FEMSIM - FINITE ELEMENT METHOD SIMULATOR

1st Workshop on Advances in CFD and MD modeling of Interface Dynamics in Capillary Two-Phase Flows

Gustavo R. ANJOS, Erik GROS

http://www.uerj.br http://2phaseflow.org

http://www.gesar.uerj.br http://www.gustavorabello.org

Lausanne - Switzerland

October 5th, 2016

OUTLINE

- Bibliography;
- Intro to femSIM2d;
- interface modeling;
- mesh storage;
- remeshing;
- code structure in C++
- the Python API
- Tasks: 2D examples;

BIBLIOGRAPHY

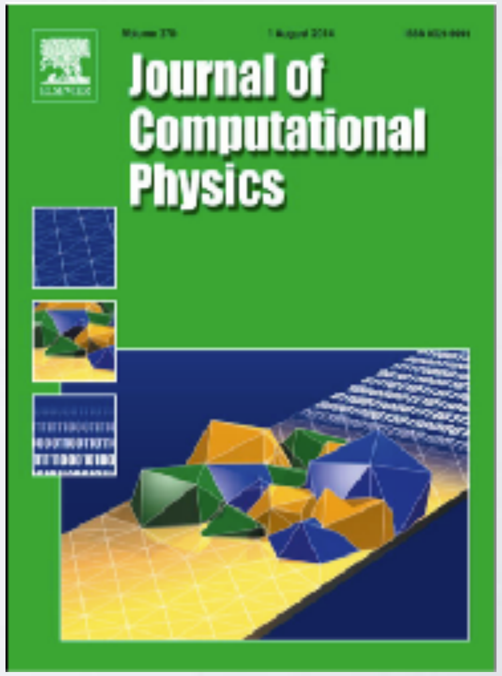
PhD thesis

A 3D ALE Finite Element Method for Two-Phase Flows with Phase Change
Thèse . 5426 2012
EPFL, 2012

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

JCP article

The Finite Element Method - Linear Static and Dynamic Finite Element Analysis
Authors: Thomas J.R. Hughes



OPEN SOURCE APPS

Below is a list of recommended softwares for visualization, text editing, mesh generation, linear system solvers and APIs. All softwares are open source.

- Paraview - 2D and 3D rendering
- Vim - text editor
- Boost-Python - C++/Python bindings
- PETSc - data structure and linear solvers
- tetgen - tetrahedral mesh generator
- triangle - triangular mesh generator
- Gmsh - 2D and 3D mesh generator
- gnuplot - portable command line graphic utility
- git - git repository (see github)

UNIX COMMANDS

```
/
├── /root
├── /usr
│   ├── /usr/include
│   └── /usr/lib
├── /opt
├── /home
│   ├── /Users/tuto01
│   └── /home/femsim
└── /bin
```

- pwd - check the current path
- ssh - connect to another machine
  - Ex.: ssh username@IPaddress
- cd - change directory:
  - Ex.: cd $HOME/projects
- ls - list files and folders
  - Ex.: ls -l
  - ls -G (for coloring - mac users)
- mv - move files:
  - Ex.: mv oldfilename newfilename
- vim - text editor:
  - Ex.: vim filename
- tail - show the last few lines of file:
  - Ex.: tail filename
- head - show the first few lines of file:
  - Ex.: head filename
- find - find files by name:
  - Ex.: find . -name aaa.txt
- alias - rename command
  - Ex.: alias ls = 'ls -G'
- in gnuplot:
  - Ex.: plot 'filename.dat' using 1:2

CECAM COMPUTERS

SUSE Linux

username	IP
femsim	node000
	node001
	node002 (not working)
	node003
	node004
	node005
	node006
	node007
	node008
	node009
	node010
	node011
	node012
	node013
	node014
	node015

obs.: nodes accessible from cecampc4@epfl.ch
Ex.: ssh femsim@node003

Apple OS X

username	IP
tuto01	128.178.157.171
tuto02	128.178.157.192
tuto03	128.178.157.193
tuto04	128.178.157.194
tuto05	128.178.157.195
tuto06	128.178.157.196
tuto07	128.178.157.197
tuto08	128.178.157.198
tuto09	128.178.157.199
tuto10	128.178.157.200
tuto11	128.178.157.161
tuto12	128.178.157.162
tuto13	128.178.157.163
tuto14	128.178.157.164
tuto15	128.178.157.165
tuto16	128.178.157.175
tuto17	128.178.157.174
tuto18	128.178.157.167
tuto19	128.178.157.188
tuto20	128.178.157.186

GOVERNING EQUATIONS

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{c} \cdot \nabla \mathbf{v} = -\frac{1}{\rho(\phi)} \nabla p + \frac{1}{Re} \nabla \cdot [\mu(\phi)(\nabla \mathbf{v} + \nabla \mathbf{v}^T)] + \frac{1}{Fr^2} \mathbf{g} + \frac{1}{We} \underbrace{\mathbf{f}}_{\text{surface tension}}$$

$$\nabla \cdot \mathbf{v} = 0$$

$$\frac{\partial T}{\partial t} + \mathbf{c} \cdot \nabla T = \frac{1}{RePr} \nabla \cdot (k(\phi) \nabla T)$$

ALE formulation: $\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} - \hat{\mathbf{v}}) \cdot \nabla \mathbf{v} \begin{cases} \hat{\mathbf{v}} = \mathbf{v} \rightarrow \text{Lagrangian} \\ \hat{\mathbf{v}} = 0 \rightarrow \text{Eulerian} \end{cases}$

GOV EQ - AXISYMMETRIC

$$\frac{\partial v_x}{\partial t} + \mathbf{c} \cdot \nabla v_x = -\frac{1}{\rho(\phi)} \frac{\partial p}{\partial x} + \frac{1}{Re} \mu(\phi) \nabla^2 v_x + \frac{1}{Fr^2} g_x$$

$$\frac{\partial v_r}{\partial t} + \mathbf{c} \cdot \nabla v_x = -\frac{1}{\rho(\phi)} \frac{\partial p}{\partial x} + \frac{1}{Re} \mu(\phi) \left(\nabla^2 v_r - \frac{v_r}{r^2} \right)$$

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_r}{\partial r} + \frac{v_r}{r} = 0$$

Laplacian operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r}$$

curvature

$$\kappa = \kappa_{2d} + \frac{1}{R} = \kappa_{2d} + \frac{\sin(\theta)}{r}$$

FINITE ELEMENT METHOD

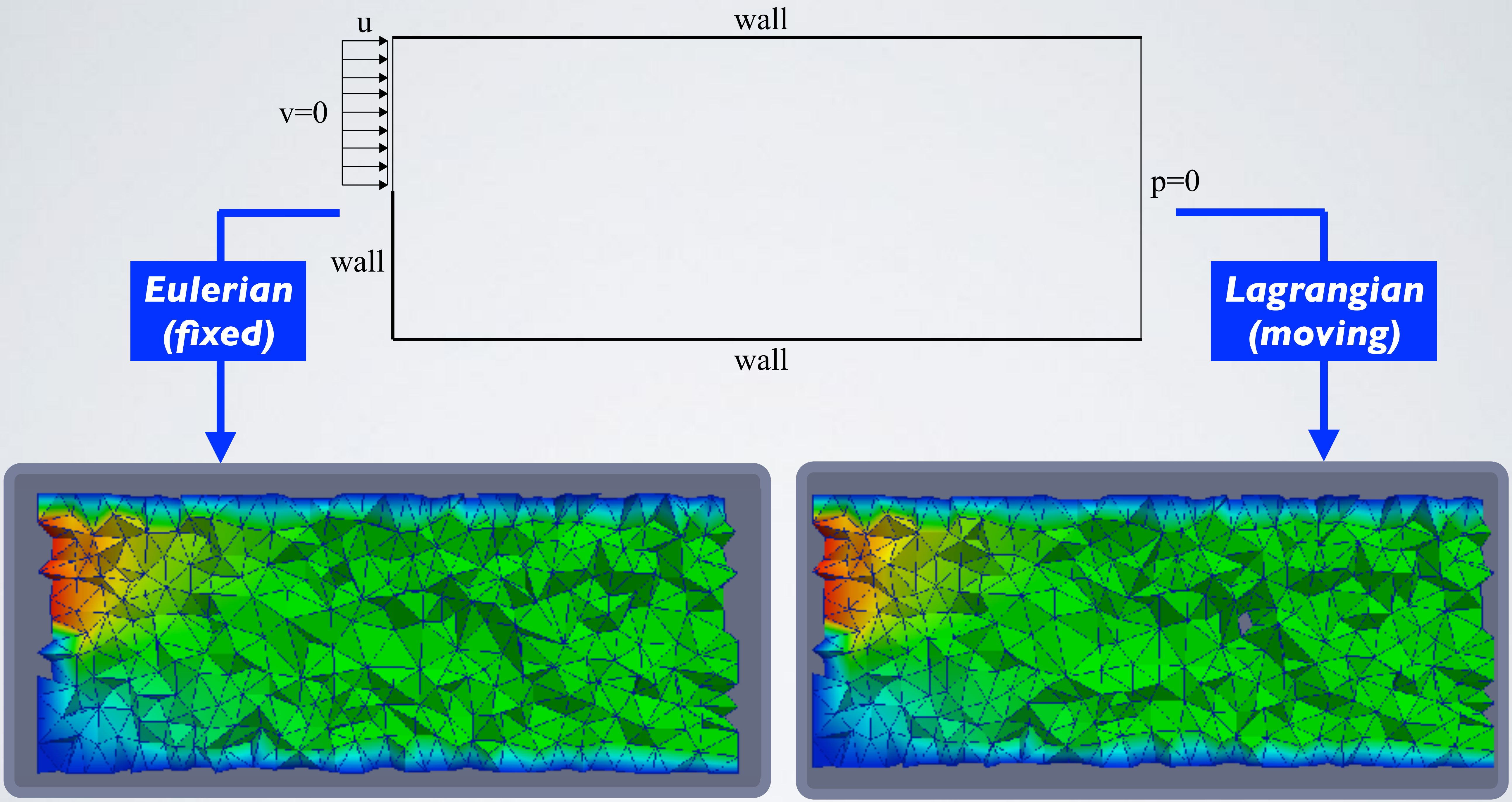
PDE → weak form → ODE → linear system → approximated solution

PDE	weak form	ODE	linear system	approximated solution
Governing equations	Galerkin Method	Ordinary Differential Equations	Ax=b	u,v,w,p and T

Code's features:

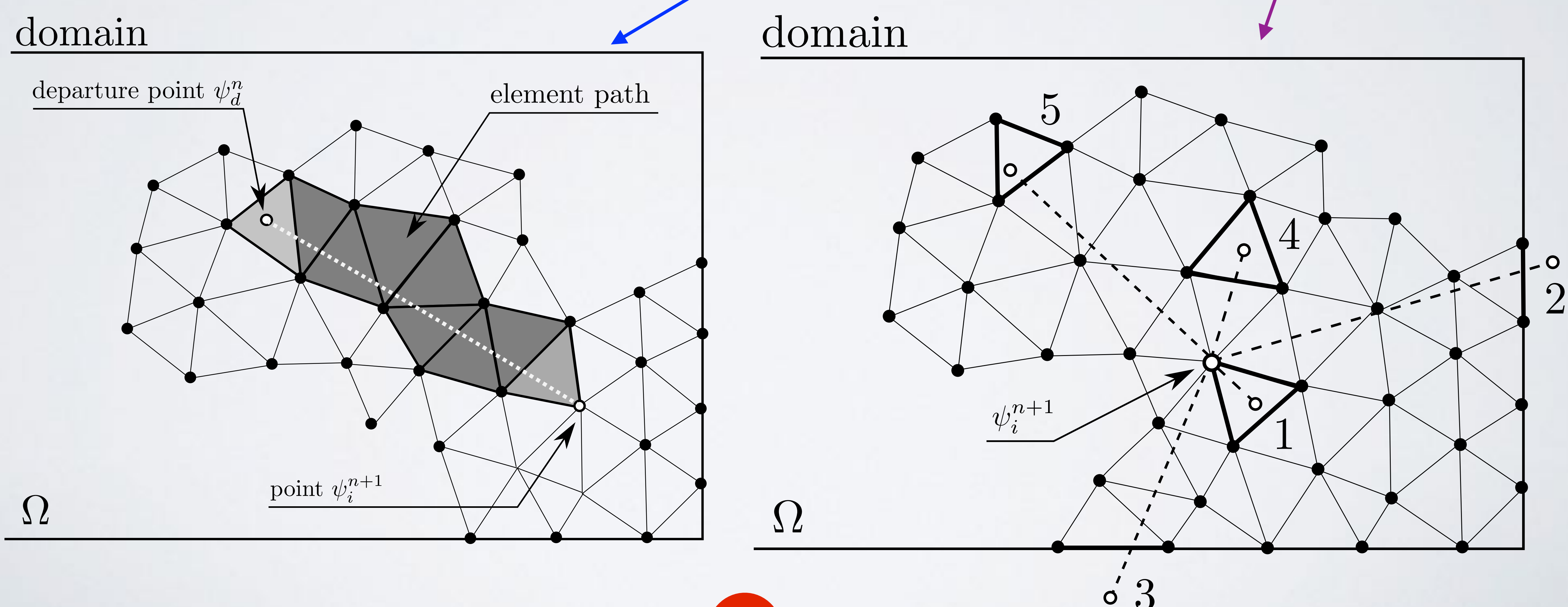
- pressure, diffusive terms → Galerkin method
- convective terms → ALE and SL methods
- time discretization → 1st. order forward difference
- linear systems → Projection method - LU
- surface tension → Geometric

EULERIAN - LAGRANGIAN



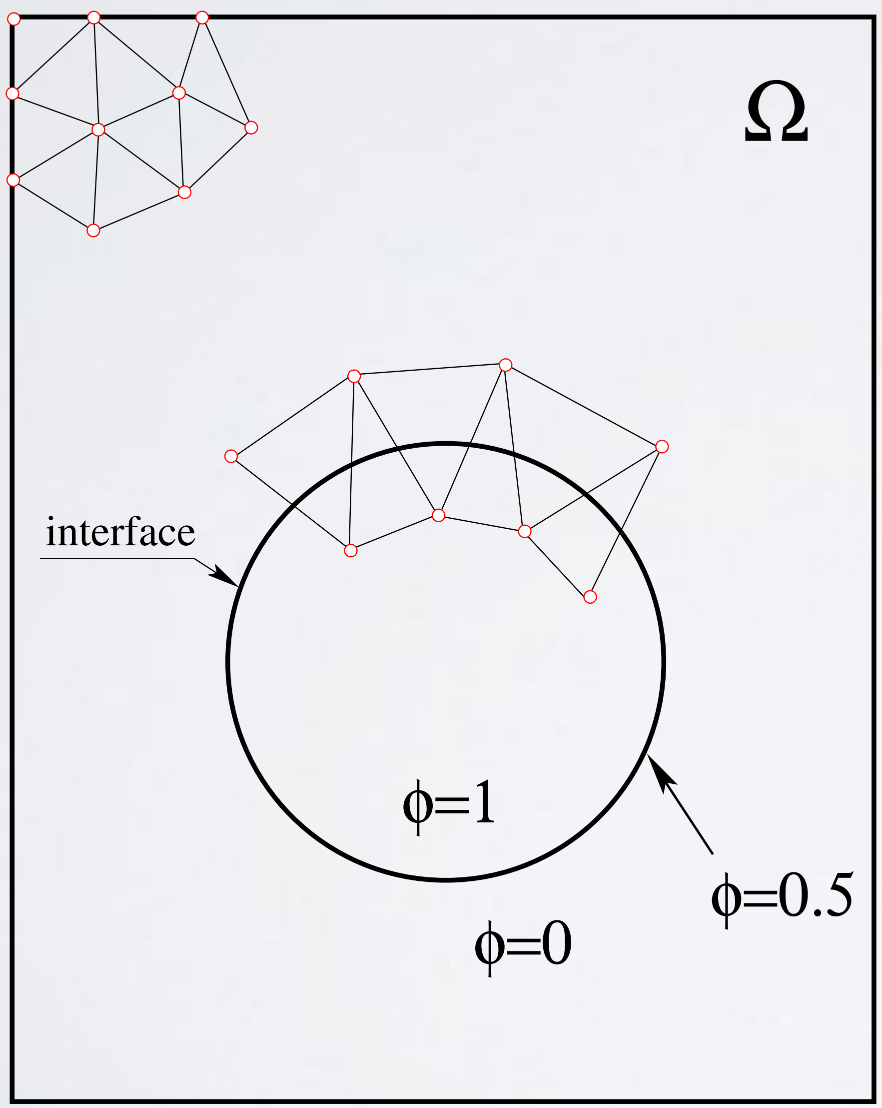
SEMI-LAGRANGIAN

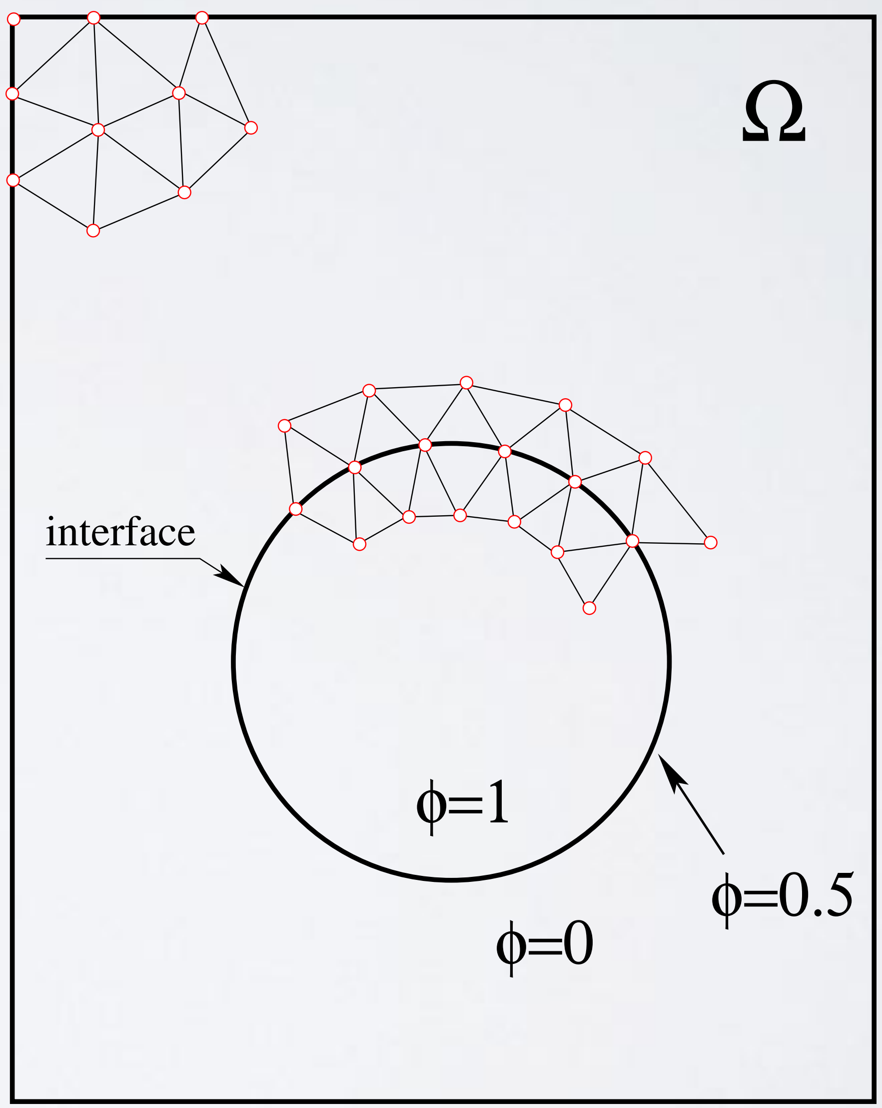
- compute advection using fixed mesh;
- discretization of material derivative: $\dfrac{D\psi}{Dt} = \dfrac{\psi^{n+1} - \psi^{n}}{\Delta t}$
- in ALE context: compute difference between flow field and node motion;
- 2-steps calculation: find departure point (trajectory) and interpolation.



