested in this simulation. The domain size was 40 *mm* x 20 *mm* x 20 *mm* with the x-y planes acting as walls. There were three bubbles placed in the domain: (i,ii) two bubbles with a 5 *mm* diameter and (iii) a third with a 6 *mm* diameter. The bubbles were placed at varied distances in the x-direction along the center line between two parallel plates. Gravity was used to provide a buoyance force on the bubbles to cause them to flow while a pressure gradient was applied to the liquid to keep it stationary within the domain. A finite element mesh containing 2M hexahedral elements. This means that the 5 mm diameter bubbles were resolved with at least 25 elements across its diameter. For more information on the domain and simulation parameters, see Figure 16 and Table 1.

The simulation was used for two tests. The first test was performed without using the coalescence control algorithm. The second test was performed with the coalescence control active. To analyze how the algorithm handles a single event, only the interaction between the two 5 mm diameter bubbles is considered.

In the first test, a coalescence event was observed in iteration 870 between the two 5 *mm* diameter bubbles (See Figure 17 Top). When the second test was performed, the coalescence control algorithm identified the event and locally adjusted the surface tension around the center of the event coordinates. Comparing the visualization of the same simulation time as in the first test, the coalescence of the two 5 *mm* bubbles was prevented (See Figure 17 Bottom).

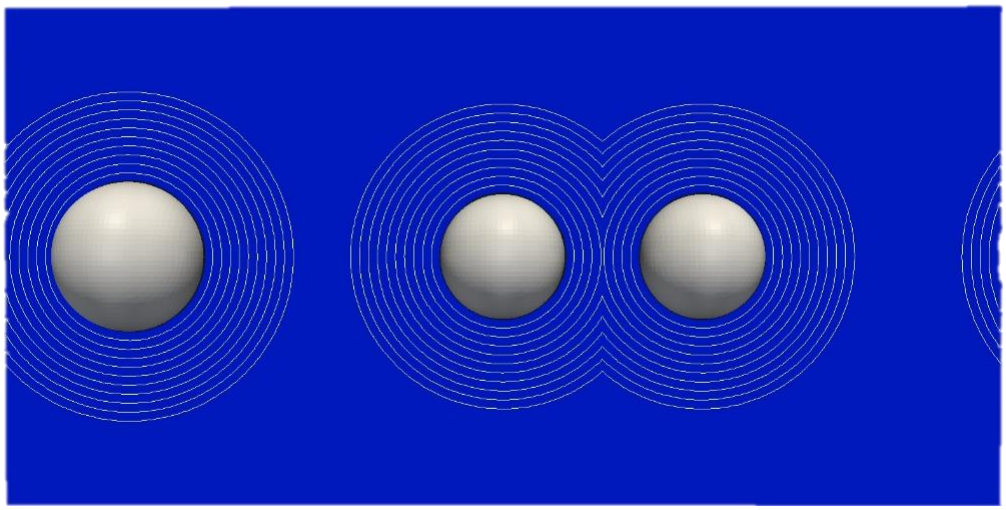


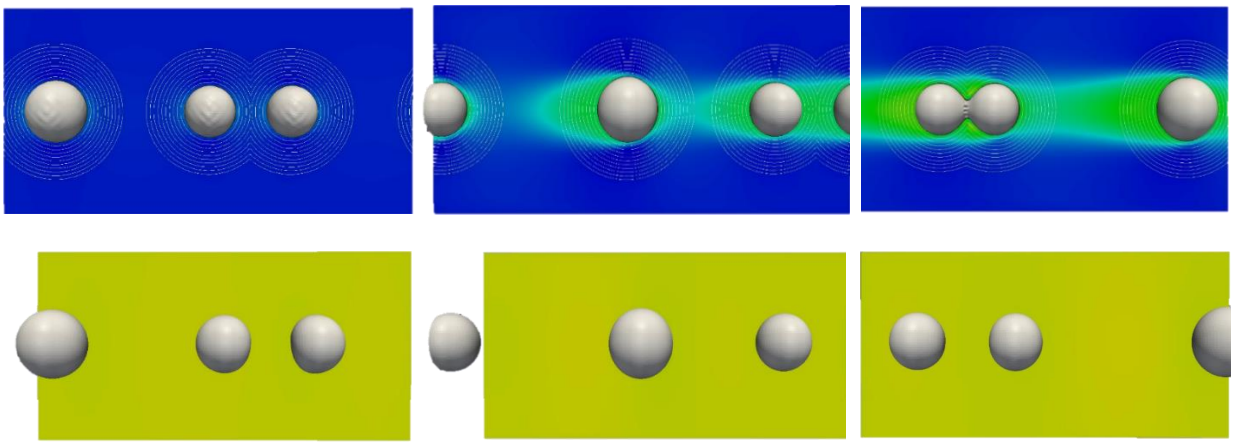
Figure 16: Initial setup to test the algorithm for a single coalescence event identification and prevention.

Table 1: Overview of simulation parameters for the single coalescence prevention simulation.

Parameter	
Liquid	water
Gas	air
Pressure, Pa	0
Temperature, °C	N/A (0)
Liquid Density, kg/m^3	997.17
Gas Density, kg/m^3	1.161
Liquid viscosity, kg/m-s	2.67E-03
Gas viscosity, kg/m-s	1.86E-02
Body Force, m/s^2	0.5
Body Force Pressure Gradient, Pa/m	498.585

As for computational cost, the simulation without the coalescence control finished 710 iterations in about 2 hours. When the coalescence control was activated, the simulation finished 710 iteration in about 2.2 hours. This shows a 10% increase in cost of running with the

coalescence control. Even though the coalescence control increases the computational cost, this increase is cost is manageable and willing trade off to the resolution required for coalescence viscosity effects to be observed.



(a) *(b)* *(c)*

Figure 17: Visualization of the simulations for the single coalescence events at iterations: (a) 20, (b) 400, and (c) 870. Top: The simulation performed without the coalescence control algorithm. In iteration 870, the 5 mm bubbles have begun to coalesce. Bottom: The simulation performed with the coalescence control algorithm. In iteration 870, the coalescence event has been prevented.

4.2 Multi-Bubble Coalescence Prevention

In order to test the coalescence control algorithm's ability to handle multiple events as well as turbulence, a large simulation was created. This simulation is based on one reported on by Bolotnov and Podowski in 2012 [42]. The simulation consists of a bubbly gas flow in turbulent flow conditions with an equivalent $Re \approx 12{,}000$. The flow is between two parallel places and a total of 32 bubbles were placed randomly throughout the domain. The mesh consisted of about 10 million hexahedral elements resulting in approximately 18 elements

across the diameter of the bubbles. For more information on the simulation parameters and the domain, see Figure 18 and Table 2.

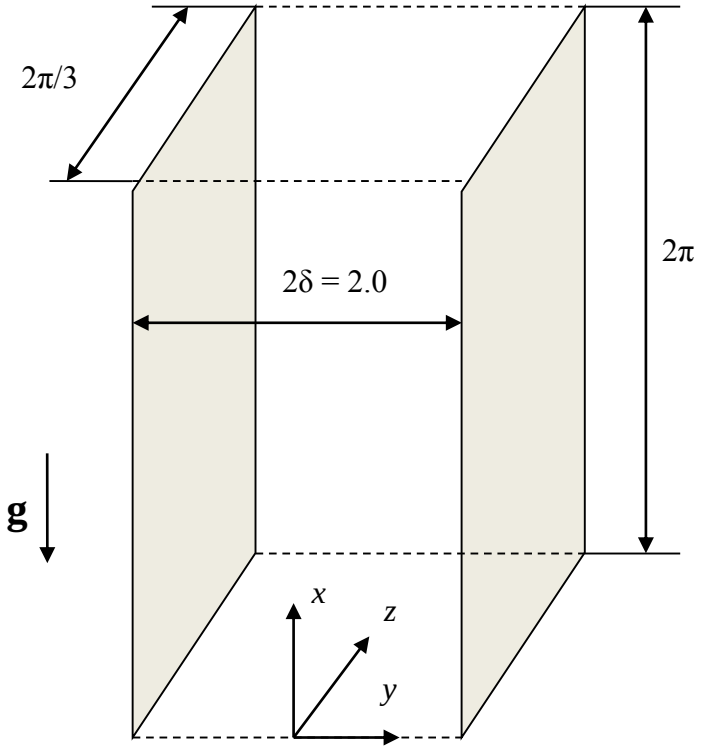


Figure 18: Overview of the simulation domain dimensions and axis orientation. The shaded planes represent walls.

The described simulation was tested twice: one without using the coalescence control algorithm and one with the coalescence control algorithm active. The starting point for these tests started at iteration 7800 (See Figure 19). To determine if bubble coalescence events have occurred, later iterations were chosen and the number of bubbles was counted. The iterations that were chosen were matched based on time within the simulation. The times

Table 2: Overview of simulation parameters for the multi-bubble coalescence prevention simulation.

Parameter	
Pressure, bar	10
Liquid	water
Gas	air
Temperature, °C	27
Liquid Density, kg/m^3	996.5
Gas Density, kg/m^3	11.636
Liquid viscosity, kg/m-s	$8.514 \cdot 10^{-4}$
Gas viscosity, kg/m-s	$1.858 \cdot 10^{-5}$

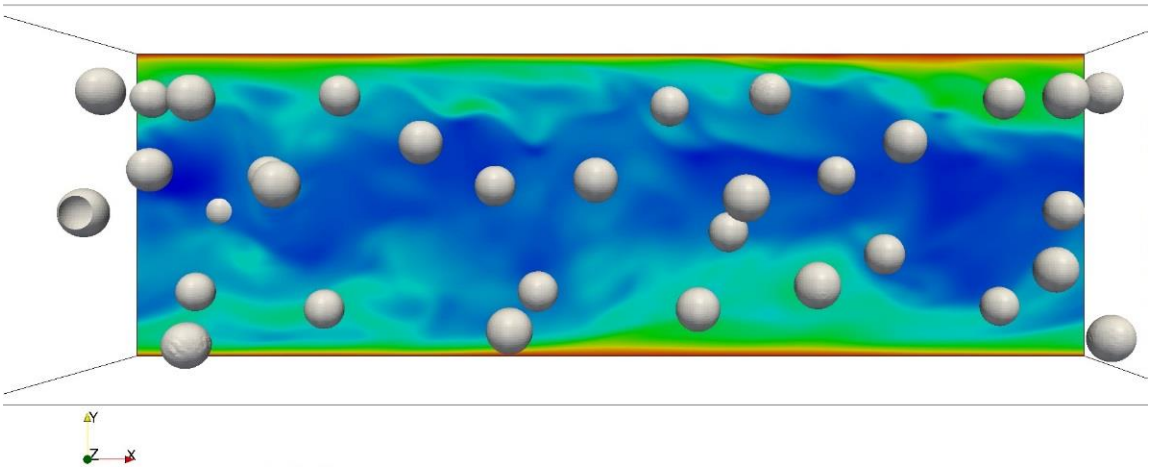
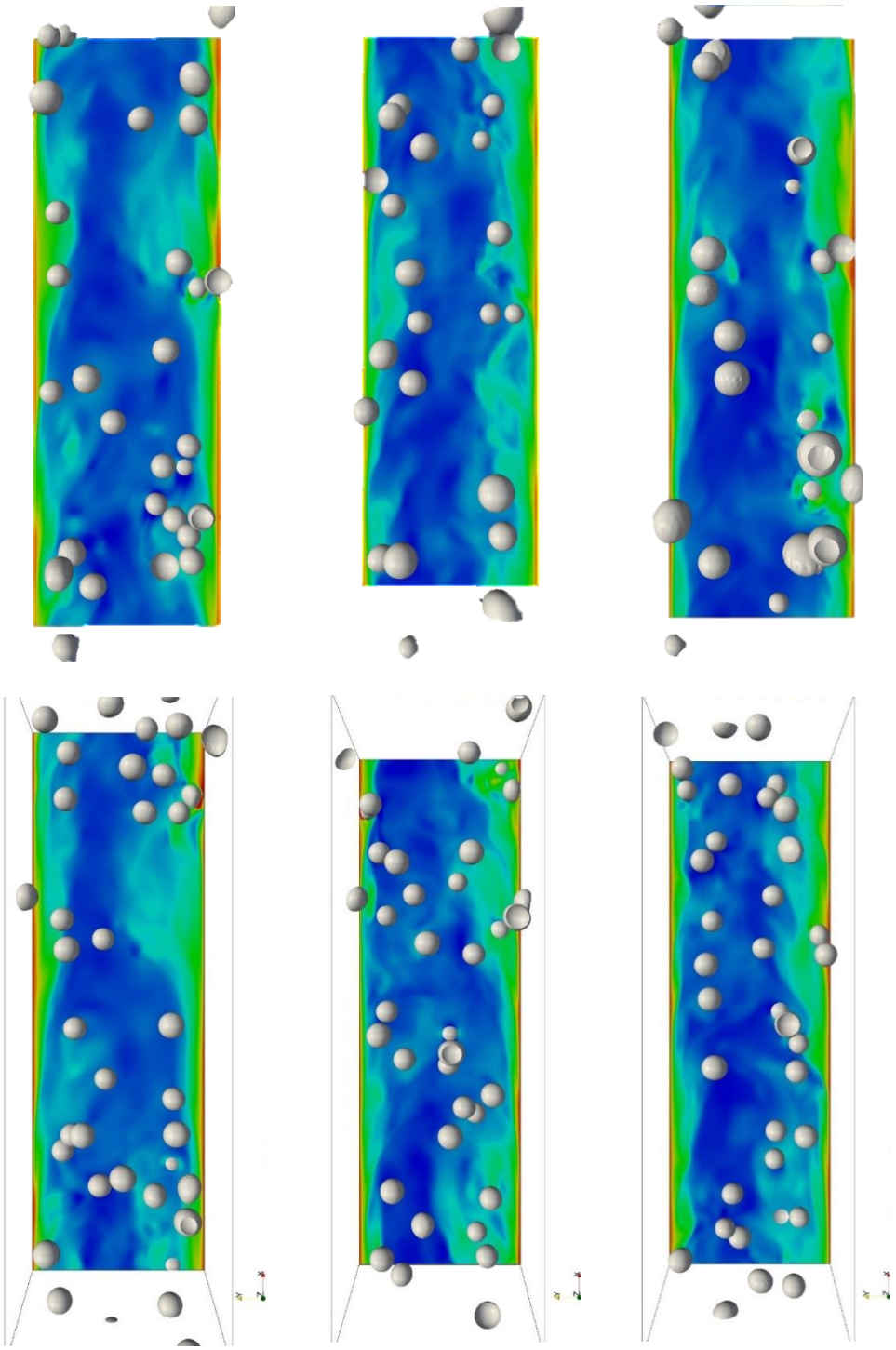


