

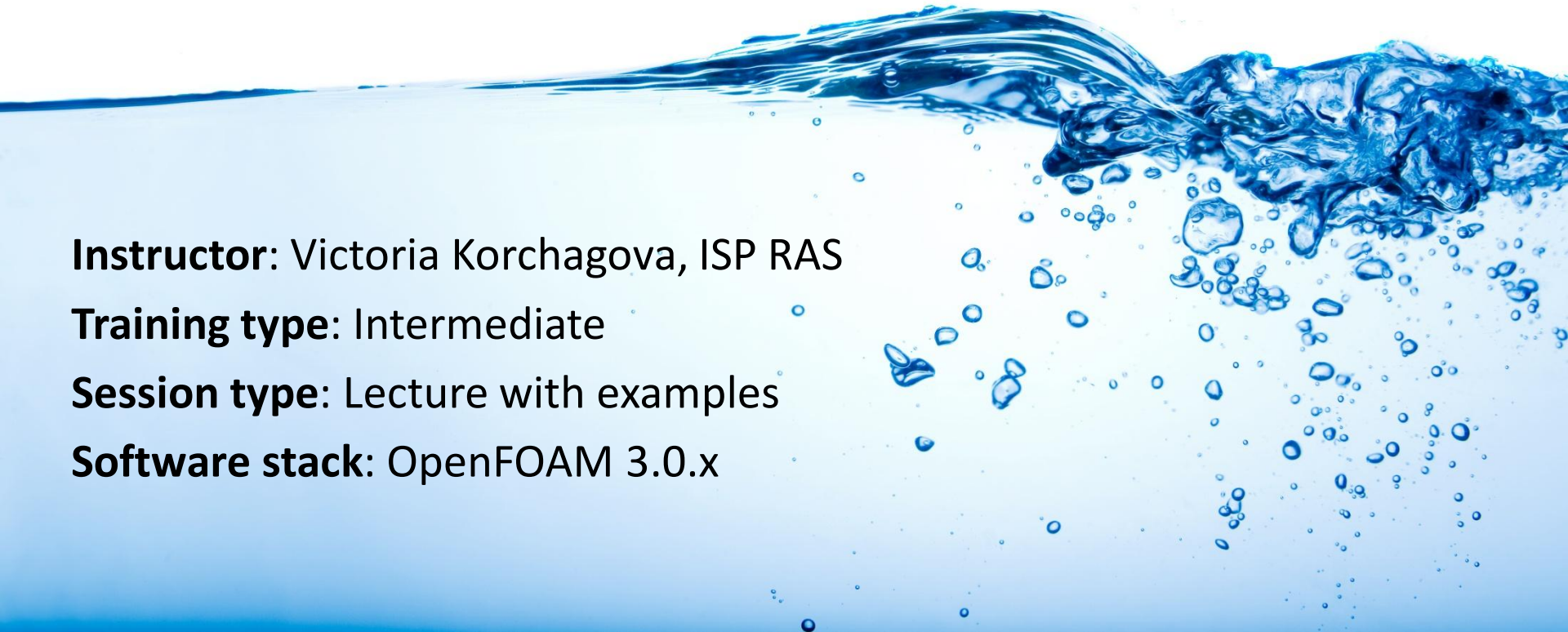
# How to use free-surface flows in OpenFOAM® 3.0

**Instructor:** Victoria Korchagova, ISP RAS

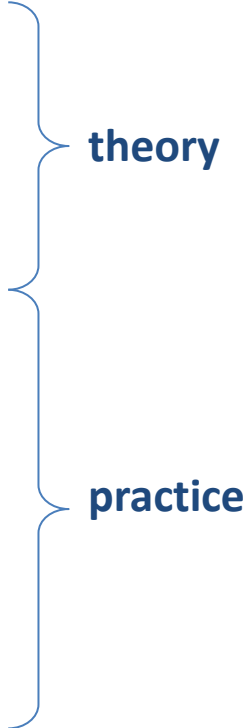
**Training type:** Intermediate

**Session type:** Lecture with examples

**Software stack:** OpenFOAM 3.0.x



# Plan of training course

- Introduction
  - Key points of training course
  - interFoam solver: how it works
    - Governing equations
    - Volume of Fluid method
    - Solution process
    - **Boundary conditions**
  - Spillway tutorial
    - Model setup
    - Physical properties setup
    - Mesh generation
    - **Boundary conditions setup**
    - Numerical schemes and time advancement settings. Running.
    - Results
  - Conclusions and discussion
- 
- The diagram consists of two blue curly braces on the right side of the list. The top brace groups the 'interFoam solver: how it works' item and its sub-points, with the word 'theory' in blue text to its right. The bottom brace groups the 'Spillway tutorial' item and its sub-points, with the word 'practice' in blue text to its right.

# Introduction

## Applicability:

- printing;
- engines;
- ecological cases;
- dams, spillways;
- etc.

## Complexities:

- large deformations of the interface;
- creation of different subregions (droplets, bubbles...);
- solution should:
  - be stable;
  - have small diffusivity in the interface region;
  - satisfy to conservation laws;
  - be correct in different scales (example – droplet impact to the liquid layer);
  - require not so much resources for computations.



# Key points of training course

The main key points of training track:

Look how a solver for free-surface flows works.

Study how to set boundary conditions in different versions of OpenFOAM.

Boundary conditions are critically important in the successful modeling.

There are strong differences between OpenFoam 2.2.x and 2.3.0+.

Another key point:

Look to all stages of modeling of free-surface flows

in OpenFOAM v.3.0.0:

from mesh generation to post-processing

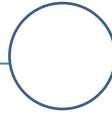
---

**The main tool:** an interFoam solver in OF v. 3.0.0.

We will study it with **Spillway tutorial**: the turbulent flow of fluid.

