

U L a M B A t o R

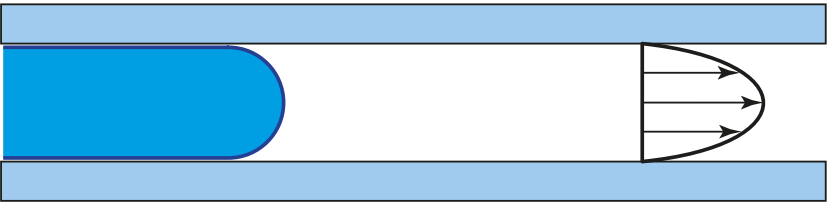
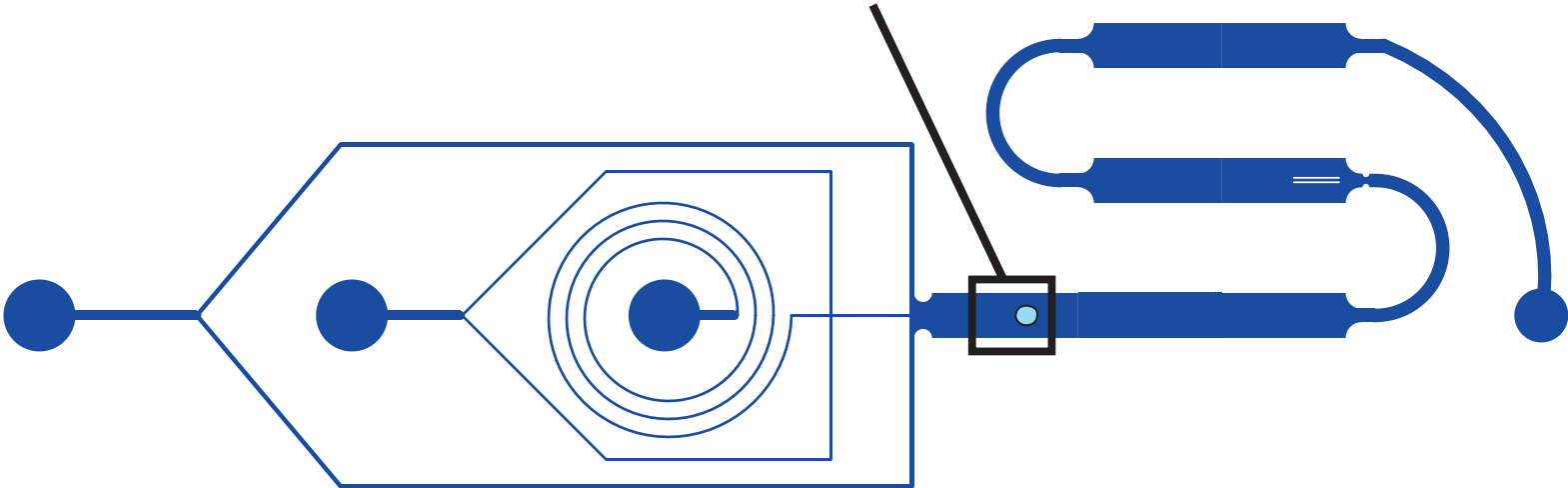


1. Workshop (part II)

December 11th 2015
TIPs Lab



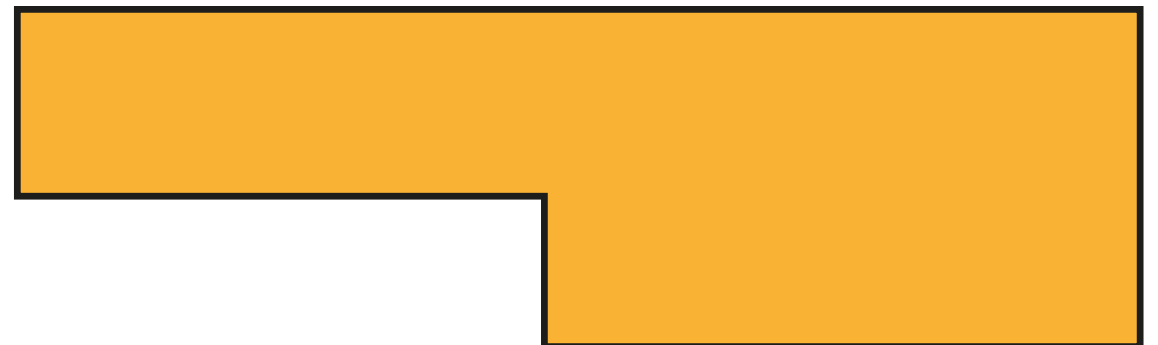
Geometry building



Geometry building

- Invent a 2D geometry
(the height is specified later)
- How many interfaces will there be? (Solid and Liquid)

BndBlock *my_bem*(N);



Geometry building

- Invent a 2D geometry
(the height is specified later)
- How many interfaces will there be? (Solid and Liquid)

BndBlock *my_bem*(N);

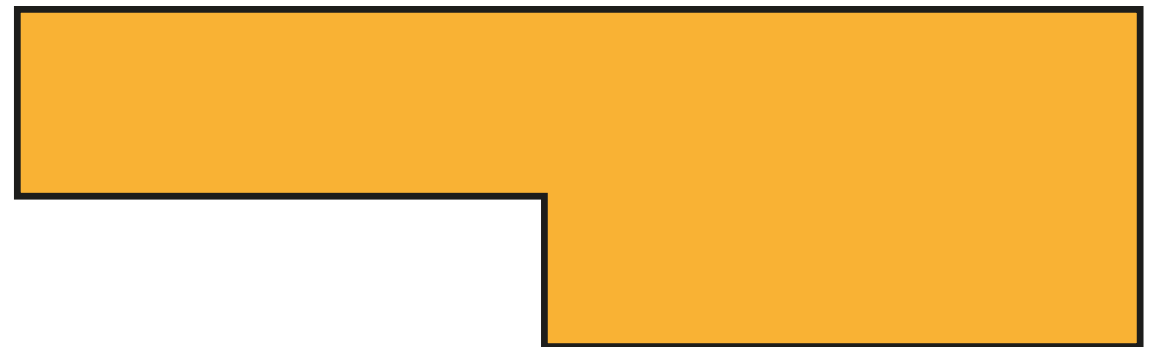
- Of how many panels consists the boundary? How much points would they use at most? (Panels are used to decompose the geometry and separate boundary conditions)

my_bem.**Elements**[ID].**Init**(M,P,0);



Geometry building

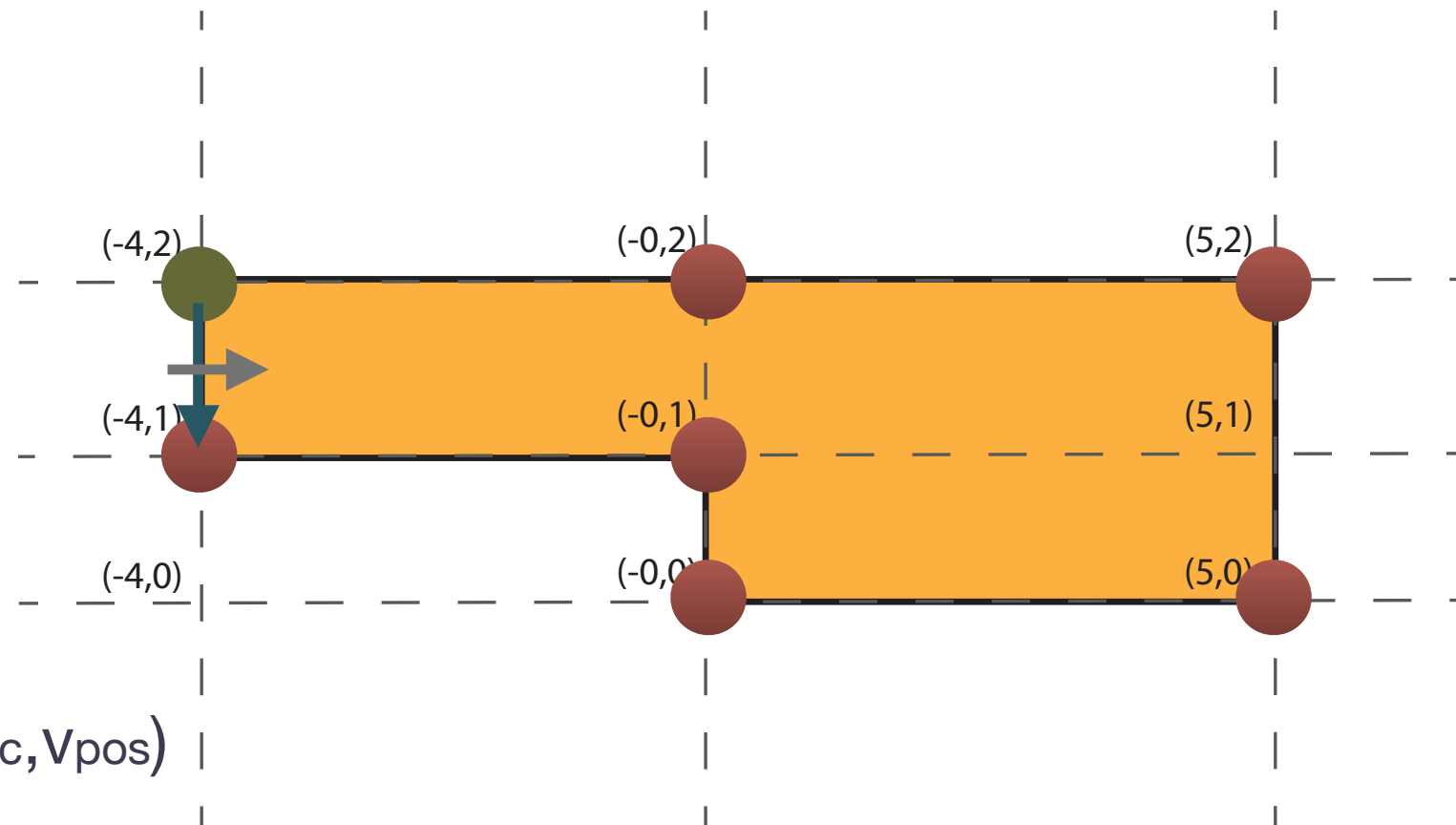
- Chose a length scale L for non-dimensionalization
- Transform the 2D geometry onto a non-dimensional grid