Figure 19: Initial setup at iteration 7800 of the bubbly flow in turbulent conditions. There are 32 bubbles randomly placed throughout the domain.

chosen represent two, four, and five full flow-throughs of the domain where a single flow-through means a bubble has moved through the domain to return to its initial position on the x-axis. For the test when the coalescence control was not used, it can be seen in iteration 15600, only 25 bubbles remained in the domain. Then in iteration 23800, only 22 bubbles remained,

and in iteration 27800, only 19 bubbles remained. (See Figure 20 Top). This signifies that there were multiple coalescence events after only two flow-throughs. More events continued to follow in subsequent flow-throughs. For the second test using the coalescence control algorithm, in iterations 15000, 22600, and 26800, 32 bubbles remained in the domain instead of 25, 22, or 19 bubbles. (See Figure 20 Bottom). This shows that 13 coalescence events that occurred in the first test were prevented by the coalescence control algorithm. This means that the algorithm is performing properly for turbulent conditions with multiple bubbles. The computational cost for these simulations was also analyzed. The simulation without the coalescence control finished 2150 iterations in 3 hours. When the coalescence control was activated, the simulation finished 2150 iterations in 3.8 hours. This shows a 25% increase in cost of running with the coalescence control. This computational cost is higher than the single coalescence event but is to be an expected increase. However, even this increase is cost is more desirable over the resolution required for coalescence viscosity effects to be observed.



(a) *(b)* *(c)*

Figure 20: Visualization of both 32 bubble simulation at multiple iterations. Top: No coalescence control, Iterations (a) 15600, (b) 23800, (c) 27800. Bottom: Coalescence control active, Iterations (a) 15000, (b) 22600, (c) 26800.

4.3 Coalescence Time Tracking

The third test simulation was designed to test the coalescence control algorithm's time tracking model. This simulation design consists of a single bubble below a free surface of gas placed at the top of the domain. The domain size was 20 *mm* x 35 *mm* x 20 *mm*. The bubble diameter was 5 *mm* and the center point of the bubble was placed 20 *mm* below the free surface (See Figure 21). Gravity was used to provide a buoyance force to force the bubble to approach the free surface. A wall was placed on the top and bottom of the domain normal to the gravity and buoyance force. The mesh was made of 0.9M hexahedral elements resulting in approximately 20 elements across the diameter of the bubble. For more information on simulation parameters, see Table 3.

Table 3: Overview of simulation parameters for the time tracking tests.

Parameter	
Liquid	water
Gas	air
Pressure, Pa	0
Temperature, °C	N/A (0)
Liquid Density, kg/m^3	997.17
Gas Density, kg/m^3	1.161
Liquid viscosity, kg/m-s	2.67E-03
Gas viscosity, kg/m-s	1.86E-02
Body Force, m/s²	1
Body Force Pressure Gradient, Pa/m	997.17

Two simulations were performed for this test (See Figure 22). Both tests were performed with the coalescence control portion of the code active. However, in the second simulation, the

time tracking portion of the coalescence control was active. The first test was performed without using the time tracking portion of the code. It can be seen in iteration 800 that the force is implemented. Later, in iteration 1150, the simulation shows that the coalescence event has

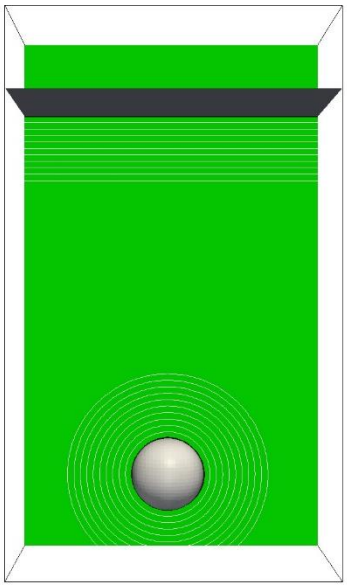


Figure 21: Initial setup for the bubble rising towards a free surface. The simulation was used to test the time tracking portion of the algorithm.

been prevented (See Figure 22 Top). For the test when the time tracking portion of the code was active, it can be seen that the force is implemented in the same iteration (800) as the first test. However, by comparing iteration 920 between the simulations, it can be seen that the force is removed in the second simulation. By iteration 1150, the bubble and free surface have coalesced in the second simulation (See Figure 22 Bottom). The second simulation demonstrates that the time tracking portion of the code is properly removing the application volume after the maximum coalescence time has been exceeded.

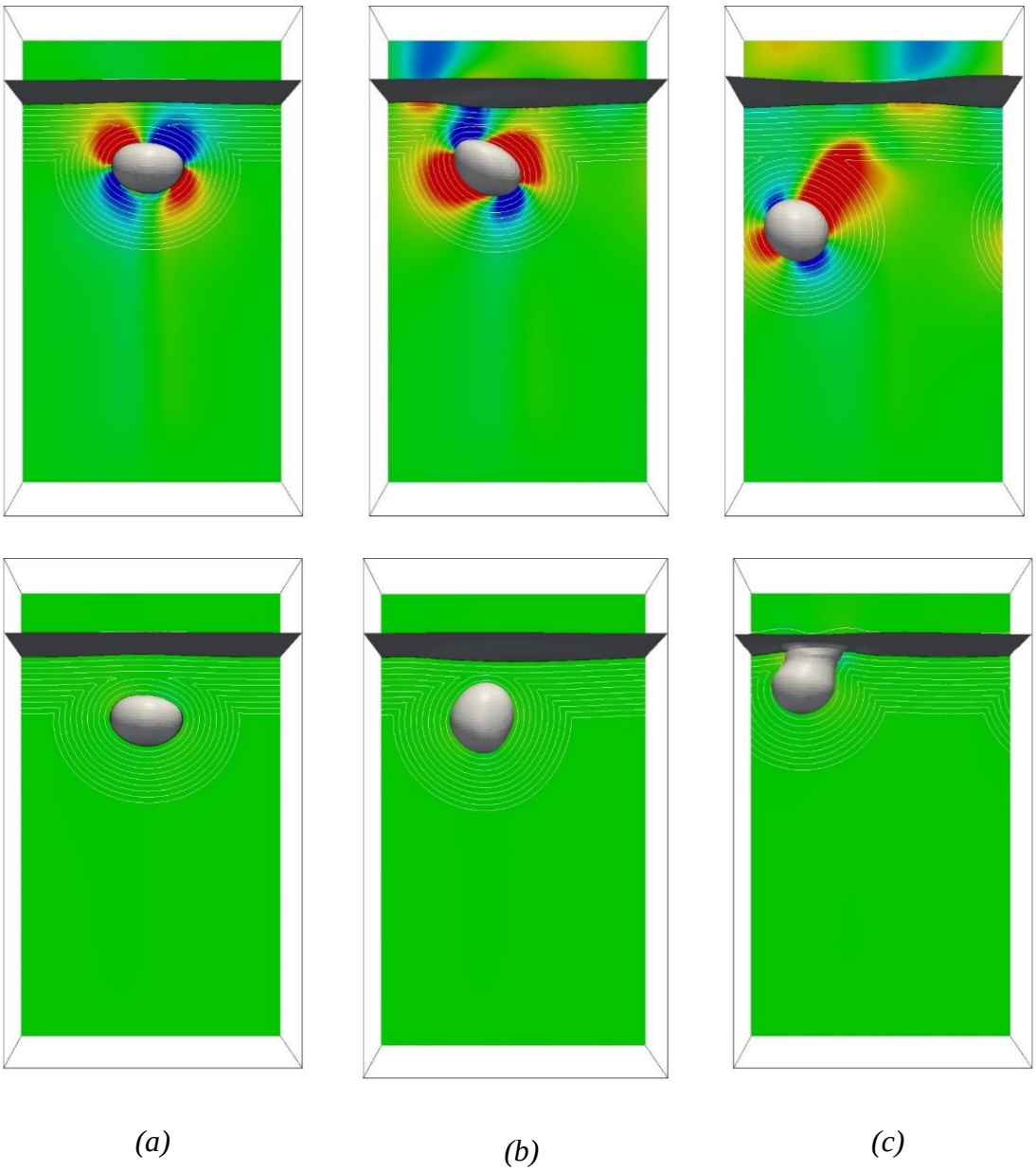


Figure 22: Visualization of both simulations at iterations (a) 800, (b) 920, and (c) 1150. Top: Simulation run without using the time tracking portion of the algorithm. Bottom: Simulation that uses the time tracking portion of the algorithm.

A new capability was used in these simulations so the computational cost changed from the previous simulations. Without the time tracking algorithm active, the simulation finished 1500 iterations in 2.33 hours. When the tracking algorithm was active, the simulation finished 1500 iterations in 2.36 hours. This signifies only a 1% increase in the running cost of adding the time tracking portion of the algorithm. This running cost increase is insignificant to the cost of running the rest of the coalescence control algorithm and is therefore an easy tradeoff.

CHAPTER 5 VERIFICATION AND VALIDATION

5.1 Mesh Study

To test whether the algorithm was calculating the average coordinates correctly irrespective of the mesh, a simple domain of 20mm x 15mm x 15mm was created with two 5 *mm* diameter bubbles (See Figure 23). Only gravity in the negative x-direction was used to generate a buoyancy force to move the bubbles. The distance between the bubble centers was small enough to activate the coalescence control algorithm. Several different cases were run where the mesh resolution was increased from 20 elements per diameter (288K discrete elements and 1.8M parasolid elements) to 40 elements across diameter (2.3M discrete elements and 13.3M parasolid elements). The distance between the bubble centers was kept constant. The value of the interface half-thickness input into PHASTA was also unchanged with the increasing resolution. Two separate simulations were created for each case: one mesh contained discrete hexagonal elements while the other contained adaptive tetrahedral elements. The same parameters were used for both simulations (See Table 4).

