boilingFoam

Federico Municchi[1], Mirco Magnini[2,*]

[1]Colorado School of Mines, 1500 Illinois St., Golden, CO 80401

[2]Department of Mechanical, Materials and Manufacturing Engineering, University of Nottingham

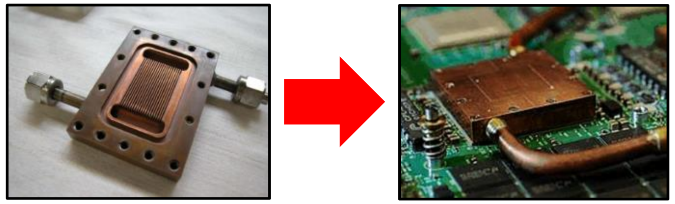
[*]E-mail: mirco.magnini@nottingham.ac.uk

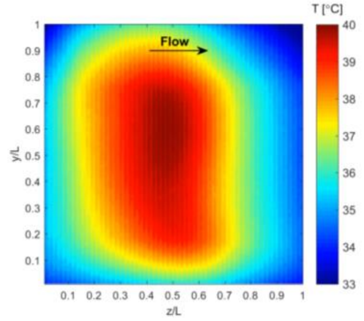
Content

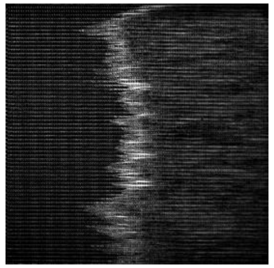
Phase change

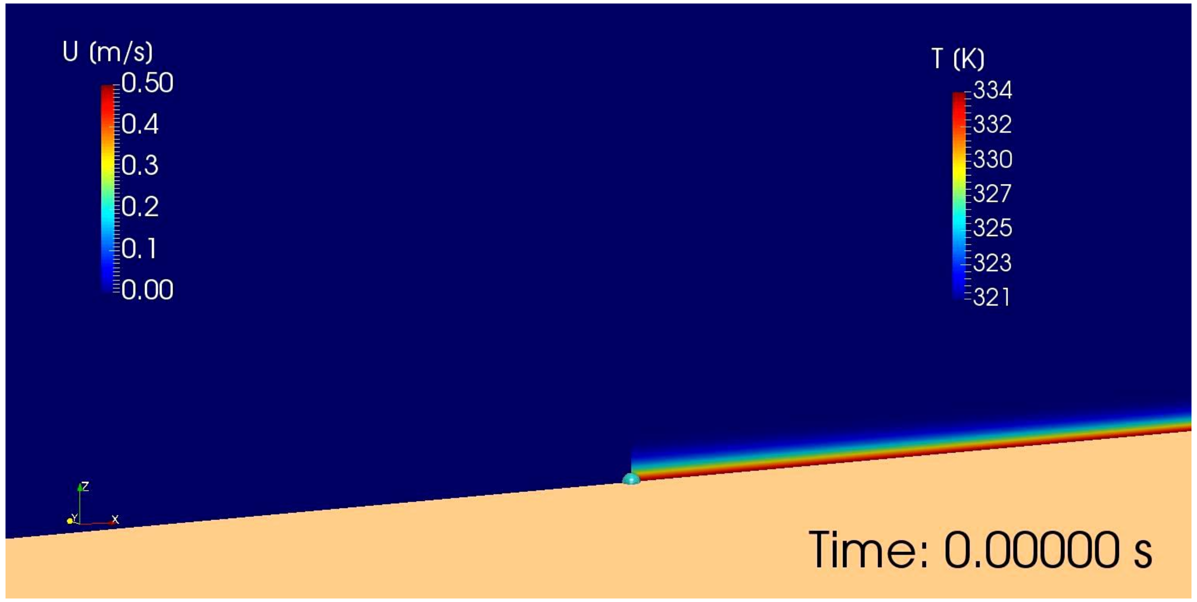
Phase change is the physical process underpinning boiling, condensation and evaporation, with many applications in engineering starting with thermal management or heating/refrigeration. For example, boiling is used as an effective cooling method:

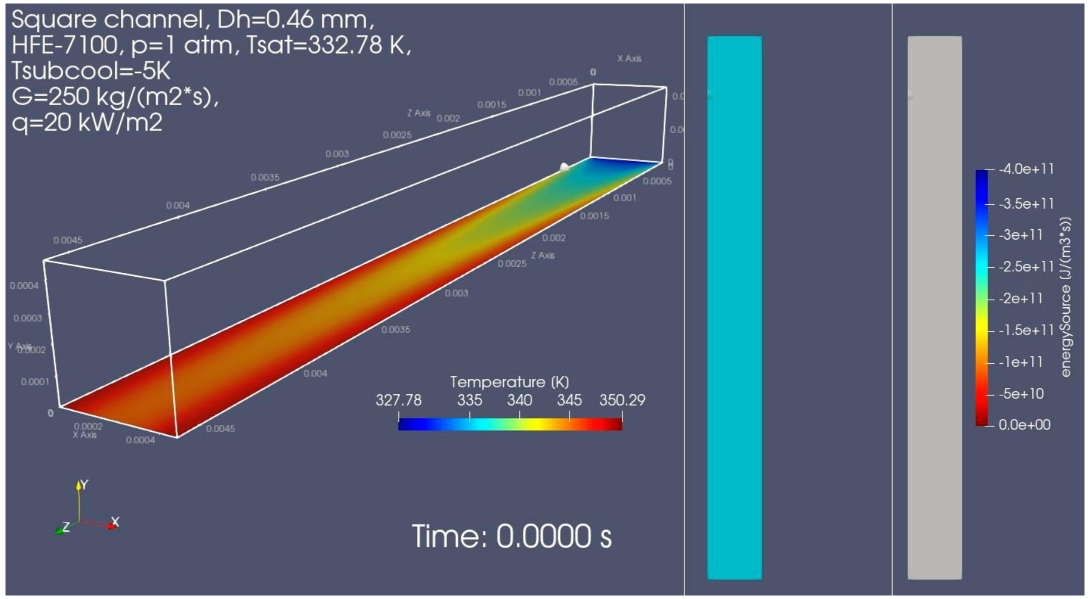
- Immersion cooling using **nucleate boiling**: https://www.youtube.com/watch?v=U6LQeFmY-IU
- Microchannel cooling using **flow boiling** [1,2]:











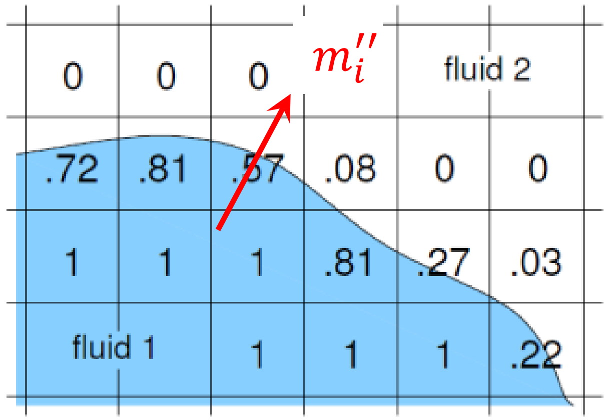
[1] J. Park, PhD Thesis, EPFL, 2008.
[2] C. Falsetti et al., *Int. J. Heat Mass Transfer* **107** (2017) 805.

Phase change

Governing equations for the incompressible flow of two immiscible phases based on the Volume of Fluid method, including phase change [1]:

$$I(\boldsymbol{x},t) = \begin{cases} 1 & \text{in fluid 1} \\ 0 & \text{in fluid 2} \end{cases} \qquad \alpha = \frac{1}{V}\int_V I(\boldsymbol{x},t)\,dV$$

Volume fraction field across an interface



$$\rho = \rho_2 + (\rho_1 - \rho_2)\alpha$$

$$\frac{\partial \alpha}{\partial t} + \nabla\cdot(\alpha\boldsymbol{u}) = -\frac{1}{\rho_1} m_i'' |\nabla\alpha|$$

m_i'': phase change rate from fluid 1 to 2 [kg/(m^{2}s)]

$$\nabla\cdot\boldsymbol{u} = \left(\frac{1}{\rho_2} - \frac{1}{\rho_1}\right) m_i'' |\nabla\alpha|$$

$$\frac{\partial(\rho\boldsymbol{u})}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla\cdot\boldsymbol{\tau} + \boldsymbol{F_\sigma}$$

$$\frac{\partial(\rho c_p T)}{\partial t} + \nabla\cdot(\rho c_p \boldsymbol{u} T) = \nabla\cdot(\lambda\nabla T) - m_i'' h_{lv}|\nabla\alpha|$$

[1] L. Malan et al., *J Comput Phys* 426 (2021) 109920.

Phase change

Most popular models for phase change rate at interface m_i'':

1) Approach based on thermal equilibrium at the interface [1]:
$T_{sat}(p_l) = T_l \cong T_v = T_{sat}(p_v)$

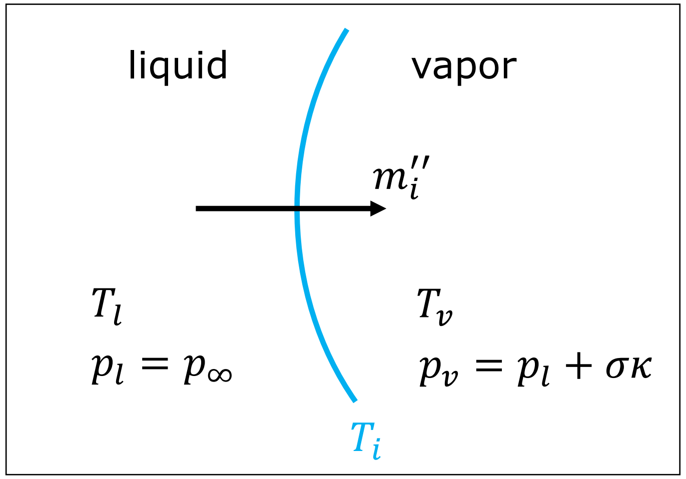
$$\rightarrow \boxed{m_i'' = \frac{1}{h_{lv}}(\boldsymbol{q}_l'' - \boldsymbol{q}_v'') \cdot \boldsymbol{n}}$$

2) Approach based on thermal equilibrium at the interface

according to Lee's [2] model:

$$\rightarrow \boxed{m_i''|\nabla\alpha| = r(1-\alpha)\rho_l \frac{T_i - T_{sat}}{T_{sat}}}$$

Schematic of interface conditions



3) Approach based on departure from thermal equilibrium at the interface [3]: $T_{sat}(p_l) = T_l \neq T_v = T_{sat}(p_v)$

$$\rightarrow \boxed{m_i'' = \frac{h_i}{h_{lv}}\left(T_i - T_{sat}(p_v)\right)} \quad h_i = \frac{2\gamma}{2-\gamma}\left(\frac{\bar{M}}{2\pi\bar{R}}\right)^{1/2} \frac{\rho_v h_{lv}^2}{{T_{sat}}^{3/2}(p_v)}$$

[1] L. Malan et al., *J Comput Phys* 426 (2021) 109920.
[2] W. H. Lee, Tech. report, Los Alamos Lab. (1980).
[3] S. Hardt and F. Wondra, *J Comput Phys* 227 (2008) 5871.