Geometry building

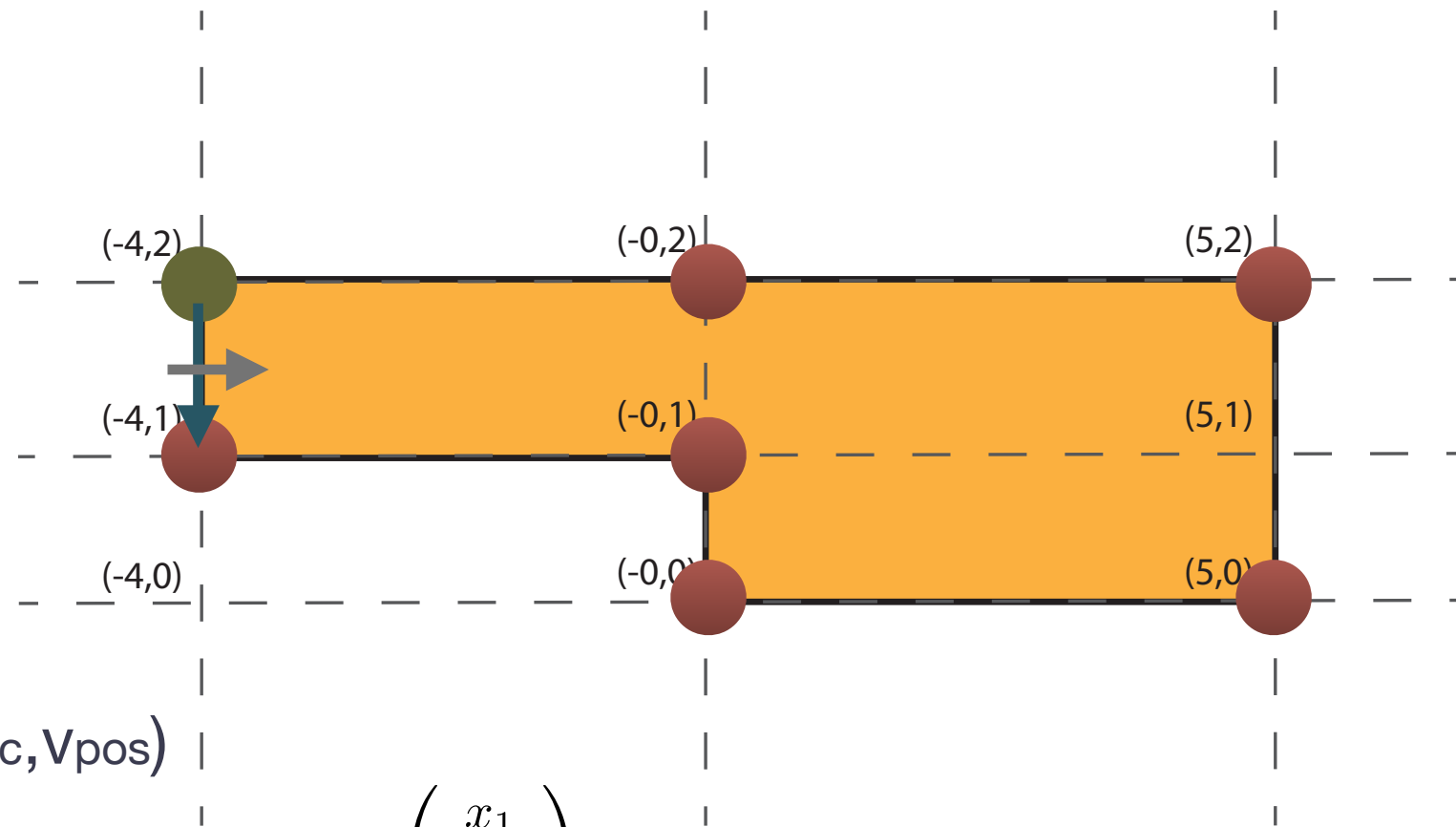
- Chose a length scale L for non-dimensionalization
- Transform the 2D geometry onto a non-dimensional grid
- Define discrete panels (not necessarily straight lines)
- Initialize each panel in left hand sense
(to the left side of the panel is the domain
 \Rightarrow counterclockwise for enclosed domain)

my_bem.**PanelInit**(ID,P_l,type,bc,v_{bc},v_{pos})



Geometry building

- Chose a length scale L for non-dimensionalization
- Transform the 2D geometry onto a non-dimensional grid
- Define discrete panels (not necessarily straight lines)
- Initialize each panel in left hand sense
(to the left side of the panel is the domain
=> counterclockwise for enclosed domain)



my_bem.**PanelInit**(ID,P_l,type,bc,v_{bc},v_{pos})

type = 0

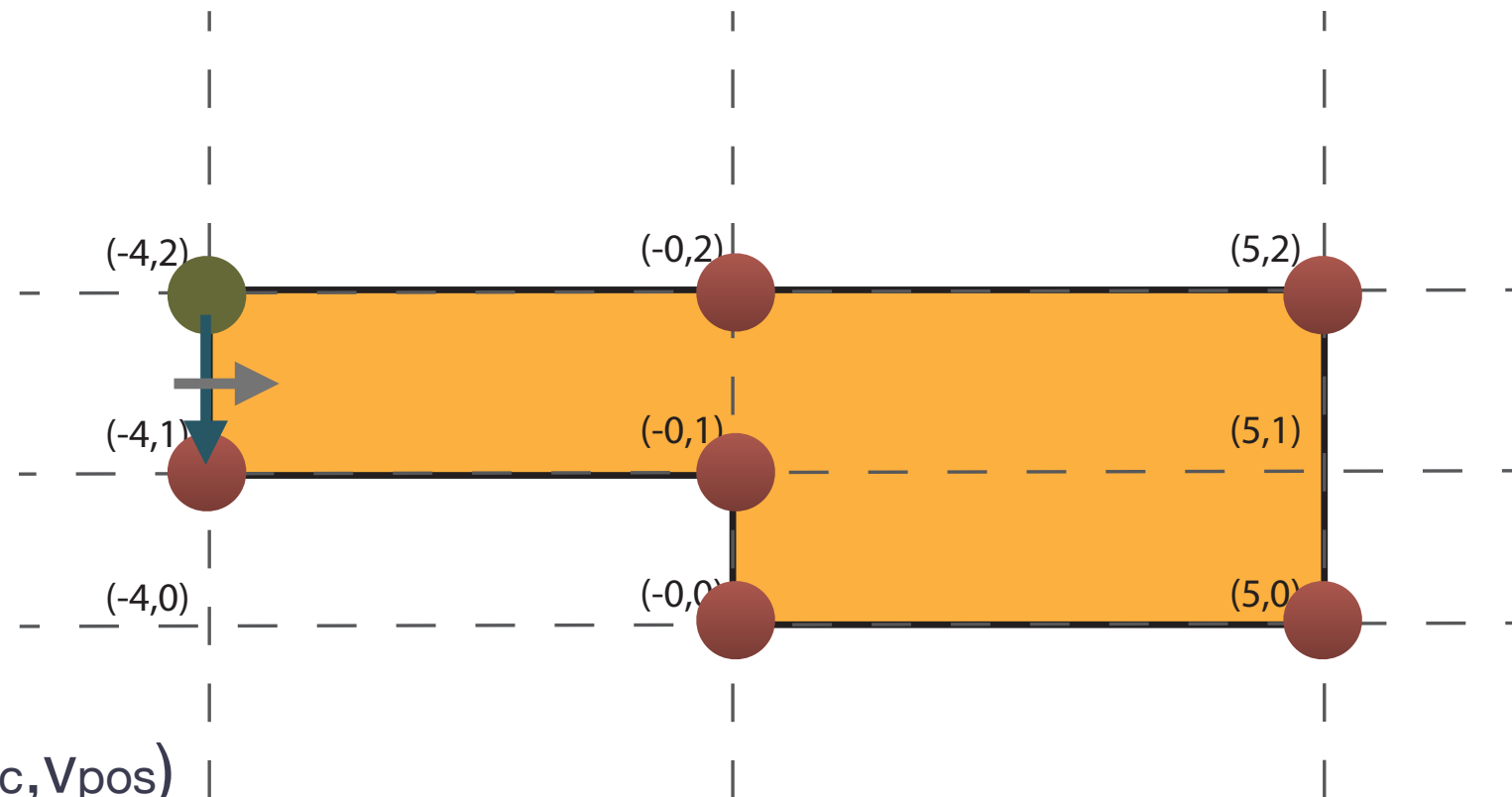
bc = 1

$$v_{bc} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

$$v_{pos} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix}$$

Geometry building

- Chose a length scale L for non-dimensionalization
- Transform the 2D geometry onto a non-dimensional grid
- Define discrete panels (not necessarily straight lines)
- Initialize each panel in left hand sense
(to the left side of the panel is the domain
=> counterclockwise for enclosed domain)



```
my_bem.PanelInit(ID,Pi,type,bc,vbc,vpos)
```

```
bem.Elements[0].Init(12, 2200, 0);
```

```
pos[0] = -6; pos[1]= 1.0; pos[2]= -6.0; pos[3] = -1.0;
```

```
bem.PanelInit(0,100, 0, 3, bcs, pos); // an inflow panel
```


Geometry building

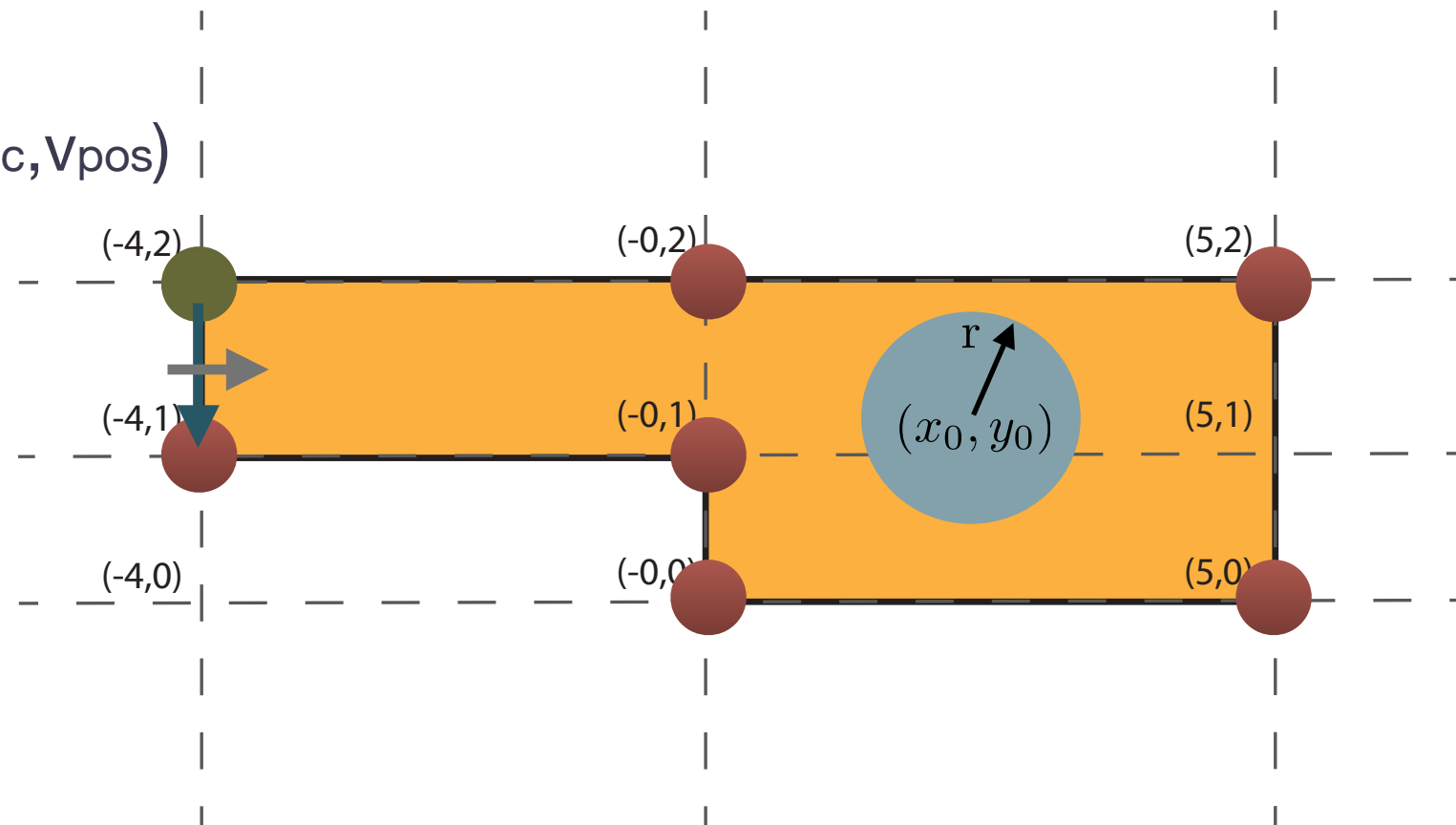
- Continue with other solid boundaries
- Continue with liquid interfaces