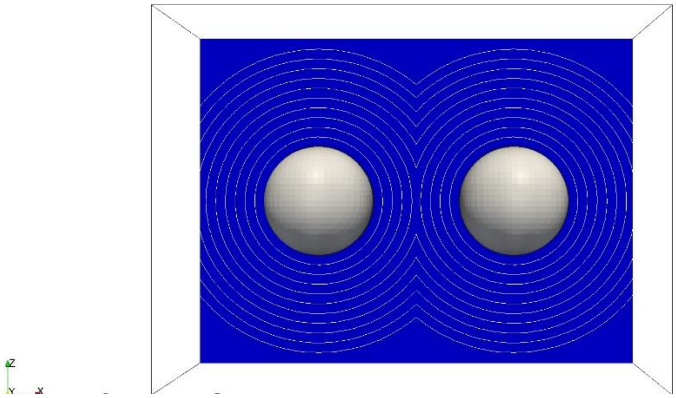


Figure 23: Initial setup of the simulation used in the mesh study. The domain contains two 5 mm bubble.

Table 4: Overview of simulation parameters for both mesh types.

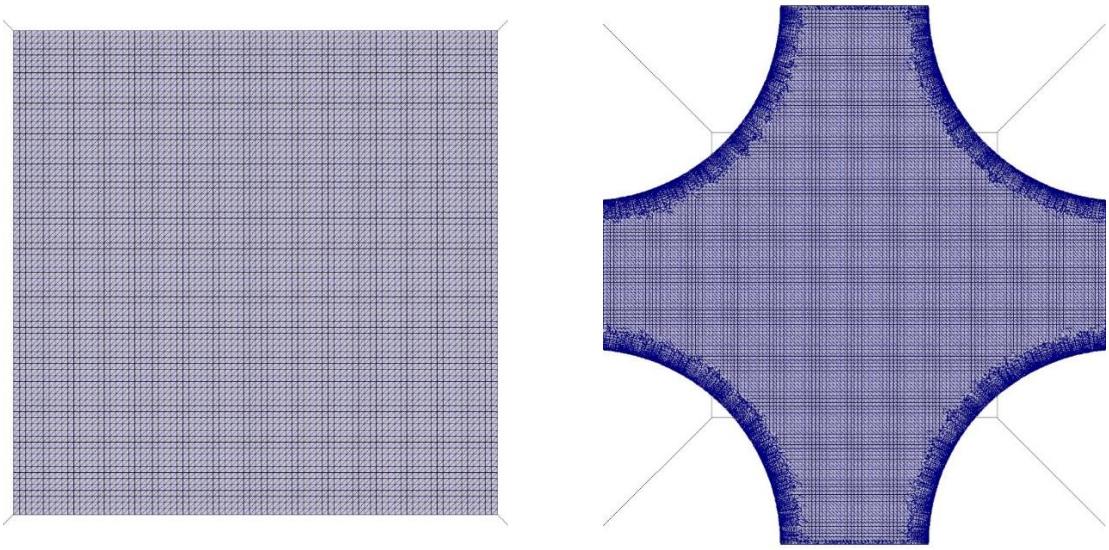
Parameter	
Liquid	water
Gas	air
Pressure, Pa	0
Temperature, °C	N/A (0)
Liquid Density, kg/m^3	997.17
Gas Density, kg/m^3	1.161
Liquid viscosity, kg/m-s	2.67E-03
Gas viscosity, kg/m-s	1.86E-02
Body Force, m/s^2	0.5
Body Force Pressure Gradient, Pa/m	498.585

The reason to use two different mesh types comes from the fact that each mesh type is used for different situations. The discrete mesh uses hexagonal elements within a structured grid (See Figure 24 *(a)*). This grid is simple and efficient grid but unable to represent shapes other than a cube. On the other hand, a parasolid mesh uses tetrahedral elements in an unstructured mesh (See Figure 24 *(b)*). This allows non-cube shapes to be simulated but may result in more

complex and less efficient grid. The coalescence control algorithm will be used in both meshes and therefore a mesh study must be performed using both meshes.

5.1.1 Discrete Mesh

For this test, the interface half-thickness was kept at $4.5 * 10^{-4}$ which is equal to 1.8 times a 20 point per diameter resolution. The bubbles in the diameter were setup such that the midpoint between them was set at (1.0E-2, 7.5E-3, 7.5E-3). Each simulation was run for five iterations and the coalescence event coordinates reported were taken from the third iteration.



(a) *(b)*

Figure 24: Visualization of a (a) discrete mesh and a (b) parasolid mesh.

The third iteration was chosen to allow the algorithm to perform a couple of iterations while not waiting too long such that the bubbles moved sufficiently to drastically move the coalescence event location. These coalescence event coordinates can be seen in Table 5. Each set of coordinates was compared to the exact midpoint value and the normalized based on the distance between the center points of the bubble (See Table 6).

Table 5: Contains the average coordinates for different discrete domain resolutions. It also contains the exact geometric coordinates and distance between bubble centers.

Reported Center Points for Coalescence Events in Discrete Simulation							
3rd Iteration Coordinates						Exact	Bubble Center Distance
	20 Pts.	25 Pts	30 Pts.	35 Pts.	40 Pts.		
X Coord	1.01E-02	1.01E-02	1.00E-02	1.00E-02	1.00E-02	1.00E-02	9.00E-03
Y Coord	7.49E-03	8.08E-03	8.32E-03	7.89E-03	8.21E-03	7.50E-03	9.00E-03
Z Coord	7.49E-03	7.11E-03	6.85E-03	7.07E-03	7.38E-03	7.50E-03	9.00E-03

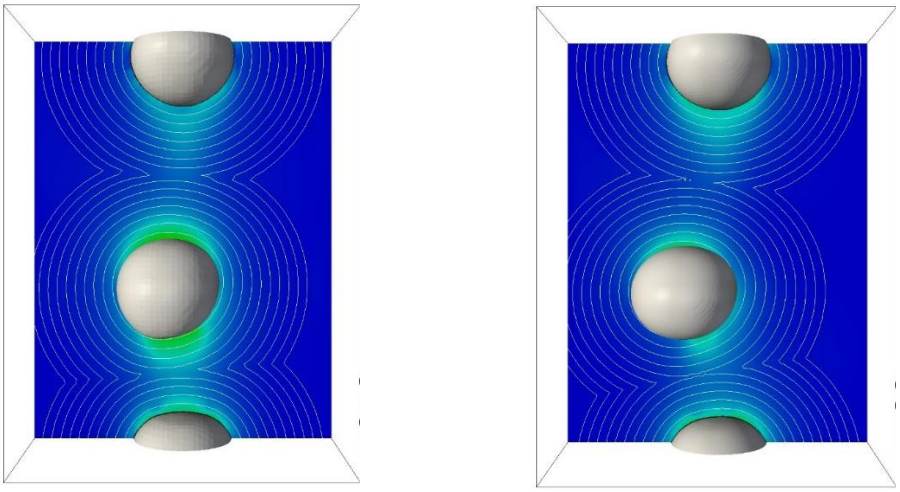
Table 6: Contains the error between the reported coordinates and the exact geometric coordinates normalized by the distance between bubble centers.

Percent Error Normalized by Distance between Bubble Centers					
	20 Pts.	25 Pts	30 Pts.	35 Pts.	40 Pts.
X Coord	1.006611539	0.67252167	0.409646053	0.510499203	0.451368364
Y Coord	0.081854474	6.452301016	9.107695555	4.34685935	7.86000409
Z Coord	0.086528576	4.298314523	7.271514561	4.79368714	1.344575277

Among all the cases, the highest error was seen to be about 9%. The largest errors were also only seen for the y and z coordinates. A possible explanation for this result can be found when considering that the intersecting contours of the two bubbles are found in the y-z plane. It is possible that the concentration of points found around the intersecting contours was not evenly

distributed. This would cause a skew in the average y and z coordinates without affecting the x coordinate.

To determine whether the errors affect the outcome in later iterations, the 20 elements and 40 elements across the 5 *mm* bubble diameter simulations were run longer. These simulations were performed for 500 iterations each. The 250th iteration was chosen for the 20 element resolution since the simulation matched the 500th iteration of the 40 element resolution. As expected for the 20 elements per diameter resolution, the coalescence event was prevented (See Figure 25 *(a)*). It can also be seen in Figure 25 *(b*) that the coalescence event was prevented in the 40 element resolution as well. This shows that the errors in the higher resolution are insufficient to cause the algorithm to work improperly.



(a) *(b)*

Figure 25: Visualization of the discrete mesh study for the 20 element and 40 element resolutions. Left: The 20 element resolution at iteration 250. Right: The 40 element resolution at iteration 500.

5.1.2 Parasolid Mesh

For this test, the interface half-thickness was kept at 3.0×10^{-4} which is equal to 1.8 times a 30 point per diameter resolution. The bubbles in the diameter were setup such that the midpoint between them was set at (1.0E-2, 0, 0). Again, each simulation was run for five iterations and the coalescence event coordinates reported were taken from the third iteration. These coalescence event coordinates can be seen in Table 7. Each set of coordinates was compared to the exact midpoint value and normalized based on the bubble center distance (See Table 8).

Table 7: Contains the average coordinates for different parasolid domain resolutions. It also contains the exact geometric coordinates and distance between bubble centers.

Reported Center Points for Coalescence Events (30 Pts. Eps.)							
3rd Iteration Coordinates						Exact	Bubble Center Distance
	20 Pts.	25 Pts	30 Pts.	35 Pts.	40 Pts.		
X Coord	1.01E-02	1.00E-02	1.00E-02	1.01E-02	1.01E-02	1.00E-02	7.20E-03
Y Coord	5.87E-05	4.23E-05	4.50E-05	3.08E-04	6.15E-04	0.00E+00	7.20E-03
Z Coord	1.18E-05	-1.69E-08	-4.81E-05	7.92E-04	1.21E-03	0.00E+00	7.20E-03

Table 8: Contains the error between the reported coordinates and the exact geometric coordinates normalized by the distance between bubble centers.

Percent Error Normalized by Distance between Bubble Centers					
	20 Pts.	25 Pts	30 Pts.	35 Pts.	40 Pts.
X Coord	1.342338337	0.390327151	0.554685182	0.934134495	0.843749435
Y Coord	0.815865397	0.587218111	0.625412608	4.271283867	8.547559288
Z Coord	0.163391908	0.000234887	0.667659593	11.00519116	16.77109136

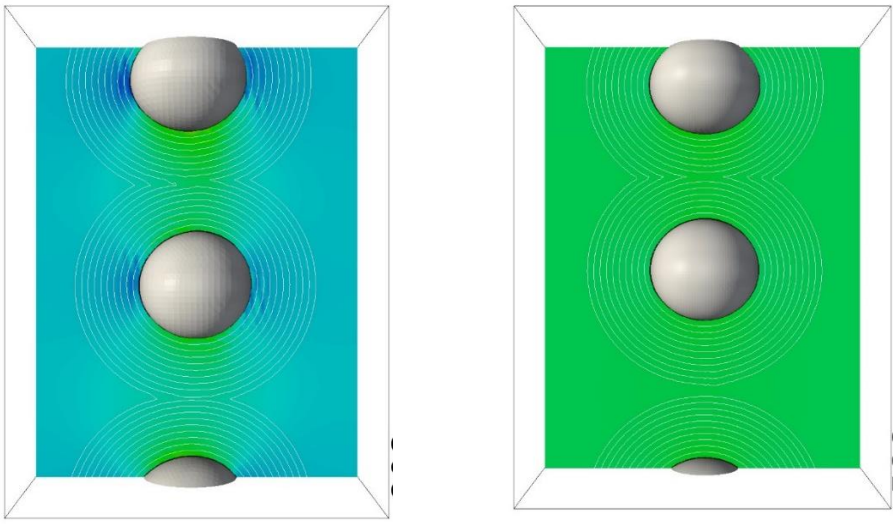
Among all the cases, the highest error was seen to be about 16%. The largest error were only seen in the y and z coordinates. This result shows high accuracy with the expected results. The higher error in the y and z coordinates can be explained by the same reasoning in the discrete domain. The y and z average coordinates are generated from intersecting contours in the y-z plane. It is possible that the concentration of points found around the intersecting contours was not evenly distributed. This would cause a skew in the average y and z coordinates without affecting the x coordinate.

To determine whether the errors affect the outcome in later iterations, the same process was used as with the discrete mesh. The two simulations of 20 elements and 40 elements across the 5 *mm* bubble diameter were performed for 500 iterations each. Visualizations of each case was chosen based on matching time within the simulation (See Figure 26). For these two simulations, the interface half-thickness was not a proper match. However, for both simulations, the coalescence event was still prevented (See Figure 26). This shows that the errors due to a mismatched interface half-thickness are insufficient to make the algorithm work improperly.

