interPhaseChangeBubbleFoam: A hybrid Eulerian-Lagrangian solver to capture nuclei effects on hydrodynamic cavitation

Mahdi Lavari

Mechanical & Materials Engineering,
Worcester Polytechnic Institute
Worcester, US

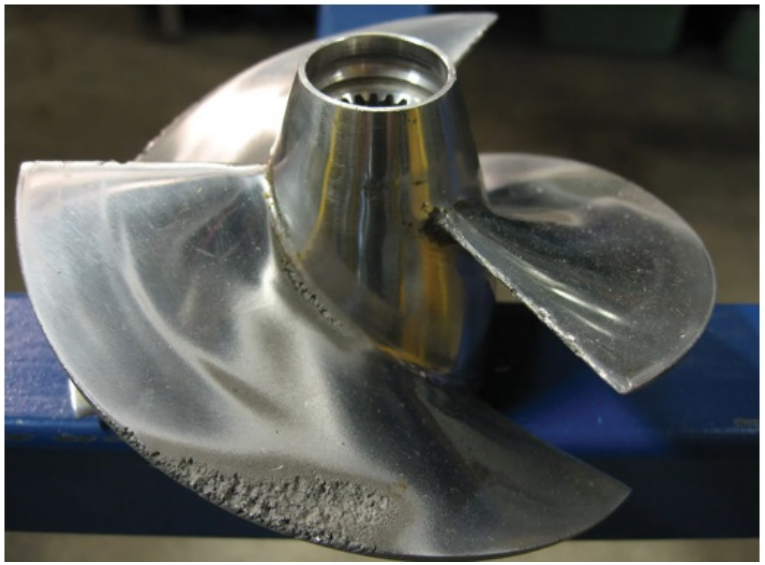
January 19, 2025

Motivation

Cavitation occurs when the pressure in a liquid drops below its vapor pressure, leading to vapor formation.

- Common in industrial hydrodynamic systems.
- Causes erosion due to violent microbubble collapse over short time scales.
- Exhibits a range cavity structures, from microbubbles to large sheet cavities.

Cavitation downstream of a propeller [ref]

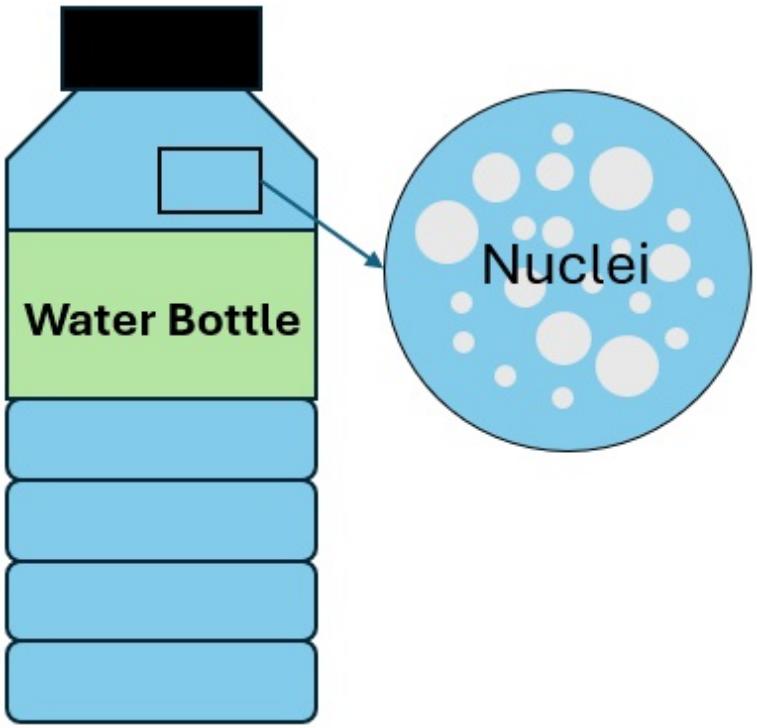


Erosion caused by cavitation [ref]

Problem Description

The main reason for the formation of vapor is the rapid expansion of nuclei in the water, which indicates that the water is not pure. Considering the presence of nuclei in cavitation has two primary effects:

1. The process of cavitation inception can be investigated.
2. The damage caused by the collapse of unresolved bubbles can be captured.



The presence of nuclei in water.

Solutions

The cavitation phenomenon can be modeled using two common numerical approaches:

Homogeneous Mixture Model (Volume of Fluid)

- Treats liquid and vapor as a single-phase mixture.
- Tracks vapor using the volume fraction (α).
- Governing equations include mass conservation, momentum rate conservation, and transport equation for volume fraction of phases.

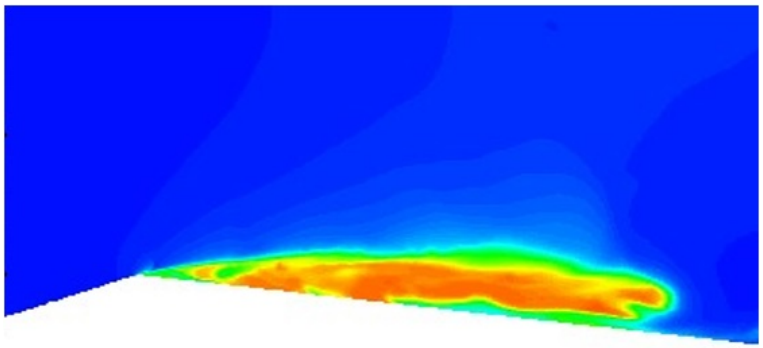
Two-Fluid Model

- Treats liquid and vapor as separate fluids with distinct velocity and pressure fields.
- Governing equations include separate mass, momentum, and energy conservation for each phase, coupled with interphase exchange terms.

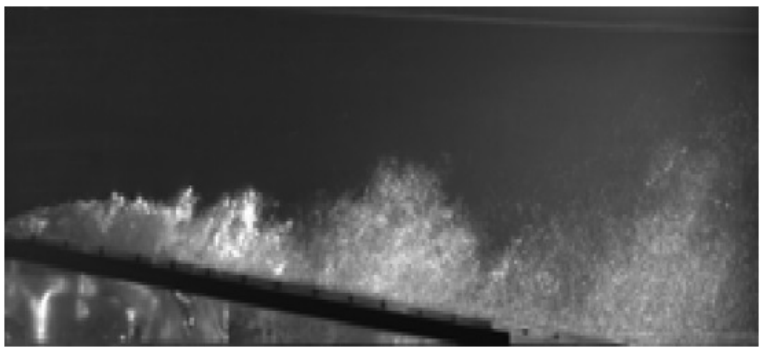
Solutions (cont.)

