Tutorial 5: Breaking of a Dam

Asmaa HADANE

ENS Paris-Saclay

March 4, 2025

Problem

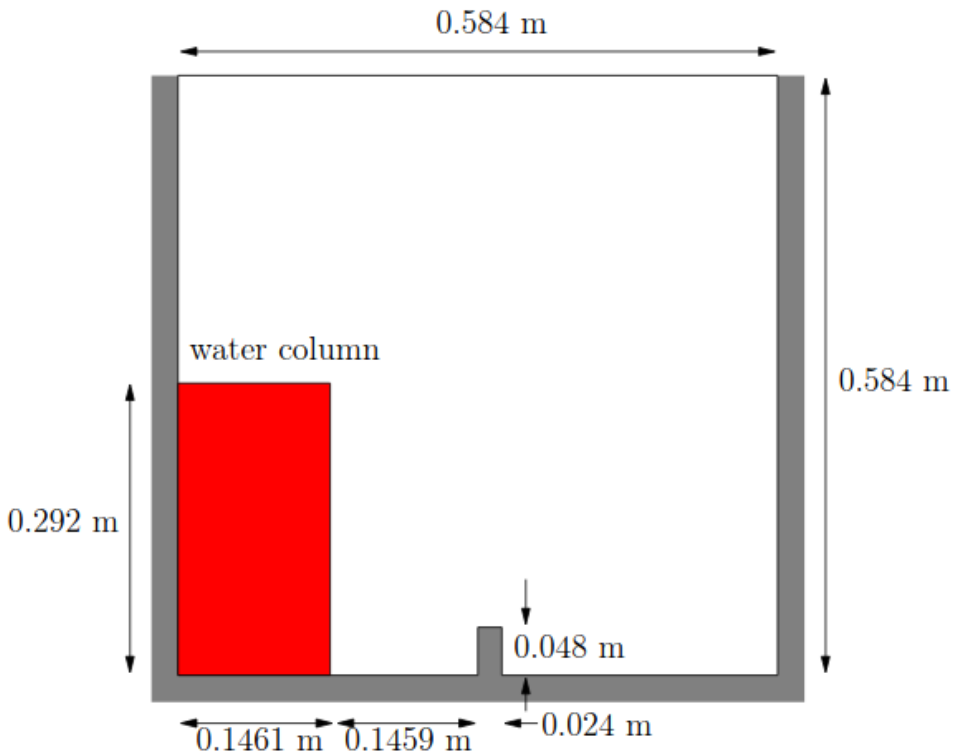
In this tutorial we will solve a problem of simplified dam break in:

- 2D
- transient flow
- two fluids
- solver: interFoam

The two-phase algorithm in interFoam is based on the volume of fluid (VOF) method in which a specie transport equation is used to determine the relative volume fraction of the two phases, or phase fraction α, in each computational cell.

Geometry

The test setup consists of a column of water at rest located behind a membrane on the left side of a tank. At time t= 0 s, the membrane is removed and the column of water collapses. During the collapse, the water impacts an obstacle at the bottom of the tank and creates a complicated flow structure, including several captured pockets of air.