`my_bem.Elements[ID].Init(M,P,2);`

`my_bem.PanelInit(ID,Pi,type,bc,vbc,vpos)`

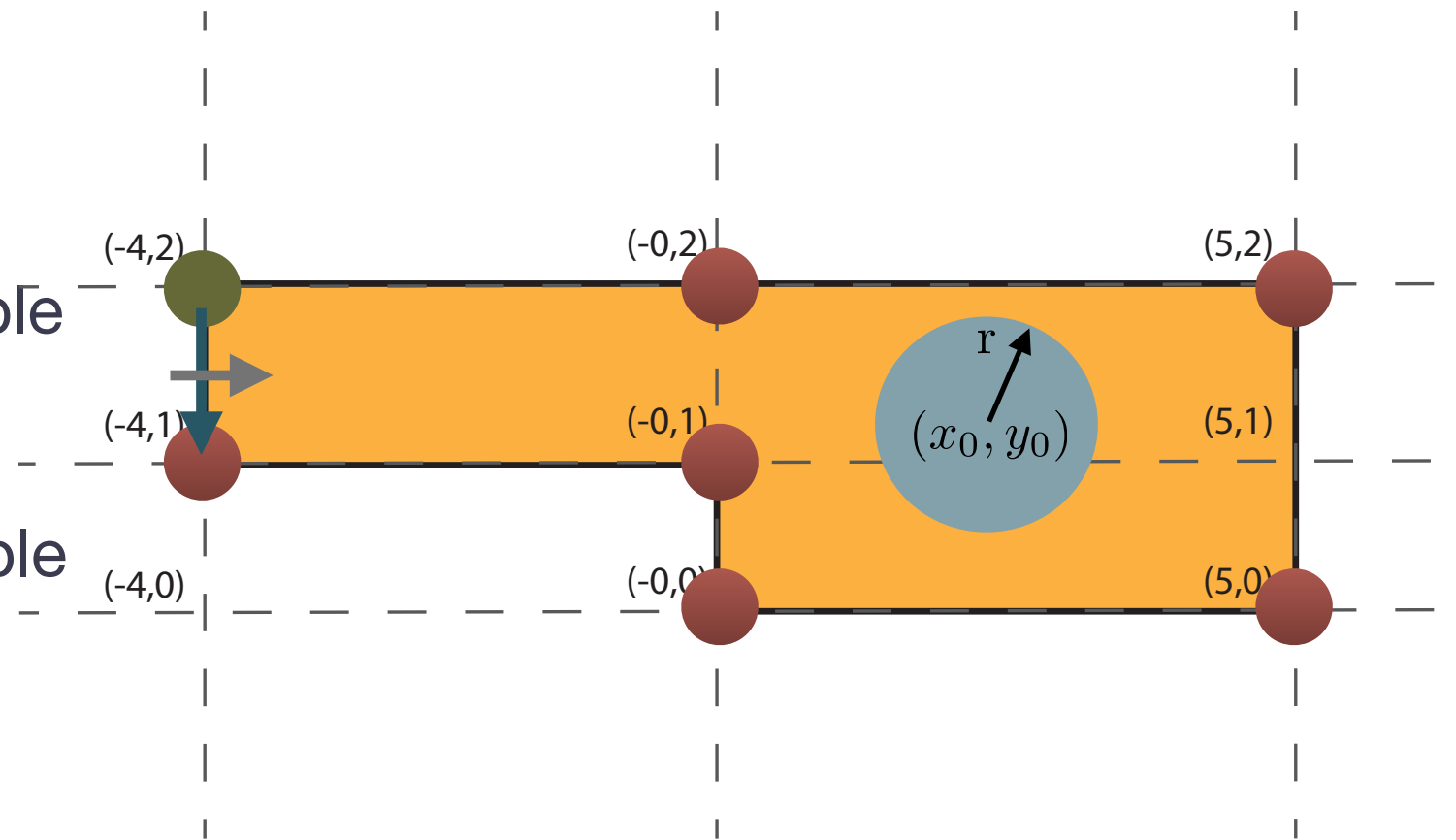
$$\text{type} = 2 \quad \text{bc} = 21 \quad v_{bc} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$v_{pos} = \begin{pmatrix} x_0 \\ y_0 \\ r \\ r_{\text{perturb}} \\ k_{\text{perturb}} \end{pmatrix}$$

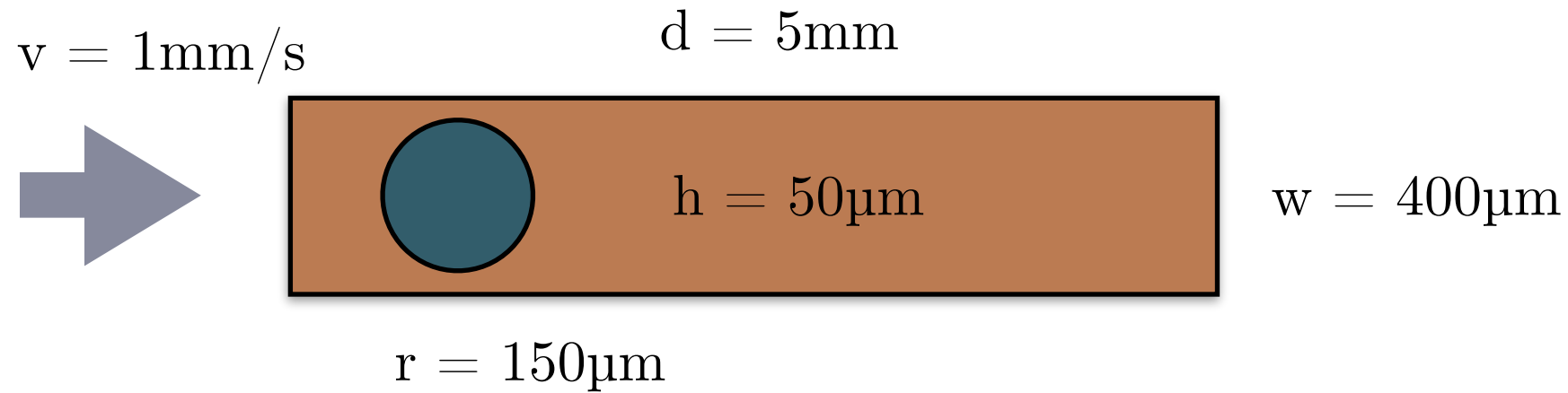


Geometry building

- More complicated shapes are possible
- Reading in geometries from point data is possible
- Remeshing is available
- On the fly manipulation is possible
- Liquid interfaces that are in contact with the walls are possible



Non-dimensionalization

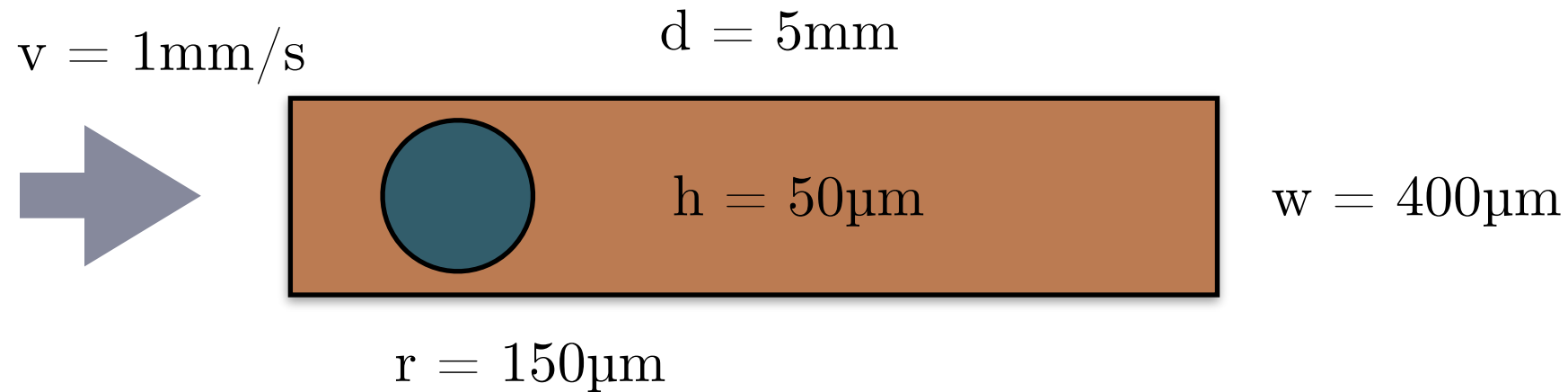


$$\eta = 4\text{mPas}$$

$$\eta_d = 1\text{mPas}$$

$$\sigma_{\alpha\beta} = 12\text{mPam}$$

Non-dimensionalization



$$\eta = 4\text{mPas}$$

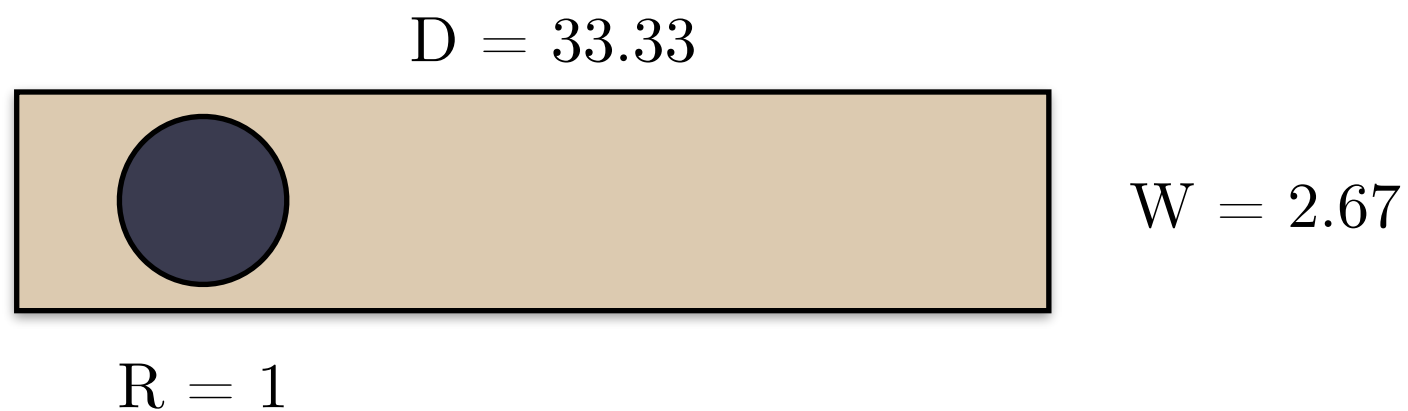
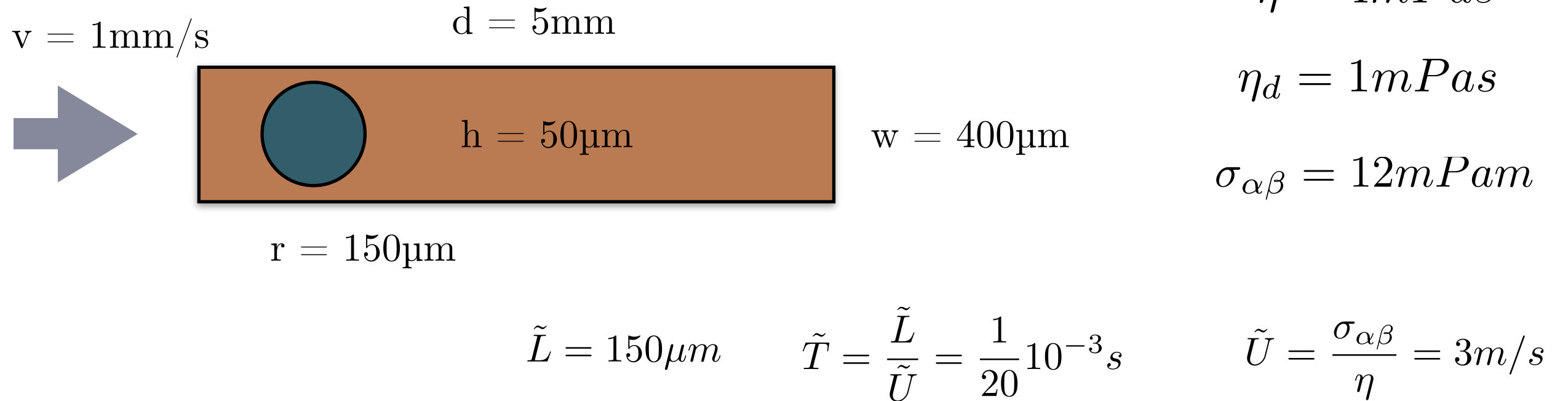
$$\eta_d = 1\text{mPas}$$