These methods effectively capture large-scale structures but struggle with small-scale structures, leading to potential inaccuracies in:

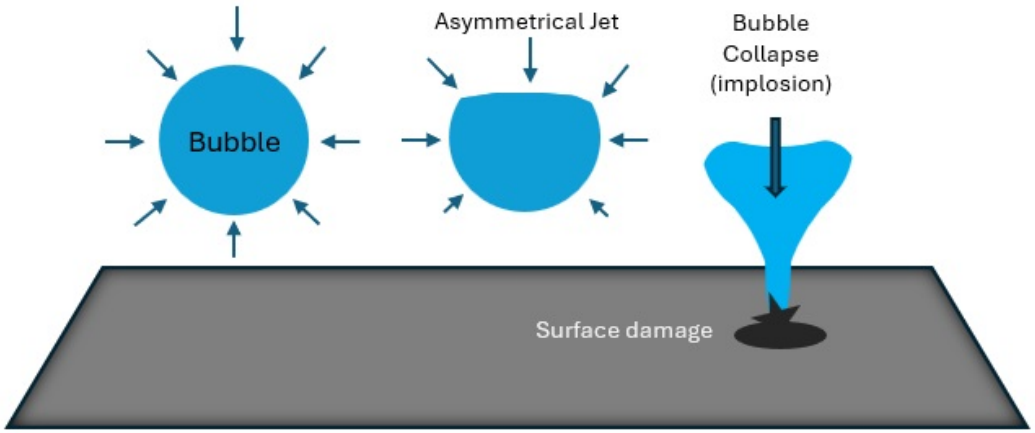
- Cavitation mechanisms
- Surface damage



Flow over a wedge (VOF).



Flow over a wedge. [ref]

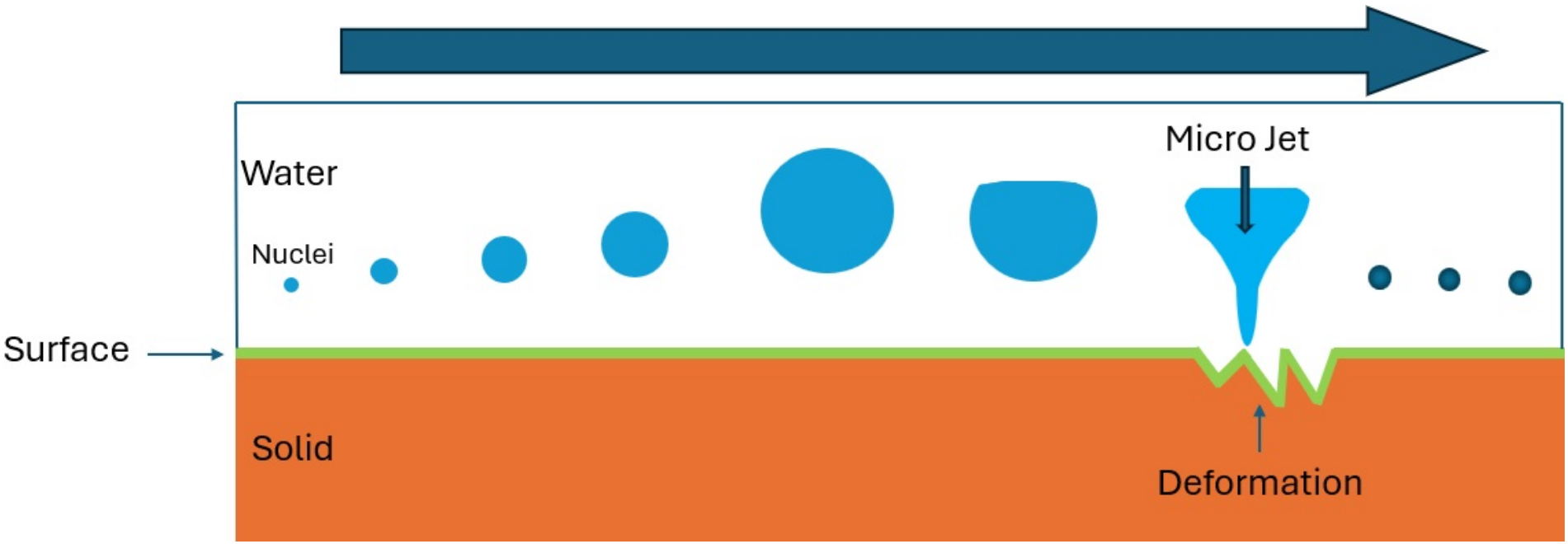


Damage caused by the collapse of a bubble.

Hybrid Multiscale Approach

A hybrid approach couples Lagrangian tracking with large-scale structure models to address nuclei and small-scale bubbles:

- Captures cavitation inception.
- Interactions between small- and large-scale structures.



Considering the presence of nuclei, the location and time of cavitation inception can be captured.

Hybrid Multiscale Approach(cont.)

This project aims to address the knowledge gap caused by neglecting nuclei. It involves injecting a nucleus, tracking its dynamics within the Lagrangian framework using the Rayleigh-Plesset equation, and transitioning to the Eulerian framework if the structure grows large. The project workflow is outlined as follows:

Objectives:

1. Integrate the Rayleigh-Plesset equation to model changes in bubble diameter within the Lagrangian library.
2. Couple the modified Lagrangian library with `interPhaseChangeFoam`.
3. Modify `interPhaseChangeFoam` to incorporate source terms.
4. Develop a transition mechanism between Eulerian and Lagrangian frameworks.

interPhaseChangeFoam

The `interPhaseChangeFoam` solver uses the Volume of Fluid (VOF) method to model vapor-liquid mixtures:

- Flow parameters (density, viscosity) are defined as mixture forms.
- The volume fraction of the liquid, α_l, represents the fraction of liquid in each computational cell.