# Part I

Theoretical part



**interFoam:** how it works

# Structure of theoretical part

1. Governing equations.
2. Volume of Fluid method.
3. Block scheme of `interFoam`.
4. Pressure-velocity coupling.
5. Boundary conditions (most common types):
  1. walls and inlets;
  2. outlets;
  3. open boundaries.

# Governing equations for incompressible flow

- Continuity equation:

$$\nabla \cdot \vec{U} = 0;$$

- Navier — Stokes equations:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot (\rho \vec{U} \otimes \vec{U}) = -\nabla p + \nabla \cdot \hat{\tau} + \rho \vec{g};$$

where  $\hat{\tau} = \mu(\nabla \vec{U} + \nabla \vec{U}^T)$  — viscous stress tensor;

- boundary conditions on the interface:

$$[-p\mathbf{I} + \hat{\tau}] \cdot \vec{n} = \sigma \kappa \vec{n}; \quad [\vec{U}] = 0;$$

- initial and boundary conditions on the flow region boundaries (different types — walls, inlets, outlets, open boundaries).

# Volume of Fluid

Add a volume fraction function:

$$\alpha_1 = \begin{cases} 1 & \text{if cell full of liquid;} \\ 0 & \text{if cell full of gas;} \\ (0; 1) & \text{if cell is on the interface.} \end{cases}$$

For two phases

$$\alpha_1 + \alpha_2 = 1.$$

Solve a transport equation for this function:

$$\frac{\partial \alpha_1}{\partial t} + \nabla \cdot (\vec{U} \alpha_1) + \nabla \cdot (\vec{U}_R \alpha_1 \alpha_2) = 0,$$

where  $\vec{U} = \alpha_1 \vec{U}_1 + \alpha_2 \vec{U}_2$  — mixture velocity;

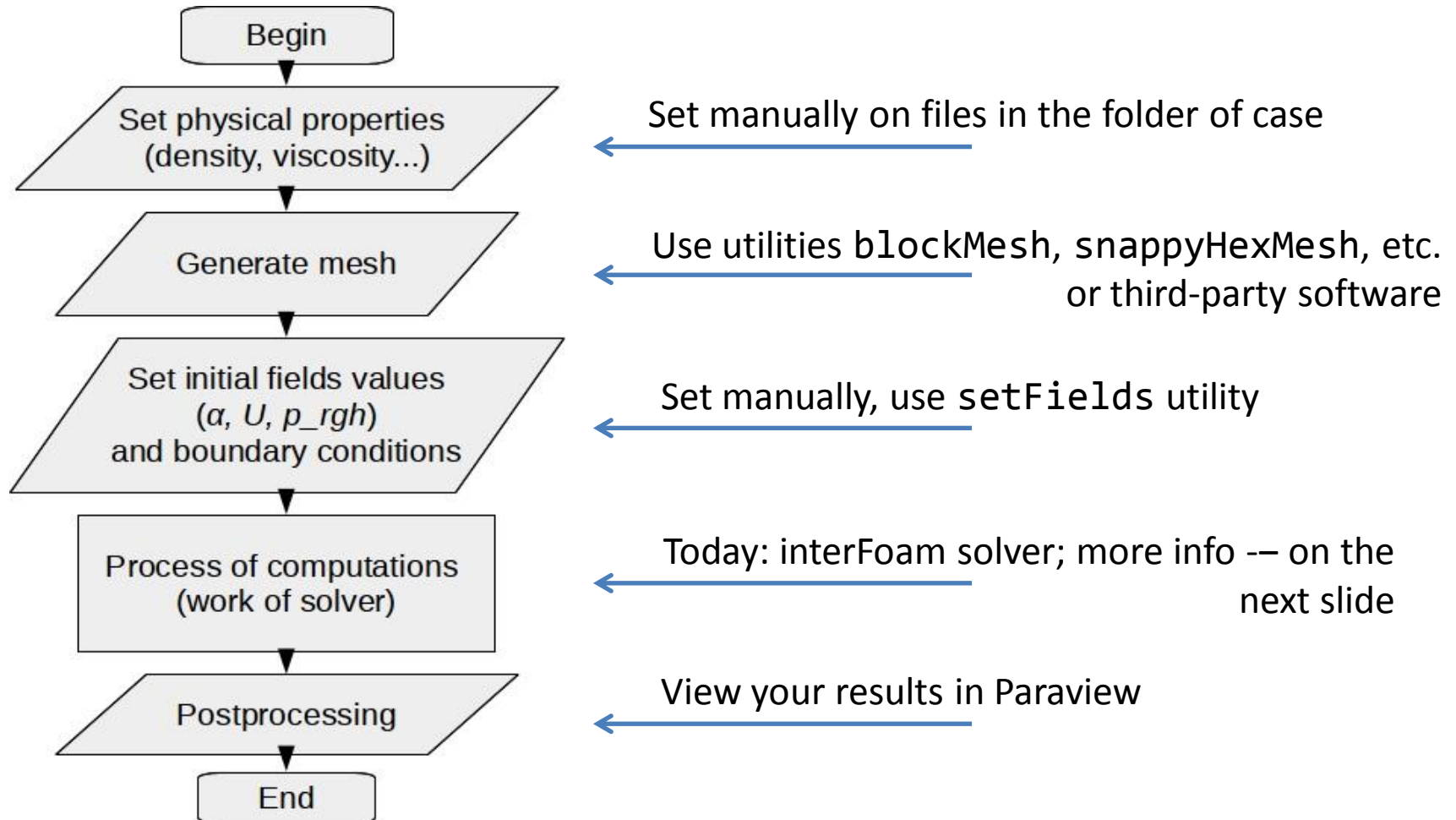
$\vec{U}_R = \vec{U}_1 - \vec{U}_2$  — velocity field suitable to compress the interface.

