ULaMBAtoR

1. Workshop

December 11th 2015
TIPs Lab

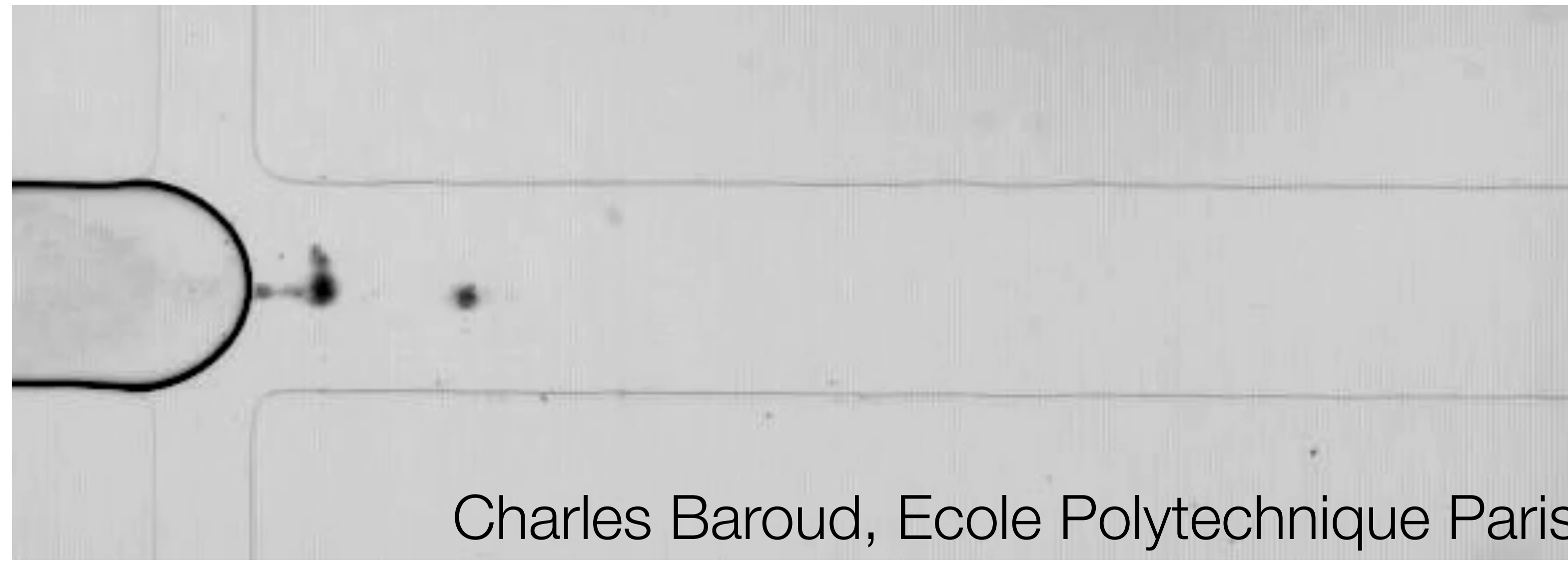
ULB

Course contents

1. History and raison d’être
2. Get to know each other
3. Numerical method
4. User control
5. Reference information
6. Running examples
7. Your projects

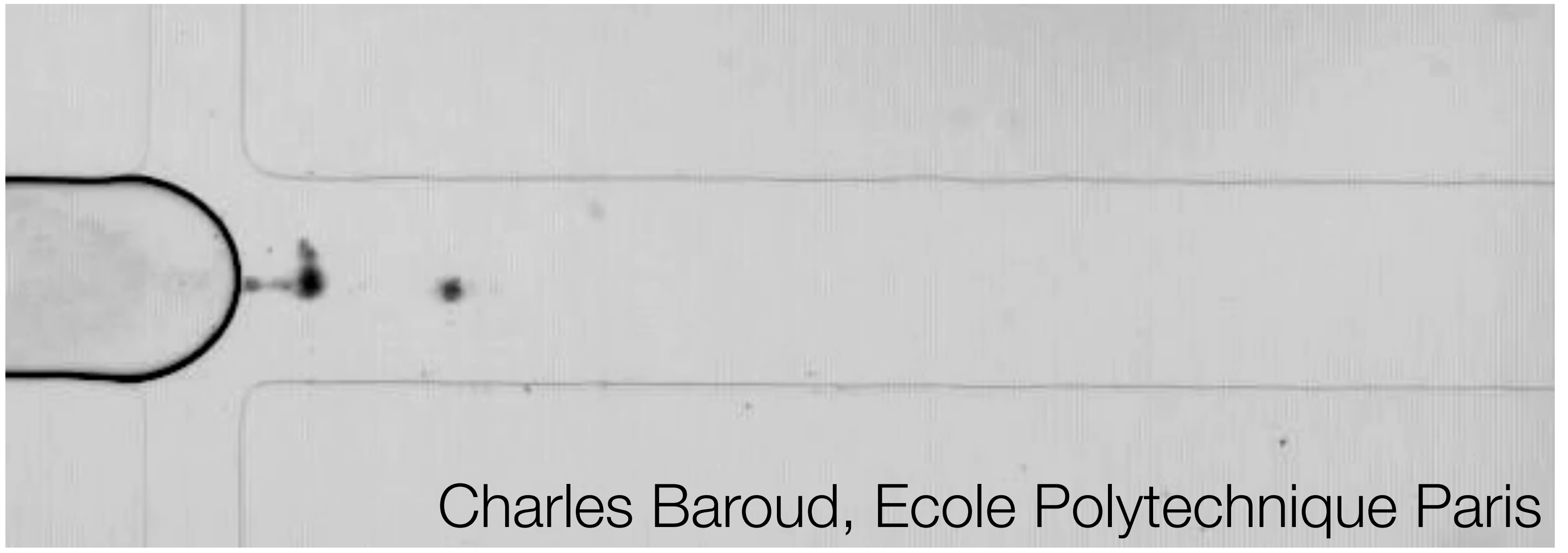
History

A solution scheme for complicated two phase flow?



History

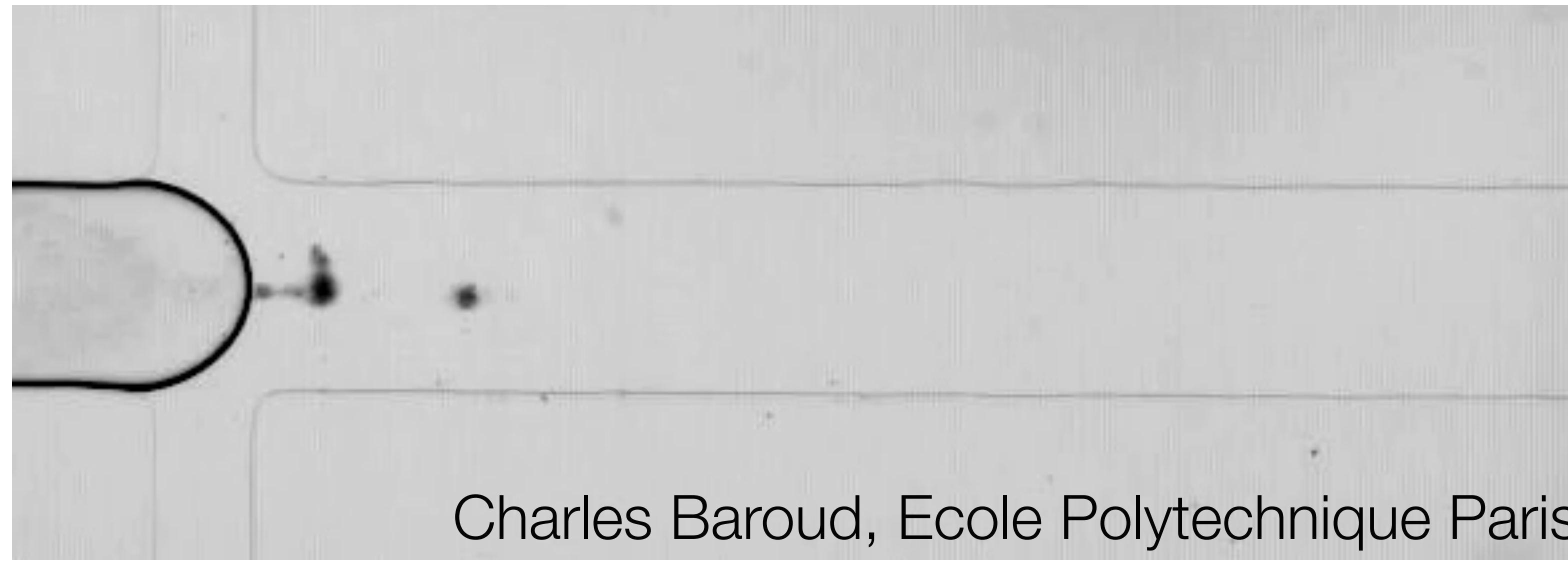
A solution scheme for complicated two phase flow?

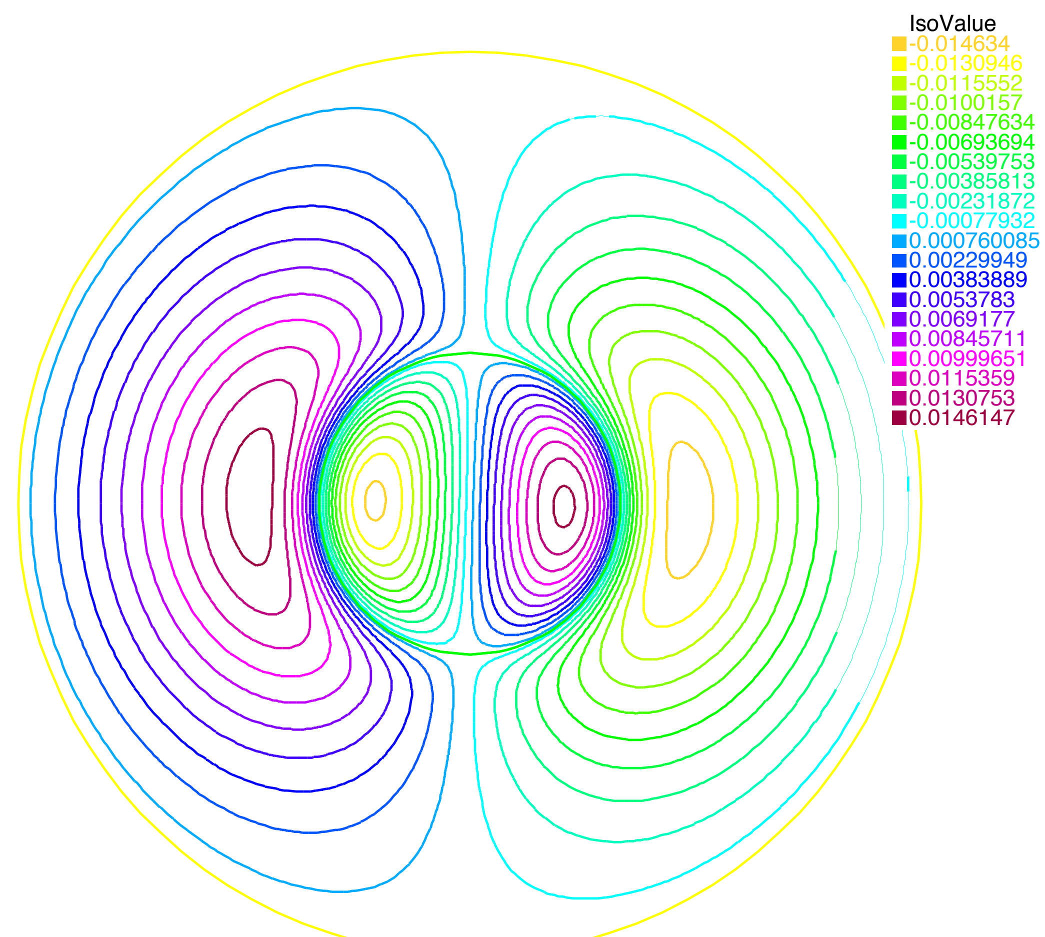


History

A solution scheme for complicated two phase flow?

- FreeFem from UPMC

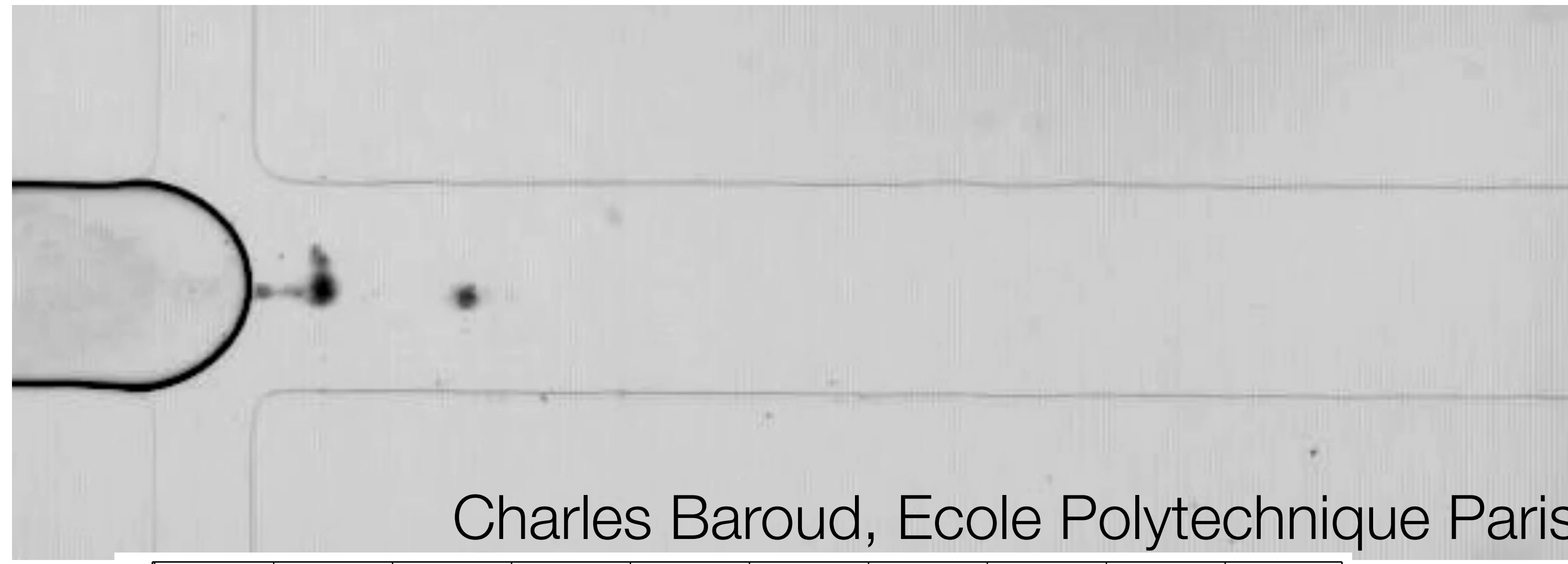


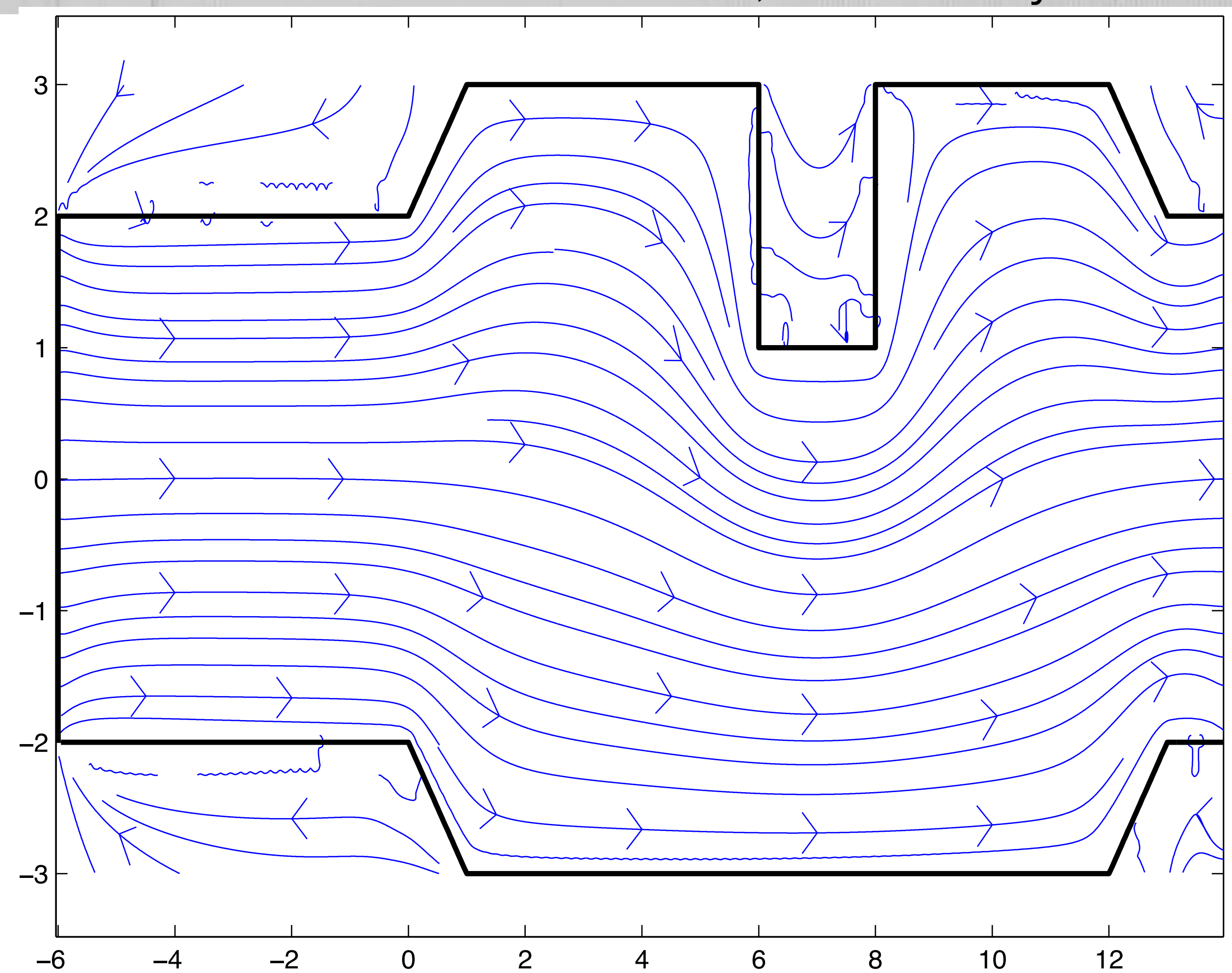


History

A solution scheme for complicated two phase flow?

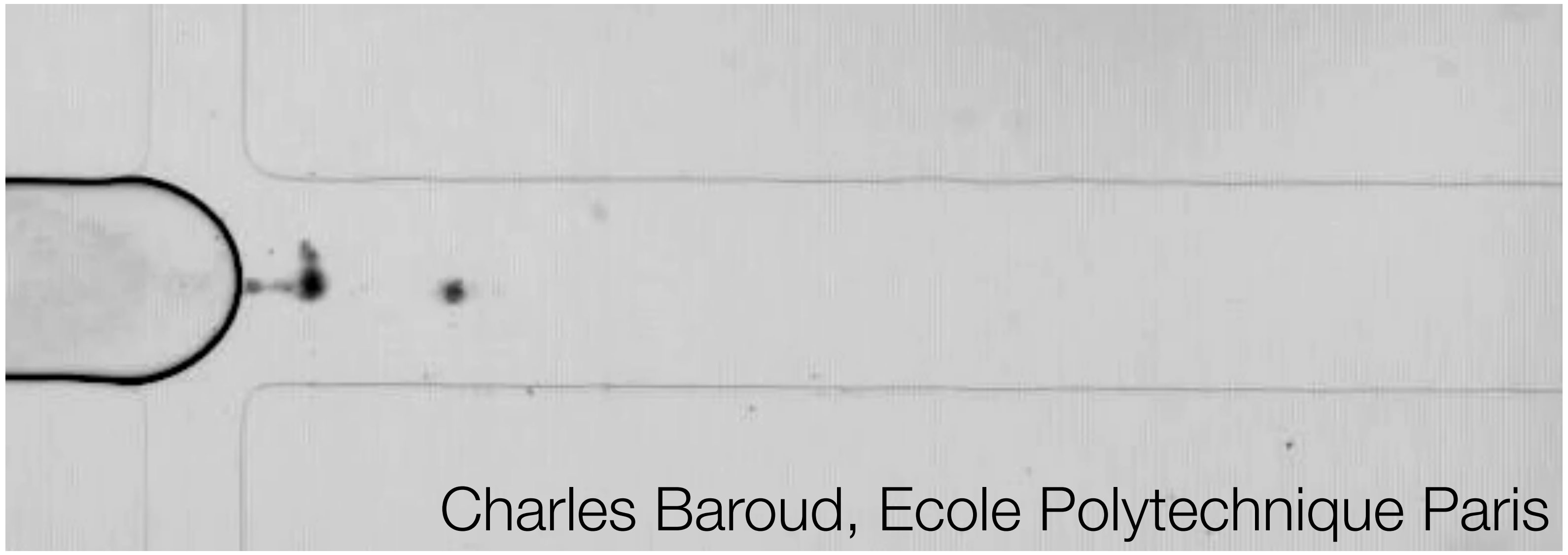
- FreeFem from UPMC
- Fundamental solutions





History

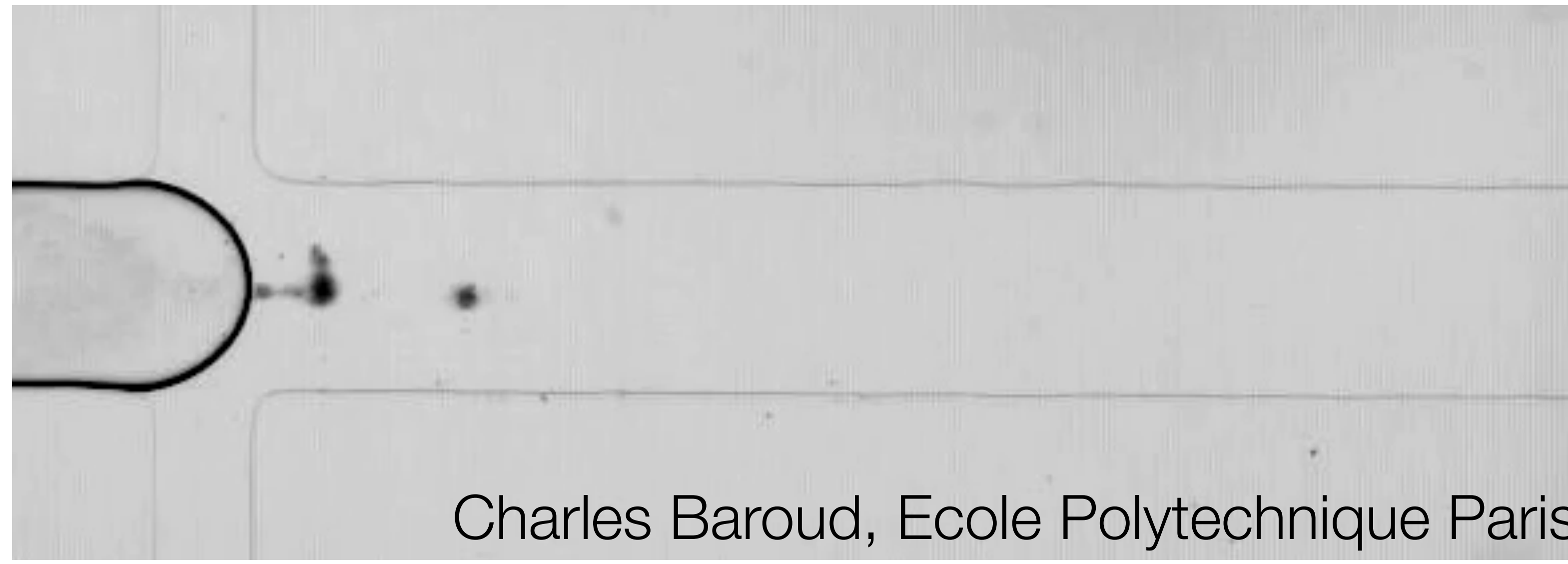
A solution scheme for complicated two phase flow?



- FreeFem from UPMC
- Fundamental solutions
- USBEC (Matlab)

History

A solution scheme for complicated two phase flow?



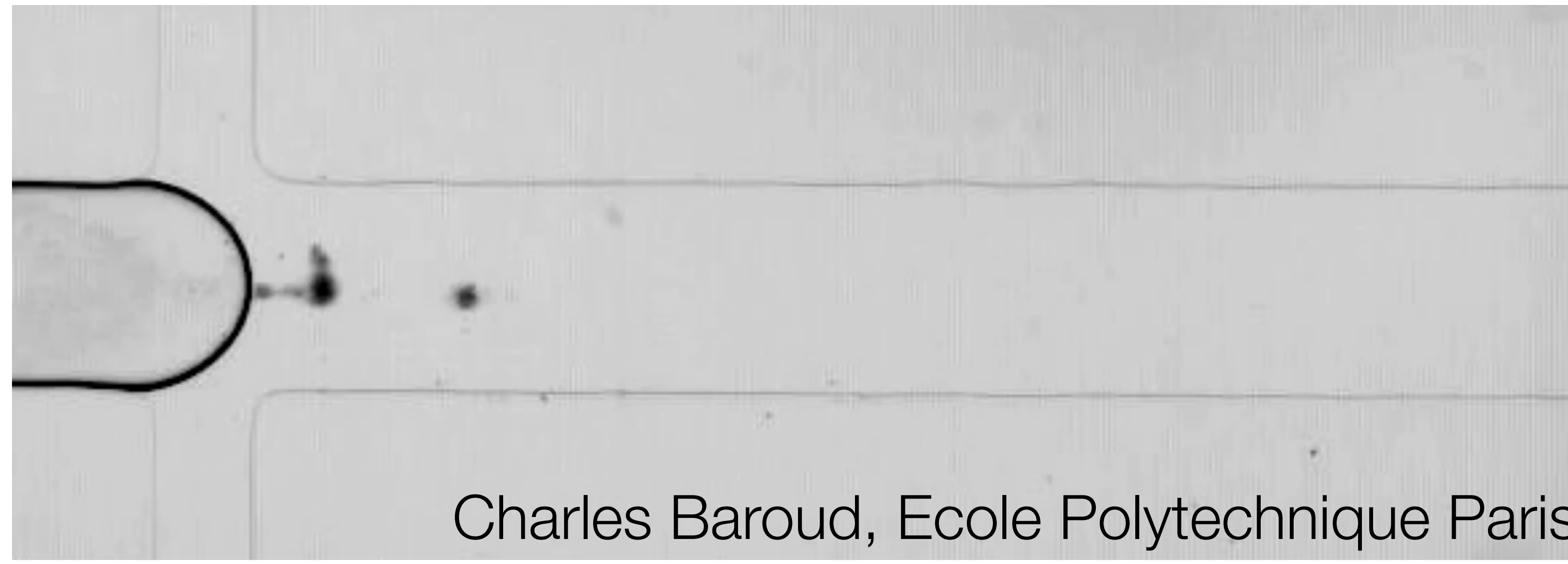
- FreeFem from UPMC
- Fundamental solutions
- USBEC (Matlab)
- ULAMBATOR (C++)

ULaMBATOR

UNSTEADY LAMINAR MICROFLUIDIC BEM ALGORITHM TO OBTAIN RESULTS

History

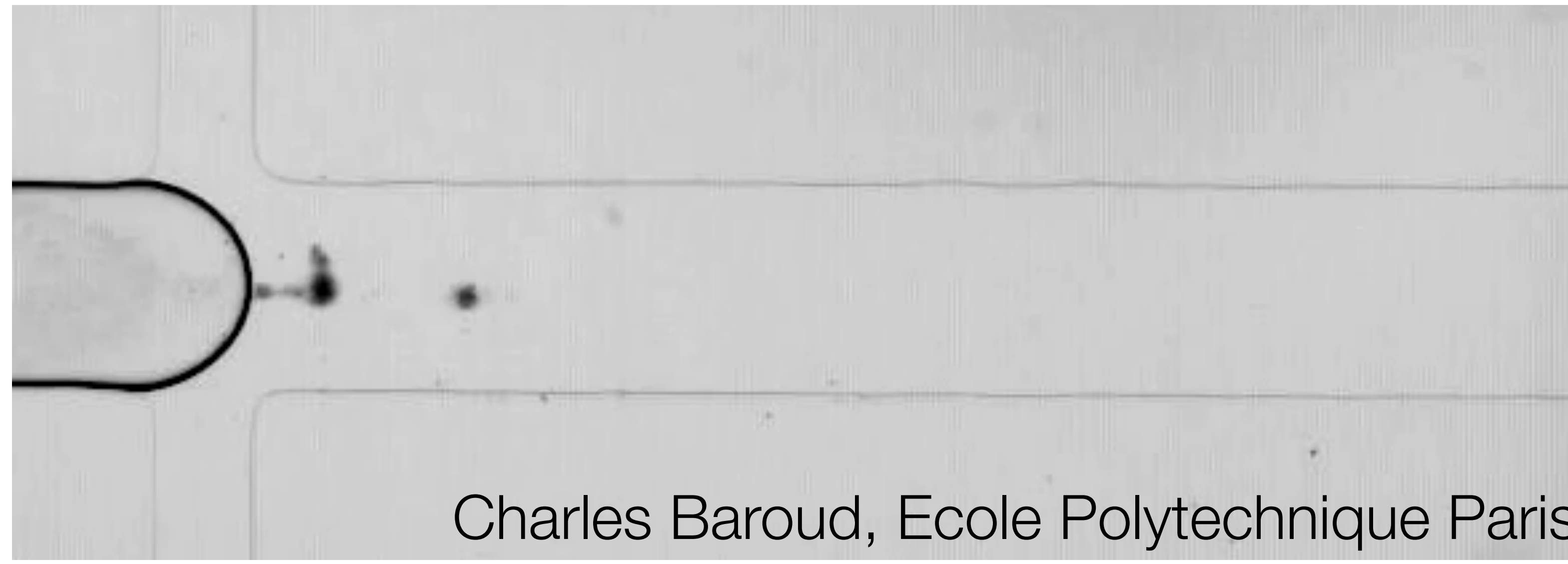
A solution scheme for complicated two phase flow?



- FreeFem from UPMC
- Fundamental solutions
- USBEC (Matlab)
- ULAMBATOR (C++)

History

A solution scheme for complicated two phase flow?



- FreeFem from UPMC
- Fundamental solutions
- USBEC (Matlab)
- ULAMBATOR (C++)
- ULAMBATOR 1.0

Demo

- Write/Modify an Ulambator script
- Launch it from the Terminal
- Analyze the output

Demo

- Write/Modify an Ulambator script
- Launch it from the Terminal
- Analyze the output

Tutorial 1 - Single phase flow in a bend

In this tutorial you will learn how to create fixed boundaries and to solve a single phase flow.

```
/*
Ulambator configuration file
Date: 09.12.2014
Version 0.7
*/
```

File Header

Together with included classes we add Ulambators `matblc1.h` and `bndblock.h`, which manage the geometry and generate the matrix that solves the posed problem.

```
#include "../source/matblc1.h"
#include "../source/bndblock.h"
#include <iostream>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <math.h>
```

The program begins with `int main` and prints a greeting to the screen.

```
int main (int argc, char * const argv[]) {
    std::cout <<"Welcome to ULAMBATOR++ v.7\n";
    std::cout <<"Unsteady LAminar Microfluidic Boundary element Algorithm To Obtain Results\n";
    char* savedir;
```

Defining the geometry

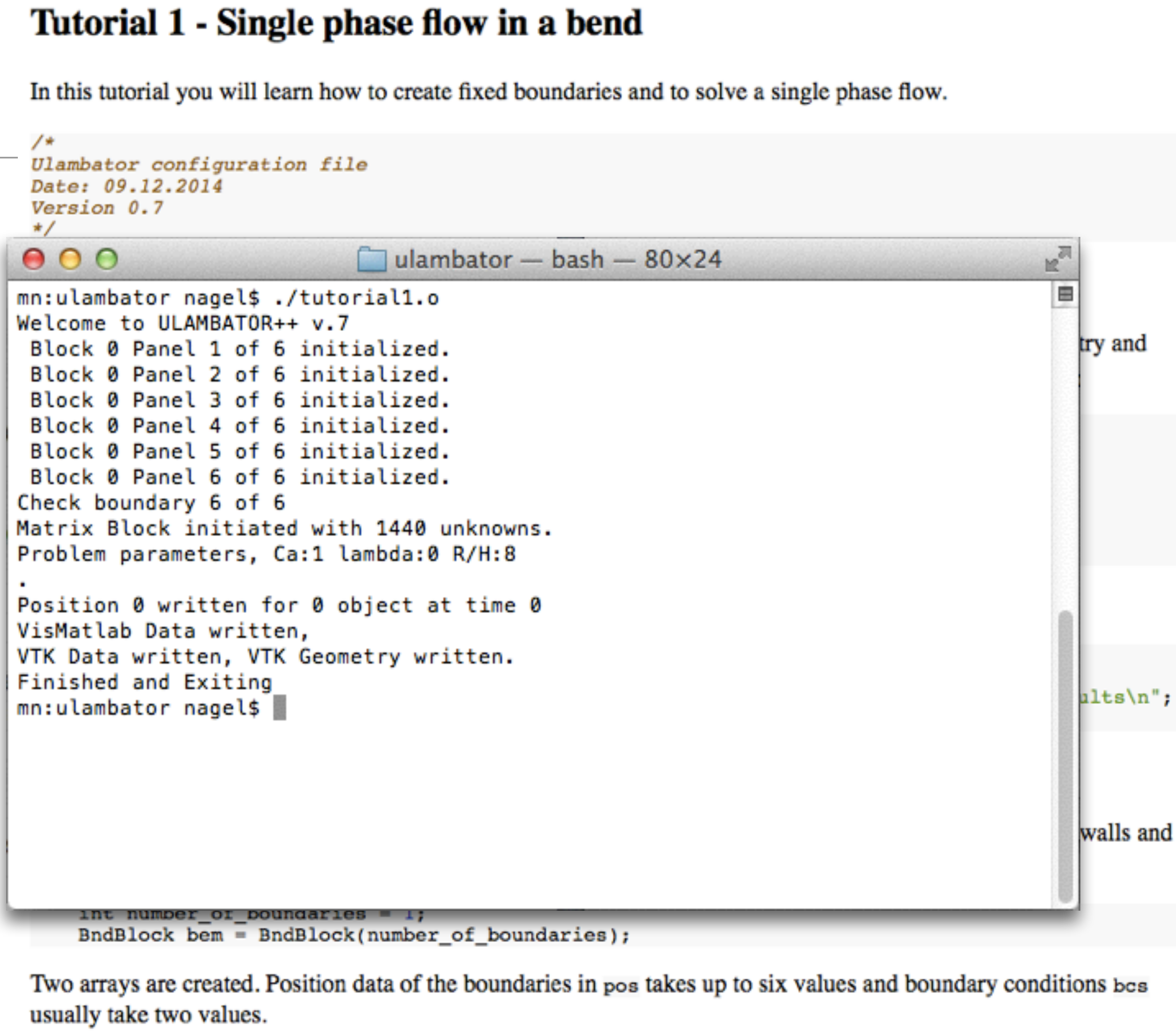
The object `bem` is a boundary block, which works as a container that will manage all boundaries like fixed walls and liquid interfaces. Here we initialize the container to contain only a single boundary.

```
    int number_of_boundaries = 1;
    BndBlock bem = BndBlock(number_of_boundaries);
```

Two arrays are created. Position data of the boundaries in `pos` takes up to six values and boundary conditions `bcs` usually take two values.