The vapor volume fraction is then given by

$$\alpha_v = 1 - \alpha_l, \quad \alpha_l \in [0, 1], \quad \alpha_v \in [0, 1]. \tag{1}$$

interPhaseChangeFoam - Conservation Equations

The conservation of mass and momentum for VOF in a Newtonian fluid are expressed as

$$\frac{\partial \rho_m}{\partial t} + \nabla \cdot (\rho_m \mathbf{U}) = 0, \tag{2}$$

$$\begin{aligned} \frac{\partial}{\partial t}(\rho_m u_j) + \frac{\partial}{\partial x_j}(\rho_m u_i u_j) = & -\frac{\partial p}{\partial x_i} \\ & + \frac{\partial}{\partial x_j}\left[(\mu_m + \mu_{\text{laminar}})\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)\right] \\ & + \rho_m g. \end{aligned} \tag{3}$$

In these equations:

$$\rho_m = \alpha_l \rho_l + (1 - \alpha_l)\rho_v, \tag{4}$$

$$\mu_m = \alpha_l \mu_l + (1 - \alpha_l)\mu_v. \tag{5}$$

interPhaseChangeFoam - Governing Equation for α_l

To govern the void volume fraction, the Reynolds transport equation for the liquid phase is written as

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l \mathbf{U}) = \frac{\dot{m}}{\rho_l}. \tag{6}$$

In OpenFOAM, `interPhaseChangeFoam` employs the PIMPLE algorithm. The divergence of velocity acts as a constraint in the conservation of momentum equation, here, it is given by

$$\nabla \cdot \mathbf{U} = \left(\frac{1}{\rho_l} - \frac{1}{\rho_v} \right) \dot{m}. \tag{7}$$

Finite Mass Transfer; Schnerr-Sauer model

The key to closing the system of equations is modeling the finite mass transfer, $\dot{m}$:

- OpenFOAM includes Kunz, Merkel, and Schnerr-Sauer mass transfer models.
- This project uses the Schnerr-Sauer model, which simplifies the Rayleigh-Plesset equation.
- Cavitation begins where pressure drops below the saturation vapor pressure, driven by mass transfer source terms:

$$\dot{m}_c = C_c \alpha_l (1-\alpha_l) \frac{3\rho_l \rho_v}{\rho_m R_B} \left(\frac{2}{3\rho_l} |p - p_{\text{sat}}| \right)^{1/2}, \quad p \geq p_{\text{sat}},$$

$$\dot{m}_v = C_v \alpha_l (1+\alpha_{Nc}-\alpha_l) \frac{3\rho_l \rho_v}{\rho_m R_B} \left(\frac{2}{3\rho_l} |p - p_{\text{sat}}| \right)^{1/2}, \quad p < p_{\text{sat}}. \tag{8}$$

Lagrangian Library

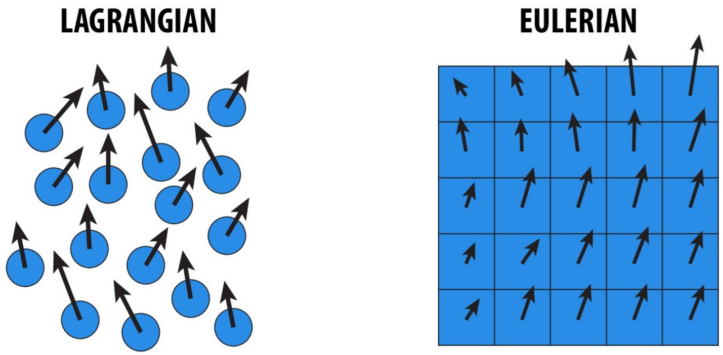
Lagrangian tracking in OpenFOAM relies on two main classes for non-evaporating, isothermal particles:

- `solidParticle`: Tracks individual particles with a preliminary design.
- `KinematicParcel`: Tracks parcels of particles sharing size, velocity, and trajectory.

Bubble trajectories are computed using Lagrangian equations that account for forces acting on the bubbles:

$$\frac{dx_b}{dt} = u_b, \tag{9}$$

$$m_b \frac{du_b}{dt} = F_d. \tag{10}$$



Lagrangian vs Eulerian frameworks. [ref]

Bubble Dynamics

Tracking nuclei diameter evolution requires bubble dynamics:

- `KinematicParcel` tracks particle parcels, not bubble parcels.
- Diameter changes must be defined in `KinematicParcel` to evolve nuclei size.
- Transfer to the Eulerian field.

The most well-known bubble dynamics equation, ignoring compressibility effects, is the Rayleigh-Plesset equation.

$$\rho_l \left(\frac{1}{2} R\ddot{R} + \frac{1}{2}\dot{R}^2 \right) = p_v + p_{g0} \left(\frac{R_0}{R} \right)^{3k} - p_{xR} - 4\mu_l \frac{\dot{R}}{R} - \frac{2\sigma}{R}. \tag{11}$$

The modified classes, `KinematicBubbleCloud` and `KinematicBubbleParcel`, incorporate this localized Rayleigh-Plesset equation.

Theory of interPhaseChangeBubbleFoam

Vallier [ref] coupled Lagrangian tracking with `icoFoam` and developed transition algorithms between Eulerian and Lagrangian frameworks.

- Limitation: Their approach allowed vapor coexistence in the same cell, causing unphysical results.

Ghahramani [ref] introduced β_l, a Lagrangian liquid volume fraction, to prevent vapor phase coexistence.

- Limitation: Unmodified source terms failed to accurately update the Eulerian field, hindering proper nuclei tracking.

To resolve the issues, multiscale Lagrangian-Eulerian cavitation can be understood as a flow divided into resolved and unresolved phases:

- Resolved liquid (continuous phase).
- Dispersed vapor:
  - Resolved vapor (Eulerian).
  - Unresolved vapor (Lagrangian).

Theory of interPhaseChangeBubbleFoam(cont.)

Important assumptions:

- The two dispersed vapors share the same physical properties (density, viscosity).
- The two vapors cannot coexist in the same location; resolved liquid can form mixtures with both vapors.
- There is no slip velocity between resolved liquid and vapor; slip is considered between unresolved vapor and liquid.

The transition of a structure from Eulerian to Lagrangian framework does not remove it from the Eulerian field; rather, its position, mass, and momentum are solved in the Lagrangian framework.

Addition of Unresolved Liquid Volume Fraction

In each cell, β_l represents the unresolved vapor fraction in the Lagrangian framework:

- $\beta_l \in [0, 1)$, with $\alpha_l = 1$ for unresolved vapor.
- $\alpha_l \in [0, 1]$, with $\beta_l = 1$ for resolved liquid.