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.8	0.9	0.5	0	0	0
1	1	1	0.2	0	0
1	1	1	0.2	0	0

Approximation: **Finite Volume Method**

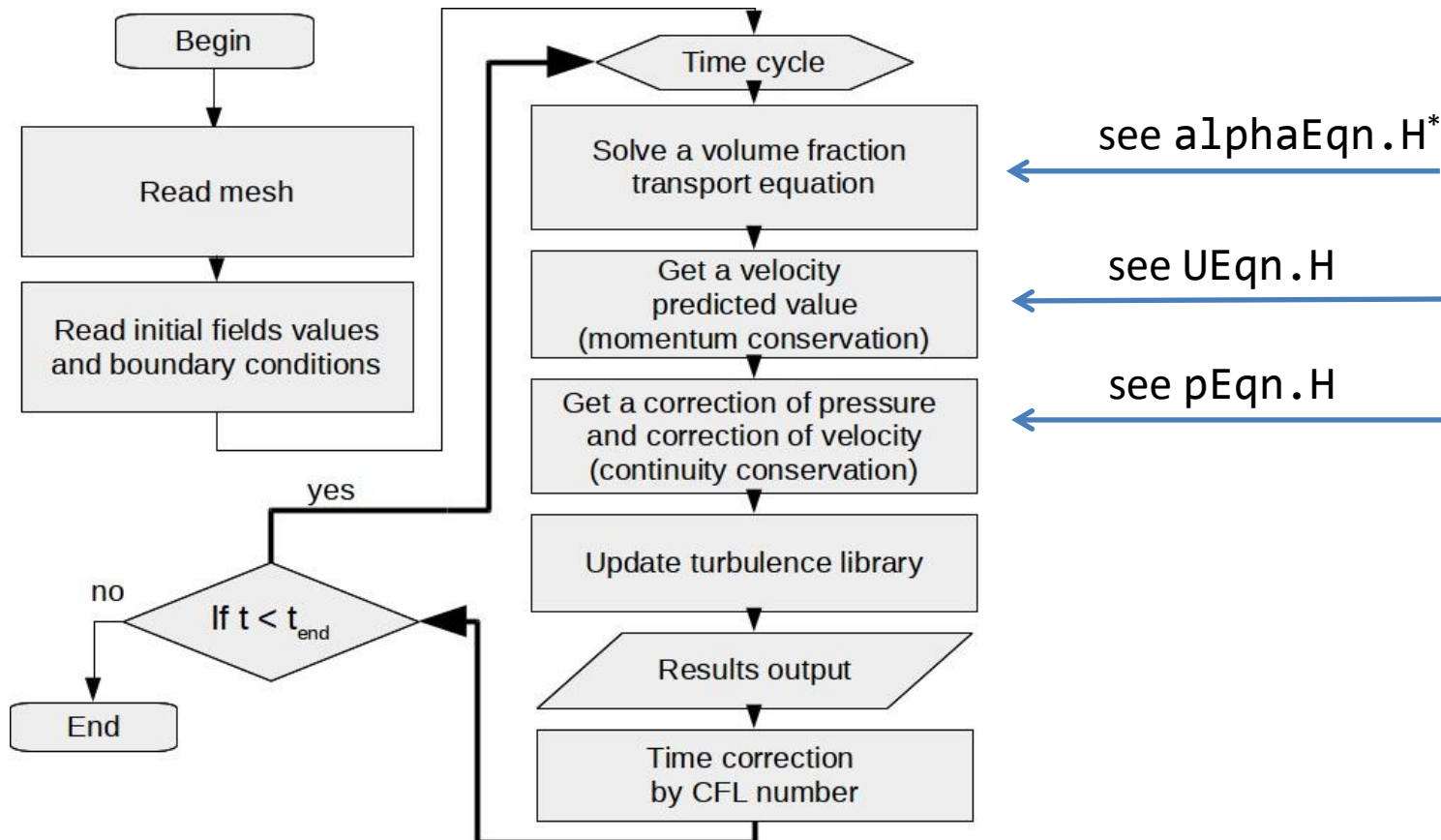


# Solution process



# interFoam solver

## Block scheme of solver



**Location:** OpenFoam-3.0.0/applications/solvers/multiphase/interFoam

# interFoam solver

## Modified Navier — Stokes equations

Surface tension forces are approximated as an additional term  $\vec{F}_\sigma$  in Navier — Stokes equation\*:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot (\rho \vec{U} \otimes \vec{U}) = -\nabla p + \nabla \cdot \hat{\tau} + \rho \vec{g} + \vec{F}_\sigma.$$

Use **modified pressure**:

$$p^* = p - \rho(\vec{g} \cdot \vec{r}).$$

Its gradient:

$$\nabla p^* = \nabla p - (\nabla \rho)(\vec{g} \cdot \vec{r}) - \rho \vec{g}.$$

Navier — Stokes equations in terms of modified pressure:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot (\rho \vec{U} \otimes \vec{U}) - \nabla \cdot \hat{\tau} = -\nabla p^* - (\nabla \rho)(\vec{g} \cdot \vec{r}) + \vec{F}_\sigma.$$

Approximation of surface tension forces — by **volume fraction gradient**:

$$\vec{F}_\sigma \approx \sigma \kappa \nabla \alpha_1.$$

# interFoam solver

## Pressure-velocity coupling

Use velocity as sum of prediction and correction parts:

$$\vec{U} = \vec{U}^* + \vec{U}'. \quad (*)$$

**Semi-discrete form** of momentum equation:

$$A\vec{U} = H - \nabla p^* - (\nabla \rho)(\vec{g} \cdot \vec{r}) + \vec{F}_\sigma.$$

Here  $A$  is the diagonal part of initial matrix system,

$H$  is the non-diagonal part of matrix + r.h.s. without pressure gradient and terms for gravity and surface tension.