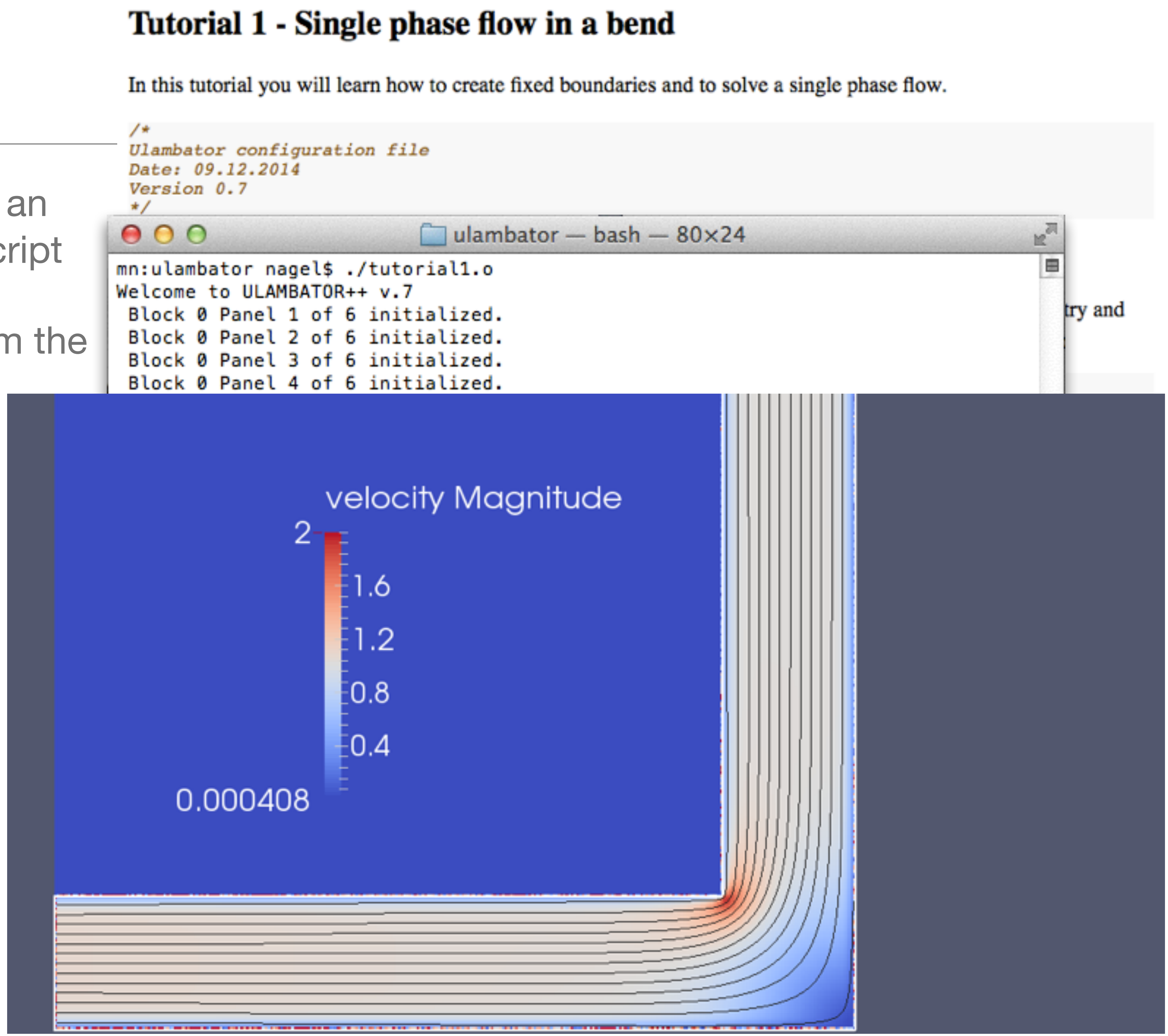
Demo

- Write/Modify an Ulambator script
- Launch it from the Terminal
- Analyze the output



Demo

- Write/Modify an Ulambator script
- Launch it from the Terminal
- Analyze the output



Mutual expectations

Mutual expectations

- This is the first workshop
  It’s experimental in style and contents.
  The open-source tool is based on a personal code.

Mutual expectations

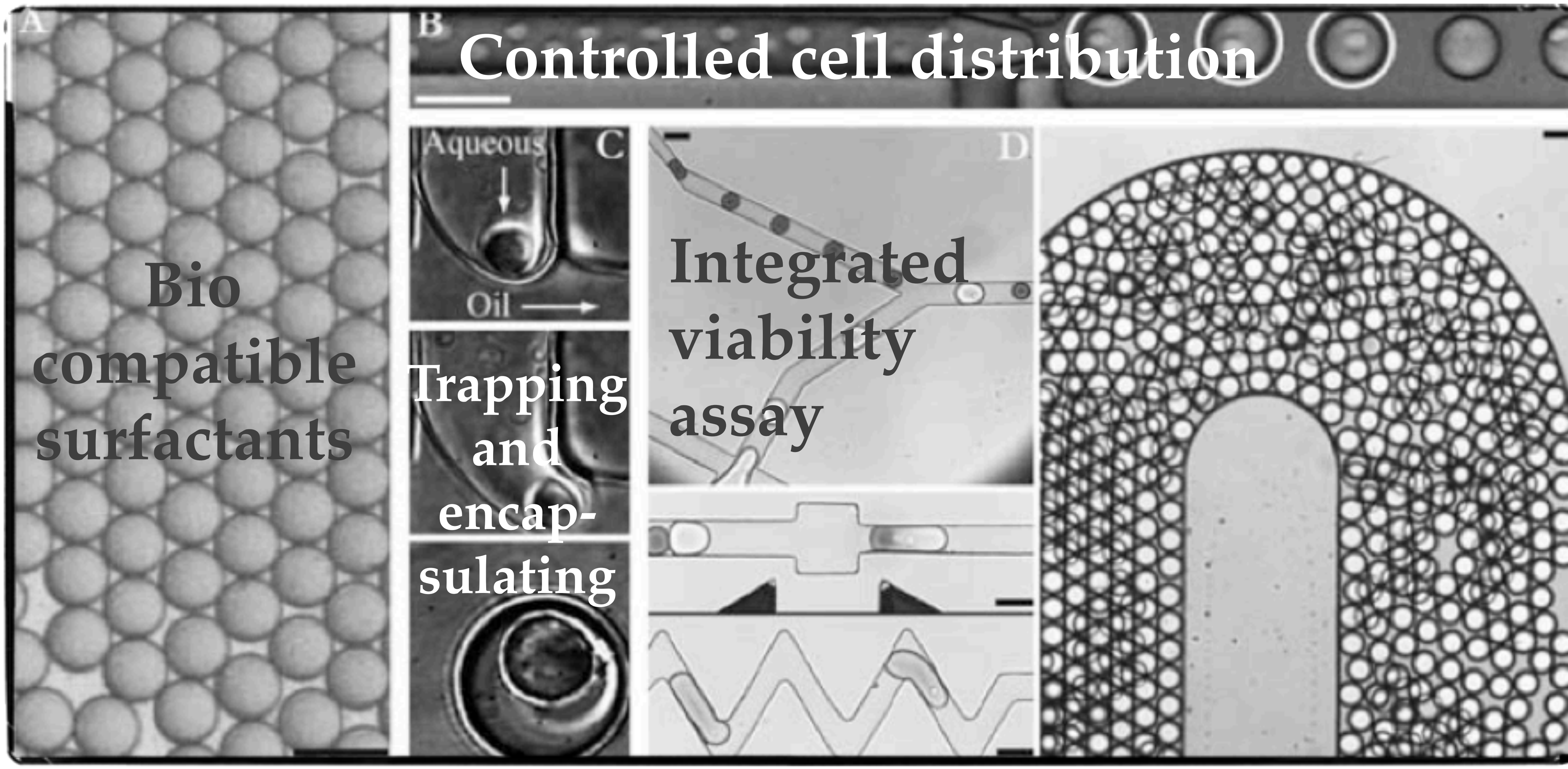
- This is the first workshop
  It’s experimental in style and contents.
  The open-source tool is based on a personal code.

- Your feedback
  Improves the functionalities and usability.
  Improves the presentation and the didactics.

Getting to know each other

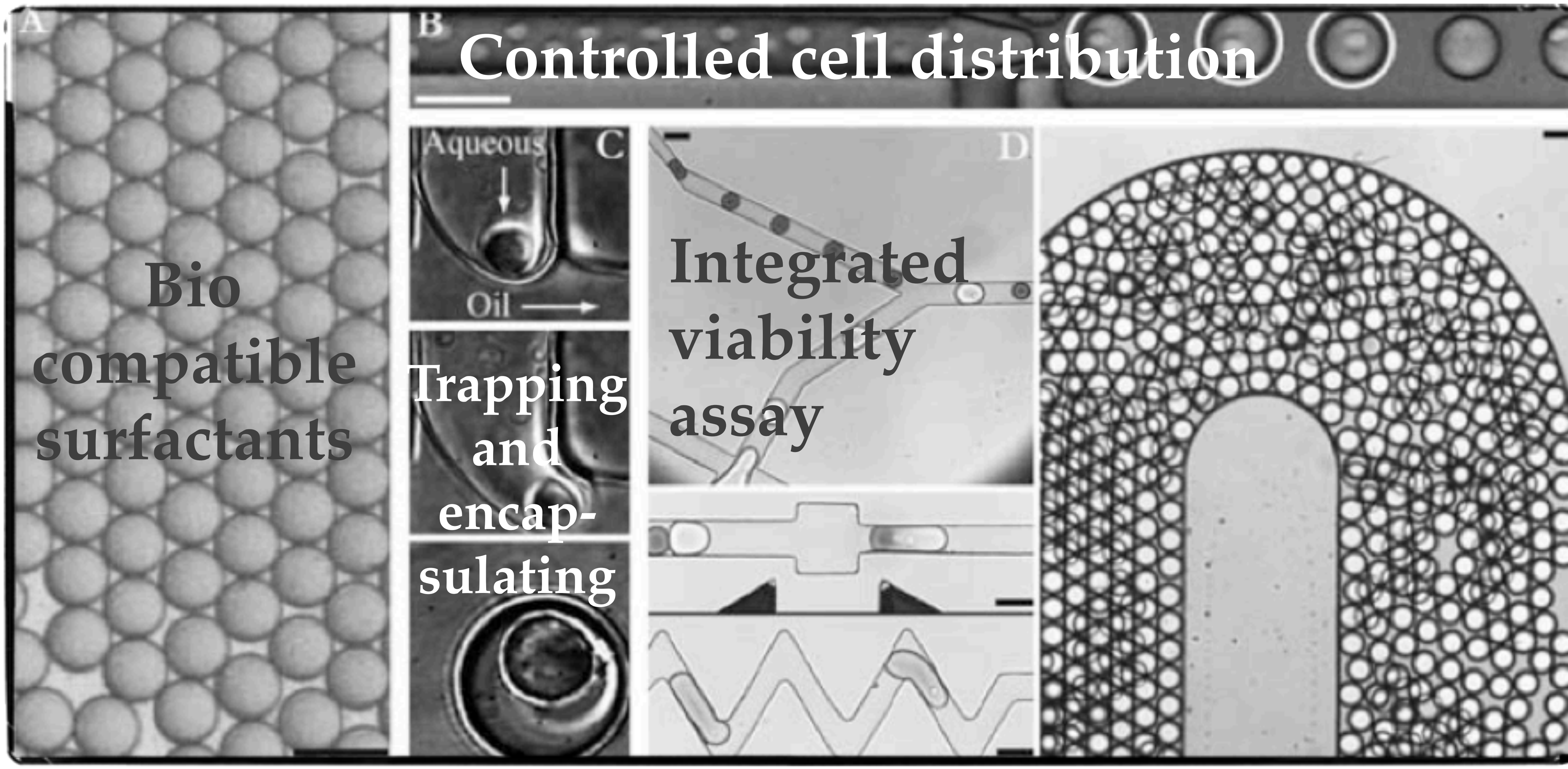
Numerical method

Numerical method



Lindström and Anderssohn-Svahn in Lab on a Chip 10 (2010)
Overview of single-cell analyses: microdevices and applications

Numerical method



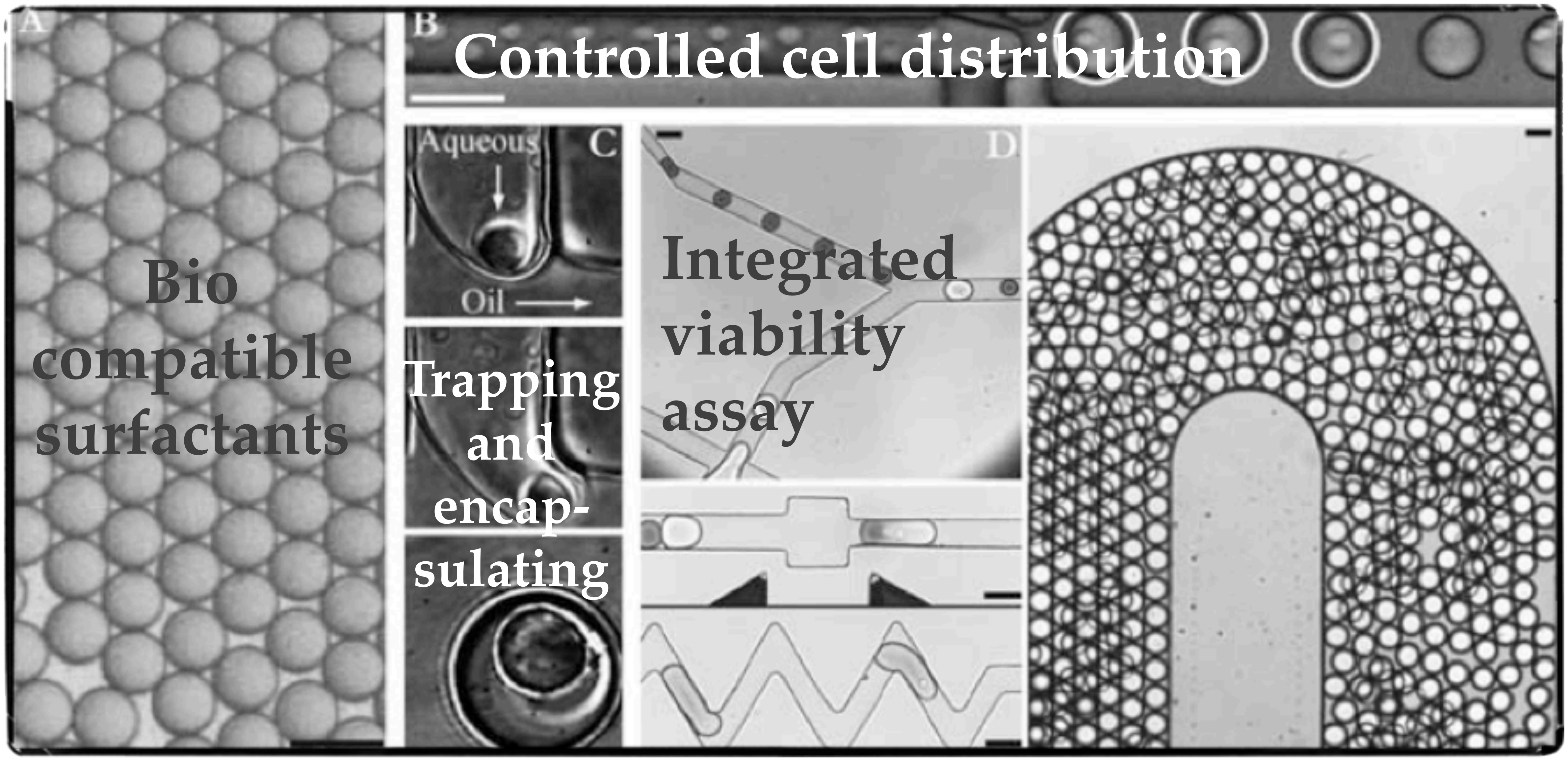
Lindström and Anderssohn-Svahn in Lab on a Chip 10 (2010)
Overview of single-cell analyses: microdevices and applications

$$Ca = \frac{\mu U}{\gamma}$$

$$Re = \frac{\rho U H}{\mu}$$

$$\frac{W}{H}$$

Numerical method

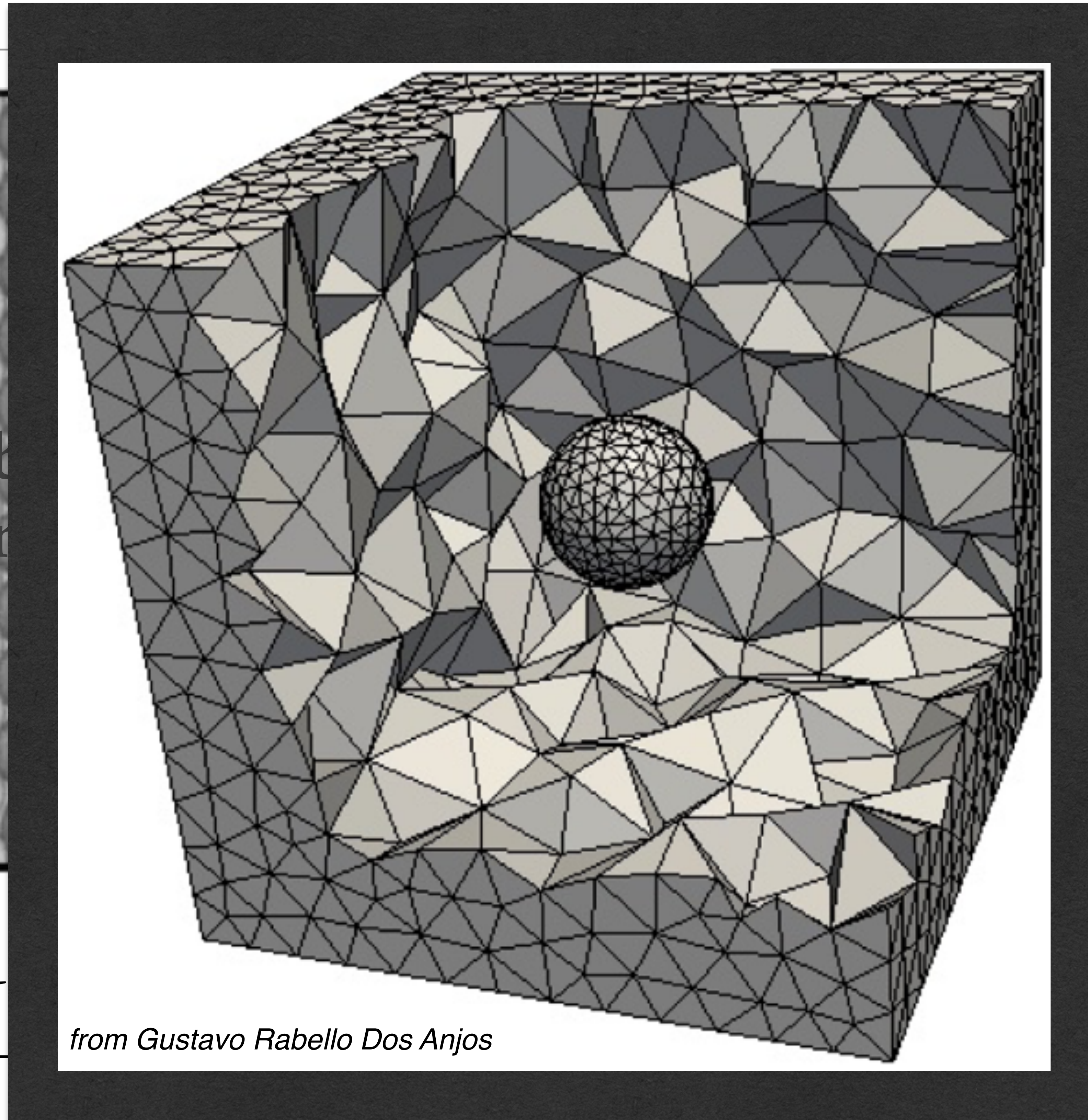


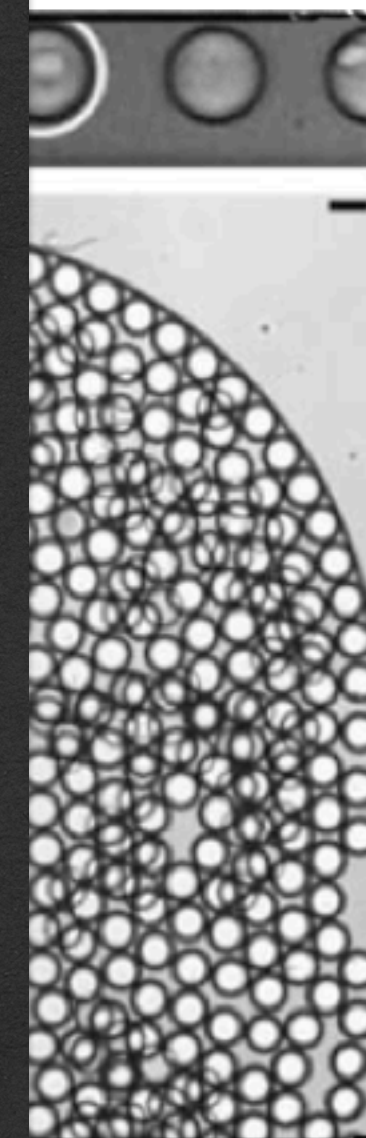
Lindström and Anderssohn-Svahn in Lab on a Chip 10 (2010)
Overview of single-cell analyses: microdevices and applications