The density, viscosity, and momentum are defined as:

$$\rho_m = (\alpha_l + \beta_l - 1)\rho_l + (1 - \alpha_l)\rho_v + (1 - \beta_l)\rho_v \tag{12}$$

$$\mu_m = (\alpha_l + \beta_l - 1)\mu_l + (1 - \alpha_l)\mu_v + (1 - \beta_l)\mu_v \tag{13}$$

$$\rho_m u_i = (\alpha_l + \beta_l - 1)\rho_l u_i + (1 - \alpha_l)\rho_v u_i + (1 - \beta_l)\rho_v u_{Li} \tag{14}$$

New Hybrid Equations

The convergence of velocity for this hybrid model is given by:

$$\nabla \cdot \mathbf{U} = \left(\frac{1}{\rho_l} - \frac{1}{\rho_v} \right) \dot{m} - \frac{1}{\beta_l \rho_l} \left[\dot{m}_{\text{bubble}} \right] - \frac{1}{\beta_l} \left[\frac{D\beta_l}{Dt} \right] \tag{15}$$

The conservation of momentum for this hybrid model is given by:

$$\begin{aligned} &\left[\frac{\partial \left(\alpha \rho_l + (1-\alpha)\rho_v \right) u_i}{\partial t} + \frac{\partial \left(\alpha \rho_l + (1-\alpha)\rho_v \right) u_i u_j}{\partial x_j} \right] \\ &\quad - \left[\frac{\partial \left((1-\beta_l)\rho_l u_i \right)}{\partial t} + \frac{\partial \left((1-\beta_l)\rho_l u_i u_j \right)}{\partial x_j} \right] \\ &= - \left[\frac{F}{V} \right] - \frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho_m g_i \end{aligned} \tag{16}$$

New Hybrid Equations(cont.)

To simplify the coding process, γ as the product of α_l and β_l is defined.

$$\gamma = \alpha_l \beta_l, \tag{17}$$

The momentum equation can also be expressed in a compact form, which simplifies coding and implementation and is given by

$$\begin{aligned} &\left[\frac{\partial\left(\gamma\rho_l + (1-\alpha)\rho_v\right)u_i}{\partial t}\right. \\ &\left. + \frac{\partial\left(\gamma\rho_l + (1-\alpha)\rho_v\right)u_i u_j}{\partial x_j}\right] \\ &= -\left[\frac{F}{V}\right] - \frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho_m g_i. \end{aligned} \tag{18}$$

Coupling the Rayleigh-Plesset class with KinematicCloud

The implementation of the Rayleigh-Plesset equation was explored in the proceedings of this course.

- Ghahramani [ref] implemented a Rayleigh-Plesset solver within `move()` member function of `solidParticle` cloud.
- Chen [ref] coded the Rayleigh-Plesset solver within `KinematicParcel`, utilizing ODE solver provided by OpenFOAM.

In this work, the author adopted the same concept from Chen, with modifications:

- Upgraded to the version 2406 of OpenFOAM
- Localized Rayleigh-Plesset equation
- Modification of inputs
- Tracking the diameter of the bubble
- Transfer the dictionary containing bubble dynamics inputs to the `KinematicParcel` constructor

KinematicBubbleCloud

To incorporate the Rayleigh-Plesset class into `KinematicCloud`, the following steps can be taken:

- The RPEqu class should be instantiated, since Rayleigh-Plesset equation is defined within `RPEqu.H`.
- `bubbleDynamics.H` serves as a coupling agent, collecting inputs from `KinematicParcel` and `RPEqu.H`.
- The equation is solved within `bubbleDynamics.H`.

This structure enables modular and efficient integration of the Rayleigh-Plesset model into `KinematicParcel`.

Listing 1: Inside `bubbleDynamics.H`

```
RPEquSolver-solve(xStart, xEnd, yStart, dxEst);
```

Integrating Rayleigh-Plesset into KinematicBubbleCloud

The integration of `bubbleDynamics.H` occurs in the `calcVelocity` function, which is a member of the `KinematicBubbleParcel` class. This function is responsible for calculating the velocity of a bubble parcel based on various forces acting on it.

Listing 2: calcVelocity

```
//- ML: Include bubble dynamics
if(this->bubble_activation())
{
    include "bubbleDynamics.H"
}

//dUTrans -= massEff*deltaUcp;
vector Unew2 = U_ + deltaUcp;
scalar mass2 = this->mass();
const scalar massEff2 = forces.massEff(p, ttd, mass2);
dUTrans -= ((massEff2*Unew2)-(massEff*U_));
```

Source terms in KinematicBubbleCloud

The variables `mDotBubble_` and `Dbetal_` are used to close source terms of the divergence of velocity equation.

Listing 3: calc in `KinematicBubbleParcel`

```
//-ML: Update return rate of change of mass
cloud.mDotBubble()[cellJ] += normalizedWeight*np0*mDotB;
```

Listing 4: evolveCloud in `KinematicBubbleCloud`

```
//-ML: Reset the beta; should be reset after injection
// since in injection, we have move function triggered
betal() = 1.0;

// Assume motion will update the cellOccupancy as needed
cloud.motion(cloud, td);

//-ML: DBeta for non-free-divergence equations
Dbetal() = (betal() - betal_old()) / (this->db().time().deltaT());
```

interPhaseChangeBubbleFoam; Beta and Gamma

The variables β_l and γ are defined as follows:

- α_l is a member of the `phaseChangeTwoPhaseMixture` class, which inherits from `incompressibleTwoPhaseMixture` and `twoPhaseMixture`.
- α_l is find in `twoPhaseMixture`,and read from the time directory dictionary.
- Consequently, β_l and γ should be defined in the `twoPhaseMixture` class.

To couple `KinematicBubbleCloud` with `interPhaseChangeBubbleFoam`:

- Include `basicKinematicBubbleCloud.H`.
- Add source terms from the Lagrangian field to the conservation equations.
- Create `KinematicBubbleCloud` for bubble tracking.

Coupling Eulerian with Lagrangian

1.Include the declaration of `basicKinematicBubbleCloud.H` in `interPhaseChangeBubbleFoam`.

Listing 5: Modifications in interPhaseChangeBubbleFoam

```
//-ML : declaration of basicKinematicBubbleCloud
#include "basicKinematicBubbleCloud.H"
```

2. In `createField.H`, a cloud of bubbles is created, with the name specified as parcels.

Listing 6: createField.H

```
basicKinematicBubbleCloud parcels
(
    kinematicBubbleCloudName,
    rho,
    U,
    muc,
    p,  //-ML: Added for bubble dynamics-Rayleigh Plesset equation
    g
);
```

Coupling Eulerian with Lagrangian (cont.)