We can write comparing with the velocity splitting (\*):

$$\vec{U}^* = A^{-1}H,$$

$$\vec{U}' = -A^{-1}\nabla p^* - A^{-1}(\nabla \rho)(\vec{g} \cdot \vec{r}) + A^{-1}\vec{F}_\sigma.$$

Continuity equation in the discrete form:

$$\int_V \nabla \cdot \vec{U} dV = \int_S \vec{U} \cdot \vec{n} dS \approx \sum_f \underbrace{\vec{U}_f \cdot \vec{S}_f}_{\phi_f} = 0 \Rightarrow \boxed{\sum_f \phi_f = 0.} \quad (**)$$

# interFoam solver

## Pressure-velocity coupling

Let's calculate fluxes through one face  $f$ :

$$\underbrace{\vec{U}_f \cdot \vec{S}_f}_{\phi} = \underbrace{(\vec{U}^*)_f \cdot \vec{S}_f}_{\phi_{H/A}} - \underbrace{(A^{-1})_f (\nabla p^*)_f \cdot \vec{S}_f}_{Dp} - \underbrace{-(A^{-1}(\nabla \rho)(\vec{g} \cdot \vec{r}))_f \cdot \vec{S}_f + (A^{-1} \vec{F}_\sigma)_f \cdot \vec{S}_f}_{\phi_g}. \quad (***)$$

Create a **pressure equation** which is derived from continuity equation (slide 12, (\*\*)):

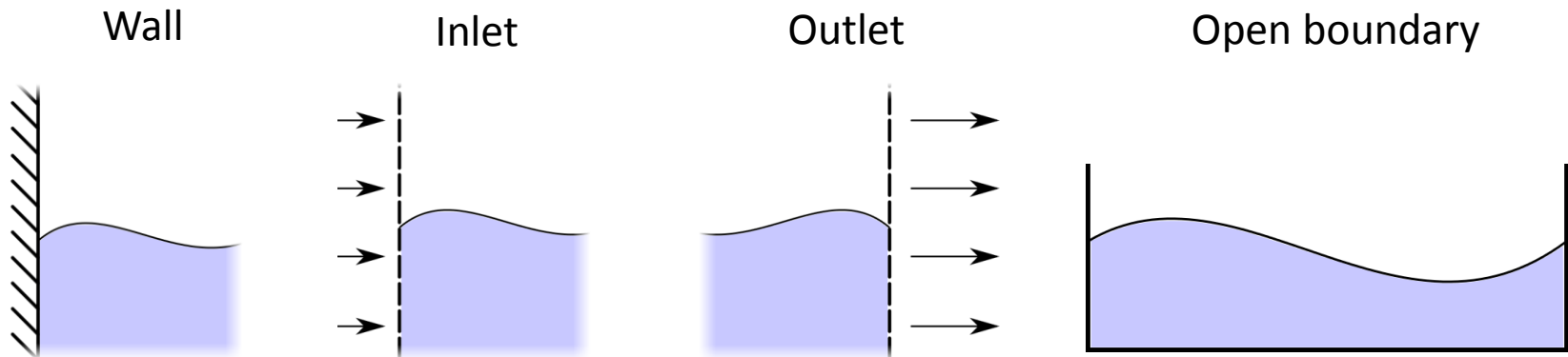
$$\sum_f (\phi_{H/A} + \phi_g)_f = \sum_f Dp (\nabla p^*)_f \cdot \vec{S}_f.$$

**Pressure gradient:**

$$(\nabla p^*)_f = \frac{\phi_{H/A} + \phi_g - \phi}{Dp \cdot \vec{S}_f}. \quad (****)$$

# interFoam solver

## Types of boundary conditions



### Note

For incompressible fluids it is important to calculate correctly the pressure gradient. Absolute value of pressure is calculated up to a constant. So, it is enough to know the pressure reference value only in one point.

**Location:** `OpenFoam-3.0.0/src/finiteVolume/fields/fvPatchFields`

# interFoam solver

## Walls & inlets

Boundary condition for **volume fraction**:

$\alpha = 0$  or  $\alpha = 1$  (fixedValue) — for inlets;

$\nabla\alpha_1 = 0$  (zeroGradient) — for walls.

Boundary condition for **velocity**:

$\vec{U} = \{U_x U_y U_z\}$  (fixedValue).

### Note

Capillary effects are neglected; to account for these effects — special boundary conditions for volume fraction must be imposed.

Boundary conditions for **pressure** are critically important here.

Let's consider the correct boundary condition and changes in OpenFOAM v.2.3.0+.

Influence of boundary conditions setup is demonstrated in the videos after the practical part.

# interFoam solver

## Walls & inlets

On boundaries for walls and inlets:

$$\vec{U} = \{U_x U_y U_z\}, \quad \vec{U}^* = \{U_x U_y U_z\} \Rightarrow \vec{U}' = \{0 0 0\}.$$

According to BC for volume fraction :

$$\vec{F}_\sigma \approx \sigma \kappa \nabla \alpha_1 = 0 \quad \text{on the boundary.}$$

So, according to expression for volumetric flux (slide 12, (\*\*\*)):

$$-Dp(\nabla p^*)_f \cdot \vec{S}_f - (A^{-1}(\nabla \rho)(\vec{g} \cdot \vec{r}))_f \cdot \vec{S}_f = 0.$$

and, therefore, we can derive boundary condition for pressure:

$$(\nabla p^*)_f = -(\nabla \rho)(\vec{g} \cdot \vec{r})_f.$$

---

## Implementation in OpenFOAM

In *OpenFOAM* v.2.2.x — **buoyantPressure**: this expression is in the source code of boundary condition.

In *OpenFOAM* v.2.3.0+ — **fixedFluxPressure**: boundary condition is satisfied automatically by pressure gradient (slide 12, (\*\*\*\*)) calculating in the solver's source code.