$$Ca = \frac{\mu U}{\gamma} << 1$$

$$Re = \frac{\rho U H}{\mu} << 1 \qquad \frac{W}{H} >> 1$$

Numerical method





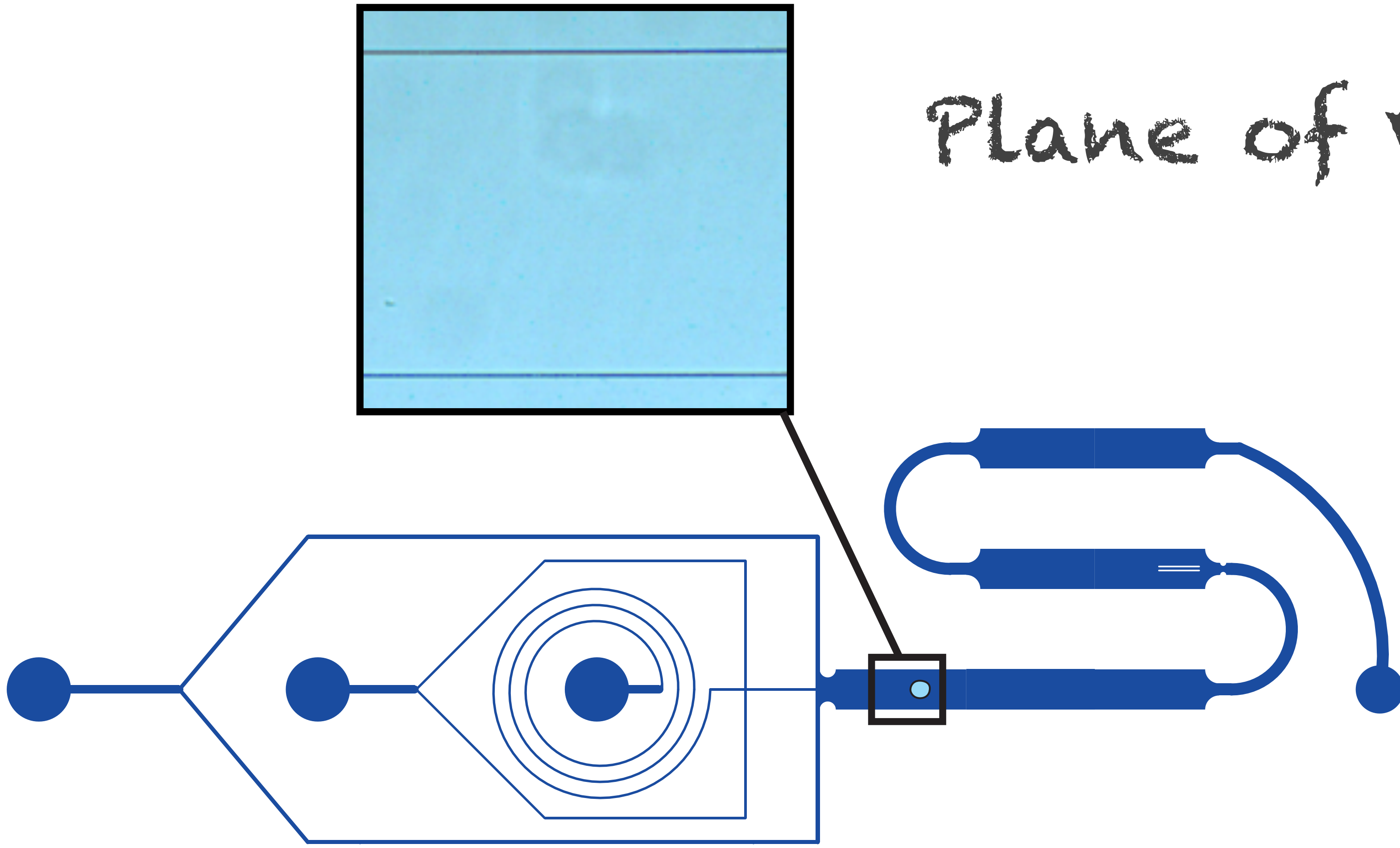
Chip 10 (2010)
d applications

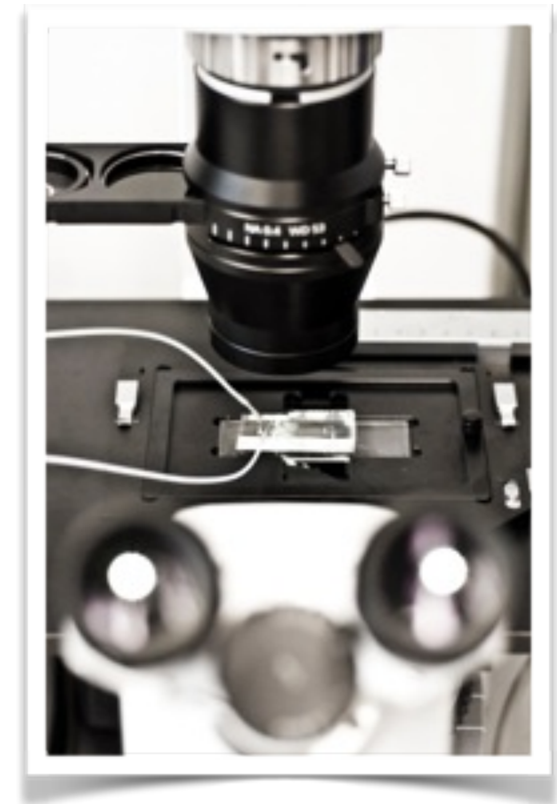
$$Ca = \frac{\mu U}{\gamma}$$

μ

H >> 1

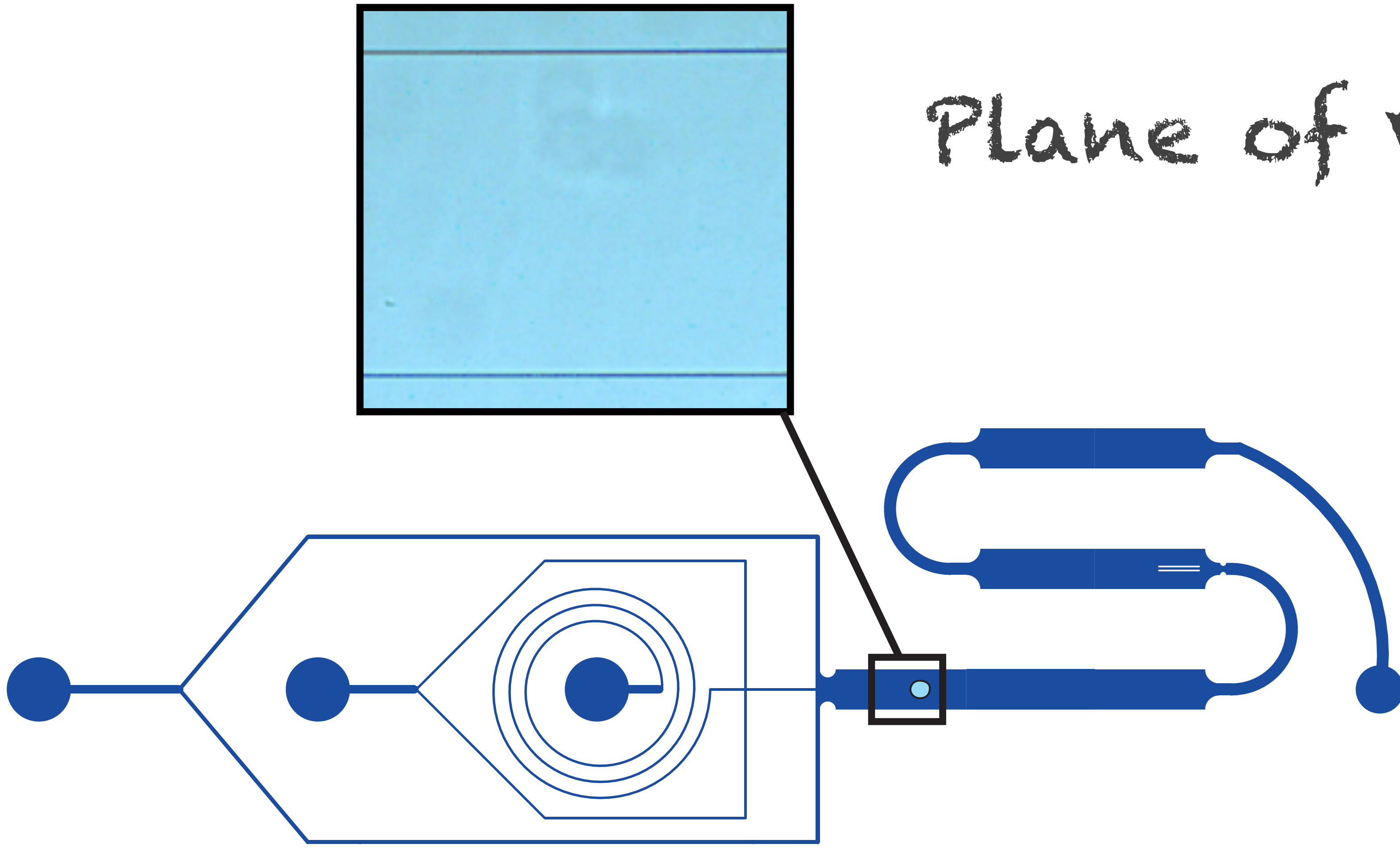
Plane of Variance

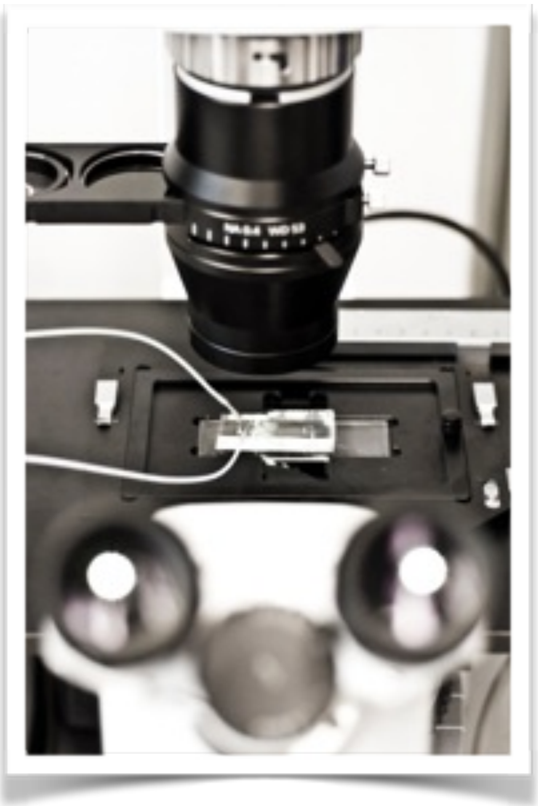




Method

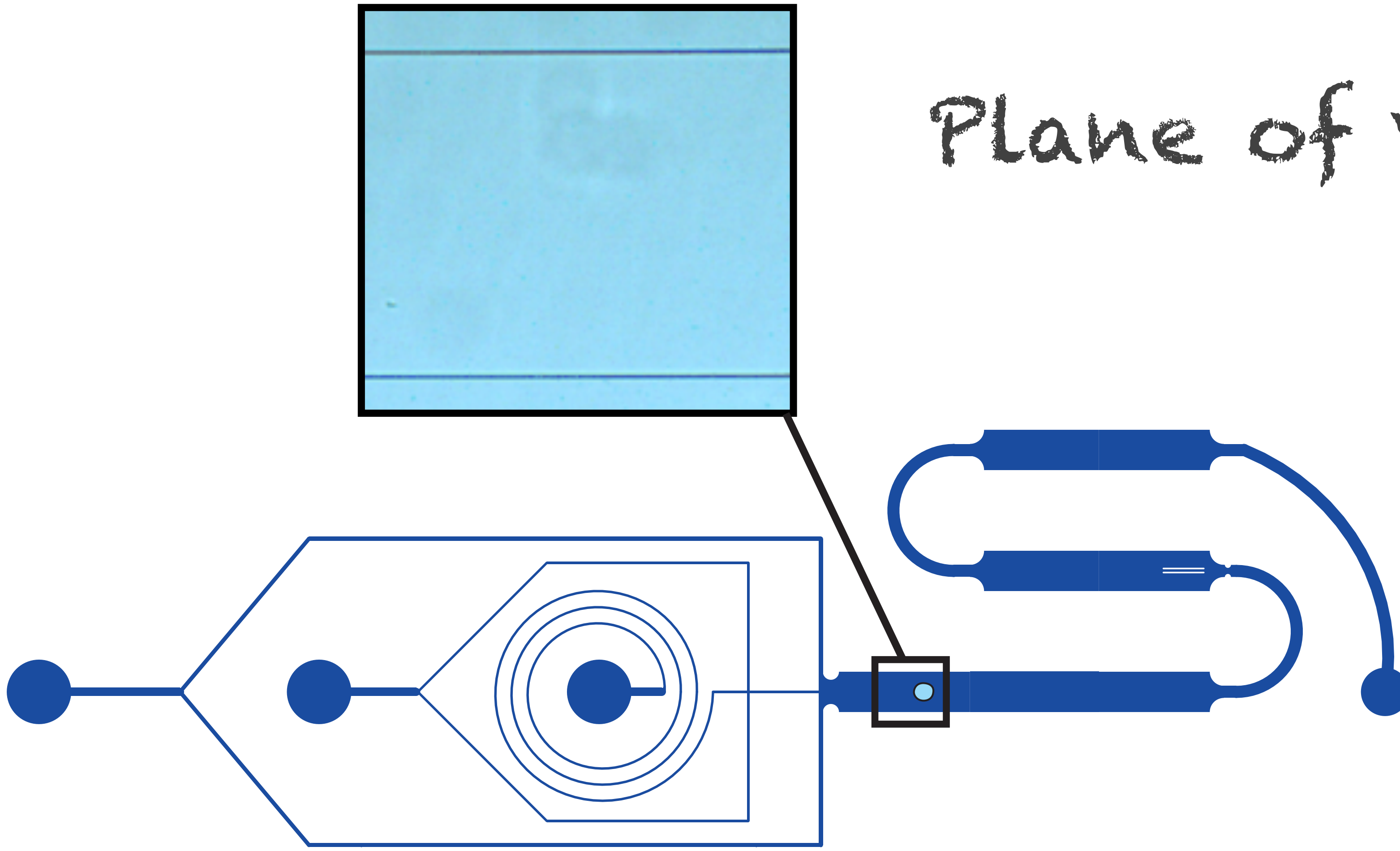
Plane of Variance

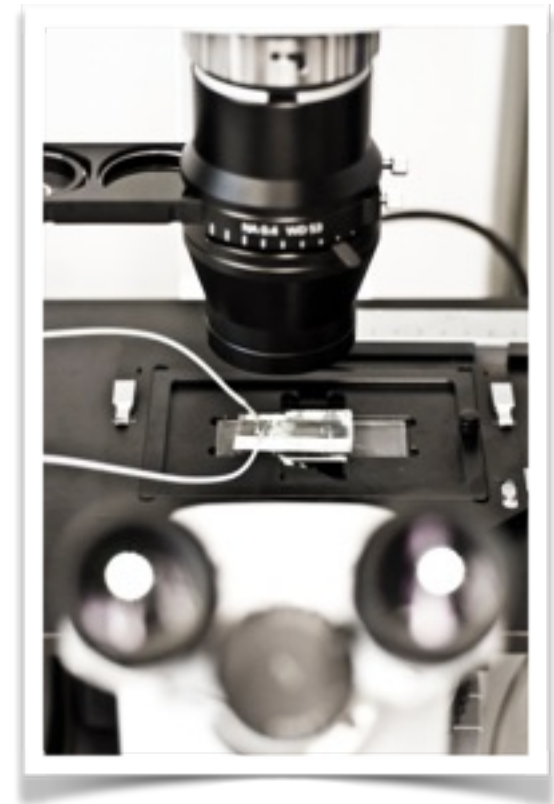




Method

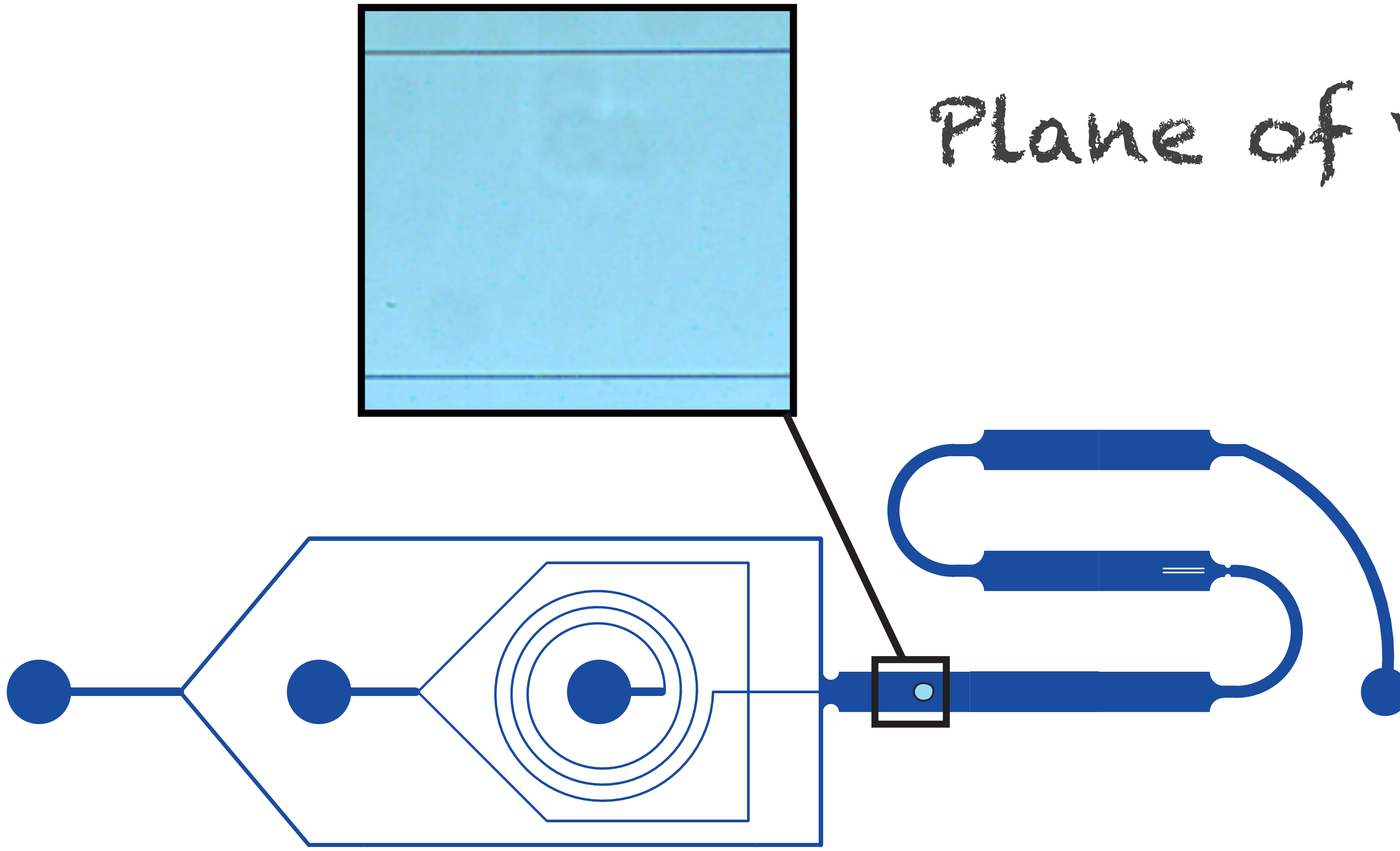
Plane of Variance

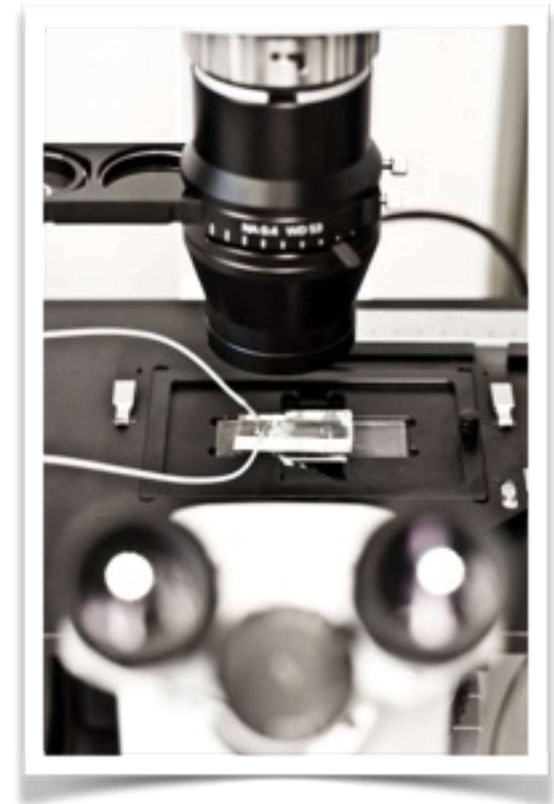


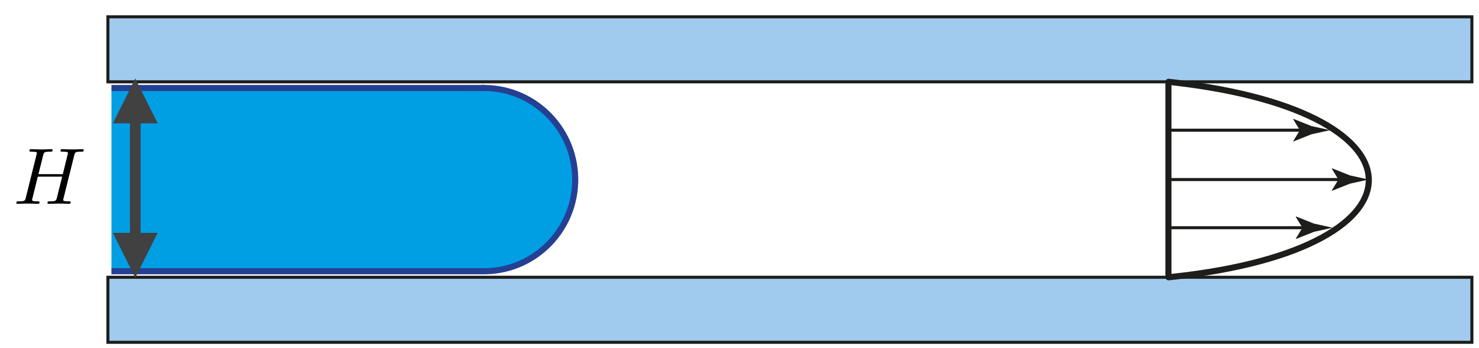


Method

Plane of Variance



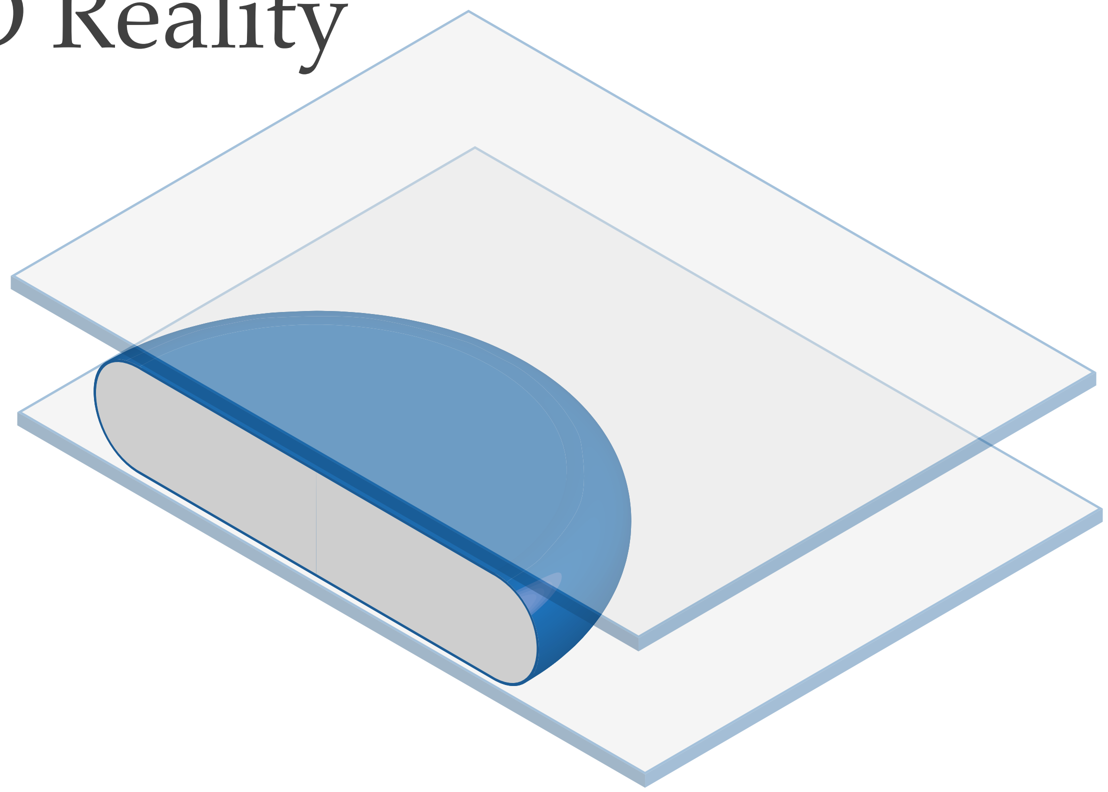




Direction of Dominance

Method

3D Reality



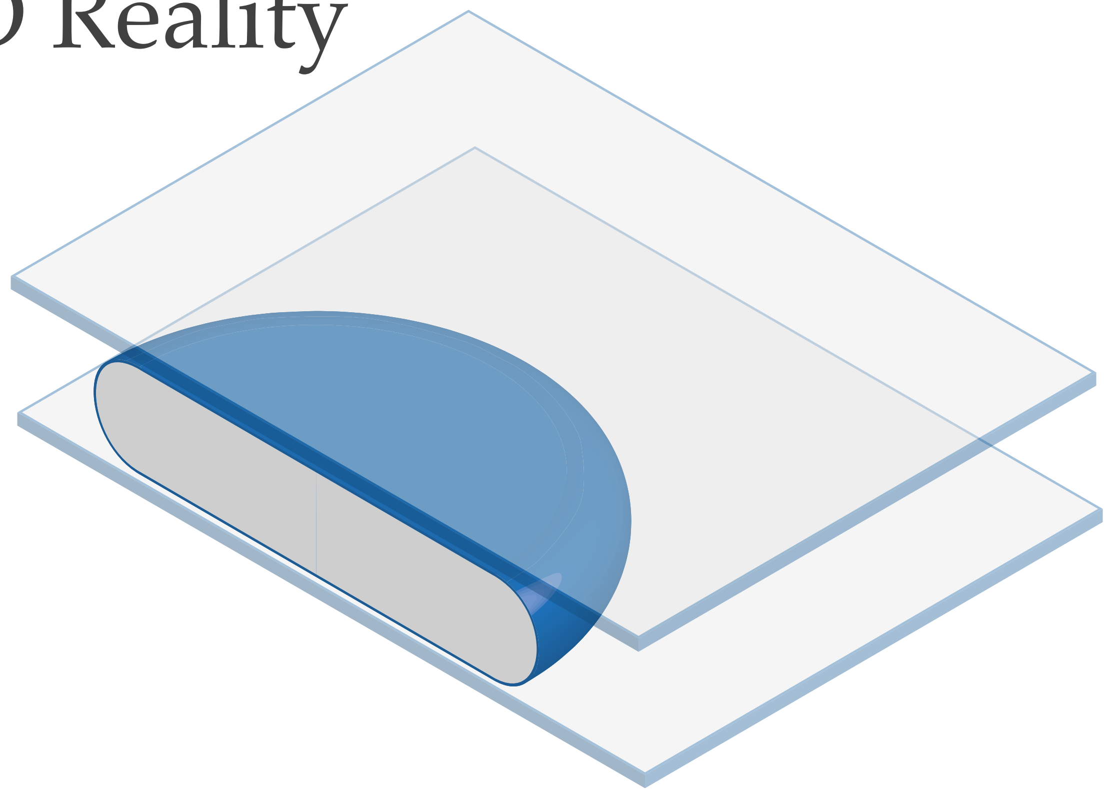
3D Stokes equation

$$\mu \Delta \mathbf{u} - \nabla p = 0$$

2D Model

Method

3D Reality

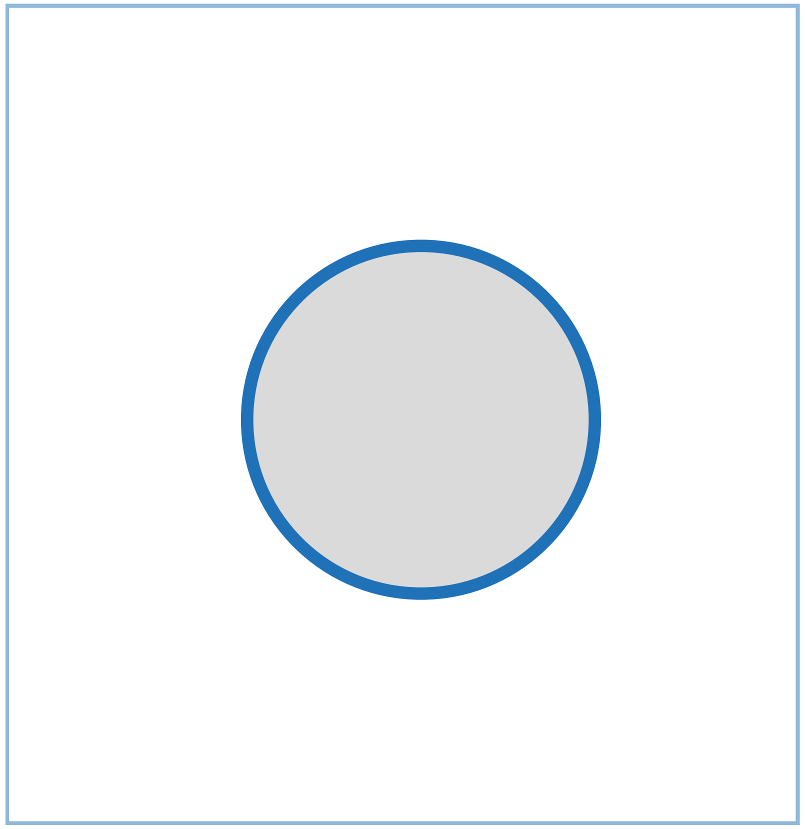


3D Stokes equation

$$\mu \Delta \mathbf{u} - \nabla p = 0$$

2D Brinkman equation

$$\mu \left(\Delta_{\shortparallel} \overline{\mathbf{u}} - \frac{12}{H^2} \overline{\mathbf{u}} \right) - \nabla_{\shortparallel} \overline{p} = 0$$

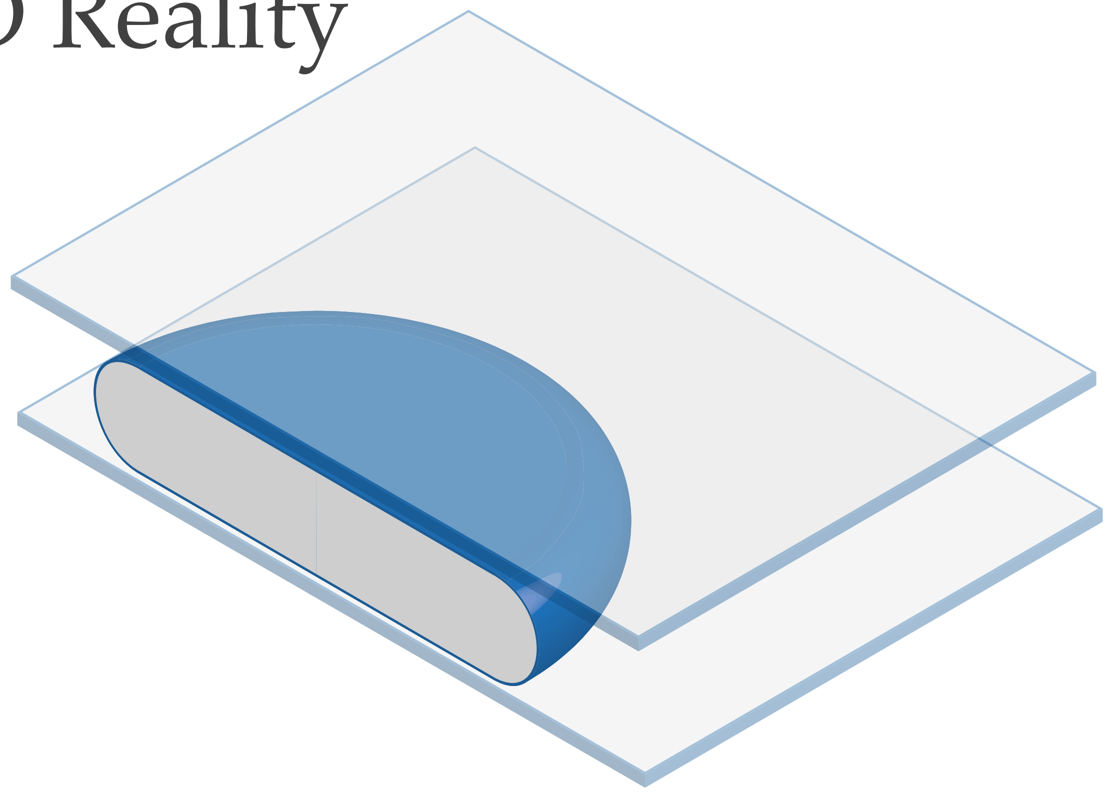


2D Model

Method

Boos and Thess, 1997
Gallaire, Laure, Baroud and Meliga, 2014

3D Reality



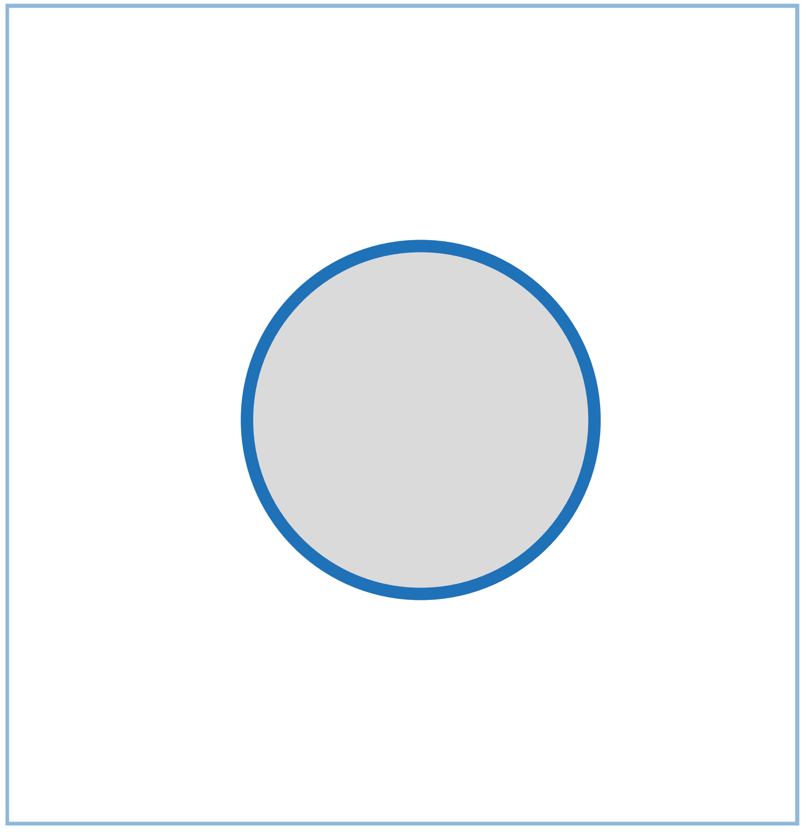
3D Stokes equation

$$\mu \Delta \mathbf{u} - \nabla p = 0$$

2D Brinkman equation

Darcy

$$\mu \left(\Delta_{\shortparallel} \overline{\mathbf{u}} - \boxed{\frac{12}{H^2} \overline{\mathbf{u}}} \right) - \nabla_{\shortparallel} \overline{p} = 0$$

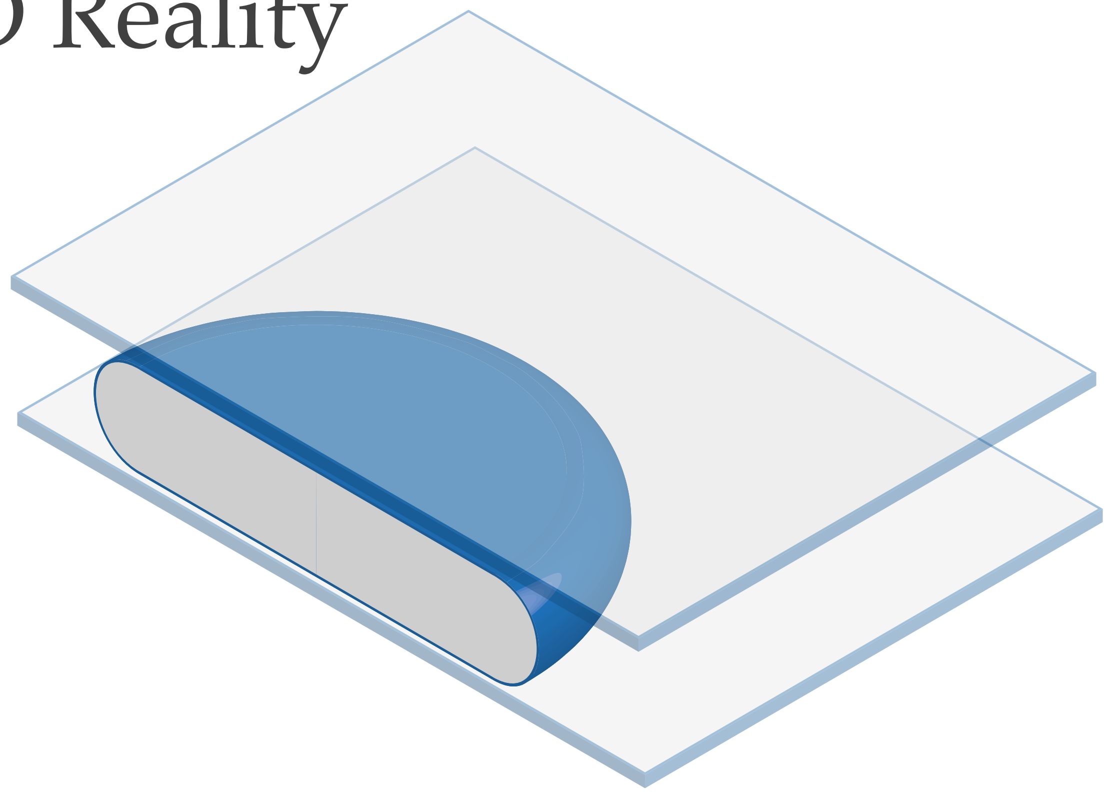


2D Model

Method

Boos and Thess, 1997
Gallaire, Laure, Baroud and Meliga, 2014

3D Reality



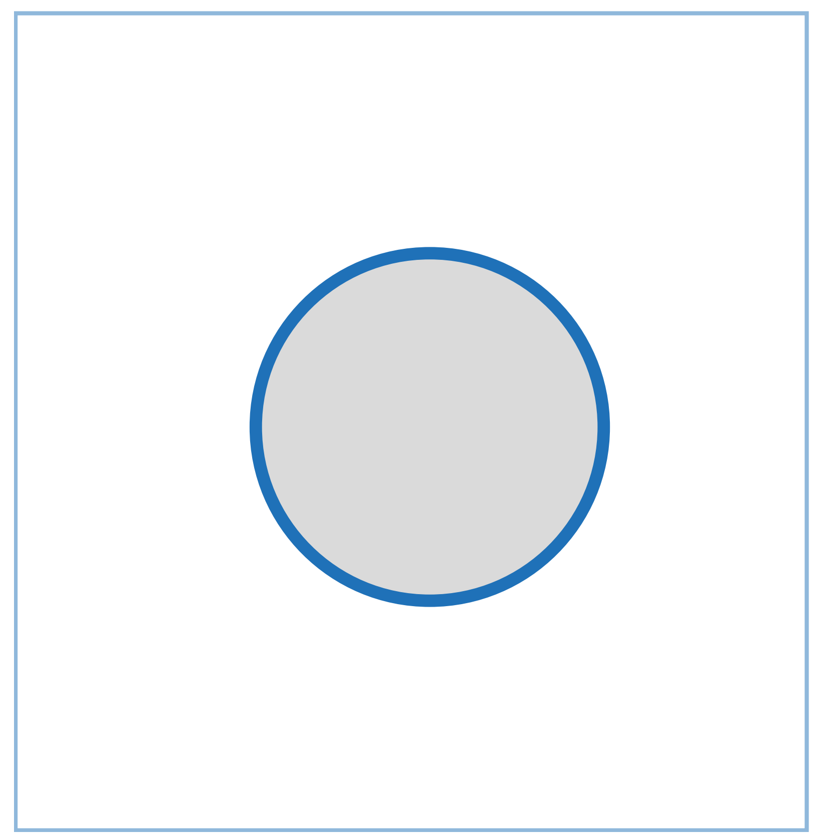
3D Stokes equation

$$\mu \Delta \mathbf{u} - \nabla p = 0$$

2D Brinkman equation

2D Stokes Darcy

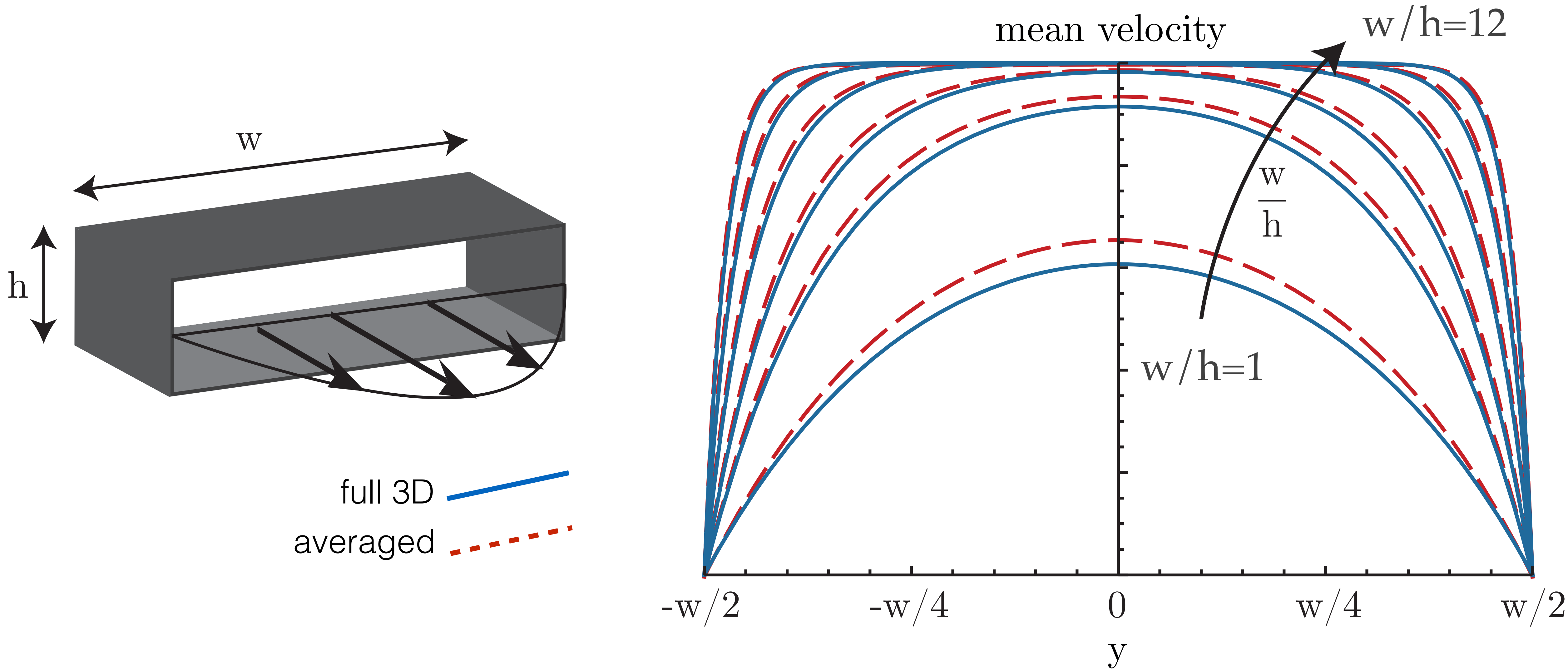
$$\mu \left(\boxed{\Delta_{\shortparallel} \overline{\mathbf{u}}} - \boxed{\frac{12}{H^2} \overline{\mathbf{u}}} \right) - \nabla_{\shortparallel} \overline{p} = 0$$



2D Model

Method

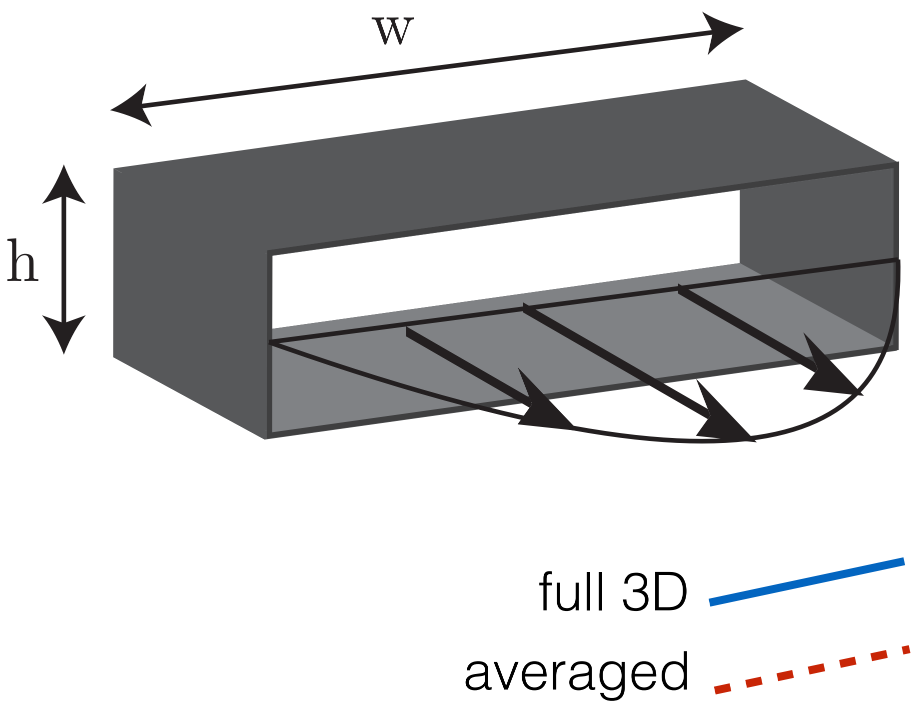
Boos and Thess, 1997
Gallaire, Laure, Baroud and Meliga, 2014

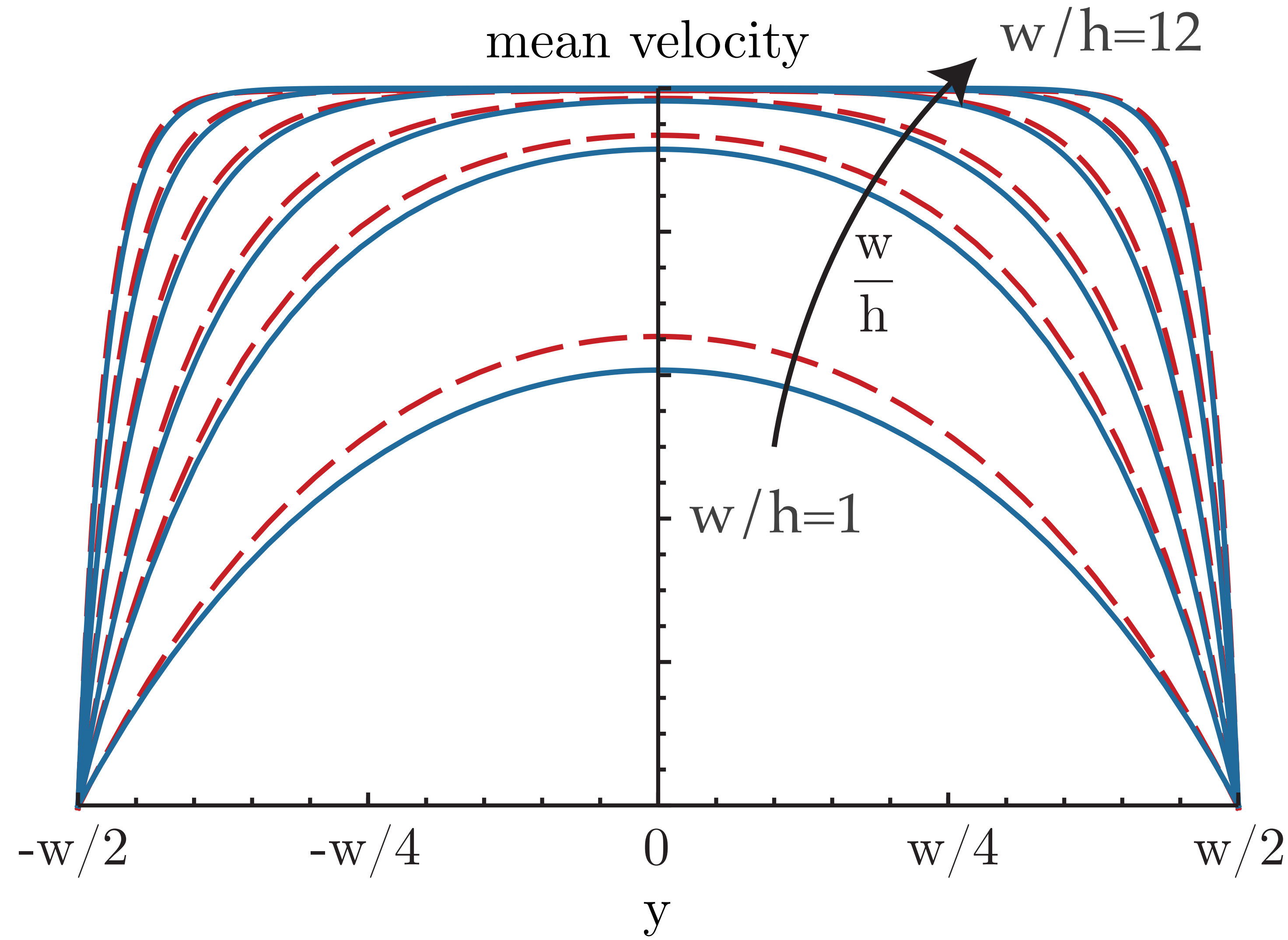


$$\mu\left(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial v_x}{\partial y^2}-k^2 v_x\right)-\frac{\partial p}{\partial x}=0$$

$$\mu\left(\frac{\partial^2 v_y}{\partial x^2}+\frac{\partial v_y}{\partial y^2}-k^2 v_y\right)-\frac{\partial p}{\partial y}=0$$