$$\sigma_{\alpha\beta} = 12\text{mPam}$$

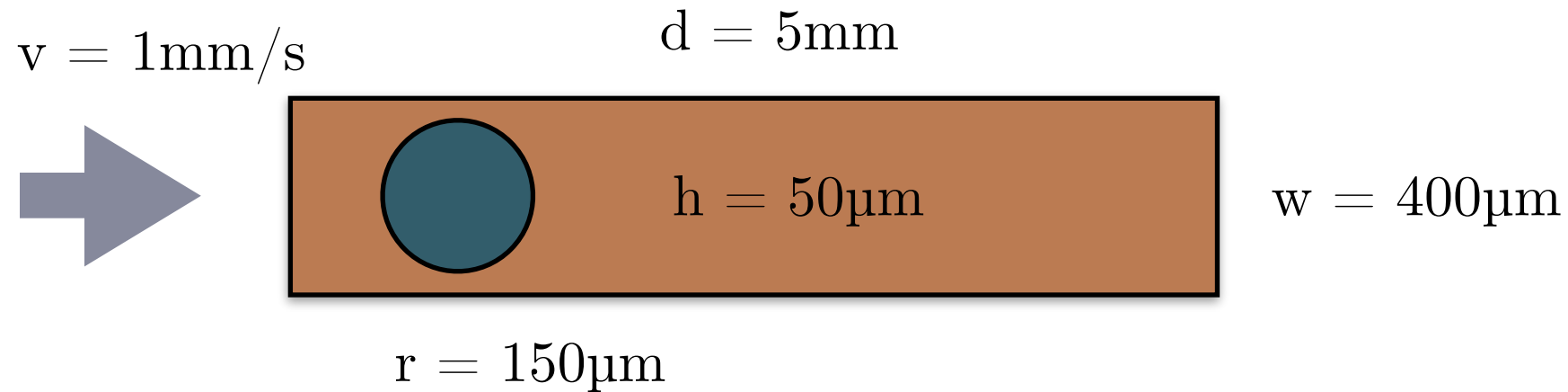
$$\tilde{L} = 150\mu\text{m} \quad \tilde{T} = \frac{\tilde{L}}{\tilde{U}} = \frac{1}{20}10^{-3}\text{s}$$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta} = 3\text{m/s}$$

Non-dimensionalization



Non-dimensionalization



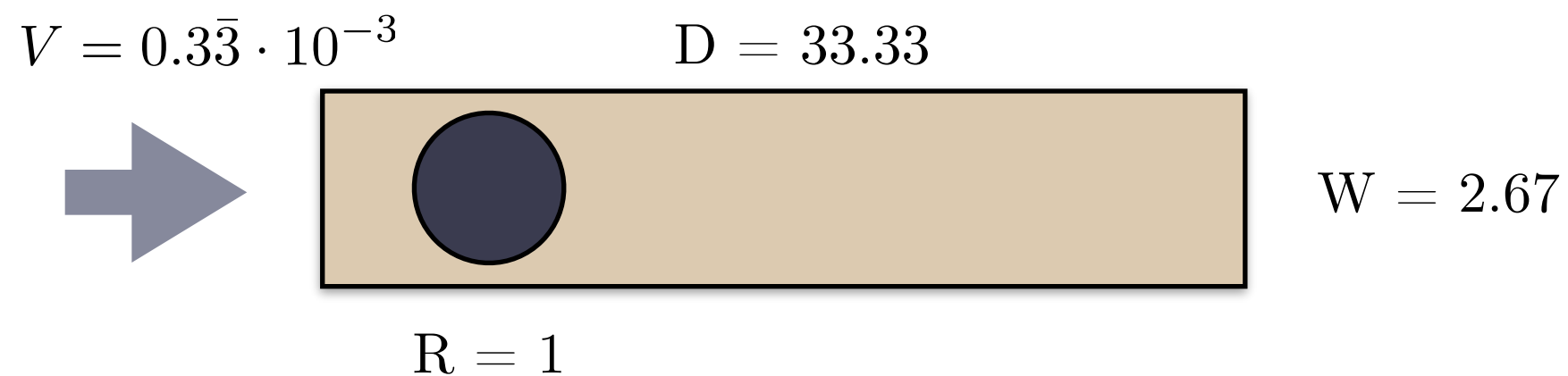
$$\eta = 4 \text{ mPas}$$

$$\eta_d = 1 \text{ mPas}$$

$$\sigma_{\alpha\beta} = 12 \text{ mPa}$$

$$\tilde{L} = 150 \mu\text{m} \quad \tilde{T} = \frac{\tilde{L}}{\tilde{U}} = \frac{1}{20} 10^{-3} \text{ s}$$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta} = 3 \text{ m/s}$$

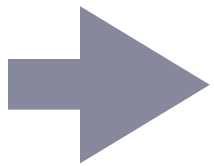


$$\lambda = \frac{\eta_d}{\eta} = 1/4$$

$$R/H = 3$$

Non-dimensionalization

$$v = 1 \text{ mm/s}$$



$$\eta = 4 \text{ mPas}$$

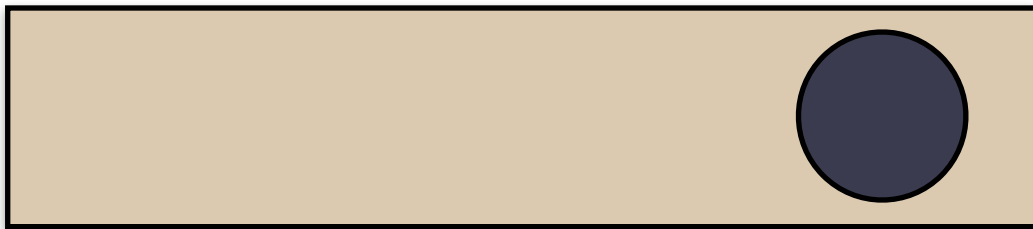
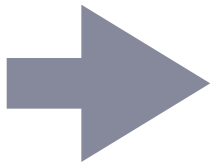
$$\eta_d = 1 \text{ mPas}$$

$$\sigma_{\alpha\beta} = 12 \text{ mPam}$$

$$\tilde{L} = 150 \mu\text{m} \quad \tilde{T} = \frac{\tilde{L}}{\tilde{U}} = \frac{1}{20} 10^{-3} \text{ s}$$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta} = 3 \text{ m/s}$$

$$V = 0.3\bar{3} \cdot 10^{-3}$$

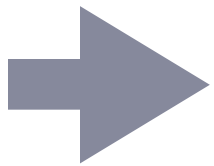


$$\lambda = \frac{\eta_d}{\eta} = 1/4$$

... at $T=3000$

Non-dimensionalization

$$v = 1 \text{ mm/s}$$



$$\eta = 4 \text{ mPas}$$

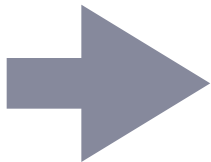
$$\eta_d = 1 \text{ mPas}$$

$$\sigma_{\alpha\beta} = 12 \text{ mPam}$$

$$\tilde{L} = 150 \mu\text{m} \quad \tilde{T} = \frac{\tilde{L}}{\tilde{U}} = \frac{1}{20} 10^{-3} \text{ s}$$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta} = 3 \text{ m/s}$$

$$V = 0.3\bar{3} \cdot 10^{-3}$$

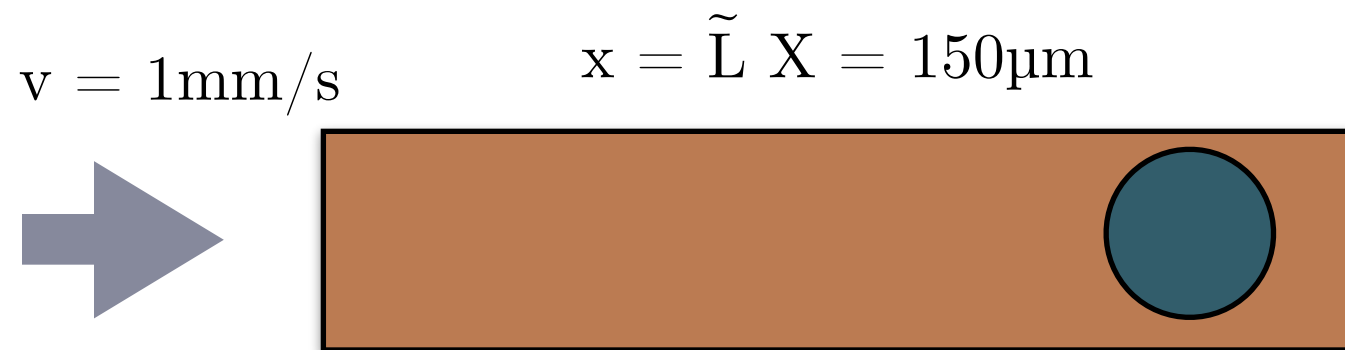


$$X = 1$$

$$\lambda = \frac{\eta_d}{\eta} = 1/4$$

... at T=3000

Non-dimensionalization



$$\eta = 4 \text{ mPas}$$

$$\eta_d = 1 \text{ mPas}$$

$$\sigma_{\alpha\beta} = 12 \text{ mPa}$$

$$\tilde{L} = 150 \mu\text{m} \quad \tilde{T} = \frac{\tilde{L}}{\tilde{U}} = \frac{1}{20} 10^{-3} \text{ s}$$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta} = 3 \text{ m/s}$$