# interFoam solver

## Outlets & open boundaries

Boundary condition for **volume fraction**:

$$\nabla \alpha_1 = 0 \quad (\text{zeroGradient}).$$

Boundary condition for **velocity**:

$$\nabla \vec{U} = 0 \quad (\text{zeroGradient}).$$

Boundary condition for **pressure**:

- reference level of pressure — if there are no pressure boundary conditions in any another boundary:

$$p_p^* = p_0 - \frac{U^2}{2} \quad (\text{totalPressure}),$$

where  $p_0$  — total pressure,  $U$  — velocity magnitude.

In the source code we use:

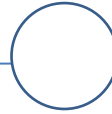
$$p = p_0 - 0.5 * (1 - \text{pos}(\phi)) * \text{magSqr}(U).$$

Here  $\text{pos}()$  is the boolean function which equals to 1 when the flux  $\phi > 0$ ;

- **zeroGradient** — if you have some another boundary with derived reference level of pressure.

# Part II

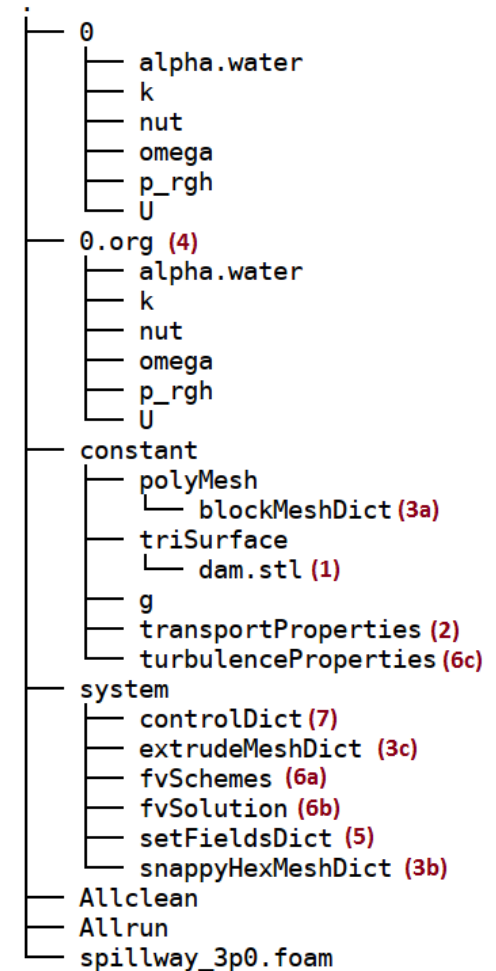
Practical part: hands-on training



**Spillway** tutorial

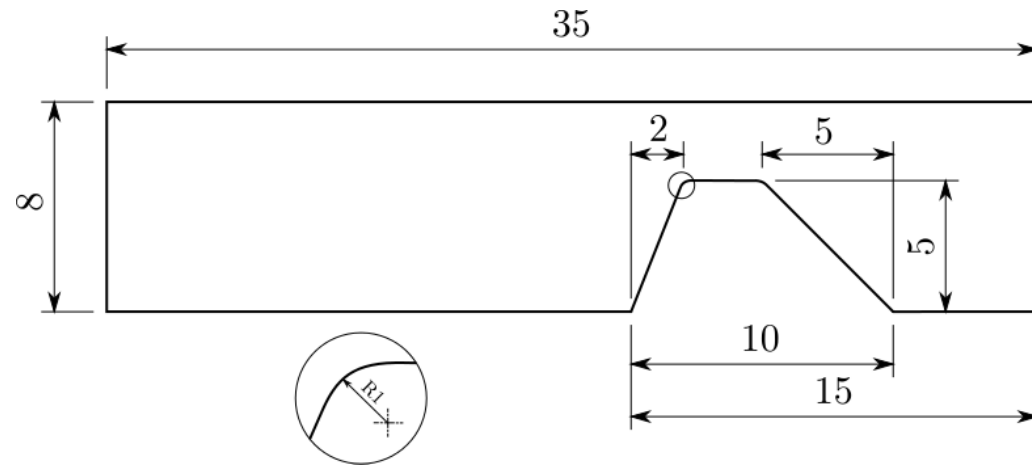
# Stages of solution

1. **Geometry**: make STL-surface to draw the dam (in SALOME).
2. **Liquid/gas properties**: write density, kinematic viscosity...
3. **Mesh generation**:
  - a) blockMesh: create a “0” level of mesh;
  - b) snappyHexMesh: refine mesh near the dam surface;
  - c) extrudeMesh: make a 2D-mesh for fast calculations.
4. **Boundary conditions**: describe it into files in “0.org” folder.
5. **Set fields**: set initial liquid phase volume fraction.
6. **Numerical settings**:
  - a) describe interpolation of terms;
  - b) describe solvers for SLAE;
  - c) setup turbulence models.
7. **Time settings**: set the end time, CFL-number...
8. **Running**: an interFoam command.
9. **Post-processing**: open file <filename>.foam in Paraview.
10. **Enjoy!**