3. Add `parcels.SU(U)` to the momentum equation in `UEqn.H`.

Listing 7: Modifications in UEqn

```
fvVectorMatrix UEqn
(
    fvm::ddt(rho, U)
  + fvm::div(rhoPhi, U)
  - fvm::Sp(fvc::ddt(rho) + fvc::div(rhoPhi), U)
  - fvm::ddt(rho2Beta2, U) //-ML: subtract effects of 1-beta1
  - fvm::div(rho2PhiBeta2, U) //-ML: subtract effects of 1-beta1
  + turbulence->divDevRhoReff(rho, U)
  + parcels.SU(U) //-ML: Return momentum source from Lagrangian framework
);
```

4. Add terms `parcels.SU_mDotBubble()` and `parcels.Dbetal()` in `pEqn.H`.

Listing 8: Modifications in pEqn.H

```
fvScalarMatrix p_rghEqn
(
    fvc::div(phiHbyA) - fvm::laplacian(rAUf, p_rgh)
  - (EuvDotcvP)*(mixture->pSat() - rho*gh)
  + fvm::Sp(EuvDotcvP, p_rgh)
  + invRho1Beta*parcels.SU_mDotBubble()
  + invBeta*parcels.Dbetal()
);
```

Coupling Eulerian with Lagrangian (cont.)

5. Add the `evolve` member function of `basicKinematicBubbleCloud` in `interPhaseChangeBubbleFoam` to manipulate parcels (e.g., moving, injecting, and updating source terms).

Listing 9: Modifications in interPhaseChangeBubbleFoam

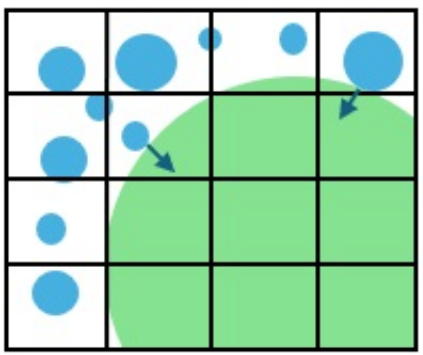
```
//-ML: Adding evolution of cloud
Info<< "Evolution of Cloud: "<< parcels.name() << endl;

parcels.evolve();
//-ML: update the gamma value from multiplication of alpha1_*beta1_
mixture->update_gamma();

runTime.write();
```

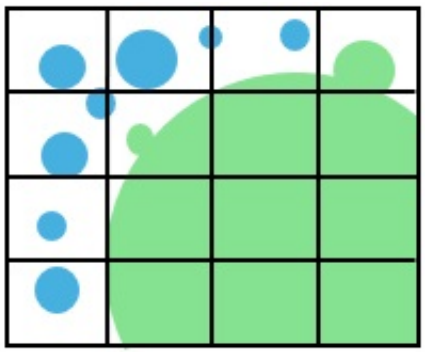
Other minor modifications in `pEqn.H`, `UEqn.H`, and `alphaEqn.H` can be found in the accompanying files.

Lagrangian to Eulerian Framework Transition

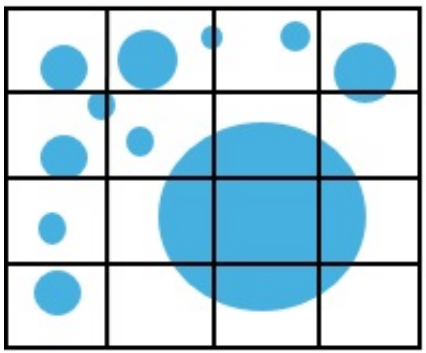
The project aims to inject nuclei as unresolved vapor and track their size and movement. Transition from the Lagrangian to Eulerian framework occurs in the following cases:

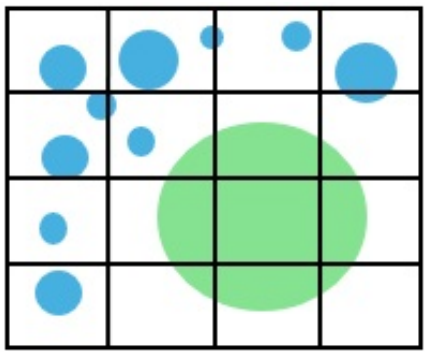
1. Expansion: Vapor structures grow beyond a size threshold, making Eulerian tracking more efficient.
2. Coalescence: Microscale vapor structures merge with macroscale structures near resolved vapor.





Transition due to Coalescence





Transition due to Expansion

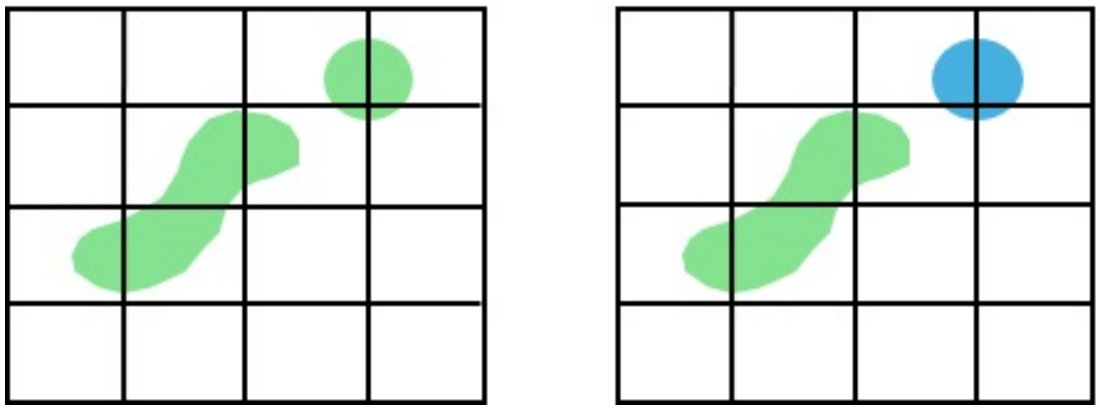
Lagrangian to Eulerian Framework Transition (Cont.)

Key steps in managing the transition between Lagrangian and Eulerian frameworks:

1. Bubble Position and Radius: Determine bubble position (`posB`), radius (`radB`), and compute distances from the center.
2. Dynamic List of Cells: Use a `DynamicList` to track cells within $2 \times$ `radB`, ensuring 100% vapor distribution.
3. Volume Redistribution: Apply Gaussian weights for smooth, volume-conserving redistribution.
4. Proximity to Interface: Evaluate alpha values in nearby cells. Bubbles near Eulerian structures (below threshold) trigger deactivation.
5. Box Limit for Optimization: Confine bubble tracking within user-defined box boundaries to reduce computational cost.