5.2 Minimum Liquid Film Thickness during Coalescence Control

As mentioned earlier, the initial distance estimated by Kickpatrick and Lockett [16] was 0.1 *mm*. However, in our moderate resolution simulations, the initial distance is ten times the interface half-thickness which is equal to 4.5 *mm*. This value is much larger than desired and means the coalescence control process will start sooner than expected. To see if this distance could be reduced, a simple test was performed using the same simulation of the bubble rising

towards a free surface simulation (See Section 4.3). The same domain and simulation parameters were used (See Figure 21 and Table 3).



(a) *(b)*

Figure 26: Visualization of the parasolid mesh study for the 20 element and 40 element resolution. (a) The 20 element resolution at iteration 280. (b) The 40 element resolution at iteration 500.

In the original algorithm, the identification of a coalescence event starts once the sixth distance field contours begin to intersect (See Figure 9). This means that the application volume is implemented before the bubbles begin to enter the volume. However, to reduce the initial coalescence distance, there are two options: increase the resolution or change the distance field contour that activates coalescence control. Since the purpose of the coalescence control algorithm is to reduce the required resolution to prevent coalescence, changing the distance field contour was investigated. For this test, three simulations were performed: (i) using the third distance field contour, (ii) using the fourth distance field contour, and (iii) using

the original sixth distance. In order to be successful, the algorithm using closer contours still needed to prevent the coalescence event. The time application portion of the algorithm was not used for these tests since it was necessary to prevent the coalescence event for the test.

The simulation setup and parameters were the same as those used in the coalescence time tracking section (See Section 4.3). For distance field contours closer than the fifth contour, the initial thickness for coalescence then becomes twice the distance to the specific contour. The distances for the three different tests can be seen in Table 9. When the original algorithm is used, it can be seen that the force is implemented in iteration 880 and the coalescence event was prevented by iteration 940 (See Figure 27 *(a)*). Changing the algorithm to activate at the fourth distance field contour was performed next. It can be seen in iteration 880 that the force is implemented. After the force had been implemented, it can be seen in iteration 940 that the coalescence event is prevented (See Figure 27 *(b)*). This means that the initial coalescence thickness could be reduced to 3.6 *mm*.

Table 9: Initial liquid film thickness based on which distance field contour is used to activate the coalescence control algorithm.

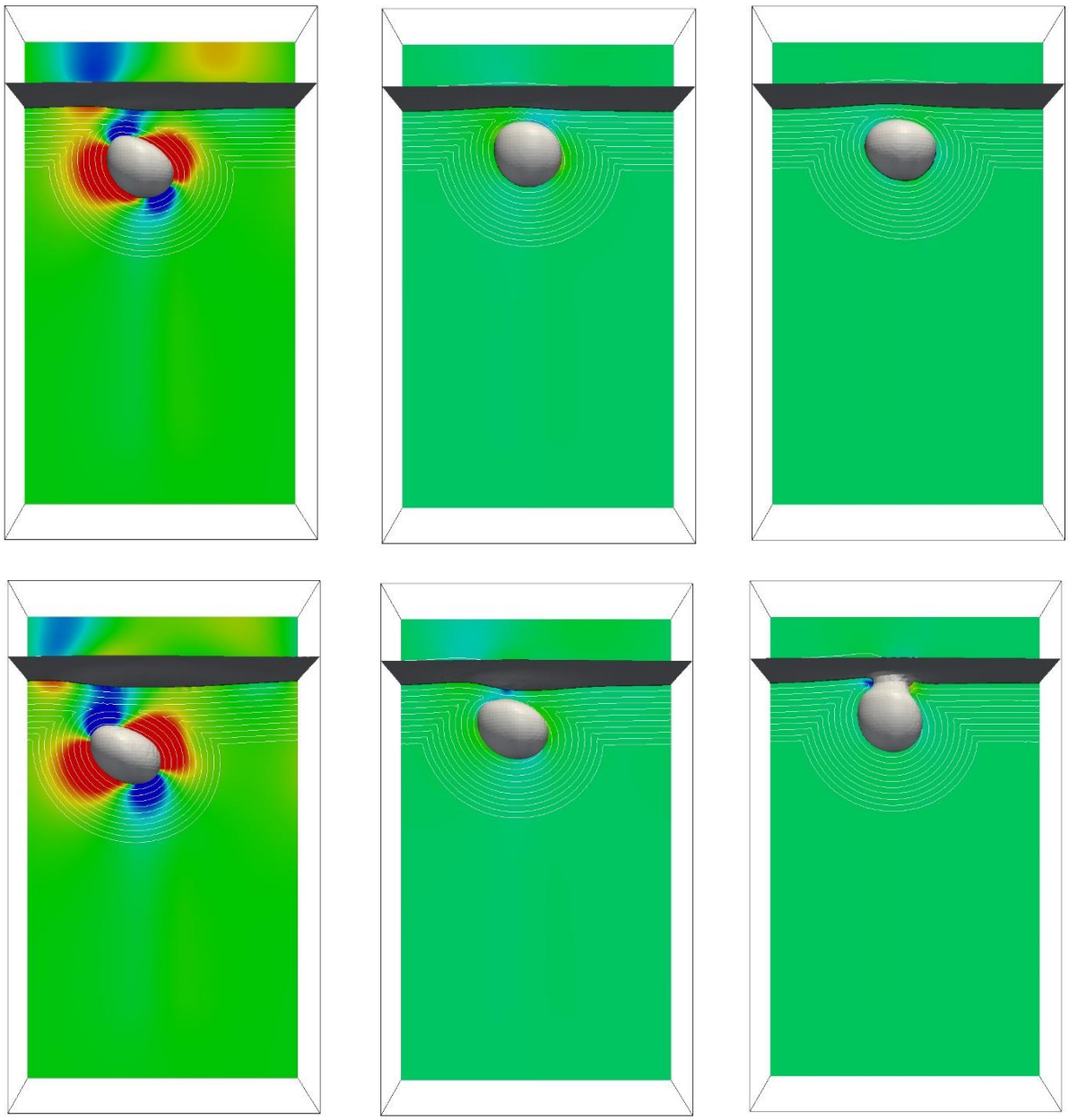
Initial Liquid Film Distance for Coalescence Events		
Identification Contour	Initial Thickness (mm)	5th Contour Application Diameter
3rd	2.7	4.5
4th	3.6	4.5
6th	4.5	4.5

Lastly, the algorithm was changed to activate on the third distance field contour. It can be seen in iteration 880 that the force activates to prevent the coalescence event. However, in

iteration 940, the timing of the force application and force strength was insufficient in preventing coalescence between the bubble and free surface (See Figure 27 *(c)*). This shows that the third distance field contour is insufficient in preventing coalescence events. Even though it was unable to prevent the coalescence in this simulation, it may still be possible to make this algorithm perform successfully. To do this, the magnitude of the force must be increased which results in adjusting the surface tension within the application volume. This means the distance could be reduced even further. This shows that even though the original algorithm cannot accurately represent the initial coalescence thickness, it is possible to reduce the distance by adjusting some of the parameters used in the algorithm.

5.3 Initial and Final Bubble Diameter Distribution

The first validation test planned for the algorithm uses a large simulation of bubbles to look at the initial and final bubble diameter distribution. The simulation that will be used is the same one with bubbles in turbulent flow conditions used for the multi-bubble coalescence prevention (See Section 4.2). The same domain setup and simulation parameters were used (See Figure 18 and Table 2). The distribution of bubble diameters and the initial and final void fractions will be compared to experimental results obtained by Colin *et al.* [20, 21]. Unlike when the multi-bubble coalescence prevention was tested, this simulation will be performed using the time tracking portion of the code as well. To match the experimental results obtained by Colin *et al*, the number of domain flow-throughs will be tracked to match the 2 *m* length used in the experimental apparatus.



(a) *(b)* *(c)*

Figure 27: Visualization of liquid film thickness simulations at iteration 880 and 940. (a) Original algorithm. (b) Force activates at fourth contour. (c) Force activates at the third contour.

5.4 Bubble Approach Velocity Effect on Coalescence

The second validation test to be performed for the algorithm uses the single bubble rising towards a free surface. This simulation design was based off the same setup as the simulation in coalescence time tracking testing (See Section 4.3 and Figure 21). However, the simulation will be modified to match an experiment performed by Kirkpatrick and Lockett [16]. A 5 *mm* diameter bubble will be placed in a liquid domain below a free surface of gas at the top of the domain. Two different distances below the free surface will be used: 45 *cm* and 67 *cm*. Gravity will be used to provide a buoyance force to force the bubble to approach the free surface. A wall will be placed on the top and bottom of the domain normal to the gravity and buoyance force. The mesh is made of 8.5M or 12.6M hexahedral elements resulting in approximately 18 elements across the diameter of the bubble. The Morton number (Mo) will be used to relate the adjusted simulation properties to the experimental properties (See Table 10).

Table 10: Overview of experiment and simulation parameters for the bubble approach velocity effect on coalescence simulation.

Parameter	Experimental	Simulation
Liquid	water	water
Gas	air	air
Pressure, Pa	0	0
Temperature, °C	N/A (0)	N/A (0)
Liquid Density, kg/m^3	997.17	997.17
Gas Density, kg/m^3	1.161	1.161
Liquid viscosity, kg/m-s	2.67E-03	3.57E-03
Gas viscosity, kg/m-s	1.86E-02	1.86E-02
Body Force, m/s²	9.81	1
Body Force Pressure Gradient, Pa/m	9782.24	997.17
Surface Tension, N/m^2	7.28E-02	5.00E-02

These tests will be performed with the time tracking portion of the code active. The results will be compared to the experimental results of Kirkpatrick and Lockett [16]. They observed that for bubble to free surface distances less than 61 *cm* the coalescence process was almost immediate. However, for distances larger than 61 *cm*, the coalescence process was much longer. By the time the bubble first makes contact with the free surface, it had gained sufficient speed to bounce off the free surface before coalescing. The coalescence time model taken from Prince and Blanch [7] did not take bubble approach velocity into account. However, testing this capability of the algorithm will help determine which simulations can be expected to provide proper physical results.

CHAPTER 6 CONCLUSION

6.1 Discussion

An algorithm was developed for the LS method that represents the effect of liquid film drainage during a coalescence event. Instead of using a high resolution grid to simulate the viscous drainage of the film, it prevents or slows the coalescence process by applying a force to both bubbles. The force is generated by locally changing the surface tension on a portion of each interface. This method was tested for single and multiple coalescence events in both laminar and turbulent flow regimes. In both cases, the algorithm successfully identified coalescence events and prevented the bubbles from coalescing.

A portion of the developed algorithm also tracks the amount of time a coalescence event has been active. This tracking time is then compared to a coalescence time model developed by Prince and Blanch [7]. The time tracking capability of the code was tested using a bubble

rising towards a free surface. The time tracking portion of the code successfully removed the locally modified surface tension allowing the bubble and free surface to coalesce.

We have observed some non-critical issues with the algorithm. One of these issues pertains to the deformation of the bubble interface. When the surface tension is changed, that portion of the bubble begins to flatten. In some situations, this is a physical process (e.g. representing the effect of the liquid film and the other interface on the bubble). However, the algorithm applies the changed surface tension before the liquid film is sufficiently thin to cause the flattening of the surface. Another issue is related to the initial film thickness that develops between the bubbles. With a moderate resolution, the initial thickness generated by the coalescence control algorithm is equal to 4.5 *mm*. This is much larger than the estimated thickness of 0.1 *mm* by Kirkpatrick and Lockett [16]. However, it has been demonstrated that the initial thickness can be reduced by adjusting parameters within the coalescence control algorithm. For multiple bubble simulations it is way more important to maintain the correct coalescence probability then to correctly resolve the thinnest film when the bubbles bounce off each other. Even with these minor issues, the presented coalescence control algorithm is the only feasible way to simulate multiple bubble behavior using LS approach at large scale.

6.2 Future Work

Even though the coalescence control algorithm has been tested and is working properly, there is still some development and testing that needs to be done. One of the first issues with the algorithm that needs addressed is how the coalescence control algorithm interacts with periodic boundary conditions. In some early simulations, when a coalescence event would

travel through a periodic boundary, some of the area near the boundary retains the changed surface tension. The changed surface tension would then change the trajectory of some of the bubbles as they pass through the boundary. The problem is probably caused by the way the boundary values are updated after each iteration. More work would need to be done in finding out how the changed surface tension and boundary conditions interact.

