Adaptive Solution Techniques For Simulating Underwater Explosions and Implosions

Samet Y. Kadioglu *

Department of Mathematics, Phillips Hall, University of North Carolina, Chapel Hill, NC 27599

Mark Sussman[†]

Department of Mathematics, 208 Love Building, Florida State University, Tallahassee, FL, 32306

December 18, 2006

Abstract

Adaptive solution techniques are presented for simulating underwater explosions and implosions. The liquid is assumed to be an adiabatic fluid and the solution in the gas is assumed to be uniform in space. The solution in the water is integrated in time using a semi-implicit time discretization of the adiabatic Euler equations. Results are presented either using a non-conservative semi-implicit algorithm or a conservative semi-implicit algorithm. A semi-implicit algorithm allows one to compute with relatively large time steps compared to an explicit method. The interface solver is based on the coupled level set and volume-of-fluid method (CLSVOF) [19, 20]. Several underwater explosion and implosion test cases are presented to show the performances of our proposed techniques.

Key Words: AMR; Semi-implicit; CLSVOF; Multi-phase; Under-water explosions and implosions

1 Introduction

In this paper, we present adaptive solution techniques for simulating underwater explosion and implosion problems. These are multi-phase flow problems in which we solve

---

*email: kadioglu@email.unc.edu

[†]email:sussman@math.fsu.edu

a liquid component (water obeying Tait equation of state) and a gas component (gas assume spatially uniform, and adiabatic). A Multi-phase flow calculation can be based on either solving a two-fluid system, i.e, both gas and liquid (compressible or incompressible) equations are solved, or based on solving a single fluid system, i.e, only liquid (compressible or incompressible) equations are solved, and the gas contents are treated as spatially uniform. For instance Fedkiw *et al* [10, 7] used compressible(gas)-compressible(water) and compressible(gas)-incompressible (water) models. Compressible(gas)-compressible(water) models are also used by [22, 25, 13]. Our method solves the compressible liquid equations together with a compressible, spatially uniform, gas model. Multi-phase flow methods, in which the liquid is treated as compressible, are often formulated as an explicit method(i,e [22, 10, 25, 13]). The time step for an explicit method is constrained by both the magnitude of the underlying velocity field and the magnitude of the sound speed of water. Alternatively, the time step for a semi-implicit method is not constrained by the sound speed of water. The following semi-implicit approaches have been developed for treating water as a compressible fluid [28, 27, 9, 29]. In the context of a multi-phase flow problem, very little attention has been given to semi-implicit approaches for discretizing the equations for the compressible flow of water [30].

Here we use semi-implicit discretizations to solve underwater explosion and implosion problems. The semi-implicit discretization approach enables us to solve many problems involving underwater explosions/implosions with a much larger time step than using an explicit discretization approach. An efficiency comparison between our semi-implicit methods and an explicit method due to Wardlaw [25] is made in the results section.

Adaptive Mesh Refinement (AMR) is another core feature of our algorithms. Adding adaptivity to our calculations makes our semi-implicit methods even more efficient. The need for the employment of the AMR technology in our computations is as follows. When solving underwater explosion and implosion problems, it is important to resolve the flow only around the high gradient regions such as shocks or material discontinuities in order to obtain efficient and accurate solution representations and save CPU time. Automating dynamic Adaptive Mesh Refinement techniques are customized to serve these purposes. The idea behind the Adaptive Mesh Refinement technique [5, 4] is to overlay successively finer resolution grid patches on top of underlying coarse grids, and to introduce a recursive time integration algorithm, then to synchronize the data between different grid levels. The time integration algorithm can be applied in two ways. One way is that all grid levels are advanced with the finest grid level time step. This is called the *no-time-subcycling* procedure. Another way is that a coarser grid level is advanced with a coarse time step and a finer grid level is advanced with multiple fine time steps until the finer level time reaches to the coarser level time. This is called the *time-subcycling* procedure.

Early Adaptive Mesh Refinement (AMR) techniques [5, 4, 3] are developed for solving hyperbolic conservation laws. Later, they are extended to solve the compressible Navier-Stokes equations [17, 16]. There have been significant efforts to solve incompressible or weakly compressible flows adaptively [1, 14, 11, 12, 2, 8, 18].

Our method shares some common features with [1] in the sense that we both subcycle in time and we both provide velocity continuity across the coarse/fine grids. Almgren's *et*

al [1] method includes a synchronization projection step to provide velocity and pressure continuity across the coarse/fine grids. In [1], firstly, the velocity field is advanced on each level separately allowing velocity mismatch across the coarse/fine levels, then a multi-level composite projection step is applied at the end of each coarse level to correct the velocity differences. As a result their correction procedure modifies the solution on both coarse and fine levels. On the other hand, during our time advancing step, we solve the pressure equation on the current level and all levels above simultaneously. This produces accurate pressure boundary conditions for the next finer level. Also, since we solve on all levels $l' \geq l$ simultaneously, the velocity mismatch error, that gets corrected at the end of the ensuing fine time level integration steps, is significantly reduced. During the synchronization step, we solve the synchronization equations in the underlying coarse regions with Neumann boundary conditions. In this way, we maintain the velocity continuity without modifying the solution at the fine levels, i.e, we assume that the fine level velocity is correct and should not be changed.

The contents of the present paper is as follows. In Section 2, we describe the governing dynamics equations. In Section 3, we review the AMR grid hierarchy and define the components of our adaptive semi-implicit algorithms. In Section 4, we present the numerical results from the computations of underwater explosion/implosion test cases . Section 5 includes some concluding remarks.

2 Governing equations

Here we are interested in studying non-linear bubble dynamics by simulating underwater explosions and implosions. An underwater explosion can be modeled as a high pressure gas bubble which generates a shock wave. An underwater implosion can be modeled as a collapse of a low pressure gas bubble. We make the following assumptions when modeling the flow in water: the bubble growth and collapse are assumed to be axisymmetric. The system is assumed to be adiabatic, i.e, we ignore the heat convection in water. We assume that surface tension and viscous effects are negligible. Finally we assume that water is compressible and obeys the Tait equation of state (see equation (32) in Section 4), which is independent of internal energy [25].

With the above assumptions, the governing fluid dynamics equations for water become the inviscid compressible Euler equations,

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{1}$$

where

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \end{pmatrix},$$

$$F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \end{pmatrix},$$

and

$$G = \begin{pmatrix} \rho u \\ \rho uv \\ \rho v^2 + p \end{pmatrix},$$

where $\rho, \mathbf{u} = (u, v)$, and p denote density, velocity, and pressure per unit volume, the momentum field is defined as $\mathbf{m} = \rho \mathbf{u}$. We enforce Dirichlet type moving boundary conditions for the water pressure, i.e, we specify the water pressure as the bubble pressure at the bubble wall. We treat the gas as uniform in space, but varies in time. In other words, we don't solve the gas dynamics equations inside the bubble. Instead, we compute the time dependent bubble pressure by the JWL (Jones-Wilkins-Lee) gas relation,

$$p_{bubble}(t) = Ae^{-R_1(\frac{V(t)}{V_I})} + Be^{-R_2(\frac{V(t)}{V_I})} + C(\frac{V(t)}{V_I})^{-(\omega+1)}, \tag{2}$$

where V_I is the volume of the bubble at the radius R_I, and A, B, R_1, R_2, R_I, and ω are standard constants for explosive material, and finally C is an arbitrary constant which can be determined from initial pressure and volume by letting $V(t) = V_0$, $p_{bubble}(t) = p_{bubble0}$ at $t = 0$.

The material surface (bubble interface) is tracked by the following level set equation,

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \tag{3}$$

where ϕ is the level set function and $\mathbf{u} = (u, v)$ is the external velocity field (water velocity). The level set equation (3) states that ϕ is constant along the particle paths. This means that if the zero level set of ϕ is initialized as the material surface (bubble interface) between water and gas, then the zero level set will always be a material surface at later times [19]. We take $\phi < 0$ in the gas region and $\phi > 0$ in water region (Figure 1). Hence, we have

$$\phi(\mathbf{x}, t) \begin{cases} > 0 & \text{if } \mathbf{x} \in \text{water} \\ = 0 & \text{if } \mathbf{x} \in \Gamma \\ < 0 & \text{if } \mathbf{x} \in \text{gas}, \end{cases} \tag{4}$$

where Γ represents the material surface (bubble interface), and it is defined as the zero level set of ϕ,

$$\Gamma = \{\mathbf{x} \mid \phi(\mathbf{x}, t) = 0\}. \tag{5}$$

The the material surface (bubble interface) is reconstructed through the coupled level set and volume-of-fluid method (CLSVOF)[19, 20] which requires that we have to solve the following volume fraction equation in addition to solving the level set equation (3),

$$\frac{\partial F}{\partial t} + \mathbf{u} \cdot \nabla F = 0, \tag{6}$$

where F represents the volume-of-fluid function (volume fraction) and $\mathbf{u}$ is water velocity. We identify water and gas region via the volume-of-fluid function (volume fraction). For instance, let Ω be a computational cell, then according to the value of the volume-of-fluid function (volume fraction) we will decide that either Ω contains water, gas, or both. In other words, if $F(\Omega, t) = 1$, then the region Ω is all water. If $F(\Omega, t) = 0$, then the region Ω is all gas. If $0 < F(\Omega, t) < 1$, then Ω contains both water and gas (Figure 1). Incorporating with the definition of the level set function and making use of the Heaviside function, F can be written as

$$F(\Omega, t) = \frac{1}{\mid \Omega \mid} \int_{\Omega} H(\phi) d\mathbf{x}, \tag{7}$$

where H is the Heaviside function,

$$H(\phi) = \begin{cases} 1 & \text{if } \phi \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

3 Components of the new adaptive solution techniques

In this section, we will describe the components of the new Adaptive Mesh Refinement (AMR) techniques. In the first subsection, we will describe the basic structure of the multi-level grids and give detail information about the grid management. In the second subsection, we will describe the time integration algorithms and point out some algorithmic facts. The third subsection will be dedicated to the synchronization procedure.

3.1 Multi-level grid refinement and management

The AMR methods are based on solving partial differential equations on a sequence of locally refined rectangular grids. Therefore, the grid refinement plays an important role in an AMR algorithm. There are different levels of grid refinement ranging from the coarsest to the finest. Each level is a finite union of rectangular grid patches. In other words, a level G_l for $l = 0, 1, ..., l_{max}$ is defined as

$$G_l = \cup_k G_{l,k}, \tag{9}$$

where $G_{l,k}$'s are the rectangular grids. In the grid hierarchy, grids at different levels must be properly nested. This means that the union of level $l + 1$ grids is contained in the union of level l grids for $l = 0, 1, ..., l_{max}$. Moreover, there must be (except at the physical boundaries) at least one level l cell wide border surrounding each level $l + 1$ grid [1] . At the physical boundaries, the proper nesting is not so strict, i.e, grids from all levels can

5

extend to the physical boundaries. When a fine grid has more than one parent grid, the fine grid can cross a coarser grid boundary and still be properly nested. We refer Figure 2 for a clear illustration of a properly nested grid.