$$V = 0.3\bar{3} \cdot 10^{-3}$$

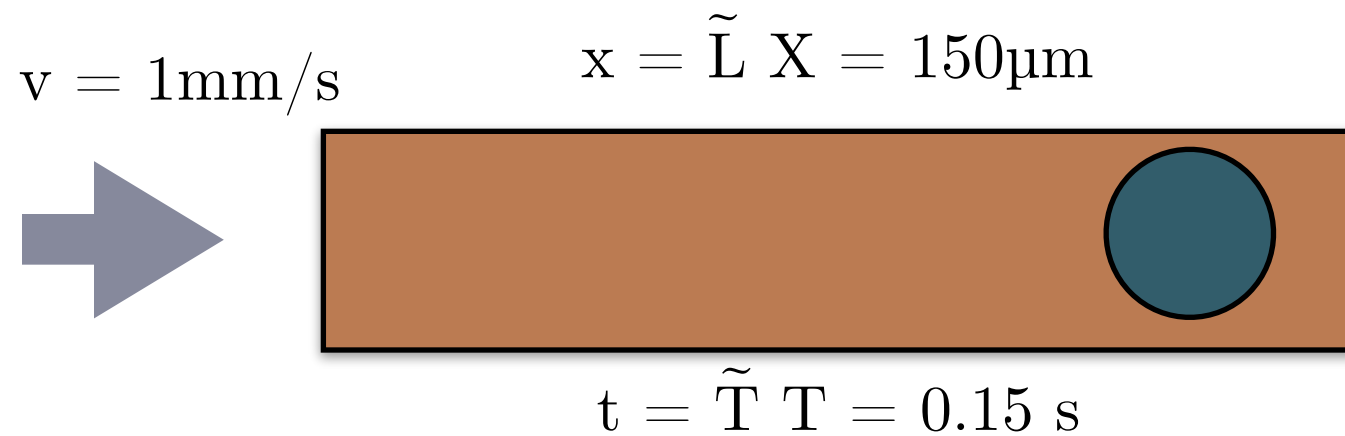


$$X = 1$$

$$\lambda = \frac{\eta_d}{\eta} = 1/4$$

... at $T=3000$

Non-dimensionalization



$$\eta = 4 \text{ mPas}$$

$$\eta_d = 1 \text{ mPas}$$

$$\sigma_{\alpha\beta} = 12 \text{ mPa}$$

$$\tilde{L} = 150 \mu\text{m} \quad \tilde{T} = \frac{\tilde{L}}{\tilde{U}} = \frac{1}{20} 10^{-3} \text{ s}$$

$$\tilde{U} = \frac{\sigma_{\alpha\beta}}{\eta} = 3 \text{ m/s}$$

$$V = 0.3\bar{3} \cdot 10^{-3}$$



$$X = 1$$

$$\lambda = \frac{\eta_d}{\eta} = 1/4$$

... at $T=3000$

Matrix building

- What is the height of the channels?
The viscosity ratio?

MatBIC1 *my_mat*(&*my_bem*,L/H,lamda,Ca);

$$\lambda = \frac{\eta_d}{\eta}$$

- Build and solve with given geometry and boundary conditions

my_mat.**Build**(0);

my_mat.**Solve**();

Matrix building

- What is the height of the channels?
The viscosity ratio?

MatBIC1 *my_mat*(&*my_bem*,L/H,lamda,Ca);

$$\lambda = \frac{\eta_d}{\eta}$$

- Build and solve with given geometry and boundary conditions

my_mat.**Build**(0);

my_mat.**Solve**();

$$\begin{pmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{pmatrix} \begin{pmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \\ \vec{u}_4 \end{pmatrix} = \begin{pmatrix} \vec{G}_1 \cdot \vec{f} \\ \vec{G}_2 \cdot \vec{f} \\ \vec{G}_3 \cdot \vec{f} \\ \vec{G}_4 \cdot \vec{f} \end{pmatrix}$$

Matrix building

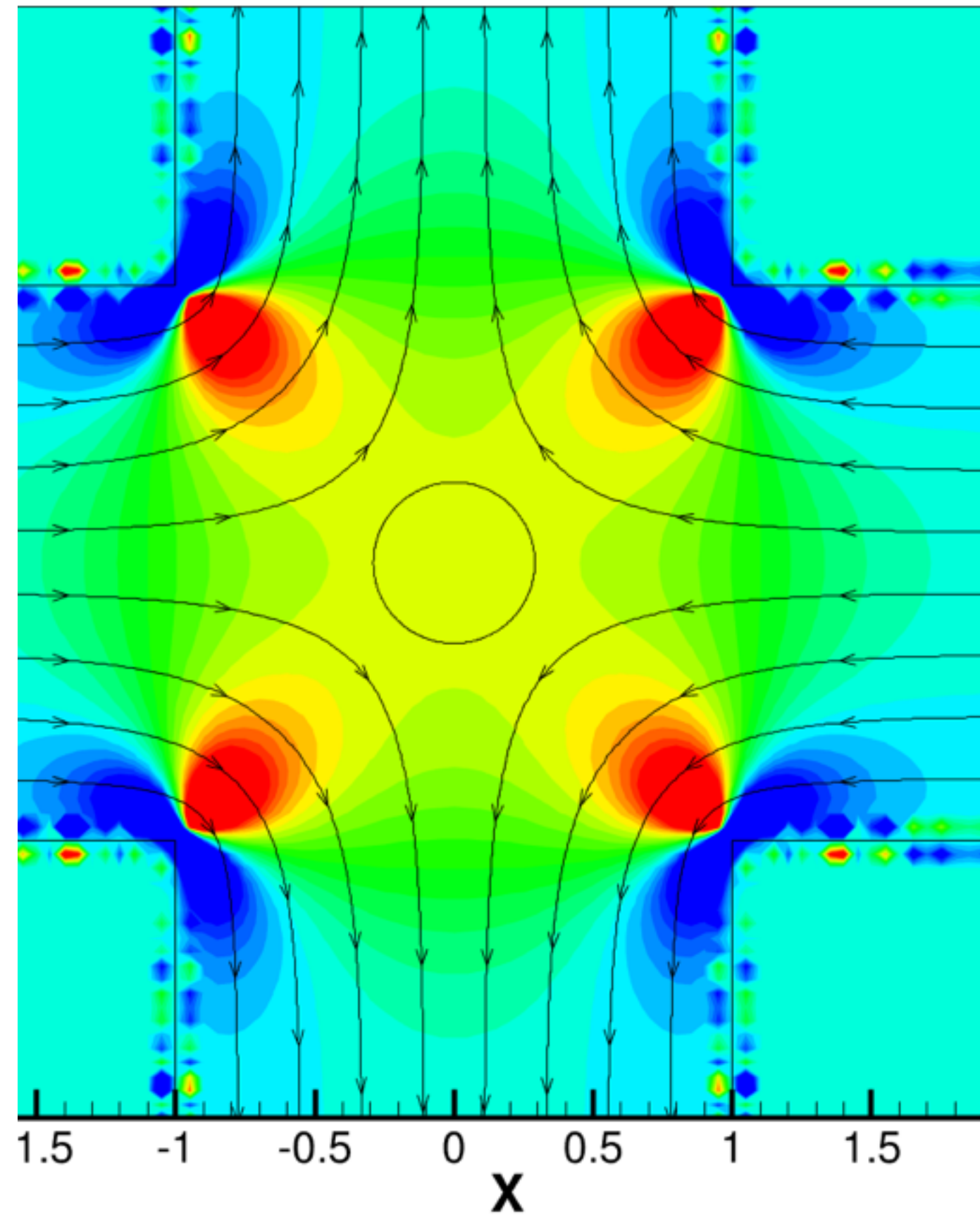
- More advances time stepping routines are available
- Particular liquid interface boundary conditions are available
- Interfacial stabilization for larger time steps is available

Matrix building

- More advances time stepping routines are available
- Particular liquid interface boundary conditions are available
- Interfacial stabilization for larger time steps is available

$$\begin{pmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{pmatrix} \begin{pmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \\ \vec{u}_4 \end{pmatrix} = \begin{pmatrix} \vec{G}_1 \cdot \vec{f} \\ \vec{G}_2 \cdot \vec{f} \\ \vec{G}_3 \cdot \vec{f} \\ \vec{G}_4 \cdot \vec{f} \end{pmatrix}$$

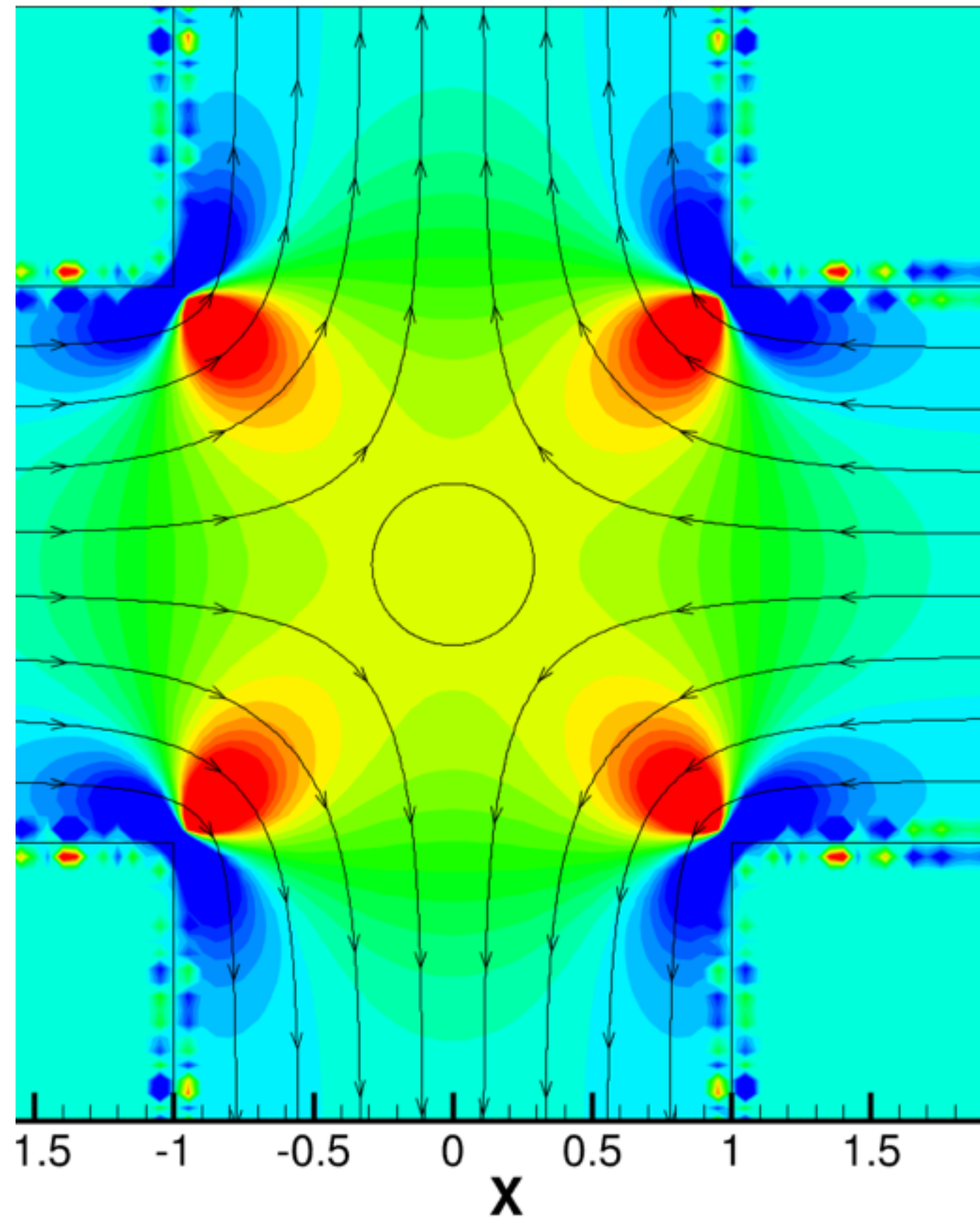
Output generation



Output generation

- Specify a location on the disk

my_mat.**EnableWrite**(*directory*);