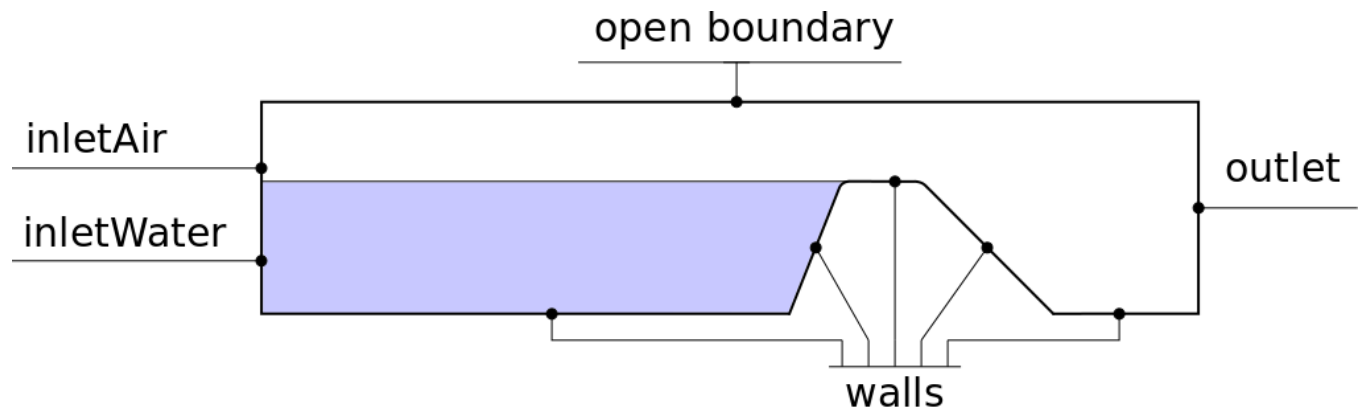


# Model setup

Geometry



Boundary conditions scheme



# Physical properties

See: folder constant/

File transportProperties: set values of

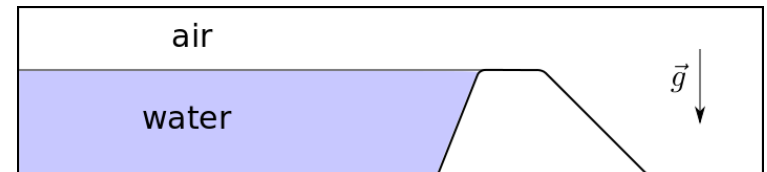
- density ( $\rho$ ),
- kinematic viscosity coefficient ( $\nu$ ),
- surface tension coefficient ( $\sigma$ ).

```
phases (water air);  
  
water  
{  
    transportModel    Newtonian;  
    nu                nu [ 0 2 -1 0 0 0 0 ] 1e-06;  
    rho               rho [ 1 -3 0 0 0 0 0 ] 1000;  
}  
  
air  
{  
    transportModel    Newtonian;  
    nu                nu [ 0 2 -1 0 0 0 0 ] 1.48e-05;  
    rho               rho [ 1 -3 0 0 0 0 0 ] 1;  
}  
  
sigma                sigma [ 1 0 -2 0 0 0 0 ] 0.07;
```

File g:

set values and direction of the gravity acceleration.

```
dimensions    [0 1 -2 0 0 0 0];  
value         (0 0 -9.81);
```

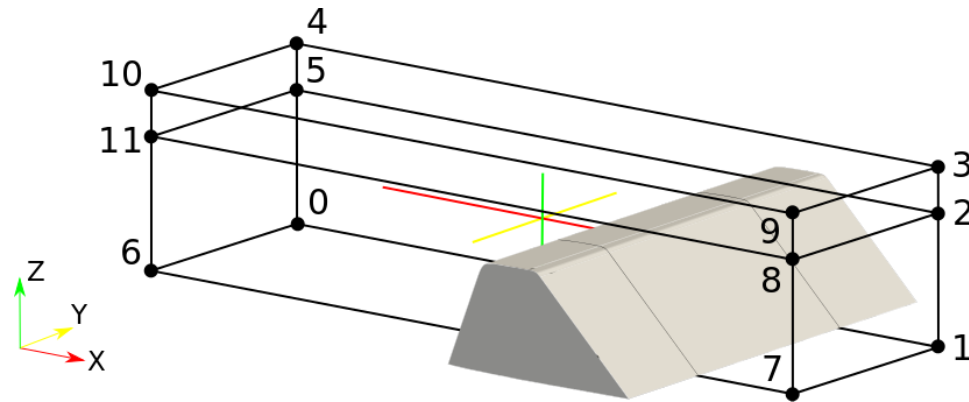


# blockMesh utility

Create mesh of “0” level in the flow region and mark the boundaries.

**See:** `constant/polyMesh/blockMeshDict`

**Command:** `blockMesh`



**Exercise:** inspect results in Paraview.

You should see the rectangular region with the coarse mesh.

# blockMesh utility

(1) Create vertices and two boxes:

```
convertToMeters 1;

vertices
(
    (-20 -0.45 0 )
    ( 15 -0.45 0 )
    ( 15 -0.45 6 )
    ( 15 -0.45 8 )
    (-20 -0.45 8 )
    (-20 -0.45 6 )
    (-20 -0.55 0 )
    ( 15 -0.55 0 )
    ( 15 -0.55 6 )
    ( 15 -0.55 8 )
    (-20 -0.55 8 )
    (-20 -0.55 6 )
);

blocks
(
    hex (0 1 2 5 6 7 8 11) (70 12 1) simpleGrading (1 1 1)
    hex (5 2 3 4 11 8 9 10) (70 4 1) simpleGrading (1 1 1)
);

edges
(
);
```

# blockMesh utility

## (2) Describe boundaries:

```
boundary
(
    inletAir
    {
        type patch;
        faces
        (
            (4 5 11 10)
        );
    }

    inletWater
    {
        type patch;
        faces
        (
            (5 0 6 11)
        );
    }
}
```

```
outlet
{
    type patch;
    faces
    (
        (1 2 8 7)
        (2 3 9 8)
    );
}

atmosphere
{
    type patch;
    faces
    (
        (3 4 10 9)
    );
}

bottomWall
{
    type wall;
    faces
    (
        (0 1 7 6)
    );
}
```

```
front
{
    type empty;
    faces
    (
        (1 0 5 2)
        (2 5 4 3)
    );
}

back
{
    type empty;
    faces
    (
        (6 7 8 11)
        (11 8 9 10)
    );
}

);
```