Eulerian to Lagrangian Framework Transition

When Eulerian vapor structures shrink below the resolution of the cell size, they can be transitioned to the Lagrangian framework. Heinrich et al[ref] utilized Connected Component Labeling (CCL) in nozzle atomization to convert small Eulerian elements into Lagrangian parcels, reducing computational cost.



Transition from Lagrangian to Eulerian due to expansion

Eulerian to Lagrangian Framework Transition (1/2)

Key steps in managing the transition from Eulerian to Lagrangian frameworks:

- Thresholds Definition:
  - `alpha1Lim`: Cutoff for vapor-liquid distinction.
  - `minCells`: Minimum number of cells to identify significant structures.
- Volume ID Assignment: Identifies and marks connected regions with unique volume IDs through recursive exploration.
- Parallel Processing: Ensures consistency of `vofID_` across coupled patches with a logarithmic merging process based on processor count.
- Property Calculation: The code calculates:
  - Number of cells, volume, centroid position, and average velocity.
- Bubble Injection: For structures below the `minCells` threshold, properties like center position and velocity are adjusted before injection into a Lagrangian cloud.

Eulerian to Lagrangian Framework Transition (2/2)

The `EulerianToLagrangian.H` file should be integrated into `interPhaseChangeBubble` because it handles the update of resolved vapors. Since `EulerianToLagrangian.H` injects bubbles into the Lagrangian framework, it needs to be included before the `evolve` function in `interPhaseChangeBubble`.

Listing 10: Modifications in interPhaseChangeBubbleFoam

```
//-ML: Transition from Eulerian to Lagrangian
if(EulerToLagrang_activation)
{
    #include "EulerianToLagrangian.H"
}
mixture->update_gamma();

//-ML: Adding evolution of cloud
Info<< "Evolution of Cloud: "<< parcels.name() << endl;
```

Tutorials

Four test cases have been developed to assess the capabilities of the new solver, `interPhaseChangeBubbleFoam`. These cases are designed for short simulation times, making them unsuitable for real engineering applications. However, they allow users to adapt them to their specific needs.

The four test cases are as follows:

- **Case 1:** Tracks the interface between unresolved bubbles and resolves the merging process with vapor as the interface is tracked.
- **Case 2:** Transfers unresolved vapor to resolved vapor when its size exceeds a user-defined threshold.
- **Case 3:** Transfers resolved vapor to unresolved vapor when its size falls below a user-defined threshold.
- **Case 4:** Deletes Lagrangian bubbles that are outside the region of interest or a user-defined box.

Case Study 1: Interface Tracking

Case 1 involves a 3D box where the lower half is filled with water and the upper half with air. The dimensions of the box are as follows:

- x-direction: 0.1 m
- y-direction: 0.2 m
- z-direction: 0.1 m

The box is discretized with 50 cells in the x-direction, 100 cells in the y-direction, and 50 cells in the z-direction, resulting in a uniform cell size of 0.002 m. This gives a total of 250,000 cells. The boundary conditions are defined as:

- leftWall, rightWall, lowerWall, backWall, frontWall, atmosphere patch (top).

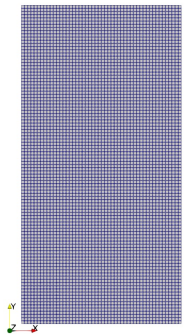


Figure: z-plane of the mesh

Simulation Details for Case 1

The `setFields` utility is used to fill the lower half of the box with water. The simulation runs for 0.1 s, with a time step of 0.001 s.
At time 0, a bubble is injected at the position $(0.05, 0.085, 0.05)$ using the `manualInjection` model, and it rises due to buoyancy.
Bubble dynamics are disabled, meaning the Rayleigh-Plesset equation is not applied, and bubble diameter changes are deactivated.

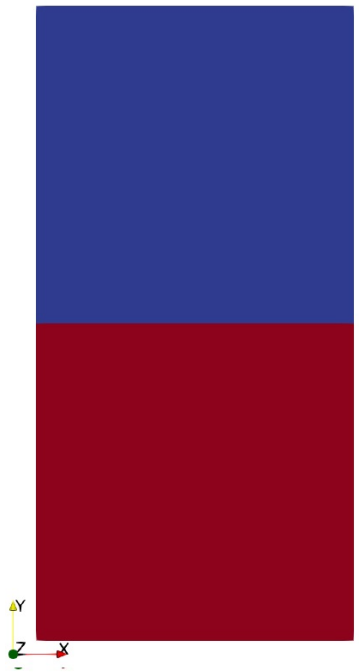
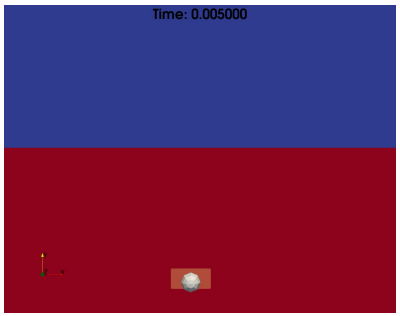


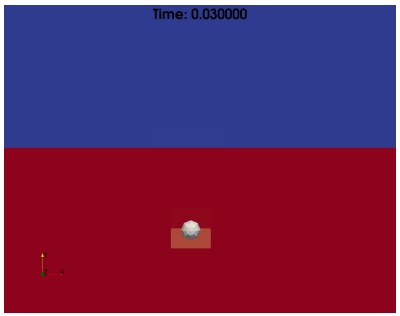
Figure: Red represents water,and blue represents air.

Results for Case 1

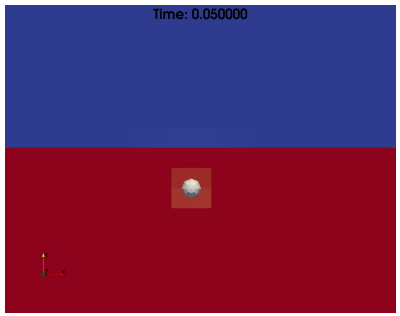
At time 0.065 s, the bubble rises and touches the interface between air and water. At this point, it is removed from the Lagrangian framework, but its effects persist in the domain as the alpha volume fraction.