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}=0$$





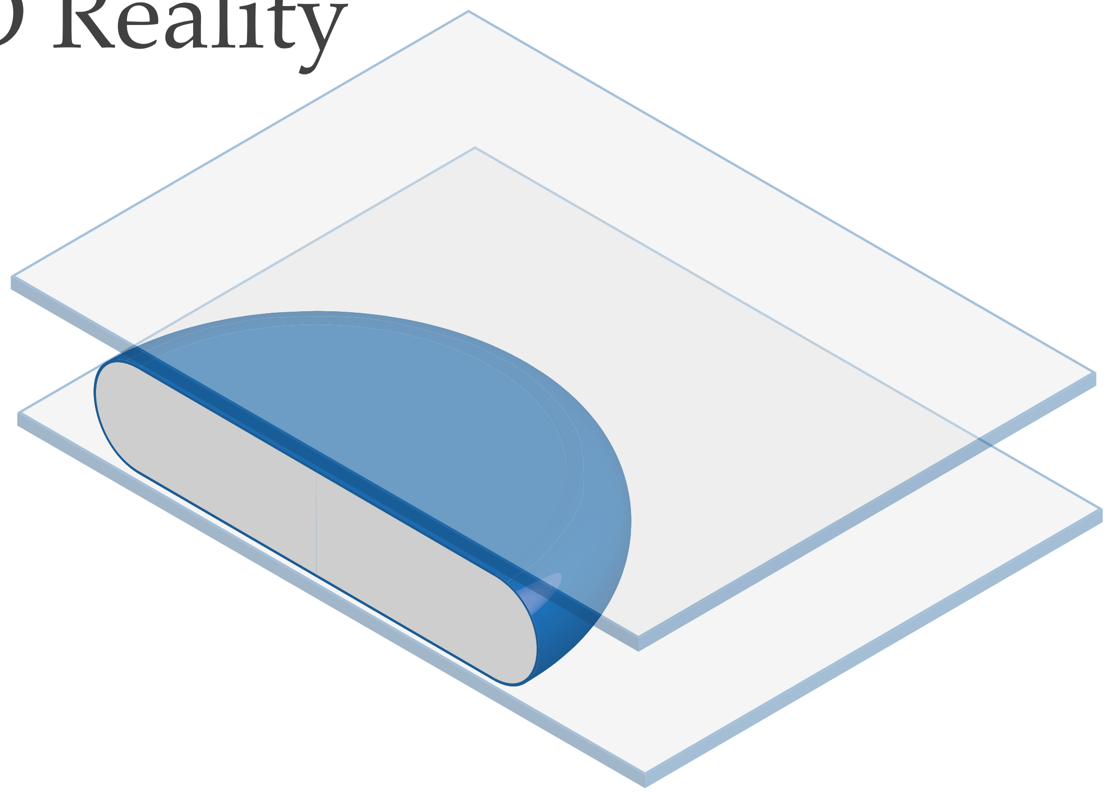
The solution of the averaged Stokes equation is not the averaged solution of the Stokes equation!

$$\mu\left(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial v_x}{\partial y^2}-k^2 v_x\right)-\frac{\partial p}{\partial x}=0$$

$$\mu\left(\frac{\partial^2 v_y}{\partial x^2}+\frac{\partial v_y}{\partial y^2}-k^2 v_y\right)-\frac{\partial p}{\partial y}=0$$

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}=0$$

3D Reality



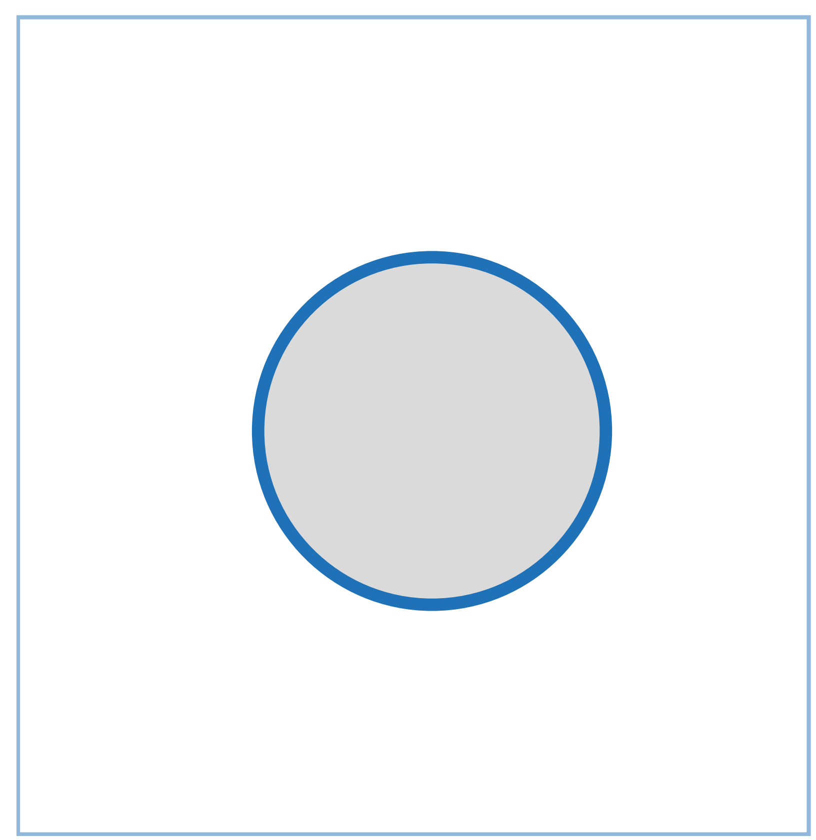
3D Stokes equation

$$\mu\Delta\mathbf{u} - \nabla p = 0$$

2D Brinkman equation

2D Stokes Darcy

$$\mu\left(\boxed{\Delta_{\parallel}\overline{\mathbf{u}}} - \boxed{\frac{12}{H^2}\overline{\mathbf{u}}}\right) - \nabla_{\parallel}\overline{p} = 0$$

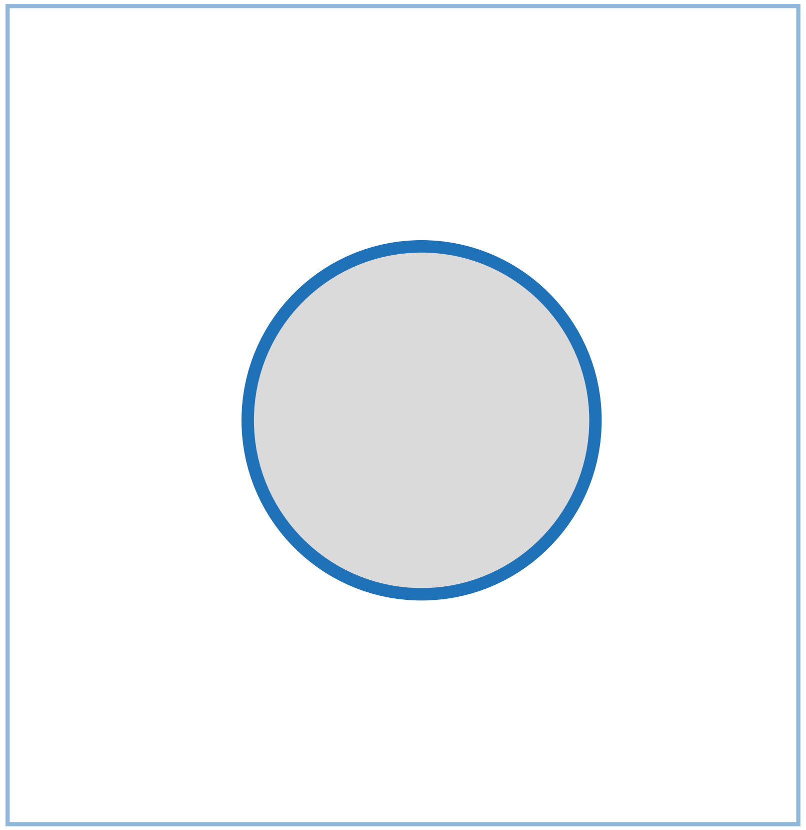


2D Model

Method

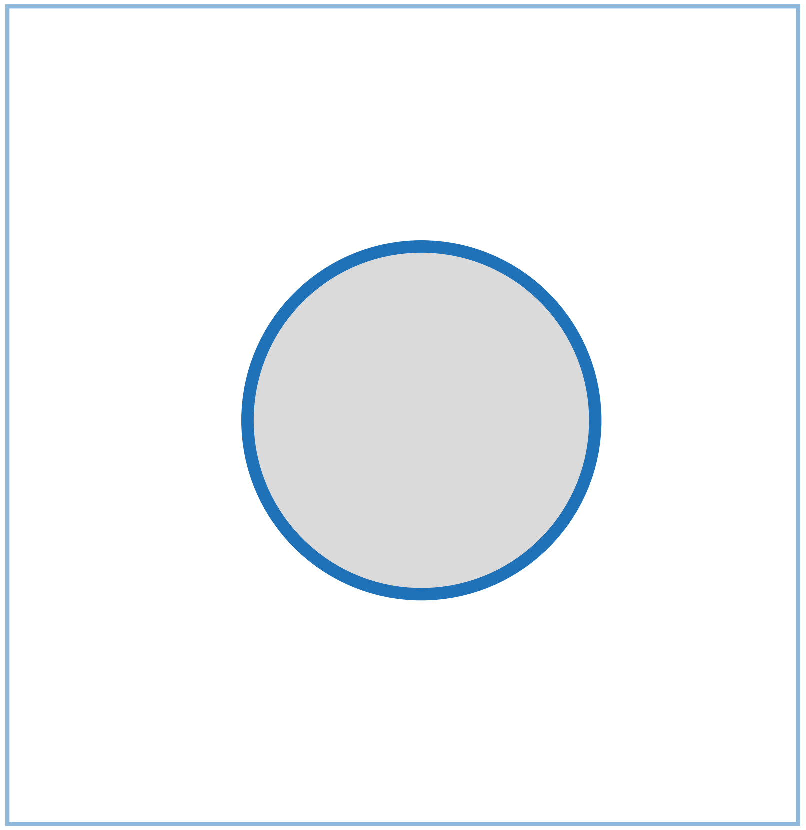
Boos and Thess, 1997
Gallaire, Laure, Baroud and Meliga, 2014

2D Model

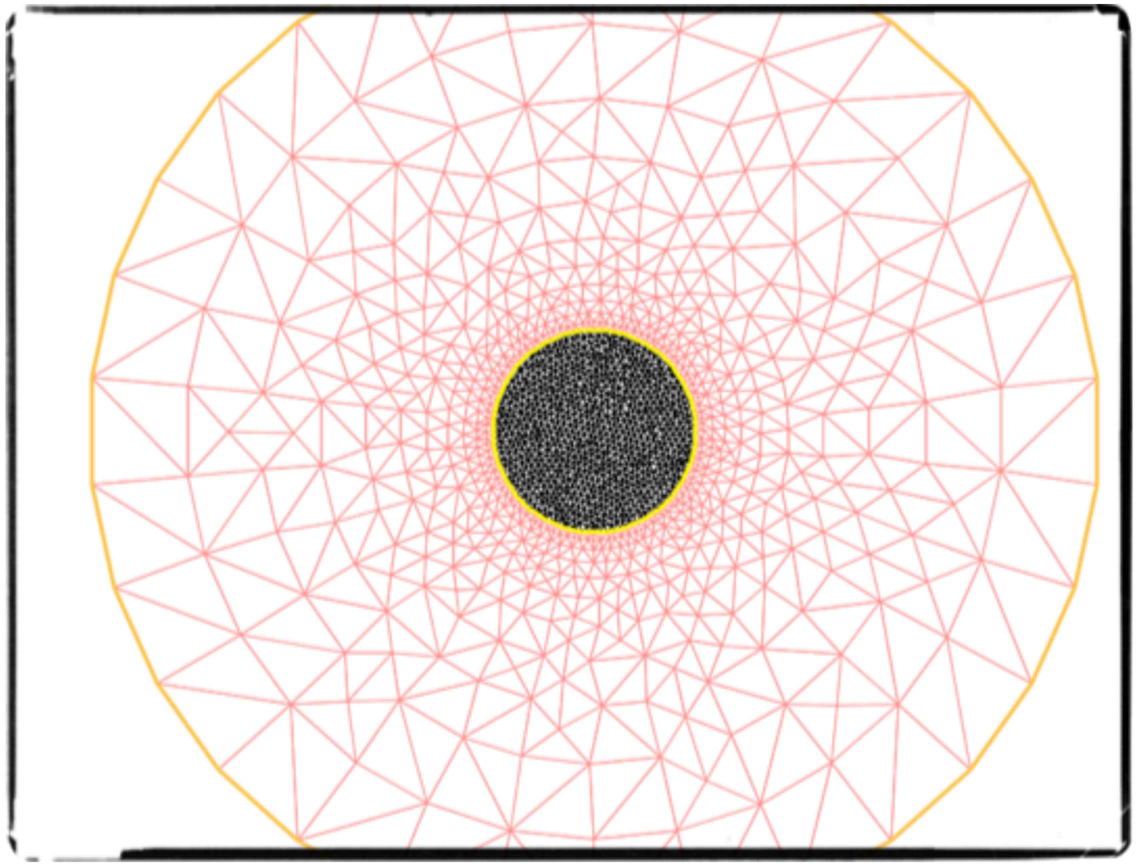


$$\mu\left(\Delta_{||}\overline{\mathbf{u}} - \frac{12}{H^2}\overline{\mathbf{u}}\right) - \nabla_{||}\overline{p} = 0$$

2D Model



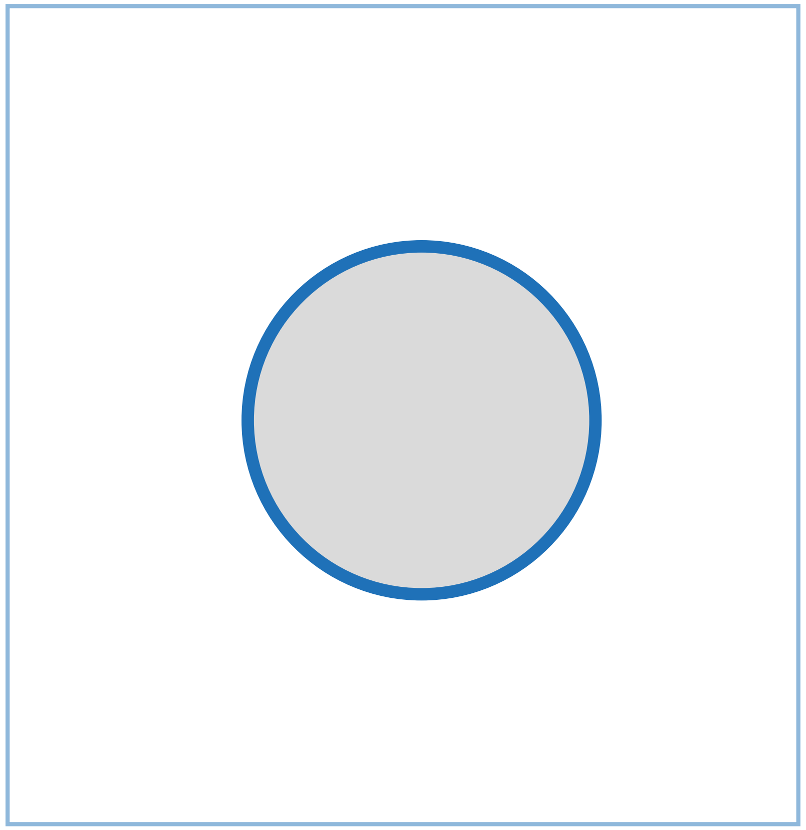
$$\mu \left(\Delta_{\shortparallel} \overline{\mathbf{u}} - \frac{12}{H^2} \overline{\mathbf{u}} \right) - \nabla_{\shortparallel} \overline{p} = 0$$



Method

2D Discretization

2D Model

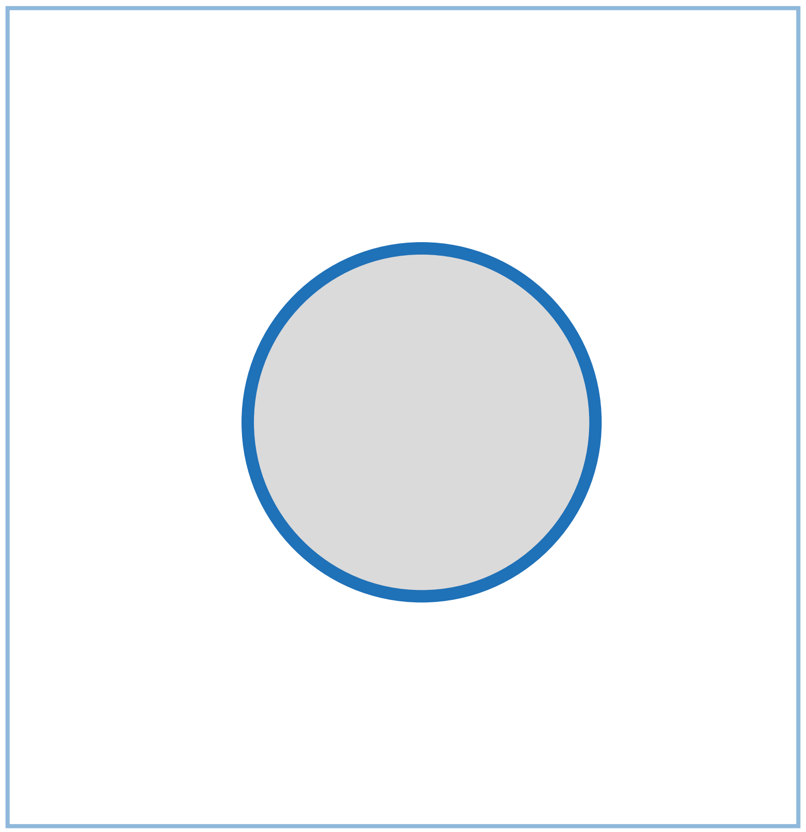


$$\mu\left(\Delta_{\shortparallel}\overline{\mathbf{u}} - \frac{12}{H^2}\overline{\mathbf{u}}\right) - \nabla_{\shortparallel}\overline{p} = 0$$

Method

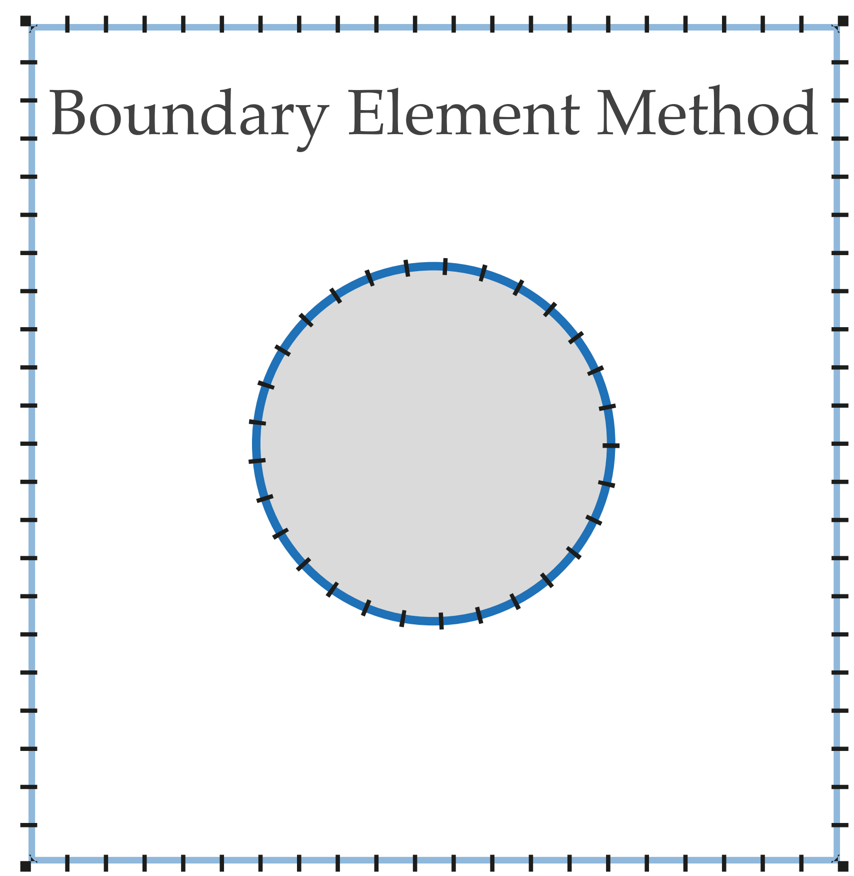
2D Discretization

2D Model



$$\mu \left(\Delta_{\shortparallel} \overline{\mathbf{u}} - \frac{12}{H^2} \overline{\mathbf{u}} \right) - \nabla_{\shortparallel} \overline{p} = 0$$

ULaMBAtoR

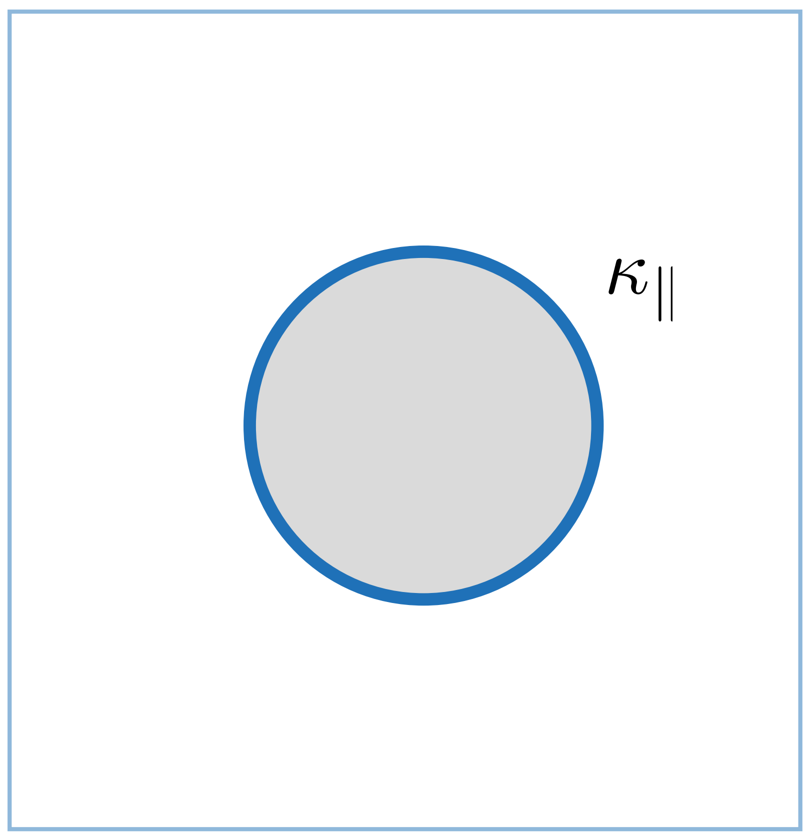


$$\oint \mathbf{G} \cdot \sigma \mathbf{n} - \mathbf{T}\mathbf{n} \cdot \mu \mathbf{u} \, ds = 2\pi\mu \mathbf{u}_0$$

Method

1D Discretization

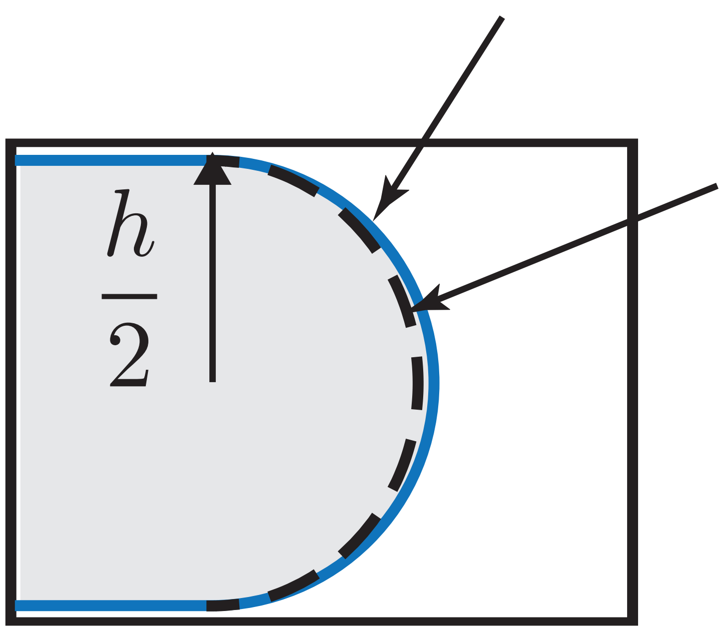
2D Model - Boundary condition



Torus shaped droplets have a curvature of:

$$\kappa = \frac{2}{h} + \frac{\pi}{4}\kappa_{||}$$

ideal torus shape



correct shape, since the in-plane curvature is not constant over h.

The π/4 correction
for toroidal droplets
is an asymptotic result
by Park and Homsy
JFM vol.139 (1984)

Understanding Boundary Elements

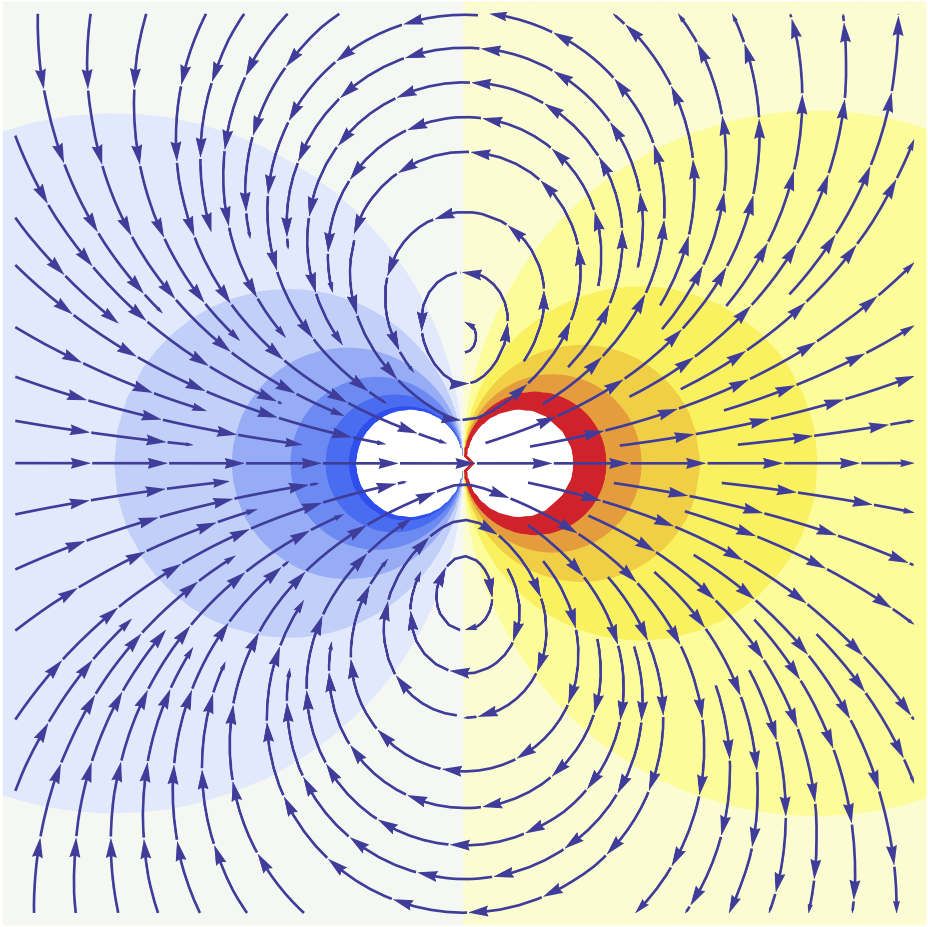
Two routes to understand the BEM …

- as a continuous distribution of fundamental solutions.
- as a finite element scheme with particular test functions.

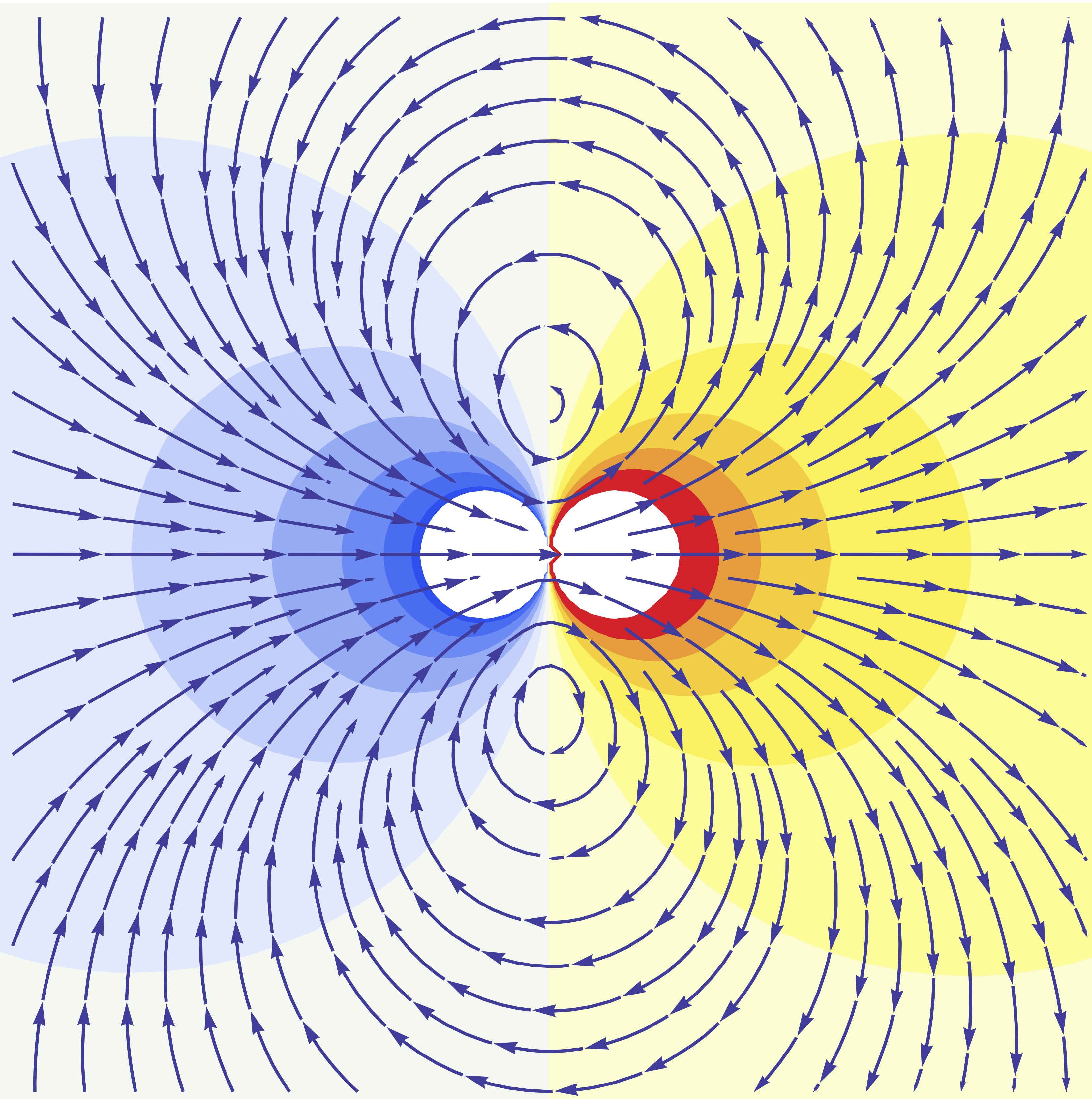
Understanding Boundary Elements

Two routes to understand the BEM …

- as a continuous distribution of fundamental solutions.
- as a finite element scheme with particular test functions.