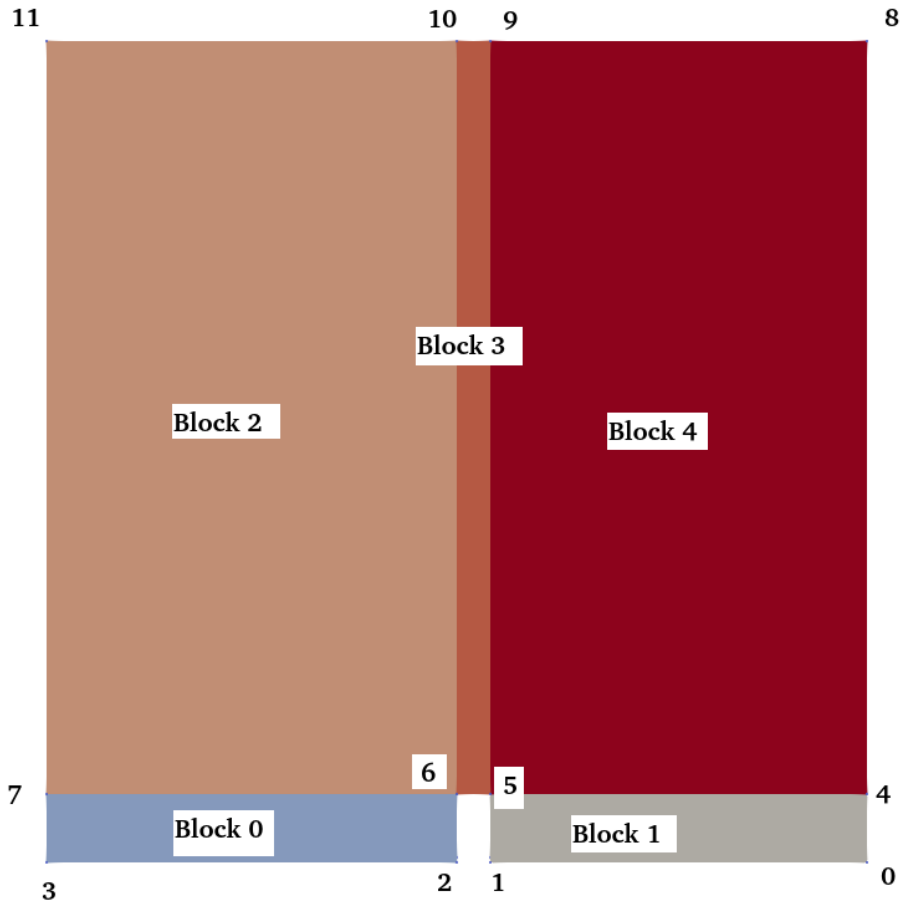


Test case

- go to run directory: $ run
- copy the damBreak case to the run directory:
  cp -r $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak/damBreak .

Mesh generation

Go into the damBreakcase directory and generate the mesh running blockMesh.
The damBreak mesh consist of 5 blocks.



Mesh generation

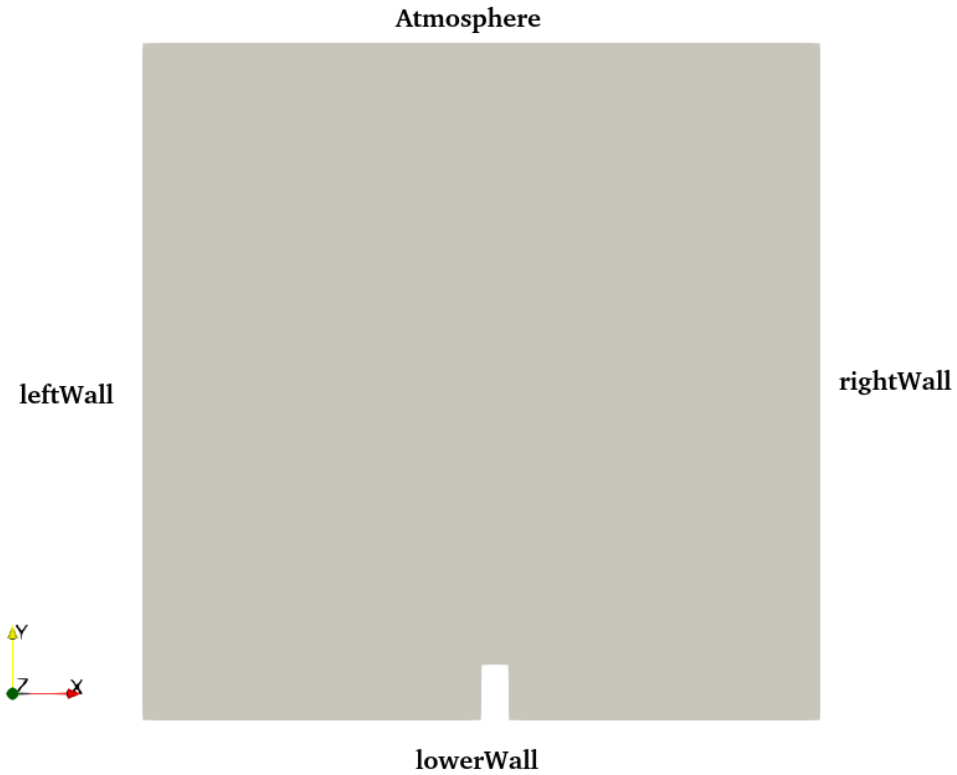
```
convertToMeters 0.146;

vertices
(
    (0 0 0)           //0
    (2 0 0)           //1
    (2.16438 0 0)     //2
    (4 0 0)           //3
    (0 0.32876 0)     //4
    (2 0.32876 0)     //5
    (2.16438 0.32876 0) //6
    (4 0.32876 0)     // 7
    (0 4 0)           // 8
    (2 4 0)            // 9
    (2.16438 4 0)      //10
    (4 4 0)            //11
    (0 0 0.1)
    (2 0 0.1)
    (2.16438 0 0.1)
    (4 0 0.1)
    (0 0.32876 0.1)
    (2 0.32876 0.1)
    (2.16438 0.32876 0.1)
    (4 0.32876 0.1)
    (0 4 0.1)
    (2 4 0.1)
    (2.16438 4 0.1)
    (4 4 0.1)
);

blocks
(
    hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1) // block 0
    hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1) // block 1
    hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1) // block 2
    hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1) // block 3
    hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1) // block 4
);
```

Boundaries

The user can examine the boundary geometry generated by blockMesh by viewing the boundary file in the constant/polyMesh directory. The file contains a list of 5 boundary patches:leftWall,rightWall,lowerWall,atmosphere and defaultFaces.