Phase change

Validation: 1D Stefan problem: Motion of a flat interface due to evaporation. Heat in the vapor phase is transferred by conduction [1,2]:

$$\frac{\partial T}{\partial t} = a_v \frac{\partial^2 T}{\partial x^2}, \qquad 0 < x < x_i(t) \qquad a_v = \frac{\lambda_v}{\rho_v c_{p,v}}$$

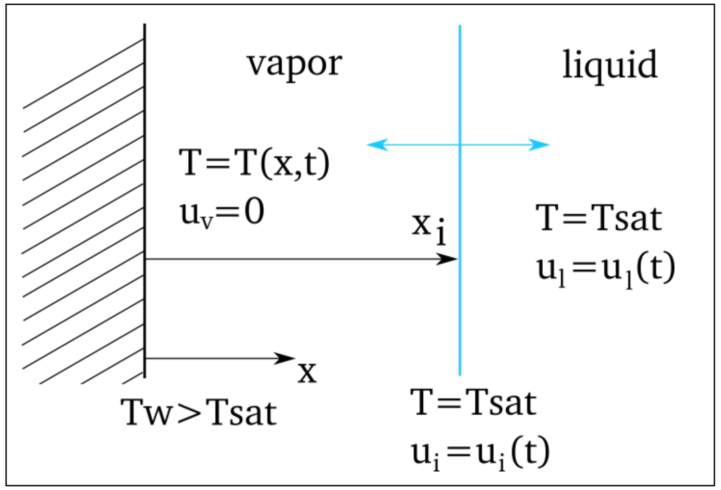
$$\left.\begin{aligned} T(x=0,t) &= T_w \\ T(x=x_i(t),t) &= T_{sat} \end{aligned}\right\}$$ boundary conditions

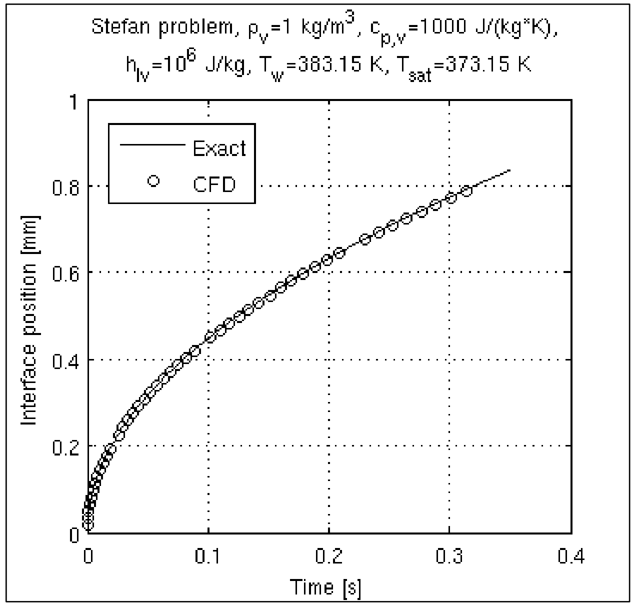
$x_i(t=0) = 0$ initial condition

$$\rho_v h_{lv} u_i = -\lambda_v \left.\frac{\partial T}{\partial x}\right|_{x=x_i(t)}$$ interface energy jump condition

$$\boxed{\begin{aligned} x_i(t) &= 2\beta\sqrt{a_v t} \\ T(x,t) &= T_w + \frac{T_{sat}-T_w}{\mathrm{erf}(\beta)}\mathrm{erf}\left(\frac{x}{2\sqrt{a_v t}}\right) \end{aligned}}$$

with β from: $$\beta\exp(\beta^2)\mathrm{erf}(\beta) = \frac{c_{p,v}(T_w - T_{sat})}{\sqrt{\pi} h_{lv}}$$





[1] S.W.J. Welch and J. Wilson, J. Computational Physics **160** (2000) 662.
[2] H. Scheufler and J. Roenby, https://arxiv.org/abs/2103.00870

Phase change

Validation: 1D sucking interface: heat now provided from the liquid side [1,2]:

$$\xi = x - \int_0^t u_i(t)dt$$

Heat transfer in the liquid phase is governed by mixed conduction and convection:

$$\frac{\partial T}{\partial t} + (u_l - u_i)\frac{\partial T}{\partial \xi} = a_l \frac{\partial^2 T}{\partial \xi^2}, \qquad 0 < \xi < \infty$$

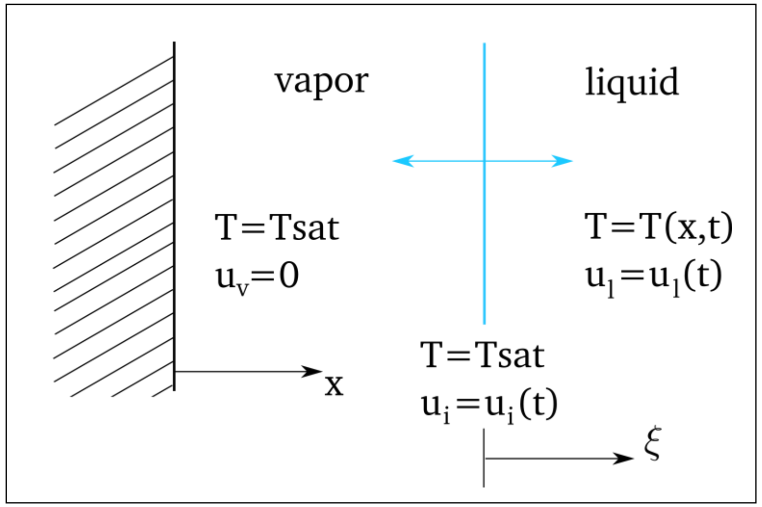
$$\left.\begin{aligned} T(\xi = 0, t) &= T_{sat} \\ T(\xi \to \infty, t) &= T_\infty \end{aligned}\right\} \text{ boundary conditions}$$

$$T(\xi, t = 0) = T_\infty \quad \text{initial condition}$$

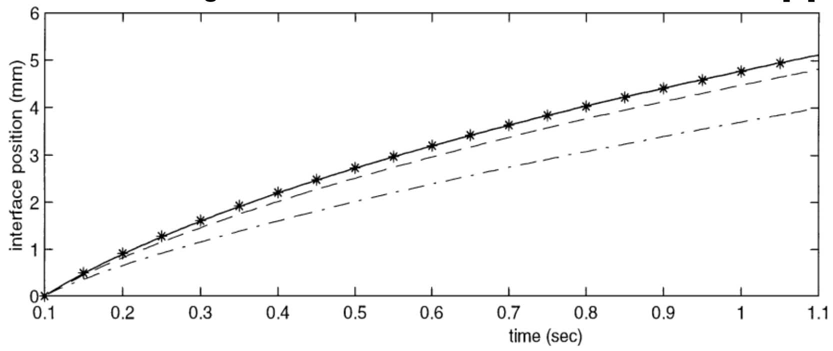
$$-\rho_v u_i = \rho_l (u_i - u_l) \quad \text{normal velocity jump condition}$$

$$\rho_l (u_l - u_i) h_{lv} = -\lambda_l \left.\frac{\partial T}{\partial \xi}\right|_{\xi=0} \quad \text{energy jump condition}$$

$\Rightarrow$ *$x_i(t)$* and *$T(x,t)$* are obtained by numerical solution of a 2nd order ODE



Sucking interface case, from Welch and Wilson [1]



[1] S.W.J. Welch and J. Wilson, J. Computational Physics **160** (2000) 662.
[2] H. Scheufler and J. Roenby, https://arxiv.org/abs/2103.00870

Phase change

Validation: vapour bubble growth: a vapor bubble grows in an extensive pool of uniformly superheated liquid, after homogeneous nucleation. There exist two growth stages [1]:

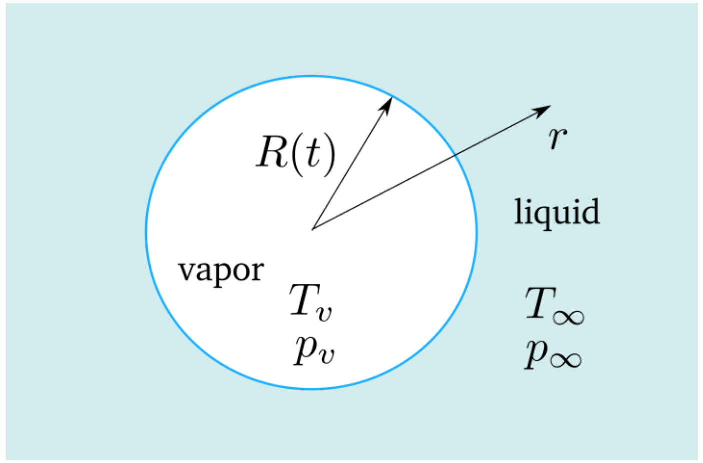
- *Inertia-controlled stage* $(t \downarrow)$

$$p_v \cong p_{sat}(T_\infty), T_v \cong T_\infty$$

$$R(t) = \left\{\frac{2}{3}\left[\frac{T_\infty - T_{sat}(p_\infty)}{T_{sat}(p_\infty)}\right]\frac{\rho_v h_{lv}}{\rho_l}\right\}^{1/2} t$$

- *Heat-transfer-controlled stage* $(t \uparrow)$

$$p_v \cong p_\infty, T_v \cong T_{sat}(p_\infty)$$



Schematic of vapor bubble growth [1]

$$\left.\begin{aligned} T(\infty, t) &= T_\infty \\ T(R, t) &= T_{sat}(p_v) \end{aligned}\right\}$$ Boundary conditions

Heat transfer in the liquid region is governed by:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial r} = \frac{a_l}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T}{\partial r}\right), \qquad R(t) < r < \infty$$

$$u(r,t) = \frac{dR}{dt}\left(\frac{R}{r}\right)^2$$ velocity in the liquid

$$T(r > R, t = 0) = T_\infty$$ Initial condition

Mass and energy conservation:

$$\rho_v h_{lv}\frac{dR}{dt} = \lambda_l \left.\frac{\partial T}{\partial r}\right|_{r=R(t)}$$

[1] V. P. Carey, Liquid-Vapor Phase Change Phenomena (1992).