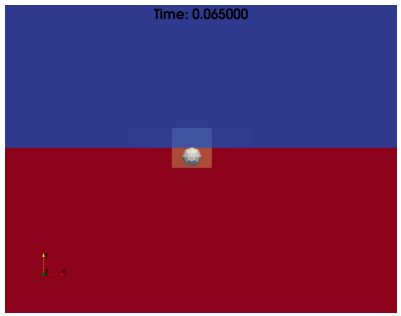
Time 0.005 s



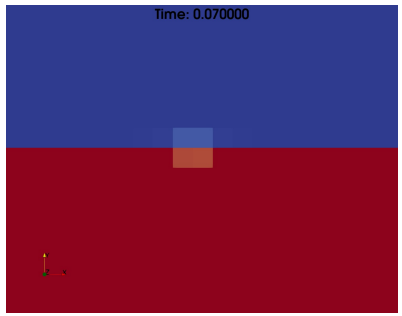
Time 0.03 s



Time 0.05 s



Time 0.065 s



Time 0.07 s

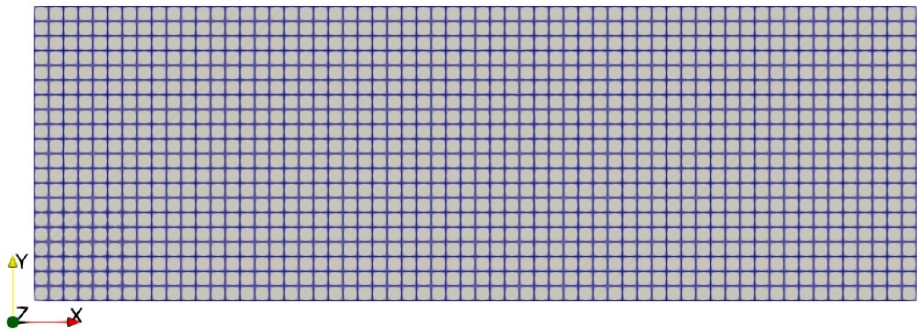
Case Study 2: Lagrangian to Eulerian Transition

Case 2 involves injecting a nucleus into a box where the pressure is below the vapor pressure (2300 Pa). As a result, the Lagrangian bubble moves and its radius increases. A threshold value of 44 is set, at which point unresolved vapor is transferred to the Eulerian framework. This case is critical to the project as it demonstrates the primary goal of transitioning between Lagrangian and Eulerian frameworks.

The box dimensions are as follows:

- x-direction: 0.3 m
- y-direction: 0.1 m
- z-direction: 0.1 m

The simulation uses 24,000 cells, with a uniform cell size of 0.005 m.



z-plane of the mesh

Simulation Details for Case 2

The pressure is set from 2000 Pa at the inlet to 1000 Pascal at the outlet. The simulation runs for 0.25 s with a time step of 0.0005 s.

A bubble is injected at time 0.02 s using the `manualInjection` model, with an initial diameter of 0.001 m and velocity of 2 m/s. All forces on the bubble are disabled, so it moves with prescribed motion.

The transition from Lagrangian to Eulerian is triggered by an `LE_cellThreshold` value of 44

Pressure distribution

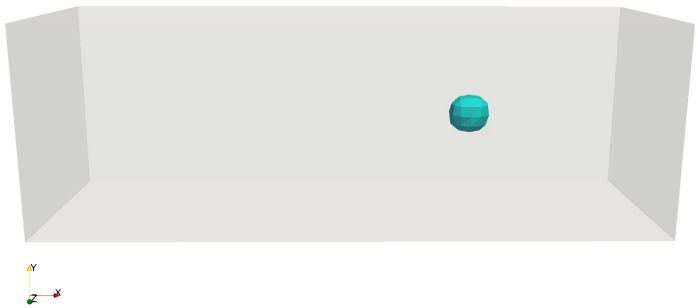
Results for Case 2

At time 0.124 s, the number of cells occupied by the unresolved vapor exceeds the threshold of 44. The Lagrangian vapor is removed, but its `alpha` values remain in the domain and move with the fluid flow, driven by the pressure difference between the inlet and outlet.

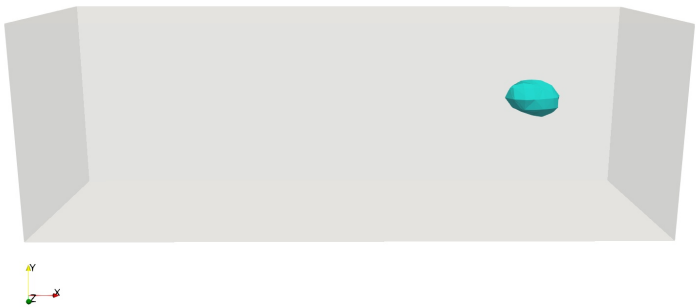
Time: 0.090000

Time: 0.124000

Time: 0.125000

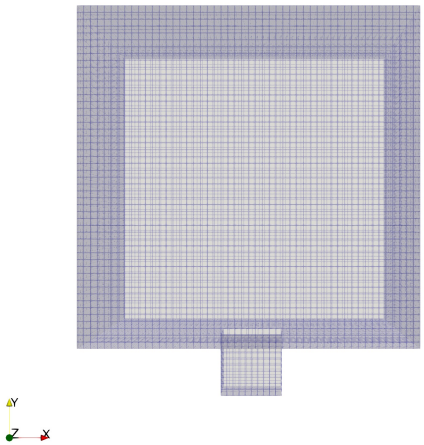


Time: 0.155000

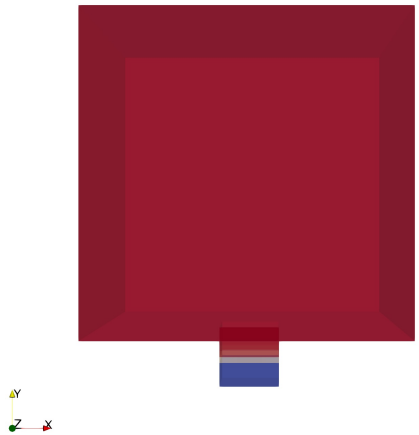


Case Study 3: Eulerian to Lagrangian Transition

Case 3 involves a tank filled with water, with an air inlet at the bottom. The cells have a size of 0.01 m, resulting in 126,000 cells.
The `setFields` utility fills a small region of the nozzle with air. The simulation runs for 0.025 s with a time step of 0.0005 s.
The Eulerian-to-Lagrangian transition threshold is set to 4, meaning unresolved vapor structures smaller than 4 cells are injected at the center of the previously resolved structure.