The grids are generated and refined depending on evolving flow conditions. In other words, a user specified error procedure is employed in order to identify the regions where additional refinement is required, and delete the formerly fine grids which are not needed anymore. In our case, grids are needed to be refined around the bubble interface and the shock front. Our error procedure is as follows. We make use of the level set function when refining the grids around the bubble interface. For instance, we tag a cell if $|\phi_{i,j}| < \Delta x$. Figure 3 illustrates the tagging procedure. The refinement around the shock region is done by the following criteria,

$$\sigma = \sqrt{(p_{i+1,j} - p_{i-1,j})^2 + (p_{i,j+1} - p_{i,j-1})^2}, \qquad (10)$$

where $p_{i,j}$ represents water pressure at $(i,j)^{th}$ cell. Here we tag the cells in which σ is greater than some cutoff value. In addition to the tagged cells according to the above given criteria, we tag more cells in the vicinity of the tagged cells. These additionally tagged cells are called the "*buffer cells*". Adding buffer cells insures that the discontinuities do not propagate out from a fine grid into coarser regions before the next regridding time. After the tagging process is done, rectangular grid patches are created by using the clustering algorithm which is defined in [6]. The grid patches do not contain only tagged cells. They will contain also untagged cells. We don't want to include too many untagged cells, thus we define the *grid efficiency* as the ratio of the tagged cells to the untagged cells that would appear on a level. A typical grid efficiency is taken about 60 %. After the rectangular grid patches are created, they are refined to generate the next level of grids by obeying the proper nesting requirement. The refinement ratio that we use is $r = 2$, i.e, in x-coordinate direction,

$$\Delta x^{l+1} = \frac{1}{r}\Delta x^l. \qquad (11)$$

The initial grid generation at all levels (from the coarsest to the finest) uses the initial data at $t = 0$. Because of the changing flow conditions, grids except the coarsest one are redefined at every user specified number of time steps. For instance, level $l + 1$ grids are modified at the end of every k_l level l time steps. Since we subcycle in time, e.g, $\Delta t^{l+1} = \frac{1}{r}\Delta t^l$ with $r = 2$, grids at $l + 2$ can be created or modified in the middle of a level l time step if $k_{l+1} < r$. When the new grids are created at level $l + 1$, we check if these new level $l + 1$ grids overlap with the previous level $l + 1$ grids. If the overlapping occurs, then the information from the overlapping previous $l + 1$ grids is simply copied into the newly created $l+1$ grids. For the non-overlapping regions, the information from the level l grids is interpolated into the new $l + 1$ grids.

The boundary conditions for a given grid is provided either by physical boundary conditions or by ghost cells which are seeded in a narrow band around the grid. In our case we use a band consisting of three ghost cells as in [1]. The ghost values around a

fine grid are defined as time and space interpolation of the data from the previous coarse grid.

3.2 Multilevel time integration algorithm

Here, first we will outline the basic structure of our recursive AMR algorithm. Then we will define the semi-implicit time integration algorithms. Lastly, we will briefly describe our interface evolution procedure.

Basic structure of the AMR algorithm:

Define all levels from $l = 0$ to $l = l_{max}$ with the initial data at $t = 0$.
Recursive procedure Advance (level l)
    if time to regrid at level $l + 1$
        Estimate errors at level $l + 1$
        Generate new grids at level $l + 1$
        if $l + 1 < l_{max}$ then regrid $l + 2$
    endif
    if $l = 0$ obtain boundary data from physical boundaries
    else obtain boundary data from coarser grids and physical boundaries.
    Integrate level l in time with the semi-implicit algorithm.
    if $l < l_{max}$
        for $k_l = 1, ..., r_{timecycle}$
            Advance (level $l + 1$)
        end
    endif
    Synchronize the data between levels l and $l + 1$.
  End recursive procedure Advance

Our recursive adaptive procedure is the same as described by [15, 1] except that our integration and synchronization steps are different (see section 3.3).

3.2.1 The semi-implicit algorithms

In this section, we present two versions of the semi-implicit techniques. The first version is a non-conservative semi-implicit technique and is similar to [31] except that we don't use CIP procedure(refer to [31]) to solve the advective quantities. The second version is a conservative one, and it is based on the work by Wesseling *et al* [23, 26].

3.2.2 Outline of semi-implicit algorithms

The general outline of our semi-implicit algorithms on a single level is as follows:

1. Calculate provisional values for velocity, density and pressure by computing the non-linear advective force terms.

2. Calculate the new pressure via an implicit procedure in which one solves a Helmholtz equation for p^{n+1}.

3. Update the new velocity and density, extrapolate the velocity and density from liquid regions into gas regions.

4. Update the location of the gas-liquid interface.

3.2.3 Non-conservative semi-implicit algorithm

The non-conservative semi-implicit algorithm consists of two separate phases, advection and non-advection phases. During the advection phase, provisional advective quantities for density and velocity are calculated by solving the following equation,

$$\frac{\partial S}{\partial t} + \mathbf{u} \cdot \nabla S = 0,$$

for some quantity S. The advective pressure is computed by the Tait equation of state (32), i.e, $p^{advect} = p_{Tait}(\rho^{advect})$. After the advection phase, an implicit algorithm for calculating the updated pressure is applied.

In our discretization, advective velocities, $\mathbf{u}$, are face-centered and the other fluid variables (i.e. the advective quantities S) are cell-centered.

Advection phase:

$$\frac{\rho^{advect} - \rho^n}{\Delta t} + [\mathbf{u} \cdot \nabla \rho]^n = 0, \tag{12}$$

$$\frac{\mathbf{u}^{advect} - \mathbf{u}^n}{\Delta t} + [\mathbf{u} \cdot \nabla \mathbf{u}]^n = 0. \tag{13}$$

Here advective flux terms are evaluated by a second order van Leer [24] slope limiting procedure.

Non-advection phase:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^{advect}_{edge}}{\Delta t} = -\frac{1}{\rho^{advect}} \nabla p^{n+1}, \tag{14}$$

$$\frac{p^{n+1} - p^{advect}}{\Delta t} = -\rho^{advect} c^2(\rho^{advect}) \nabla \cdot \mathbf{u}^{n+1}, \tag{15}$$

$$\rho^{n+1} = \rho^{advect} e^{-\nabla \cdot \mathbf{u}^{n+1} \Delta t}. \tag{16}$$

Taking the divergence of equation (14) and replacing $\nabla \cdot \mathbf{u}^{n+1}$ terms in equation (15), we obtain the following elliptic system for the pressure,

$$\nabla \cdot \left(\frac{1}{\rho^{advect}} \nabla p^{n+1} \right) - \frac{1}{\Delta t^2 \rho^{advect} c^2 (\rho^{advect})} p^{n+1} \quad = \quad \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^{advect}_{edge} - \tag{17}$$

$$\frac{1}{\Delta t^2 \rho^{advect} c^2 (\rho^{advect})} p^{advect},$$

where $\mathbf{u}^{advect}_{edge} = \mathbf{u}^n_{edge} + I^{edge}_{cell} (\mathbf{u}^{advect} - \mathbf{u}^n)$ with I^{edge}_{cell} denoting interpolation from cell-centered to edge-centered (or face-centered in 3D) velocity. For each time step, Δt, the new pressure field is computed by solving the implicit system (18), then the new velocity and density fields are explicitly computed via (14) and (16). We note that at a given level, (18) is solved at all levels above simultaneously to ensure the pressure continuity across the coarse/fine grids. The simultaneous solution of (18) requires the calculation of the divergence of the advective velocities at all levels above a given level. The advective velocities on finer levels are calculated by interpolating the advective velocities from the current level.

3.2.4 Conservative semi-implicit algorithm

Below, we describe the conservative semi-implicit method. As reported in [23, 26], this method can handle both compressible or weakly compressible flows. Its conservative form implies correct shock solutions when the flow is compressible, and its implicit pressure treatment ensures stable solutions for weakly compressible flows or flows at the incompressible limits. The method consists of two steps. In the first step, the *prediction step*, intermediate momentum values are predicted. In the second step, the *correction step*, a pressure correction is postulated. Solving the pressure correction equation, the new pressure field is calculated. Then the other field variables are updated accordingly.

Prediction step:
$$\frac{m^*_\alpha - m^n_\alpha}{\Delta t} + (m^*_\alpha u^n_\beta)_{,\beta} = -p^n_{,\alpha}, \tag{18}$$

where α and β denote the values and derivatives in x, y, or z space directions. For instance, if the $\alpha = x$, then m_α represents the x-momentum, and $p_{,\alpha}$ represents the x-derivative of the pressure field.

Correction step:
$$\frac{m^{n+1}_\alpha - m^*_\alpha}{\Delta t} = -\delta p_{,\alpha} \tag{19}$$

where $\delta p = p^{n+1} - p^n$. This postulate is substituted in to the continuity equation

$$\frac{\rho^{n+1} - \rho^n}{\Delta t} + (m^*_\alpha - \Delta t \delta p_{,\alpha})_{,\alpha} = 0. \tag{20}$$

Making use of the implicit form of the equation of the state, i.e, $\rho_t = \frac{1}{c^2}p_t$ and $\frac{\rho^{n+1} - \rho^n}{\Delta t} = \frac{1}{c^2}\frac{p^{n+1} - p^n}{\Delta t}$, we obtain

$$(\delta p_{,\alpha})_{,\alpha} - \frac{1}{c^2 \Delta t^2}p^{n+1} = \frac{1}{\Delta t}(m_\alpha^*)_{,\alpha} - \frac{1}{c^2 \Delta t^2}p^n. \tag{21}$$

Solving the equation (21) for δp, we update the pressure immediately with $p^{n+1} = \delta p + p^n$. Then the new momentums are calculated by $m_\alpha^{n+1} = m_\alpha^* - \Delta t \delta p_{,\alpha}$.

3.2.5   Evolution of the material surface (bubble interface)

The level set equation (3) and the volume fraction equation (6) are solved in a narrow band about the zero level set. The narrow band velocity is formed by extending the water velocity into the gas region. The extension procedure is described in [21] in detail. The interface equations, the level set equation (3) and the volume fraction equation (6), are solved by the second order coupled level set volume-of-fluid method (CLSVOF))[20, 19]. The CLSVOF uses second order operator split (Strang-Splitting) advection algorithms for both the level set function and the volume fractions [20, 19]. The discrete level set function $\phi_{i,j}^n$ and the discrete volume fraction function $F_{i,j}^n$ are located at cell centers. The motion of the material surface is determined by face-centered velocities. At every time steps, the level set function has to be reinitialized as a signed distance function in order to preserve volumes. In the CLSVOF method, the volume fractions are used to construct a volume preserving distance function, and the level set function is used to truncate the volume fractions, i.e, the truncation step removes spurious volumes. For more algorithmic details about CLSVOF method, we refer to [20, 19].

3.2.6   Calculation of Δt

We use two different time steps in our calculations and refer to them as $\Delta t_{explicit}$ (explicit time step) and $\Delta t_{semi-implicit}$ (semi-implicit time step). The explicit time step is defined by

$$\Delta t_{explicit} = \frac{\Delta x}{|u| + c}, \tag{22}$$

where c denotes the sound speed. We define c as

$$c^2 = \begin{cases} \frac{\partial p}{\partial \rho} & \text{if } \rho > \rho_c \\ c_\epsilon^2 & \text{if } \rho < \rho_c \ , \end{cases} \tag{23}$$

where ρ_c is the critical density below which water cavitates, and c_ϵ^2 is set to *zero* in most of the calculations except $c_\epsilon^2 = \frac{\partial p}{\partial \rho}|_{\rho = \rho_c}$ when we do subcycling calculations. The Semi-implicit time step is defined by

$$\Delta t_{semi-implicit} = \frac{\Delta x}{|u|}. \tag{24}$$

The explicit time step, $\Delta t_{explicit}$, will be used when we are interested in calculating highly resolved socks. The semi-implicit time step, $\Delta t_{semi-implicit}$, will be used when simulating the evolution of the material interface. We remark that at the very first time step regardless of whether we are using $\Delta t_{explicit}$ or $\Delta t_{semi-implicit}$, we always use $\Delta t_{explicit}$. Detailed discussion about the employment of both time steps is given in Sections 4.1 and 4.2.