Phase change

Validation: vapour bubble growth. Analytical solution for heat-transfer-controlled growth stage [1,2]:

$$\boxed{R(t) = 2\beta\sqrt{a_l t}} \qquad \beta = Ja\sqrt{3/\pi} \qquad Ja = \frac{(T_\infty - T_{sat})\rho_l c_{p,l}}{\rho_v h_{lv}}$$

Numerical solution [3]

- 2D axisymmetrical

  domain 0.8×0.4 mm

- Initial bubble radius

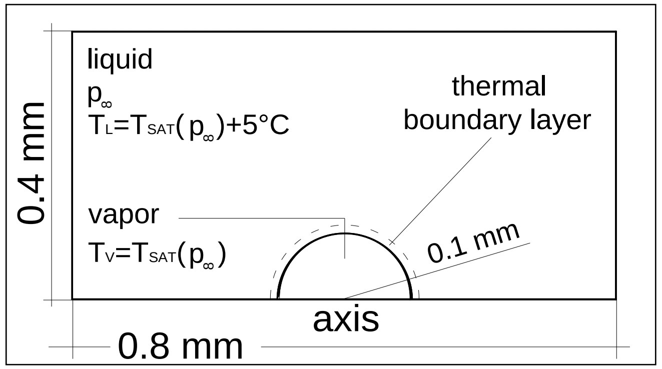
  R_0=0.1 mm

- Mesh size 1 μm

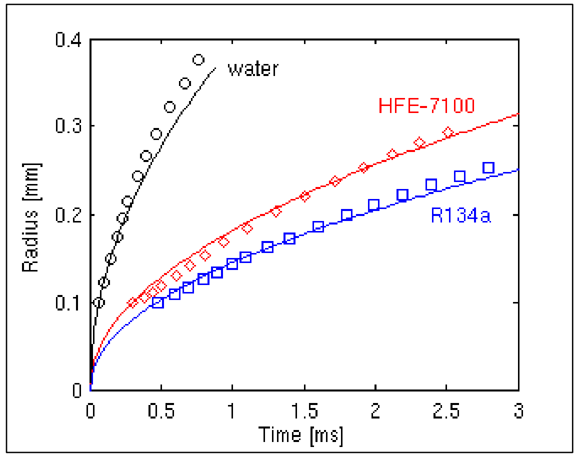
- CFD bubble growth rate:

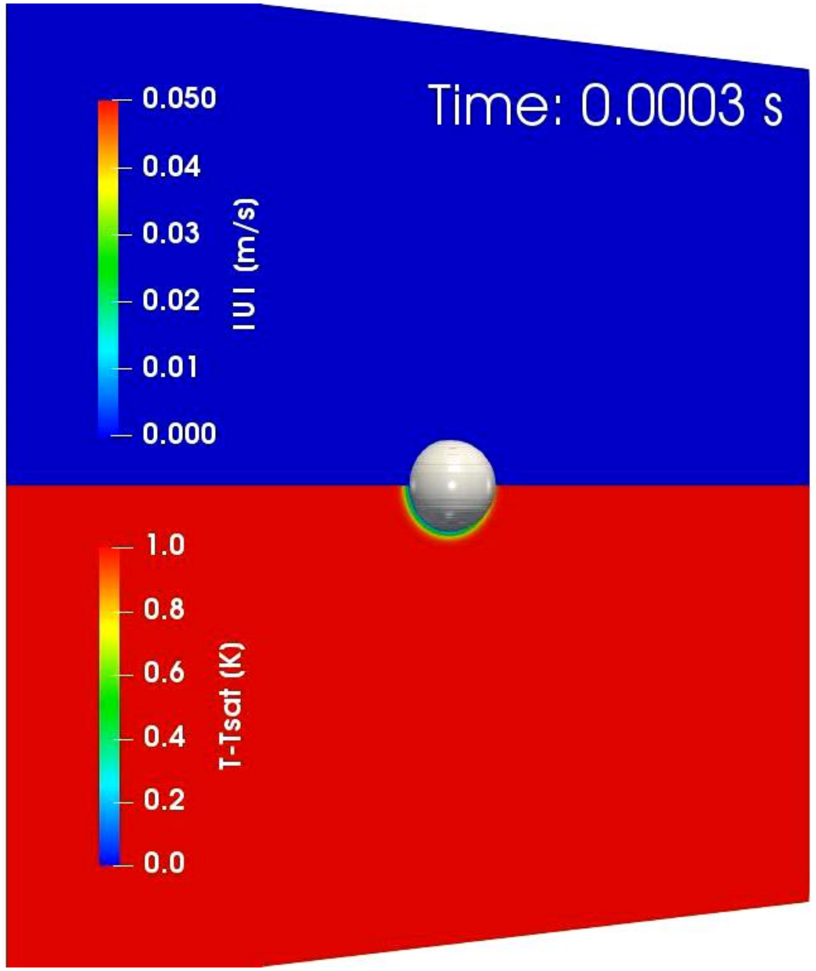
$$R(t) = \left(\frac{3V_b(t)}{4\pi}\right)^{1/3}$$

Simulation setup



Growth vs time





[1] L. E. Scriven, On the dynamics of phase growth, *Chem. Eng. Sci.* 10 (1959) 1.
[2] H. Scheufler and J. Roenby, https://arxiv.org/abs/2103.00870

[3] M. Magnini et al., Int. J. Heat Mass Transfer **59** (2013) 451.

boilingFoam

Incompressible two-phase VOF solver with phase change and conjugate heat transfer [1]

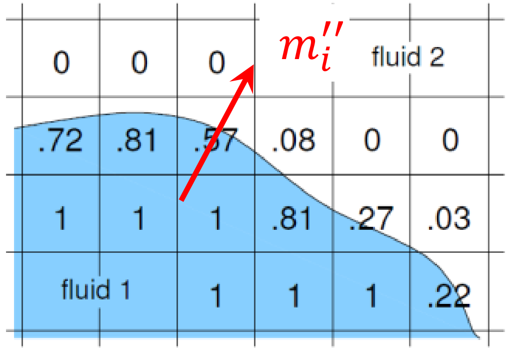
$$\nabla \cdot \boldsymbol{u} = \frac{\dot{\rho}}{\rho}$$

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \boldsymbol{u}) = \frac{\dot{\rho}}{\rho} \alpha$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F_\sigma}$$

$$\frac{\partial (\rho c_p T)}{\partial t} + \nabla \cdot (\rho c_p \boldsymbol{u} T) = \nabla \cdot (\lambda \nabla T) + \dot{h}$$

Solid: $$\frac{\partial (\rho_s c_{p,s} T)}{\partial t} = \nabla \cdot (\lambda_s \nabla T)$$

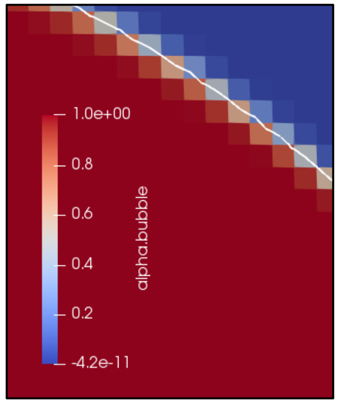


Phase-change: Hardt and Wondra [2]

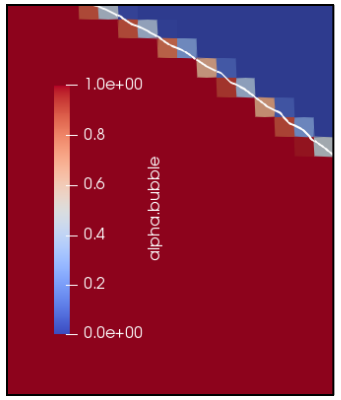
$$m_i'' = \frac{2\gamma}{2-\gamma} \left(\frac{\bar{M}}{2\pi \bar{R}} \right)^{1/2} \frac{\rho_v h_{lv}}{T_{sat}^{\ 3/2}} (T_i - T_{sat})$$

VoF advection: both algebraic and geometric [3]

Algebraic: interFoam



Geometric: isoAdvector



Surface tension: CSF [4]+ρcorr+smoothing

$$\boldsymbol{F_\sigma} = \frac{2\rho}{\rho_l + \rho_v} \sigma \kappa |\nabla \alpha|$$

$$\kappa = \nabla \cdot \left(\frac{\nabla \tilde{\alpha}}{|\nabla \tilde{\alpha}|} \right) \qquad \tilde{\alpha} = \frac{\sum_f \alpha_f S_f}{\sum_f S_f}$$

[1] F. Municchi et al., *Int J Heat Mass Transf* 195 (2022) 123166.
[2] S. Hardt and F. Wondra, *J. Comp Phys* 227 (2008) 5871.
[3] H. Scheufler and J. Roenby, *J. Comp Phys* 383 (2019) 1.
[4] J. U. Brackbill et al., *J. Comp Phys* 100 (1992) 335.

boilingFoam

The evaporation mass source calculated as $m_i''|\nabla\alpha|$ is concentrated on a narrow band of cells. This may yield **numerical instabilities**. Hardt and Wondra devised a smoothing procedure for these source terms:

1. An initial evaporation rate is calculated based on the temperature on the liquid side of the interface:

$$\dot{\rho}_0 = N\alpha_l m_i''|\nabla\alpha| \qquad N = \int_\Omega |\nabla\alpha| d\Omega = \int_\Omega \alpha_l |\nabla\alpha| d\Omega$$

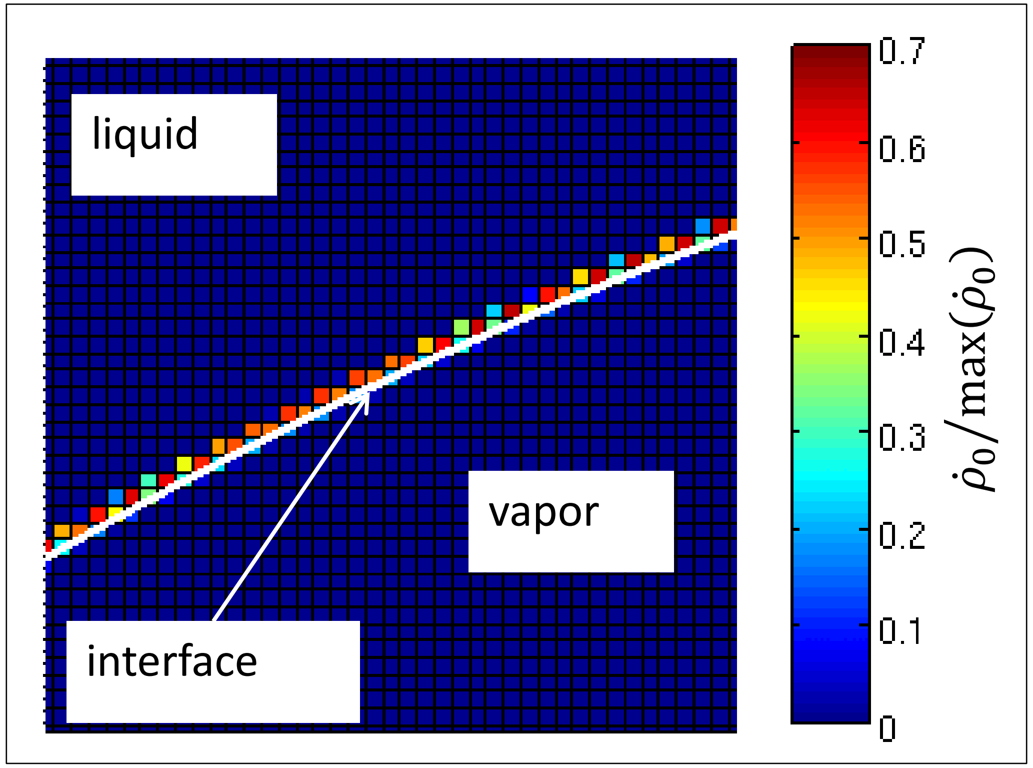
2. A smeared evaporation rate $\dot{\rho}_1$ is obtained by solving:

$$\begin{cases} D\nabla^2\dot{\rho}_1 = \dot{\rho}_1 - \dot{\rho}_0, & \boldsymbol{x} \in \Omega \\ \boldsymbol{n}\cdot\nabla\dot{\rho}_1 = 0, & \boldsymbol{x} \in \partial\Omega \end{cases}$$

   where ***n*** is the unit normal to the domain boundary. The Neumann boundary condition ensures that:

$$\int_\Omega \dot{\rho}_1 d\Omega = \int_\Omega \dot{\rho}_0 d\Omega$$

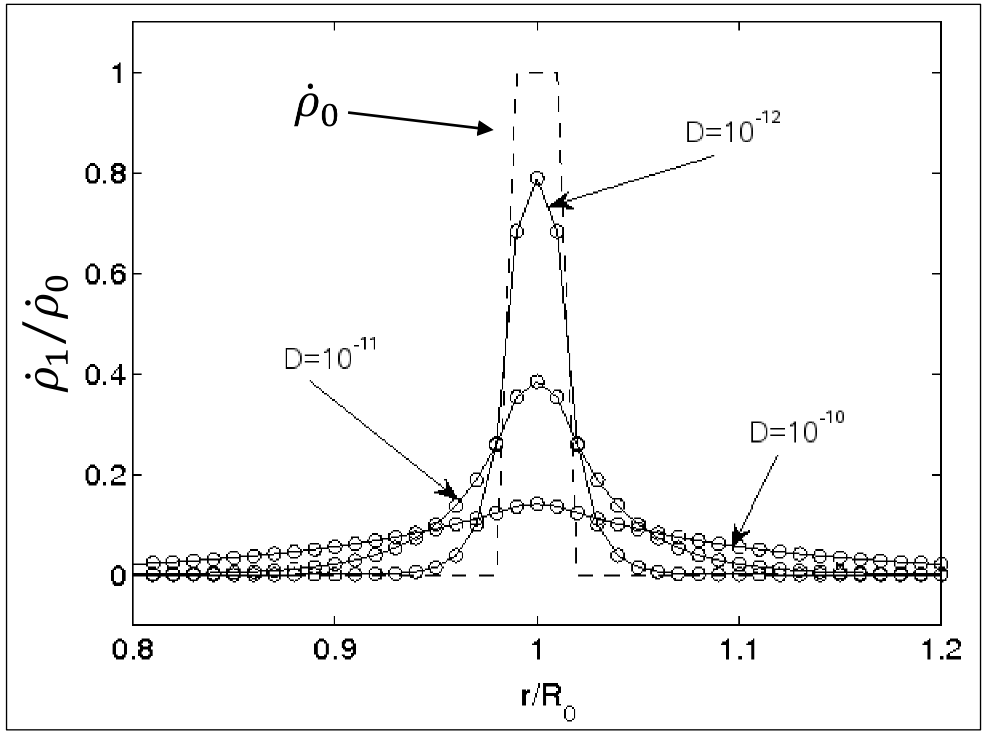
   i.e. the global evaporation rate is preserved.

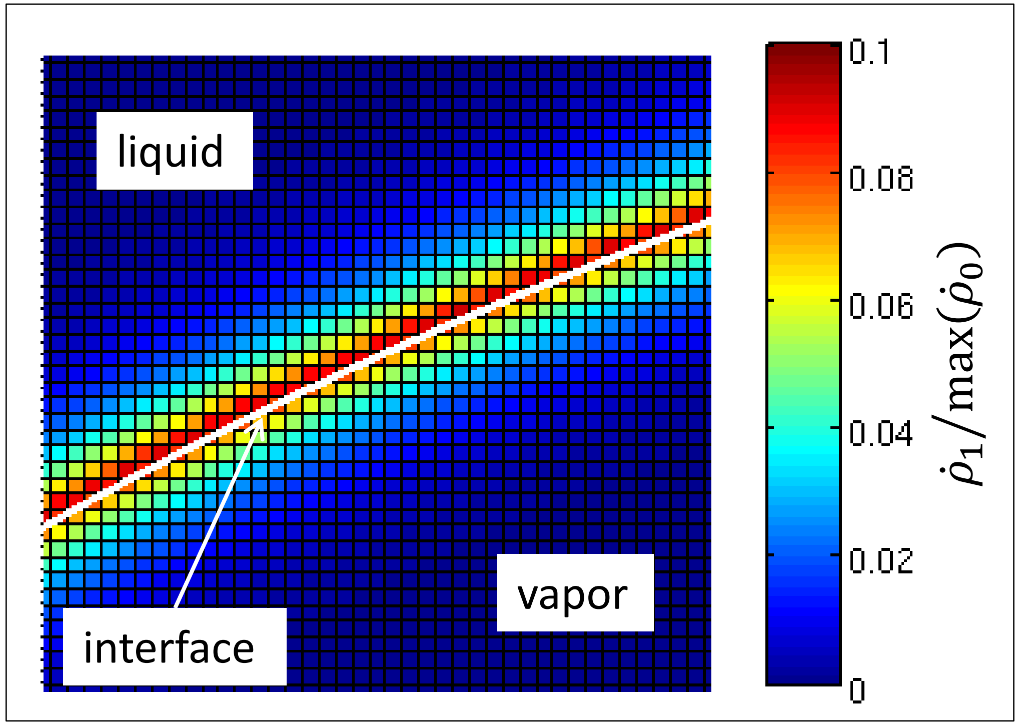


boilingFoam

The value of the diffusion constant *D* tunes the smoothing of the source terms:

$$\begin{cases} D\nabla^2\dot{\rho}_1 = \dot{\rho}_1 - \dot{\rho}_0, & \boldsymbol{x} \in \Omega \\ \boldsymbol{n}\cdot\nabla\dot{\rho}_1 = 0, & \boldsymbol{x} \in \partial\Omega \end{cases} \Longrightarrow \dot{\rho}_1$$





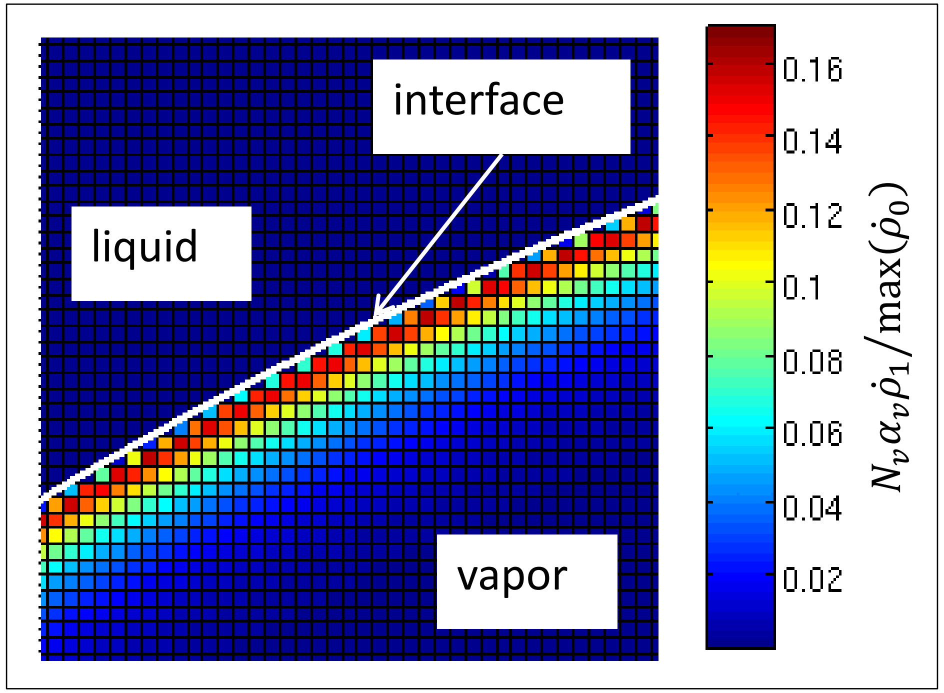
In principle, $\dot{\rho}_1$ locally gives the mass of vapor created and $-\dot{\rho}_1$ that of liquid disappeared.

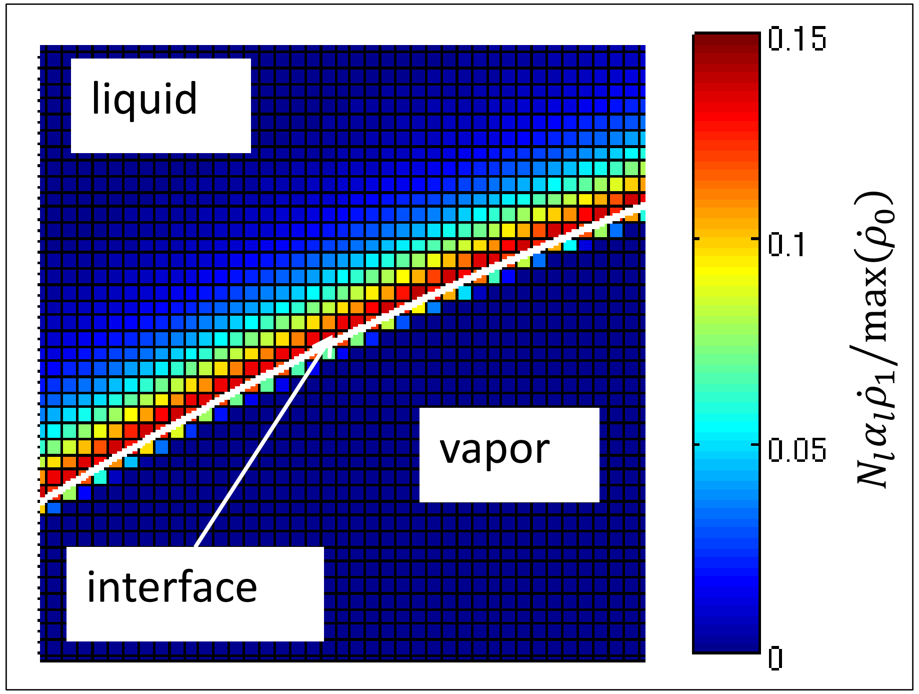
boilingFoam

3. Vapor creation is concentrated on the vapor side of the interface, liquid disappearance on the liquid side, by expressing the final evaporation rate as:

$$\boxed{\dot{\rho} = N_v \alpha_v \dot{\rho}_1 - N_l \alpha_l \dot{\rho}_1} \qquad N_v = \int_\Omega \dot{\rho}_1 d\Omega \Big/ \int_\Omega \alpha_v \dot{\rho}_1 d\Omega, \quad N_l = \int_\Omega \dot{\rho}_1 d\Omega \Big/ \int_\Omega \alpha_l \dot{\rho}_1 d\Omega$$

$$\boxed{\dot{h} = -\dot{\rho}_0 h_{lv}}$$





boilingFoam

The public repository can be found here: https://github.com/fmuni/boilingFoam-PUBLIC

For OpenFoam beginners, an OpenFOAM tutorial covering from the solution of heat conduction problems to boiling flows using boilingFoam is available open-access on the website of the Virtual International Research Institute of Two-Phase Flow and Heat Transfer (VIR2AL): http://2phaseflow.org/publicpages:edutools. The tutorial is based on a Virtual Machine running Ubuntu 20L as operating system, containing a working version of OpenFOAM v2106 already installed in it and boilingFoam already compiled. If you are new to OpenFOAM, that’s a good place to start.

Tutorials for boilingFoam are available in the github repository within the folder *cases*.

Here, we focus on the tutorials based on the previous slides: stefanProblem, suckingInterface, scrivenWedge. These are already all set to run thanks to files *Allrun* and *Allclean*.

These 3 tutorials are adapted from the repository TwoPhaseFlow of Henning Scheufler [1].