INTERFACE DEFINITION

- Eulerian approach: (fixed mesh)
- Lagrangian approach: (moving mesh)

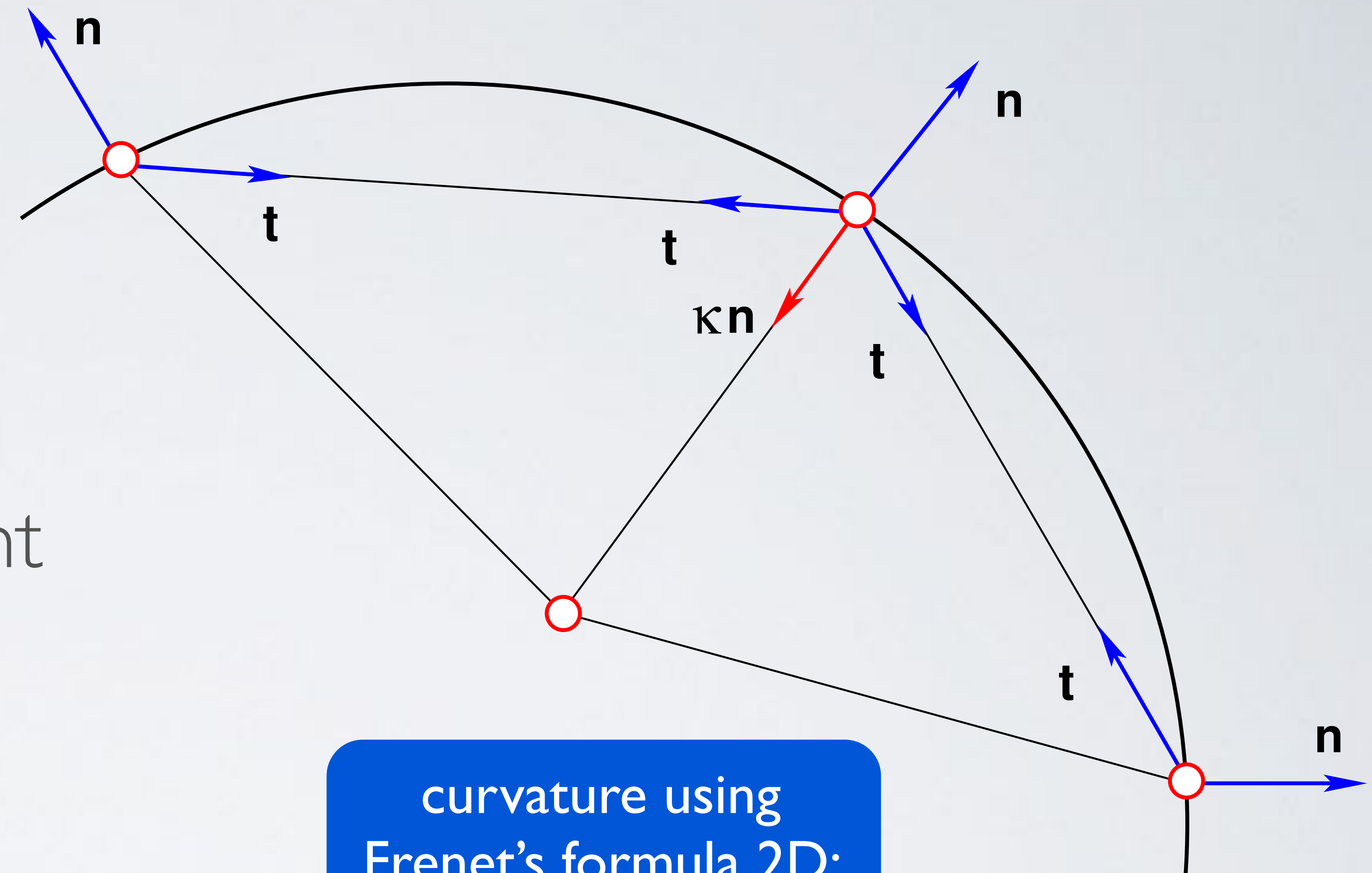




SURFACE TENSION

$$\mathbf{f} = \sigma\kappa\mathbf{n}\delta$$

continuum surface force model
Brackbill and Kothe (1992)



σ : surface tension coefficient

κ : curvature

δ : Dirac delta function

$\mathbf{n}$: normal vector

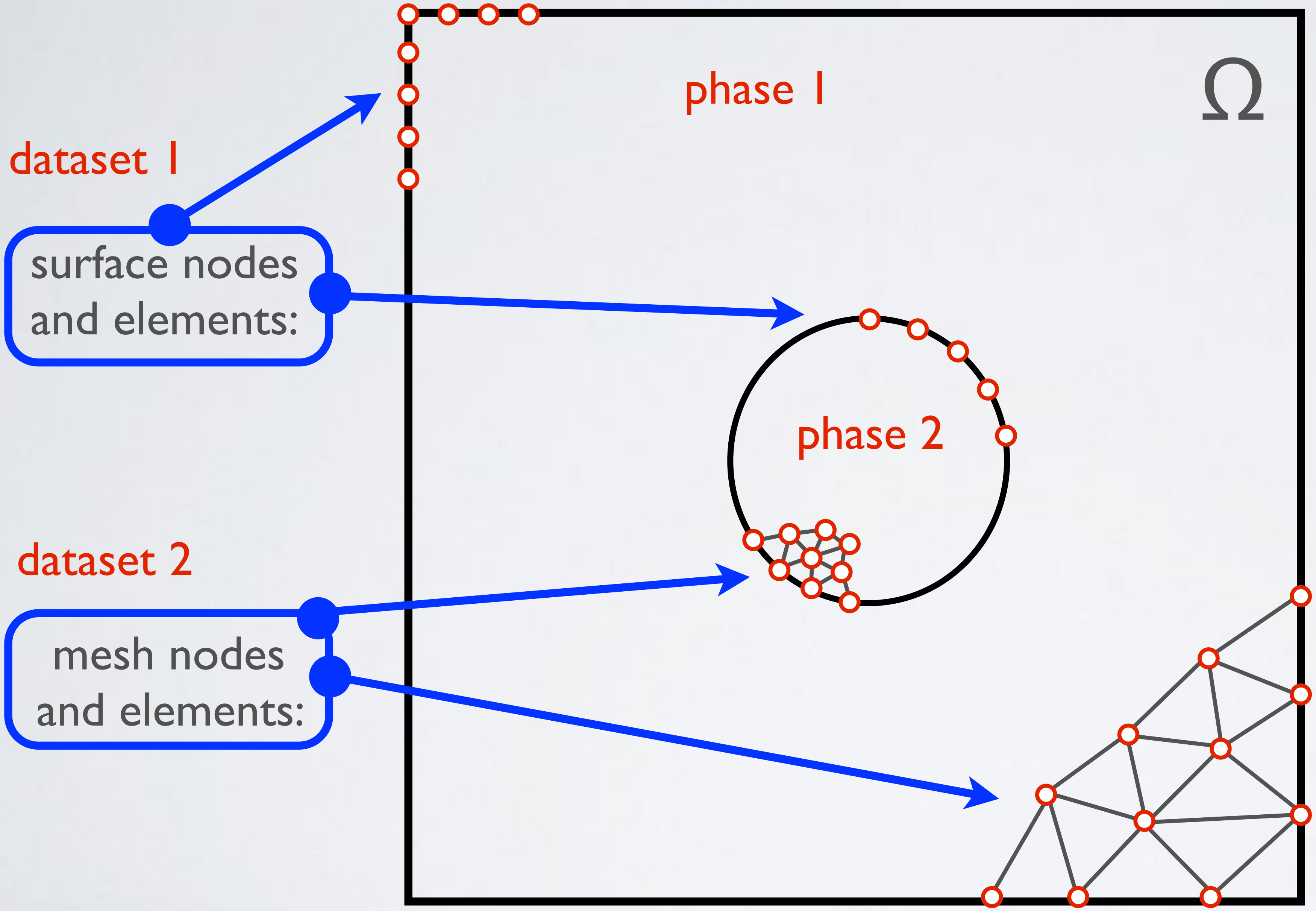
$\mathbf{t}$: tangent vector

ϕ : marker function (Heaviside)

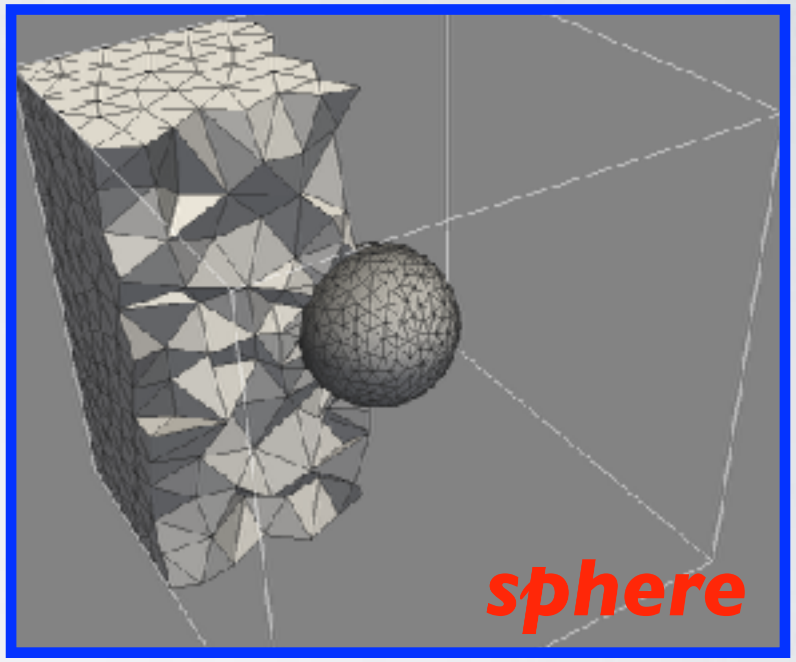
curvature using Frenet's formula 2D:

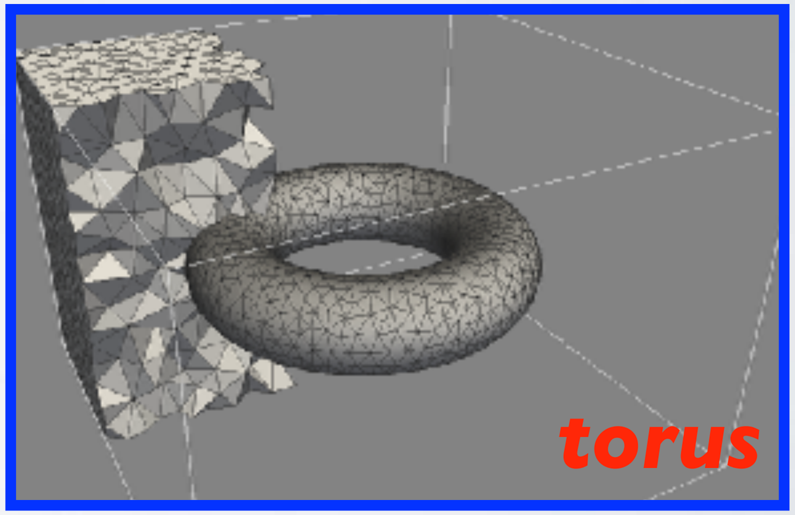
$$\kappa_i = \frac{\left| \sum_{j=1}^{2} \frac{t_j}{|t_j|} \right|}{\sum_{j=1}^{2} \frac{|d_j - d_i|}{2}}$$

MESH STORAGE

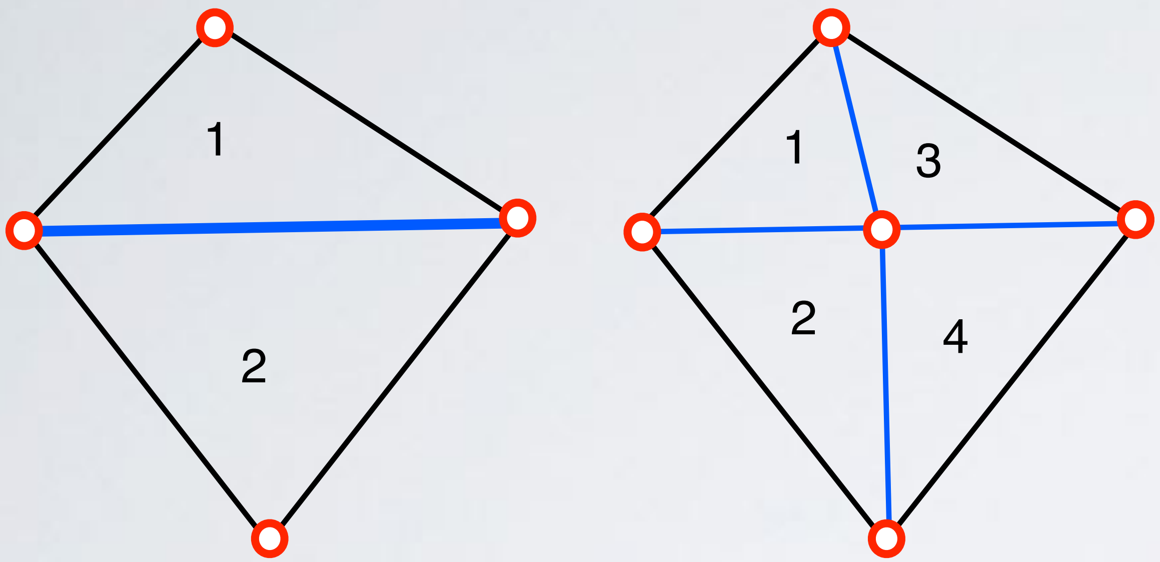


Examples:

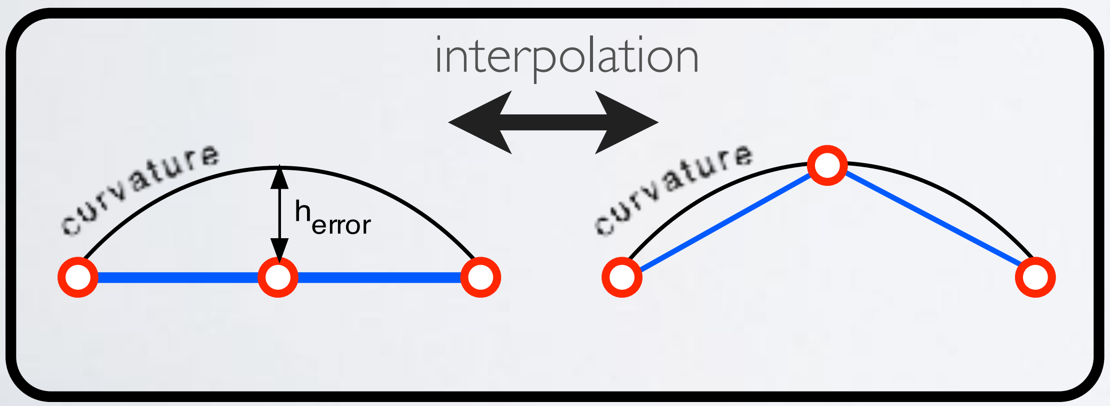




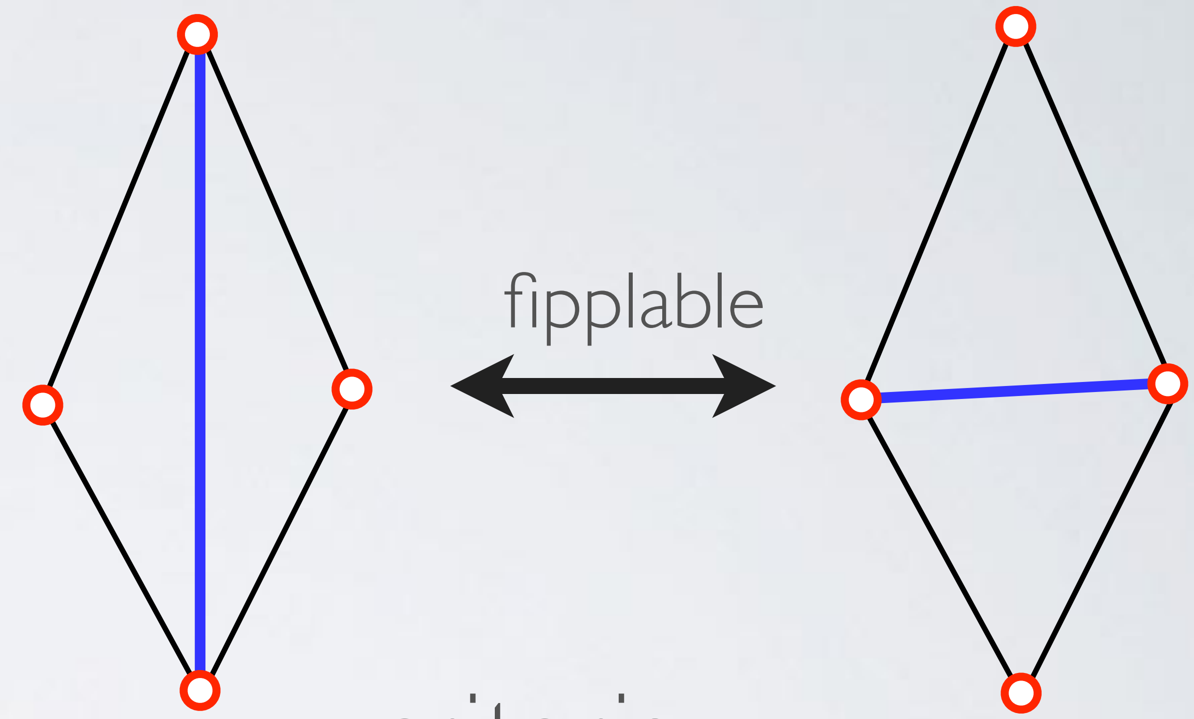
INSERTION



2D view:



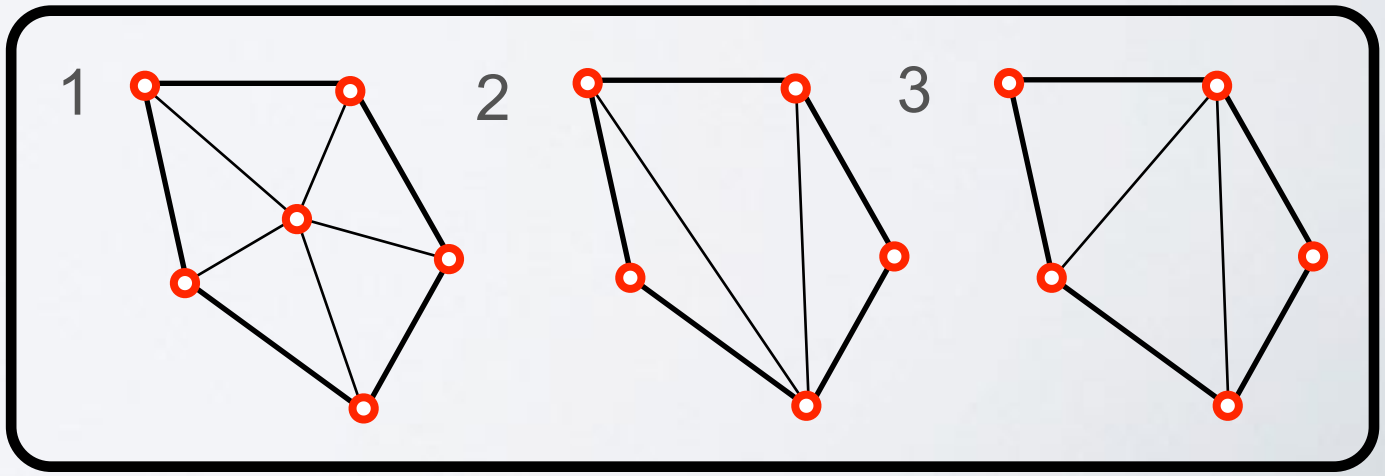
FLIPPING



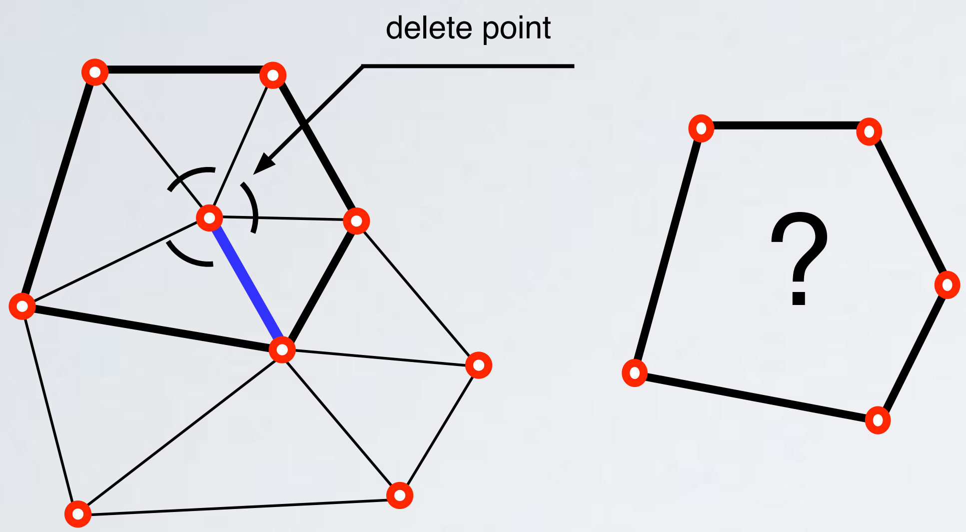
criteria:

aspect ratios, curvature, area and circumcenter

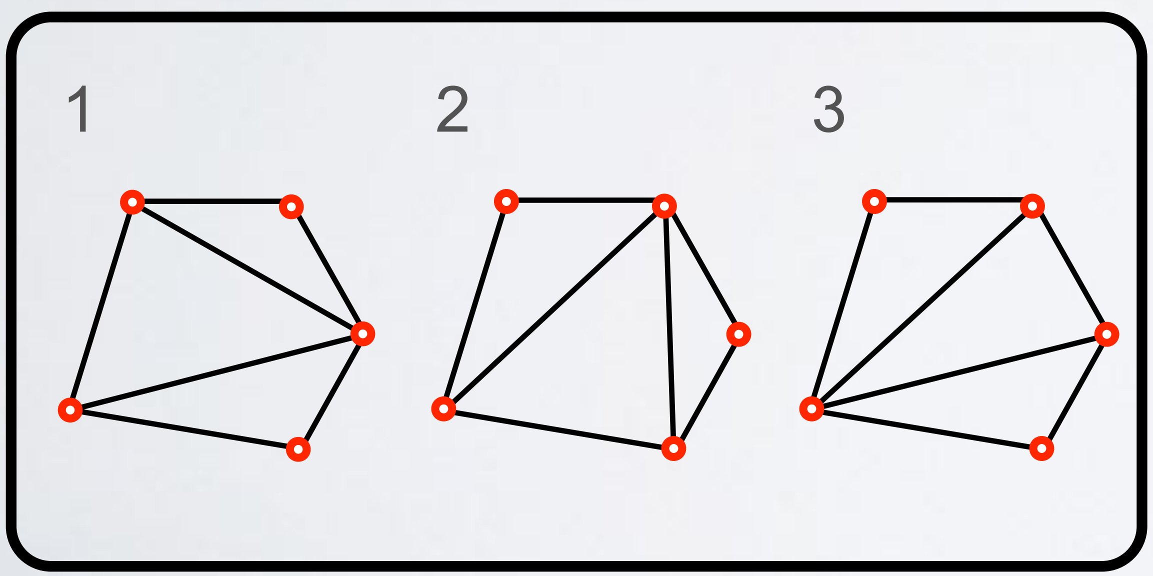
example:



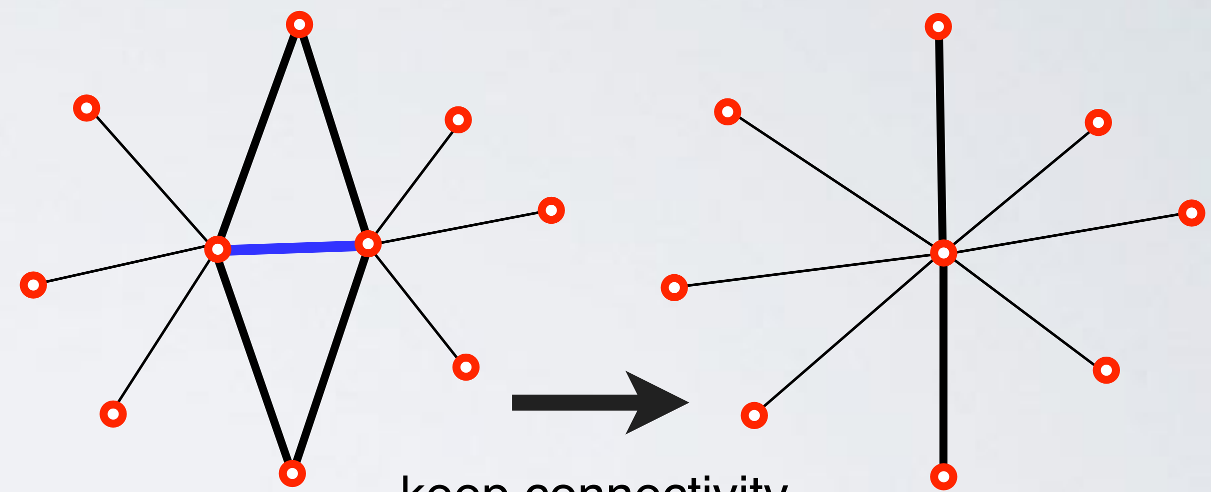
DELETION



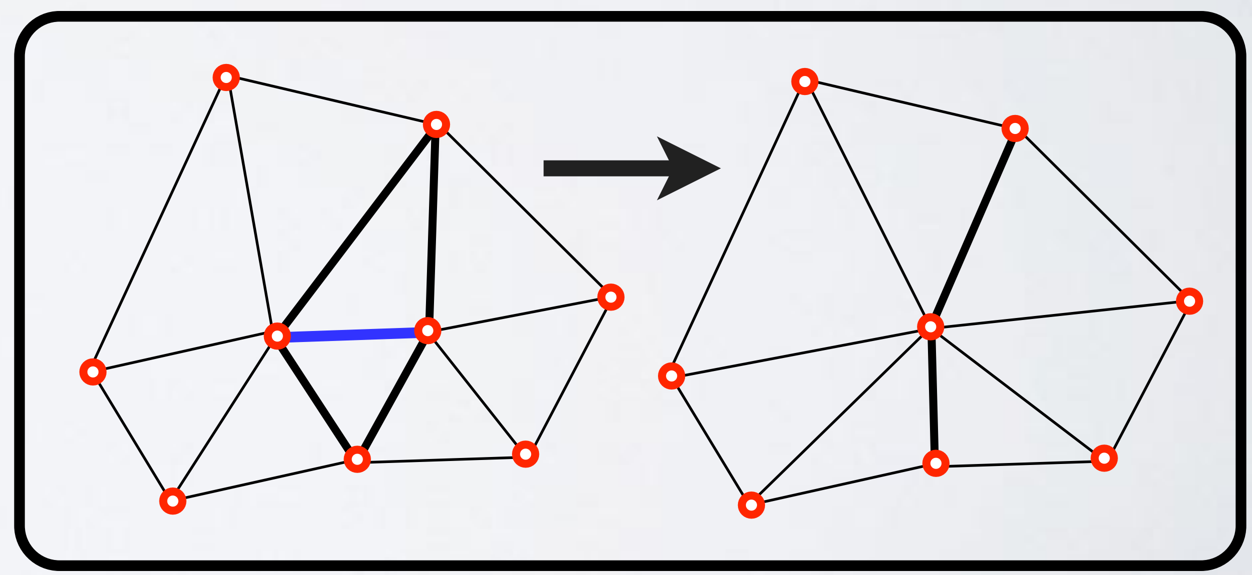
options:



CONTRACTION



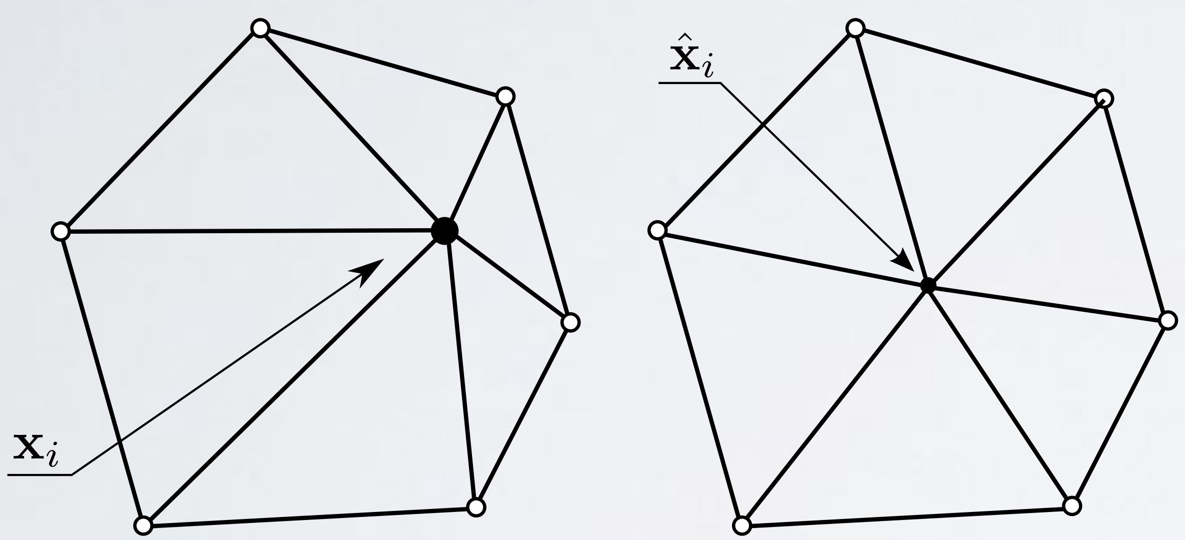
example:



NODES MOTION

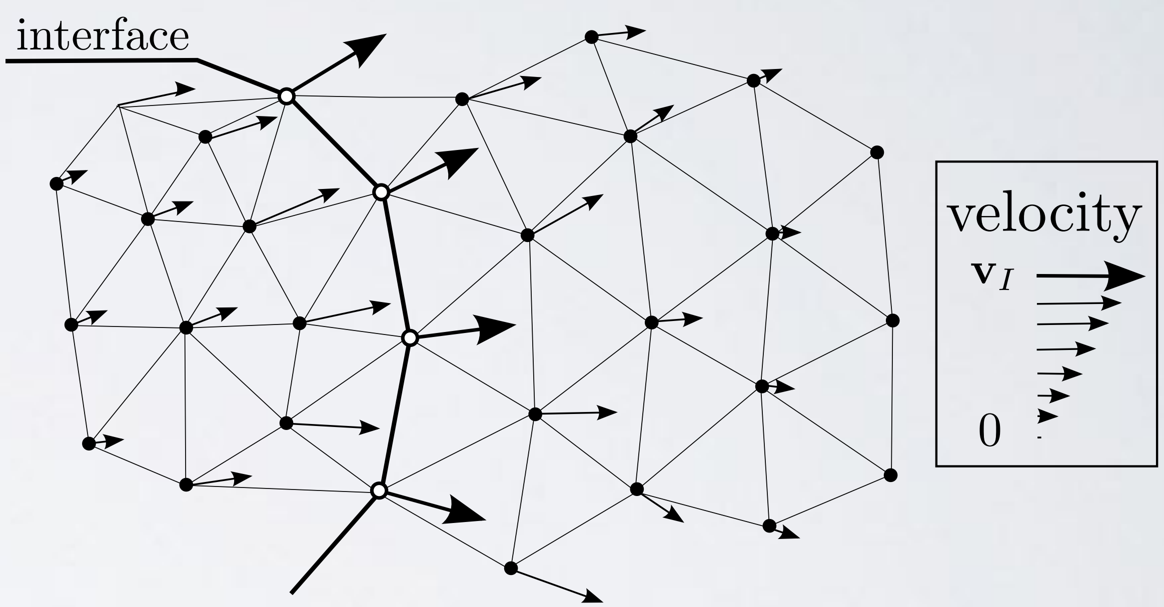
coordinates:

$$\hat{\mathbf{v}}_{e_i} = \frac{\sum_{i \in N_1(j)} e_{ij}^{-1}(\mathbf{x}_j - \mathbf{x}_i)}{dt}$$



velocities:

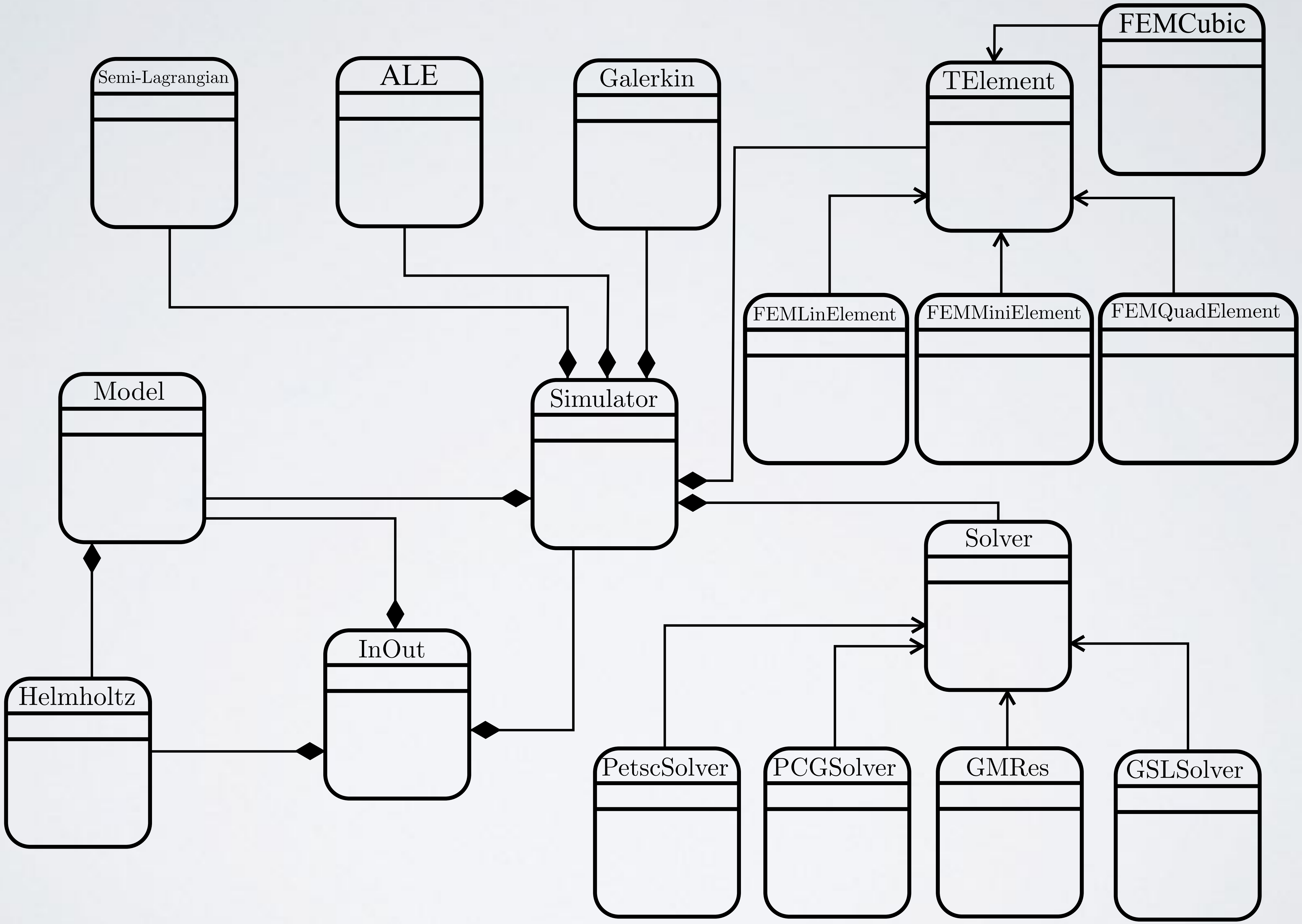
$$\hat{\mathbf{v}}_{v_i} = \frac{1}{n} \sum_{j \in N_1(j)} \mathbf{v}_j$$