Another possible change in the algorithm that can make it more robust is developing an analytical model to determine the expected curvature for intersecting contours. The current method was developed empirically by analyzing the curvature values for a simple two bubble simulation. Multiple cases were performed where each the resolution was adjusted between each case. There is a significant jump in magnitude for curvature values where the distance field contours intersect. A curvature value close to the curvature jumps was chosen for each resolution. These values were then plotted against the resolution and an equation was generated using a linear fit (See Figure 11). The algorithm then uses this equation and multiplies the resulting values by 1.45 for a buffer to generate the limiting curvature values. Developing the analytical analysis will make the identification of the coalescence event locations more accurate.

One aspect of the algorithm that can be explored even further is reducing the initial thickness of the coalescence event. Some work has already been done investigating this change (See Section 5.2). More effort can be devoted to this in optimizing which contour may be used and how the surface tension needs to be changed in order to prevent the coalescence event. Another option that could be used is adjusting the shape of the application volume. Instead of using a sphere, as the application volume, an ellipsoid or rectangular prism could be used. A

third option would be to change the surface tension profile across the application volume. Currently, the surface tension is adjusted uniformly throughout the application volume. Instead of using a constant profile, a quadratic or exponential profile could be used. This would then reduce effect on the coalescence process until both bubbles have moved farther into the application volume. This would then allow the force to be implemented within a volume between the bubbles that is much thinner than the spherical volume. Finally, time dependent coalescence control force can be introduced which will monitor the bubble velocities and change its value based on the bubble dynamics. Adjusting the algorithm in such a way would allow the method to simulate the coalescence process closer to what is observed experimentally.

Another option to make the algorithm simulate coalescence events with increased accuracy would be to update the coalescence time model used. The model developed by Prince and Blanch [7] was developed without considering the approach velocity of the bubbles. It was also not able to accurately predict coalescence rates when electrolyte solutions with salt concentrations past the transition concentration. The inability of the model to predict the rates was attributed to the effects of turbulence on surface mobility, the dynamics of bubble collisions, and the solute concentration at the gas-liquid interface of the coalescing bubbles. A better time model that includes these effects could be implemented to make the time tracking portion of the algorithm closer to experimentally observed values.

Currently, the algorithm requires the user to provide the bubble radius and the estimated number of coalescence events to perform properly. The algorithm uses the bubble radius for calculating the limiting curvature values. However, this is redundant information because

information on the bubble radius is built into the distance field. By investigating other parts of the PHASTA code, it would be possible to pull the radius from the distance field information instead of providing it manually. The estimated number of coalescence events is used to determine how many times to loop through the portion of the code that identifies multiple coalescence events. Manually providing this looping value is not the most computationally efficient. The algorithm could be smarter by developing a portion of the algorithm to adjust this looping value. It would simple to decrease the looping value if some of the average coordinate values are not changed from the default value for a specific number of time steps. It would require more work to add to the looping value if more coalescence events then being identified are occurring. However, these adjustments would make the algorithm more robust and easier to use.

For future development and study, the algorithm could be analyzed for simulations when the bubbles are no longer spherical. Since the algorithm uses the curvature of the distance field to identify coalescence events. However, bubbles are only spherical at very small flow velocities or very small diameters. This adjustment to the algorithm will be necessary if the study wants some bubbles to coalesce. These coalescence events will increase the bubble diameter which may make it large enough to start deforming. However, it is undesirable for the algorithm to identify a false coalescence event due to sharp curvature changes in the bubble shape.

Lastly, studies could be more accurately performed by analyzing the influence of bubble diameter on other aspects of the simulation. One example could be extended to a paper by Bolotnov [28]. The purpose of the paper was to study the influence of bubbles on turbulence

anisotropy. Several ensemble runs had to be performed in this study to limit the number of coalescence events in each and to provide good approximation for mean bubble diameter. If this same study were redone, the coalescence control algorithm could be used to prevent all coalescence events and thus perform it using single run with constant mean bubble diameter.

REFERENCES

1. Bolotnov, I. A., Jansen, K. E., Drew, D. A., Oberai, A. A. & Lahey, Jr., R. T., Detached Direct Numerical Simulation of Turbulent Two-phase Bubbly Channel Flow. *Int. J. Multiphase Flow* **37**, 647-659 (2011).

2. Sanada, T., Watanabe, M. & Fukano, T., Effects of viscosity on coalescence of a bubble upon impact with a free surface. *Chemical Engineering Science* **60**, 5372-5384 (2005).

3. Chen, J.-D., Hahn, P. S. & Slattery, J. C., Coalescence time for a small drop or bubble at a fluid-fluid interface. *AIChE Journal* **30** (4), 622-630 (1984).

4. Lee, J. C. & Hodgson, T. D., Film flow and coalescence - I basic relations, film shape and criteria for interface mobility. *Chemical Engineering Science* **23**, 1375-1397 (1968).

5. Li, D., Coalescence between two small bubbles or drops. *Journal of Colloid and Interface Science* **163**, 108-119 (1994).

6. Marrucci, G., A theory of coalescence. *Chemical Engineering Science* **24**, 975-985 (1969).

7. Prince, M. J. & Blanch, H. W., Bubble coalescence and break-up in air-sparged bubble columns. *AIChE Journal* **36** (10), 1485-1499 (1990).

8. Marrucci, G. & Nicodemo, L., Coalescence of gas bubbles in aqueous solutions of inorganic electrolytes. *Chemical Engineering Science* **22**, 1257-1265 (1967).

9. Lessard, R. R. & Zieminski, S. A., Bubble coalescence and gas transfer in aqueous electrolytic solutions. *Ind. Eng. Chem. Fundam.* **10** (2), 260-269 (1971).

10. Cain, F. W. & Lee, J. C., A technique for studying the drainage and rupture of unstable liquid films formed between two captive bubbles: measurements on KCL solution. *Journal of Colloid and Interface Science* **106** (1), 70-85 (1985).

11. Kim, J. W. & Lee, W. K., Coalescence behavior of two bubbles in stagnant liquids. *Journal of Chemical Engineering of Japan* **20** (5), 448-453 (1987).

12. Craig, V. S. J., Ninham, B. W. & Pashley, R. M., The effect of electrolytes on bubble coalescence in water. *J. Phys. Chem.* **97** (39), 10192-10197 (1993).

13. Danov, K. D., Valkovska, D. S. & Kralchevsky, P. A., Hydrodynamic instability of coalescence in trains of emulsion drops or gas bubbles moving through a narrow capillary. *Journal of Colloid and Interface Science* **267**, 243-258 (2003).

14. Crabtree, J. R. & Bridgwater, J., Bubble coalescence in viscous liquids. *Chemical Engineering Science* **26**, 839-851 (1971).

15. de Nevers, N. & Wu, J.-L., Bubble coalescence in viscous fluids. *AIChE Journal*, 182-186 (1971).

16. Kirkpatrick, R. D. & Lockett, M. J., The influence of approach velocity on bubble coalescence. *Chemical Engineering Science* **29**, 2363-2373 (1974).

17. Stewart, C. W., Bubble interaction in low-viscosity liquids. *Int. J. Multiphase Flow* **21** (6), 1037-1046 (1995).

18. Sanada, T., Sato, A., Shirota, M. & Watanabe, M., Motion and coalescence of a pair of bubbles rising side by side. *Chemical Engineering Science* **64**, 2659-2671 (2009).

19. Kang, Q., Cui, H. L., Duan, L. & Hu, W. R., Experimental investigation on bubble coalescence under nonuniform temperature distribution in reduced gravity. *Journal of Colloid and Interface Science* **310**, 546-549 (2007).

20. Colin, C., Fabre, J. & Dukler, A. E., Gas-liquid flow at microgravity conditions - I: dispersed bubble and slug flow. *Int. J. Multiphase Flow* **17** (4), 533-544 (1991).

21. Colin, C., Riou, X. & Fabre, J., Bubble coalescence in gas-liquid flow at microgravity conditions. *Microgravity Sci. Technol.* **20**, 243-246 (2008).

22. Kamp, A. M., Chesters, A. K., Colin, C. & Fabre, J., Bubble coalescence in turbulent flows: a mechanistic model for turbulence-induced coalescence applied to microgravity bubbly pipe flow. *International Journal of Multiphase Flow* **27**, 1363-1396 (2001).

23. Chesters, A. K., The modelling of coalescence processes in fluid-liquid dispersions - a review of current understanding. *Trans. I. Chem. E.* **69** (part A), 353-361 (1991).

24. Mattson, M. D. & Mahesh, K., A one-way coupled, euler-lagrangian simulation of bubble coalescence in a turbulent pipe flow. *International Journal of Multiphase Flow* **40**, 68-82 (2012).

25. Olmos, E., Gentric, C., Vial, C., Wild, G. & Midoux, N., Numerical simulation of multiphase flow in bubble column reactors. influence of bubble coalescence and break-up. *Chemical Engineering Science* **56**, 6359-6365 (2001).

26. Sommerfeld, M., Bourloutski, E. & Broder, D., Euler/lagrange calculations of bubbly flows with consideration of bubble coalescence. *The Canadian Journal of Chemical Engineering* **81**, 508-518 (2003).

27. van den Hengel, E. I. V., Deen, N. G. & Kuipers, J. A. M., Application of coalescence and breakup models in a discrete bubble model for bubble columns. *Ind. Eng. Chem. Res.* **44** (14), 5233-5245 (2005).

28. Bolotnov, I. A., Influence of bubbles on the turbulence anisotropy. *Journal of Fluids Engineering* (2013).

29. Dabiri, S., Lu, J. & Tryggvason, G., Transition between regies of a vertical channel bubbly upflow due to bubble deformability. *Physics of Fluids* **25**, 102110-1 - 102110-12 (2013).

30. Crowe, C. T., Troutt, T. R. & Chung, J. N., Numerical models for two-phase turbulent flows. *Annu. Rev. Fluid. Mech.* **28**, 11-43 (1996).

31. van Sint Annaland, M., Dijkhuizen, W., Deen, N. G. & Kuipers, J. A. M., Numerical simulations of behavior of gas bubbles using a 3-D front tracking method. *AIChE Journal* **52** (1), 99-110 (2006).

32. Sussman, M. *et al.*, An adaptive level set approach for incompressible two-phase flows. *J. Computational Physics* **148** (1), 81-124 (1999).

33. Unverdi, S. O. & Tryggvason, G., A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics* **100**, 25-37 (1992).

34. van Sint Annaland, M., Deen, N. G. & Kuipers, J. A. M., Numerical simulation of gas bubbles behaviour using a three-dimensional volume of fluid method. *Chemical Engineering Science* **60**, 2999-3011 (2005).

35. Passandideh-Fard, M. & Farhangi, M. M., *A numerical study on bubble rise and interaction in a viscous liquid*, presented at Fifth International Conference on Transport Phenomena in Multiphase Systems, Bialystok, Poland, 2008.

36. Takada, N., Misawa, M., Tomiyama, A. & Hosokawa, S., Simulation of bubble motion under gravity by lattice boltzmann method. *Journal of Nuclear Science and Technology* **38** (5), 330-341 (2001).

37. Inamuro, T., Ogata, T., Tajima, S. & Konishi, N., A lattice boltzmann method for incompressible two-phase flows with large density differences. *Journal of Computational Physics* **198**, 628-644 (2004).

38. Sussman, M., Smereka, P. & Osher, S., A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics* **114**, 146-159 (1994).

39. Osher, S. & Fedkiw, R. P., Level set method: an overview of some recent results. *Journal of Computational Physics* **169**, 463-502 (2001).

40. Yang, X., James, A. J., Lowengrub, J., Zheng, X. & Cristini, V., An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids. *Journal of Computational Physics* **217**, 364-394 (2006).

41. Yu, Z. & Fan, L.-S., Direct simulation of the bouyant rise of bubbles in infinite liquid uisng level set method. *The Canadian Journal of Chemical Engineering* **86**, 267-265 (2008).

42. Bolotnov, I. A. & Podowski, M. Z., *Informing the development of the turbulence models for bubbly gas/liquid flow using interface tracking simulations*, presented at ANS Winter Meeting and Nuclear Technology Exposition, San Diego, CA, 2012.

43. Sussman, M., Fatemi, E., Smereka, P. & Osher, S., An improved level set method for incompressible two-phase flows. *Journal of Computational Fluids* **27** (5), 663-680 (1998).