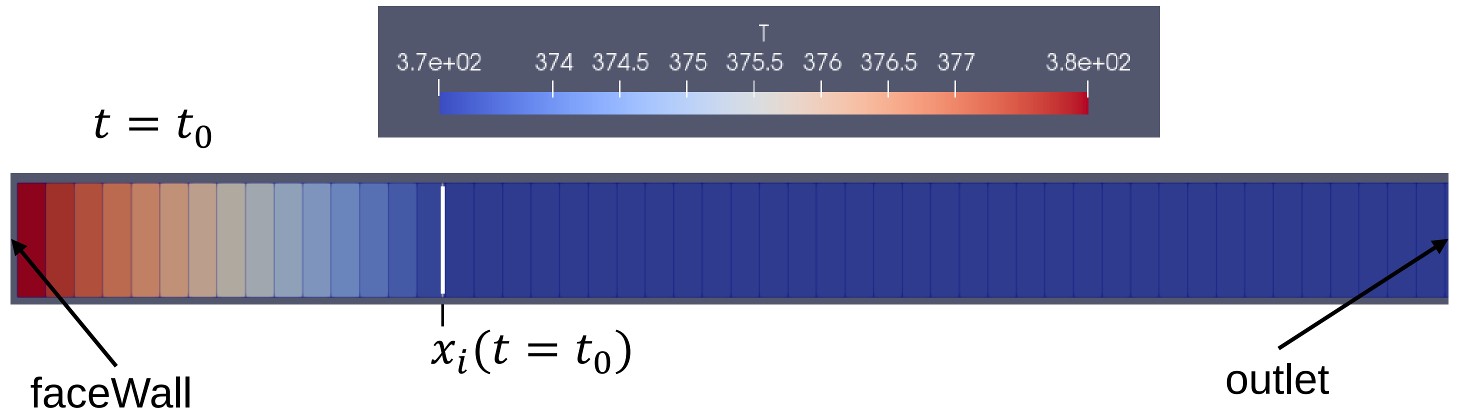
[1] https://github.com/DLR-RY/TwoPhaseFlow/tree/master

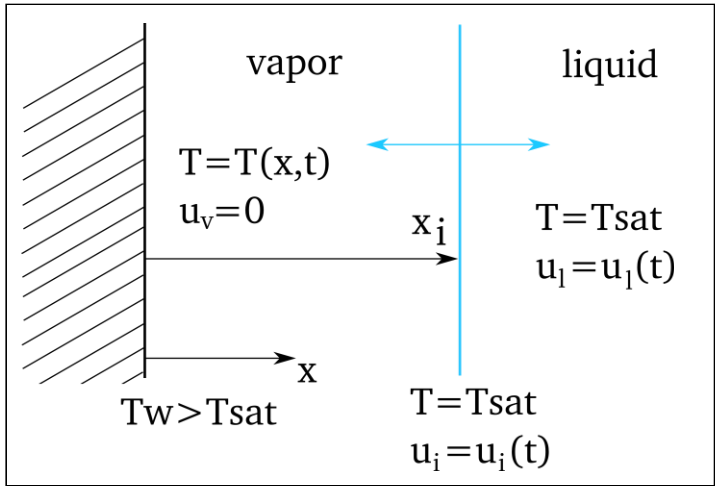
Stefan problem

Let's understand the structure of a boilingFoam case file by looking at the one-dimensional *stefanProblem*.

boilingFoam is a multiregion solver. As such, within 0, constant and system, all files are arranged into subfolders that take names coinciding with the different regions of the domain. The tutorials that we are running only have one fluid region and no other region, as such only one subfolder named *fluid* is present within each OpenFOAM's folder.

The working fluid is water at 1 atm, with Tsat=373.15 K.





	U	p_rgh	alpha.liquid	T	mDot ($\dot{\rho}_1$)
faceWall	*noSlip*	*fixedFluxPressure*	*zeroGradient*	*fixedValue* *value uniform 378.15*	*zeroGradient*
outlet	*zeroGradient*	*fixedValue* *value uniform 0*	*zeroGradient*	*zeroGradient*	*zeroGradient*

Stefan problem

/constant/fluid/transportProperties

```
phases (liquid vapour);

liquid
{
    transportModel          Newtonian;
    nu                      nu [ 0 2 -1 0 0 0 0 ] 2.922e-7;  //mu = 2.8e-4 Pa s
    rho                     rho [ 1 -3 0 0 0 0 0 ] 958.4;
    kappa           0.679;
    Cp              4216;
}

vapour
{
    transportModel      Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 2.11e-5;   //mu = 1.26e-5
    rho             rho [ 1 -3 0 0 0 0 0 ] 0.597;
    kappa           0.025;
    Cp              2030;
}

sigma           sigma [ 1 0 -2 0 0 0 0 ] 0.0;

//- Re-distribute surface tension force based on the fluid density
sigmaRhoCorrection false;

//- Hard bound alpha such that only interface gradients are allowed
alphaInterfaceOnly no;

//- The reconstruction scheme employed must be specified here and it is
//  required even with the MULES
reconstructionScheme    isoAlpha;


curvatureModel
{
    type gradAlpha;

    //- Requires several smoothing iterations for isoAdvector
    KSmoothingIter  0;
}

deltaModel
{
    type gradAlpha;
}
```

- The file is adapted from interFoam. We need to specify the thermal transport properties of the fluids, i.e. specific heat *Cp* and thermal conductivity *kappa*
- *sigmaRhoCorrection true/false:* implements the correction term $2\rho/(\rho_l + \rho_v)$ in the surface tension force (slide 106).
- *alphaInterfaceOnly yes/no*: execute a loop in all the cells of the domain to "clean up" the alpha field far from the interface.
- *reconstructionScheme gradAlpha/isoAlpha/plicRDF*: scheme used to calculate the interface normal vector for use in the surface tension algorithm [1].
- *curvatureModel>type gradAlpha/gradRDF*: field used to compute the interface curvature in the surface tension algorithm [2].
- *KSmoothingIter n*: number of smoothing iterations applied before differentiating alpha [3] or the RDF.
- *deltaModel>type gradAlpha*: only option available to calculate the delta function: $\delta = |\nabla\alpha|$

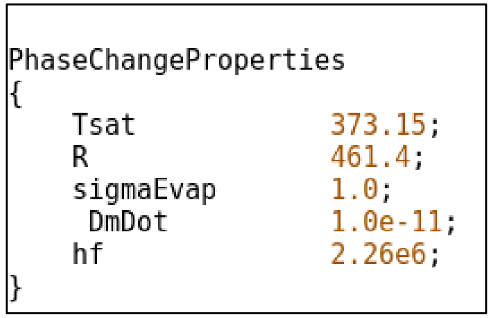
[1] H. Scheufler and J. Roenby, *J. Comp Phys* 383 (2019) 1.
[2] H. Scheufler, J. Roenby, arXiv:2103.00870.
[3] F. Municchi et al., *Int J Heat Mass Transf* 195 (2022) 123166.

Stefan problem

boilingFoam adopts Hardt and Wondra [1] phase change model:

$$m_i'' = \frac{h_i}{h_{lv}}(T_i - T_{sat}), \qquad h_i = \frac{2\gamma}{2-\gamma}\left(\frac{\bar{M}}{2\pi\bar{R}}\right)^{1/2}\frac{\rho_v h_{lv}^2}{T_{sat}{}^{3/2}}$$

/constant/fluid/transportProperties

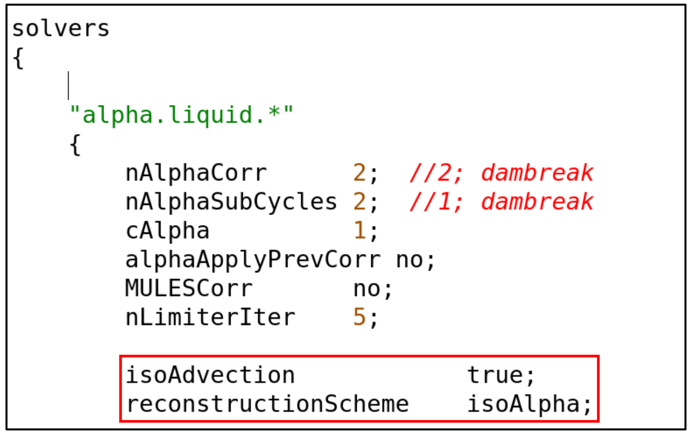


```
PhaseChangeProperties
{
    Tsat            373.15;
    R               461.4;
    sigmaEvap       1.0;
     DmDot          1.0e-11;
    hf              2.26e6;
}
```

- *Tsat*: Saturation temperature of the fluid. *hf*: latent heat.
- *R*: Universal gas constant, 8314 J/(kmol*K), divided by the molar mass of the fluid, e.g. water 18 kg/kmol. It coincides with the ratio $\bar{R}/\bar{M}$.
- *DmDot*: dimensional [m^2] smoothing coefficient in Hardt and Wondra method, used to distribute the evaporation source term across the interface. This is usually set to about Δ^2, where Δ is the mesh size.
- *sigmaEvap*: evaporation coefficient γ.

Interface advection. boilingFoam can run in algebraic VOF (MULES) or geometric VOF (isoAdvector [2]) mode:

- *isoAdvection true/false*.
- If isoAdvection is set to true, the *reconstructionScheme* (*isoAlpha, isoRDF, plicRDF*) must be specified.



```
solvers
{

    "alpha.liquid.*"
    {
        nAlphaCorr      2;  //2; dambreak
        nAlphaSubCycles 2;  //1; dambreak
        cAlpha          1;
        alphaApplyPrevCorr no;
        MULESCorr       no;
        nLimiterIter    5;

        isoAdvection            true;
        reconstructionScheme    isoAlpha;
```

/system/fluid/fvSolutions

[1] S. Hardt and F. Wondra, *J. Comp Phys* 227 (2008) 5871.
[2] H. Scheufler and J. Roenby, *J. Comp Phys* 383 (2019) 1.

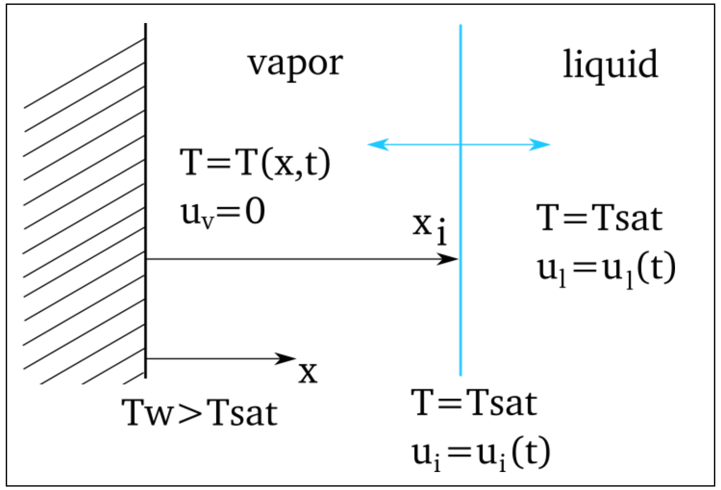
Stefan problem

Initial conditions. The Stefan problem has analytical solution for the time-dependent interface location and temperature field:

$$x_i(t) = 2\beta\sqrt{a_v t}$$

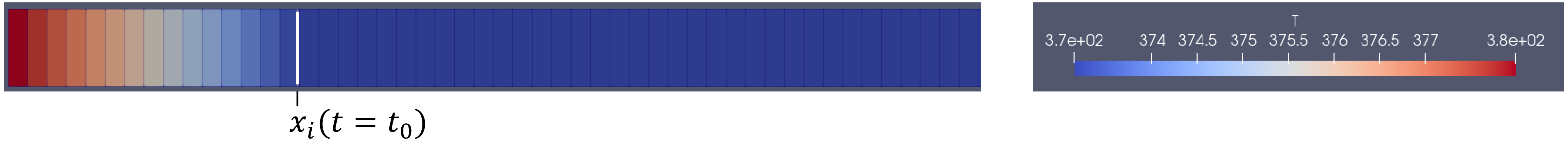
$$T(x,t) = T_w + \frac{T_{sat} - T_w}{\mathrm{erf}(\beta)}\mathrm{erf}\left(\frac{x}{2\sqrt{a_v t}}\right)$$

with β from: $$\beta\exp(\beta^2)\mathrm{erf}(\beta) = \frac{c_{p,v}(T_w - T_{sat})}{\sqrt{\pi}h_{lv}}$$



The reference t=0 is considered to be the instant where x_i=0.