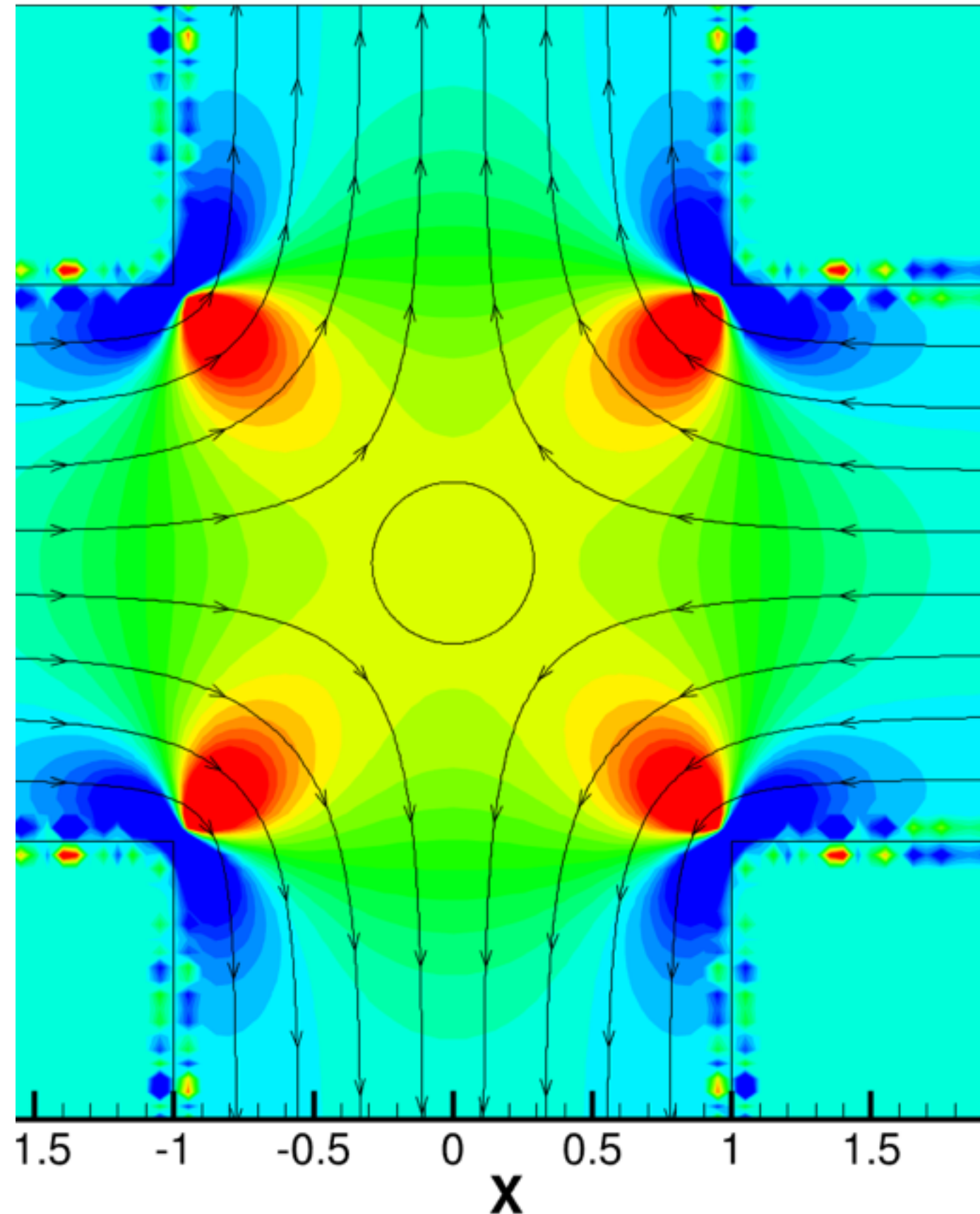
Output generation

- Specify a location on the disk

```
my_mat.EnableWrite(directory);
```

- Write solid wall to file
(point coordinates)

```
my_mat.WritePos(ID, T);
```



Output generation

- Specify a location on the disk

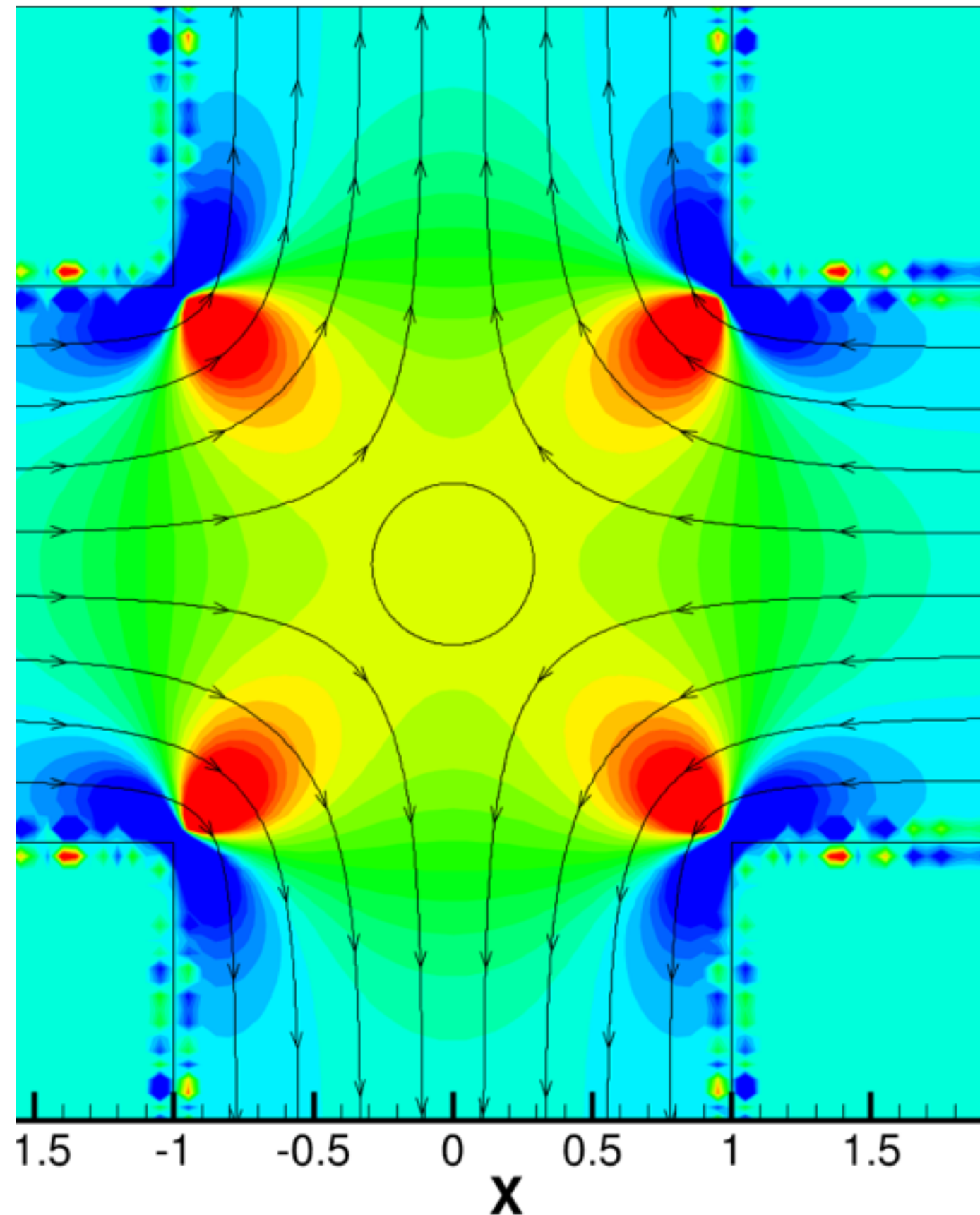
```
my_mat.EnableWrite(directory);
```

- Write solid wall to file
(point coordinates)

```
my_mat.WritePos(ID, T);
```

- Write liquid interface to file
(points and curvature)

```
my_mat.WritePos(ID, T);
```



Output generation

- Specify a location on the disk

```
my_mat.EnableWrite(directory);
```

- Write solid wall to file
(point coordinates)

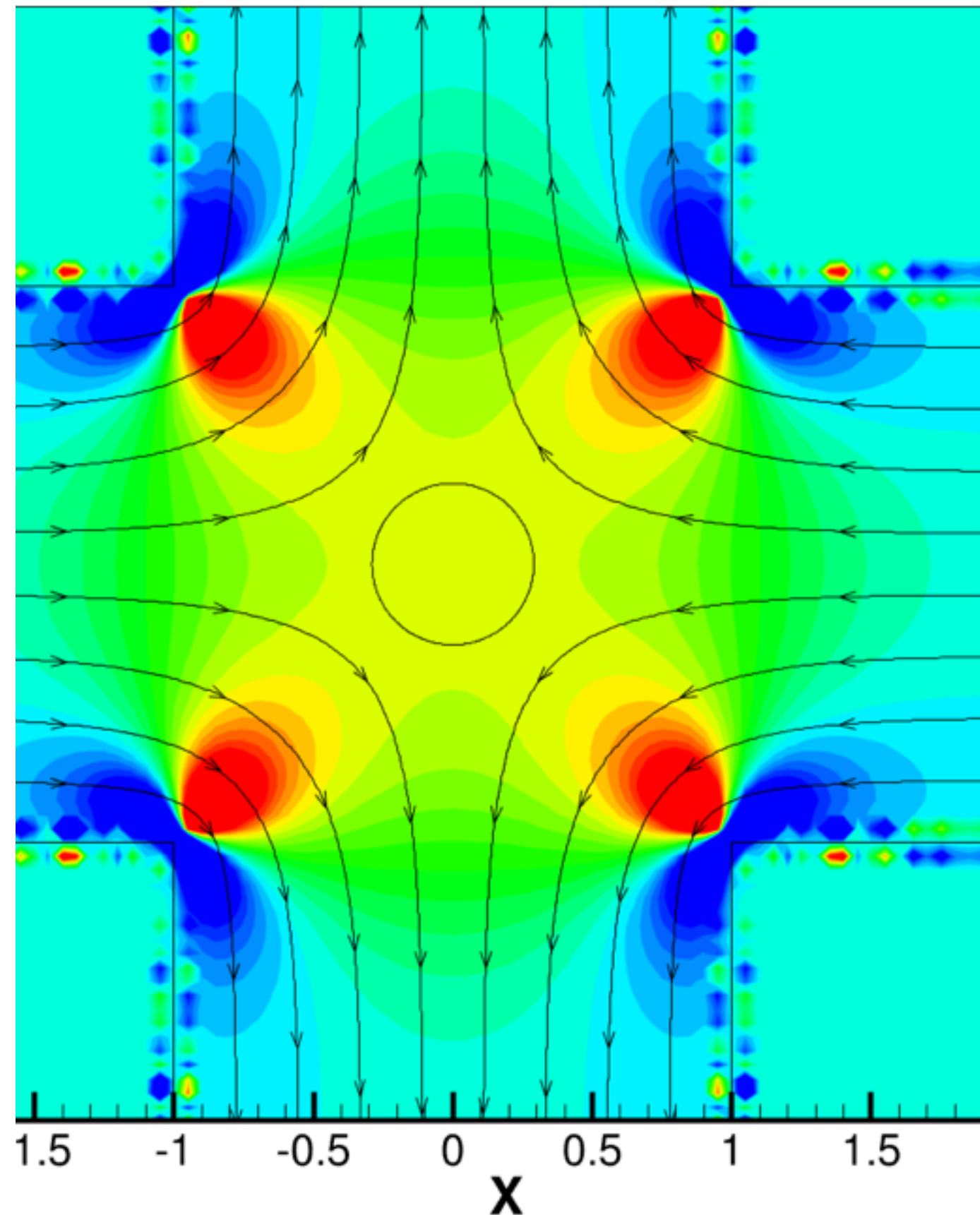
```
my_mat.WritePos(ID, T);
```

- Write liquid interface to file
(points and curvature)

```
my_mat.WritePos(ID, T);
```