Proposed scheme:

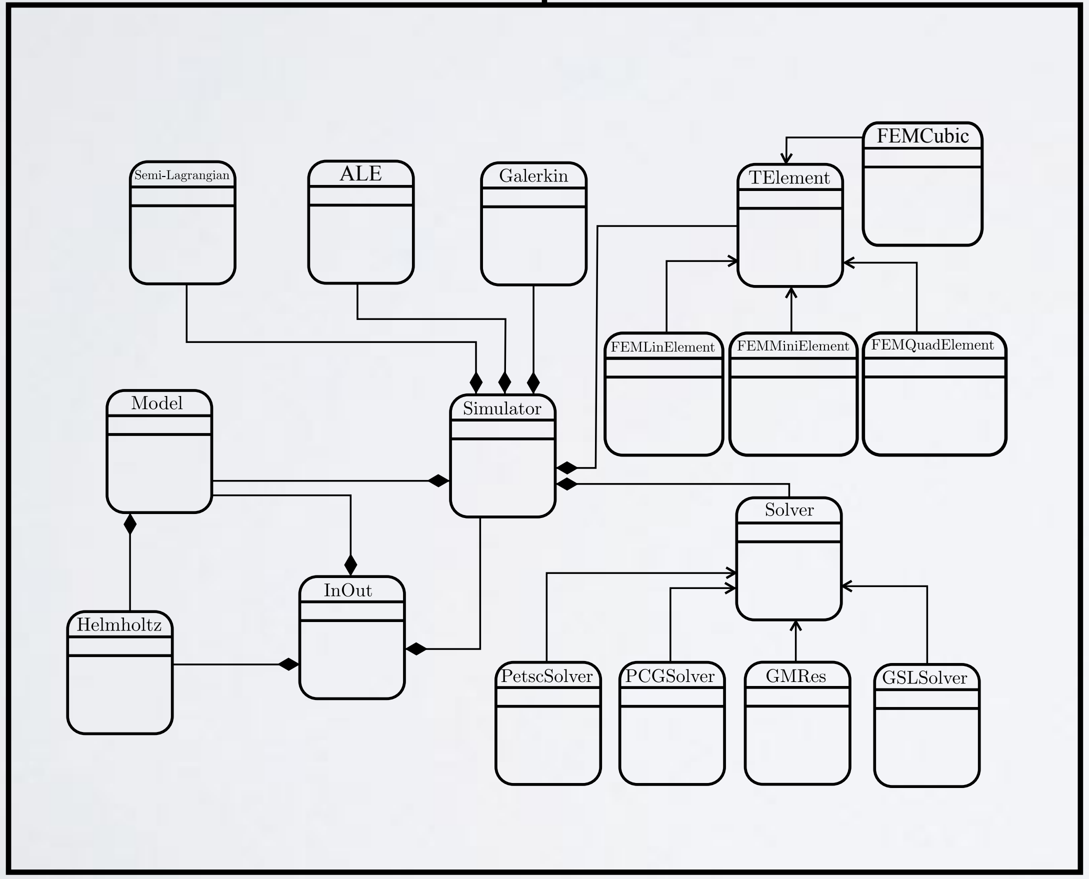
$$\hat{\mathbf{v}}(\mathbf{x}) = \begin{cases} c_1\mathbf{v} + c_2\mathbf{v_v} + c_3\mathbf{v_e} & \text{if } \mathbf{x} \text{ does not belong to the interface} \\ \mathbf{v} - d_1(\mathbf{v}\cdot\mathbf{t})\mathbf{t} + d_2(\mathbf{v_e}\cdot\mathbf{t})\mathbf{t} & \text{if } \mathbf{x} \text{ belongs to the interface} \end{cases}$$

SIMPLIFIED UML DIAGRAM



PYTHON API

Python2.7 API



- step.py
- vortex.py
- stepALE.py
- poiseuilleAxi.py

- risingBubble.py
- risingBubbleTaylor.py
- sessileDrop.py
- oscillatingDrop.py
- micro.py
- microAxi.py
- risingBubbleAxi.py
- risingBubbleTaylorAxi.py

SCRIPT STRUCTURE

Python2.7 API

- simulation parameters
- mesh velocity
- linear system solver

- python procedures for copying and saving files

- Model constructor
- Simulator initialization

- for loop of ODE (time)

- remeshing
- interpolation new mesh

```
• stepALE()                    // convective term
• setDt()                      // set dt based on cfl
• printSimulationReport()
• movePoints()                 // move nodes
• setRHS()                     // set right hand side vector
• setUncoupledBC()             // apply B.C.
• setGravity()                 // set gravity vector
• setInterface()               // set interface force vector
• setUncoupled()               // solve linear system
• saveMesh()                   // save mesh in Gmsh format
• saveVTK()                    // save VTK for preview
• saveSol()                    // save solution in binary
• saveOldData()                // save solution vectors
• timeStep()                   // iterate in time
```

SOLUTION PROCEDURE

- Departure parameters $(\mathbf{v}_h^n, p_h^n)$ on mesh $\mathbf{x}_h^n$

1. Compute mesh velocities $\hat{\mathbf{v}}_h(\mathbf{x})$
2. Calculate time step Δt based on defined CFL
3. Move mesh points to new position: $\mathbf{x}_h^{n+1} = \mathbf{x}_h^n + \Delta t \hat{\mathbf{v}}_{\mathbf{h}}(\mathbf{x})$
4. Solve ALE N-S linear system for $(\mathbf{v}_h^{n+1}, p_h^{n+1})$
5. Perform mesh operations (deletions, insertions, flippings etc.)
6. Generate new mesh: $\mathbf{x}_h^{n+1} \longrightarrow \mathbf{x}_{\tilde{h}}^{n+1}$
7. Interpolate solution to the new mesh: $(\mathbf{v}_h^{n+1}, p_h^{n+1}) \longrightarrow (\mathbf{v}_{\tilde{h}}^{n+1}, p_{\tilde{h}}^{n+1})$

HOW TO RUN

Apple OS X | Linux SUSE

check 1st test case

Apple OS X

- open terminal
- cd $HOME/projects/python/femSIM2d
- python2.7 step.py

Linux SUSE

- open terminal
- cd $HOME/projects/python/femSIM2d
- python step.py

generate mesh

Apple OS X

- cd $HOME/projects/db/gmsh/2d
- chose one test case: Ex. rising
- gmsh -1 airWaterSugar.geo

Linux SUSE

- cd $HOME/projects/db/gmsh/2d
- chose one test case: Ex. rising
- gmsh -1 airWaterSugar.geo

visualize geometry

Apple OS X

- cd $HOME/projects/db/gmsh/2d
- chose one test case: Ex. axi
- gmsh circular.geo

Linux SUSE

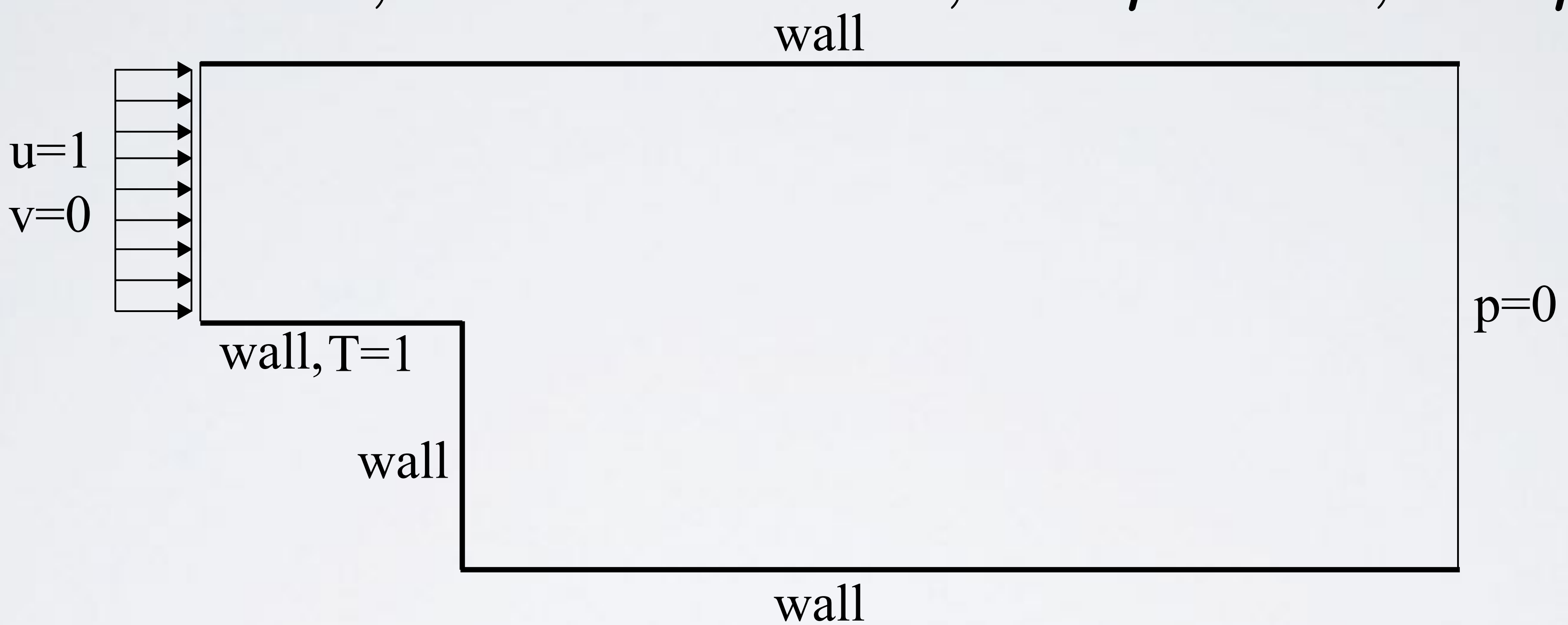
- cd $HOME/projects/db/gmsh/2d
- chose one test case: Ex. axi
- gmsh circular.geo

2D TESTCASES

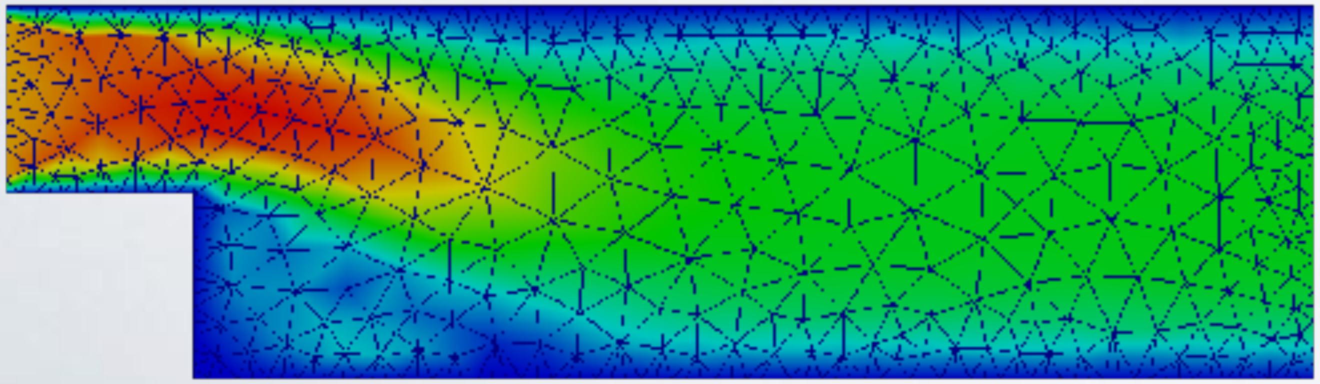
Backward facing step

stepALE.py

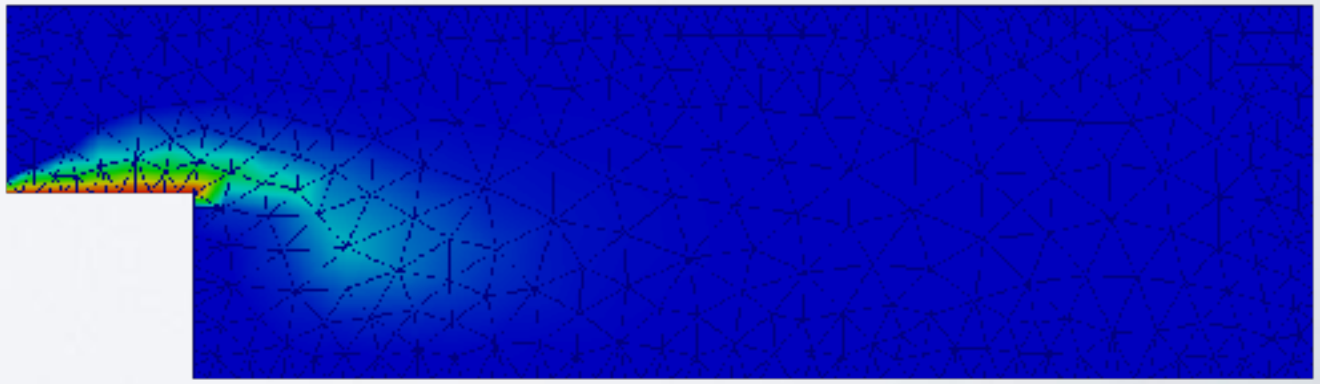
$$Re = 1000, \qquad Sc = 200, \qquad \mu = 1, \qquad \rho = 1$$



time = 37, velocity field



time = 37, temperature field

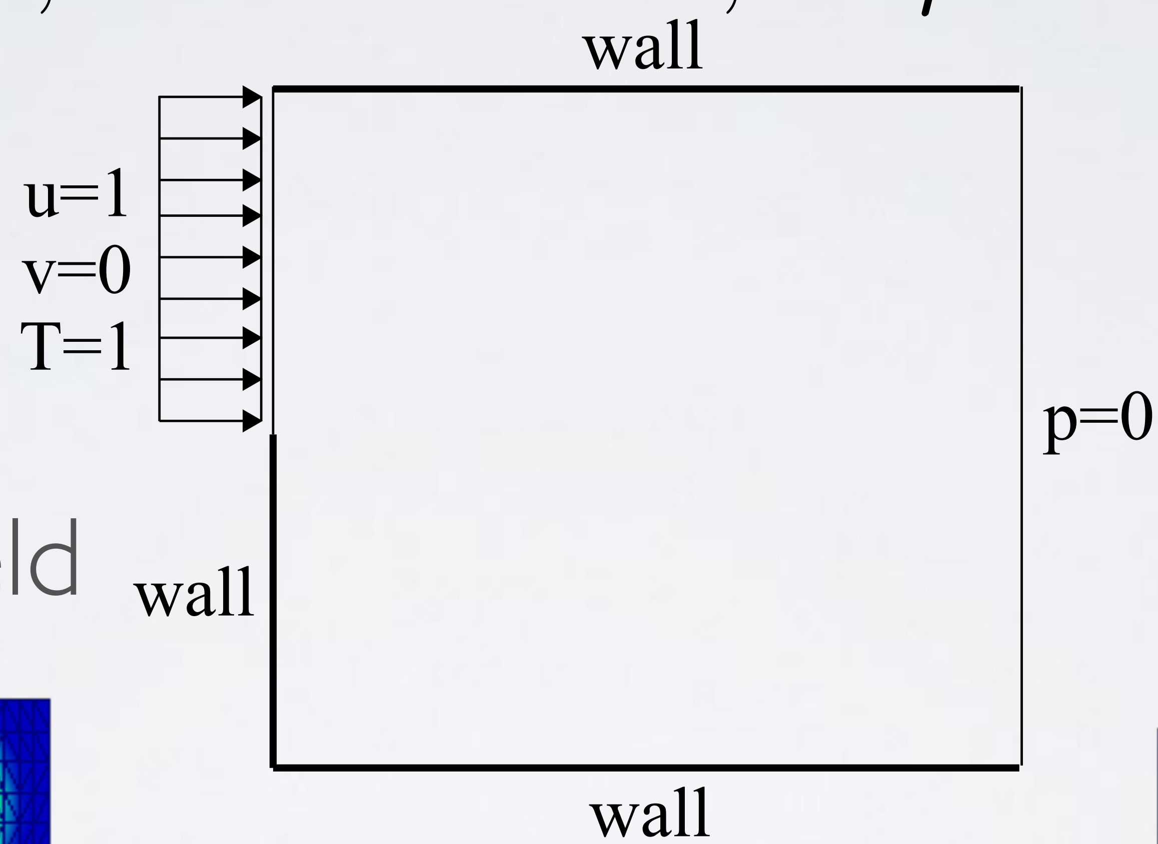


2D TESTCASES

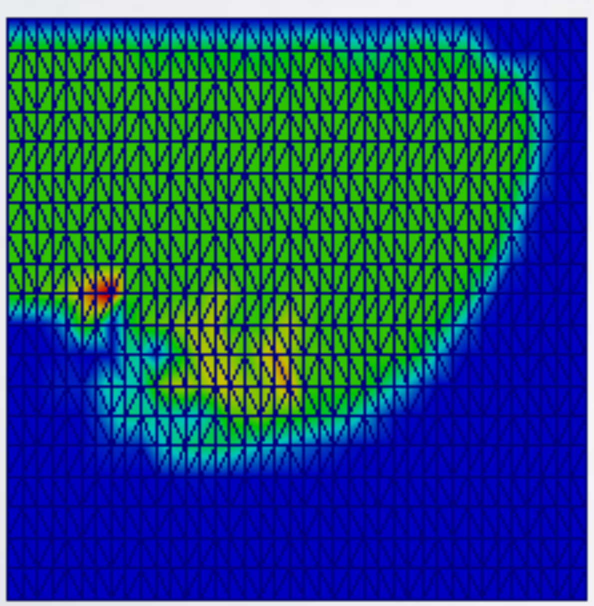
Backward facing step

step.py

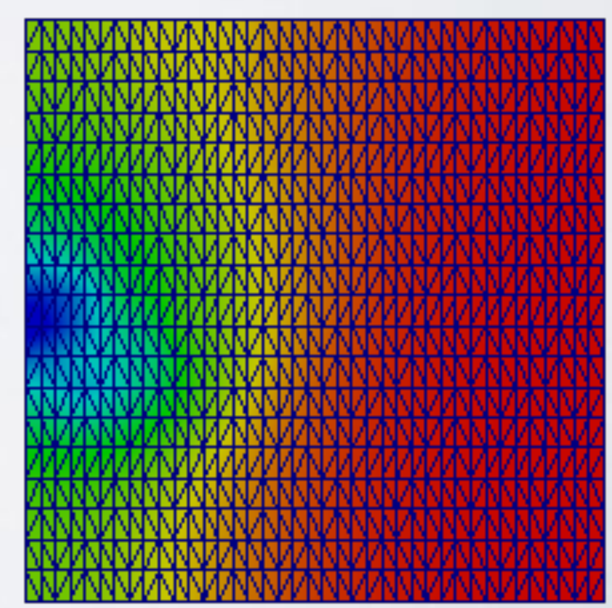
$$Re = 10000, \quad Sc = 2000, \quad \mu = 1, \quad \rho = 1$$