Since the simulation must start from the interface being x_i>0, we consider that the initial time is the arbitrary value t_0=0.03 s, which is sufficient for the initial position of the interface to be a few cells downstream the left boundary. The corresponding $x_i(t_0)$ and $T(x,t_0)$ are pre-calculated by the python code python/setup.py and saved into the files python/initPos.H and python/T0.csv. These files are used to set the initial conditions of the simulation via OpenFOAM's files system/fluid/setAlphaFieldDict and system/fluid/setFieldFromTableDict.



Stefan problem

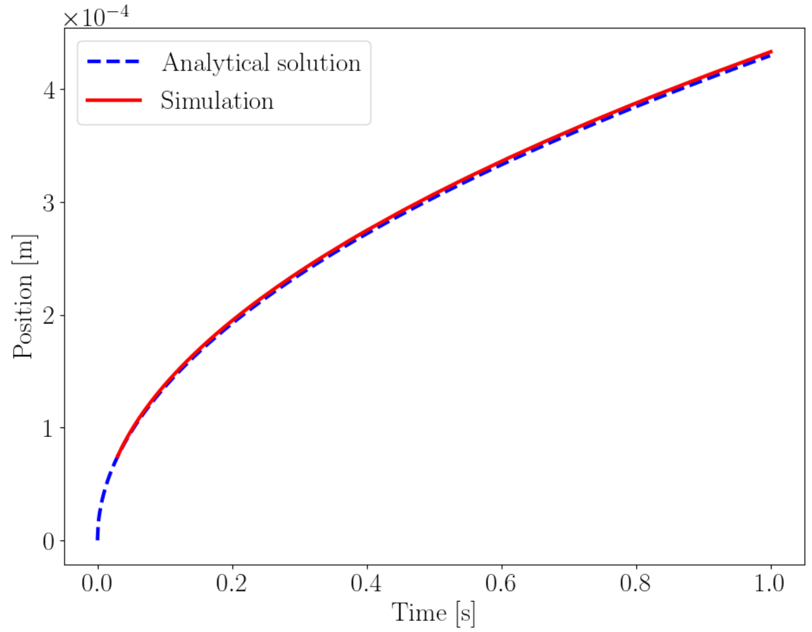
The case is run using the file Allrun. Note that all OpenFOAM utilities require the option *-region fluid* due to the fact that the solver is multiregion. To run it:

1. Browse into the stefanProblem folder
2. Open a terminal and load the OpenFOAM environment
3. Enter "./Allrun"

Allrun

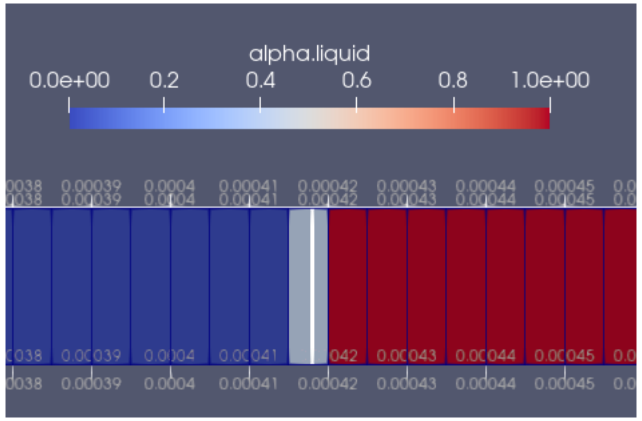
```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

runApplication blockMesh -region fluid
cp -r 0.orig 0.03
runApplication setFieldfromTable -region fluid
runApplication setAlphaField -region fluid
runApplication $(getApplication)
python3 python/plotBenchmark.py
```

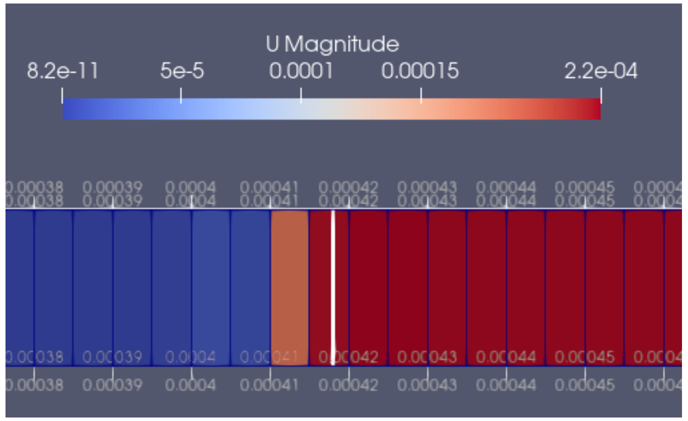
The final python3 command within Allrun will plot the comparison of the result of the simulation in terms of x(t) with the analytical solution:

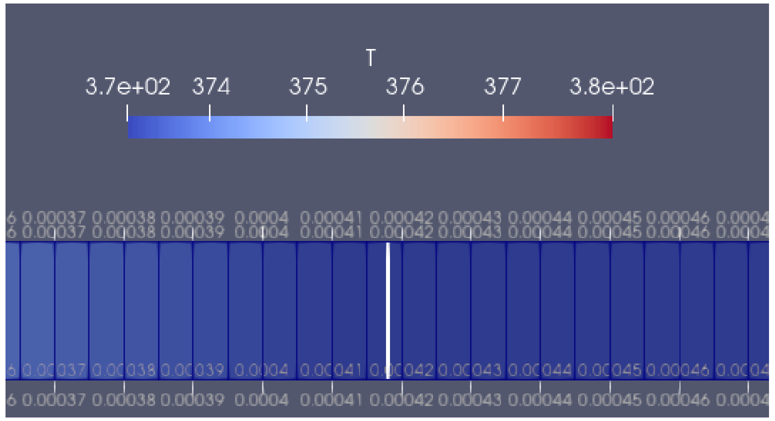


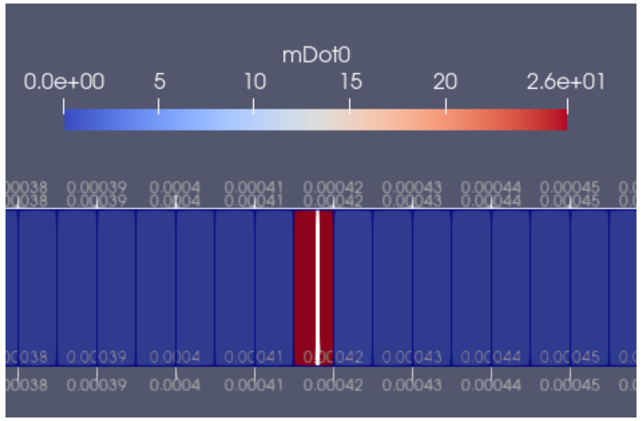
Stefan problem

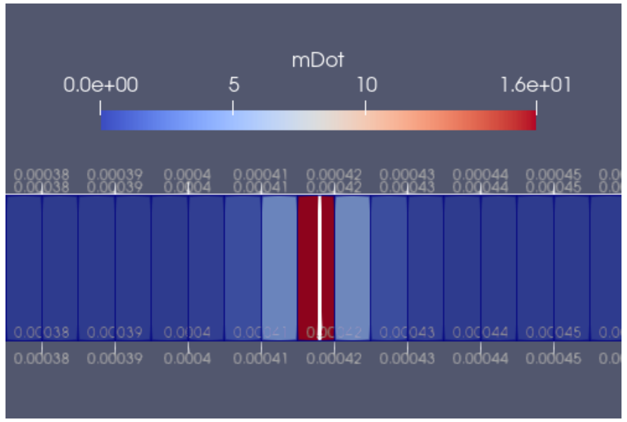
We now look at some of the results using Paraview, at time t=0.93 s.











Observations.

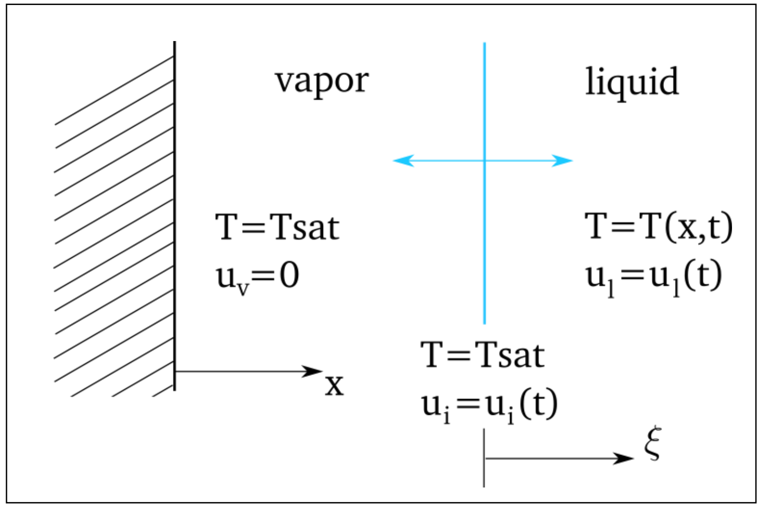
- mDot0 is localised on one cell.
- mDot is distributed across about 5 cells after the smoothing procedure on mDot0.
- Temperature decays linearly from x=0 to x_i.
- Vapour is stagnant
- The liquid speed is constant and equal to 0.22 mm/s. This is consistent with the mass balance at the interface [1]:

$$U_l - U_v = m_i''\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right), m_i'' = \frac{\dot{\rho}_0}{|\nabla\alpha|}, |\nabla\alpha| = \frac{1}{\Delta}$$

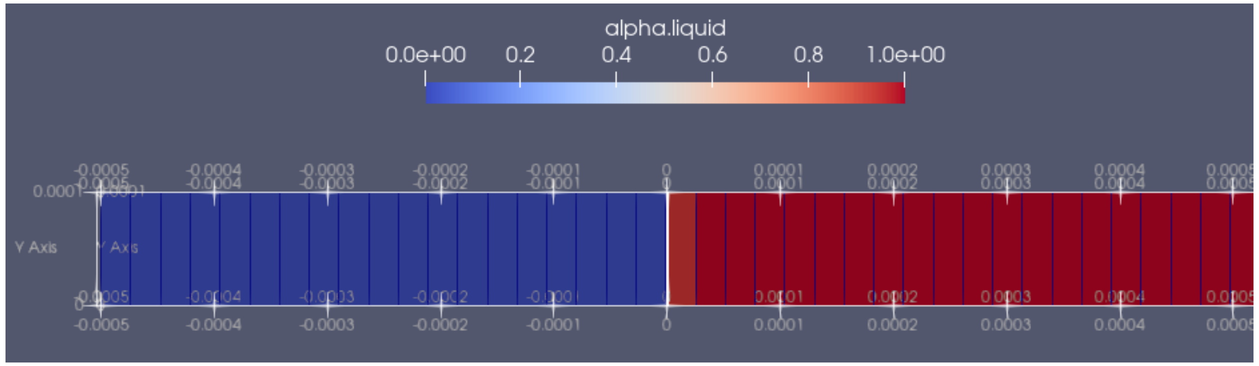
[1] V. P. Carey, Liquid-Vapor Phase Change Phenomena (1992).