Fundamental solution



The fundamental solution solves a Brinkman equation which is forced by a Dirac distribution at x_0

$$\eta(\nabla^2 \mathbf{u} - k^2 \mathbf{u}) - \nabla p =$$

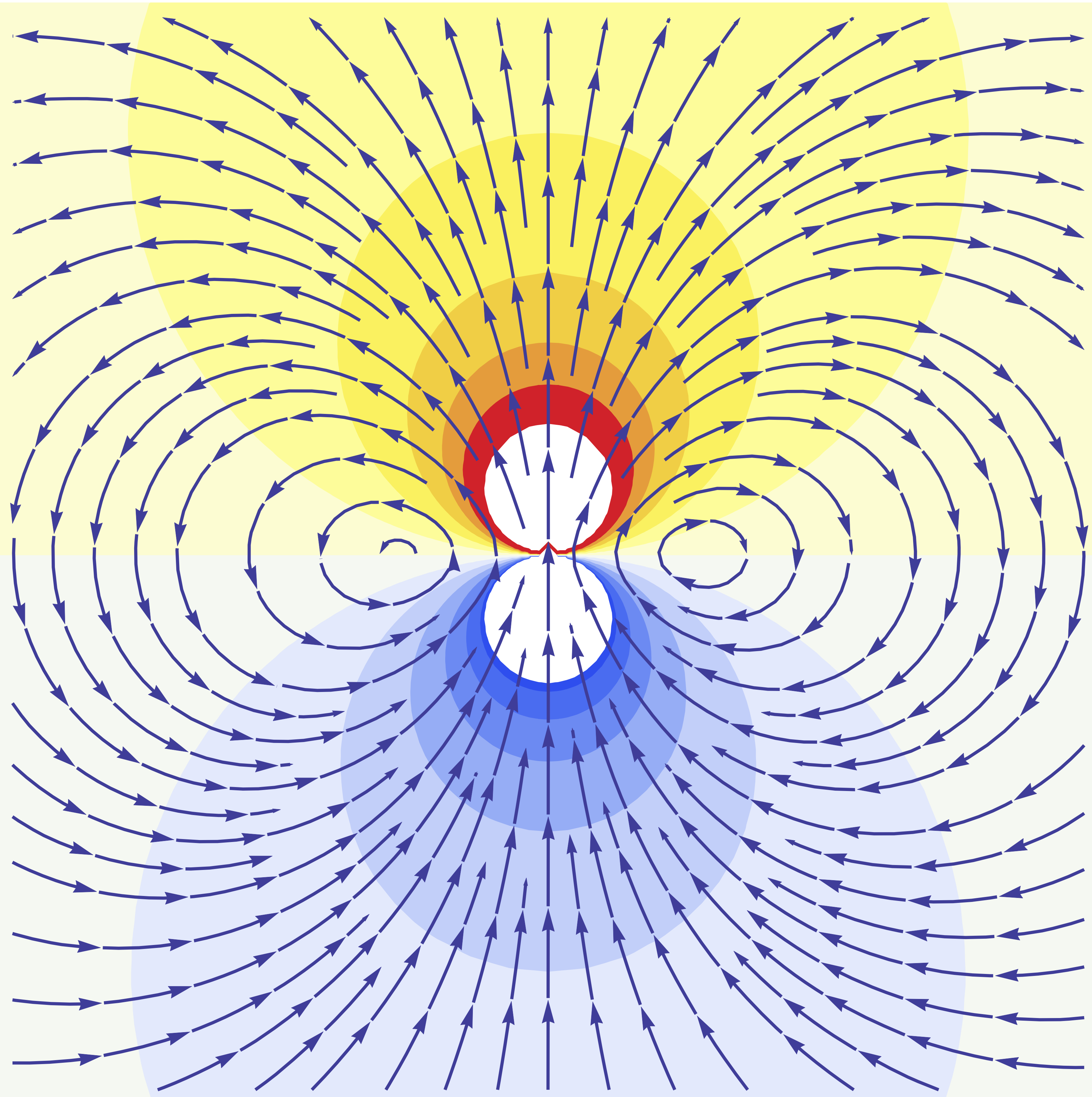
$$\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u} = \begin{pmatrix} \delta(\mathbf{x}_0) \\ 0 \end{pmatrix}$$

$$\bar{\bar{\sigma}} = \mathbf{T}_1(k, \mathbf{x}_0)$$

$$\mathbf{u} = \frac{1}{\eta} \mathbf{g_1}(k, \mathbf{x_0})$$

Nagel and Gallaire,
Computers and Fluids (2015)

Fundamental solution



The fundamental solution solves a Brinkman equation which is forced by a Dirac distribution at x_0

$$\eta(\nabla^2 \mathbf{u} - k^2 \mathbf{u}) - \nabla p = \nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u} = \begin{pmatrix} 0 \\ \delta(\mathbf{x}_0) \end{pmatrix}$$

$$\bar{\bar{\sigma}} = \mathbf{T}_2(k, \mathbf{x}_0)$$

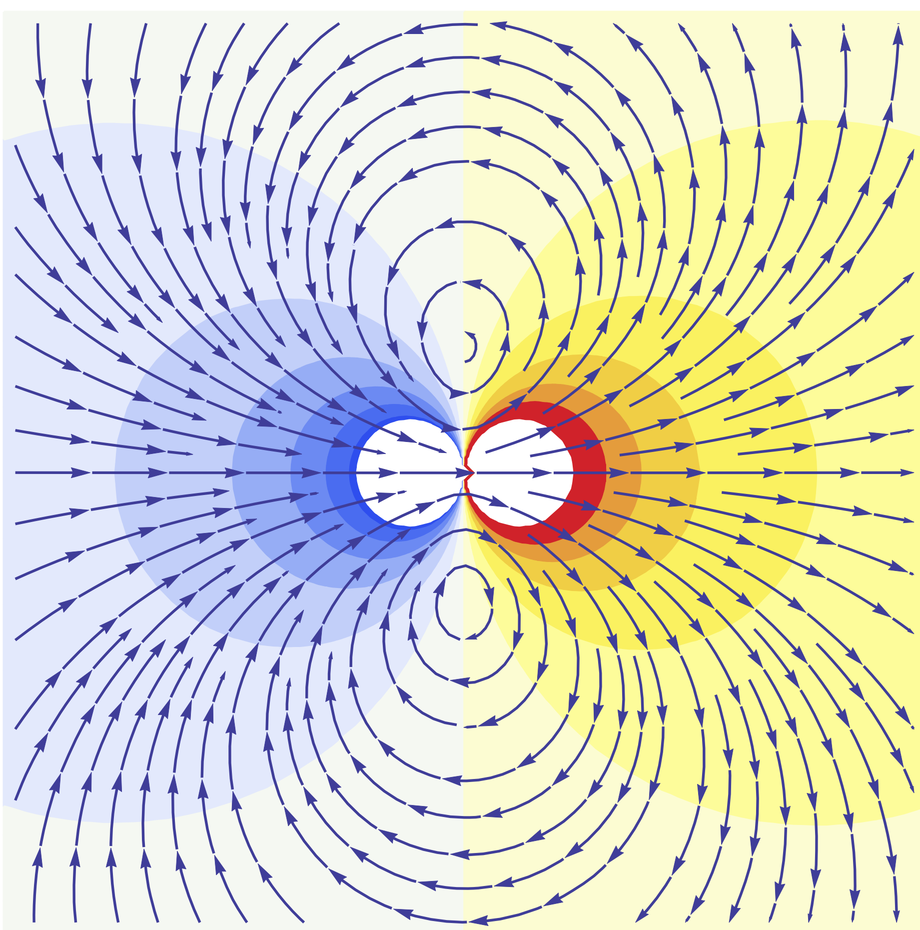
$$\mathbf{u} = \frac{1}{\eta}\mathbf{g_2}(k, \mathbf{x_0})$$

Nagel and Gallaire, Computers and Fluids (2015)

Understanding Boundary Elements

Two routes to understand the BEM …

- as a continuous distribution of fundamental solutions.
- as a finite element scheme with particular test functions.



Sketch

Understanding Boundary Elements

$$\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u} = \mathbf{0}$$

$$\int_{\Omega} (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = 0$$

Understanding Boundary Elements

- Brinkman equation is integrated over test functions:

$$\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u} = \mathbf{0}$$

$$\int\limits_{\Omega} (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = 0$$

Understanding Boundary Elements

- Brinkman equation is integrated over test functions:

$$\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u} = \mathbf{0} \qquad \int_\Omega (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = 0$$

- Like-wise for a Brinkman equation with test variables :

$$\nabla \cdot \mathbf{T_i} - \eta k^2 \mathbf{g_i} = \bar{\delta}(\mathbf{x}_0) \qquad \int_\Omega \big(\nabla \cdot \mathbf{T}_i - \eta k^2 \mathbf{g}_i\big) \cdot \mathbf{u} \, dA = 2\pi u_i(\mathbf{x_0})$$

Understanding Boundary Elements

- Brinkman equation is integrated over test functions:

$$\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u} = \mathbf{0} \qquad \int_{\Omega} (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = 0$$

- Like-wise for a Brinkman equation with test variables :

$$\nabla \cdot \mathbf{T_i} - \eta k^2 \mathbf{g_i} = \bar{\delta}(\mathbf{x}_0) \qquad \int_{\Omega} \left(\nabla \cdot \mathbf{T}_i - \eta k^2 \mathbf{g}_i\right) \cdot \mathbf{u} \, dA = 2\pi u_i(\mathbf{x_0})$$

- Then one applies integration by parts

$$\int_{x_0}^{x_1} u'v \, dx = uv\Big|_{x_0}^{x_1} - \int_{x_0}^{x_1} uv' dx$$

$$0 = \int_{\Omega} (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = \oint_{\omega} \bar{\bar{\sigma}} \mathbf{n} ds$$

$$- \int_{\Omega} (\eta \nabla \mathbf{u} : \nabla \mathbf{g}_i - p \nabla \cdot \mathbf{g}_i + \eta k^2 \mathbf{u} \cdot \mathbf{g}_i) \, dA$$

Understanding Boundary Elements

$$0 = \int\limits_{\Omega} (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = \oint_{\omega} \bar{\bar{\sigma}} \mathbf{n} ds$$

$$- \int\limits_{\Omega} (\eta \nabla \mathbf{u} : \nabla \mathbf{g}_i - p \nabla \cdot \mathbf{g}_i + \eta k^2 \mathbf{u} \cdot \mathbf{g}_i) \, dA$$

$$\int\limits_{\Omega} \left(\nabla \cdot \mathbf{T}_i - \eta k^2 \mathbf{g}_i \right) \cdot \mathbf{u} \, dA = 2\pi u_i(\mathbf{x_0})$$

$$2\pi \eta u_i(\mathbf{x}_0) = \eta \int\limits_{\Omega} (\nabla \cdot \mathbf{T}_i - k^2 \mathbf{g}_i) \cdot \mathbf{u} \, dA = \oint_{\omega} \eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} \, ds$$

$$- \int\limits_{\Omega} (\eta \nabla \mathbf{u} : \nabla \mathbf{g}_i - \eta \, q \nabla \cdot \mathbf{u} + \eta k^2 \mathbf{u} \cdot \mathbf{g}_i) \, dA$$

Understanding Boundary Elements

$$0 = \int\limits_{\Omega} (\nabla \cdot \bar{\bar{\sigma}} - \eta k^2 \mathbf{u}) \cdot \mathbf{g}_i \, dA = \oint_{\omega} \bar{\bar{\sigma}} \mathbf{n} ds$$

$$- \int\limits_{\Omega} (\eta \nabla \mathbf{u} : \nabla \mathbf{g}_i - p \nabla \cdot \mathbf{g}_i + \eta k^2 \mathbf{u} \cdot \mathbf{g}_i) \, dA$$

$$2\pi\eta u_i(\mathbf{x}_0) = \eta \int\limits_{\Omega} (\nabla \cdot \mathbf{T}_i - k^2 \mathbf{g}_i) \cdot \mathbf{u} \, dA = \oint_{\omega} \eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} \, ds$$

$$- \int\limits_{\Omega} (\eta \nabla \mathbf{u} : \nabla \mathbf{g}_i - \eta \, q \nabla \cdot \mathbf{u} + \eta k^2 \mathbf{u} \cdot \mathbf{g}_i) \, dA$$

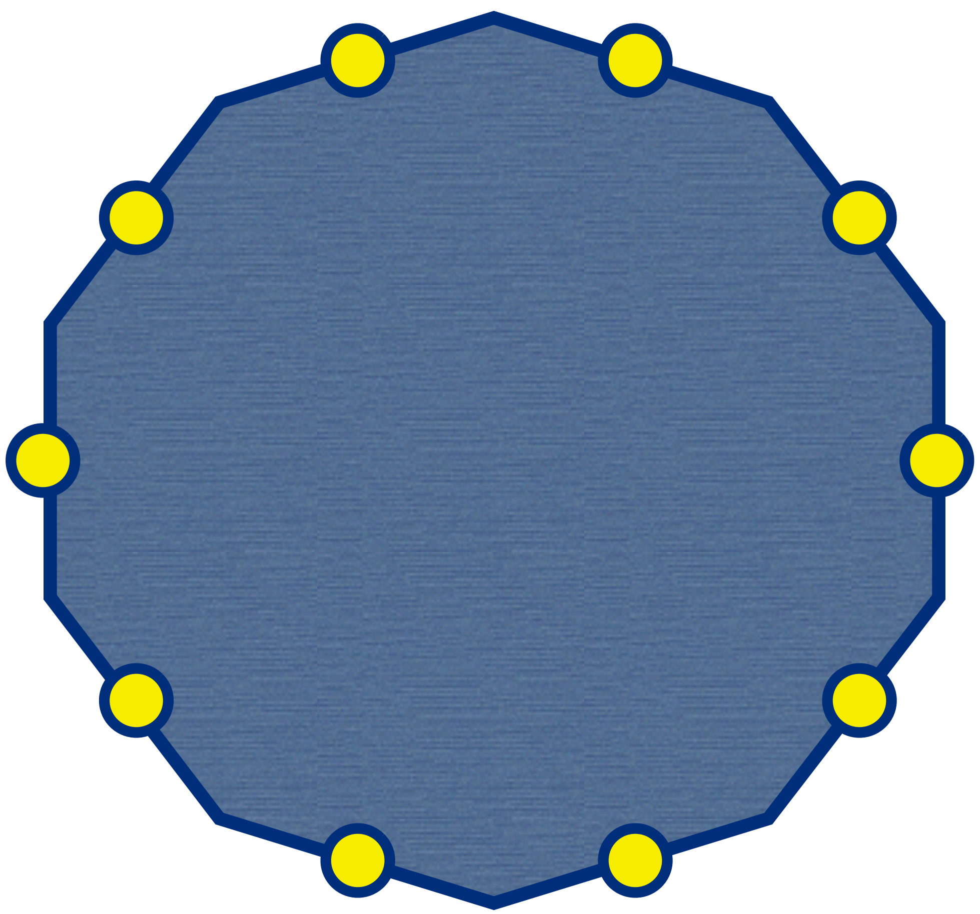
$$2\pi\eta u_i(\mathbf{x_0}) = \oint_{\omega} (\eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} - \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i) \, ds$$

Boundary Element Method

- N elements:

  2 x N unknowns
  2 x N equations

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_\omega (\eta \mathbf{T}_i \mathbf{n}\cdot\mathbf{u} - \bar{\bar{\sigma}}\mathbf{n}\cdot\mathbf{g}_i)\, ds$$



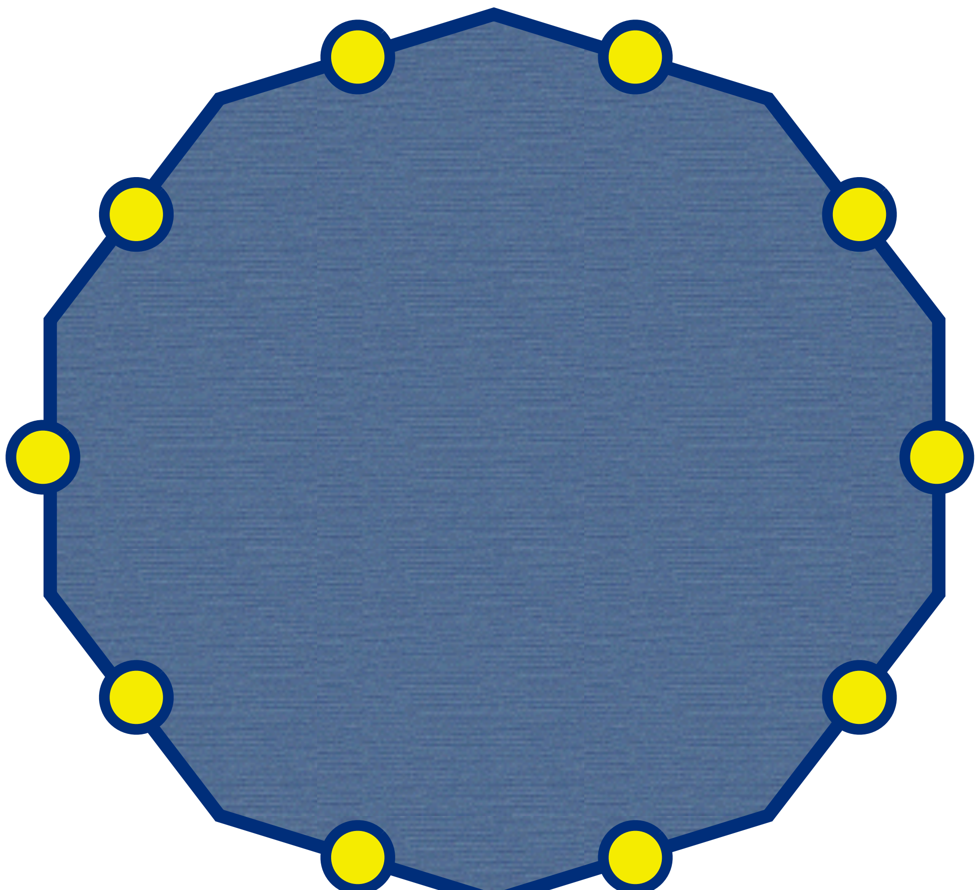
A model boundary element

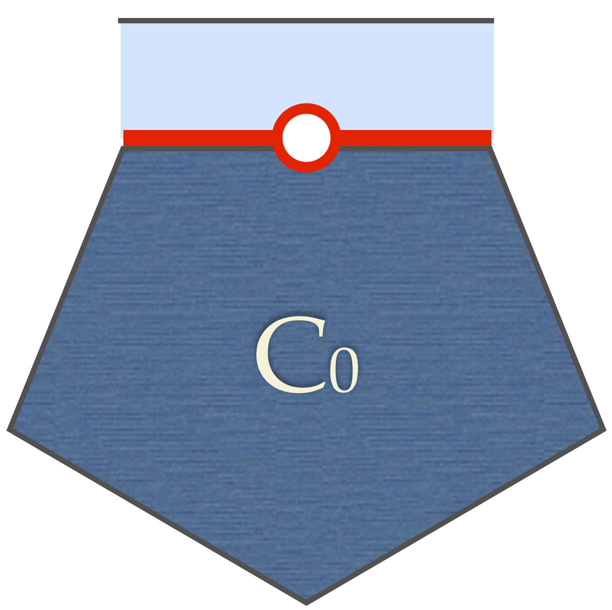
Boundary Element Method

- N elements:

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_{\omega} (\eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} - \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i)\, ds$$

2 x N unknowns
2 x N equations





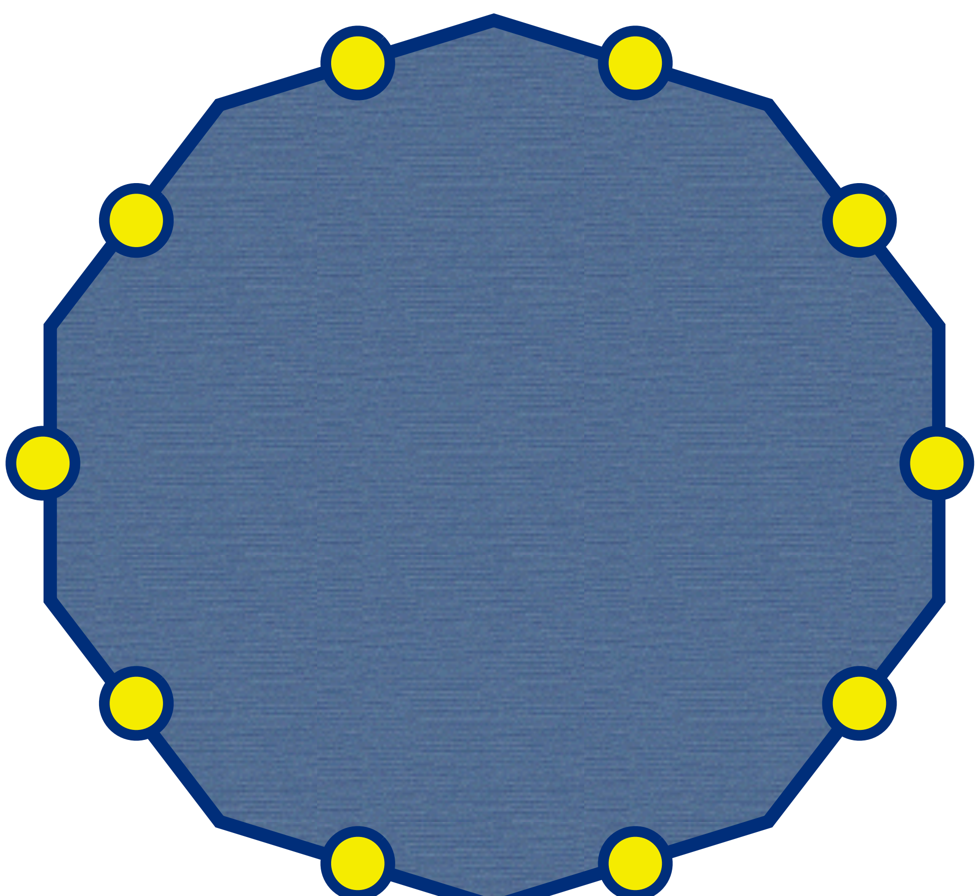
A model boundary element

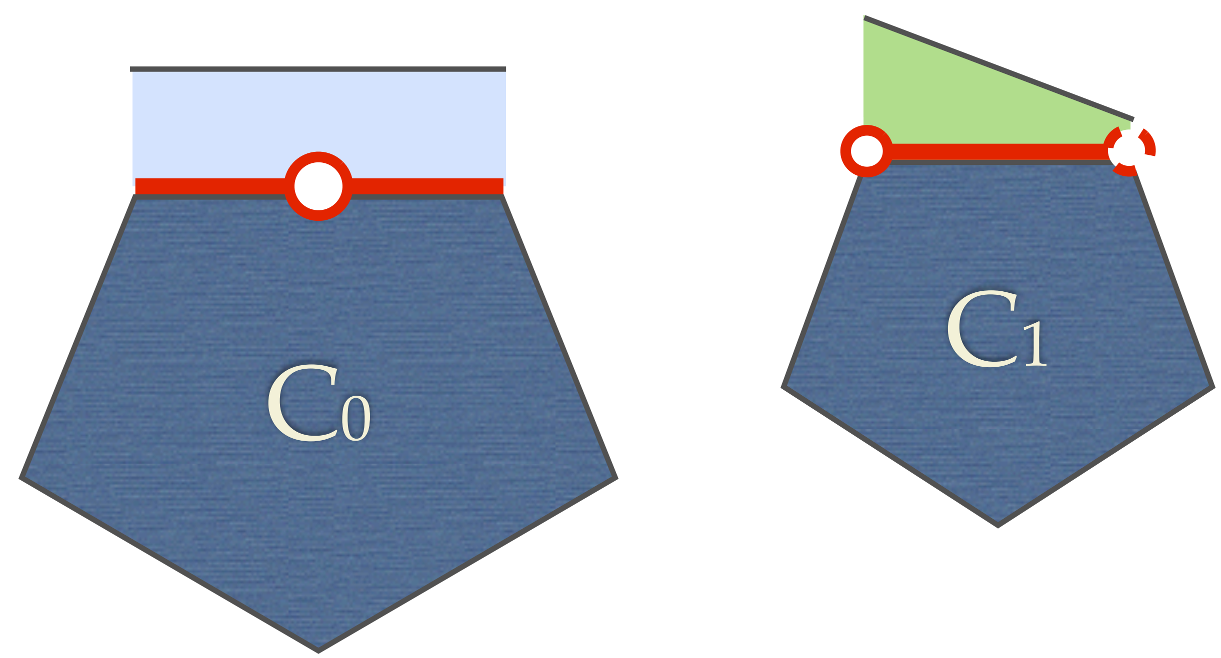
Boundary Element Method

- N elements:

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_\omega (\eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} - \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i)\, ds$$

2 x N unknowns
2 x N equations





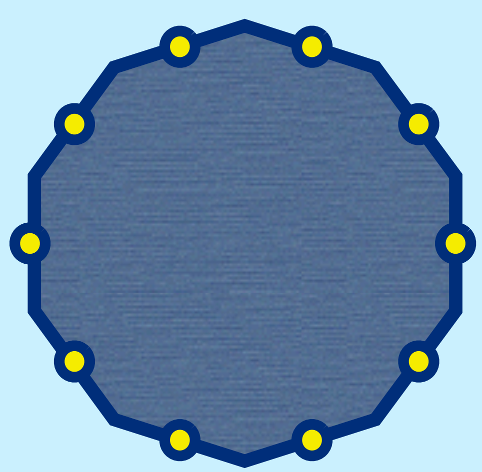
A model boundary element

Boundary Element Method

- Combining outer phase integral on the droplet

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_{\omega} \left(\eta \mathbf{T}_i \mathbf{n}\cdot\mathbf{u} - \bar{\bar{\sigma}}\mathbf{n}\cdot\mathbf{g}_i\right) ds$$

- with the inner phase integral (normal sign flipped).



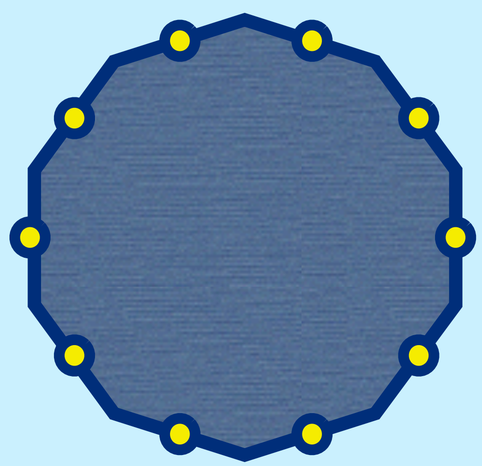
Boundary Element Method

- Combining outer phase integral on the droplet

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_{\omega} (\eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} - \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i)\, ds$$

- with the inner phase integral (normal sign flipped).

$$2\pi\eta_d u_i(\mathbf{x}_0) = \oint_{\omega} (-\eta_d \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} + \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i)\, ds$$



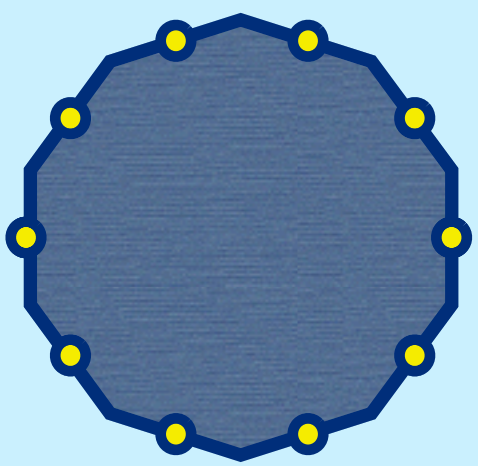
Boundary Element Method

- Combining outer phase integral on the droplet

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_\omega (\eta \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} - \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i)\, ds$$

- with the inner phase integral (normal sign flipped).

$$2\pi\eta_d u_i(\mathbf{x}_0) = \oint_\omega (-\eta_d \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} + \bar{\bar{\sigma}} \mathbf{n} \cdot \mathbf{g}_i)\, ds$$



$$2\pi(\eta + \eta_d) u_i(\mathbf{x}_0) = \oint_\omega \Big((\eta - \eta_d) \mathbf{T}_i \mathbf{n} \cdot \mathbf{u} - [\![\bar{\bar{\sigma}} \mathbf{n}]\!] \cdot \mathbf{g}_i \Big)\, ds$$

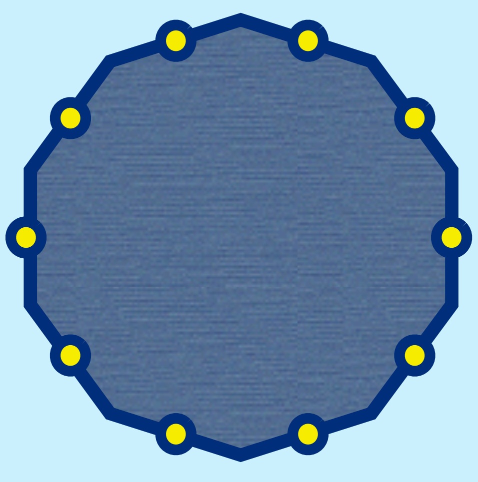
Boundary Element Method

- Combining outer phase integral on the droplet

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_\omega (\eta \mathbf{T}_i \mathbf{n}\cdot\mathbf{u} - \bar{\bar{\sigma}}\mathbf{n}\cdot\mathbf{g}_i)\, ds$$

- with the inner phase integral (normal sign flipped).

$$2\pi\eta_d u_i(\mathbf{x}_0) = \oint_\omega (-\eta_d \mathbf{T}_i \mathbf{n}\cdot\mathbf{u} + \bar{\bar{\sigma}}\mathbf{n}\cdot\mathbf{g}_i)\, ds$$

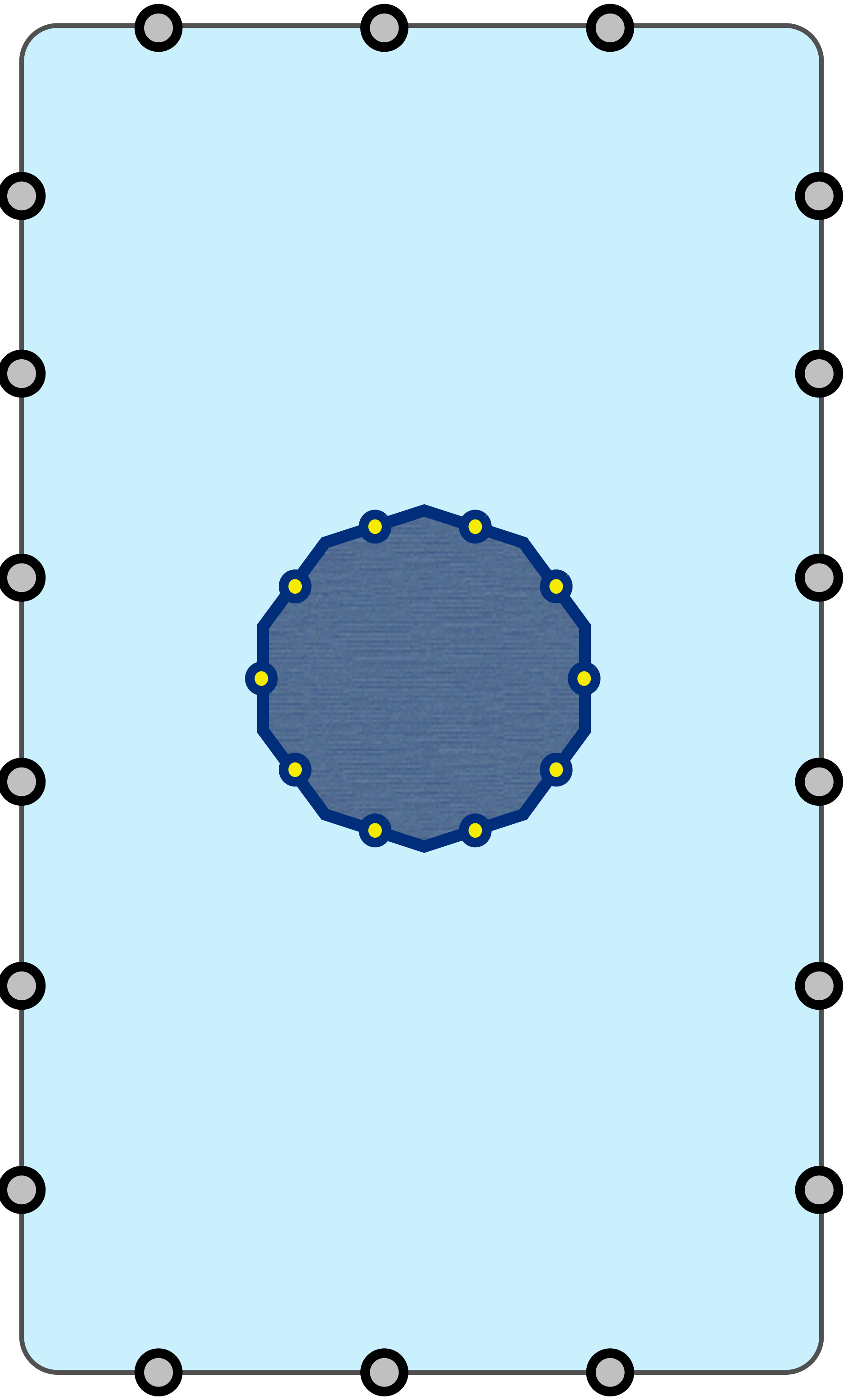


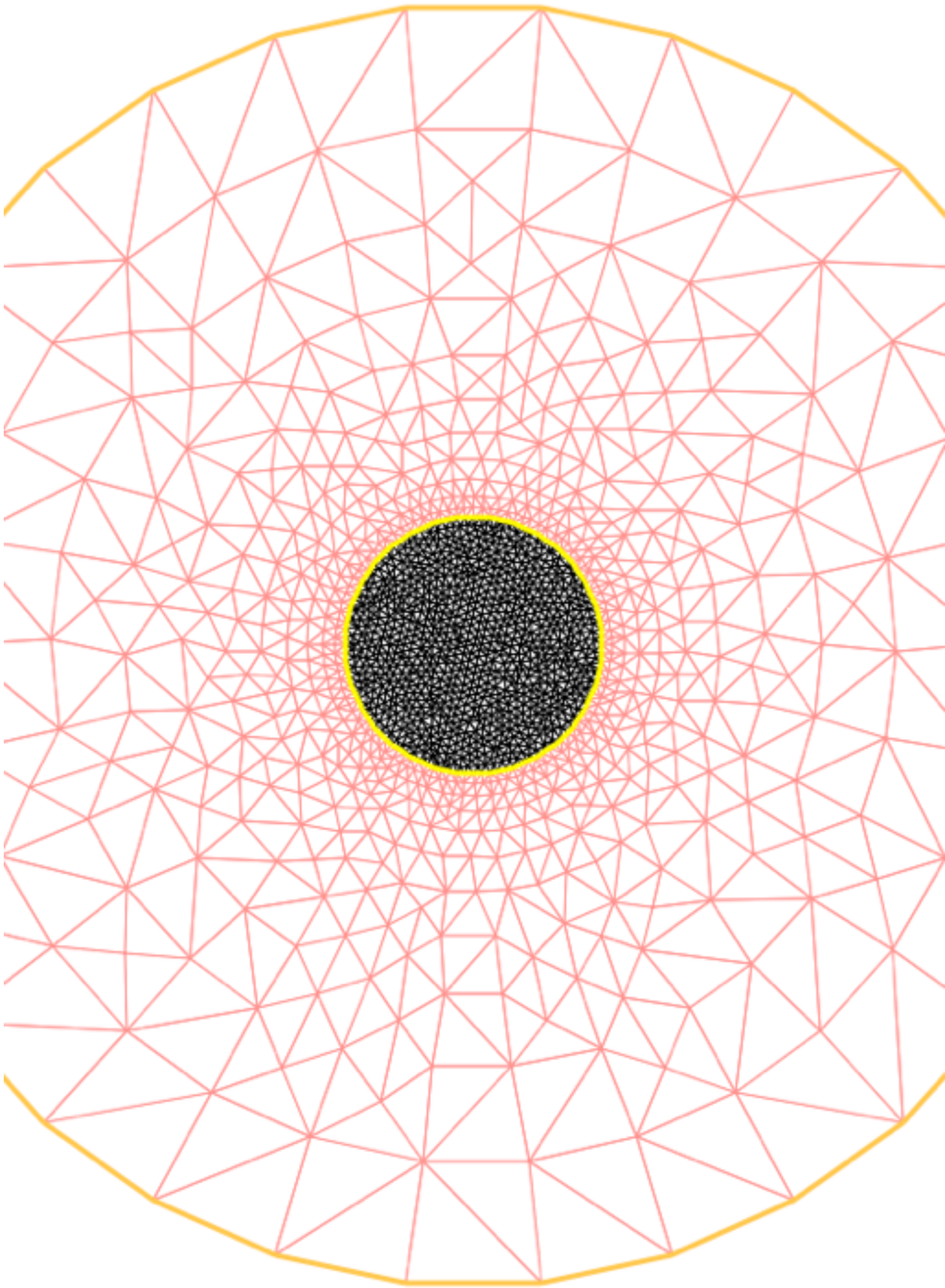
interface stress jump

viscosity difference

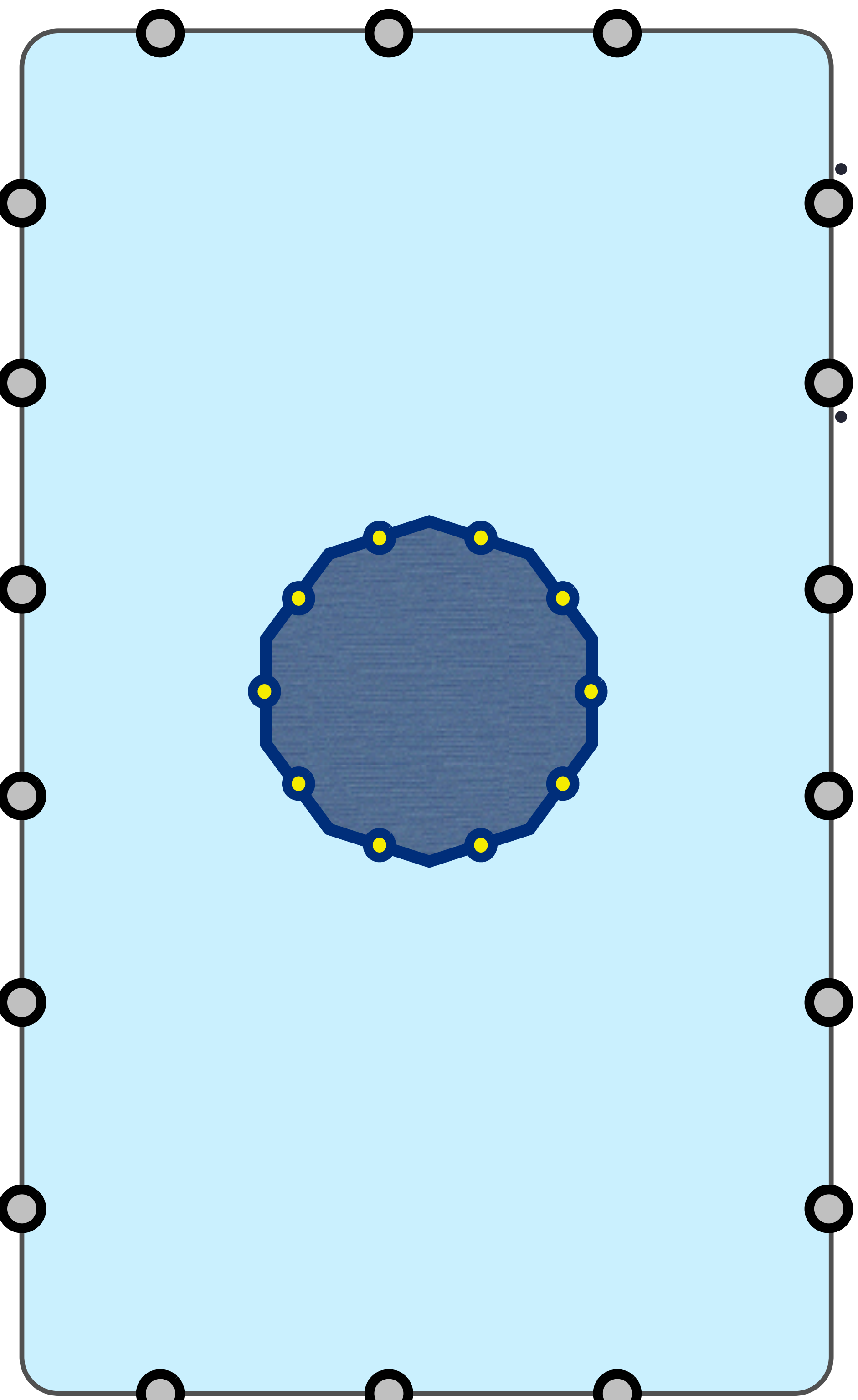
$$2\pi(\eta+\eta_d)u_i(\mathbf{x}_0) = \oint_\omega \Big((\eta-\eta_d)\mathbf{T}_i\mathbf{n}\cdot\mathbf{u} - [\![\bar{\bar{\sigma}}\mathbf{n}]\!]\cdot\mathbf{g}_i\Big)\, ds$$