3.3   Synchronization procedure

The synchronization procedure is performed at the end of a level l time integration step, i.e, all finer levels above have been updated. We execute the synchronization step in a way that first we average down the cell/face centered level $l + 1$ data to the level l grids;

$$U^l \longleftarrow \frac{1}{r^2} \sum \sum U^{l+1} \tag{25}$$

where U^{l+1} represents a computed flow variable from level $l + 1$, and r is the mesh refinement ratio. Then we solve the following synchronization equations in the underlying coarse regions by supplying Neumann pressure boundary data from the next fine level (Figure 4).

We consider,

$$\mathbf{u}^{n+1} = \mathbf{u}^{n+1,old} - \Delta t \frac{1}{\rho^{advect}} \nabla \delta p, \tag{26}$$

$$p^{n+1} = p^{advect} - \Delta t \rho^{advect}(c^{advect})^2 \nabla \cdot \mathbf{u}^{n+1}, \tag{27}$$

$$\rho^{n+1} = \rho^{advect} - \Delta t \rho^{advect} \nabla . \mathbf{u}^{n+1}, \tag{28}$$

where the superscript $(n + 1, old)$ represents the values computed on level l grids at the end of level l time steps, and $\delta p = p^{n+1} - p^{n+1,old}$. Substituting the divergence of the new velocity from the equation (26) into the equation (27), we obtain the following equation for the pressure correction,

$$\frac{\delta p}{\Delta t^2 (\rho c^2)^{advect}} - \nabla \cdot (\alpha \frac{\nabla \delta p}{\rho^{advect}}) = -\frac{\nabla \cdot \mathbf{u}^{n+1,old}}{\Delta t} + \frac{p^{advect} - p^{n+1,old}}{\Delta t^2 (\rho c^2)^{advect}}, \tag{29}$$

where α is defined as, e.g., in the x-direction,

$$\alpha_{i+\frac{1}{2},j} = \begin{cases} 10^{-3} & \text{if } x_{i+1,j} \text{ and } x_{i,j} \text{ in fine grids} \\ 1 & \text{otherwise.} \end{cases} \tag{30}$$

We note that the right hand side of the equation (29) is identically *zero*, for a uniform mesh (versus an adaptive hierarchy of levels) since the right hand side of (29) is identical

11

to (15). We also remark that we must specify a cutoff for α for otherwise (incompressible flow case) the resulting linear system for δp might violate the solvability condition. The coefficient α effectively induces a Neumann boundary condition, $\nabla \delta p \cdot \mathbf{n} = 0$, at coarse/fine grid borders. This way the velocity covered by finer levels is not modified.

4   Numerical results

4.1   Spherical explosion-bubble growth and collapse

A region of high-pressure JWL gas, with a radius of 16 cm, fills a small bubble at the lower left corner of a two dimensional domain as shown in Figure 5. The initial material states for this problem are

JWL: $\rho = 1.63 g/cm^3$, $p = 7.8039E + 10 d/cm^2$

Water: $\rho = 1.00037984 g/cm^3$, $p = 1.0E + 7 d/cm^2$, $\mathbf{u} = 0.0$.

The high pressure bubble initiates a spherically propagating shock wave. While the shock front rapidly leaves the region, the bubble continues to grow. Figure 6, produced by using our non-conservative semi-implicit method, depicts the physics of the principal flow variables. Figure 6 is created at $t = 1 msec$ using 128×128 base grid with two levels of refinement. The computational domain size is taken as 256.0×256.0. And the adaptivity cutoff value for the equation (10) is set to $4.0E + 6 d/cm^2$. From the contour plots of density, pressure, x-velocity, and y-velocity in Figure 6, we observe that the propagation of the shock front and the motion of the bubble interface are solved without excessive oscillations. This is also evident by Figure 7 (which is the x-slice of the perssure field). In Figure 7, we compare water pressure obtained by the non-conservative semi-implicit, conservative semi-implicit, and an explicit two phase method due to Wardlaw [25]. Figure 7 indicates that both the non-conservative and the conservative semi-implicit methods are comparable to the two phase explicit method. We note that the shock amplitude is higher with the two phase explicit method. We attribute this difference to the fact that we don't solve the gas dynamics inside the bubble whereas the two phase explicit method does. One key observation about our non-conservative semi-implicit technique is that it computes the evolution of the shock wave as good as its conservative semi-implicit counterpart. In other words, the non-conservative semi-implicit method gives comparable results for shock amplitude and shock speed as the conservative semi-implicit method. We remark that if we modeled the water with a non-adiabatic equation of state, then the non-conservative approach would give the incorrect shock speed. We have observed for gas dynamics, that the non-conservative approach overpredicts the shock speed by 25% for a non-adiabatic equation of state.

Figure 8 compares the bubble radii when calculated using the non-conservative and conservative semi-implicit approaches respectively. It is clear from this figure that the both methods capture the bubble interface identically. In Figure 9, we provide a convergence study. We compare 128×128 single grid calculations to the one level of refinement

and two levels of refinement results. Here we notice that the shock representations get sharper for both methods when the grids are refined; we also observe that the relative error between successive grid resolutions is decreasing which indicates our method converges as the grid is refined.

We computed the problem illustrated in Figure 8 beyond $t = 1msec$. This time we set the computational domain size to 6000.0×6000.0 and the adaptivity cutoff parameter to $4.0E + 7d/cm^2$. The bubble ultimately reaches its maximum radius and then starts collapsing. Figure 10 represents *one* collapse cycle. The bubble reaches its maximum radius around $t = 66msec$ and bounces back to a minimum radius before expanding again at around $t = 132msec$. In Figure 10, we compare the bubble radii obtained by the non-conservative semi-implicit method, conservative semi-implicit method, and an explicit method due to Wardlaw [25]. Our adaptive calculations are done on a 64×64 base grid with *six* levels of refinement. Figure 10 indicates that both non-conservative and conservative semi-implicit methods approximate the bubble radius comparable to the explicit method. For the calculations indicated in Figure 10, the time step is significantly larger than what would be used by an explicit method. The time step used for computing bubble growth and collapse was determined only by the maximum velocity; the time step did not depend on the speed of sound waves. We remark that if shock resolution is important, then we must use time steps determined by an "explicit" time step constraint (taking sound speed into account when calculating Δt). Our method remains stable for large time steps, but shock waves are severely diffused. Shock waves are characterized by significantly faster time scales than the time-scale associated with bubble growth and collapse. For measuring the dynamics of bubble growth and collapse, we find that a time step determined only by the underlying velocity field is sufficient to accurately predict the bubble radius.

In section 4.2, we show that the results for the bubble radius using an "explicit" Δt (sound speed taken into account) are comparable to the results when using a "semi-implicit" Δt.

Below we make an efficiency comparison for our adaptive techniques. In order to produce the radii in Figure 10, it took 1930 time steps with the non-conservative semi-implicit method, 2028 time steps with the conservative semi-implicit method, and 97540 time steps when using a time step determined by an "explicit" criterion. It is clear that a semi-implicit technique for measuring bubble-growth and collapse are far more efficient than an explicit method.

In Figure 11, we provide a convergence study. Taking 64×64 as a base grid, we perform two calculations with *six* and *seven* levels of refinement, respectively. Figure 11 shows that our approximations using either semi-implicit method get better with the refined mesh.

4.2   Bubble jetting

In this section, we present the computational results of the underwater explosion taking place under a flat plate. The problem description is as follows. An explosive device

13

weighing $17.08 grams$ and located below a flat plate is detonated at a depth of $98.5 meters$. A diagram of this problem is shown in Figure 12. The plate is circular with a radius of $88.9 cm$ and has a thickness of $10.0 cm$. The initial material states for this problem are:

The initial water density is taken to be 1.00039080 g/cm^3.

The initial water pressure is $1.026E + 07 d/cm^2$

Note that the water initial state corresponds to a depth of $98.5 meters$. As a consequence of gravity, the water density decreases linearly to 1 at the surface. Water density as a function of depth is calculated by

$$\rho(y) = 1 + (\rho_{dep} - 1)(h + k - y)/h, \tag{31}$$

where ρ_{dep} is water density at the depth where the explosive is located, i.e, $\rho_{dep} = 1.00039080 g/cm^3$, h is the distance between the bottom and the surface of the sea, specifically $h = 200 meters$, k is the distance between the bottom of the sea and the location of the explosive.

The water pressure is calculated by the Tait equation of state [25], i.e,

$$p = \begin{cases} p_c & \text{if } \rho < \rho_c \\ B[(\rho/\overline{\rho})^\gamma - 1] + A & \text{o.t} \end{cases} \tag{32}$$

where $B = 3.31E + 09 d/cm^2$, $A = 1.0E + 06 d/cm^2$, $\overline{\rho} = 1.0 g/cm^3$, and $\gamma = 7.15$. The cavitation pressure and density for water are $p_c = 220.2726 d/cm^2$ and $\rho_c = 1.0 - 4.225E - 5 g/cm^3$.

The initial water velocity is taken to be 0.0.

The initial bubble pressure is taken as $P_0 = 8.3837E + 10$ d/cm^2. The initial bubble radius is 1.36 cm. The JWL constants (refer to equation 2) applicable to this problem are $A = 3.71E + 12$, $B = 0.03231E + 12$, $\omega = 0.3$, $R_1 = 4.15$, $R_2 = 0.95$, and $\rho_0 = 1.63 g/cm^3$. This problem is cast in 2-D cylindrical coordinates and uses a gravitational constant of 981 cm/s^2.

At the boundaries of the computational domain, we specify outflow conditions for the velocity, and we specify hydrostatic pressure conditions for the pressure. Here, the computational domain is 400.0×400.0, and the adaptivity cutoff parameter is $4.0E + 6 d/cm^2$.

After the initial explosion, the shock wave quickly propagates towards the plate in a spherical fashion. The shock front, hits the plate and reflects back. In Figures 13 and 14, we show the time history of the pressure profile. Notice that around $t = 0.1 msec$, the shock front has already hit the plate and reflected back. Figures 15 and 16 give the time history of the solution for the bubble interface. The bubble interface first grows in a spherical fashion, after interacting with the reflected shocks and after interacting with the plate, the explosive bubble transitions into a toroidal shape.

In Figure 17, we compare the effective bubble radius generated by our adaptive non-conservative and conservative semi-implicit techniques at $t = 15 msec$. The effective radius is the radius of the sphere which corresponds to the volume of the distorted bubble. Figure 17 also shows mesh refinement study for both methods. Wardlaw with

his explicit calculations[25] reported that the maximum effective radius for this problem is approximately $18cm$. From figure 17, we see that our semi-implicit methods also predict the maximum effective bubble radius close to $18cm$. Figure 18 show the results which are calculated with our non-conservative semi-implicit technique by using the explicit time steps. As can be seen from Figures 17 and 18 that the effective bubble radius are almost identical indicating once again that one can choose not to use the explicit time steps when calculating the evolution of the material interface.

We note that all the calculations above are performed without including the subcycling procedure. In the next test, we turn the subcycling procedure on and compare the efficiency of this procedure to the no-subcycling one. We compute the bubble jetting (bubble-plate test) problem on a 320×320 base grid plus *two* levels of adaptivity. The physical domain size is 400.0×4000.0, and the final time is $15msec$. Figure 19 compares the bubble radii calculated with both procedures. The left figure compares the results of subcycling versus no-subcycling procedures from the non-conservative semi-implicit calculations using $\Delta t_{explicit}$. The right figure compares the results of subcycling versus no-subcycling procedures from the non-conservative semi-implicit calculations using $\Delta t_{semi-implicit}$. As can been seen from the figures that the subcycling results are fairly comparable to the no-subcycling ones. As far as the efficiency is concerned, we have the following findings. To produce the left figure, it took 3942 time steps and 126789 seconds of run time for the subcycling calculations, and 15846 time steps and 190866 seconds run time for the no-subcycling calculations. To produce the right figure, it took 270 time steps and 9164 seconds of run time for the subcycling calculations, and 948 time steps and 11962 seconds run time for the no-subcycling calculations. Thus it is evident that that the subcycling procedure improves the run-time of the computations.