# snappyHexMesh utility

**See:** `constant/system/snappyHexMeshDict`

**Command:** `snappyHexMesh -overwrite`

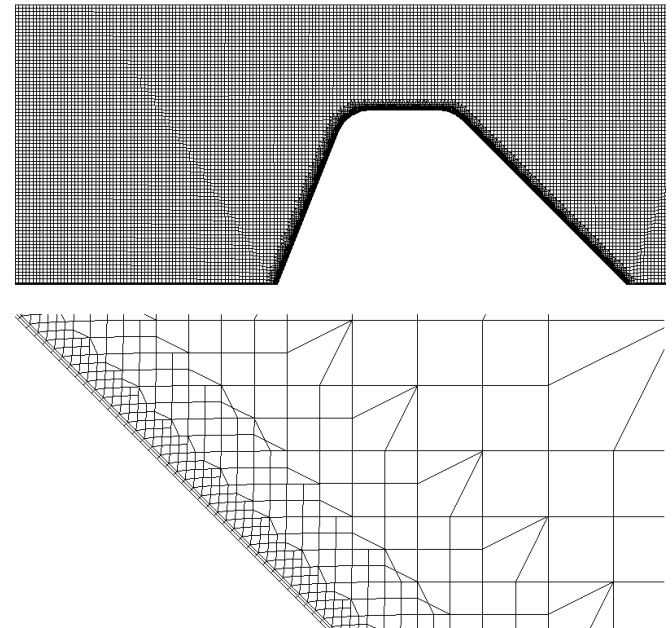
**(1)** Refine mesh near the surface of the dam.

Three stages of meshing:

- refinement  
(see `CastellatedMeshControl` section);
- smoothing  
(see `SnapControls` section);
- set of layers  
(see `addLayersControls` section).

Use STL-surface to do it  
(it is in `constant/triSurface`):

```
dam.stl
{
    type triSurfaceMesh;
    name dam;
}
```



# snappyHexMesh utility

(2) Add two refinement regions where the surface of water will flow.

Use two boxes and plane to do it:

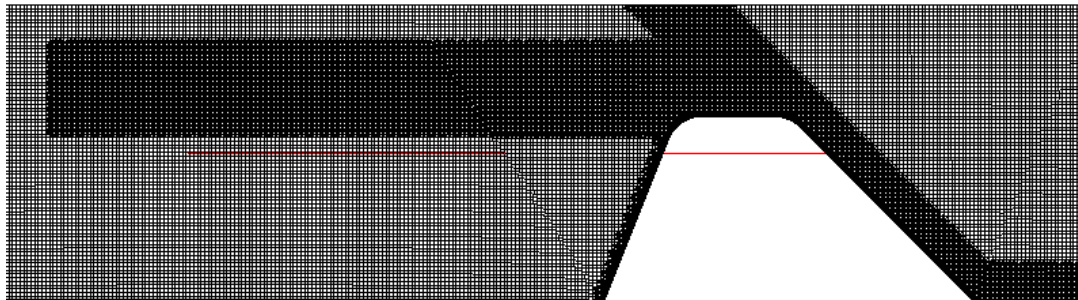
```
surface
{
    type searchableBox;
    min ( -15 -1 4.5 );
    max ( 4 0 7 );
}
```

```
aroundDam
{
    type searchablePlane;
    planeType pointAndNormal;

    pointAndNormalDict
    {
        basePoint ( 7.5 -0.5 2.5 );
        normalVector ( 1 0 1 );
    };
}
```

```
outlet
{
    type searchableBox;
    min ( 10 -1 0 );
    max ( 15 0 1 );
}
```

**Exercise:** try to run snappyHexMesh with different studies of remeshing. Watch differences.



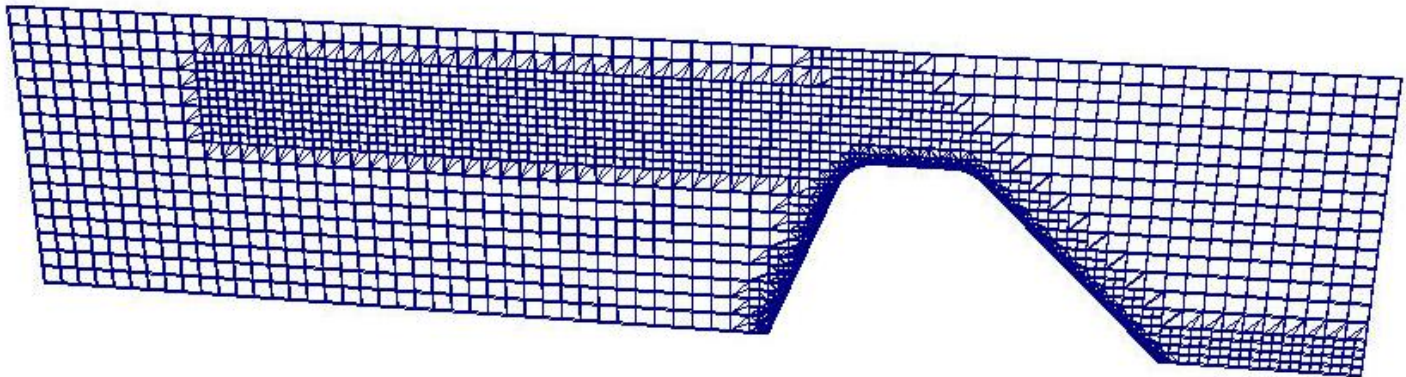
# extrudeMesh utility

**See:** `constant/system/extrudeMeshDict`

**Command:** `extrudeMesh`

`blockMesh` and `snappyHexMesh` made 3D-mesh.

`extrudeMesh` creates a 2D-mesh from the surface (in our case — from patch).



**Exercise:** run `checkMesh` utility before running `extrudeMesh` and after it.  
Compare numbers of cells.

# Boundary conditions

See: folder `0.org/`. Change needed files and copy it to folder `0/`.