44. Sussman, M. & Fatemi, E., 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 (1999).

45. Sethian, J. A., *Level set methods and fast marching methods* (Cambridge University Press, Cambridge, 1999).

APPENDIX

Appendix A

Code added to **e3ivar.f**:

```
c!.... Matt Talley's Bubble Coalescence Control
      if (coalcon.eq.1) then
!         if (update_coalcon.eq.1) then

         xx = zero

         do n  = 1,nenl
              xx(:,1) = xx(:,1) + shpfun(:,n) * xl(:,n,1)
              xx(:,2) = xx(:,2) + shpfun(:,n) * xl(:,n,2)
              xx(:,3) = xx(:,3) + shpfun(:,n) * xl(:,n,3)
         enddo

         bubradius = coalbubrad + 6.0d0*(epsilon_ls_tmp)
         bubradius2 = coalbubrad + epsilon_ls_tmp
         PtsPerDiam = (2.0d0*coalbubrad) / (epsilon_ls_tmp/1.8d0)
         Curvlim = -1.8083d2 * PtsPerDiam + 1.2873d3

!         endif
      endif

c!      MaxCurv = 30.0   ! Max curvature allowed for survace tension force usage,
Igor, April 2010
        do i = 1, npro
             weber(i) = Bo
             Ccurv = divqi(i,idflow+1)

c!.... Matt Talley's Bubble Coalescence Contorl
           if (coalcon.eq.1) then
!             if (update_coalcon.eq.1) then

              if (((Sclr(i).ge.(1.0d0*epsilon_ls_tmp)).and.
     &        (Sclr(i).le.(6.0d0*epsilon_ls_tmp)))
     &        .and.((Ccurv.ge.(-1.45d0*Curvlim)).or.
     &        (Ccurv.le.(1.45d0*Curvlim)))) then

                 xarray(i) = xx(i,1)
                 yarray(i) = xx(i,2)
                 zarray(i) = xx(i,3)
                 coordtag(i) = 1

              endif

!              endif ! update_coalcon
              if (coaltimtrak.eq.1) then
                 do k = 1, coalest
                    do n = 1,nenl
                       appvolume(i,n,k) = sqrt((xl(i,n,1)
     &                                  -avgxcoordold(k))**2 +
     &                                  (xl(i,n,2)-avgycoordold(k))**2 +
     &                                  (xl(i,n,3)-avgzcoordold(k))**2)

                       if (appvolume(i,n,k).le.5.0d0*epsilon_ls_tmp)
     &                 then
```

```
                        weber(i) = CoalInvSigma !Tension
                     endif
                  enddo
               enddo
            else
               do k = 1, coalest
               if (coalcon_rem(k).eq.0) then
                  do n = 1,nenl
                     appvolume(i,n,k) = sqrt((xl(i,n,1)
     &                                  -avgxcoordold(k))**2 +
     &                                  (xl(i,n,2)-avgycoordold(k))**2 +
     &                                  (xl(i,n,3)-avgzcoordold(k))**2)

                     if (appvolume(i,n,k).le.5.0d0*epsilon_ls_tmp)
     &                  then
                        weber(i) = CoalInvSigma !Tension
                     endif
                  enddo
               endif
               enddo
            endif ! coaltimtrak
         endif ! coalcon
```

Code added to **itrdrv.f**;

```
c!....Matt Talley's Coalescence Contorl
                     if (coalcon.eq.1) then
                        if (coaltimtrak.eq.1) then

                           do k = 1, coalest
                              avgxcoordold(k) = avgxcoordf(k)
                              avgycoordold(k) = avgycoordf(k)
                              avgzcoordold(k) = avgzcoordf(k)

                              if (avgxcoordold(k).gt.-1.0d3) then
                                 if (myrank.eq.master) write(*,*) 'Coalescence',
     &                           ' Event #: ', k
                                 if (myrank.eq.master) write(*,*) 'x average',
     &                           ' position:', avgxcoordold(k)
                                 if (myrank.eq.master) write(*,*) 'y average',
     &                           ' position:', avgycoordold(k)
                                 if (myrank.eq.master) write(*,*) 'z average',
     &                           ' position:', avgzcoordold(k)
                              endif

                           enddo ! k

                        else

                           itrtimestp = Delt(1)

                           call CoalescAppTime (avgxcoordf, avgycoordf,
     &                                          avgzcoordf, avgxcoordold2,
     &                                          avgycoordold2, avgzcoordold2,
```

```
     &                                          app_time, itrtimestp)
                        endif ! coaltimtrak
                      endif ! coalcon
```