5   Conclusions

We presented adaptive solution techniques for simulating underwater explosions and implosions. We solved several test problems to show the performance of our methods. We validated our results by performing grid refinement studies in which we added additional levels of adaptivity. Also we validated our results by comparing shock speed, shock amplitude, and material interface speed, with benchmark results produced by Wardlaw[25]. Also, we compared our two semi-implicit formulations (conservative vs. non-conservative) against each other with good agreement. Finally, we showed that for some specific cases, e.g. investigating bubble growth and collapse dynamics, that the semi-implicit approach is significantly more efficient than an explicit approach. Our semi-implicit approach affords one the flexibility of predicting early time shock dynamics (albeit with a time-step comparable to an explicit method), and at the same time, also predicting ensuing late time bubble dynamics with a much larger time step than an explicit counterpart.

References

[1] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, and M.L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

[2] A.S. Almgren, J.B. Bell, L.H. Howell, and P. Colella. An adaptive projection method for incompressible Navier-Stokes equations. In *In Proceedings of the IMACS 14th World Conference*, July 1994. Atlanta, GA.

[3] J.B. Bell, M.J. Berger, J.S. Saltzman, and M. Welcome. Three Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws. *SIAM J. Sci. Statist. Comput.*, 15:127–138, 1994.

[4] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[5] M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[6] M.J. Berger and I. Rigoustsos. An algorithm for point clustering and grid generation. In *Technical Report NYU-501*, 1991. New York University-CIMS.

[7] R. Caiden, R. Fedkiw, and C. Anderson. A numerical method for two phase flow consisting of separate compressible and incompressible regions. *J. Comput. Phys.*, 166:1–27, 2001.

[8] T.L. Clark and R.D. Farley. Severe downslope windstorm calculations in two and three spatial dimensions using anelastic interactive grid nesting: A possible mechanism for gustiness. *J. Atmos. Sci.*, 41:329–350, 1984.

[9] P. Colella and K. Pao. A projection method for low speed flows. *J. Comput. Phys.*, 149-2:245–269, 1999.

[10] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multi-material flows (the Ghost Fluid Method). *J. Comput. Phys.*, 152:457, 1999.

[11] L.H. Howell and J.B. Bell. An adaptive-mesh projection method for viscous incompressible flow. *SIAM J. Sci. Comput.*, 18:996–1013, 1997.

[12] M.L. Minion. A projection method for locally refined grids. *J. Comput. Phys.*, 127:158–178, 1996.

[13] R. R. Nourgaliev, T. N. Dinh, T. G. Theofanous, and J. M. Koning. Direct numerical simulation of disperse multiphase high-speed flows. *AIAA Aerospace Sciences Meeting*, 2004.

[14] R.B. Pember, L.H. Howell, J.B. Bell, P. Colella, W.Y. Crutchfield, W.A. Fiveland, and J.P. Jessee. An adaptive projection method for unsteady low-Mach number combustion. *Comb. Sci. Tech.*, 140:123–168, 1998.

[15] C.A. Rendleman, V.E. Beckner, M. Lijewski, W.Y. Crutchfield, and J.B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3:147–157, 2000.

[16] W.C. Skamarock and J.B. Klemp. Adaptive grid refinement for two dimensional and three dimensional nonhydrostatic atmospheric flow. *Monthly Weather Review*, 121:788–804, 1993.

[17] E. Steinthorsson, D. Modiano, W.Y. Crutchfield, J.B. Bell, and P. Colella. An adaptive semi-implicit scheme for simulations of unsteady viscous compressible flow. In *In 12th AIAA Computational Fluid Dynamics Conference*, June 1995. Orlando, FL.

[18] D.E. Stevens and C.S. Bretherton. A forward-in-time advection scheme and adaptive multilevel flow solver for nearly incompressible atmospheric flow. *J. Comput. Phys.*, 129:284–295, 1996.

[19] M. Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.*, 187:110–136, 2003.

[20] M. Sussman and E.G. Puckett. A coupled level set and volume-of-fluid method for computing 3D and Axisymmetric incompressible two-phase flows. *J. Comput. Phys.*, 162:301–337, 2000.

[21] M. Sussman, K. Smith, M.Y. Hussaini, M. Ohta, and R.-Z. Wei. A sharp interface method for incompressible two-phase flows. *J. Comput. Phys.*, to appear, 2006.

[22] F. K. L. Tan, B. C. Khoo, and J. White. A level set-boundary element method for the simulation of underwater bubble dynamics. *SIAM J. Sci. Comput.*, submitted for publication, 2006.

[23] D.R. van der Heul, C. Vuik, and P. Wesseling. A conservative pressure-correction method for flow at all speeds. *Comput. Fluids.*, 32:1113–1132, 2003.

[24] B. van Leer. Toward the ultimate conservative difference scheme ii. monotonicity and conservation combined in a second order scheme. *J. Comput. Phys.*, 14:361–370, 1974.

[25] A. Wardlaw. Underwater explosion test cases. In *Technical Report IHTR 2069, ADB238684, Office of Naval Research*, 1998.

[26] P. Wesseling and D.R. van der Heul. Uniformly effective numerical methods for hyperbolic systems. *Computing.*, 66:249–267, 2001.

[27] F. Xiaoi, R. Akoh, and S. Ii. Unified formulation for compressible and incompressible flows by using multi-integrated moments II: Multi-dimensional version for compressible and incompressible flows. *J. Comput. Phys.*, 213:31–56, 2006.

[28] T. Yabe, Y. Ogata, K. Takizawa, T. Kawai, A. Segawa, and K. Sakurai. The next generation CIP as a conservative semi-Lagrangian solver for solid, liquid, and gas. *J. Comput. Appl. Math.*, 149:267–277, 2002.

[29] T. Yabe and P.Y. Wang. Unified numerical procedure for compressible and incompressible flow. *J. Phys. Soc. Japan*, 60:2105–2108, 1991.

[30] T. Yabe, F. Xiao, and T. Utsumi. The constrained interpolation profile method for multiphase analysis. *J. Comput. Phys.*, 169:556–593, 2001.

[31] S.Y. Yoon and T. Yabe. The unified simulation for incompressible and compressible flow by the predictor-corrector scheme based on the CIP method. *Comput. Phys. Commun.*, 119:149–158, 1999.

List of Figures

Figure 1: Representing the bubble interface by the level set function and defining the corresponding volume fractions.