Boundaries

```
boundary
(
    leftWall
    {
        type wall;
        faces
        (
            (0 12 16 4)
            (4 16 20 8)
        );
    }
    rightWall
    {
        type wall;
        faces
        (
            (7 19 15 3)
            (11 23 19 7)
        );
    }
    lowerWall
    {
        type wall;
        faces
        (
            (0 1 13 12)
            (1 5 17 13)
            (5 6 18 17)
            (2 14 18 6)
            (2 3 15 14)
        );
    }
    atmosphere
    {
        type patch;
        faces
        (
            (8 20 21 9)
            (9 21 22 10)
            (10 22 23 11)
        );
    }
);
```

Setting initial fields

Unlike the previous cases, we have to specify a non-uniform initial condition for the phase fraction α_{water}.
This will be done by running the setFields utility. It requires a setFieldsDict dictionary,located in the system directory, whose entries for this case are shown below.

```
17
18  defaultFieldValues
19  (
20      volScalarFieldValue alpha.water 0
21  );
22
23  regions
24  (
25      boxToCell
26      {
27          box (0 0 -1) (0.1461 0.292 1);
28          fieldValues
29          (
30              volScalarFieldValue alpha.water 1
31          );
32      }
33  );
34
```

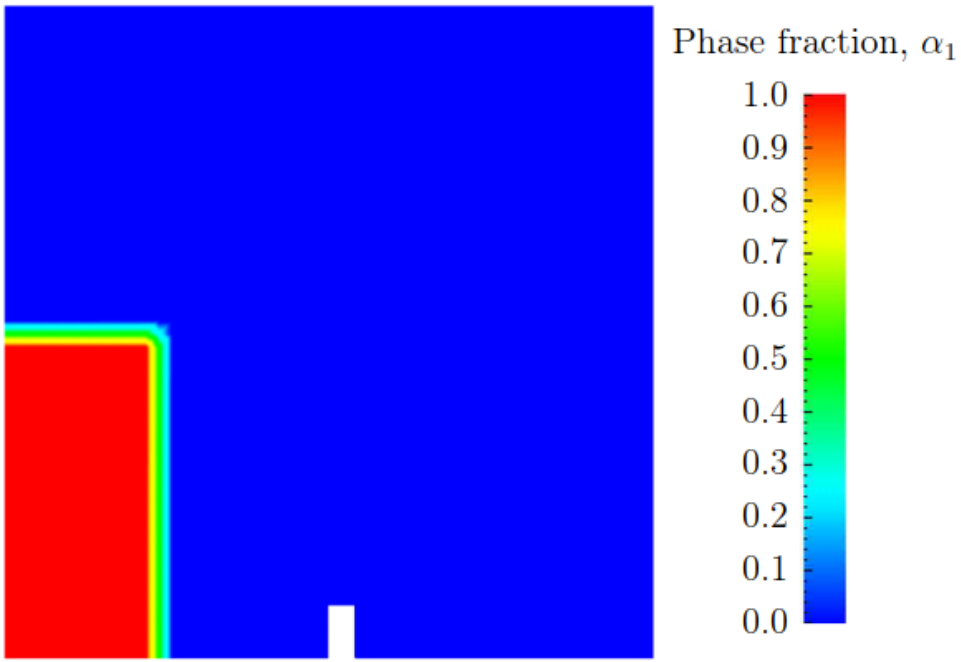
boxToCell creates a bounding box within a vector minimum and maximum to define the set of cells of the water region. The phase fraction α_{water} is defined as 1 in this region.

Setting initial fields

The user should therefore execute setFields like any other utility by:

$*setFields*

Using paraFoam, check that the initial alpha.water field corresponds to the desired distribution as in Figure below.