time = 20,
temperature field



time step = 5,
pressure field



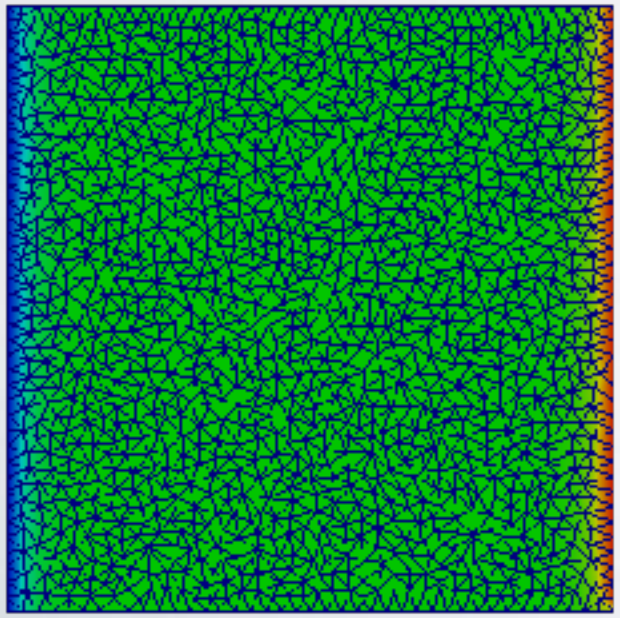
2D TESTCASES

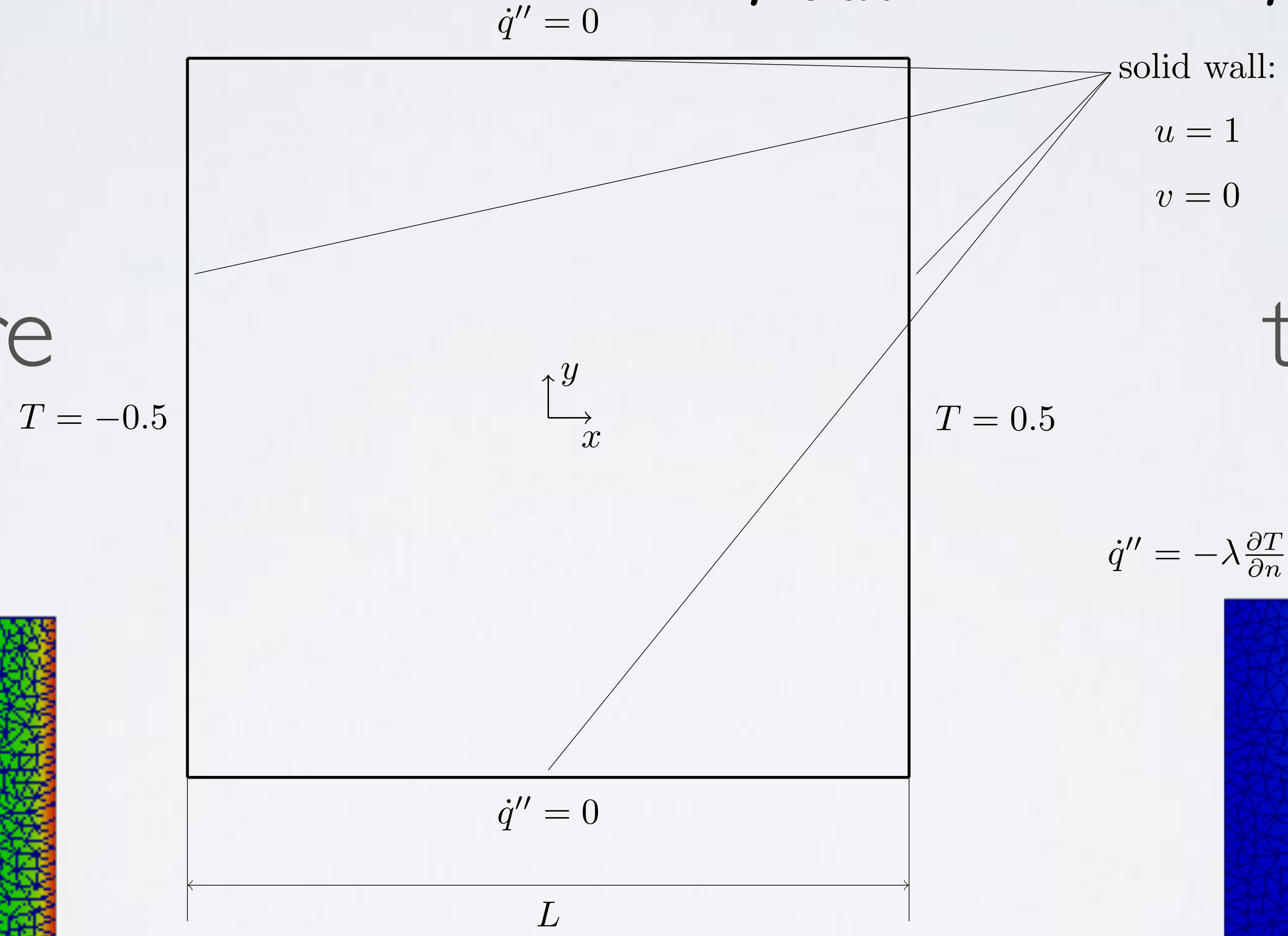
cavity with different temperature

cavity.py

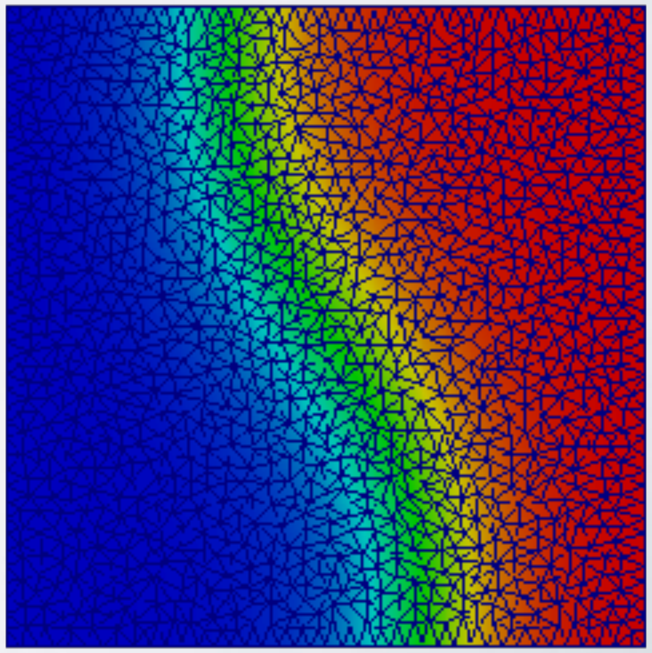
$$Re = 31.62, \quad Sc = 1, \quad \frac{\rho_{in}}{\rho_{out}} = 1, \quad \frac{\mu_{in}}{\mu_{out}} = 1$$