Figure 2: A properly nested grid around a front.
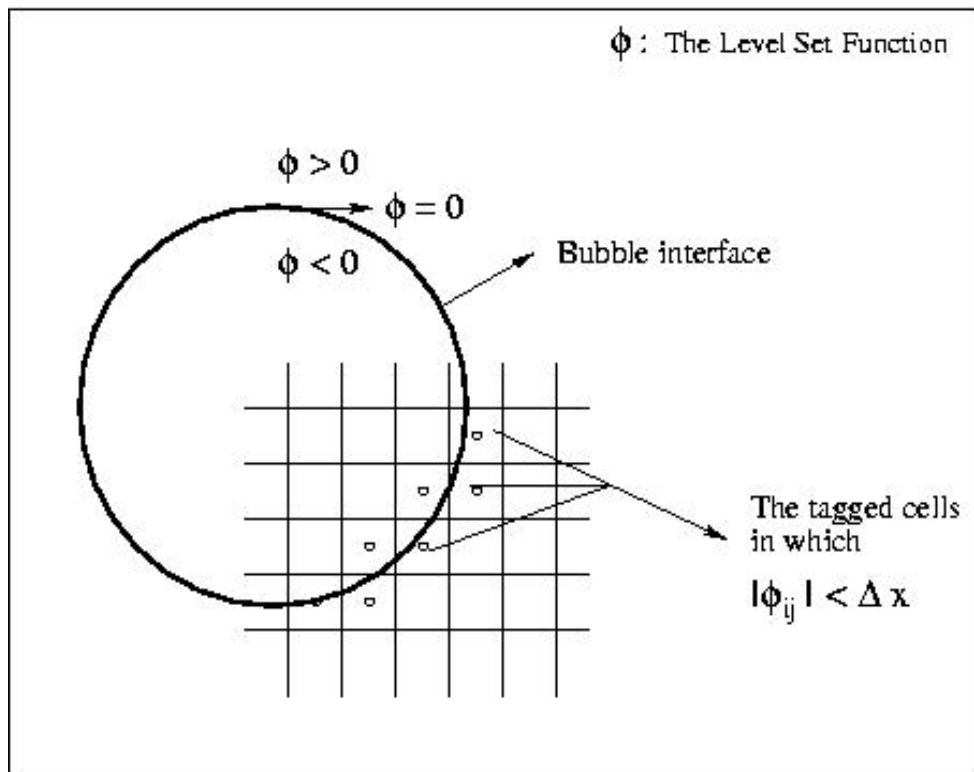
Figure 3: Tagging cells around the bubble interface using the level set function.
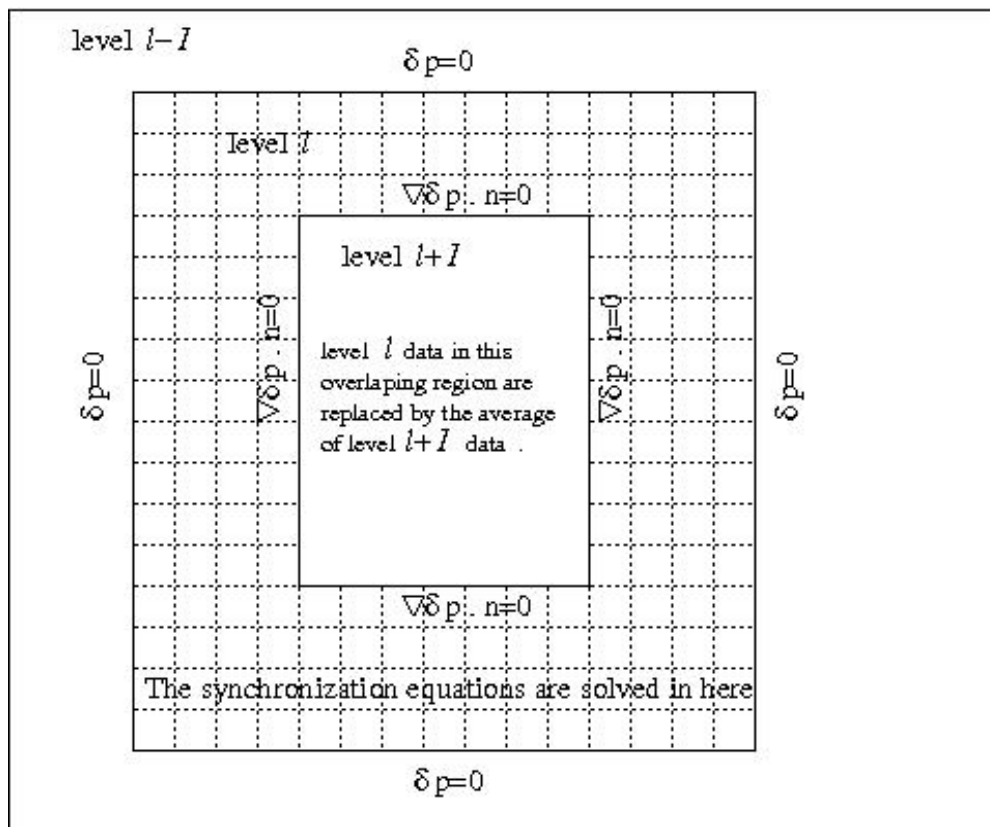
Figure 4: The synchronization step from the level $l+1$ to the level l. The synchronization equations are solved in underlying level l region with the specified boundary data.
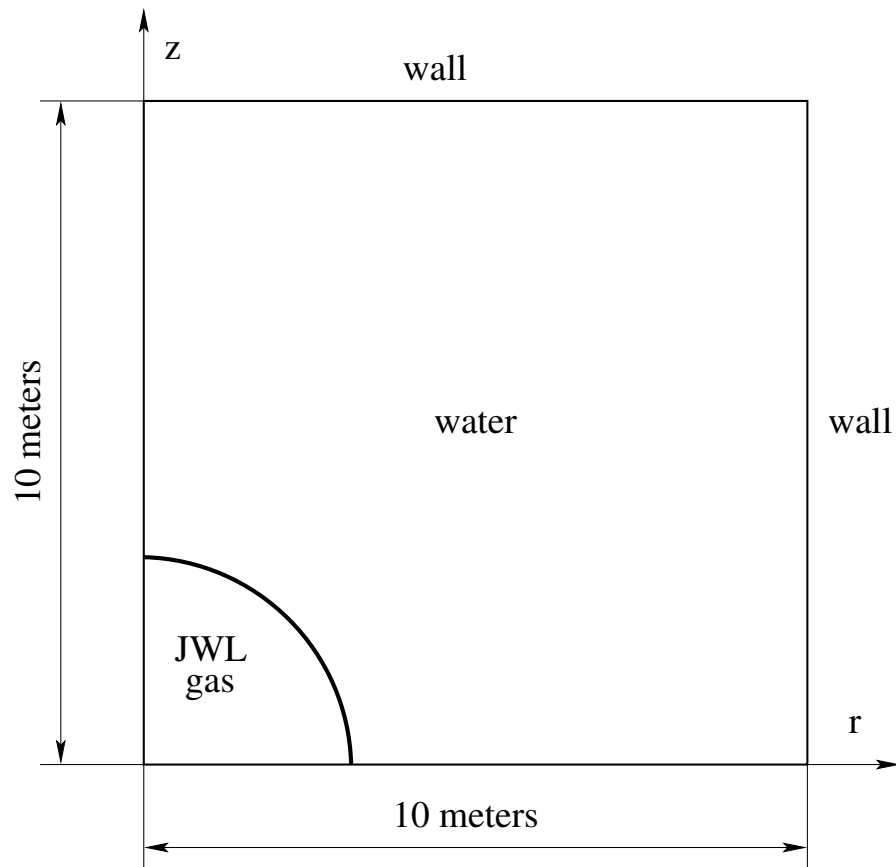
Figure 5: Sketch of the spherical explosion bubble growth problem.
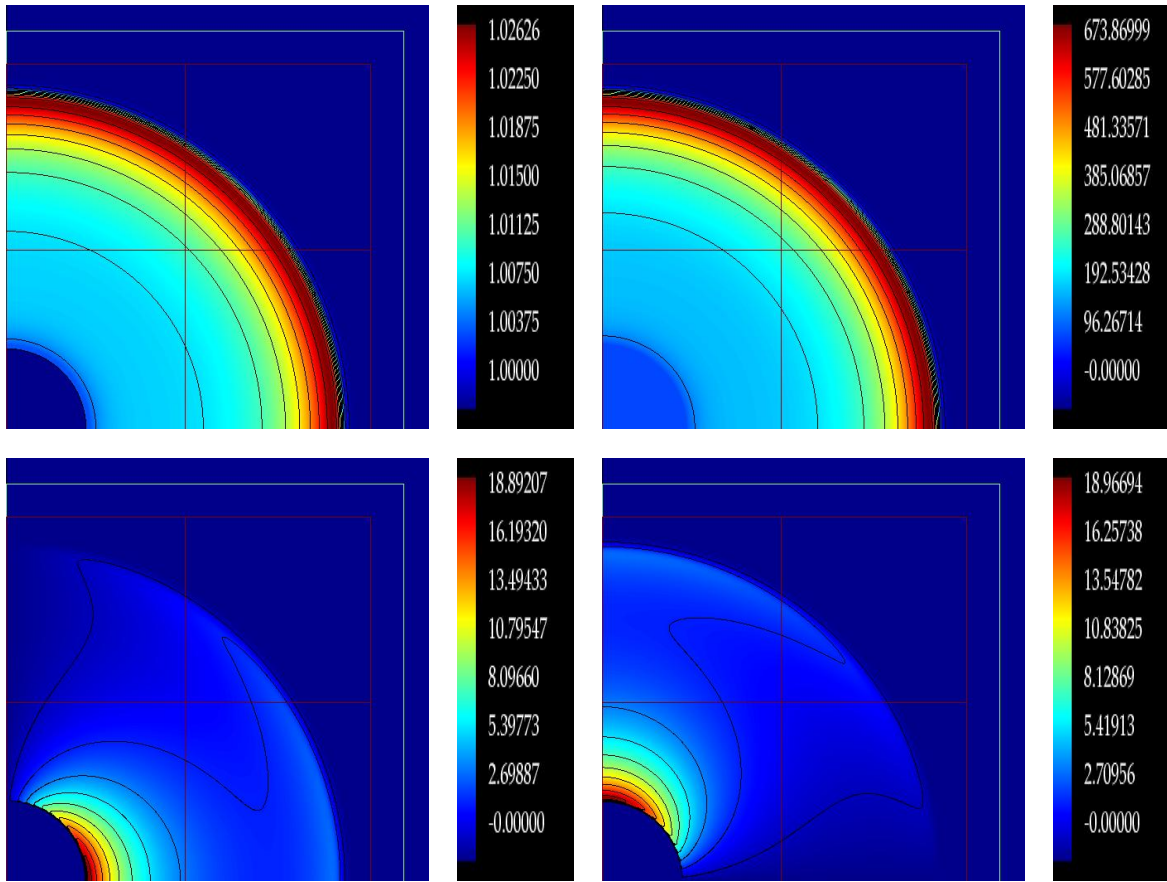
Figure 6: Axisymmetric underwater explosion test results using the non-conservative semi-implicit method at $t = 1$. The upper left figure represents the *density*, the upper right figure represents the *pressure*, the lower left figure represents the *x-velocity*, and the lower right figure represents *y-velocity*.
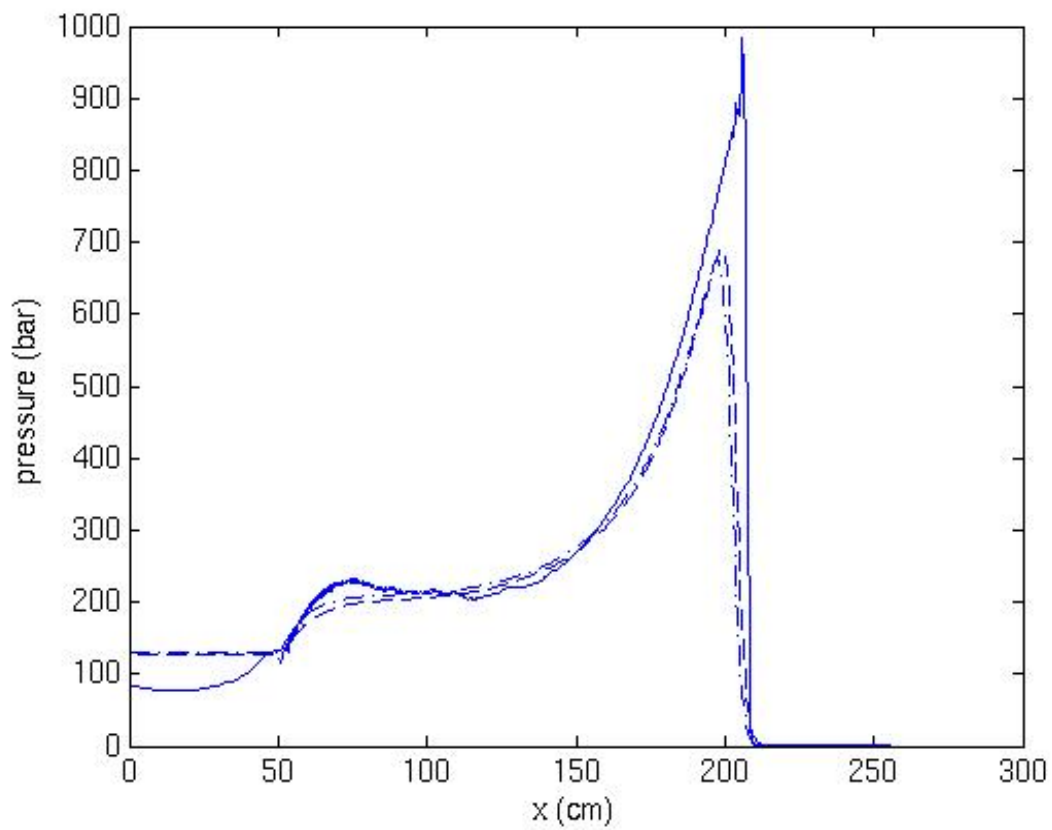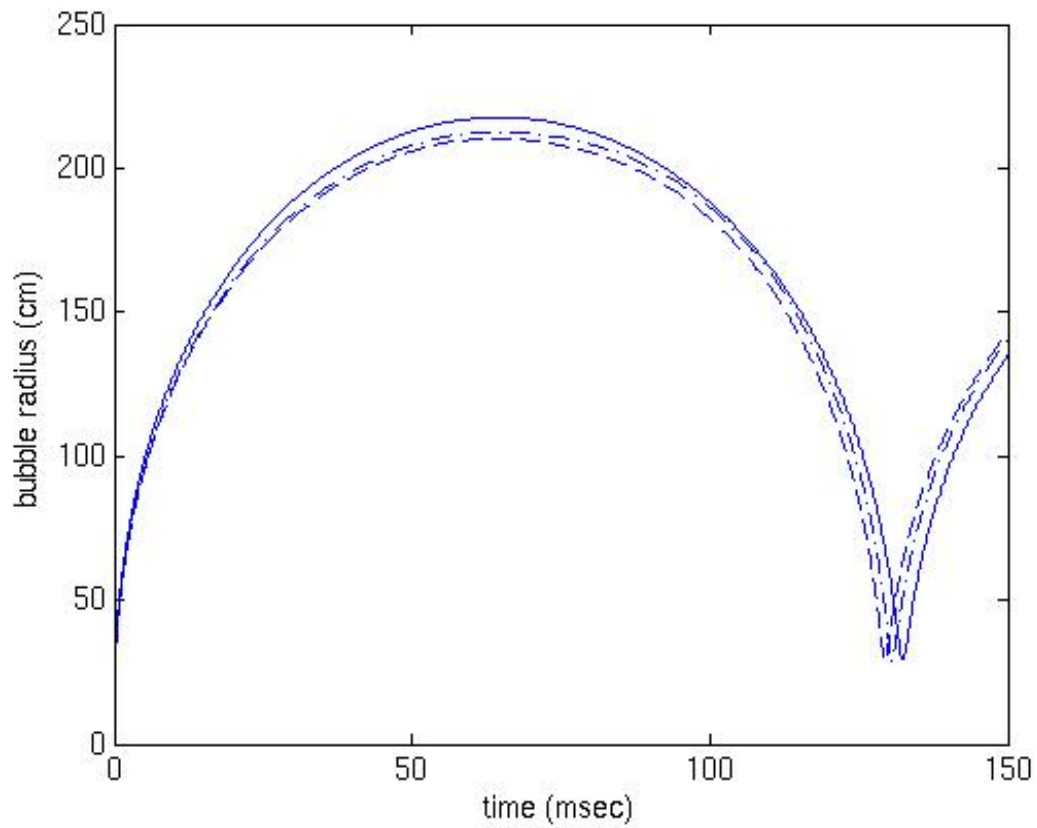