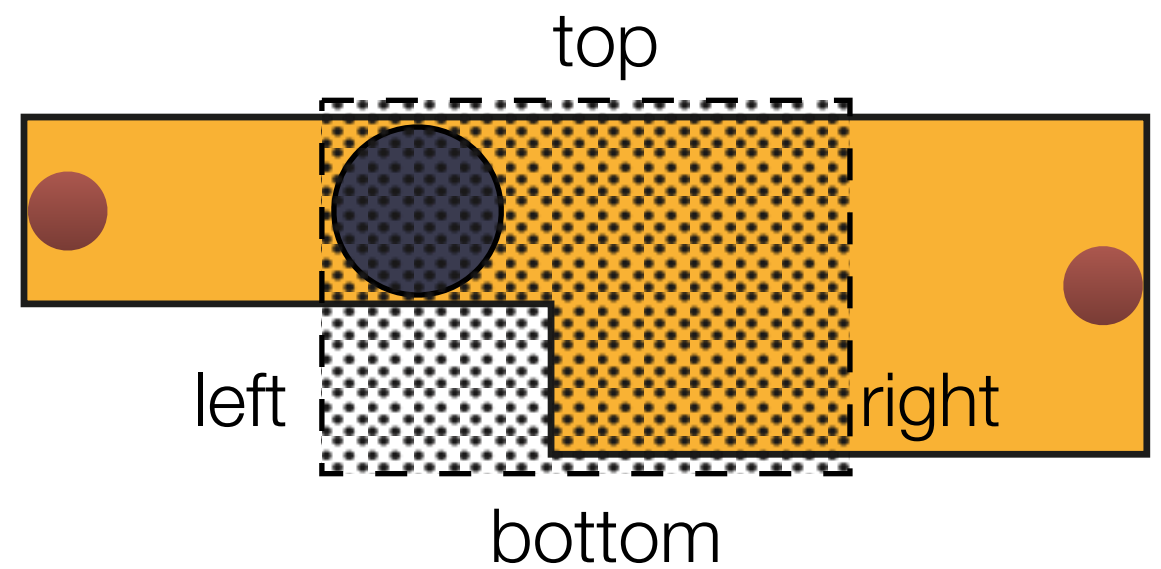
- Write all interfaces at once

```
my_mat.allWrite( );
```



Output generation

Time step T
point density M



Sensor

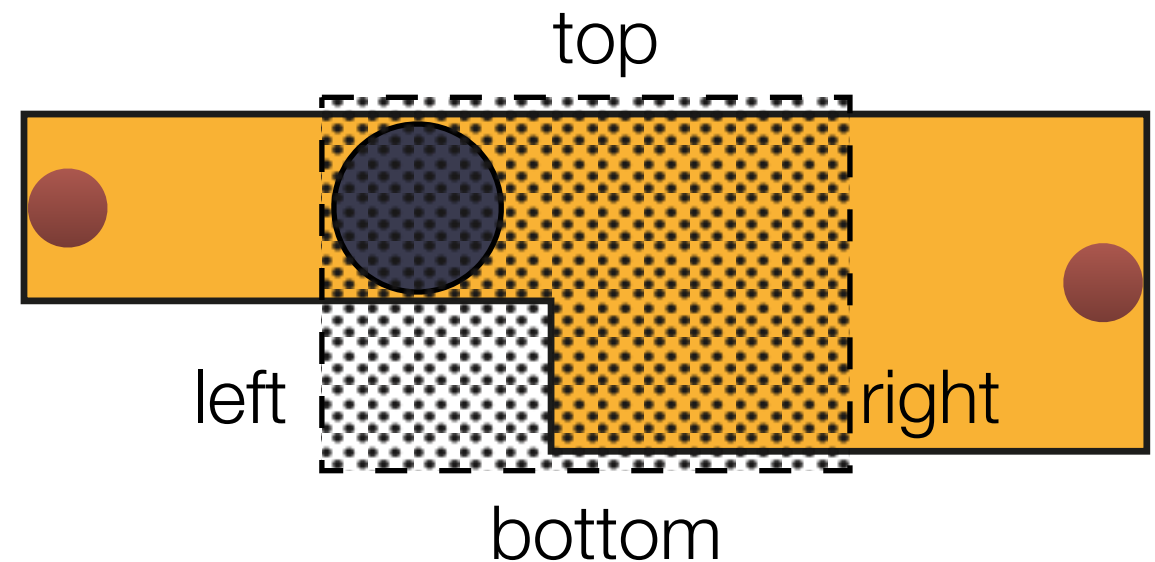
Output generation

- Write velocity and pressure fields to *.vtk or Matlab file.

```
my_mat.VTKexport(T,M,lt,rt,bt,tp);
```

```
my_mat.VisMatlab(T,M,lt,rt,bt,tp);
```

Time step T
point density M



Sensor

Output generation

- Write velocity and pressure fields to *.vtk or Matlab file.

```
my_mat.VTKexport(T,M,lt,rt,bt,tp);
```

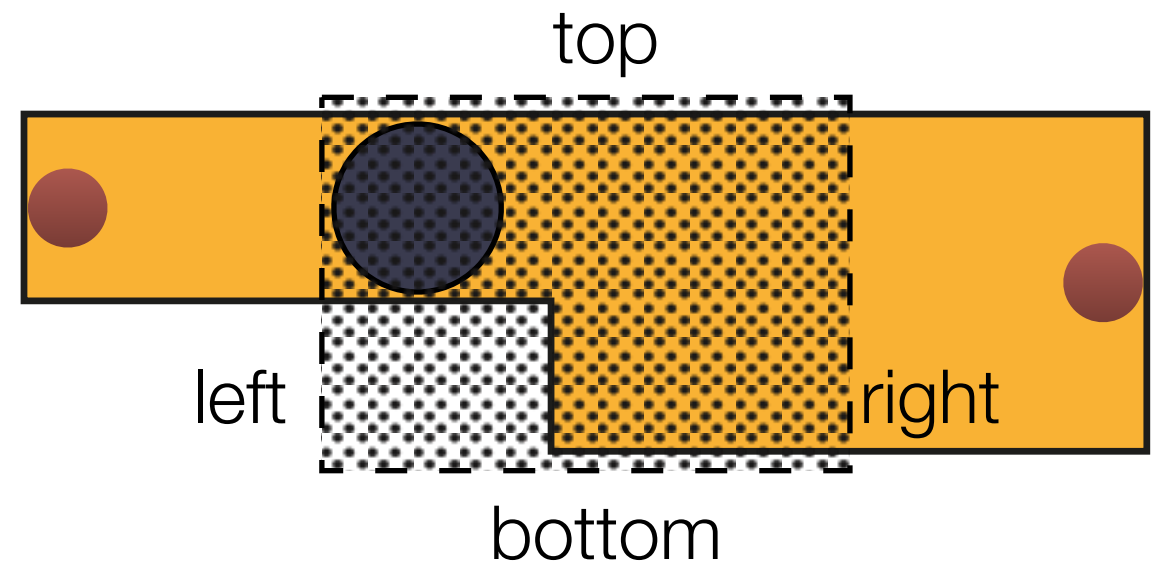
```
my_mat.VisMatlab(T,M,lt,rt,bt,tp);
```

- Capture the field in certain points by placing sensors (to record u,v,p in one spot)

```
my_mat.SensorEnable(ID);
```

```
my_mat.SensorWrite(ID,x,y);
```

Time step T
point density M

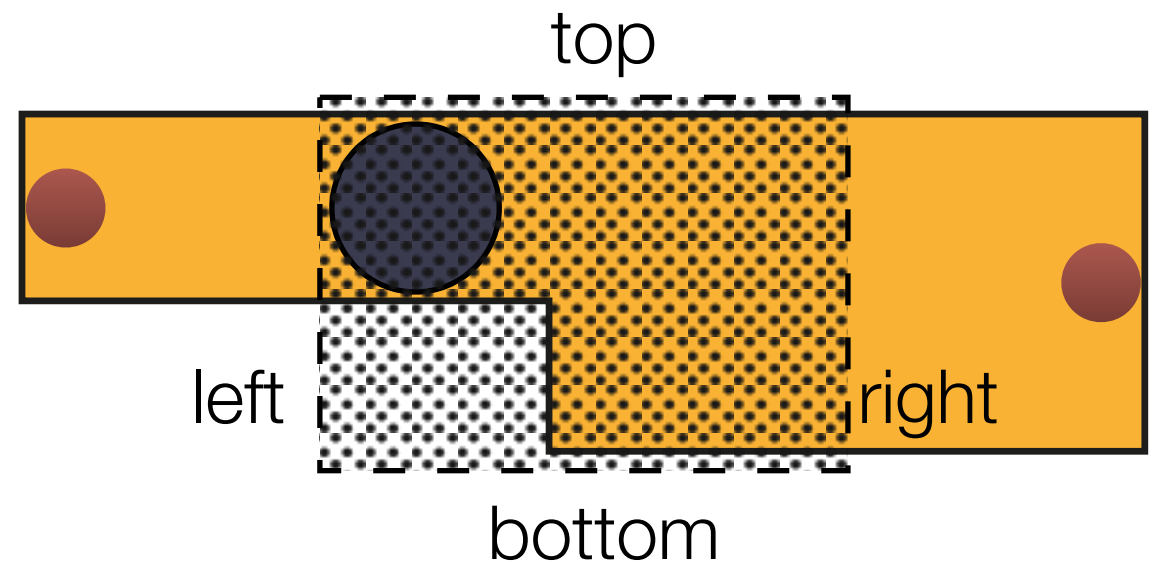


Sensor

Output generation

- Customized output is possible
- Logging droplet positions is available
- Writing the Matrix and the right hand side or solution to a file is possible.