Boundary Element Method



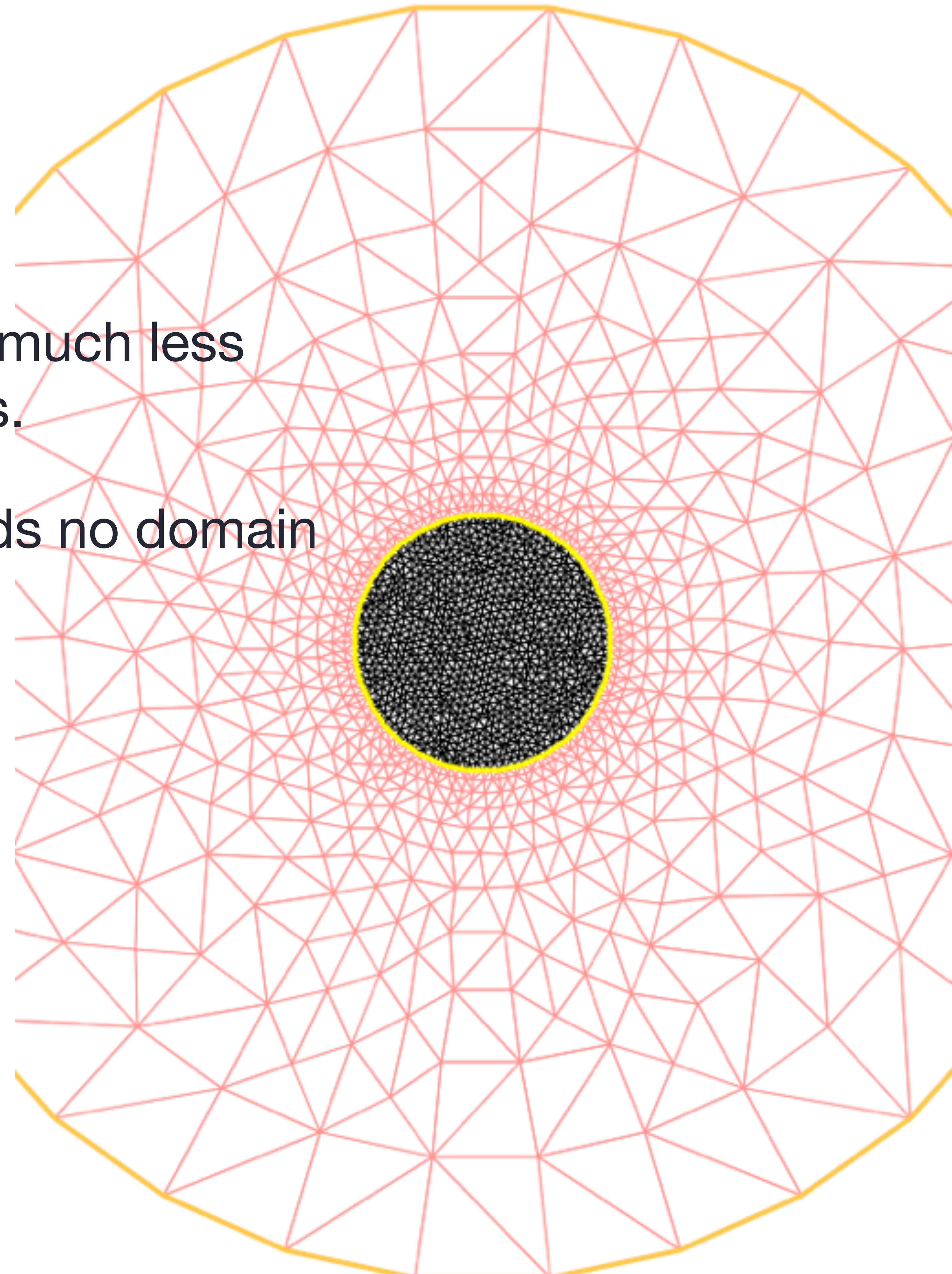


Boundary Element Method

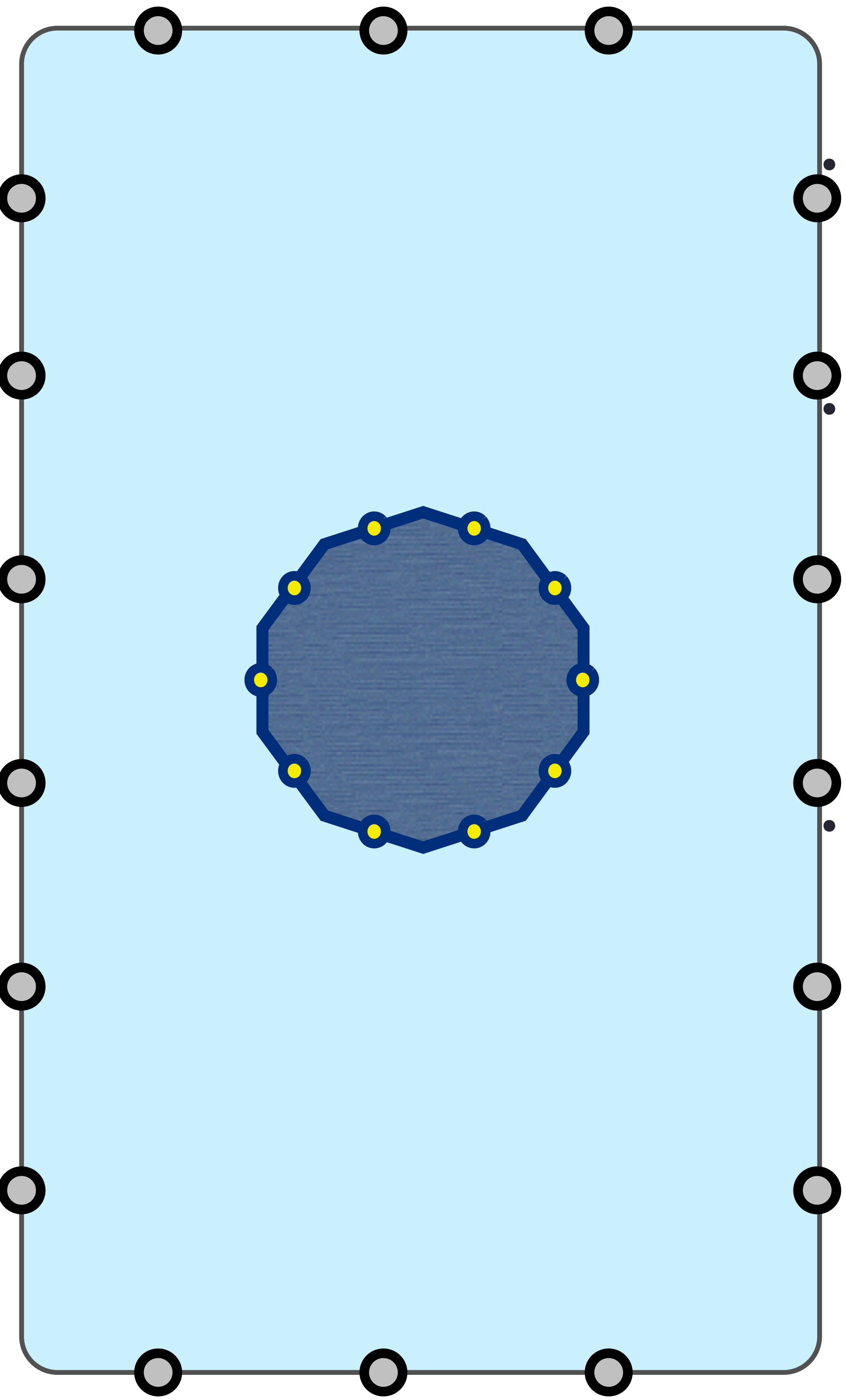


- BEM has much less unknowns.
- BEM needs no domain mesh.

But...



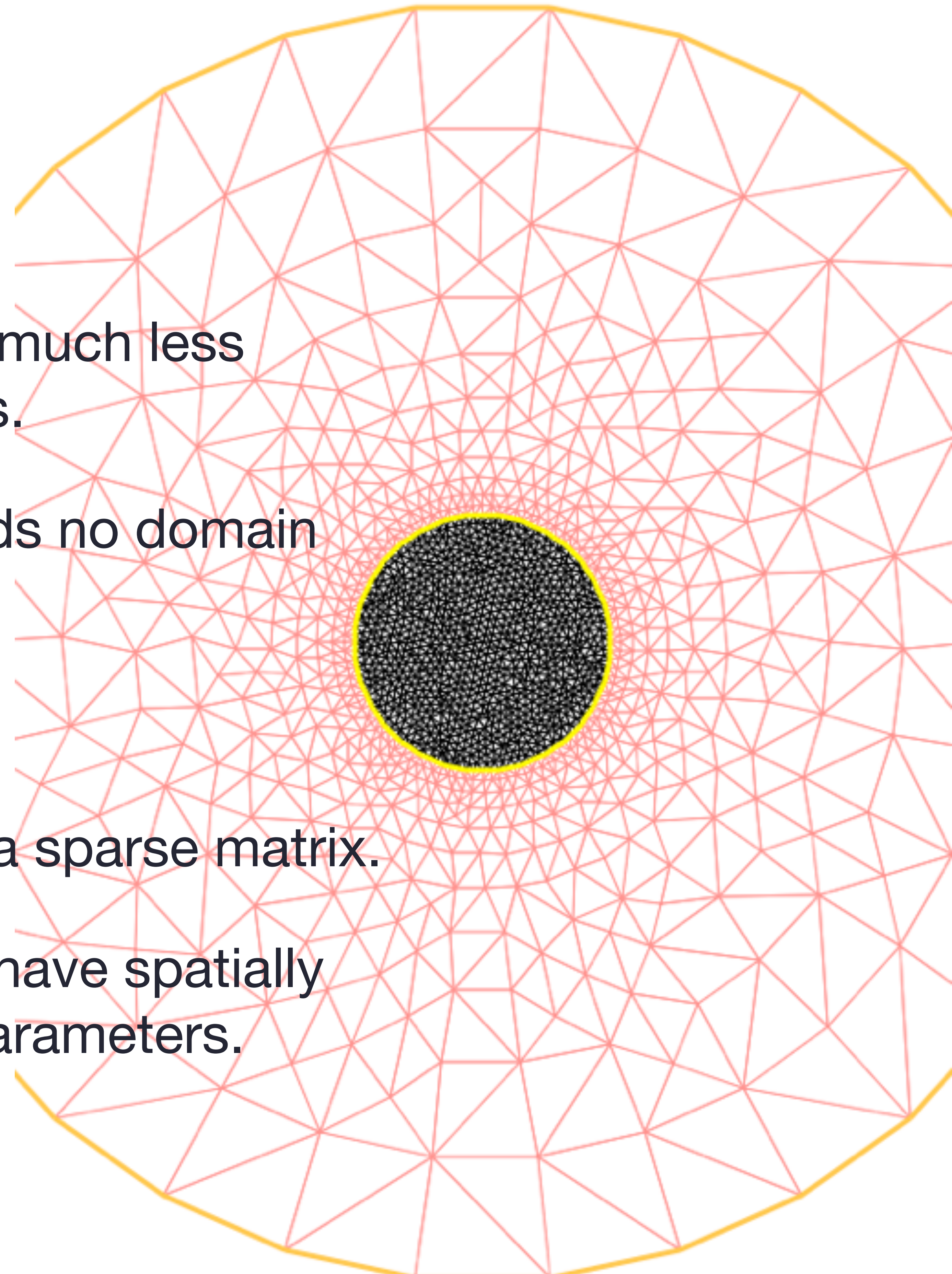
Boundary Element Method



- BEM has much less unknowns.
- BEM needs no domain mesh.

But...

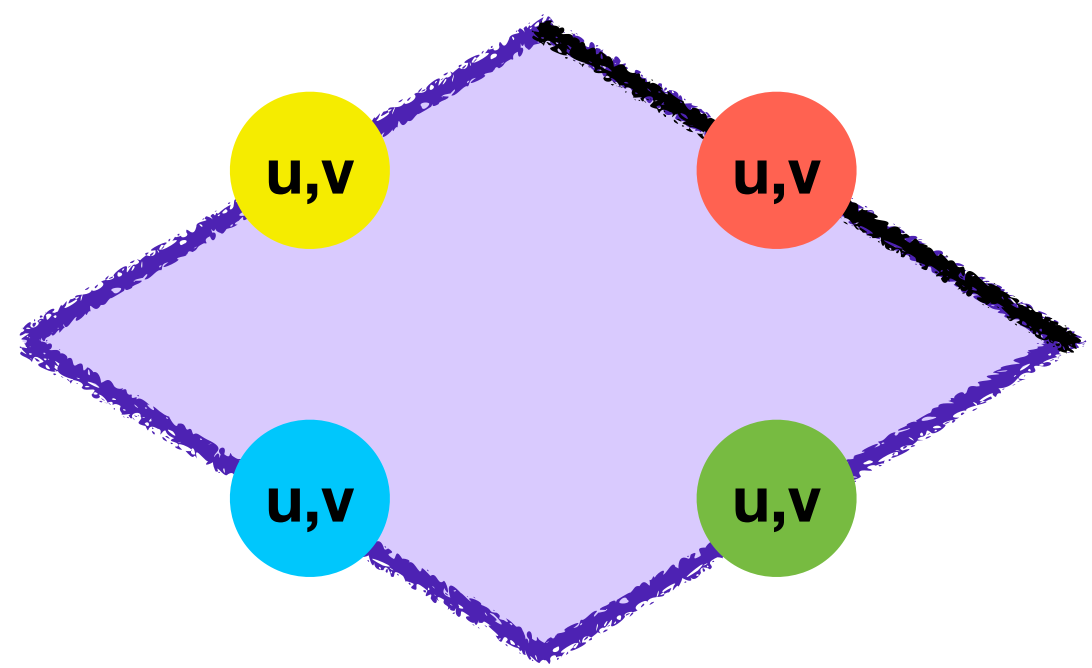
- FEM has a sparse matrix.
- FEM can have spatially varying parameters.



Computing Boundary Elements

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_\omega \left(\eta \mathbf{T}\,\mathbf{n}\cdot\mathbf{u} - \bar{\bar{\sigma}}\mathbf{n}\cdot\mathbf{g}\right) ds$$

- Discretizing the boundary with Ansatz functions
- For each boundary 2 unknowns, but for each test function 2 equations.



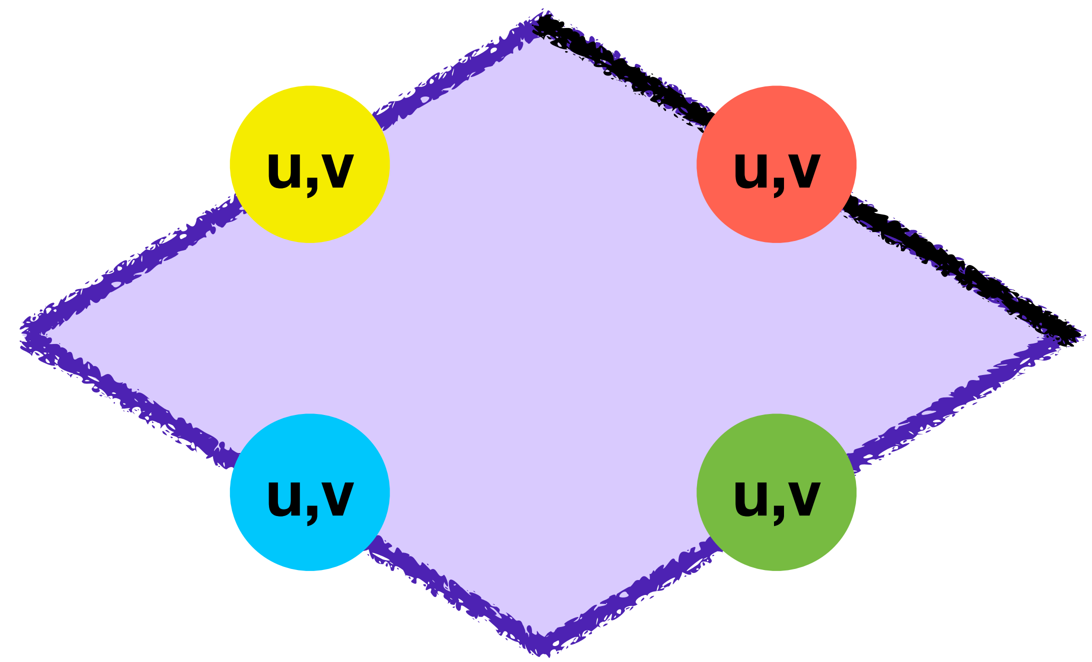
Contribution from black panel

$$\eta \begin{pmatrix} \oint_1 T_1\mathbf{n} - 2\pi & \oint_2 T_1\mathbf{n} & \oint_3 T_1\mathbf{n} & \oint_4 T_1\mathbf{n} \\ \oint_1 T_2\mathbf{n} & \oint_2 T_2\mathbf{n} - 2\pi & \oint_3 T_2\mathbf{n} & \oint_4 T_2\mathbf{n} \\ \oint_1 T_3\mathbf{n} & \oint_2 T_3\mathbf{n} & \oint_3 T_3\mathbf{n} - 2\pi & \oint_4 T_3\mathbf{n} \\ \oint_1 T_4\mathbf{n} & \oint_2 T_4\mathbf{n} & \oint_3 T_4\mathbf{n} & \oint_4 T_4\mathbf{n} - 2\pi \end{pmatrix} \cdot \begin{pmatrix} u_1, v_1 \\ u_2, v_2 \\ u_3, v_3 \\ u_4, v_4 \end{pmatrix} = \begin{pmatrix} \oint g_1\,\sigma\mathbf{n} \\ \oint g_2\,\sigma\mathbf{n} \\ \oint g_3\,\sigma\mathbf{n} \\ \oint g_4\,\sigma\mathbf{n} \end{pmatrix}$$

Computing Boundary Elements

$$2\pi\eta u_i(\mathbf{x}_0) = \oint_\omega (\eta\mathbf{T}\,\mathbf{n}\cdot\mathbf{u} - \bar{\bar{\sigma}}\mathbf{n}\cdot\mathbf{g}\,)\,ds$$

- Discretizing the boundary with Ansatz functions
- For each boundary 2 unknowns, but for each test function 2 equations.
- Results in a dense matrix.



Contribution from black panel

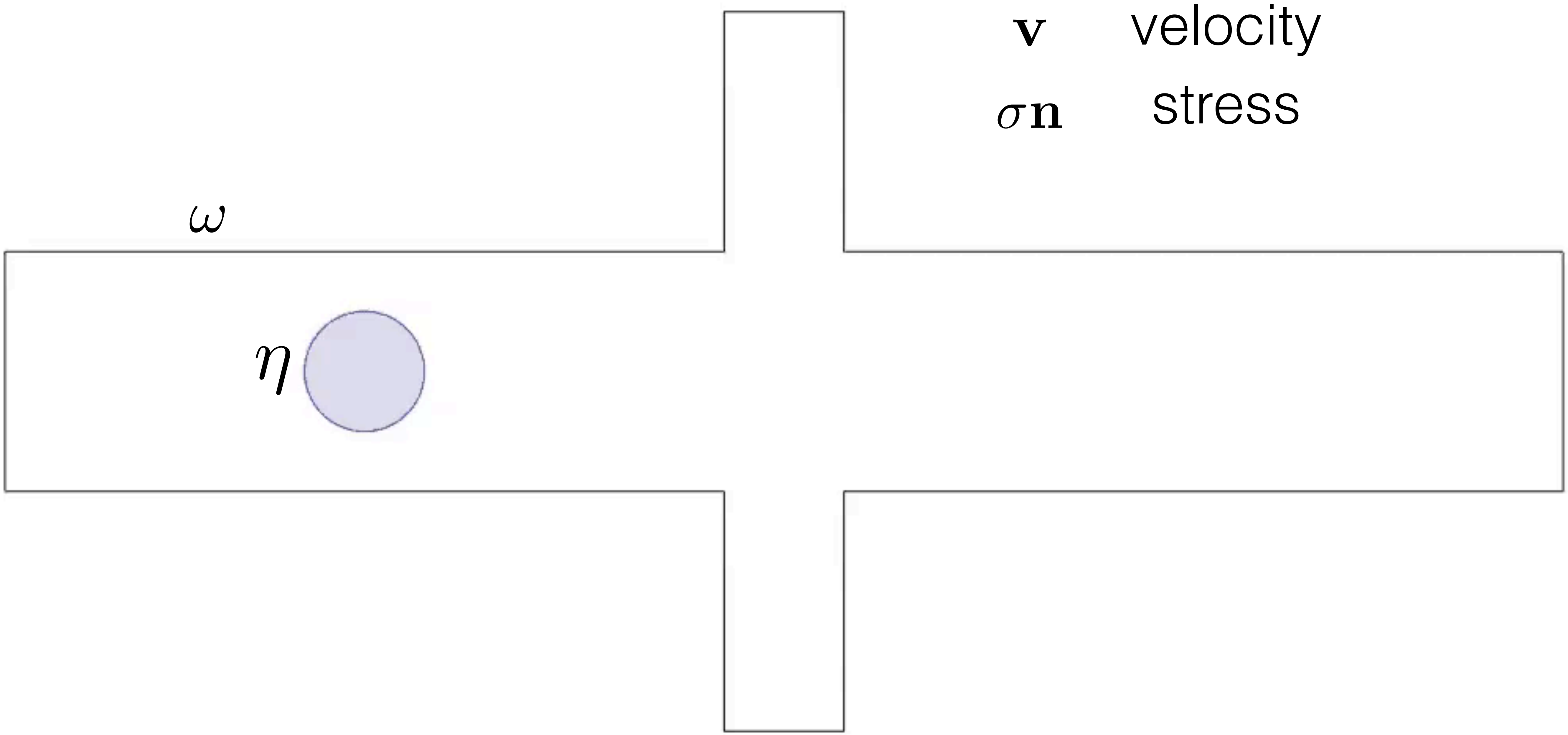
$$\eta\begin{pmatrix}
\oint_1 T_1\mathbf{n} - 2\pi & \oint_2 T_1\mathbf{n} & \oint_3 T_1\mathbf{n} & \oint_4 T_1\mathbf{n} \\
\oint_1 T_2\mathbf{n} & \oint_2 T_2\mathbf{n} - 2\pi & \oint_3 T_2\mathbf{n} & \oint_4 T_2\mathbf{n} \\
\oint_1 T_3\mathbf{n} & \oint_2 T_3\mathbf{n} & \oint_3 T_3\mathbf{n} - 2\pi & \oint_4 T_3\mathbf{n} \\
\oint_1 T_4\mathbf{n} & \oint_2 T_4\mathbf{n} & \oint_3 T_4\mathbf{n} & \oint_4 T_4\mathbf{n} - 2\pi
\end{pmatrix}\cdot\begin{pmatrix} u_1, v_1 \\ u_2, v_2 \\ u_3, v_3 \\ u_4, v_4 \end{pmatrix} = \begin{pmatrix} \oint g_1\,\sigma\mathbf{n} \\ \oint g_2\,\sigma\mathbf{n} \\ \oint g_3\,\sigma\mathbf{n} \\ \oint g_4\,\sigma\mathbf{n} \end{pmatrix}$$

Boundary Integral Equation

$$\oint_{\omega} \Big(\mathbf{Tn} \cdot \mathbf{v} - \sigma \mathbf{n} \cdot \mathbf{G} \Big) ds + \oint_{\eta} \Big(\mathbf{Tn} \cdot (\lambda - 1) \mathbf{v} - [\![\sigma \mathbf{n}]\!] \cdot \mathbf{G} \Big) ds = S \mathbf{w} \cdot \mathbf{v}(\mathbf{x}_0).$$

$\mathbf{v}$ velocity

$\sigma\mathbf{n}$ stress

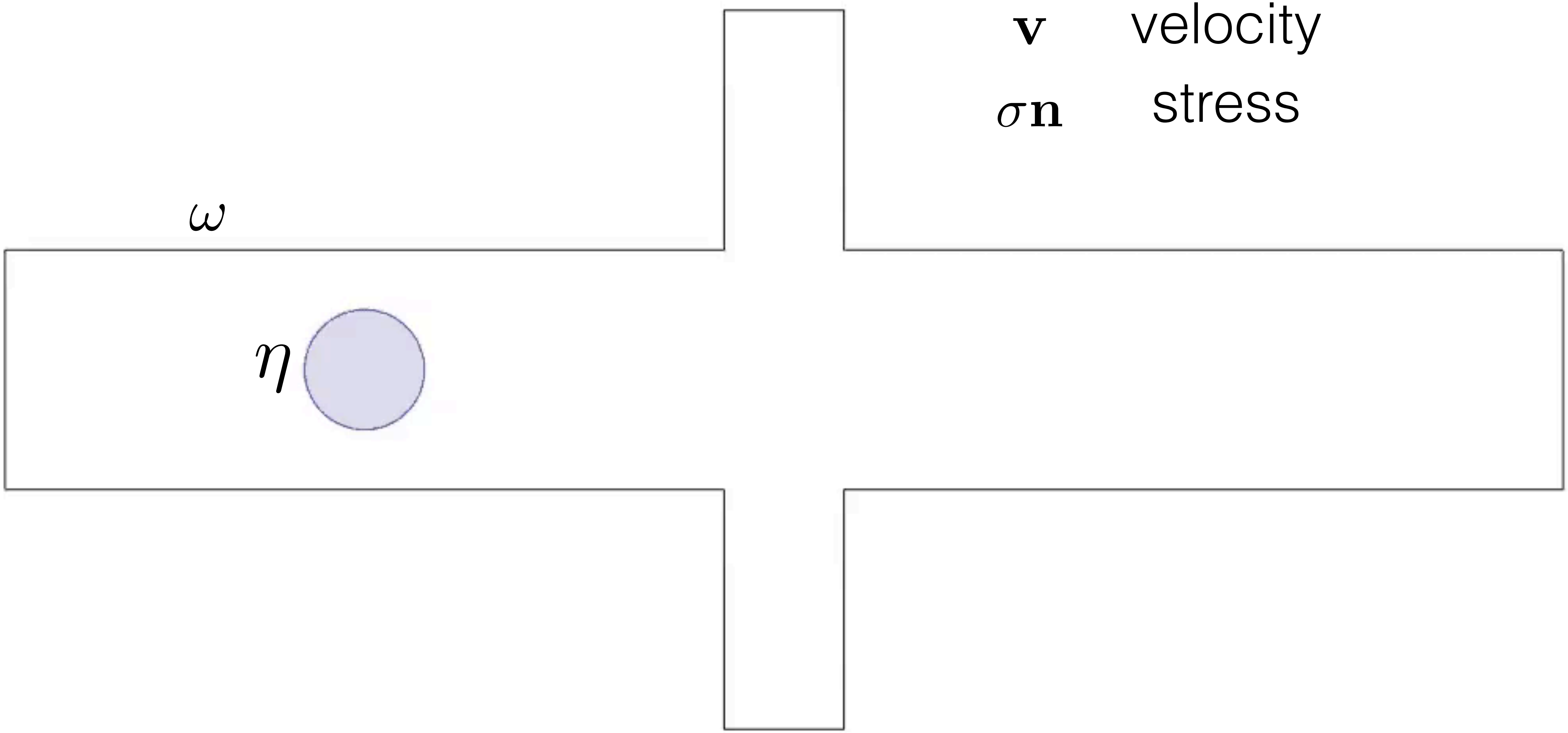


Boundary Integral Equation

$$\oint_{\omega} \Big(\mathbf{Tn} \cdot \mathbf{v} - \sigma \mathbf{n} \cdot \mathbf{G} \Big) ds + \oint_{\eta} \Big(\mathbf{Tn} \cdot (\lambda - 1)\mathbf{v} - [\![\sigma \mathbf{n}]\!] \cdot \mathbf{G} \Big) ds = S\mathbf{w} \cdot \mathbf{v}(\mathbf{x}_0).$$