Figure 7: Comparison of the liquid pressure for the bubble growth problem. Solid line represents the result from an explicit method due to Wardlaw *et al*. Dashed line represents the result from the conservative semi-implicit method. Dashed and dotted line represents the result from the non-conservative semi-implicit method. $t = 1$.
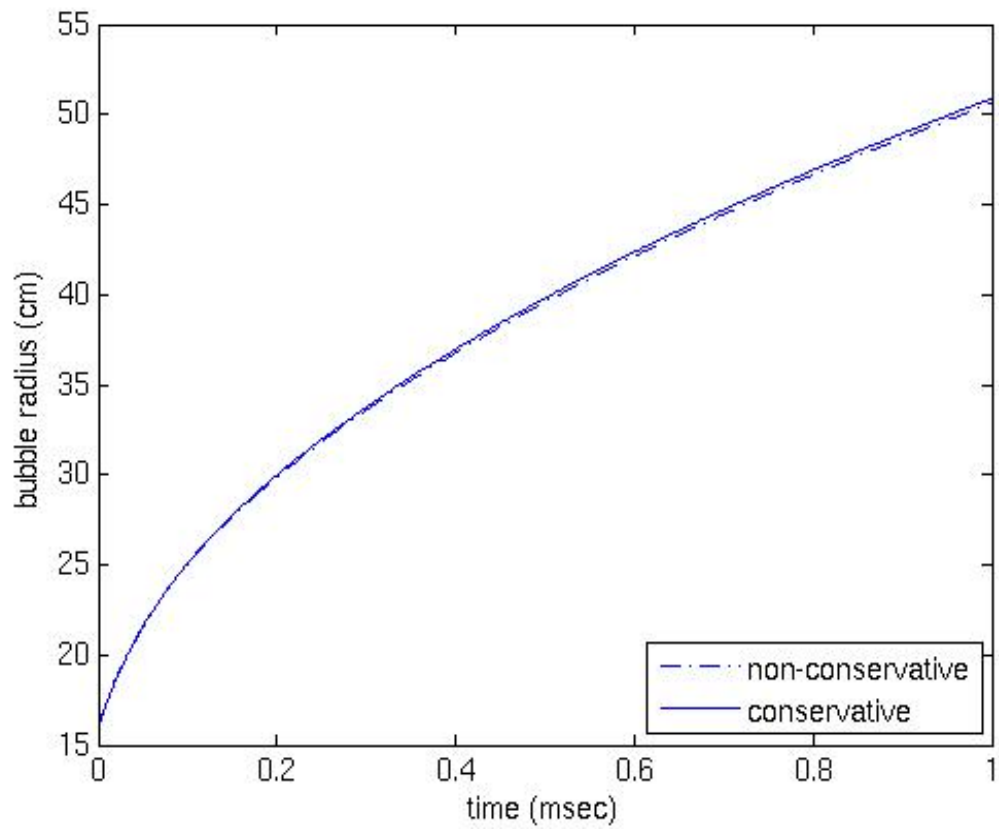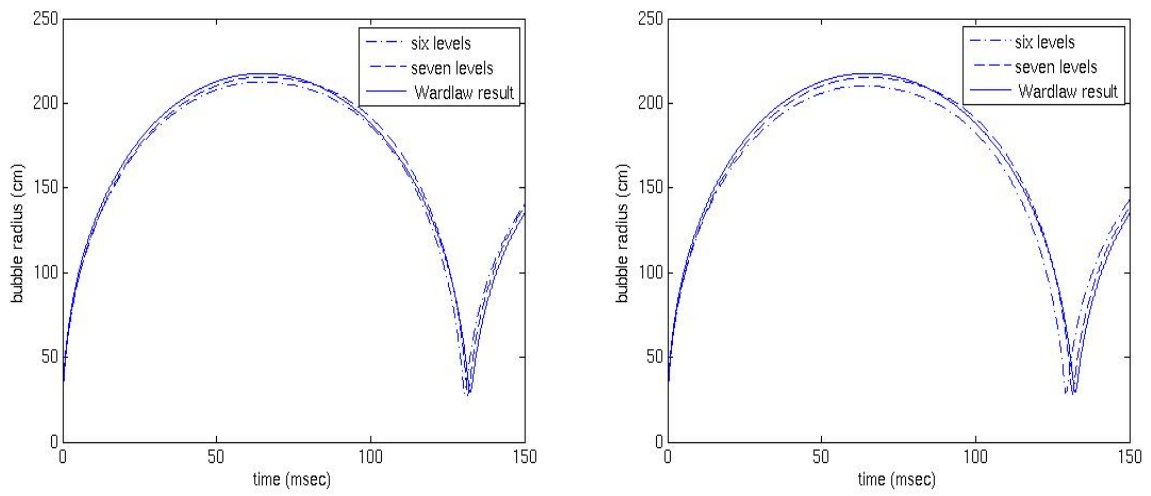
Figure 8: Comparison of the results of the non-conservative semi-implicit and the conservative semi-implicit methods for the bubble radius. $t = 1$.
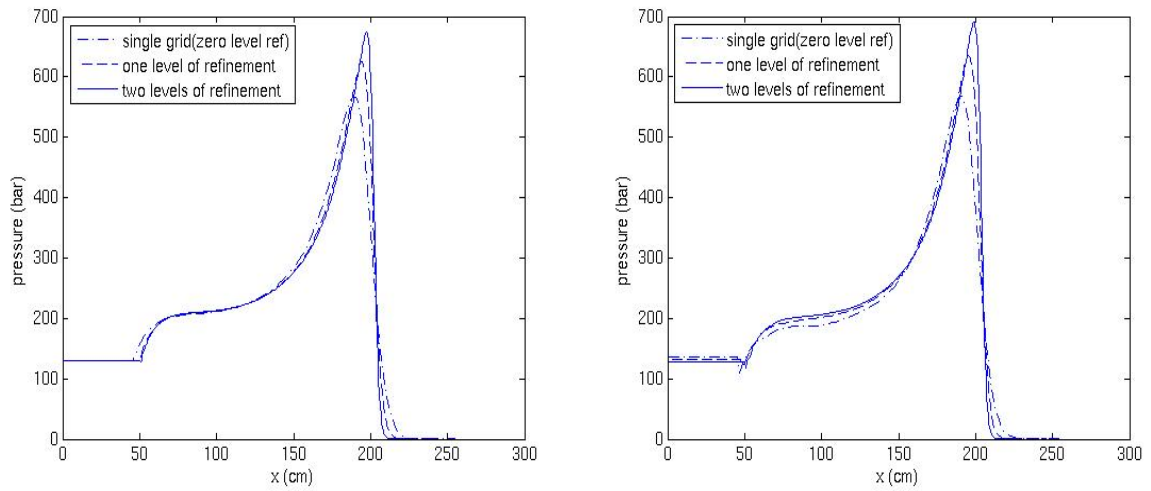
Figure 9: A grid refinement study for the bubble pressure when solving the bubble growth problem. Left figure corresponds the results from non-conservative semi-implicit method. Right figure corresponds the results from conservative semi-implicit method. $t = 1$.