Coalescence Event Identification Subroutine (coalesccon.f):

```
          subroutine CoalescCon (xarray, yarray, zarray, coordtag,
     &                           bubradius, bubradius2, avgxcoordf,
     &                           avgycoordf, avgzcoordf)
c
c-----------------------------------------------------------------------
c
c This routine computes the center coordinates for coalescence events
c during the simulation.
c
c Matt Talley, Winter 2014.
c-----------------------------------------------------------------------
c
        use pvsQbi  ! brings in NABI
        use stats   !
        use pointer_data  ! brings in the pointers for the blocked arrays
        use local_mass
        use spat_var_eps
        use timedata  ! for iblkts usage
c
        include "common.h"
        include "mpif.h"
c
        real*8 xarray(ibksiz), yarray(ibksiz), zarray(ibksiz)
        integer coordtag(ibksiz) !Passed arrays from e3ivar

        real*8 avgxcoord, avgycoord, avgzcoord, avgvectdist, !Coalescence control
center pt
     &         avgxcoordf(coalest), avgycoordf(coalest),
     &         avgzcoordf(coalest)

        real*8 totxcoordsum, totycoordsum, totzcoordsum, totvectdistsum,
     &         totxcoordsum_mult(coalest), totycoordsum_mult(coalest),
     &         totzcoordsum_mult(coalest) !Total sum of coordinates

        integer totcoordcount, totvectnumbsum,
     &          totcoordcount_mult(coalest) !Total number or coordinates

        real*8 xcoordsum, ycoordsum, zcoordsum, vectdistsum,
     &         xcoordsum_mult(coalest), ycoordsum_mult(coalest),
     &         zcoordsum_mult(coalest) !Sum from each processor

        integer totcoordnumb, vectnumbsum, diffnumbsum,
     &          totcoordnumb_mult(coalest) !Total number from each processor

        real*8 globalxcoord(ibksiz,nelblk), globalycoord(ibksiz,nelblk),
     &         globalzcoord(ibksiz,nelblk), vectdist(ibksiz,nelblk),
     &         xvectcoord(ibksiz,nelblk), yvectcoord(ibksiz,nelblk),
     &         zvectcoord(ibksiz,nelblk), vectangle(ibksiz,nelblk),
     &         vectdist2(ibksiz,nelblk) !Arrays from each processor

        real*8 dotmax(ibksiz,nelblk), vectdist_max_tmp, vect_max_xcoord,
     &         vect_max_ycoord, vect_max_zcoord, vect_max_xcoord_tmp,
```

```
     &         vect_max_ycoord_tmp, vect_max_zcoord_tmp

         real*8 phi_max, bubradius, bubradius2, length_bside_tri,
     &         angle1, angle2, hypot_len_1, hypot_len_2,
     &         avgcoordfdist(coalest,coalest)

         integer totcoordnumbarray(ibksiz,nelblk),
     &          vectnumb(ibksiz,nelblk)

         integer intone, vect_max_i, vect_max_iblk,
     &          sign_of_vect_max_xcoord, sign_of_vect_max_ycoord,
     &          sign_of_vect_max_zcoord, sign_of_vect_max_xcoord_tmp,
     &          sign_of_vect_max_ycoord_tmp, sign_of_vect_max_zcoord_tmp

         integer coalesc_tag(ibksiz,nelblk), consol_tag(coalest),
     &          avgcoordf_erase_tag(coalest)

c Initialize bubble coalecence control variables

         xcoordsum = 0.0d0 !Matt T.
         ycoordsum = 0.0d0
         zcoordsum = 0.0d0
         totcoordnumb = 0

         globalxcoord(:,:) = zero
         globalycoord(:,:) = zero
         globalzcoord(:,:) = zero
         totcoordnumbarray(:,:) = 0

         avgxcoord = -1.0d3
         avgycoord = -1.0d3
         avgzcoord = -1.0d3

         totxcoordsum = 0.0d0
         totycoordsum = 0.0d0
         totzcoordsum = 0.0d0
         totcoordcount = 0

         do iblk = 1, nelblk
            iel    = lcblk(1,iblk)
            npro   = lcblk(1,iblk+1) - iel

c....!Storing the coordinates into each globalcoord array and counting how many
c....!points are in each array. Then summing up each x, y, and z coordinate
c....!array to get a sum of each x, y, and z coordinate

            do i = 1, npro !Matt T.
               globalxcoord(i,iblk) = xarray(i)
               globalycoord(i,iblk) = yarray(i)
               globalzcoord(i,iblk) = zarray(i)
               totcoordnumbarray(i,iblk) = coordtag(i)
            enddo

            do m = 1, npro
               xcoordsum = xcoordsum + globalxcoord(m,iblk)
               ycoordsum = ycoordsum + globalycoord(m,iblk)
               zcoordsum = zcoordsum + globalzcoord(m,iblk)
               totcoordnumb = totcoordnumb + totcoordnumbarray(m,iblk)
            enddo
```

```
         enddo

          if (numpe.gt.1) then   !Matt T.
             call MPI_ALLREDUCE(xcoordsum, totxcoordsum, 1,
     &            MPI_DOUBLE_PRECISION, MPI_SUM, MPI_COMM_WORLD, ierr)
             call MPI_ALLREDUCE(ycoordsum, totycoordsum, 1,
     &            MPI_DOUBLE_PRECISION, MPI_SUM, MPI_COMM_WORLD, ierr)
             call MPI_ALLREDUCE(zcoordsum, totzcoordsum, 1,
     &            MPI_DOUBLE_PRECISION, MPI_SUM, MPI_COMM_WORLD, ierr)
             call MPI_ALLREDUCE(totcoordnumb, totcoordcount, 1,
     &            MPI_INTEGER, MPI_SUM, MPI_COMM_WORLD, ierr)

             if (totcoordcount.gt.0) then
                avgxcoord = totxcoordsum / DBLE(totcoordcount) !Matt T.
                avgycoord = totycoordsum / DBLE(totcoordcount)
                avgzcoord = totzcoordsum / DBLE(totcoordcount)
             endif
          else
             if (totcoordnumb.gt.0) then
                avgxcoord = xcoordsum / DBLE(totcoordnumb) !Matt T.
                avgycoord = ycoordsum / DBLE(totcoordnumb)
                avgzcoord = zcoordsum / DBLE(totcoordnumb)
             endif
          endif

c....!Begin the check for multiple coalescence events

          vectdistsum = 0.0d0
          vectnumbsum = 0
          vectdist(:,:) = 0.0d0
          vectnumb(:,:) = 0

          totvectdistsum = 0
          avgvectdist = 0.0d0
          totvectnumbsum = 0
          intone = 1

         do iblk = 1, nelblk
            iel    = lcblk(1,iblk)
            npro   = lcblk(1,iblk+1) - iel

            do i = 1, npro
               if (totcoordnumbarray(i,iblk).eq.intone) then

                  vectdist(i,iblk) = sqrt((globalxcoord(i,iblk) -
     &            avgxcoord)**2 + (globalycoord(i,iblk) - avgycoord)**2
     &            + (globalzcoord(i,iblk) - avgzcoord)**2)

                  vectnumb(i,iblk) = 1

               endif
            enddo ! i

            do m = 1, npro
               vectdistsum = vectdistsum + vectdist(m,iblk)
               vectnumbsum = vectnumbsum + vectnumb(m,iblk)
            enddo ! m
         enddo
```

```
            if (numpe.gt.1) then
               call MPI_ALLREDUCE(vectdistsum, totvectdistsum, 1,
     &              MPI_DOUBLE_PRECISION, MPI_SUM, MPI_COMM_WORLD, ierr)
               call MPI_ALLREDUCE(vectnumbsum, totvectnumbsum, 1,
     &              MPI_INTEGER, MPI_SUM, MPI_COMM_WORLD, ierr)

               if (totvectnumbsum.gt.0) then
                  avgvectdist = totvectdistsum / DBLE(totvectnumbsum)
               endif
            else
               if (vectnumbsum.gt.0) then
                  avgvectdist = vectdistsum / DBLE(vectnumbsum)
               endif
            endif

c....!Initialize new variables for the angle between vectors

            coalesc_tag(:,:) = 0

            xvectcoord(:,:) = 0.0d0
            yvectcoord(:,:) = 0.0d0
            zvectcoord(:,:) = 0.0d0

            xcoordsum_mult(:) = 0.0d0
            ycoordsum_mult(:) = 0.0d0
            zcoordsum_mult(:) = 0.0d0
            totcoordnumb_mult(:) = 0

            totxcoordsum_mult(:) = 0.0d0
            totycoordsum_mult(:) = 0.0d0
            totzcoordsum_mult(:) = 0.0d0
            totcoordcount_mult(:) = 0

            if ((avgxcoord.gt.-1.0d3).and.(avgvectdist.ge.bubradius)) then

               do iblk = 1, nelblk
                  iel    = lcblk(1,iblk)
                  npro   = lcblk(1,iblk+1) - iel

                  do i = 1, npro
                     if (totcoordnumbarray(i,iblk).eq.intone) then
                        xvectcoord(i,iblk) = globalxcoord(i,iblk) -
     &                  avgxcoord
                        yvectcoord(i,iblk) = globalycoord(i,iblk) -
     &                  avgycoord
                        zvectcoord(i,iblk) = globalzcoord(i,iblk) -
     &                  avgzcoord
                     endif
                  enddo
               enddo

               do k = 1, coalest

c....!Values to be re-initialized for each coalescence
                  sign_of_vect_max_xcoord = 0
                  sign_of_vect_max_ycoord = 0
                  sign_of_vect_max_zcoord = 0
                  sign_of_vect_max_xcoord_tmp = 0
                  sign_of_vect_max_ycoord_tmp = 0
```

```
            sign_of_vect_max_zcoord_tmp = 0

            vect_max_xcoord = 0.0d0
            vect_max_ycoord = 0.0d0
            vect_max_zcoord = 0.0d0
            vect_max_xcoord_tmp = 0.0d0
            vect_max_ycoord_tmp = 0.0d0
            vect_max_zcoord_tmp = 0.0d0

            vectdist_max = 0.0d0
            vectdist_max_tmp = 0.0d0
            vect_max_i = 0
            vect_max_iblk = 0

            vectangle(:,:) = 0.0d0
            dotmax(:,:) = 0.0d0
            phi_max = 0.0d0
            length_bside_tri = 0.0d0
            vectdist2(:,:) = 0.0d0
            angle1 = 0.0d0
            angle2 = 0.0d0
            hypot_len_1 = 0.0d0
            hypot_len_2 = 0.0d0

            do iblk = 1, nelblk
               iel    = lcblk(1,iblk)
               npro   = lcblk(1,iblk+1) - iel

               do i = 1, npro
                  if ((coalesc_tag(i,iblk).eq.0).and.
&                 (vectdist(i,iblk).gt.vectdist_max)) then
                     vectdist_max = vectdist(i,iblk)
                  endif
               enddo
            enddo

            if (numpe.gt.1) then

               call MPI_ALLREDUCE (vectdist_max,vectdist_max_tmp,1,
&              MPI_DOUBLE_PRECISION,MPI_MAX, MPI_COMM_WORLD,ierr)
            else
               vectdist_max_tmp = vectdist_max
            endif

            vectdist_max = vectdist_max_tmp

            do iblk = 1, nelblk
               iel    = lcblk(1,iblk)
               npro   = lcblk(1,iblk+1) - iel

               do i = 1, npro
                  if ((coalesc_tag(i,iblk).eq.0).and.
&              (abs(vectdist(i,iblk) - vectdist_max).lt.1.0d-24)) then
                     vect_max_i = i
                     vect_max_iblk = iblk !Values only known to one processor

                     exit

                  endif
```

```
            enddo
         enddo

         if ((vect_max_i.gt.0).and.(vect_max_iblk.gt.0)) then

            vect_max_xcoord = xvectcoord(vect_max_i,vect_max_iblk)
            vect_max_ycoord = yvectcoord(vect_max_i,vect_max_iblk)
            vect_max_zcoord = zvectcoord(vect_max_i,vect_max_iblk)

         endif

         if (vect_max_xcoord.lt.0.0d0) then
            sign_of_vect_max_xcoord = 1
         endif

         if (vect_max_ycoord.lt.0.0d0) then
            sign_of_vect_max_ycoord = 1
         endif

         if (vect_max_zcoord.lt.0.0d0) then
            sign_of_vect_max_zcoord = 1
         endif

         call MPI_ALLREDUCE (abs(vect_max_xcoord),
     &        vect_max_xcoord_tmp,1,MPI_DOUBLE_PRECISION,MPI_MAX,
     &        MPI_COMM_WORLD,ierr)
         call MPI_ALLREDUCE (abs(vect_max_ycoord),
     &        vect_max_ycoord_tmp,1,MPI_DOUBLE_PRECISION,MPI_MAX,
     &        MPI_COMM_WORLD,ierr)
         call MPI_ALLREDUCE (abs(vect_max_zcoord),
     &        vect_max_zcoord_tmp,1,MPI_DOUBLE_PRECISION,MPI_MAX,
     &        MPI_COMM_WORLD,ierr)
         call MPI_ALLREDUCE (sign_of_vect_max_xcoord,
     &        sign_of_vect_max_xcoord_tmp,1,MPI_INTEGER,MPI_MAX,
     &        MPI_COMM_WORLD,ierr)
         call MPI_ALLREDUCE (sign_of_vect_max_ycoord,
     &        sign_of_vect_max_ycoord_tmp,1,MPI_INTEGER,MPI_MAX,
     &        MPI_COMM_WORLD,ierr)
         call MPI_ALLREDUCE (sign_of_vect_max_zcoord,
     &        sign_of_vect_max_zcoord_tmp,1,MPI_INTEGER,MPI_MAX,
     &        MPI_COMM_WORLD,ierr)

         if (sign_of_vect_max_xcoord_tmp.eq.intone) then
            vect_max_xcoord = -vect_max_xcoord_tmp
         else
            vect_max_xcoord = vect_max_xcoord_tmp
         endif

         if (sign_of_vect_max_ycoord_tmp.eq.intone) then
            vect_max_ycoord = -vect_max_ycoord_tmp
         else
            vect_max_ycoord = vect_max_ycoord_tmp
         endif

         if (sign_of_vect_max_zcoord_tmp.eq.intone) then
            vect_max_zcoord_tmp = -vect_max_zcoord_tmp
         else
            vect_max_zcoord = vect_max_zcoord_tmp
         endif
```

```
                do iblk = 1, nelblk
                   iel    = lcblk(1,iblk)
                   npro   = lcblk(1,iblk+1) - iel

                   do i = 1, npro
                      if ((coalesc_tag(i,iblk).eq.0).and.
     &                   (totcoordnumbarray(i,iblk).eq.intone)) then
                         dotmax(i,iblk) = xvectcoord(i,iblk)
     &                   * vect_max_xcoord
     &                   + yvectcoord(i,iblk) * vect_max_ycoord
     &                   + zvectcoord(i,iblk) * vect_max_zcoord

                         if ((dotmax(i,iblk) / (vectdist(i,iblk)
     &                   * vectdist_max)).gt.1.0d0) then
                            vectangle(i,iblk) = acos(1.0d0)
                         else if ((dotmax(i,iblk) / (vectdist(i,iblk)
     &                   * vectdist_max)).lt.-1.0d0) then
                            vectangle(i,iblk) = acos(-1.0d0)
                         else
                            vectangle(i,iblk) = acos(dotmax(i,iblk)
     &                      / (vectdist(i,iblk) * vectdist_max))
                         endif
                      endif
                   enddo
                enddo

c!....Determine the distance between each vector and vector_max
                do iblk = 1, nelblk
                   iel    = lcblk(1,iblk)
                   npro   = lcblk(1,iblk+1) - iel

                   do i = 1, npro
                      if ((coalesc_tag(i,iblk).eq.0).and.
     &                   (totcoordnumbarray(i,iblk).eq.intone)) then
                         vectdist2(i,iblk) = sqrt((xvectcoord(i,iblk) -
     &                      vect_max_xcoord)**2 + (yvectcoord(i,iblk) -
     &                      vect_max_ycoord)**2 + (zvectcoord(i,iblk) -
     &                      vect_max_zcoord)**2)
                      endif
                   enddo ! i
                enddo ! iblk

c....!Initialze the first tag based on the proper processor and tag the rest
                if ((vect_max_i.gt.0).and.(vect_max_iblk.gt.0)) then
                   coalesc_tag(vect_max_i,vect_max_iblk) = k
                endif

c!....Determine the maximum angle and assign event numbers
                length_bside_tri = sqrt(vectdist_max**2 -
     &          (2.0d0*bubradius)**2)

                if (vectdist_max**2.ge.(2.0d0*bubradius)**2) then

                   do iblk = 1, nelblk
                      iel    = lcblk(1,iblk)
                      npro   = lcblk(1,iblk+1) - iel

                      do i = 1, npro
```

```
                        if ((totcoordnumbarray(i,iblk).eq.intone).and.
     &                     (vectdist(i,iblk).ge.2.0d0*bubradius)) then
                           phi_max = acos(length_bside_tri /
     &                        vectdist_max)
                        endif

                        if ((totcoordnumbarray(i,iblk).eq.intone).and.
     &                     (vectdist(i,iblk).lt.2.0d0*bubradius)) then
                           phi_max = asin((2.0d0*bubradius) /
     &                        vectdist_max)
                        endif

                        if ((coalesc_tag(i,iblk).eq.0).and.
     &                     (totcoordnumbarray(i,iblk).eq.intone)) then
                           if ((abs(vectdist(i,iblk)-vectdist_max).le.
     &                        (2.0d0*bubradius)).and.
     &                        (vectangle(i,iblk).le.phi_max)) then

                              coalesc_tag(i,iblk) = k
                           endif
                        endif
                     enddo ! i
                  enddo ! iblk

               else

c!....Calculate the maximum angle at the 1st epsilon contour
                  if (vectdist_max.gt.0.0d0) then
                     angle1 = acos(bubradius2 / vectdist_max)
                     hypot_len_1 = vectdist_max*sin(angle1)
                     hypot_len_2 = 2.0d0*bubradius - hypot_len_1
                     angle2 = atan(hypot_len_2 / bubradius2)

                     phi_max = angle1 + angle2

                     do iblk = 1, nelblk
                        iel    = lcblk(1,iblk)
                        npro   = lcblk(1,iblk+1) - iel

                        do i = 1, npro
                           if ((coalesc_tag(i,iblk).eq.0).and.
     &                        (totcoordnumbarray(i,iblk).eq.intone)) then
                              if ((vectdist2(i,iblk).le.
     &                           (2.0d0*bubradius)).and.
     &                           (vectangle(i,iblk).le.phi_max)) then

                                 coalesc_tag(i,iblk) = k
                              endif
                           endif
                        enddo ! i
                     enddo ! iblk
                  endif
               endif

               do iblk = 1, nelblk
                  iel    = lcblk(1,iblk)
                  npro   = lcblk(1,iblk+1) - iel

                  do m = 1, npro
```

```
                  if (coalesc_tag(m,iblk).eq.k) then
                     xcoordsum_mult(k) = xcoordsum_mult(k) +
     &               globalxcoord(m,iblk)
                     ycoordsum_mult(k) = ycoordsum_mult(k) +
     &               globalycoord(m,iblk)
                     zcoordsum_mult(k) = zcoordsum_mult(k) +
     &               globalzcoord(m,iblk)
                     totcoordnumb_mult(k) = totcoordnumb_mult(k) +
     &               totcoordnumbarray(m,iblk)
                  endif
               enddo ! m
            enddo ! iblk

            if (numpe > 1) then   !Matt T.
               call MPI_ALLREDUCE(xcoordsum_mult(k),
     &              totxcoordsum_mult(k),1,MPI_DOUBLE_PRECISION,
     &              MPI_SUM, MPI_COMM_WORLD,ierr)
               call MPI_ALLREDUCE(ycoordsum_mult(k),
     &              totycoordsum_mult(k),1,MPI_DOUBLE_PRECISION,
     &              MPI_SUM, MPI_COMM_WORLD,ierr)
               call MPI_ALLREDUCE(zcoordsum_mult(k),
     &              totzcoordsum_mult(k),1,MPI_DOUBLE_PRECISION,
     &              MPI_SUM, MPI_COMM_WORLD,ierr)
               call MPI_ALLREDUCE(totcoordnumb_mult(k),
     &              totcoordcount_mult(k),1,MPI_INTEGER,MPI_SUM,
     &              MPI_COMM_WORLD, ierr)

               if (totcoordcount_mult(k).gt.0) then
                  avgxcoordf(k) = totxcoordsum_mult(k) /
     &            DBLE(totcoordcount_mult(k)) !Matt T.
                  avgycoordf(k) = totycoordsum_mult(k) /
     &            DBLE(totcoordcount_mult(k))
                  avgzcoordf(k) = totzcoordsum_mult(k) /
     &            DBLE(totcoordcount_mult(k))
               endif

            else

               if (totcoordnumb_mult(k).gt.0) then
                  avgxcoordf(k) = xcoordsum_mult(k) /
     &            DBLE(totcoordnumb_mult(k)) !Matt T.
                  avgycoordf(k) = ycoordsum_mult(k) /
     &            DBLE(totcoordnumb_mult(k))
                  avgzcoordf(k) = zcoordsum_mult(k) /
     &            DBLE(totcoordnumb_mult(k))
               endif

            endif ! numpe

         enddo ! coalest

c!....Consolidate any average points that are too close to one another
         avgcoordfdist(:,:) = 2.0d0*bubradius
         consol_tag(:) = 0
         avgcoordf_erase_tag(:) = 0

         do k1 = 1, (coalest-1)
            if (avgxcoordf(k1).gt.-1.0d3) then
               do k2 = (k1+1), coalest
```

```
                avgcoordfdist(k1,k2) = sqrt((avgxcoordf(k1) -
&               avgxcoordf(k2))**2 + (avgycoordf(k1) -
&               avgycoordf(k2))**2 + (avgzcoordf(k1) -
&               avgzcoordf(k2))**2)
             enddo
          endif
       enddo

       do k1 = 1, (coalest-1)
          do k2 = (k1+1), coalest
             if (avgcoordfdist(k1,k2).lt.(2.0d0*bubradius)) then
                do iblk = 1, nelblk
                   iel    = lcblk(1,iblk)
                   npro   = lcblk(1,iblk+1) - iel

                   do i = 1, npro
                      if (coalesc_tag(i,iblk).eq.k2) then
                         coalesc_tag(i,iblk) = k1
                      endif
                   enddo ! i
                enddo ! iblk

                consol_tag(k1) = 1
                avgcoordf_erase_tag(k2) = 1

             endif ! avgcoordfdist
          enddo ! k2
       enddo ! k1

       do k1 = 1, (coalest-1)
          do k2 = (k1+1), coalest
             if (avgcoordf_erase_tag(k2).eq.intone) then
                avgxcoordf(k2) = -1.0d3
                avgycoordf(k2) = -1.0d3
                avgzcoordf(k2) = -1.0d3
             endif
          enddo ! k2

          if (consol_tag(k1).eq.intone) then

             xcoordsum_mult(k1) = 0.0d0
             ycoordsum_mult(k1) = 0.0d0
             zcoordsum_mult(k1) = 0.0d0
             totcoordnumb_mult(k1) = 0.0d0

             totxcoordsum_mult(k1) = 0.0d0
             totycoordsum_mult(k1) = 0.0d0
             totzcoordsum_mult(k1) = 0.0d0
             totcoordcount_mult = 0.0d0

             do iblk = 1, nelblk
                iel    = lcblk(1,iblk)
                npro   = lcblk(1,iblk+1) - iel

                do m = 1, npro
                   if (coalesc_tag(m,iblk).eq.k1) then
                      xcoordsum_mult(k1) = xcoordsum_mult(k1) +
&                     globalxcoord(m,iblk)
                      ycoordsum_mult(k1) = ycoordsum_mult(k1) +
```

```
&                     globalycoord(m,iblk)
                      zcoordsum_mult(k1) = zcoordsum_mult(k1) +
&                     globalzcoord(m,iblk)
                      totcoordnumb_mult(k1) = totcoordnumb_mult(k1)
&                     + totcoordnumbarray(m,iblk)
                   endif
                enddo !m
             enddo !iblk

             if (numpe > 1) then   !Matt T.
                call MPI_ALLREDUCE(xcoordsum_mult(k1),
&                    totxcoordsum_mult(k1),1,MPI_DOUBLE_PRECISION,
&                    MPI_SUM, MPI_COMM_WORLD,ierr)
                call MPI_ALLREDUCE(ycoordsum_mult(k1),
&                    totycoordsum_mult(k1),1,MPI_DOUBLE_PRECISION,
&                    MPI_SUM, MPI_COMM_WORLD,ierr)
                call MPI_ALLREDUCE(zcoordsum_mult(k1),
&                    totzcoordsum_mult(k1),1,MPI_DOUBLE_PRECISION,
&                    MPI_SUM, MPI_COMM_WORLD,ierr)
                call MPI_ALLREDUCE(totcoordnumb_mult(k1),
&                    totcoordcount_mult(k1),1,MPI_INTEGER,MPI_SUM,
&                    MPI_COMM_WORLD, ierr)

                if (totcoordcount_mult(k1).gt.0) then
                   avgxcoordf(k1) = totxcoordsum_mult(k1) /
&                  DBLE(totcoordcount_mult(k1))
                   avgycoordf(k1) = totycoordsum_mult(k1) /
&                  DBLE(totcoordcount_mult(k1))
                   avgzcoordf(k1) = totzcoordsum_mult(k1) /
&                  DBLE(totcoordcount_mult(k1))
                endif

             else

                if (totcoordnumb_mult(k1).gt.0) then
                   avgxcoordf(k1) = xcoordsum_mult(k1) /
&                  DBLE(totcoordnumb_mult(k1))
                   avgycoordf(k1) = ycoordsum_mult(k1) /
&                  DBLE(totcoordnumb_mult(k1))
                   avgzcoordf(k1) = zcoordsum_mult(k1) /
&                  DBLE(totcoordnumb_mult(k1))
                endif

             endif ! numpe
          endif ! consoltag
       enddo ! k1

    else
       avgxcoordf(1) = avgxcoord
       avgycoordf(1) = avgycoord
       avgzcoordf(1) = avgzcoord
    endif !avgxcoord and avgvectdist

    end
```