temperature
iter=0





temperature
iter=150



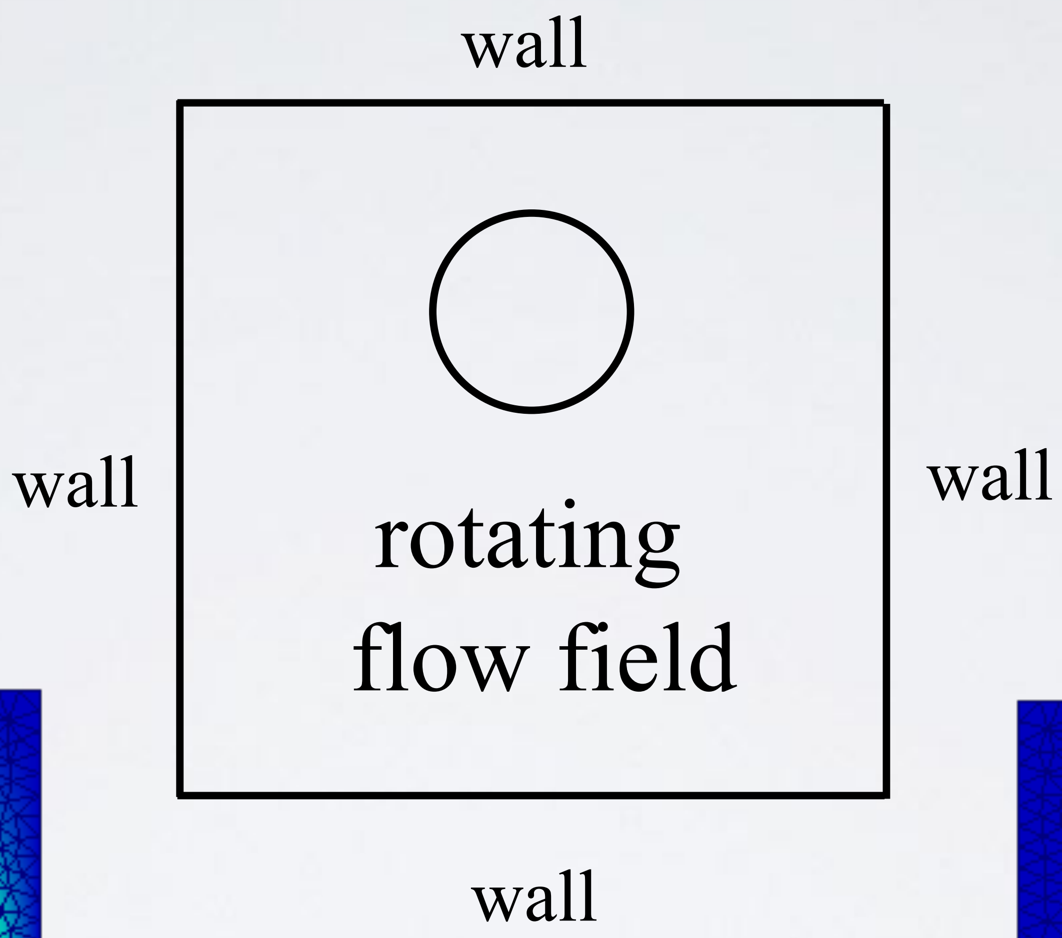
2D TESTCASES

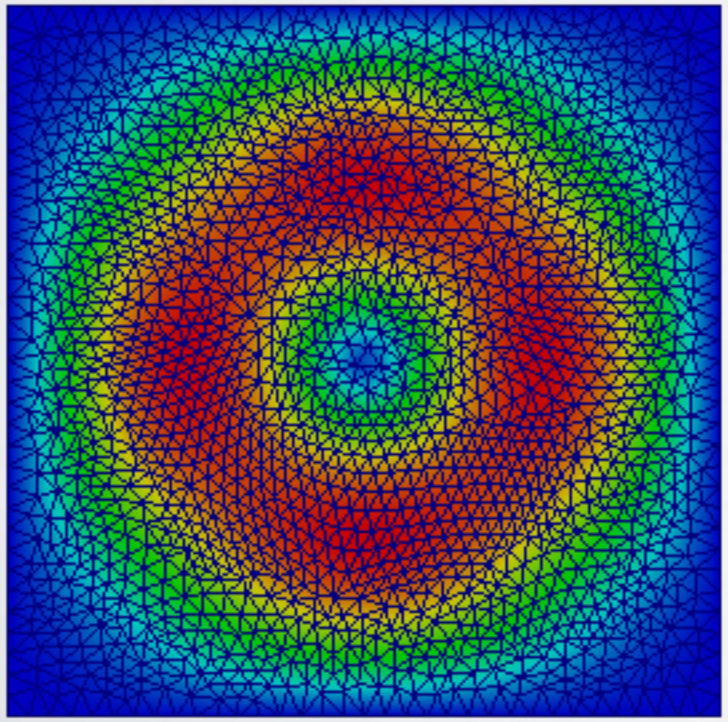
drop under rotating flow

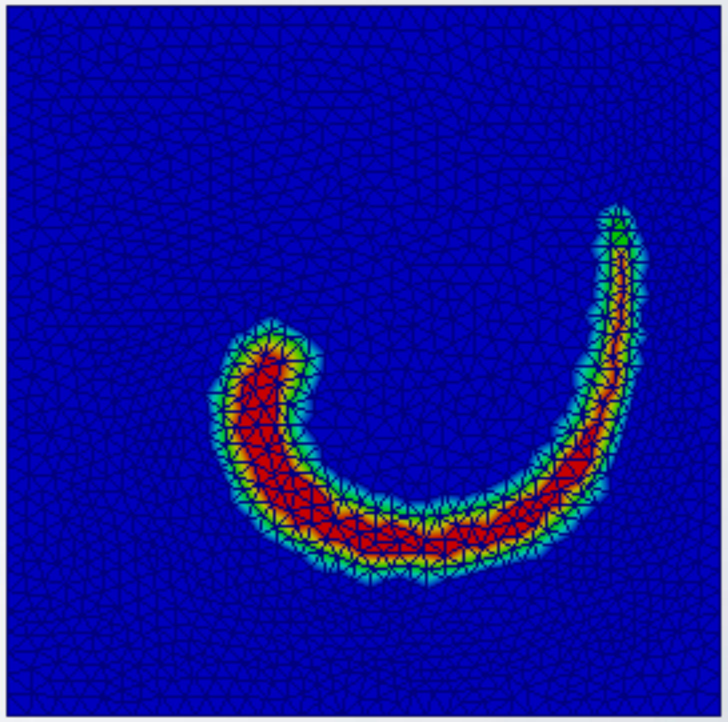
vortex.py

flow field

$$v_x = -\sin^2(\pi x)\sin(2\pi y)$$
$$v_y = \sin^2(\pi y)\sin(2\pi x)$$





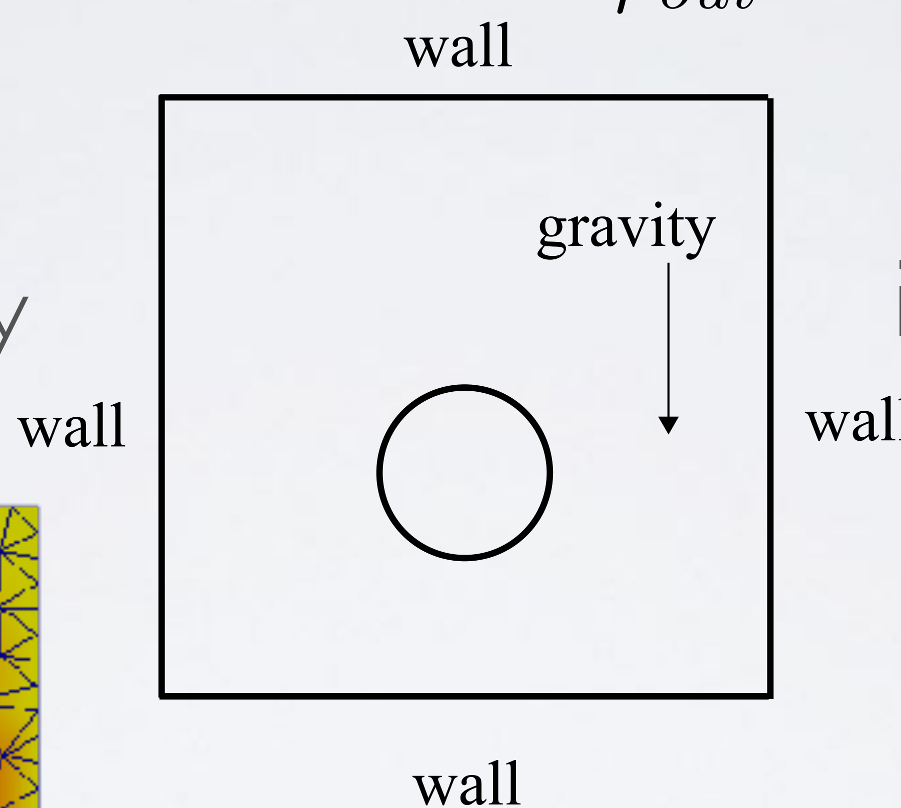


2D TESTCASES

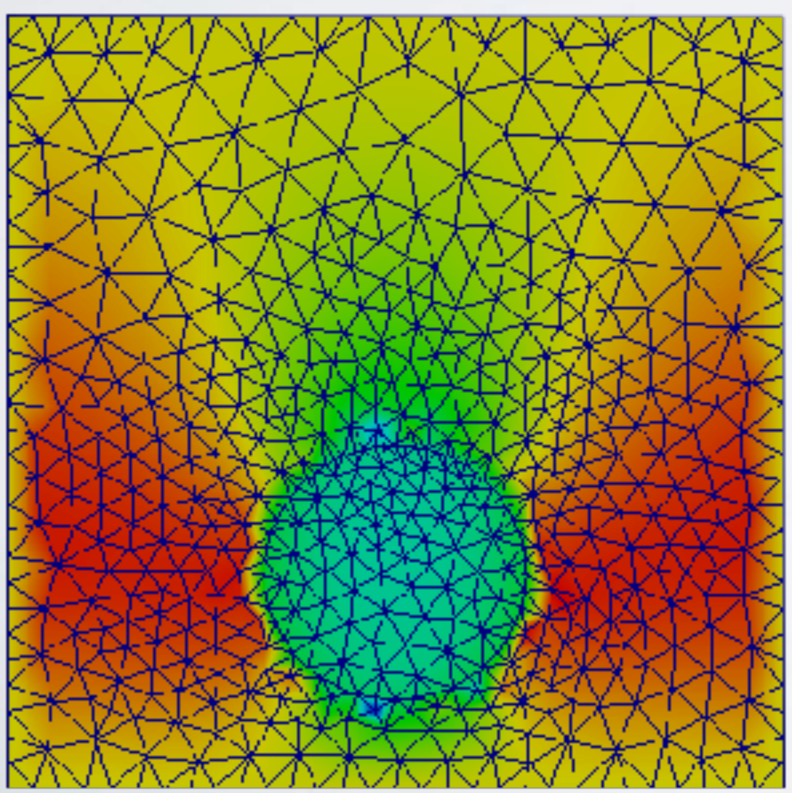
sessile drop

sessileDrop.py

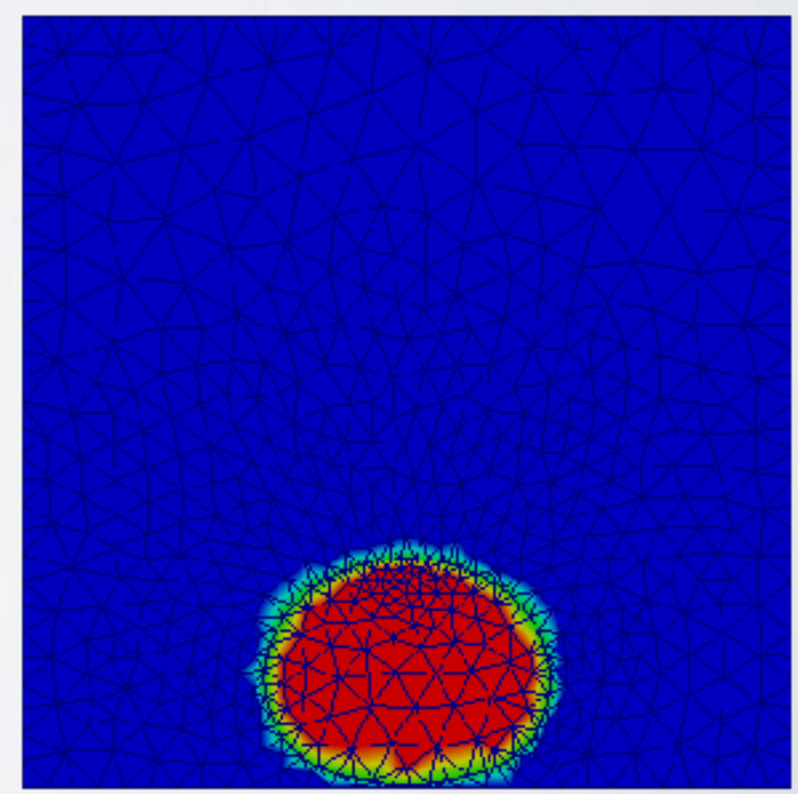
$$Re = 20, \quad We = 20, \quad Fr = 1, \quad \frac{\rho_{in}}{\rho_{out}} = 1000, \quad \frac{\mu_{in}}{\mu_{out}} = 1.111$$



iter = 10; velocity



iter = 60; heaviside

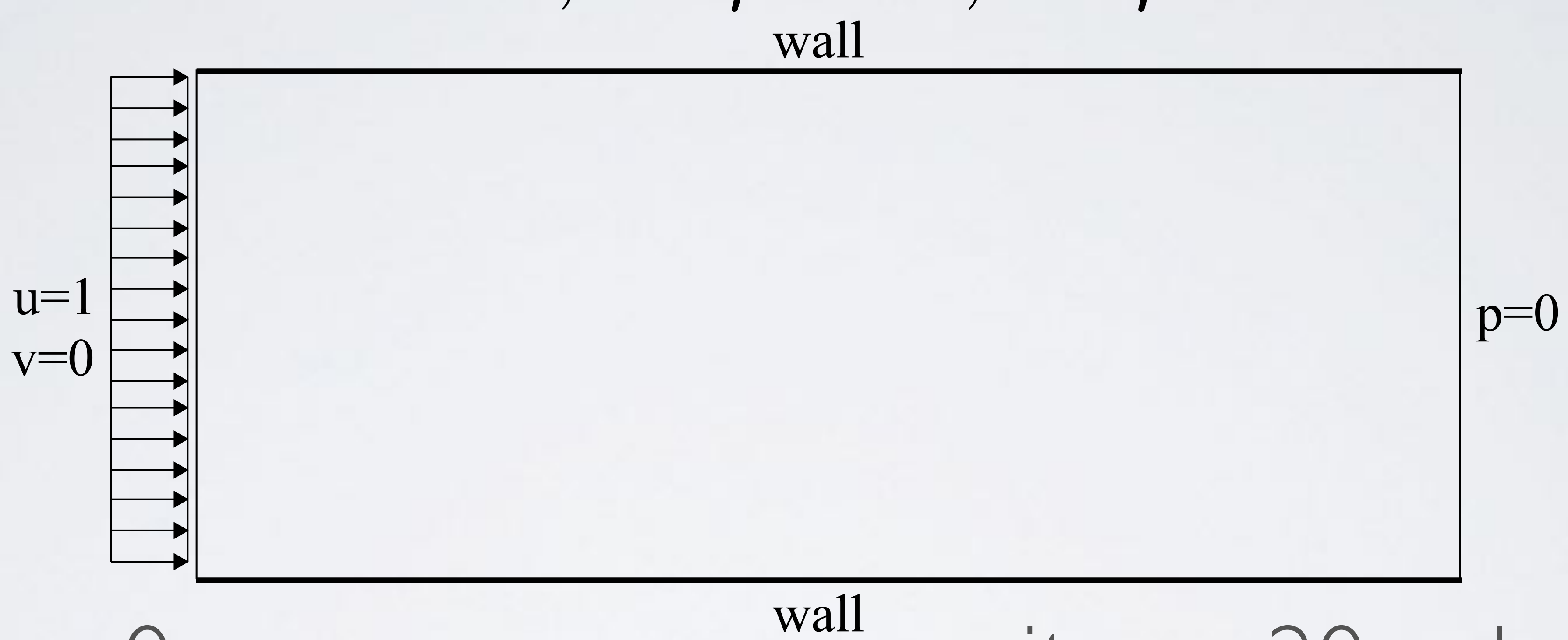


2D TESTCASES

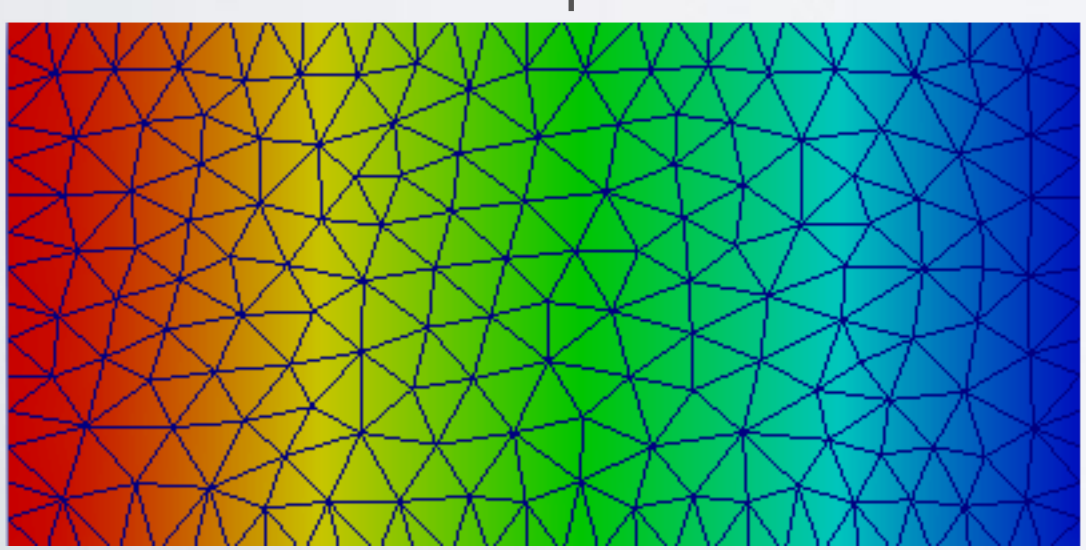
poiseuille.py

Poiseuille

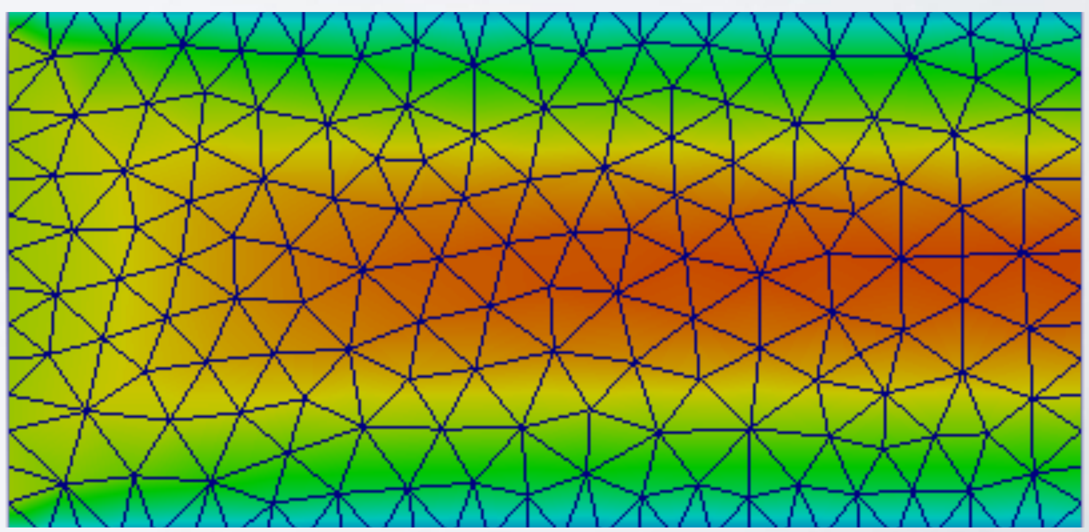
$$Re = 20, \qquad \mu = 1, \qquad \rho = 1$$



iter = 0; pressure



iter = 20, velocity



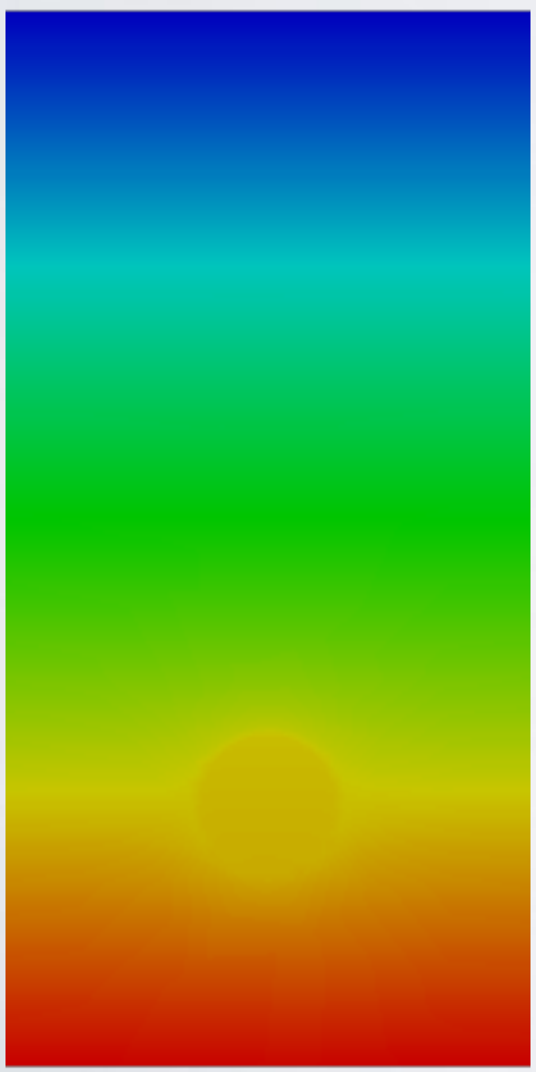
2D TESTCASES

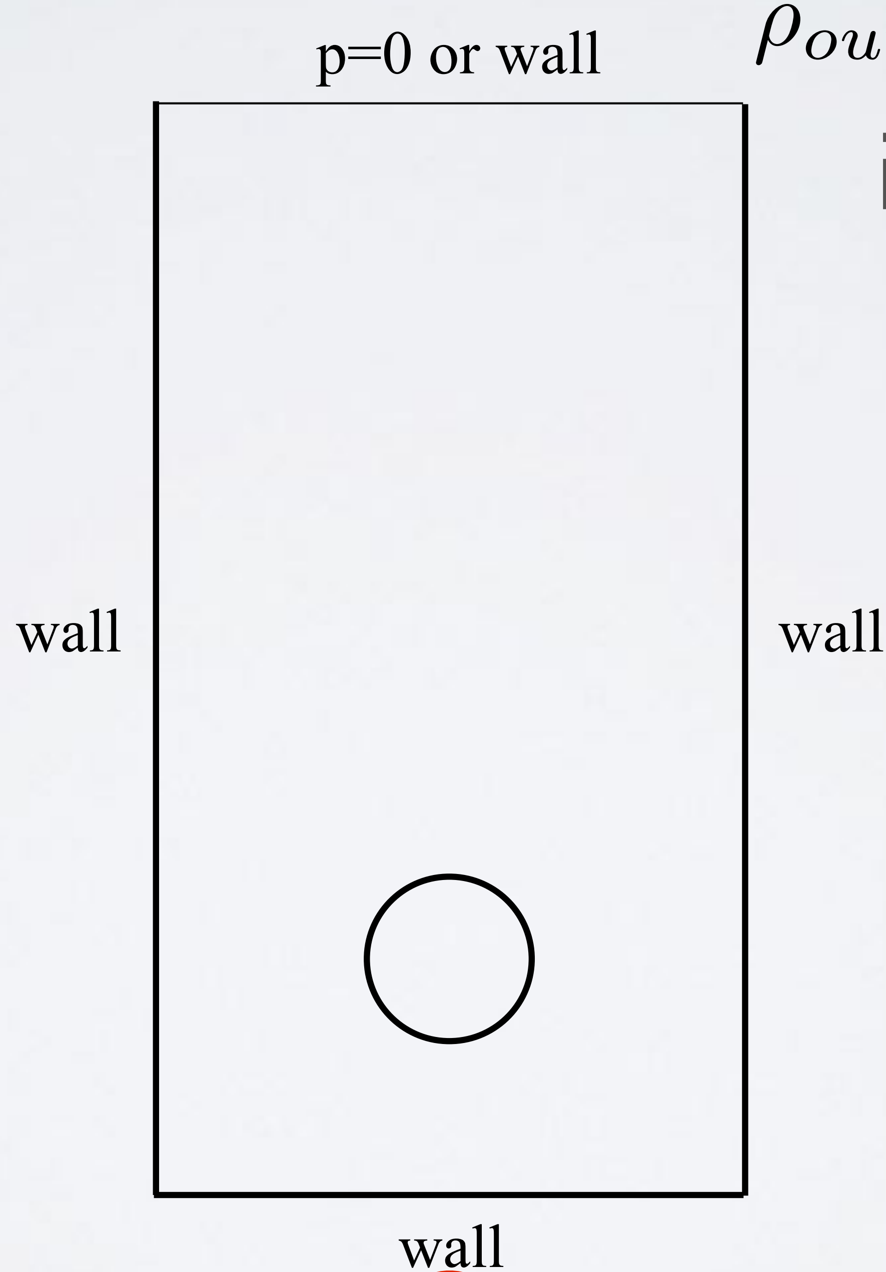
Rising bubble

risingBubble.py

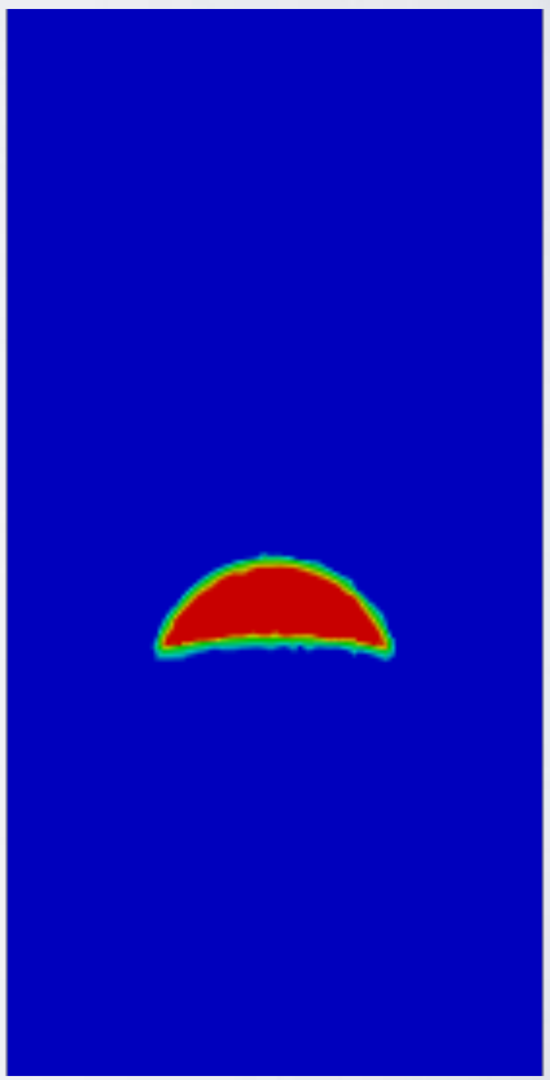
$$Re = 100, \quad We = 10, \quad Fr = 1, \quad \frac{\rho_{in}}{\rho_{out}} = 10, \quad \frac{\mu_{in}}{\mu_{out}} = 10$$