Figure 10: Comparison of the bubble radius for the bubble growth and collapse problem. Solid line represents the result from an explicit method due to Wardlaw *et al.* Dashed line represents the result from the conservative semi-implicit method. Dashed and dotted line represents the result from the non-conservative semi-implicit method. $t = 150msec$.

Figure 11: A grid refinement study for the bubble growth and collapse problem. Left figure corresponds the results from non-conservative semi-implicit method versus the explicit method. Right figure corresponds the results from conservative semi-implicit method versus the explicit method. Base grid for the both refinements is 64×64. $t = 150 msec$.
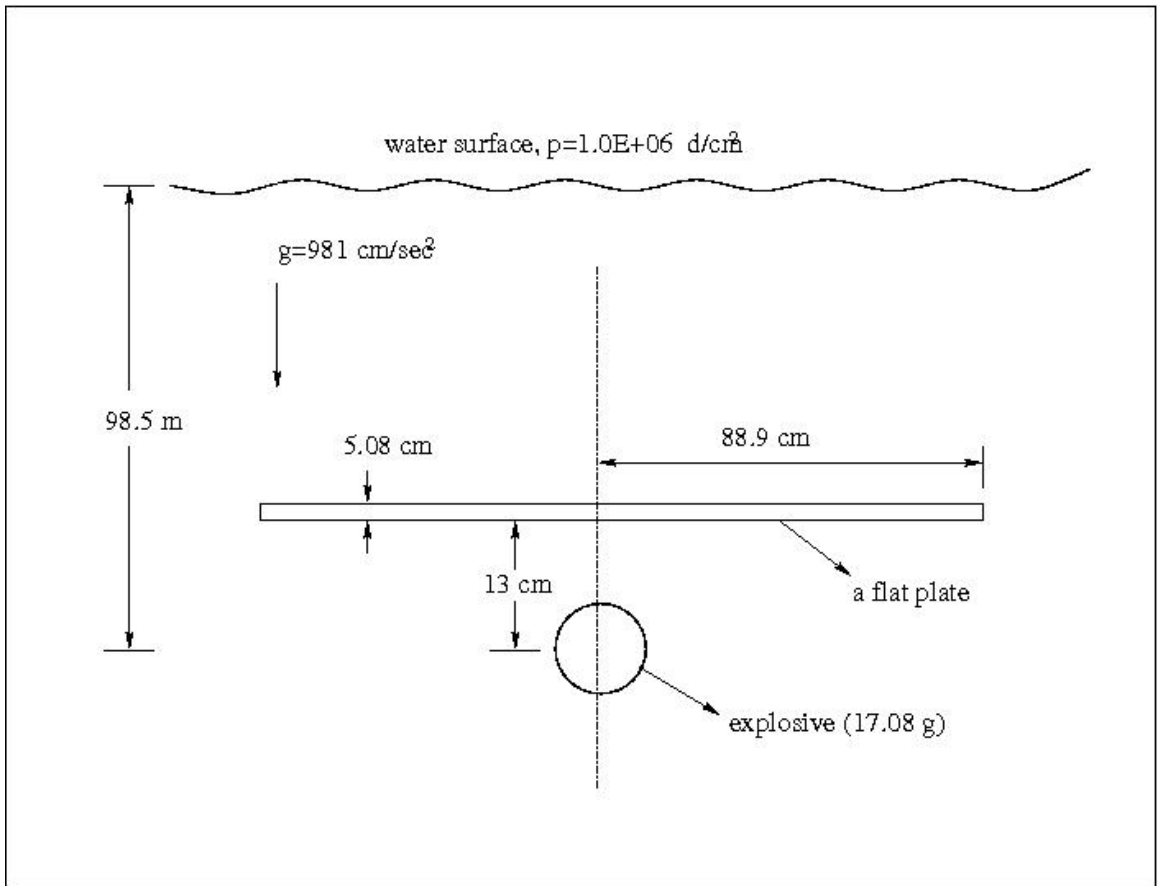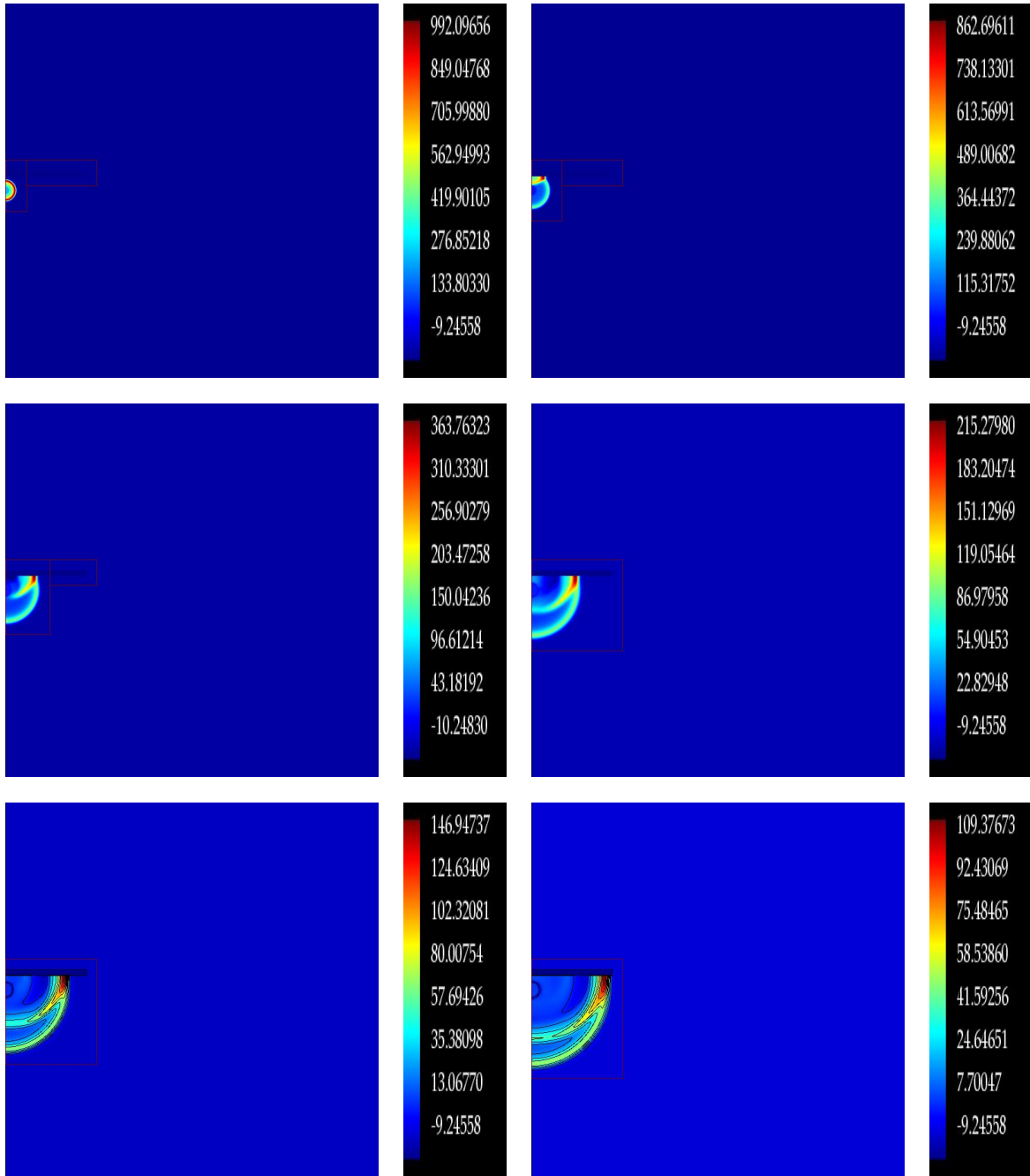
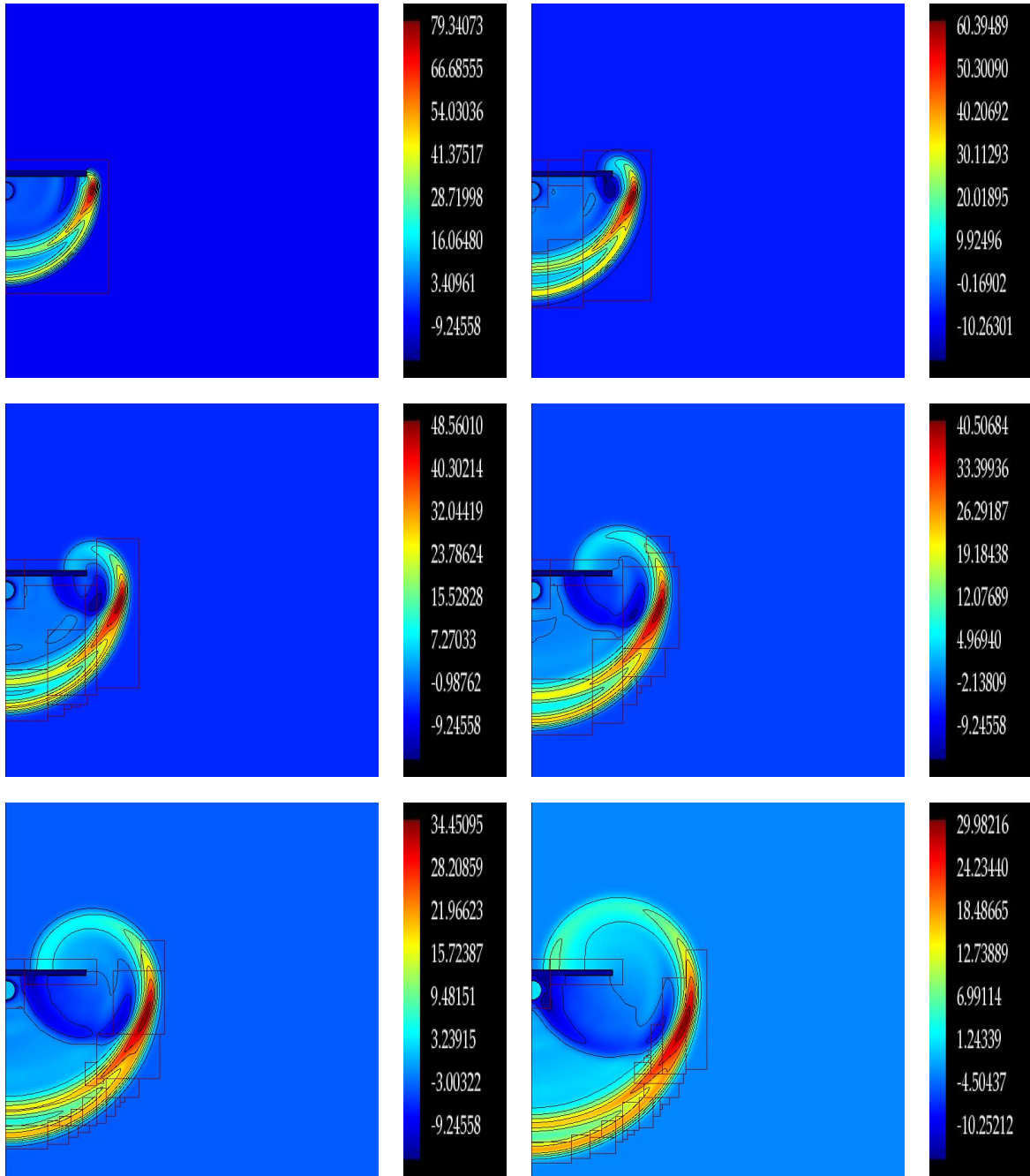Figure 12: Sketch of the underwater explosion test problem

Figure 13: The time history of the liquid pressure profile generated by using the non-conservative semi-implicit method. 320×320 base grid with *two* levels of grid refinement is used for these computations. Time for the upper left figure is $0.05msec$. Time for the upper right figure is $0.1msec$. Time for the middle left figure is $0.2msec$. Time for the middle right figure is $0.3msec$. Time for the lower left figure is $0.4msec$. Time for the lower right figure is $0.5msec$.
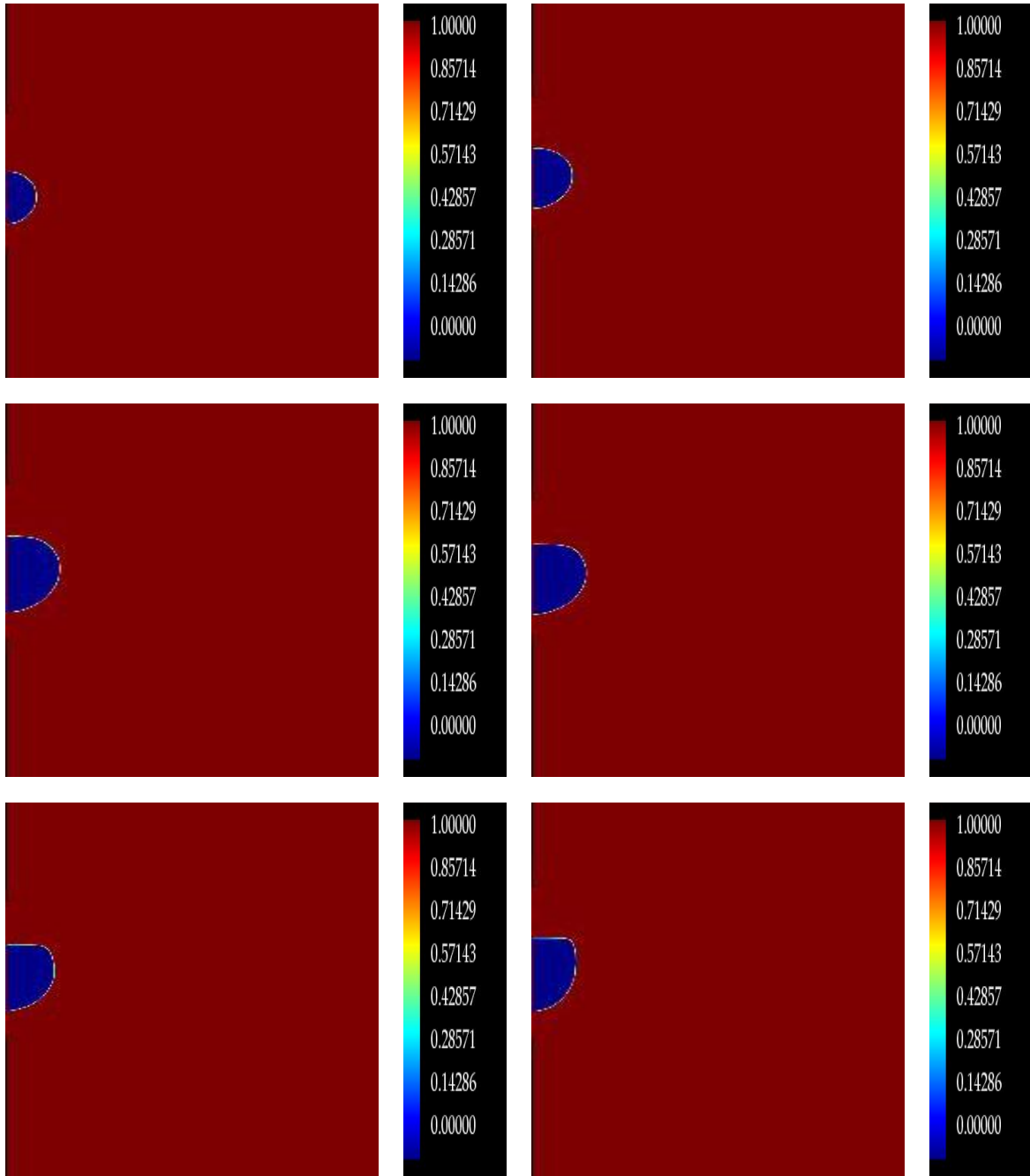
Figure 14: The time history of the liquid pressure profile generated by using the non-conservative semi-implicit method. 320×320 base grid with *two* levels of grid refinement is used for these computations. Time for the upper left figure is $0.6msec$. Time for the upper right figure is $0.7msec$. Time for the middle left figure is $0.8msec$. Time for the middle right figure is $0.9msec$. Time for the lower left figure is $1.0msec$. Time for the lower right figure is $1.1msec$.
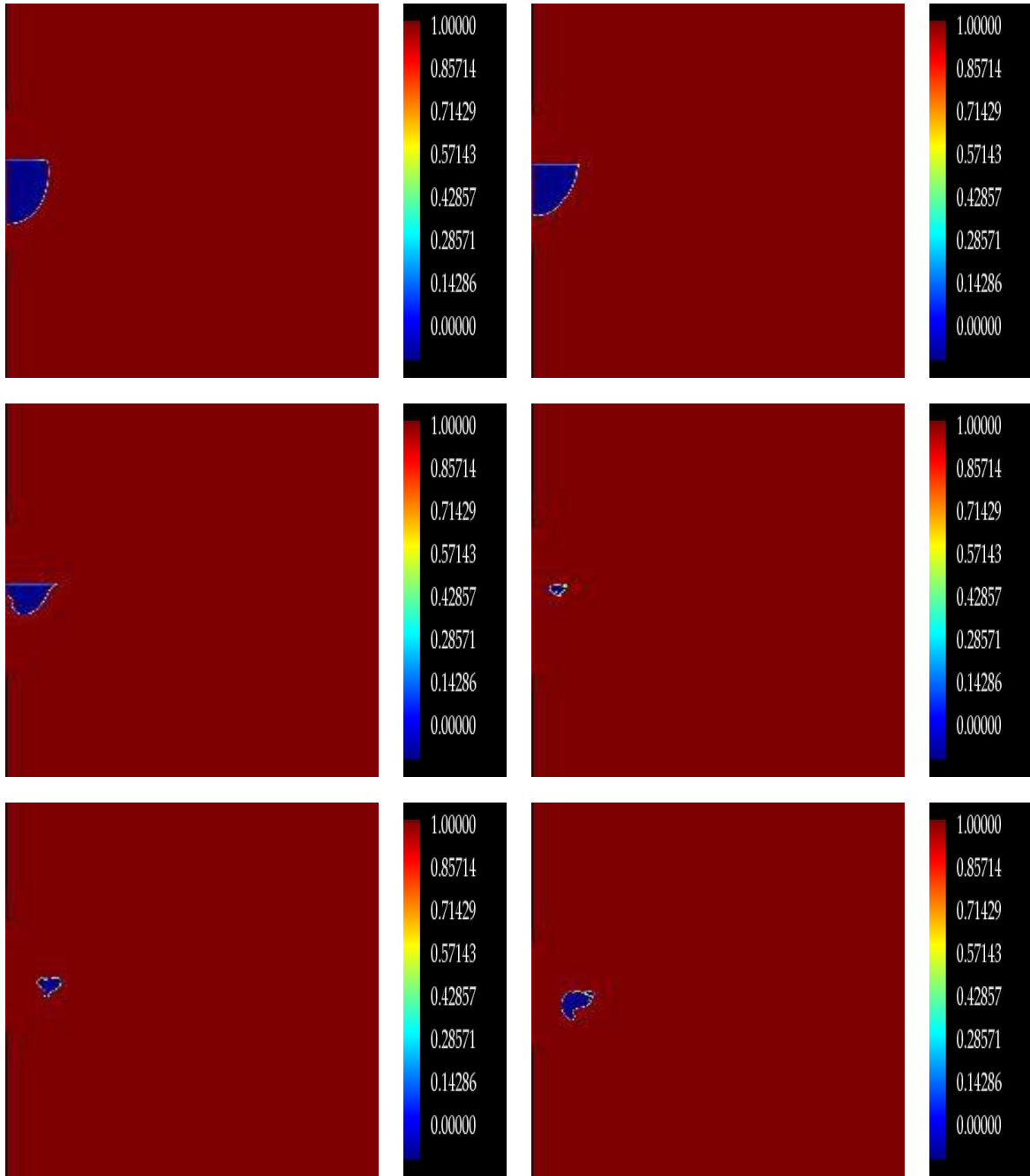
Figure 15: The time history of the bubble interface profile generated by using the non-conservative semi-implicit method. 320×320 base grid with *two* levels of grid refinement is used for these computations. Time for the upper left figure is 1.0*msec*. Time for the upper right figure is 2.0*msec*. Time for the middle left figure is 4.0*msec*. Time for the middle right figure is 6.0*msec*. Time for the lower left figure is 8.0*msec*. Time for the lower right figure is 9.0*msec*.