Coalescence Event Time Tracking Subroutine (coalescapptime.f):

```
    subroutine CoalescAppTime(avgxcoordf, avgycoordf, avgzcoordf,
```

```
     &                              avgxcoordold2, avgycoordold2,
     &                              avgzcoordold2, app_time, itrtimestp)
c
c-----------------------------------------------------------------------
c
c This routine assigns the new center coordinates for the coalescence
c control and tracks the amount of time the coalescence force has
c been active.
c
c Matt Talley, Winter 2014.
c-----------------------------------------------------------------------
c
        use pvsQbi  ! brings in NABI
        use stats   !
        use pointer_data  ! brings in the pointers for the blocked arrays
        use local_mass
        use spat_var_eps
        use timedata  ! for iblkts usage
c
        include "common.h"
        include "mpif.h"
c
        real*8 avgxcoordf(coalest), avgycoordf(coalest),
     &         avgzcoordf(coalest), avgxcoordold2(coalest),
     &         avgycoordold2(coalest), avgzcoordold2(coalest),
     &         avgcoorddist(coalest,coalest),
     &         app_time(coalest,2)

        real*8 itrtimestp, coalesc_time

        integer event_tag(coalest,coalest)

c!.... Initialize variables
        event_tag(:,:) = 0
        app_time(:,1) = 0.0d0
        avgcoorddist(:,:) = 1.0d4
        coalesc_time = 0.0d0

c!.... Calculate the maximum coalescence time
        coalesc_time = sqrt((((coalbubrad)**3) * datmat(1,1,1))
     &                 / (16.0d0 *(1/Bo))) * log(1.0d-4/1.0d-8)

        do k1 = 1, coalest
           avgxcoordold(k1) = avgxcoordf(k1)
           avgycoordold(k1) = avgycoordf(k1)
           avgzcoordold(k1) = avgzcoordf(k1)

c!....Track the amount of time the Coalescence Control Algorithm has
c!....active for each different event

           if (app_time(k1,2).le.coalesc_time) then
              if (avgxcoordold2(k1).gt.-1.0d3) then
                 do k2 = 1, coalest
                    avgcoorddist(k1,k2) = sqrt((avgxcoordold(k2) -
     &              avgxcoordold2(k1))**2 + (avgycoordold(k2) -
     &              avgycoordold2(k1))**2 + (avgzcoordold(k2) -
     &              avgzcoordold2(k1))**2)
                 enddo ! k2
```

```
            do k2 = 1, coalest
               if ((avgcoorddist(k1,k2).lt.coalbubrad).and.
&                 (event_tag(k1,k2).eq.0)) then
                  app_time(k2,1) = app_time(k1,2) + itrtimestp/2.0d0
                  event_tag(k1,:) = 1
                  event_tag(:,k2) = 1
                  coalcon_rem(k1) = 0

                  if (myrank.eq.master) write(*,*) 'Coalescence',
&                 ' Event #: ',k1,' to ',k2
                  if (myrank.eq.master) write(*,*) 'x average',
&                 ' position:', avgxcoordold(k2)
                  if (myrank.eq.master) write(*,*) 'y average',
&                 ' position:', avgycoordold(k2)
                  if (myrank.eq.master) write(*,*) 'z average',
&                 ' position:', avgzcoordold(k2)
               endif
            enddo ! k2

            do k2 = 1, coalest
               if (event_tag(k1,k2).eq.0) then
                  if ((avgcoorddist(k1,k2).gt.coalbubrad).and.
&                    (avgcoorddist(k1,k2).lt.1.0d4)) then
                     app_time(k1,2) = 0.0d0
                     coalcon_rem(k1) = 1
                     event_tag(k1,:) = 1

                     if (myrank.eq.master) write(*,*) 'Old',
&                    ' Coalescence Event #: ',k1,' has ended',
&                    ' because they bounced off one another'
                  endif
               endif
            enddo ! k2
         endif

      else

         if (avgxcoordold2(k1).gt.-1.0d3) then
            do k2 = 1, coalest
               avgcoorddist(k1,k2) = sqrt((avgxcoordold(k2) -
&              avgxcoordold2(k1))**2 + (avgycoordold(k2) -
&              avgycoordold2(k1))**2 + (avgzcoordold(k2) -
&              avgzcoordold2(k1))**2)
            enddo ! k2
            do k2 = 1, coalest
               if ((avgcoorddist(k1,k2).lt.coalbubrad).and.
&                 (event_tag(k1,k2).eq.0)) then
                  app_time(k2,1) = app_time(k1,2) + itrtimestp/2.0d0
                  event_tag(k1,:) = 1
                  event_tag(:,k2) = 1
                  coalcon_rem(k1) = 1

                  if (myrank.eq.master) write(*,*) 'Coalescence',
&                 ' Event #: ',k1,' to ',k2,' has exceeded the',
&                 ' drainage time and the force is being removed.'
               endif
            enddo ! k2

            do k2 = 1, coalest
```

```
               if (event_tag(k1,k2).eq.0) then
                  if ((avgcoorddist(k1,k2).gt.coalbubrad).and.
&                    (avgcoorddist(k1,k2).lt.1.0d4)) then
                     app_time(k1,2) = 0.0d0
                     coalcon_rem(k1) = 1
                     event_tag(k1,:) = 1

                     if (myrank.eq.master) write(*,*) 'Old',
&                    ' Coalescence Event #: ',k1,' has ended'
                  endif
               endif
            enddo ! k2
         endif
      endif !(app_time)

      if (app_time(k1,2).le.coalesc_time) then
         if (avgxcoordold2(k1).le.-1.0d3) then
            do k2 = 1, coalest
               if ((event_tag(k1,k2).eq.0).and.
&               (avgxcoordold(k2).gt.-1.0d3)) then

                  app_time(k2,1) = itrtimestp/2.0d0
                  event_tag(:,k2) = 1
                  coalcon_rem(k1) = 0

                  if (myrank.eq.master) write(*,*) 'New',
&                 ' Coalescence Event #: ',k2
                  if (myrank.eq.master) write(*,*) 'x average',
&                 ' position:', avgxcoordold(k2)
                  if (myrank.eq.master) write(*,*) 'y average',
&                 ' position:', avgycoordold(k2)
                  if (myrank.eq.master) write(*,*) 'z average',
&                 ' position:', avgzcoordold(k2)

               endif
            enddo !k2
         endif
      endif
   enddo ! k1

   app_time(:,2) = app_time(:,1)
   avgxcoordold2(:) = avgxcoordold(:)
   avgycoordold2(:) = avgycoordold(:)
   avgzcoordold2(:) = avgzcoordold(:)

   end
```