iter = 0; pressure





iter = 270; heavisde

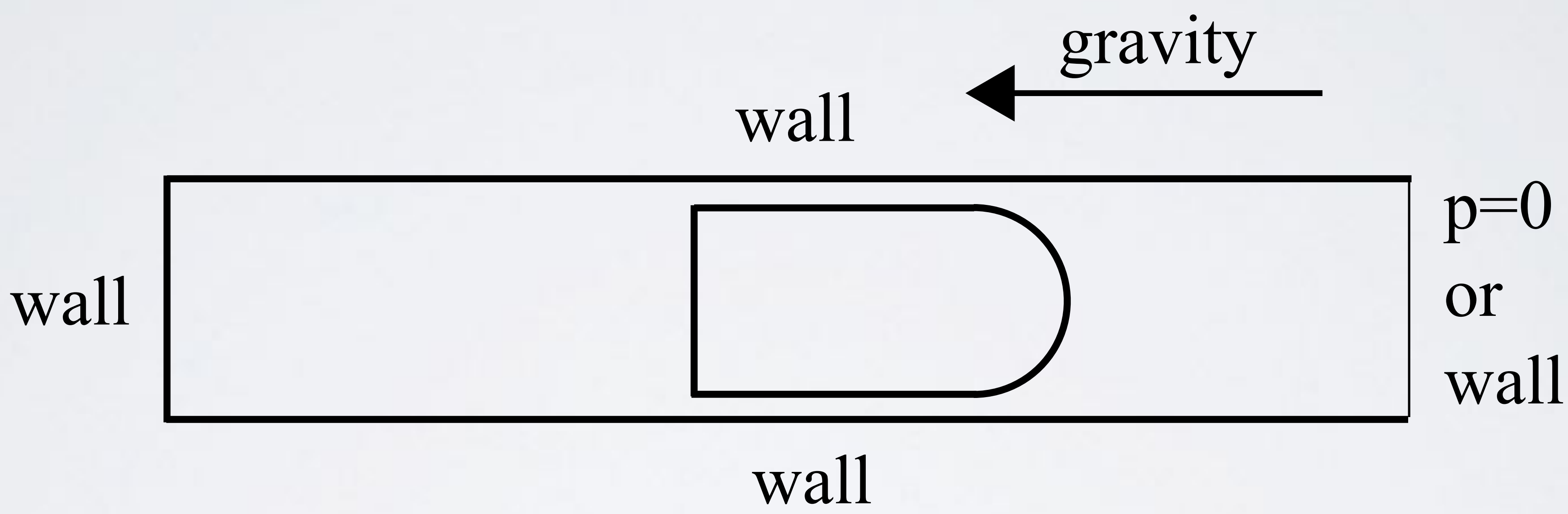


2D TESTCASES

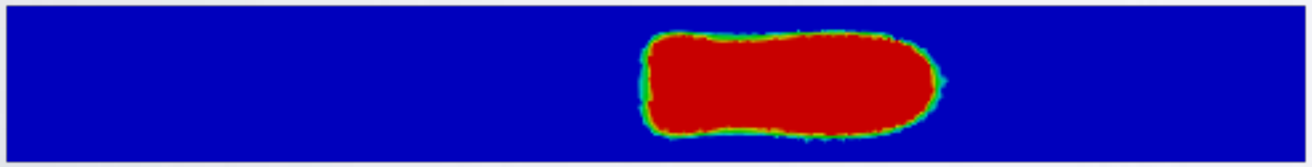
Rising of Taylor bubble - *moving frame*

risingBubbleTaylor.py

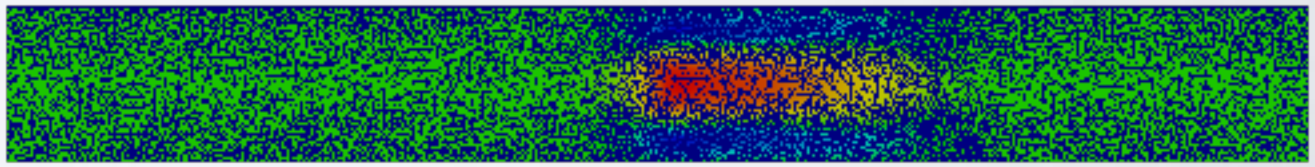
$$Re = 100, \quad We = 10, \quad Fr = 1, \quad \frac{\rho_{in}}{\rho_{out}} = 10, \quad \frac{\mu_{in}}{\mu_{out}} = 10$$



iter = 110; heaviside



iter = 110; velocity

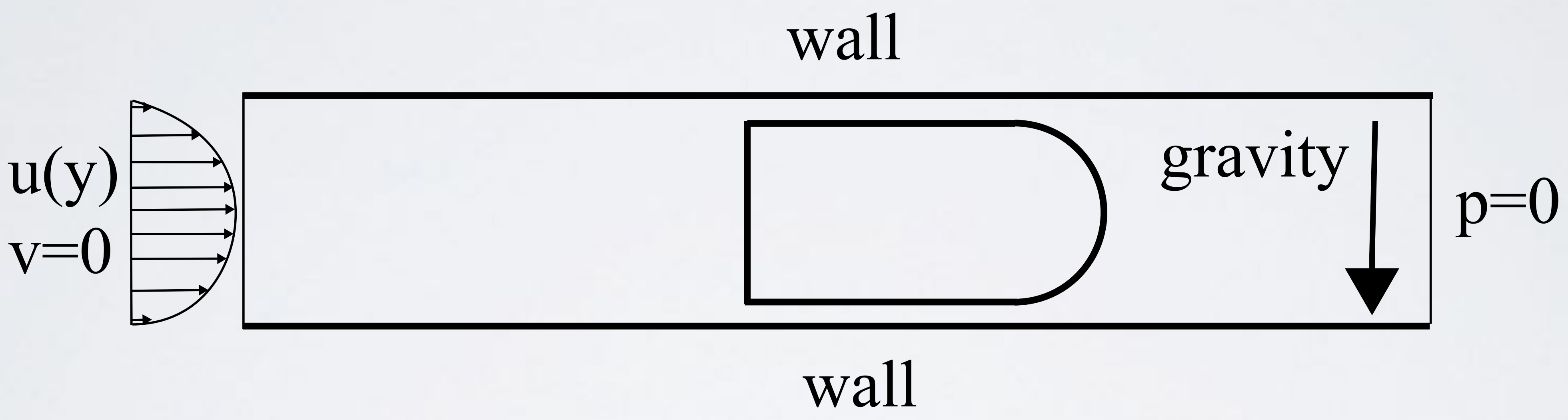


2D TESTCASES

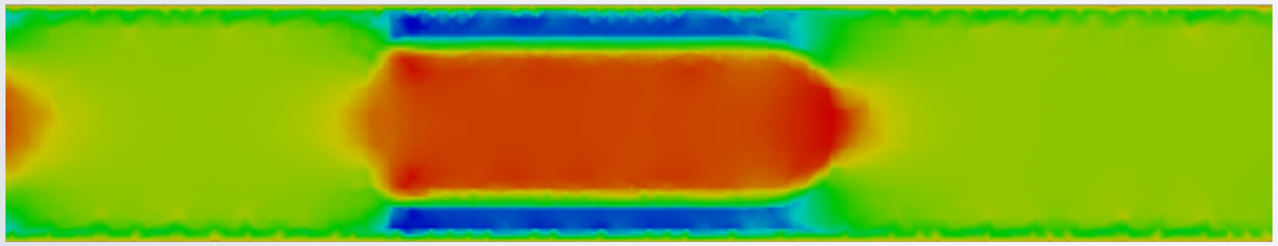
bubble in microchannel - *moving frame*

micro.py

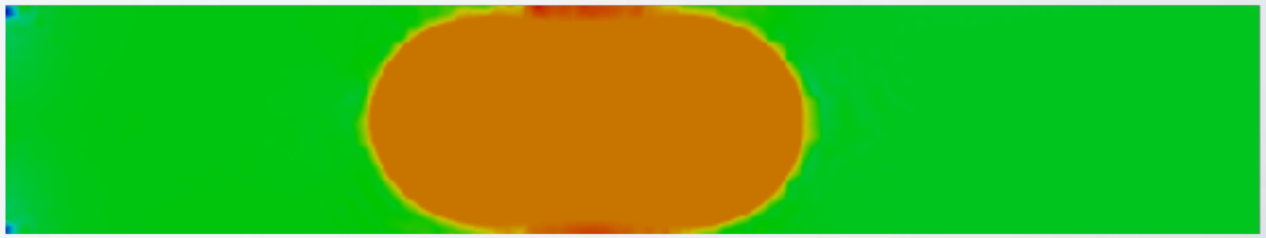
$$Re = 576.24, \quad We = 1.162, \quad Fr = 10.096, \quad \frac{\rho_{in}}{\rho_{out}} = 1509.39, \quad \frac{\mu_{in}}{\mu_{out}} = 180.169$$



iter = 05; velocity field



iter = 850; pressure

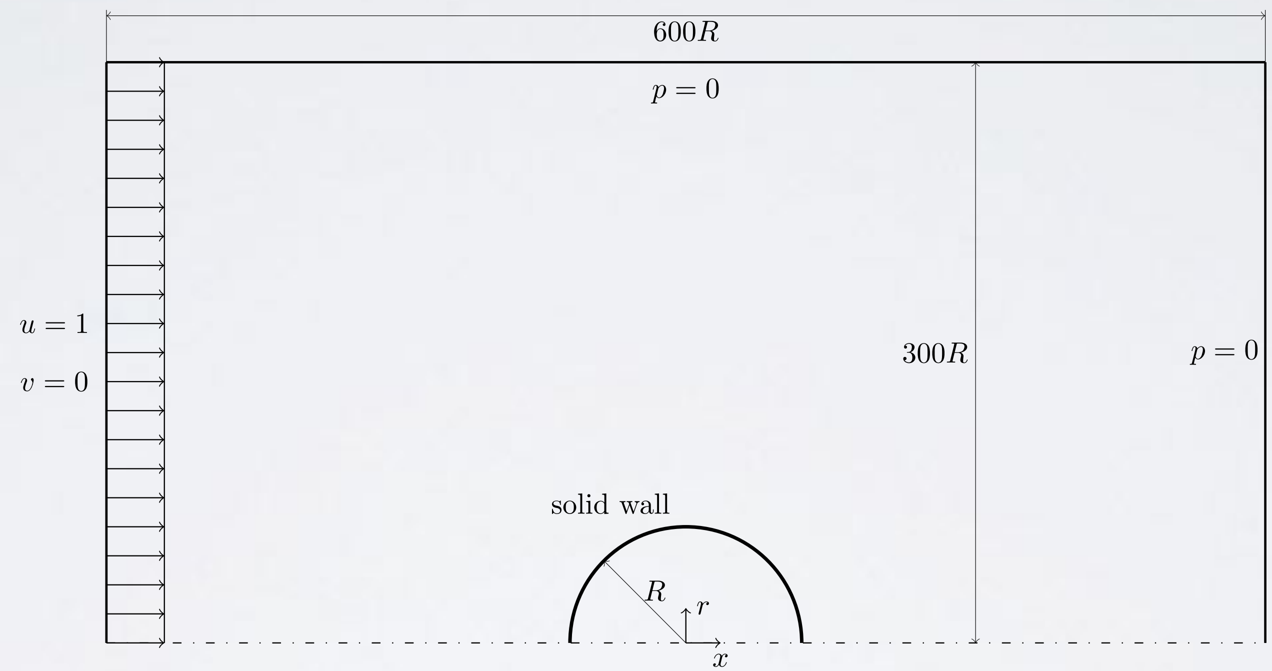


AXI TESTCASES

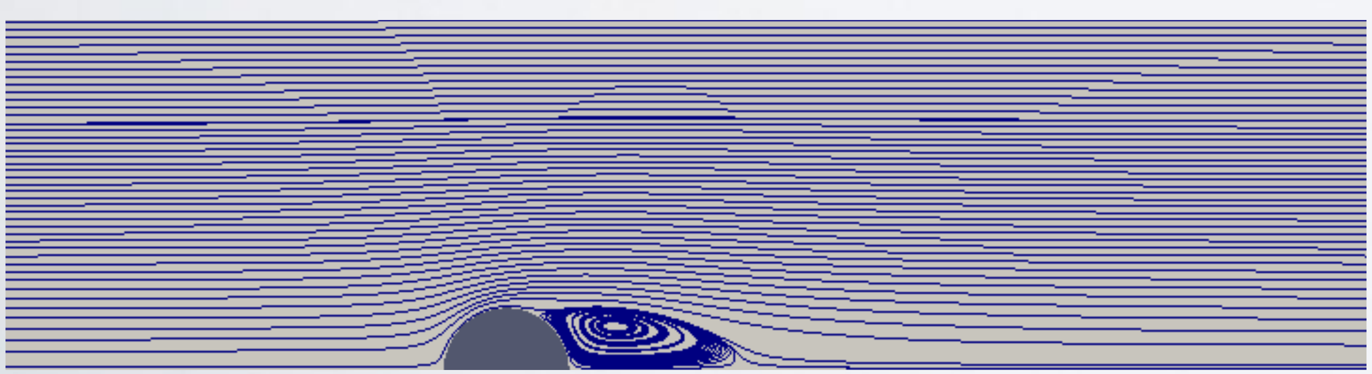
Flow around sphere

sphereAxi.py

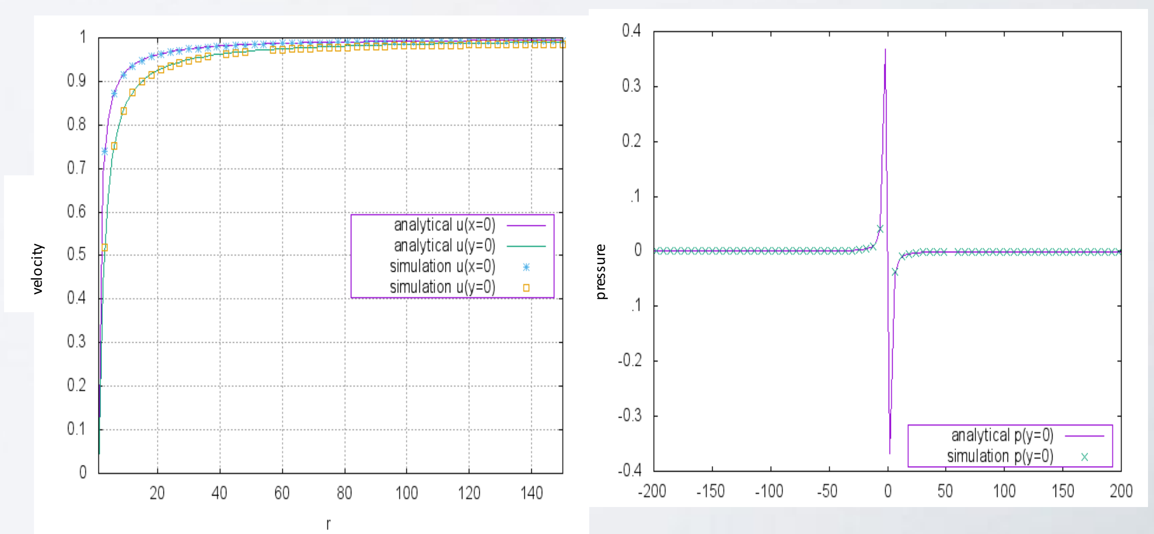
$$Re = 0.00001, \quad \frac{\rho_{in}}{\rho_{out}} = 1 \quad \frac{\mu_{in}}{\mu_{out}} = 1$$



high Re



low Re

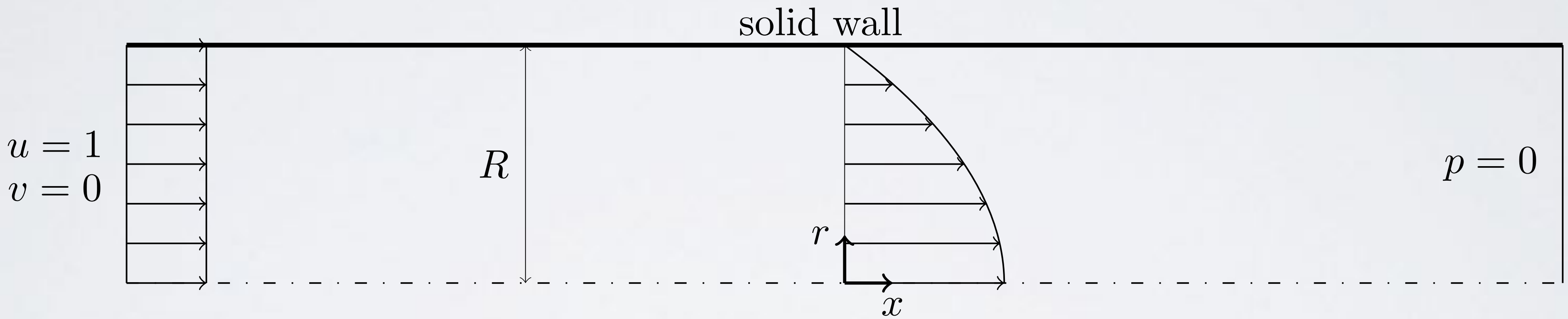


AXI TESTCASES

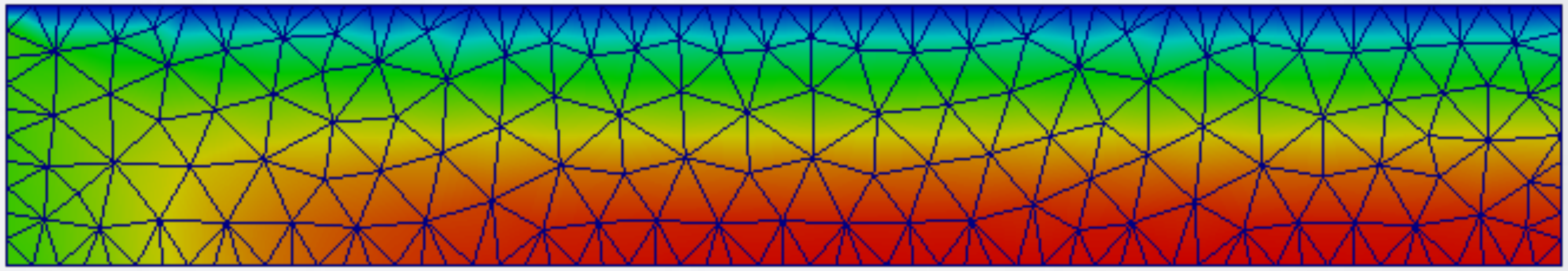
poiseiulleAxi.py

Poiseuille flow

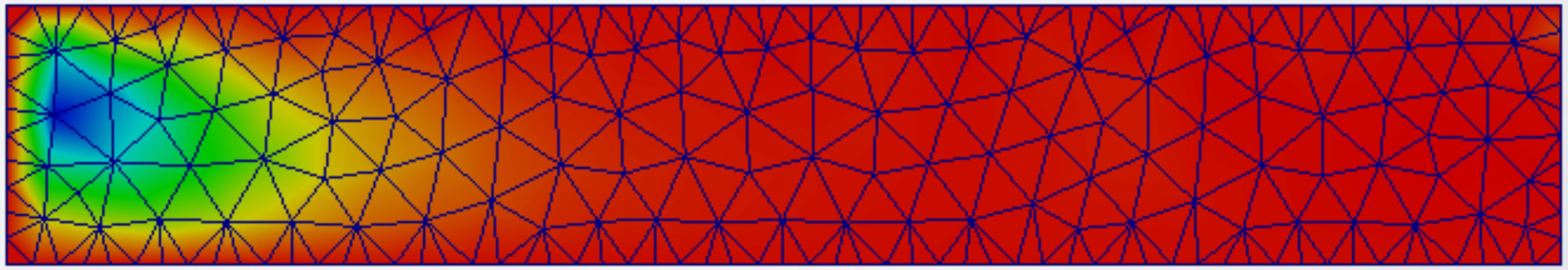
$$Re = 20, \qquad \frac{\rho_{in}}{\rho_{out}} = 1, \qquad \frac{\mu_{in}}{\mu_{out}} = 1$$