Figure 16: The time history of the bubble interface profile generated by using the non-conservative semi-implicit method. 320×320 base grid with *two* levels of grid refinement is used for these computations. Time for the upper left figure is $10.0msec$. Time for the upper right figure is $11.0msec$. Time for the middle left figure is $12.0msec$. Time for the middle right figure is $13.0msec$. Time for the lower left figure is $14.0msec$. Time for the lower right figure is $15.0msec$.
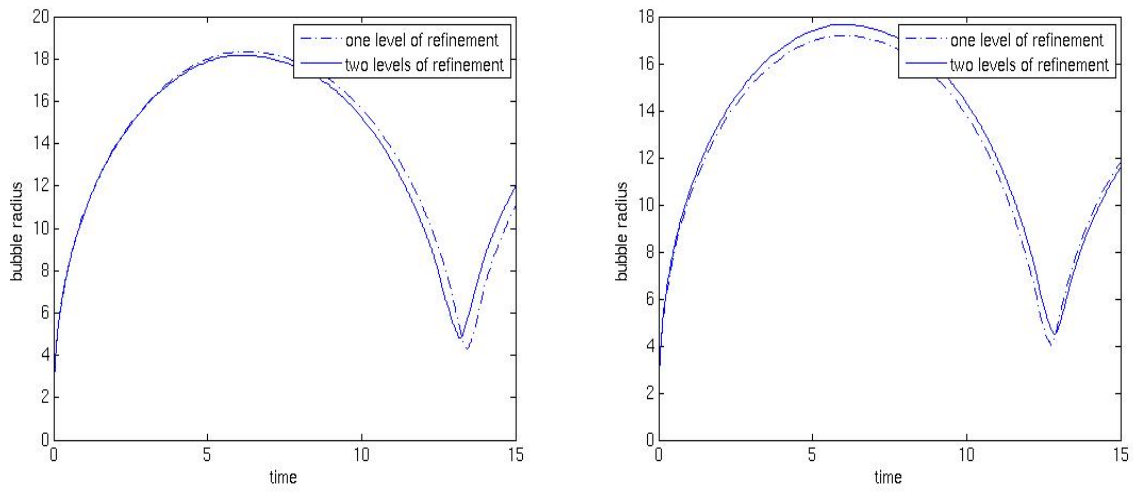
Figure 17: A grid refinement study for the effective bubble radius when solving the bubble jetting problem. Left figure corresponds the results from non-conservative semi-implicit method using $\Delta t_{semi-implicit}$. Right figure corresponds the results from conservative semi-implicit method using $\Delta t_{semi-implicit}$. Base grid for the both refinements is 320×320. $t = 15msec$.
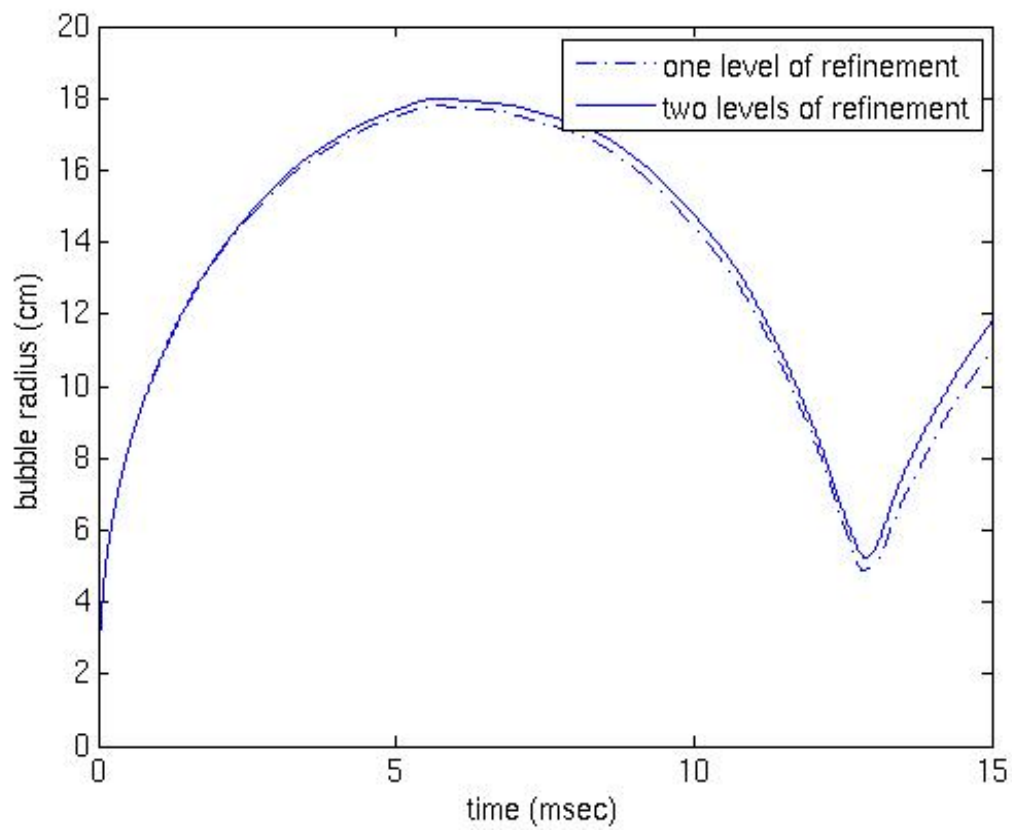
Figure 18: A grid refinement study for the effective bubble radius when using the non-conservative semi-implicit method using $\Delta t_{explicit}$. Base grid for the both refinements is 320×320. $t = 15 msec$.
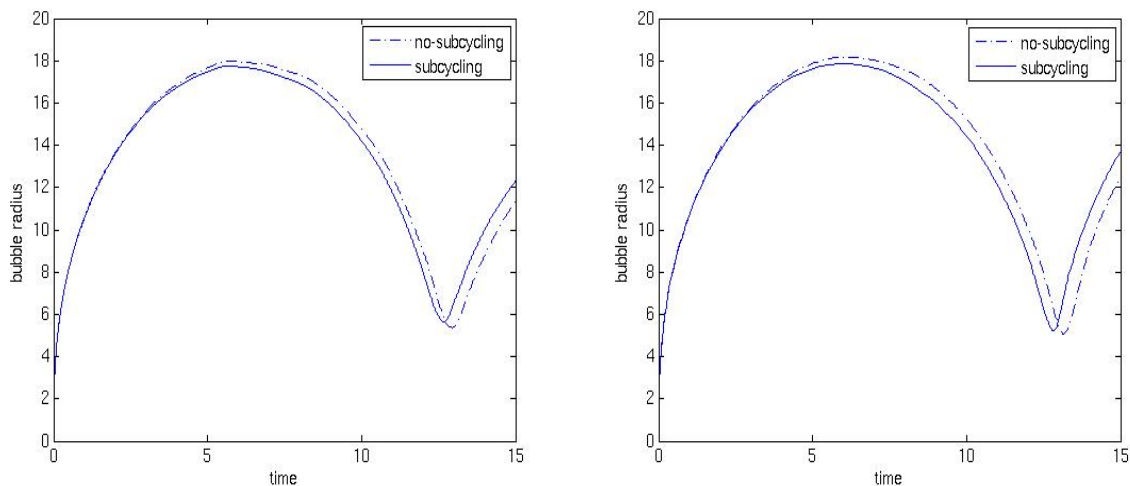
Figure 19: Comparison of the bubble radii for the bubble jetting problem. The left figure compares the results of subcycling versus no-subcycling procedures from the non-conservative semi-implicit calculations using $\Delta t_{explicit}$. The right figure compares the results of subcycling versus no-subcycling procedures from the non-conservative semi-implicit calculations using $\Delta t_{semi-implicit}$. Base grid for all calculations is 320×320 with *two* levels of refinement. $t = 15msec$.