	$\alpha_1$	$k$	$\omega$	$p^*$	$\vec{U}$
<b>inletAir</b>	fixedValue 0	fixedValue 2.16e-4	fixedValue 0.1470	fixedFluxPressure	fixedValue (0 0 0)
<b>inletWater</b>	fixedValue 1	fixedValue 2.16e-4	fixedValue 0.1470	fixedFluxPressure	fixedValue (0.6 0 0)
<b>outlet</b>	zeroGradient	zeroGradient	zeroGradient	fixedFluxPressure	zeroGradient
<b>walls</b>	zeroGradient	kqRWallFunction	omegaWallFunction	fixedFluxPressure	fixedValue (0 0 0)
<b>atmosphere</b>	inletOutlet	inletOutlet 2.16e-4	inletOutlet 0.1470	totalPressure	pressureInletOutletVelocity
<b>front, back, defaultFaces</b>	empty	empty	empty	empty	empty

## Note

Files in `0/` folder are modified after using `setFields` utility.  
We use the copy `0.org/` for comfortable changing of boundary conditions.

# setFields utility

Set an initial distribution of fields (alpha.water) in regions.  
Files in 0/ folder were modified.

```
defaultFieldValues
(
    volScalarFieldValue alpha.water 0
);

regions
(
    boxToCell
    {
        box (-20 -1 0) (3 1 5);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);
```



# Numerical schemes and time settings.

## Running

See `system/controlDict` to create time settings:

- time interval,
- CFL number,
- write interval,
- time precision.

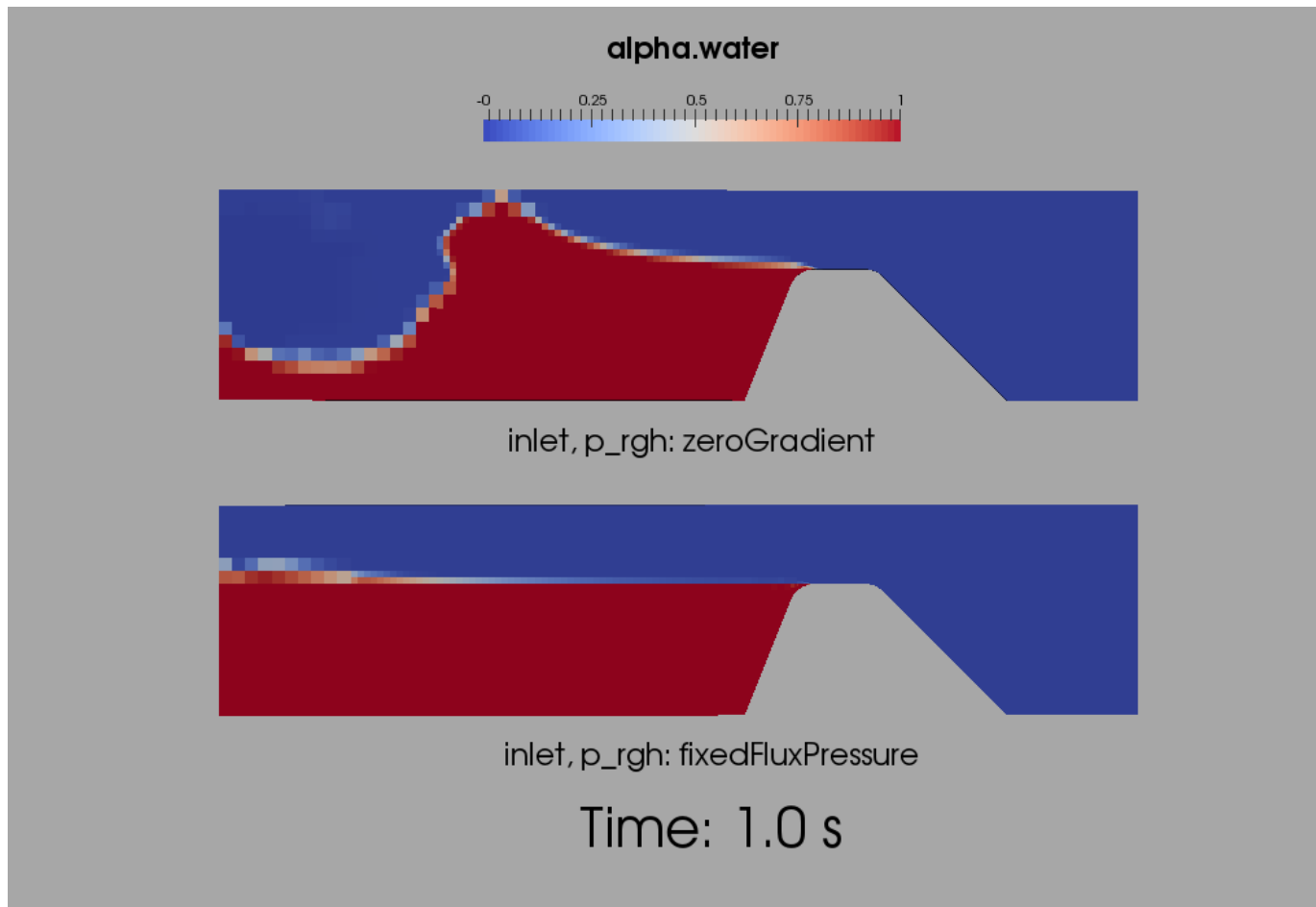
Settings for numerical schemes (use default settings):  
see `system/fvSchemes` and `system/fvSolution`.

Start application by `interFoam` command.

Sequence of all commands is placed into script file `./Allrun`.

Clean results: `./Allclean`.

# Results



Watch the video in the folder of training track!

# Summary

- We looked how `interFoam` works inside (in the source code).
- We learned how to set correct boundary conditions for free-surface flows.
- We studied how to solve cases for free-surface flows on example — Spillway tutorial.

Let's talk about training track.  
Some questions?