v_x velocity



v_r velocity

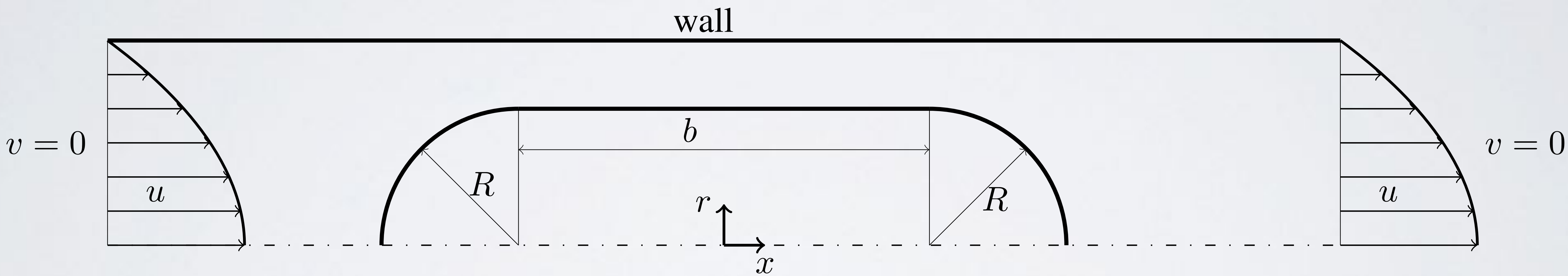


AXI TESTCASES

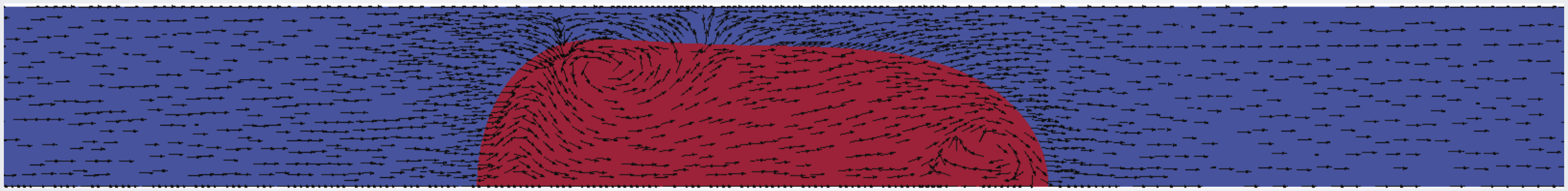
microAxi.py

Isolated air bubble in glycerol solution through microchannel

$$Re = 0.0128552, \quad We = 0.00127691, \quad \frac{\rho_{in}}{\rho_{out}} = 9.632 \cdot 10^{-4}, \quad \frac{\mu_{in}}{\mu_{out}} = 3.455 \cdot 10^{-5}$$



Initial mesh (1630 vertices), final mesh (14718 vertices)



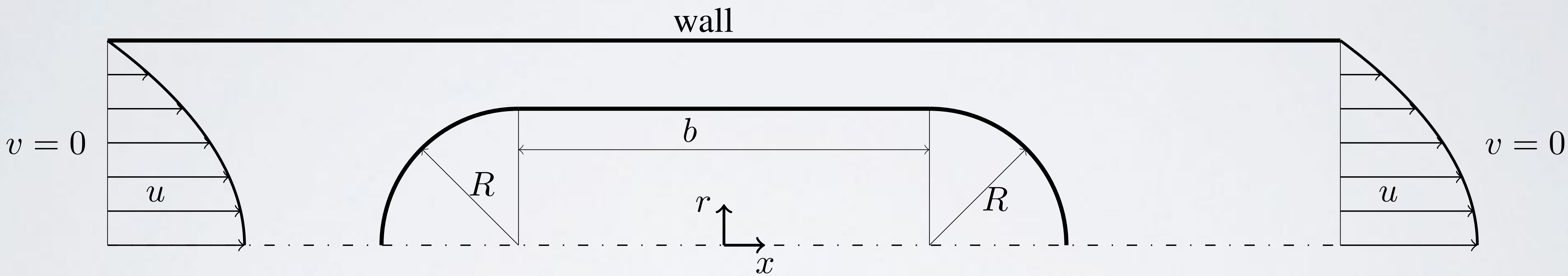
* **Khodaparast, S.; Magnini, M.; Borhani, N.; Thome, J.R.** Dynamics of isolated confined air bubbles in liquid flows through circular microchannels.- Microfluidics and Nanofluidics., **19:** 209-234 (2015)

AXI TESTCASES

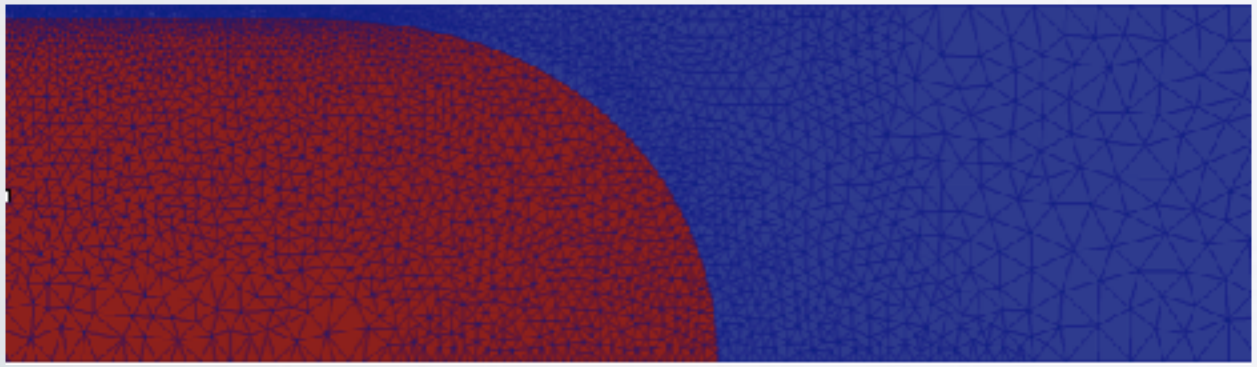
microAxi.py

Isolated air bubble in water through microchannel

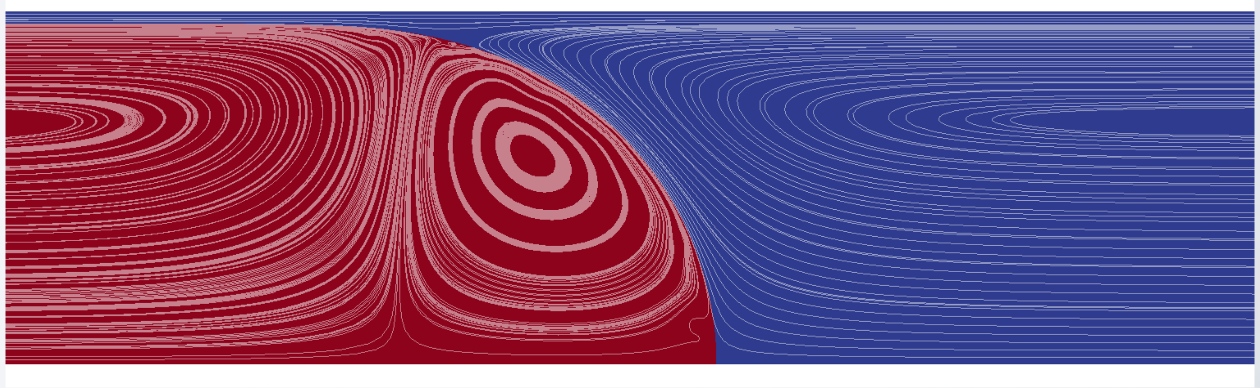
$$Re = 140.93, \quad We = 0.4122, \quad \frac{\rho_{in}}{\rho_{out}} = 1.2076 \cdot 10^{-3}, \quad \frac{\mu_{in}}{\mu_{out}} = 2.159 \cdot 10^{-2}$$



mesh



streamlines

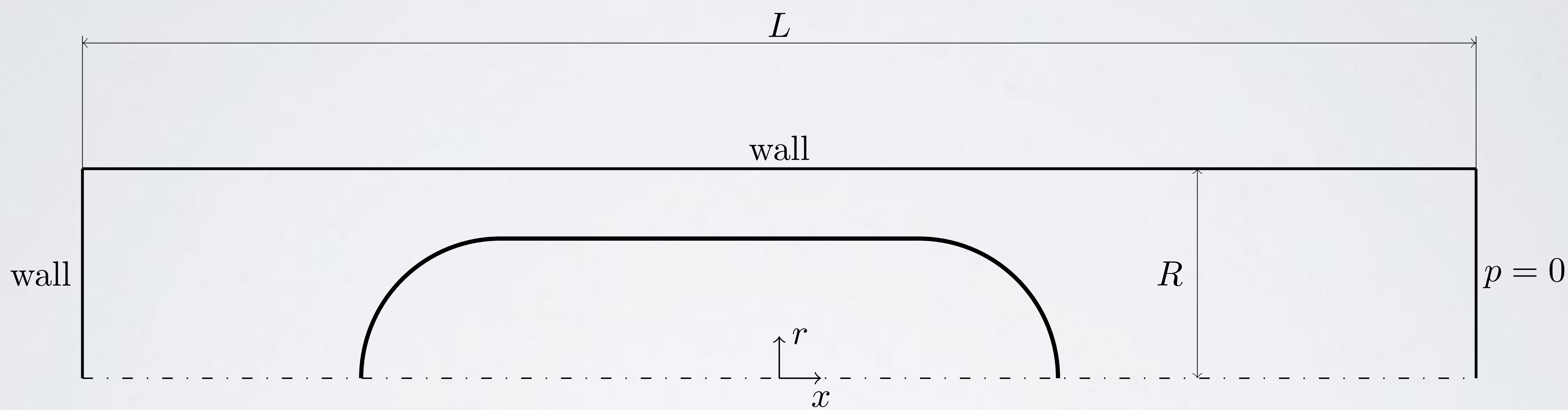


AXI TESTCASES

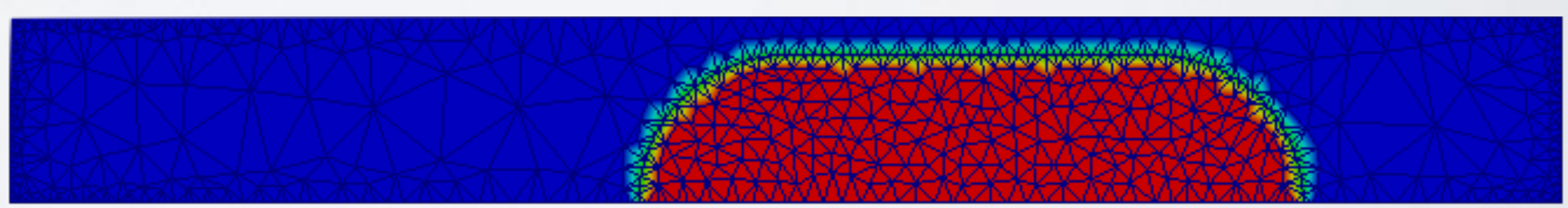
rising of isolated taylor bubble in microchannel

risingBubbleTaylorAxi.py

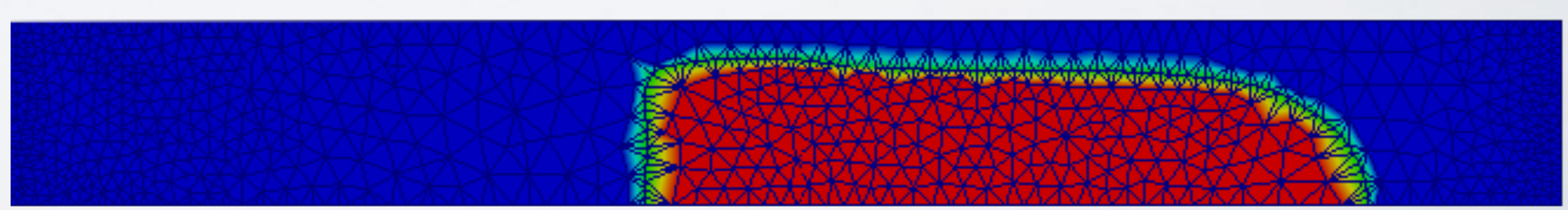
$$Re = 159.054, \quad We = 40, \quad Fr = 1, \quad \frac{\rho_{in}}{\rho_{out}} = 1038.37, \quad \frac{\mu_{in}}{\mu_{out}} = 2275.28$$



heaviside, iter=0



heaviside, iter=180

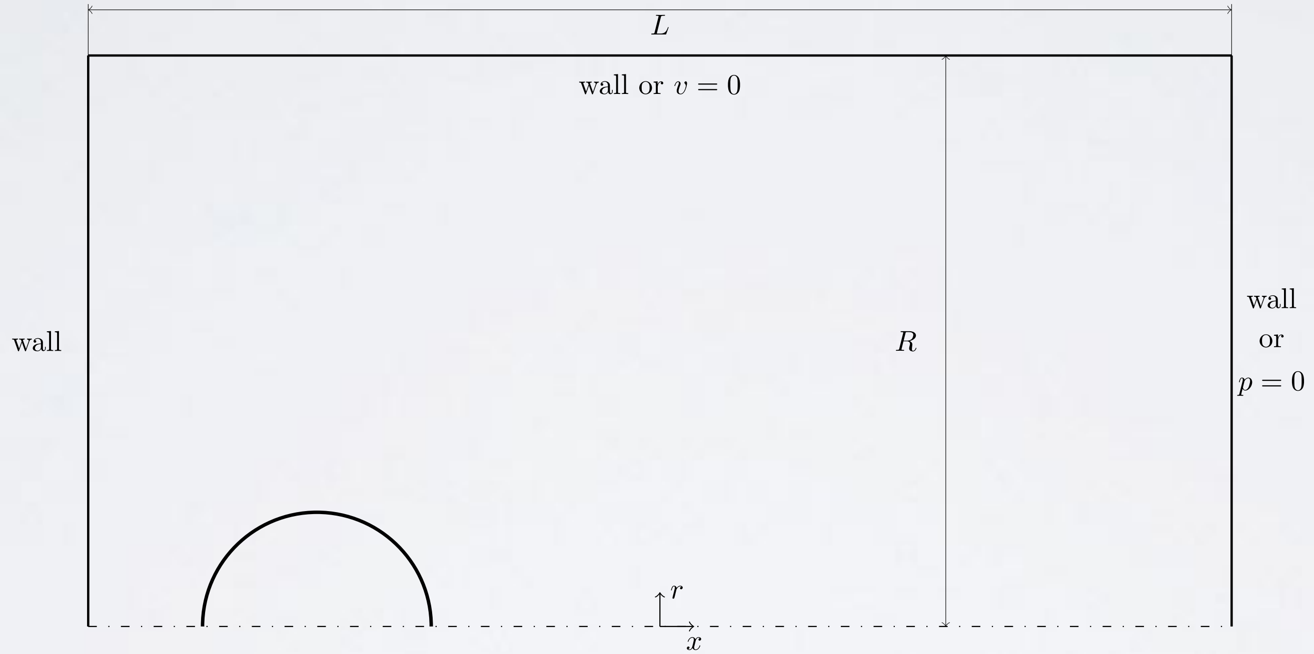


AXI TESTCASES

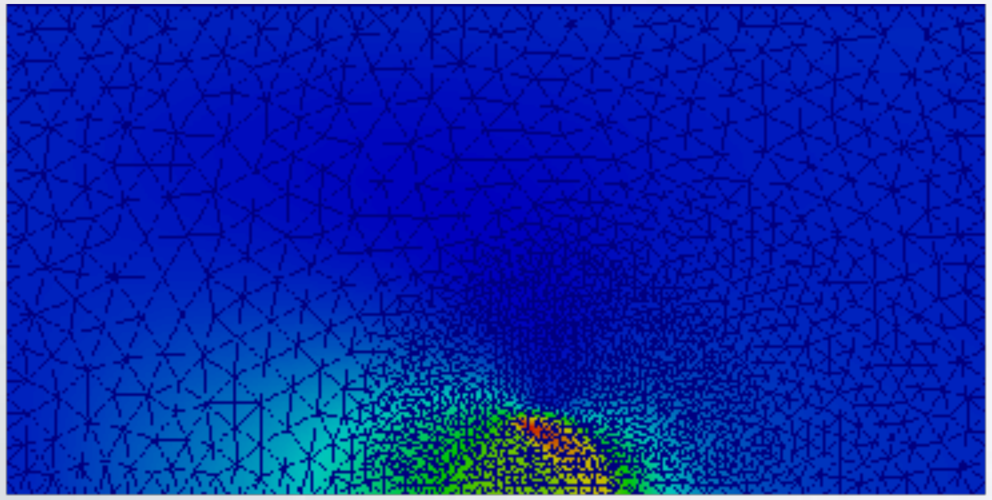
rising of air bubble
in sugar solution

risingBubbleAxi.py

$$Re = 13.84, \quad We = 115.662, \quad Fr = 1, \quad \frac{\rho_{in}}{\rho_{out}} = 1102.04, \quad \frac{\mu_{in}}{\mu_{out}} = 71910.1$$



v_x velocity



v_r velocity