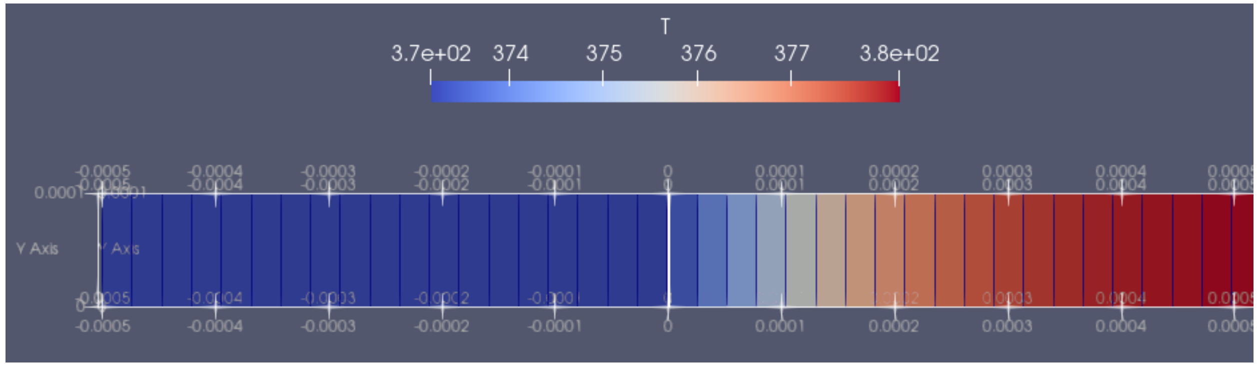
Sucking interface

The one-dimensional sucking interface test case is very similar to the Stefan problem. The only difference is that now the wall in contact with vapour is at Tsat, and heat is provided from the liquid side by setting its initial temperature to a value above Tsat. The working fluid is water at 1 atm, with Tsat=373.15 K.



$t = t_0$





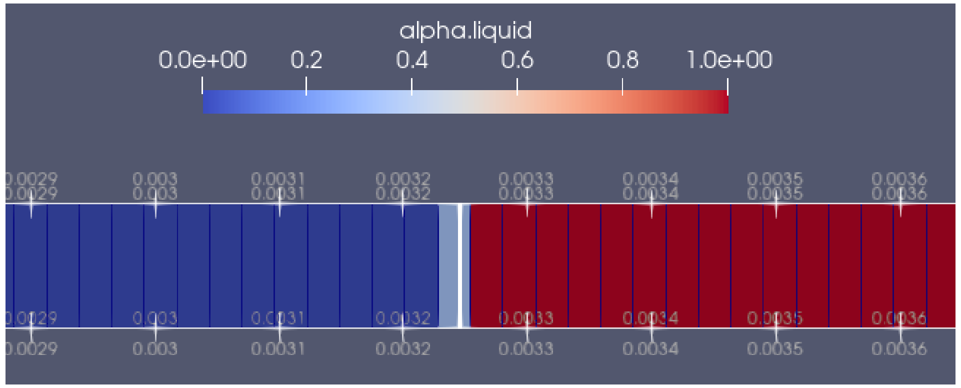
The simulation setup is almost identical to that of stefanProblem. Now t_0=0.1 s and the initial interface position is set to $x_i(t_0)$=0. The temperature field at t_0 to be used as initial condition is saved in the file T01.csv. This file has been generated using Henning Scheufler's library TwoPhaseFlow [1], where the ODE governing the problem is solved using python.

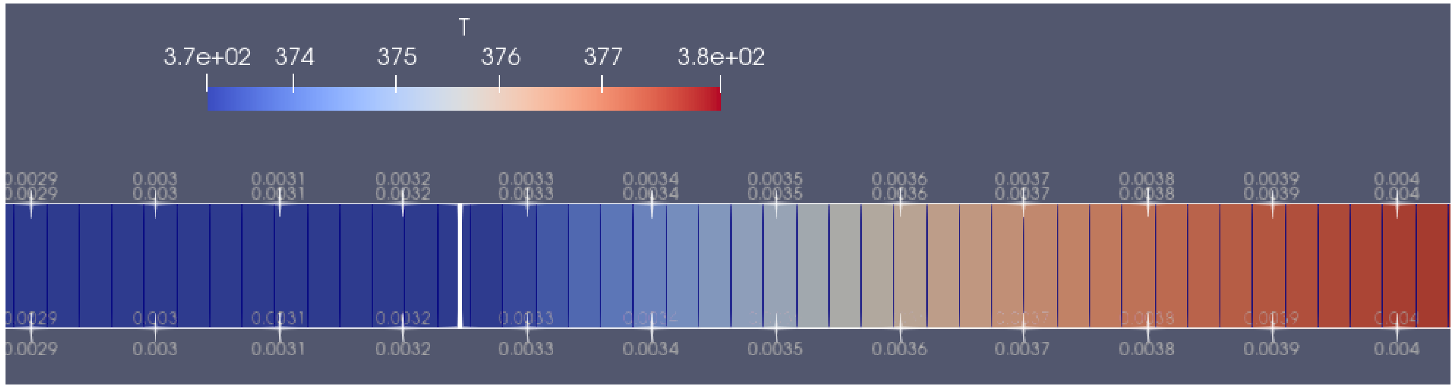
[1] https://github.com/DLR-RY/TwoPhaseFlow/tree/master

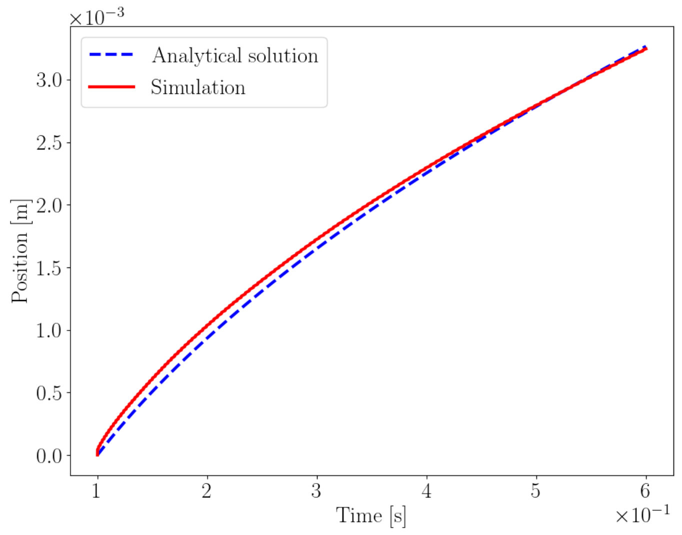
Sucking interface

The simulation setup is almost identical to that of the Stefan problem, and thus is not repeated here. The setup is assembled in cases/suckingInterface and can be run using ./Allrun. At the end of the simulation (t=0.6 s), Python will plot the comparison of the interface position versus time obtained with the simulation and compare it with the analytical solution stored in python/data.dat.

$t = 0.6\ s$







Observations:

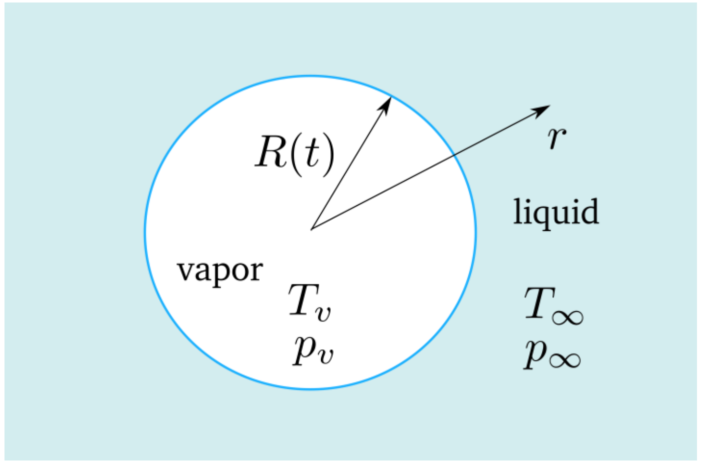
- There is a slight mismatch between analytical and numerical solution due to the coarse grid used. Agreement improves upon refinement of the mesh.
- The interface moves towards the right and the thermal boundary layer within the liquid moves with it.

Scriven problem

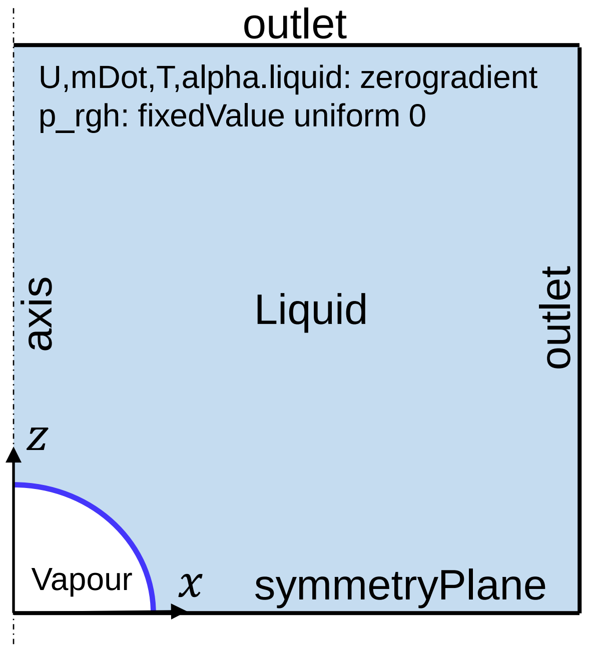
We simulate a vapour bubble growing in superheated liquid with a two-dimensional axisymmetric setup. This problem has analytical solution [1]:

$$R(t) = 2\beta\sqrt{a_l t} \qquad \beta = Ja\sqrt{3/\pi} \qquad Ja = \frac{(T_\infty - T_{sat})\rho_l c_{p,l}}{\rho_v h_{lv}}$$

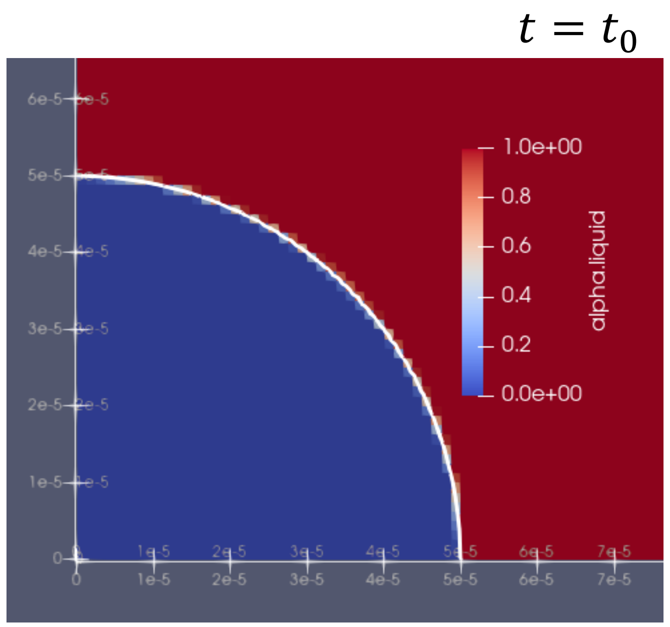
At t_0, a bubble of initial size $R_{b,0}$=50 μm is seeded. This corresponds to time t_0=0.00033 s if the bubble had R=0 at t=0. The initial temperature field at t_0 is saved in the file T01.csv, which was created using the TwoPhaseFlow library [2].