Time step T
point density M



Sensor

Examples and their particularities

```
/*  
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;
```

It's C++ code,
there are headers
and the program starts with
`int main()`

Examples and their particularities

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

File Header

Together with included classes we add Ulbators `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>
```

You can pass parameters
when executing the file

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

```
int main (int argc, char * const argv[]) {
```

Here the program catches the numbers that were passed when it was called eg.

`./ulbator_main.o 1 5 0.5` sets the `folderid=1`, `LH=5`, `eccentricity=0.5`

```
    if (argc>3) {
        folderid = atof(argv[1]);
        LH = atof(argv[2]);
        eccentricity = atof(argv[3]);
    }
```

Examples and their particularities

Directories need string or char variables
Subfolders can be created

```
savedir = new char[120];  
sprintf(savedir, ".");  
mat.EnableWrite(savedir);
```

If a `folderid` bigger than 0 is provided the following code will create a subfolder in `savedir` is created. The subfolder will be named "test" followed by a number. If the folder already exist the program is stopped to prevent overwriting.

```
if (folderid>0) {  
    sprintf(savedir, "test%d", folderid);  
    mat.SubFolder(savedir);  
}
```

Examples and their particularities

In order to preserve the mesh quality
the interface can be remeshed
and the surface area be preserved

Remeshing is activated with standard density according to method 0 (default, homogeneous distribution).
The initial area of boundary is saved and maintained.

```
bem.RemeshOn(2*M_PI*rdrop/num_elements, 0);  
bem.Elements[1].initialArea = bem.getArea(1);
```

User Reference

ulambator.sourceforge.net

SOURCEforge

Search

BrowseEnterpriseBlogJobsDealsHelp

SOLUTION CENTERS

Go Parallel

Resources

Newsletters

Home / Browse / Projects / Ulambator / Code

Ulambator

Alpha

boundary element based flow solver for microfluidics

Brought to you by: [gengiskhan](#)

SummaryFilesReviewsSupportCodeUlambator MailingListDiscussionBlog

Browse Commits

Fork

Branches

master

Tree [\[ae61b2\]](#) [master](#) /

Download Snapshot

Histo

RO

HTTP

Read Only access


git clone git://git.code.sf.net/p/ulambator/code ulambator-code

File	Date	Author	Commit
source	2015-06-24	nagel	[ae61b2] resolve error in matrixblock.cpp
tools	2015-02-24	nagel	[78f1d8] tutorial 4+5 added
tutorial	2015-03-04	nagel	[92781c] change in tutorial3
userfolder	2014-12-21	nagel	[de9241] tutorial 3 added
validation	2015-02-24	nagel	[78f1d8] tutorial 4+5 added
Makefile	2015-06-24	nagel	[ae61b2] resolve error in matrixblock.cpp
README.txt	2014-12-21	nagel	[de9241] tutorial 3 added

User Reference

ulambator.sourceforge.net

lfmi.epfl.ch/ulamsource



YOU ARE

BY SCHOOL

ABOUT EPFL

Direct

Help

EPFL > STI > IGM > LFMI > Research > Microfluidics > Simulation of Complex Microfluidic Circuits > Ulambator source > Tutorial 2

Hon

U

bo

Bro

People

Research

Publications




Seminars

Teaching

Student projects

Related links

Ryhming Prize

Share:     

Compile and run

In order to run the first example go into the ulambator folder and type 'make tutorial2'. An executable will be created. Type 'cd .' and copy it to a work directory of your choice then type './tutorial2.o' to execute the program.

Series of interface data 'drop0tsnnn.dat' and 'ulamgeo2dnnn.vtk' and velocity and pressure fields will be written in 'ulam2dnnn.vtk' or 'ulamviznn.dat'. The VTK files can be used with ParaView and DAT files can be plotted with the Matlab.

This executable is configurable from the command line, type './tutorial2.o 1 4 0.0' and you will simulate a steady droplet in subfolder 'test1' with an aspect ration radius/channel height of 4 and zero eccentricity. When you measure the pressure inside the droplet you will see that it corresponds to $p=2*4+\pi/4$, which follows from Laplaces law. Where $\pi/4$ comes from the in-plane curvature, which is 1 because the radius is 1 and $\pi/4$ come from a correction by Park and Homsy (reference in our article) for flattened droplets. And where the out-of-plane curvature $\kappa=2/h$, here $h = 1/4$.

Tutorial 2 - Droplet relaxation

In this tutorial you will learn how to create a liquid interface and let it evolve in time. To this purpose an elliptical interface is created and a time marching follows the reaxation of the droplet.

```
/*
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"
```

Microfluidics

Simulation of Complex Microfluidic Circuits

Ulambator source

Tutorial 1

Tutorial 2

Tutorial 3

Tutorial 4

Tutorial 5

Confined Jets

Instability and control

Swirling jets, vortex breakdown

Download Snapshot

History

ulambator-code

atrxblock.cpp

3

atrxblock.cpp

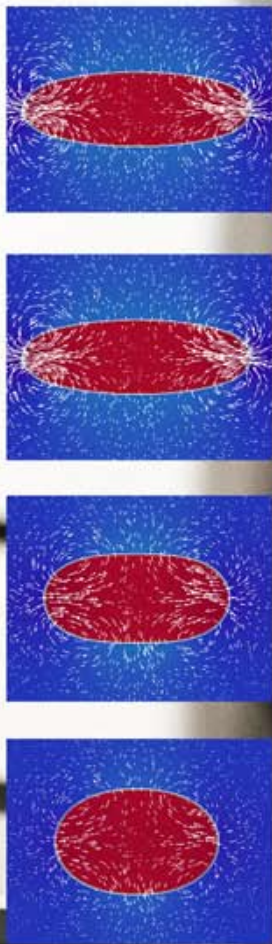
User Reference

ulambator.sourceforge.net

lfmi.epfl.ch/ulamsource



THE INCOMPLETE
ULAMBATOR REFERENCE



YOU ARE

BY SCHOOL

ABOUT EPFL

EPFL > STI > IGM > LFMI > Research > Microfluidics > Simulation of Complex Microfluidic Circuits > Ulambator source > Tutorial

LABORATORY OF FLUID MECHANICS AND INSTABILITIES

People Research Publications Seminars Teaching Student projects Related links

Share: f t in g e

Compile and run

In order to run the first example go into the ulambator folder and type 'make tutorial2'. An executable will be created. Type 'cd .' and copy it to a work directory of your choice then type './tutorial2.o' to execute the program.

Series of interface data 'drop0tsnnn.dat' and 'ulamgeo2dnnn.vtk' and velocity and pressure fields will be written in 'ulam2dnnn.vtk' or 'ulamviznn.dat'. The VTK files can be used with ParaView and DAT files can be plotted with the Matlab.

This executable is configurable from the command line, type './tutorial2.o 1 4 0.0' and you will simulate a steady droplet in subfolder 'test1' with an aspect ratio radius/channel height of 4 and zero eccentricity. When you measure the pressure inside the droplet you will see that it corresponds to $p=2\gamma\kappa + \pi/4$, which follows from Laplace's law. Where $\pi/4$ comes from the in-plane curvature, which is 1 because the radius is 1 and $\pi/4$ come from a correction by Park and Homsy (reference in our article) for flattened droplets. And where the out-of-plane curvature $\kappa=2/h$, here $h = 1/4$.

Tutorial 2 - Droplet relaxation

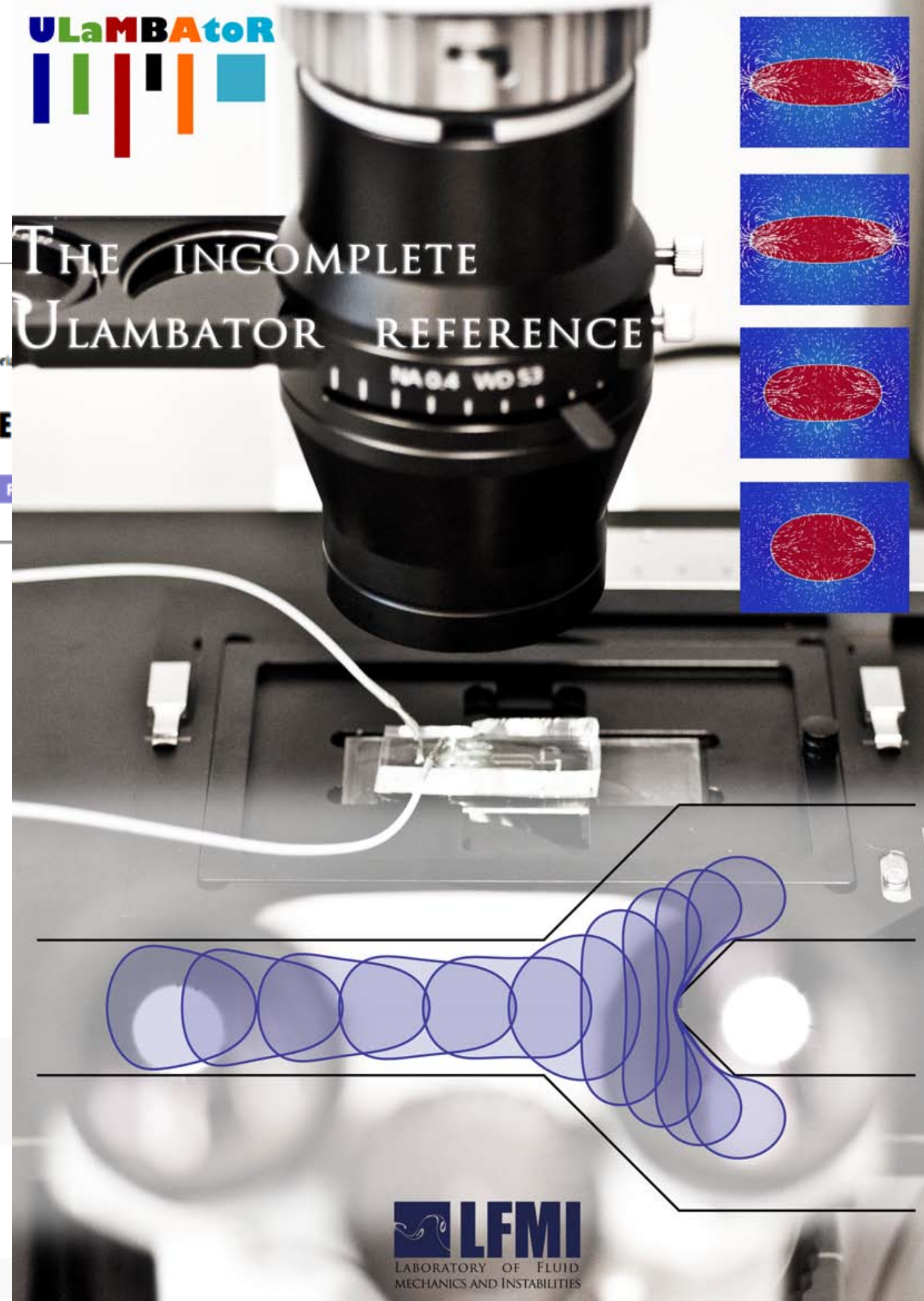
In this tutorial you will learn how to create a liquid interface and let it evolve in time. To this purpose an elliptical interface is created and a time marching follows the relaxation of the droplet.

```
/*
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"
```



Run some tutorials yourself

- type 'Make'
- type './tutorial1.o'
- Look at the result
- Change something in the code:
Geometry, confinement, viscosity ratio, velocity ...
- recompile with 'Make', run and look at the result

After the break ...

Look into:

- * Empirical hints
- * Data structures
- * Customized output routines
- * Time marching
- * **Your own cases**