$\mathbf{v}$ velocity

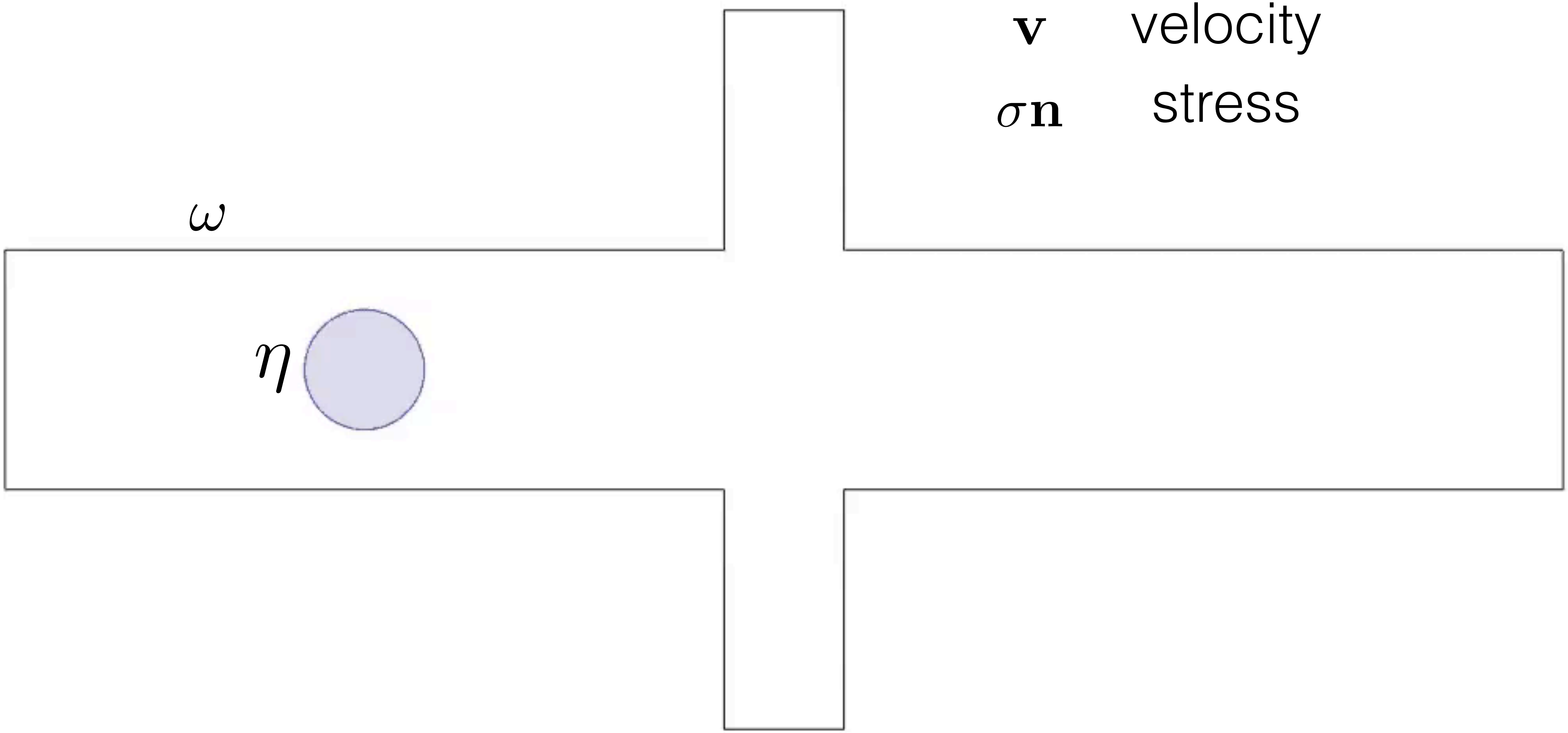
$\sigma\mathbf{n}$ stress



Boundary Integral Equation

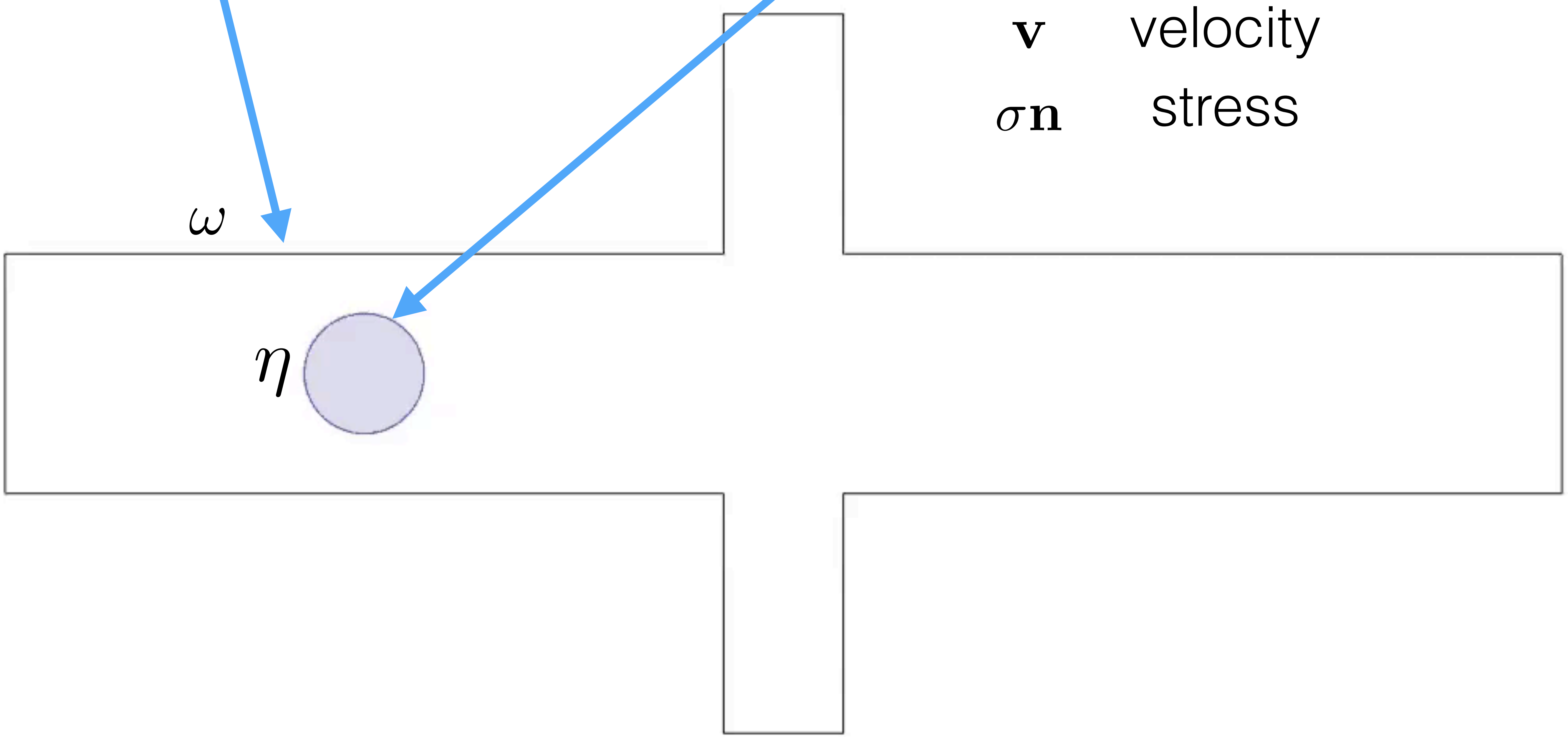
$$\oint_\omega \Big(\mathbf{Tn}\cdot\mathbf{v} - \sigma\mathbf{n}\cdot\mathbf{G}\Big)ds + \oint_\eta \Big(\mathbf{Tn}\cdot(\lambda-1)\mathbf{v} - [\![\sigma\mathbf{n}]\!]\cdot\mathbf{G}\Big)ds = S\mathbf{w}\cdot\mathbf{v}(\mathbf{x}_0).$$

$\mathbf{v}$	velocity
$\sigma\mathbf{n}$	stress



Boundary Integral Equation

$$\oint_{\omega}\Big(\mathbf{Tn}\cdot\mathbf{v}-\sigma\mathbf{n}\cdot\mathbf{G}\Big)ds+\oint_{\eta}\Big(\mathbf{Tn}\cdot(\lambda-1)\mathbf{v}-[\![\sigma\mathbf{n}]\!]\cdot\mathbf{G}\Big)ds=S\mathbf{w}\cdot\mathbf{v}(\mathbf{x}_0).$$

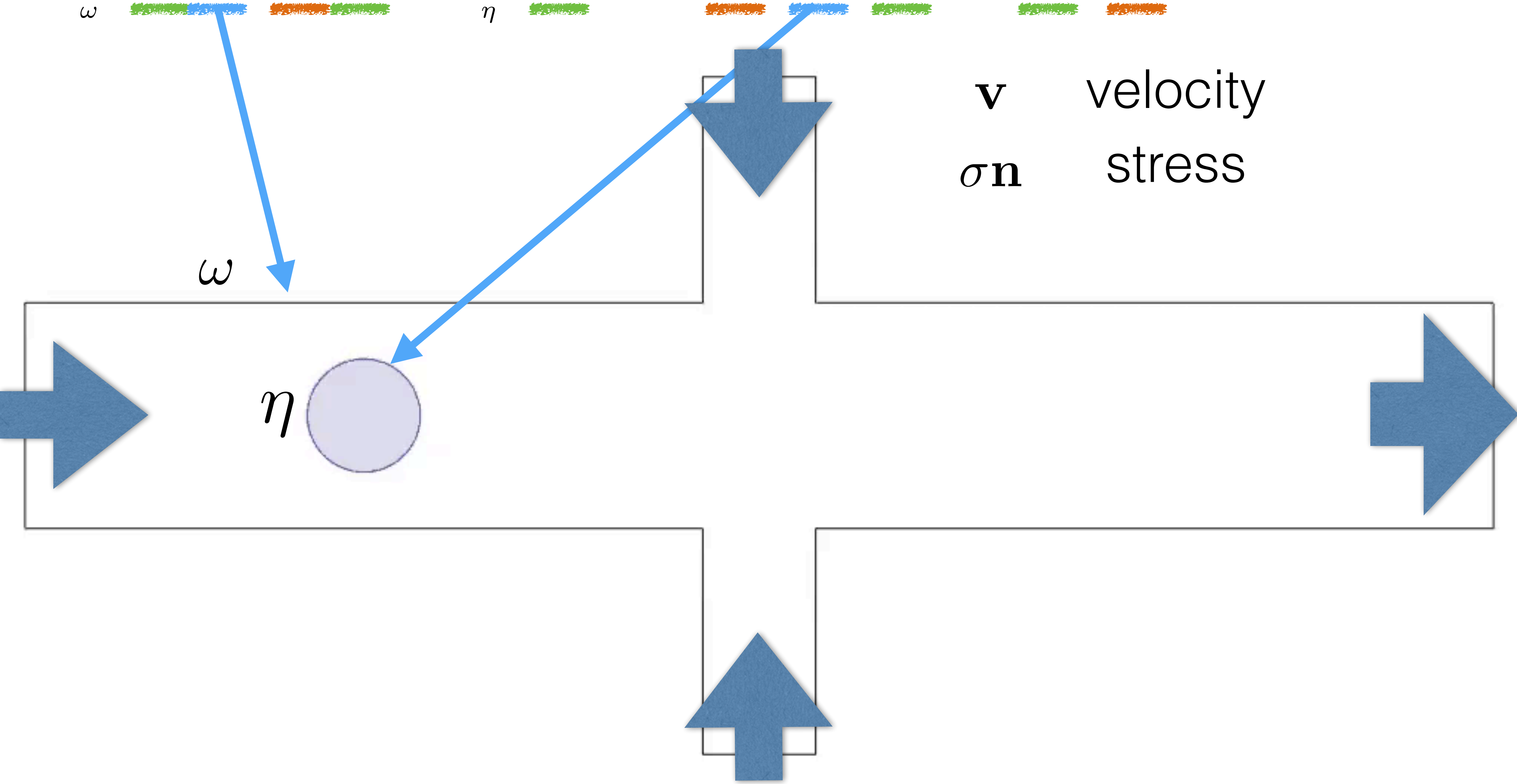


Boundary Integral Equation

$$\oint_{\omega} \left(\mathbf{T}\mathbf{n}\cdot\mathbf{v} - \sigma\mathbf{n}\cdot\mathbf{G}\right)ds + \oint_{\eta} \left(\mathbf{T}\mathbf{n}\cdot(\lambda-1)\mathbf{v} - [\![\sigma\mathbf{n}]\!]\cdot\mathbf{G}\right)ds = S\mathbf{w}\cdot\mathbf{v}(\mathbf{x}_0).$$

$\mathbf{v}$ velocity

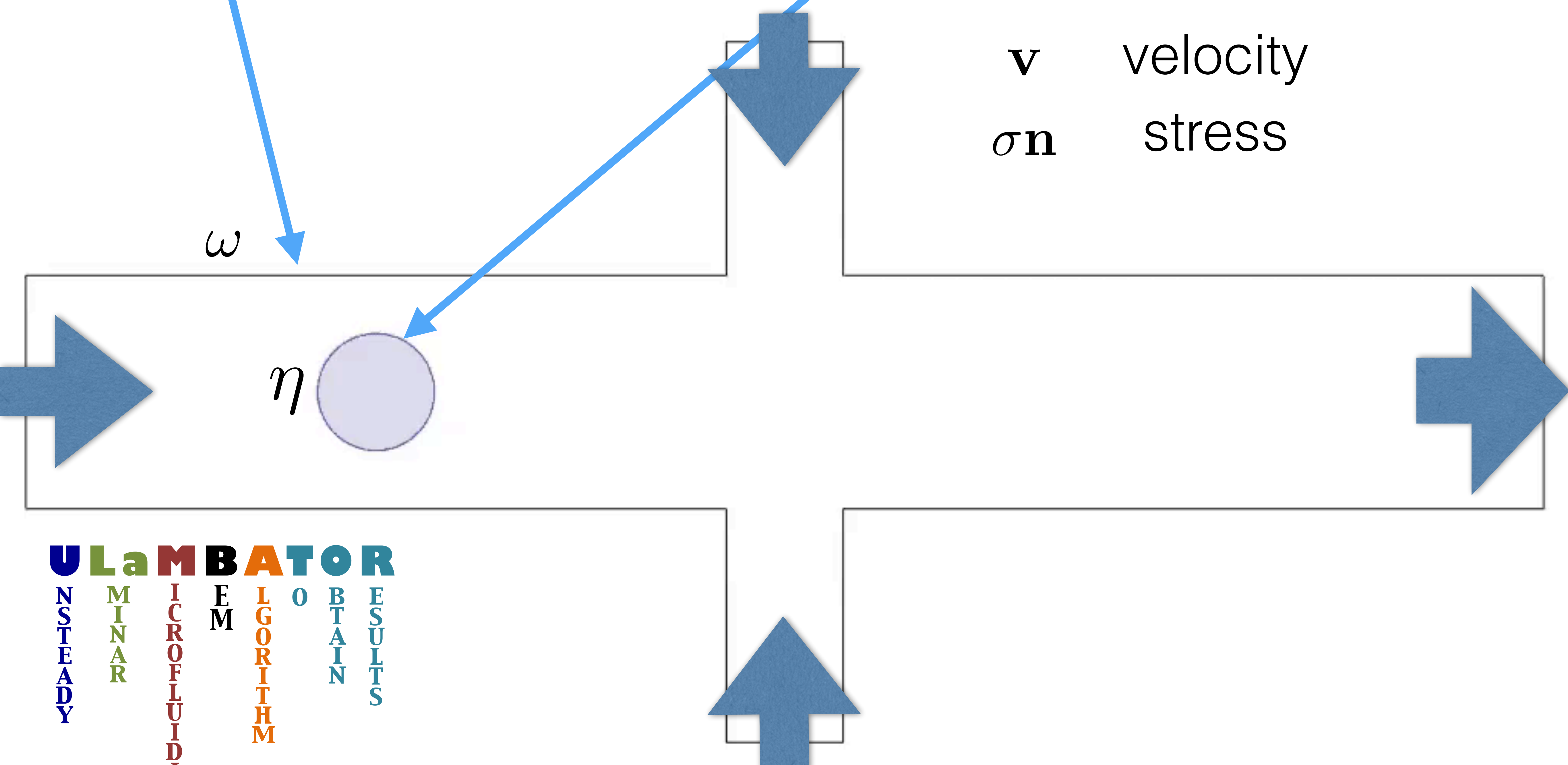
$\sigma\mathbf{n}$ stress



Boundary Integral Equation

$$\oint_{\omega}\Big(\mathbf{Tn}\cdot\mathbf{v}-\sigma\mathbf{n}\cdot\mathbf{G}\Big)ds+\oint_{\eta}\Big(\mathbf{Tn}\cdot(\lambda-1)\mathbf{v}-[\![\sigma\mathbf{n}]\!]\cdot\mathbf{G}\Big)ds=S\mathbf{w}\cdot\mathbf{v}(\mathbf{x}_0).$$

$\mathbf{v}$	velocity
$\sigma\mathbf{n}$	stress

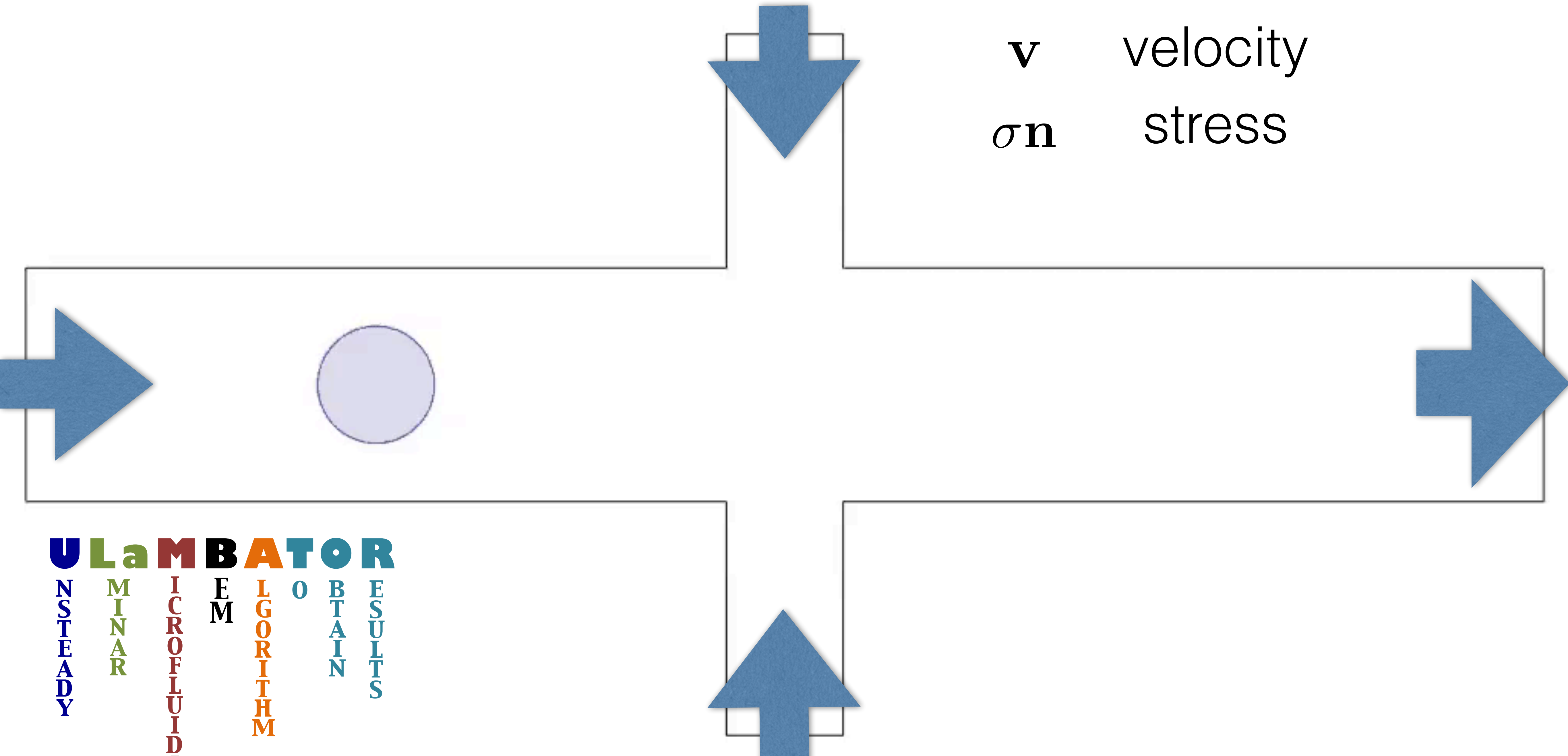


Boundary Integral Equation

$$\oint_{\omega} \Big(\mathbf{Tn} \cdot \mathbf{v} - \sigma \mathbf{n} \cdot \mathbf{G} \Big) ds + \oint_{\eta} \Big(\mathbf{Tn} \cdot (\lambda - 1) \mathbf{v} - [\![\sigma \mathbf{n}]\!] \cdot \mathbf{G} \Big) ds = S \mathbf{w} \cdot \mathbf{v}(\mathbf{x}_0).$$

$\mathbf{v}$ velocity

$\sigma \mathbf{n}$ stress



ULaMBATOR

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta_{\mathrm{bulk}}}$$ [m/s]

velocity scale

non-dimensional

$$\hat{u} = \tilde{U} u$$

dimensional

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta_{\text{bulk}}} \text{ [m/s]}$$

velocity scale

non-dimensional

$$\hat{u} = \tilde{U} u$$

dimensional

$$l = \tilde{L} L$$

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta_{\text{bulk}}}$$ [m/s]

velocity scale

non-dimensional

$$\hat{u} = \tilde{U} u$$

dimensional

$$l = \tilde{L} L \qquad \eta_d = \eta\lambda$$

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta_{\text{bulk}}}$$ [m/s]

velocity scale

non-dimensional

$$\hat{u} = \tilde{U} u$$

dimensional

$$l = \tilde{L} L \qquad \eta_d = \eta \lambda \qquad Ca = \frac{\eta \tilde{U}}{\sigma_{\alpha\beta}} = 1$$

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta_{\mathrm{bulk}}}$$ [m/s]

velocity scale

non-dimensional

$$\hat{u} = \tilde{U}u$$

dimensional

$$l = \tilde{L}L \qquad \eta_d = \eta\lambda \qquad Ca = \frac{\eta\hat{u}}{\sigma_{\alpha\beta}}$$

Problem preparation

In order calculate the problem needs to be non-dimensional.

The base dimensions are: $\eta, \sigma_{\alpha\beta}$ and $\tilde{L}$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta_{\text{bulk}}} \text{ [m/s]}$$

velocity scale

non-dimensional

$$\hat{u} = \tilde{U} u$$

dimensional

$$l = \tilde{L} L \qquad \eta_d = \eta\lambda \qquad Ca = \frac{\eta \hat{u}}{\sigma_{\alpha\beta}}$$

In order to understand the result, the results need to be dimensional.

$$p = \frac{\sigma_{\alpha\beta}}{\tilde{L}} P \qquad t = \frac{\tilde{L}\eta}{\sigma_{\alpha_\beta}} T$$

So far for the theory

User Control

User Control

- Define the boundaries (geometry and boundary condition)

User Control

- Define the boundaries (geometry and boundary condition)
- Define the liquid interface(s)

User Control

- Define the boundaries (geometry and boundary condition)
- Define the liquid interface(s)
- Specify the problem (material parameters, …)

User Control

- Define the boundaries (geometry and boundary condition)
- Define the liquid interface(s)
- Specify the problem (material parameters, …)
- Setup an iteration loop (advance and write output)

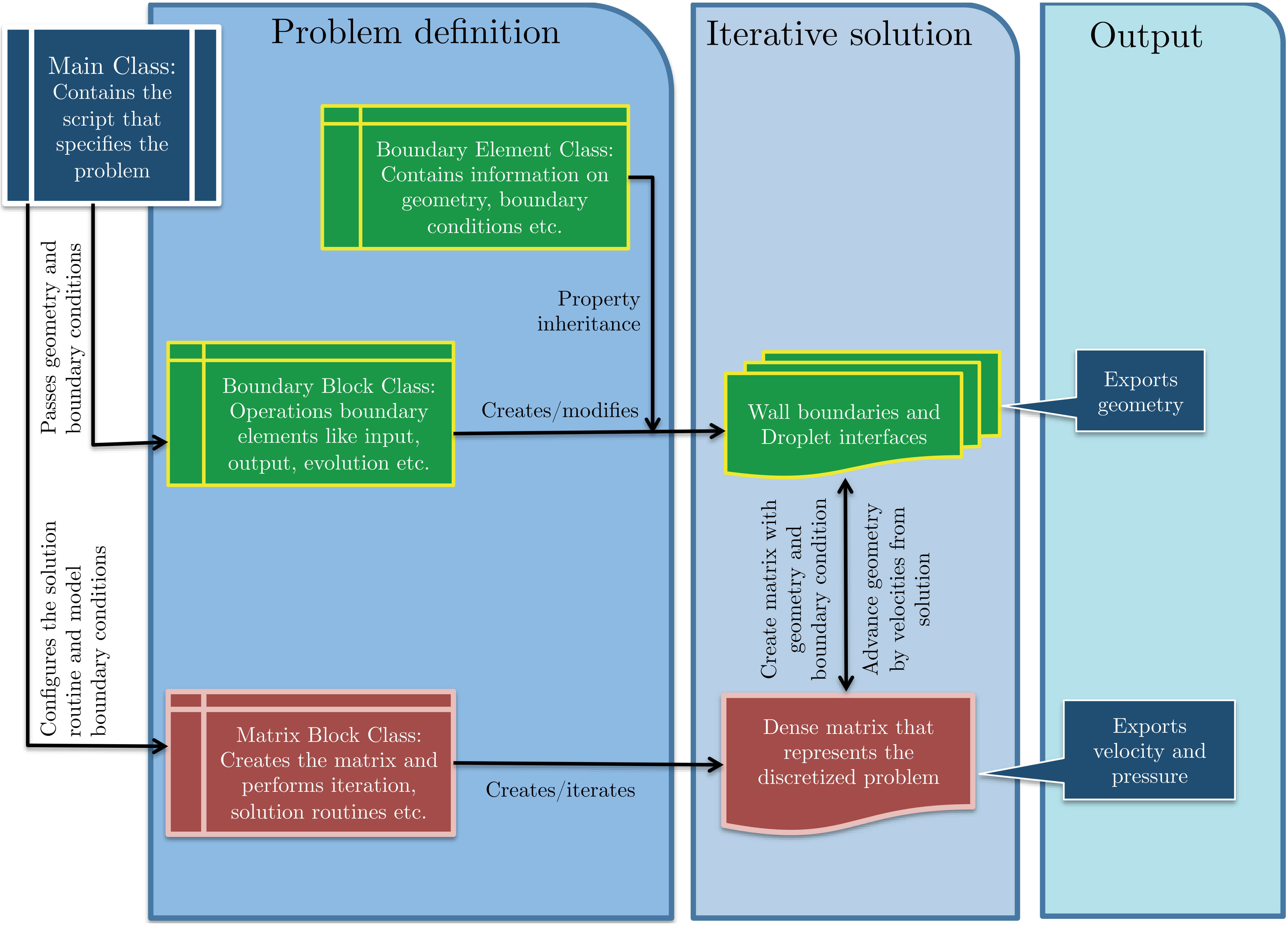
User Control

- Define the boundaries (geometry and boundary condition)
- Define the liquid interface(s)
- Specify the problem (material parameters, …)
- Setup an iteration loop (advance and write output)
- Compile and run the code

User Control

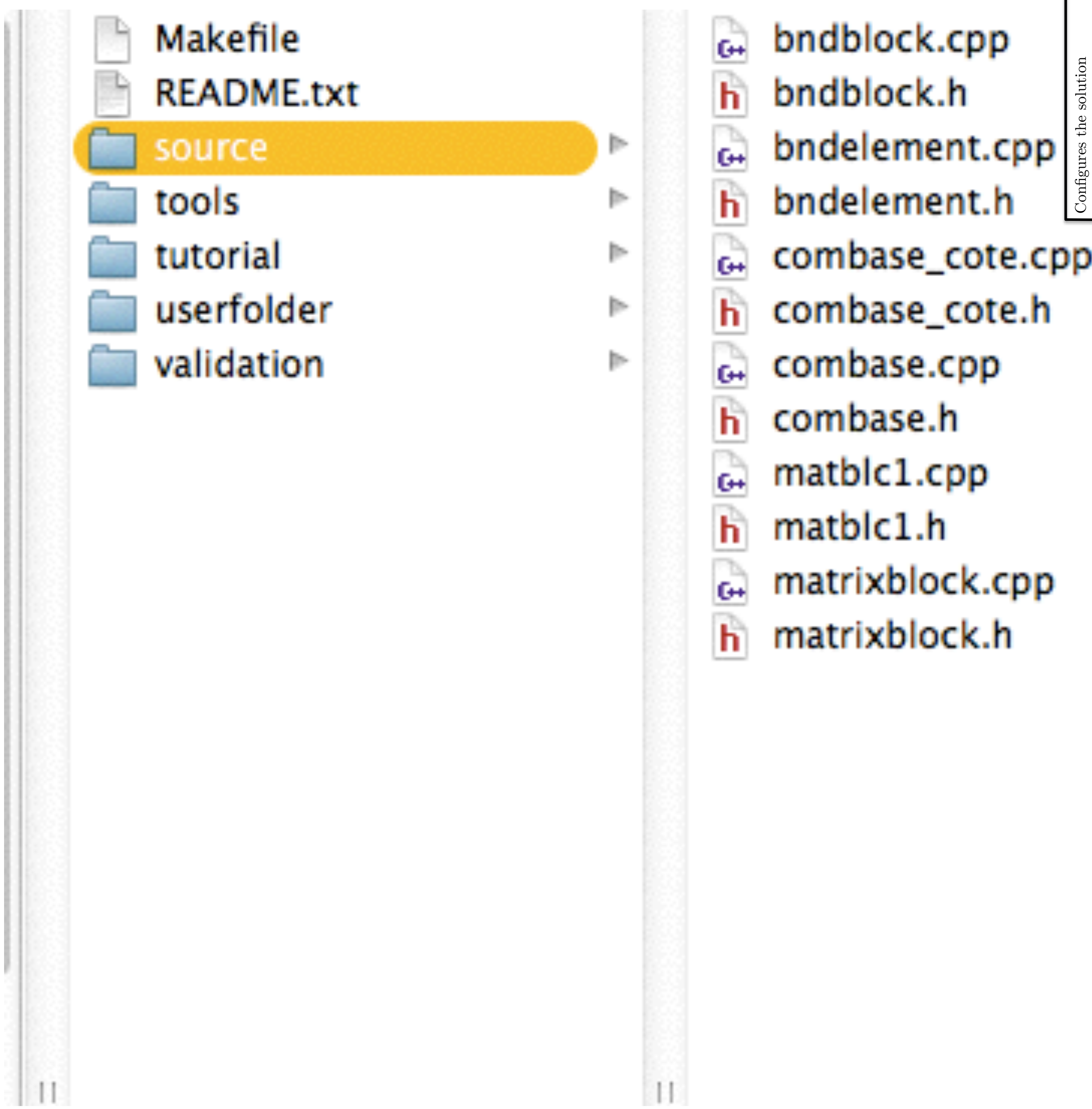
- Define the boundaries (geometry and boundary condition)
- Define the liquid interface(s)
- Specify the problem (material parameters, …)
- Setup an iteration loop (advance and write output)
- Compile and run the code

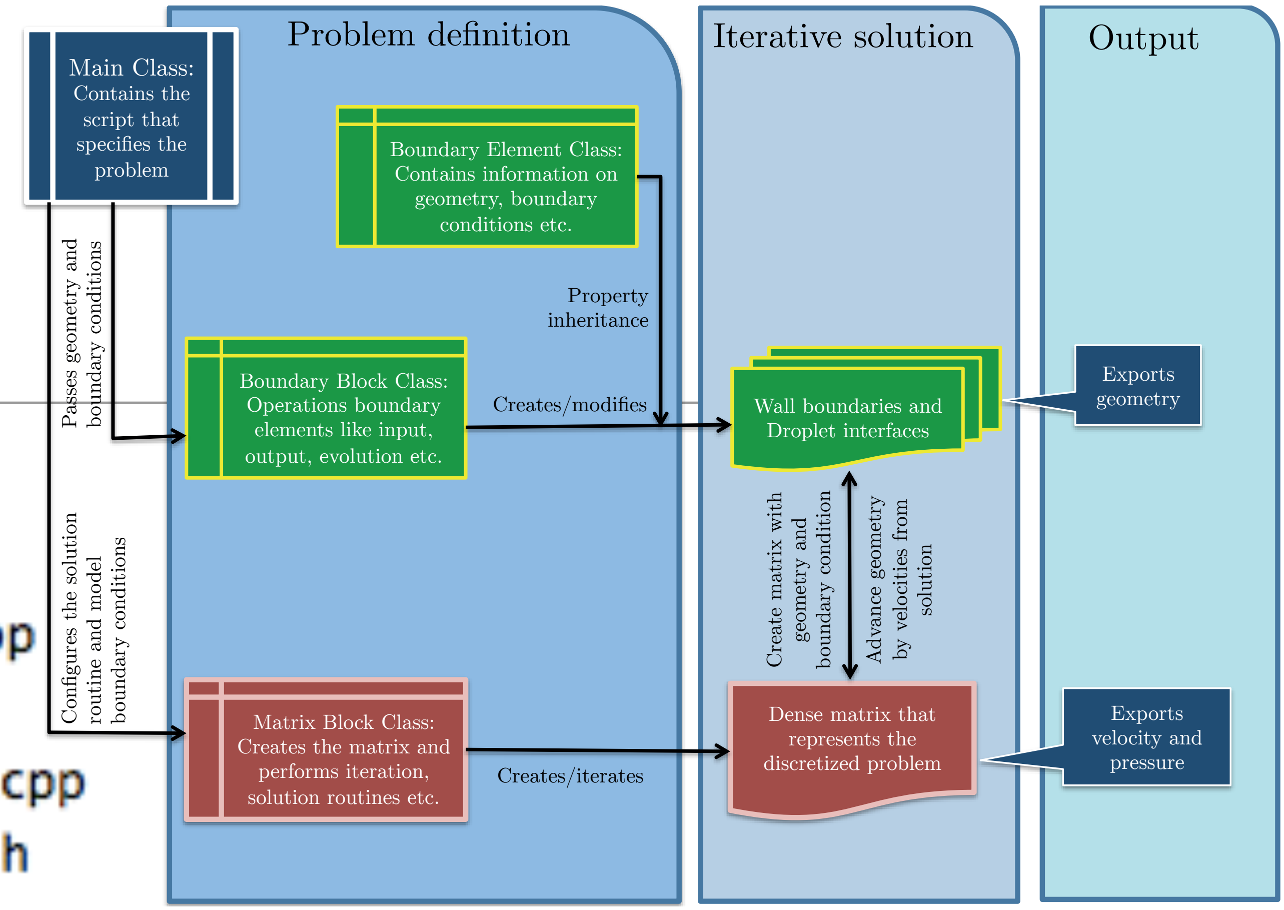
Ulambator is not run by an interpreter but compiled and runs by itself



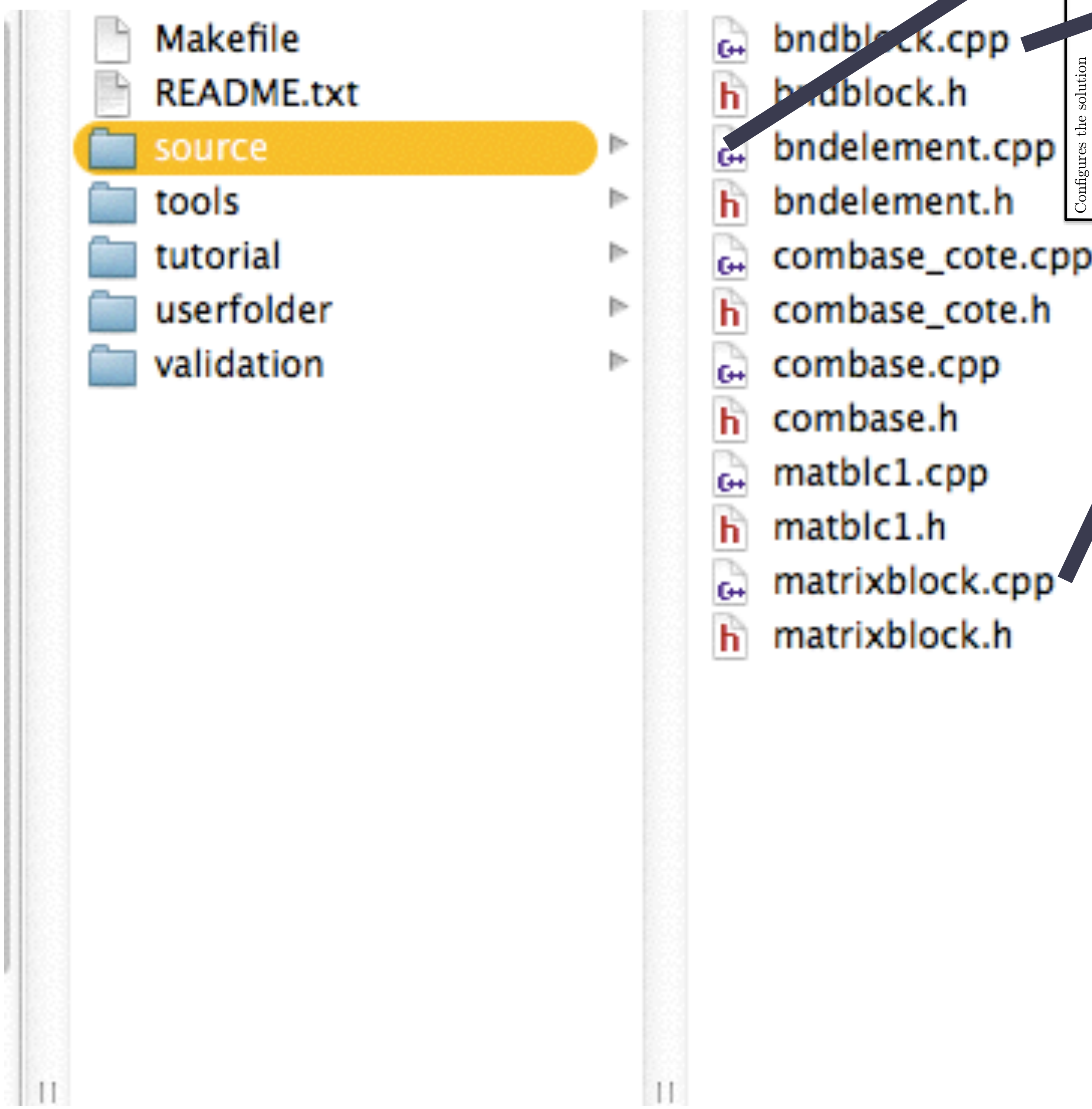
Problem definition
Iterative solution
Output
Main Class: Contains the script that specifies the problem
Boundary Element Class: Contains information on geometry, boundary conditions etc.
Property inheritance
Passes geometry and boundary conditions
Boundary Block Class: Operations boundary elements like input, output, evolution etc.
Creates/modifies
Wall boundaries and Droplet interfaces
Exports geometry
Create matrix with geometry and boundary condition
Advance geometry by velocities from solution
Configures the solution routine and model boundary conditions
Matrix Block Class: Creates the matrix and performs iteration, solution routines etc.
Creates/iterates
Dense matrix that represents the discretized problem
Exports velocity and pressure