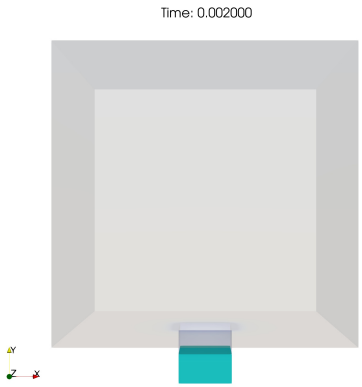
Tank with opening from bottom



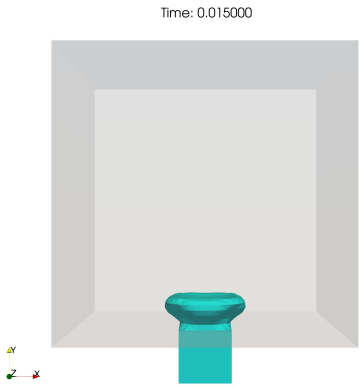
Red represents water and blue represents air

Results for Case 3

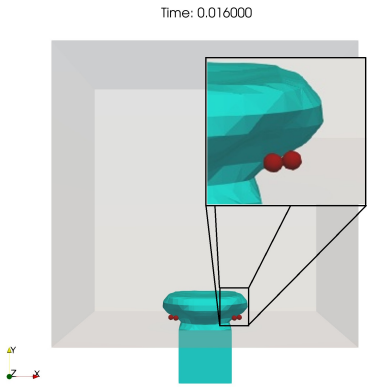
At time 0.016 s, the first resolved vapors are introduced. As the simulation progresses, the number of unresolved vapors increases, following the defined transition rules.



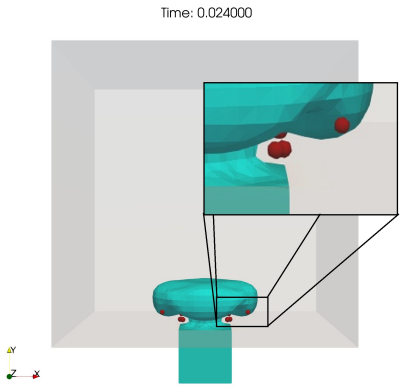
Time 0.002 s



Time 0.015 s



Time 0.016 s



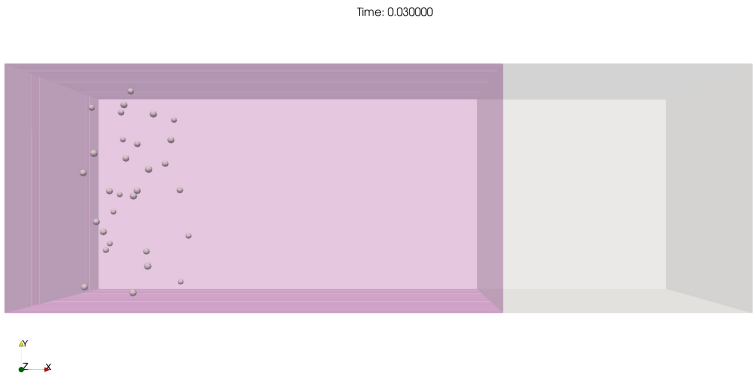
Time 0.024 s

Case Study 4: Tracking within Limited Region

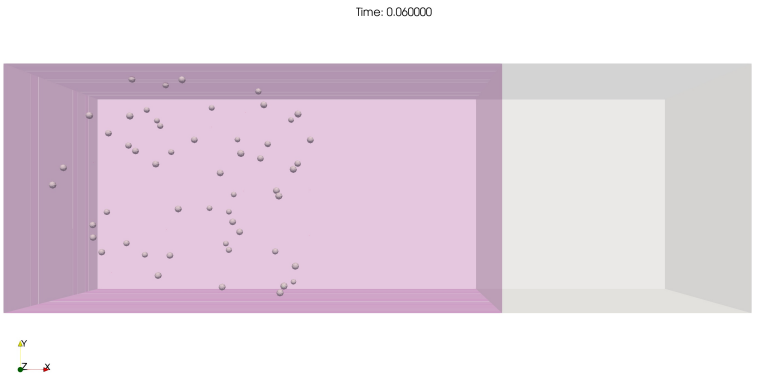
In cases with a large number of unresolved bubbles, tracking them can be computationally expensive. To address this, a zone of interest can be defined for tracking Lagrangian bubbles within a specified region.

- The mesh and configuration used are identical to Case 2.
- Approximately 200 bubbles are injected using the `patchInjection` model.
- The tracking region is defined by a box:
  - Top-left corner: (0.0, 0.1, 0.0)
  - Bottom-right corner: (0.2, 0.0, 0.1)
- A front of 200 bubbles is injected from the patch inlet.
- Bubbles moving outside the defined box are removed before reaching the left side.

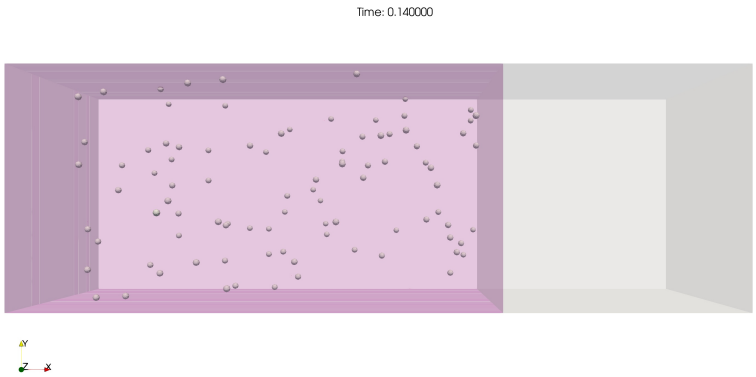
Results for Case 4



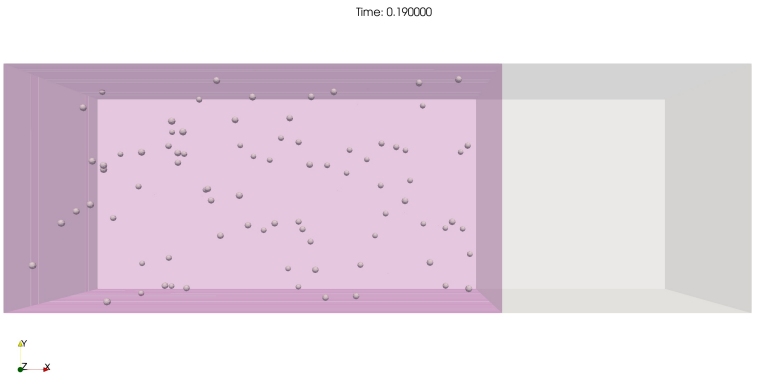
Time 0.03 s



Time 0.06 s



Time 0.14 s



Time 0.19 s

Thank you!

Thank you!