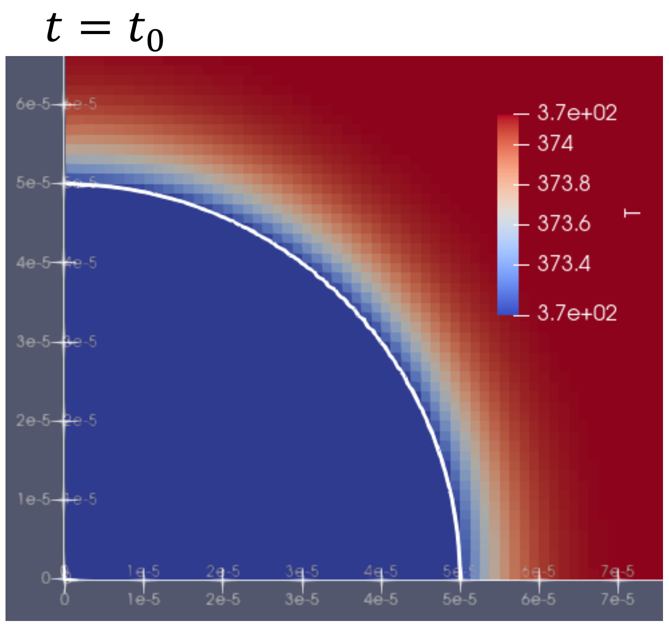


Schematic of vapour bubble growth



outlet
U,mDot,T,alpha.liquid: zerogradient
p_rgh: fixedValue uniform 0
axis
Liquid
outlet
z
Vapour
x
symmetryPlane

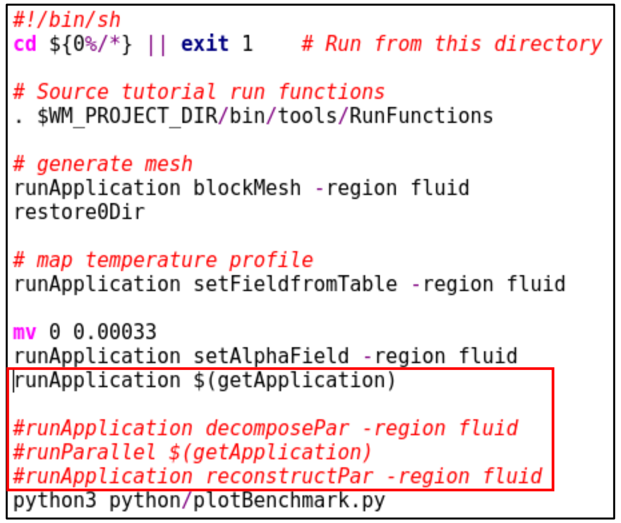




[1] L. E. Scriven, On the dynamics of phase growth, *Chem. Eng. Sci*. 10 (1959) 1.
[2] https://github.com/DLR-RY/TwoPhaseFlow/tree/master

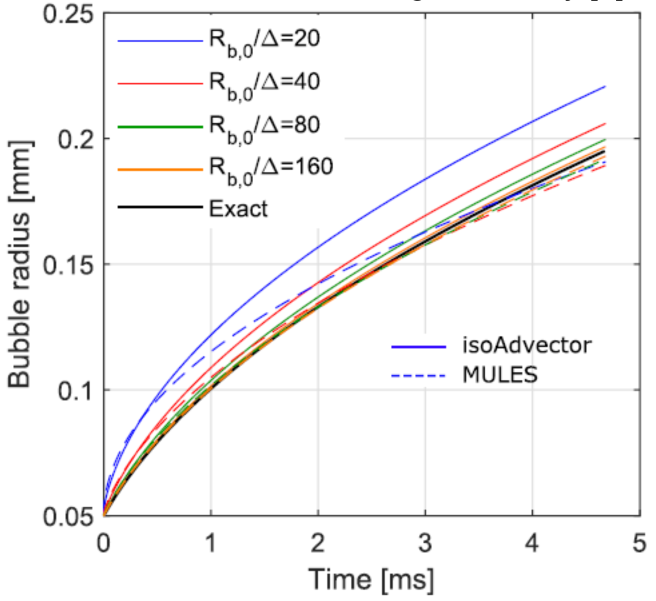
Scriven problem

The setup is assembled in cases/scrivenWedge and can be run using ./Allrun. At the end of the simulation (t=0.005 s), Python will plot the bubble radius versus time and compare it with the analytical solution stored in python/data.dat. The setup in the repository uses 400x400 cells (Δ=1.25 μm, $R_{b,0}/\Delta=40$) and would take about 24 hours to complete using 4 cores. If you are impatient, you can run it using 100x100 cells, which should take about 10 min in a single core; modify the Allrun file as indicated here →

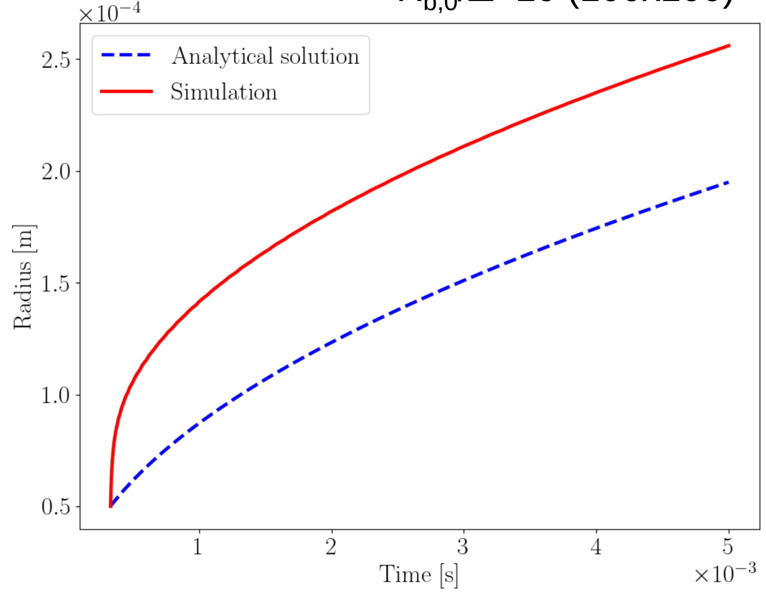


```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

# generate mesh
runApplication blockMesh -region fluid
restore0Dir

# map temperature profile
runApplication setFieldfromTable -region fluid

mv 0 0.00033
runApplication setAlphaField -region fluid
runApplication $(getApplication)

#runApplication decomposePar -region fluid
#runParallel $(getApplication)
#runApplication reconstructPar -region fluid
python3 python/plotBenchmark.py
```

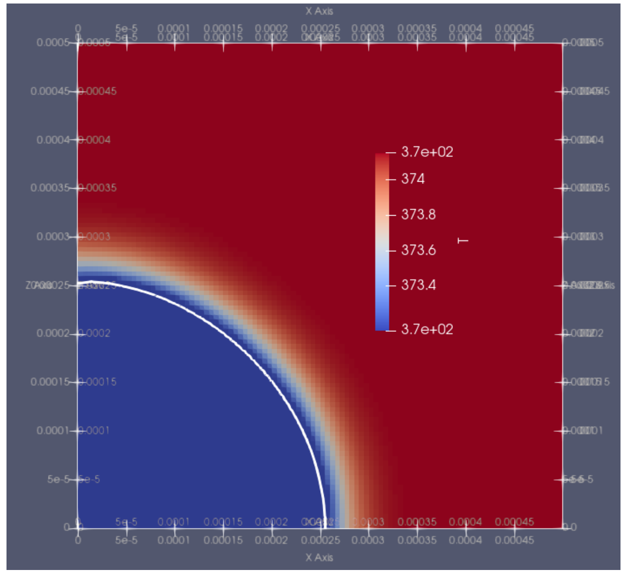
Mesh convergence study [1]



$R_{b,0}/\Delta=10$ (100x100)



t=0.005 s, $R_{b,0}/\Delta=10$ (100x100)



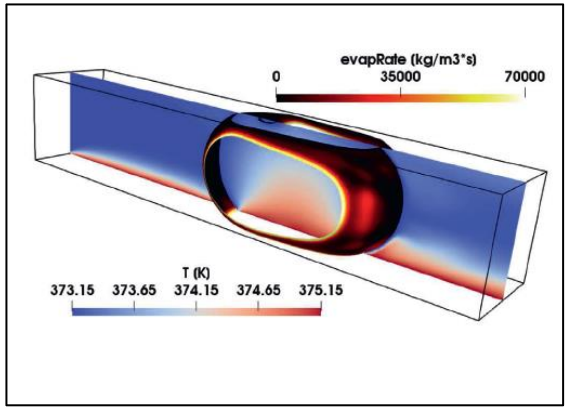
[1] F. Municchi et al., *Int J Heat Mass Transf* 195 (2022) 123166.

boilingFoam-PUBLIC/cases

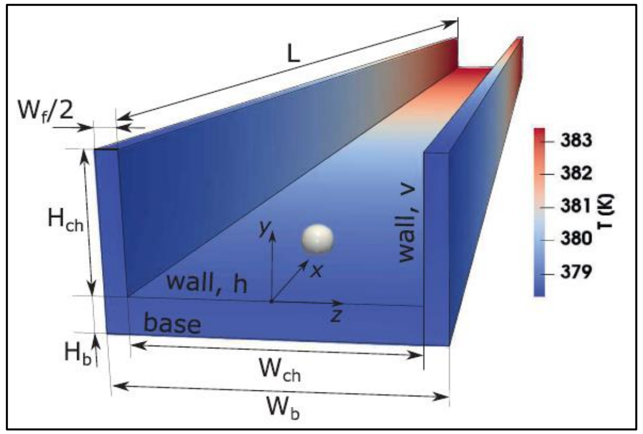
The repository of boilingFoam contains some other prepared case files:

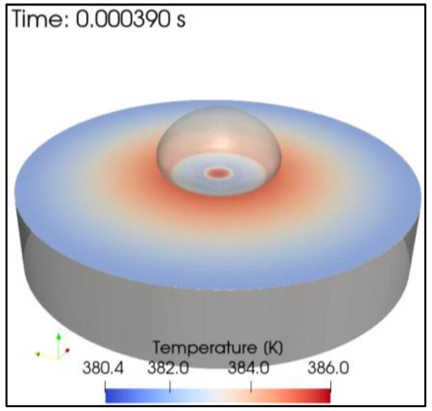
- mukherjee2011: 3D simulation setup to reproduce the experiments of Mukherjee et al. [1]. Results published by Municchi et al. [2].
- flowBoilingCHT_AR1_water_q100k: 3D simulation setup to run a flow boiling simulation with conjugate heat transfer (CHT) in a channel of aspect ratio AR=1, using water and a base heat flux of 100 kW/m^2. Results published by Municchi et al. [2].
- nucleateBoiling_BuresSato: 2D-axisymmetric simulation setup to run a nucleate boiling case based on Bures and Sato [3] numerical setup for Bucci's [4] experiment. Results published by Municchi et al. [5].

mukherjee2011



flowBoilingCHT_AR1_water_q100k





nucleateBoiling_BuresSato

[1] A. Mukherjee et al., *Int. J. Heat Mass Transf*. 54 (2011) 3702.
[2] F. Municchi et al., *Int J Heat Mass Transf* 195 (2022) 123166.
[3] L. Bures and Y. Sato, *J Fluid Mech* 933 (2022) A54.
[4] M. Bucci, Master thesis, U. of Pisa (2020). Link
[5] F. Municchi et al., *Appl Therm Eng 238* (2024) 122039.