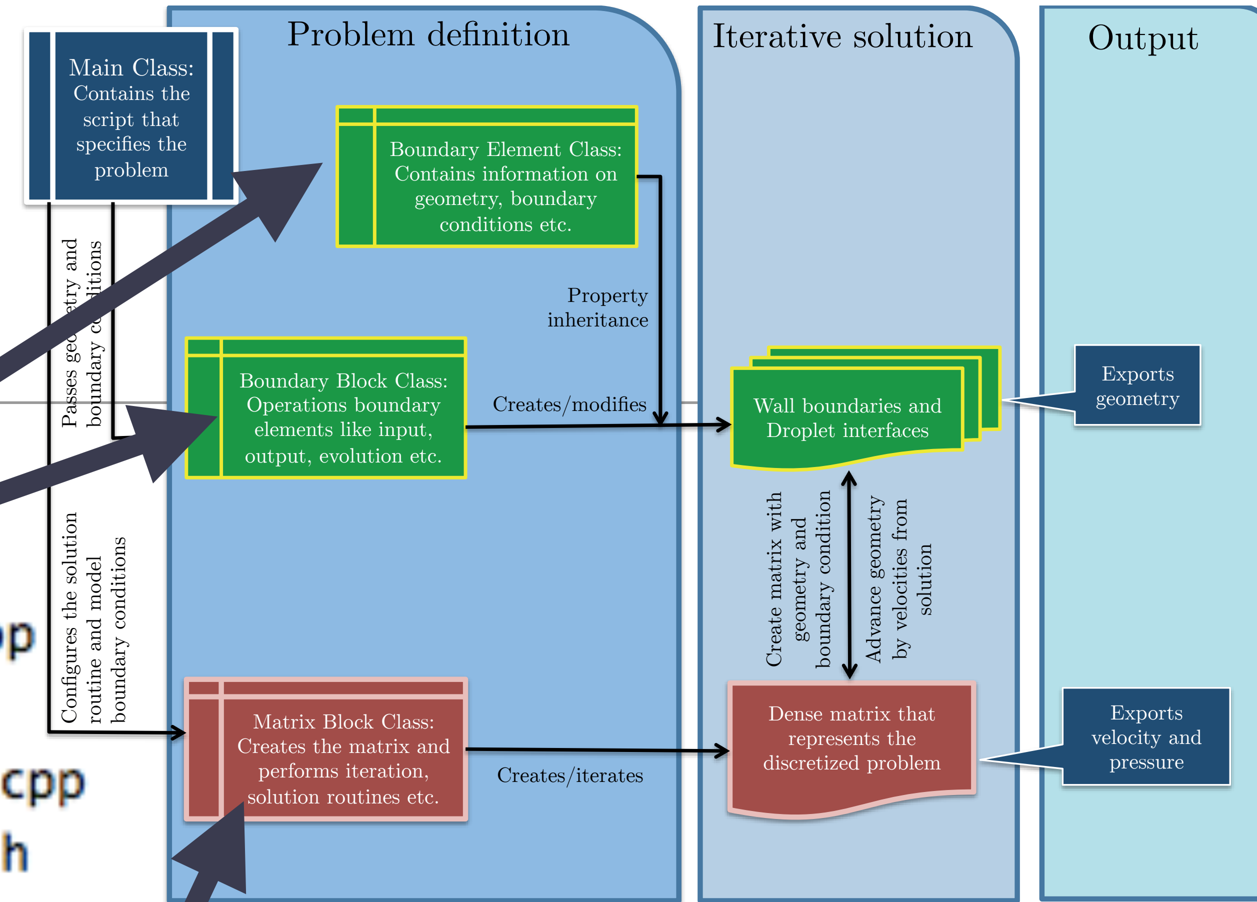
Ulambator Folder



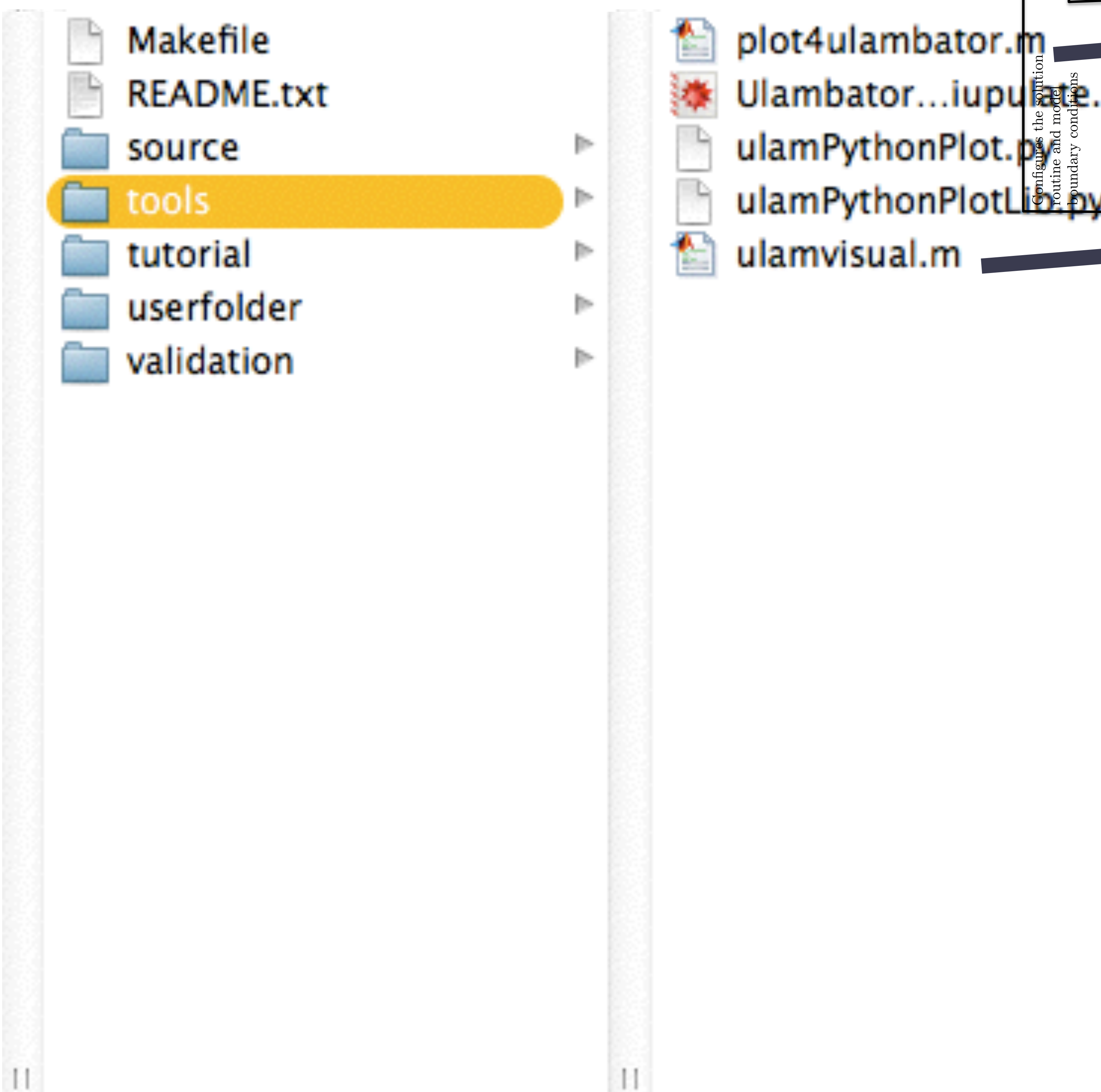


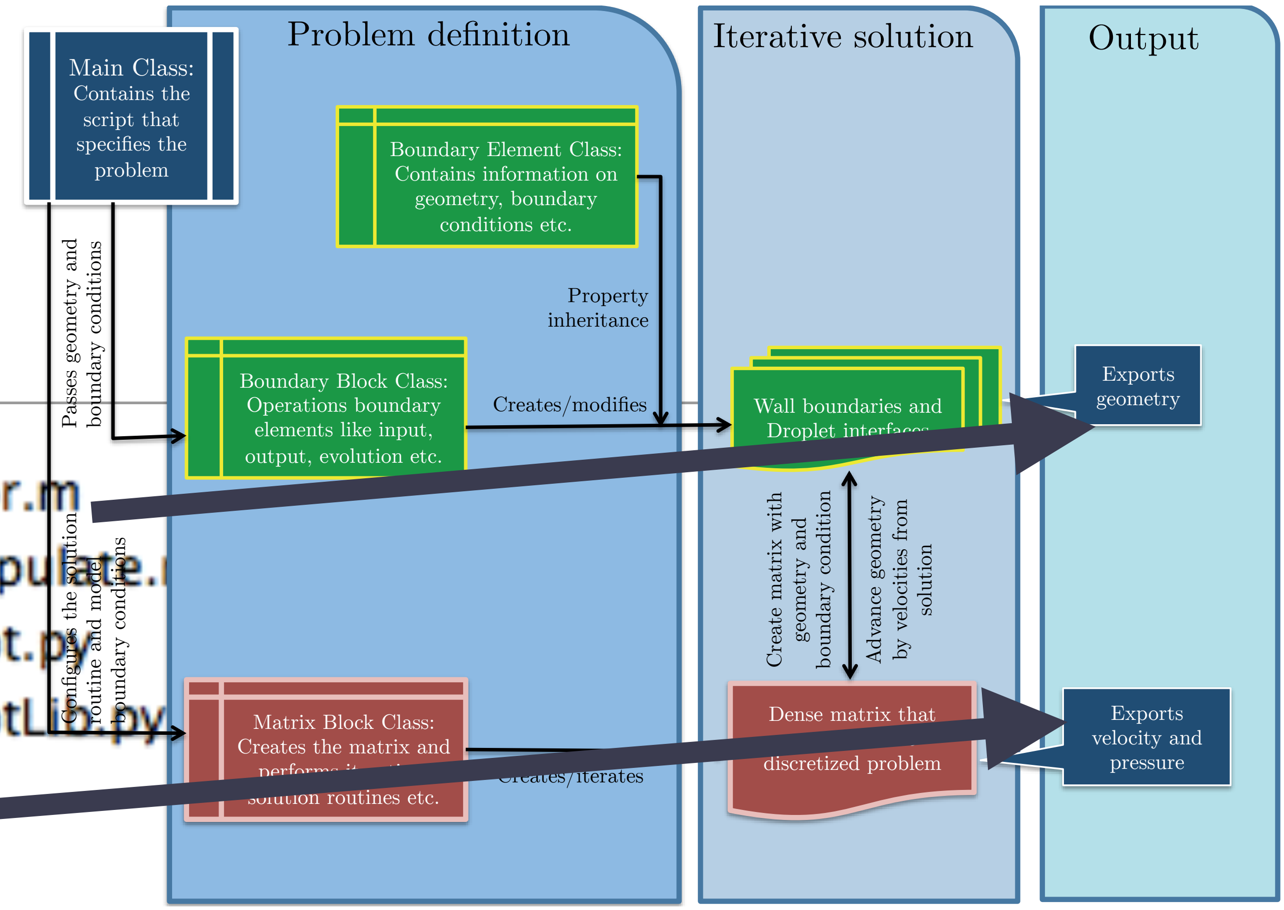
Ulambator Folder



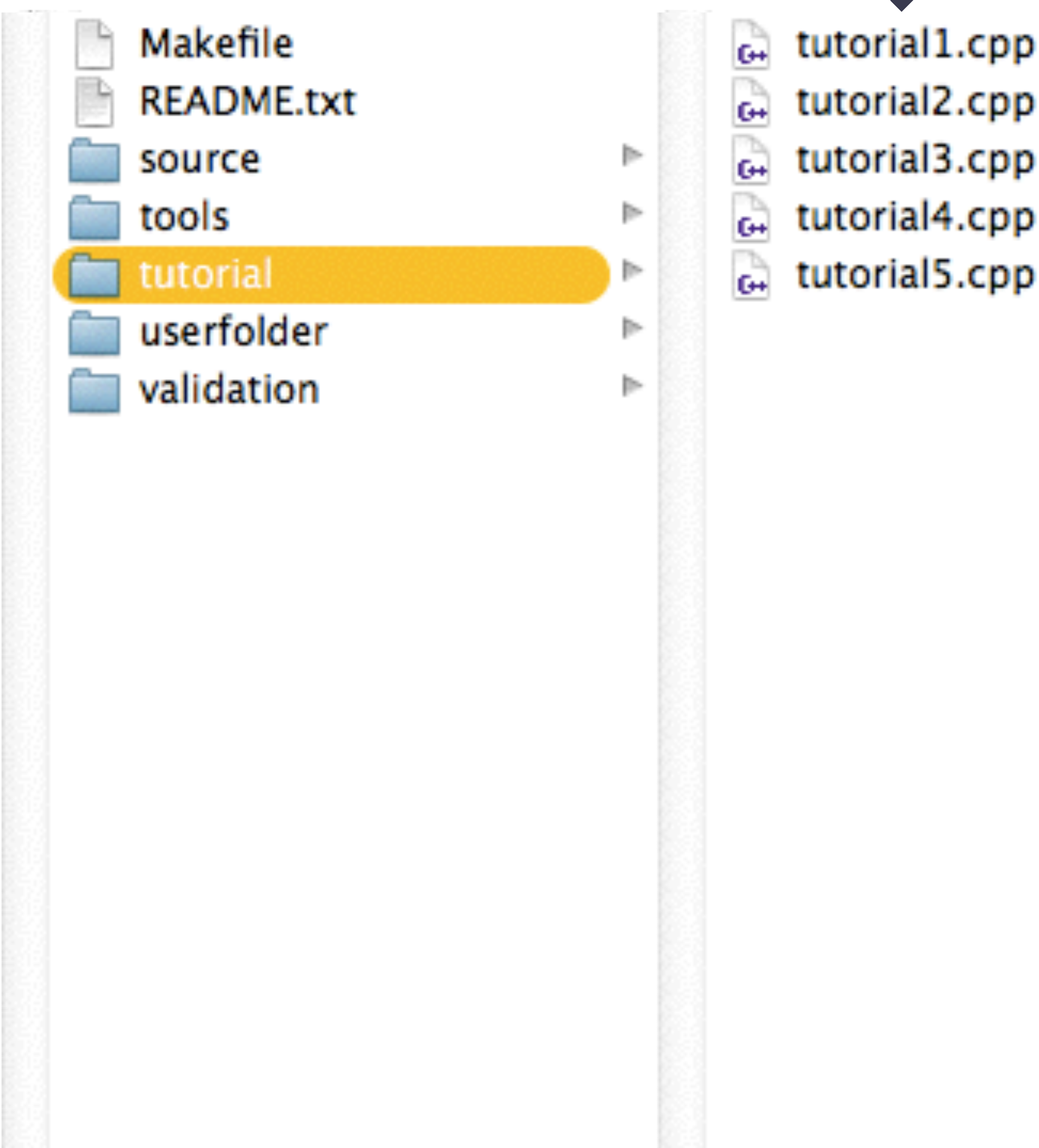


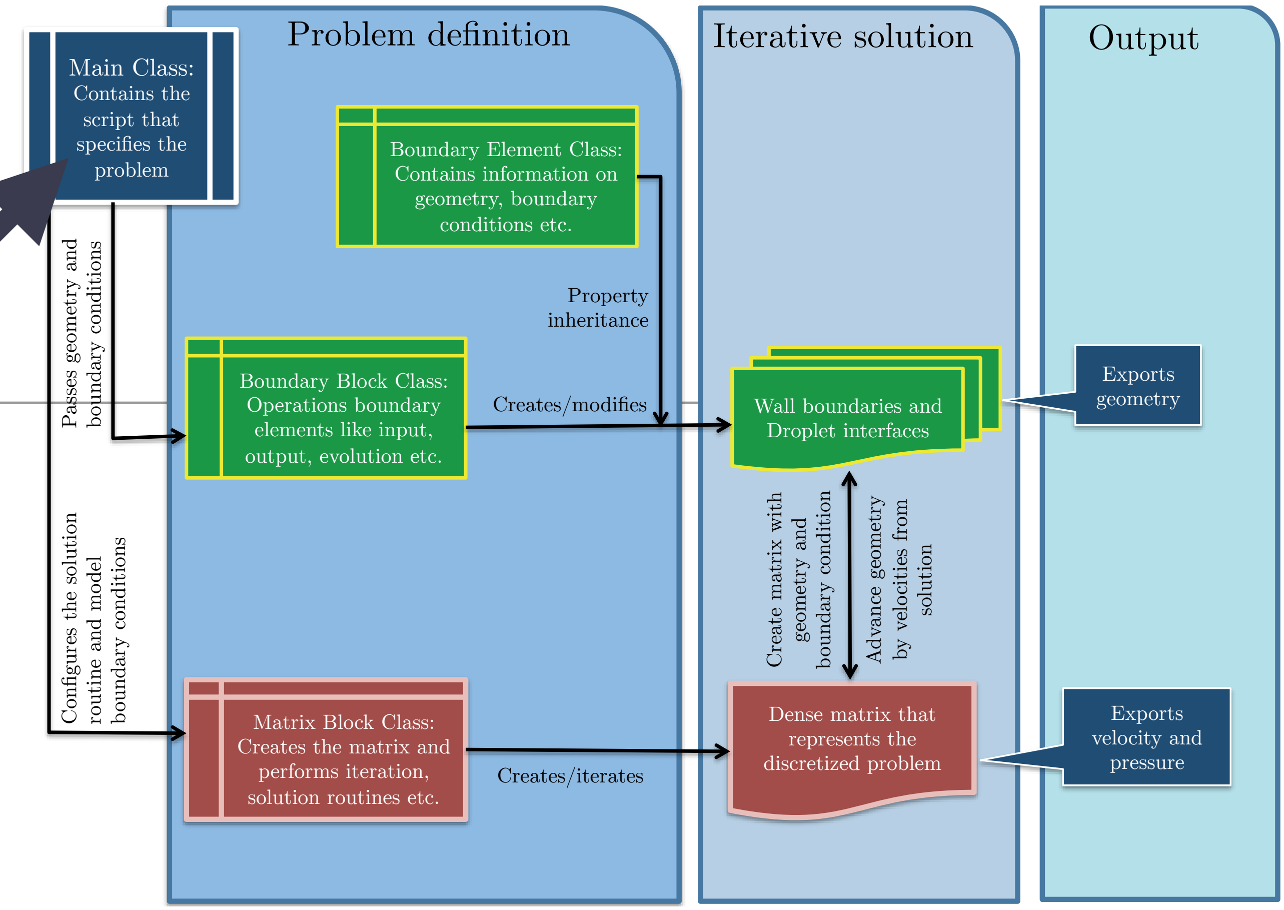
Ulambator Folder



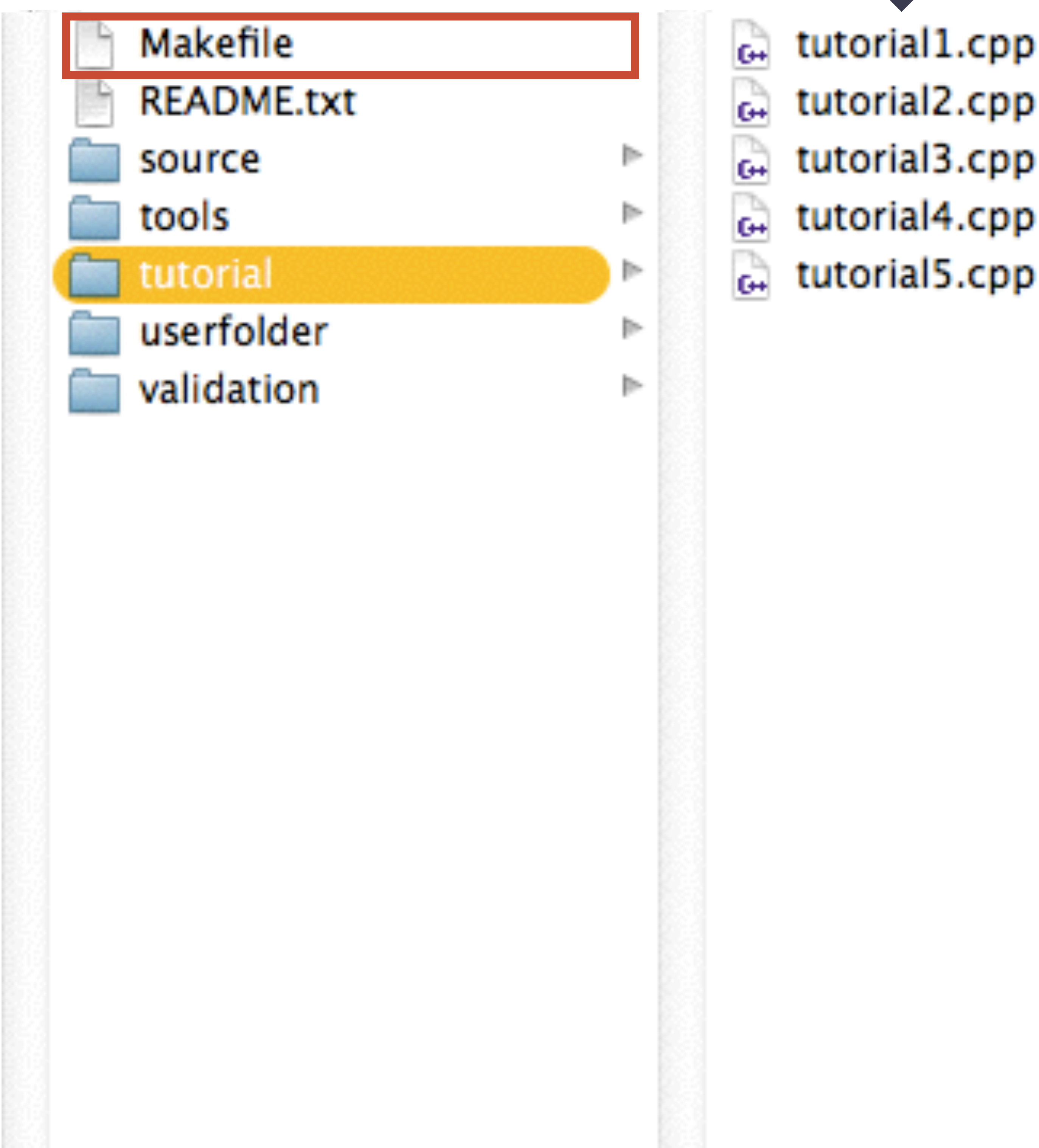


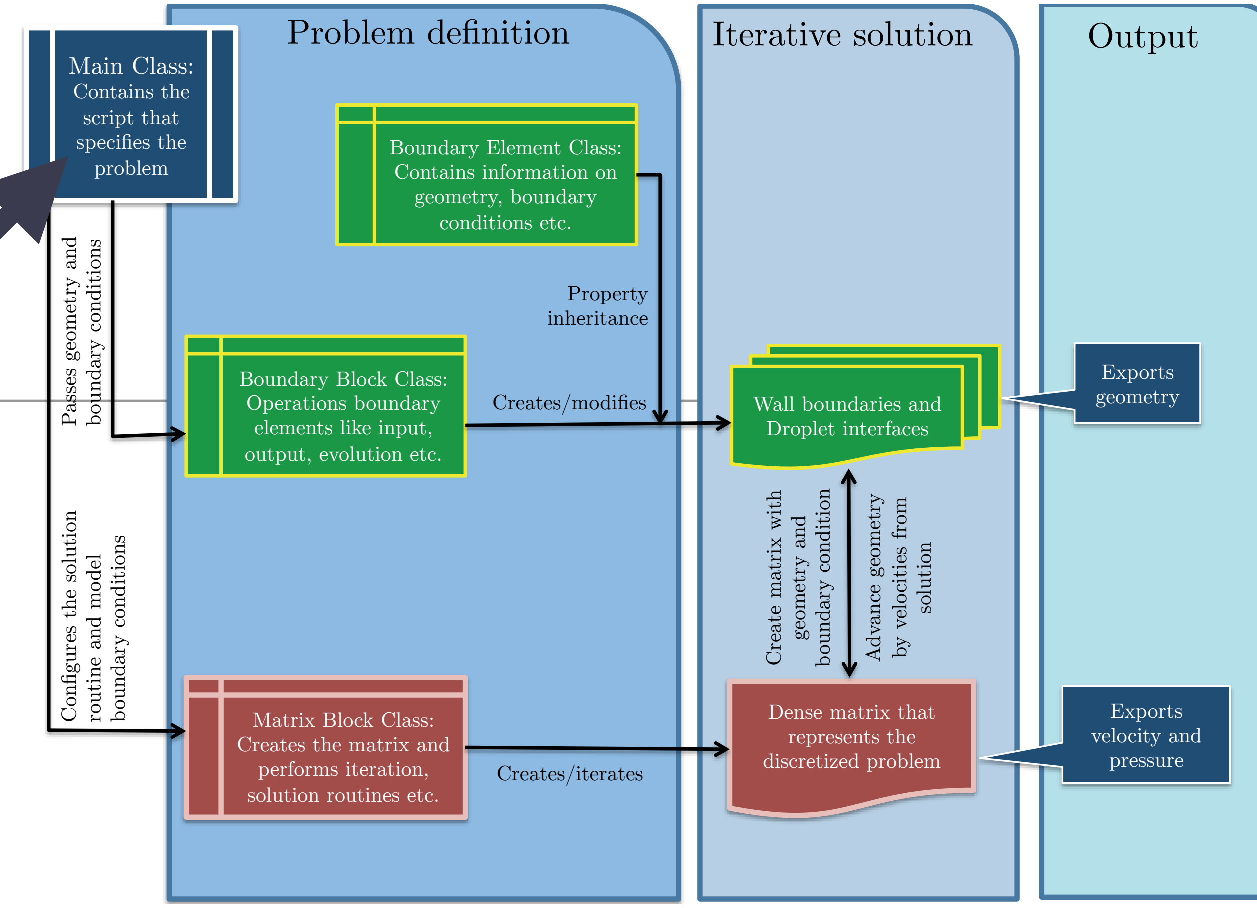
Ulambator Folder





Ulambator Folder





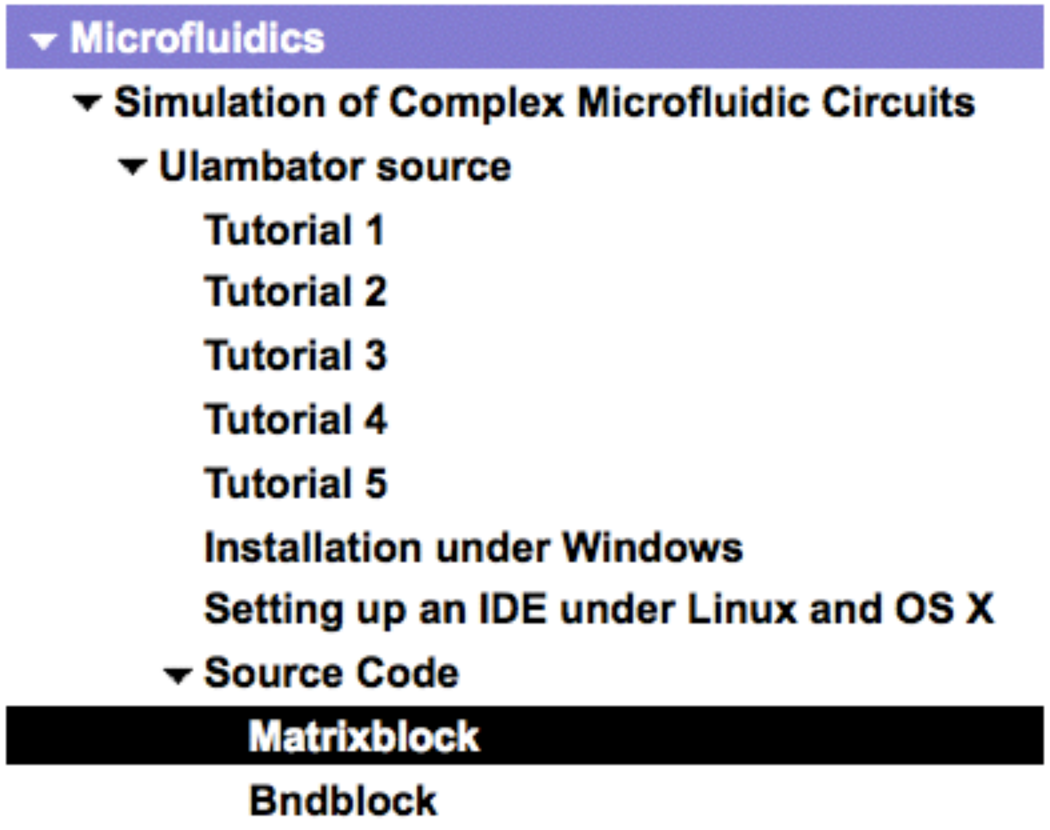
```
# Makefile to compile Ulambator, 01.12.2015
# The sharp (#) symbols indicate comment, which are not executed

# Please adapt the following lines
# compiler of choice: g++, icc, etc.
CPP = g++

# compiler options for g++ with Xcode
OPTS = -lstdc++ -framework Accelerate -O3 -lpthread

# compiler options for g++ with Lapack libraries
# OPTS = -fopenmp -lstdc++ -llapack -lblas -lpthread

# compiler options for icc and mkl (Intel)
# OPTS = -lstdc++ -mkl -O3 -openmp

# library path for instance for LAPACK, if needed
# LIBS = -L/opt/intel/composer_xe_2013.1.119/mkl/lib/

# include path for headers, if needed
# INCS = -I/Developer/SDKs/MacOSX10.7.sdk/usr/lib/gcc/i686-apple-darwin11/4.2.1/include

# That is all, only edit the lines below for new source files
# or to add user defined cases to the compiler (example: user1)

# source files except main classes
SRC = source/combase.cpp source/bndblock.cpp source/matrixblock.cpp source/bndelement.cpp source/matblc1.cpp

# source files for compiler test
TRC = source/combase_cote.cpp source/bndblock.cpp source/matrixblock.cpp source/bndelement.cpp source/matblc1.cpp -w

# Lines hereafter don't need modification unless you want to add your own cases.

all: tutorial1 tutorial2 tutorial3 tutorial4 tutorial5

# A user defined case, if more are needed copy, paste, rename (i.e. userN) and adapt

user1: userfolder/main_user1.cpp
	$(CPP)
	@if [ $$? = 0 ]; then \
	  echo "  Compiling $@ done" ; \
	else \
	  echo " Compiling $@ failed" ; \
	fi

tutorial1: tutorial/tutorial1.cpp
	$(CPP)
	@if [ $$? = 0 ]; then \
	  echo "  Compiling $@ done" ; \
	else \
	  echo " Compiling $@ failed" ; \
	fi
```

```
# Makefile to compile Ulambator, 01.12.2015
# The sharp (#) symbols indicate comment, which are not executed

# Please adapt the following lines
# compiler of choice: g++, icc, etc.
CPP = g++

# compiler options for g++ with Xcode
OPTS = -lstdc++ -framework Accelerate -O3 -lpthread

# compiler options for g++ with Lapack libraries
# OPTS = -fopenmp -lstdc++ -llapack -lblas -lpthread

# compiler options for icc and mkl (Intel)
# OPTS = -lstdc++ -mkl -O3 -openmp

# library path for instance for LAPACK, if needed
# LIBS = -L/opt/intel/composer_xe_2013.1.119/mkl/lib/

# include path for headers, if needed
# INCS = -I/Developer/SDKs/MacOSX10.7.sdk/usr/lib/gcc/i686-apple-darwin11/4.2.1/include
# That is all, only edit the lines below for new source files
# or to add user defined cases to the compiler (example: user1)

# source files except main classes
SRC = source/combase.cpp source/bndblock.cpp source/matrixblock.cpp source/
bndelement.cpp source/matblc1.cpp

# source files for compiler test
TRC = source/combase_cote.cpp source/bndblock.cpp source/matrixblock.cpp source/
bndelement.cpp source/matblc1.cpp -w

# Lines hereafter don’t need modification unless you want to add your own cases.

all: tutorial1 tutorial2 tutorial3 tutorial4 tutorial5
# A user defined case, if more are needed copy, paste, rename (i.e. userN) and adapt

user1: userfolder/main_user1.cpp
	$(CPP)
	@if [ $$? = 0 ]; then \
	  echo "  Compiling $@ done" ; \
	else \
	  echo " Compiling $@ failed" ; \
	fi

tutorial1:
	$(CPP)
	@if [ $$? = 0 ]; then \
	  echo "  Compiling $@ done" ; \
	else \
	  echo " Compiling $@ failed" ; \
	fi
```

```
# Makefile to compile Ulambator, 01.12.2015
# The sharp (#) symbols indicate comment, which are not executed

# Please adapt the following lines
# compiler of choice: g++, icc, etc.
CPP = g++

# compiler options for g++ with Xcode
OPTS = -lstdc++ -framework Accelerate -O3 -lpthread

# compiler options for g++ with Lapack libraries
# OPTS = -fopenmp -lstdc++ -llapack -lblas -lpthread

# compiler options for icc and mkl (Intel)
# OPTS = -lstdc++ -mkl -O3 -openmp

# library path for instance for LAPACK, if needed
# LIBS = -L/opt/intel/composer_xe_2013.1.119/mkl/lib/

# include path for headers, if needed
# INCS = -I/Developer/SDKs/MacOSX10.7.sdk/usr/lib/gcc/i686-apple-darwin11/4.2.1/include
# That is all, only edit the lines below for new source files
# or to add user defined cases to the compiler (example: user1)

# source files except main classes
SRC = source/combase.cpp source/bndblock.cpp source/matrixblock.cpp source/bndelement.cpp source/matblc1.cpp

# source files for compiler test
TRC = source/combase_cote.cpp source/bndblock.cpp source/matrixblock.cpp source/bndelement.cpp source/matblc1.cpp -w

# Lines hereafter don't need modification unless you want to add your own cases.

all: tutorial1 tutorial2 tutorial3 tutorial4 tutorial5
# A user defined case, if more are needed copy, paste, rename (i.e. userN) and adapt

user1: userfolder/main_user1.cpp
	$(CPP) -o user1.o userfolder/main_user1.cpp $(LIBS) $(INCS) $(SRC) $(OPTS)
	@if [ $$? = 0 ]; then \
	  echo "  Compiling $@ done" ; \
	else \
	  echo " Compiling $@ failed" ; \
	fi

tutorial1: tutorial/tutorial1.cpp
	$(CPP) -o tutorial1.o tutorial/tutorial1.cpp $(LIBS) $(INCS) $(SRC) $(OPTS)
	@if [ $$? = 0 ]; then \
	  echo "  Compiling $@ done" ; \
	else \
	  echo " Compiling $@ failed" ; \
	fi
```

Source Code

Commented and formatted code is found online
or as a pdf at lfmi.epfl.ch/ulamsource



After the break …

How to:

★ Set up a geometry

★ Set up a liquid interface

★ Define flow rates and material parameters

★ Solve the problem