Fluid properties

`water` properties			
Kinematic viscosity	$\mathrm{m^2\,s^{-1}}$	`nu`	1.0×10^{-6}
Density	$\mathrm{kg\,m^{-3}}$	`rho`	1.0×10^{3}

`air` properties			
Kinematic viscosity	$\mathrm{m^2\,s^{-1}}$	`nu`	1.48×10^{-5}
Density	$\mathrm{kg\,m^{-3}}$	`rho`	1.0

Properties of both phases			
Surface tension	$\mathrm{N\,m^{-1}}$	`sigma`	0.07

Gravitational effect

Gravitational acceleration is uniform across the domain and is specified in a file named g in the constant directory.

```
17
18  dimensions      [0 1 -2 0 0 0 0];
19  value           (0 -9.81 0);
20
21
22  // ************************************************************
```

Turbulence modeling

The choice of turbulence modeling method is selectable at run-time through the simulationType keyword in momentumTransport dictionary. In this example,we wish to run without turbulence modelling so we set laminar.

```
17
18   simulationType  laminar;
19
20
21   // ************************************************************************
```

Time step control

Time step control is an important issue in transient simulation and the surface-tracking algorithm in interface capturing solvers.

```
application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         1;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     binary;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           1;
maxAlphaCo      1;

maxDeltaT       1;
```

Running the code

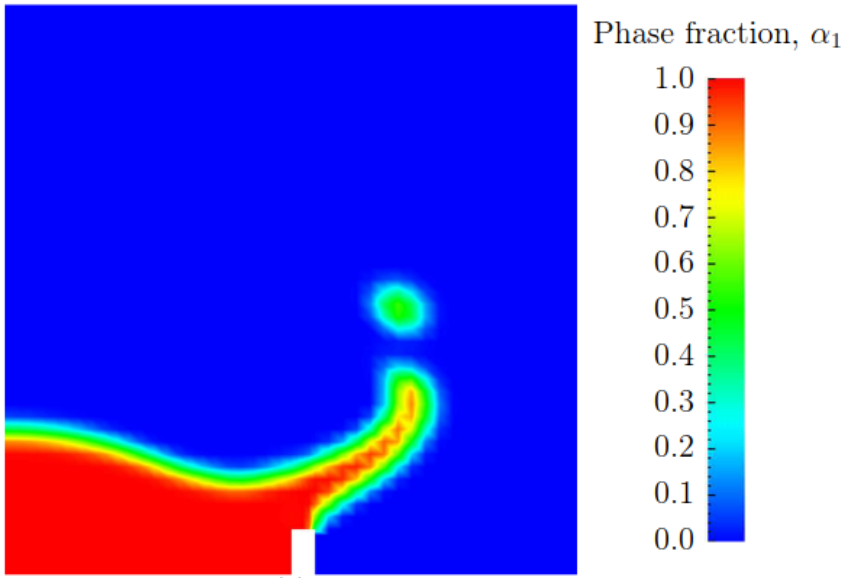
For running the code, try the following,that uses tee, a command that enables output to be written to both standard output and files:

$ *cd* $*FOAM_RUN*/*damBreak*

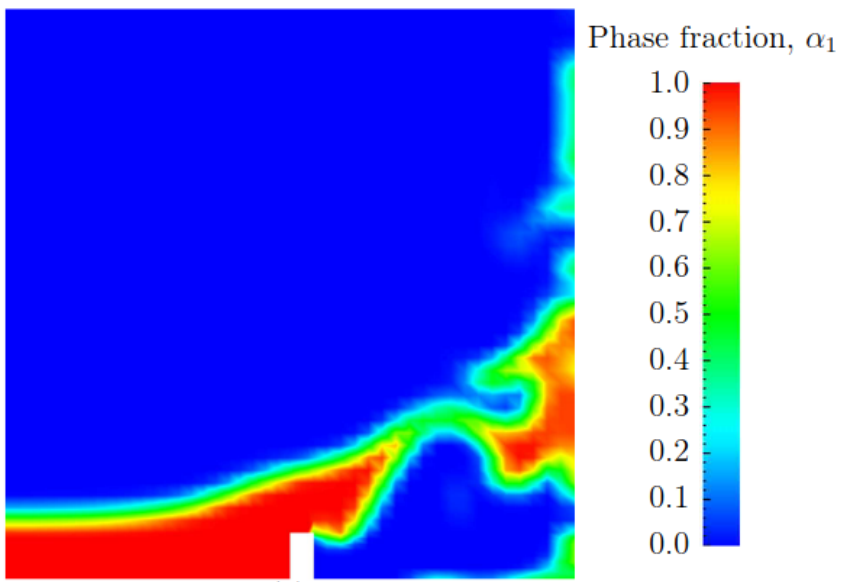
$ *interFoam* | *teelog*

The code will now be run interactively, with a copy of output stored in the log file.

Post processing



(a) At $t = 0.25$ s.



(b) At $t = 0.50$ s.