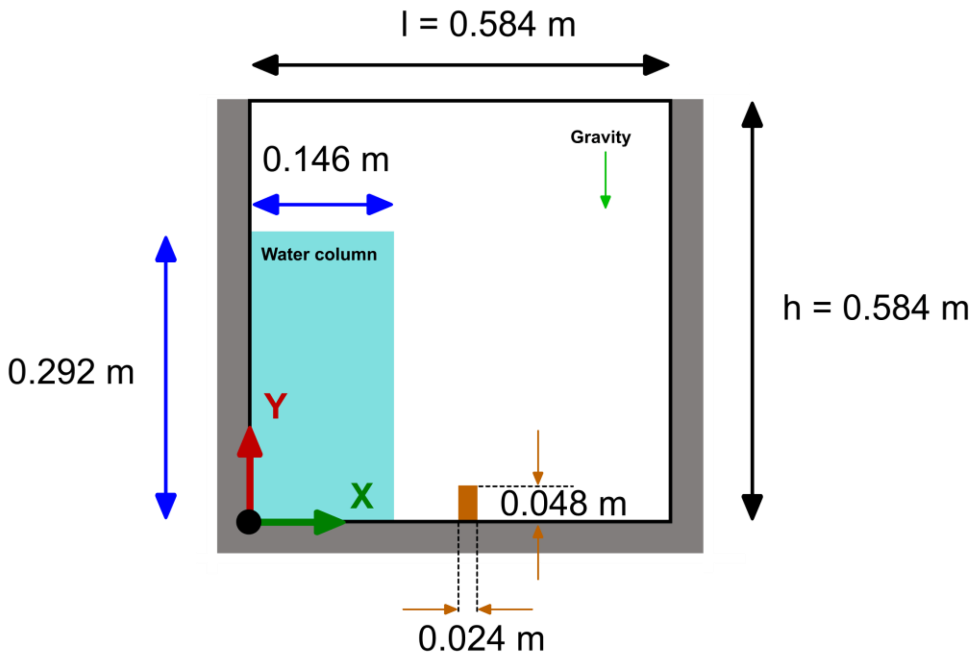
Dam break free surface flow

Dam break free surface flow

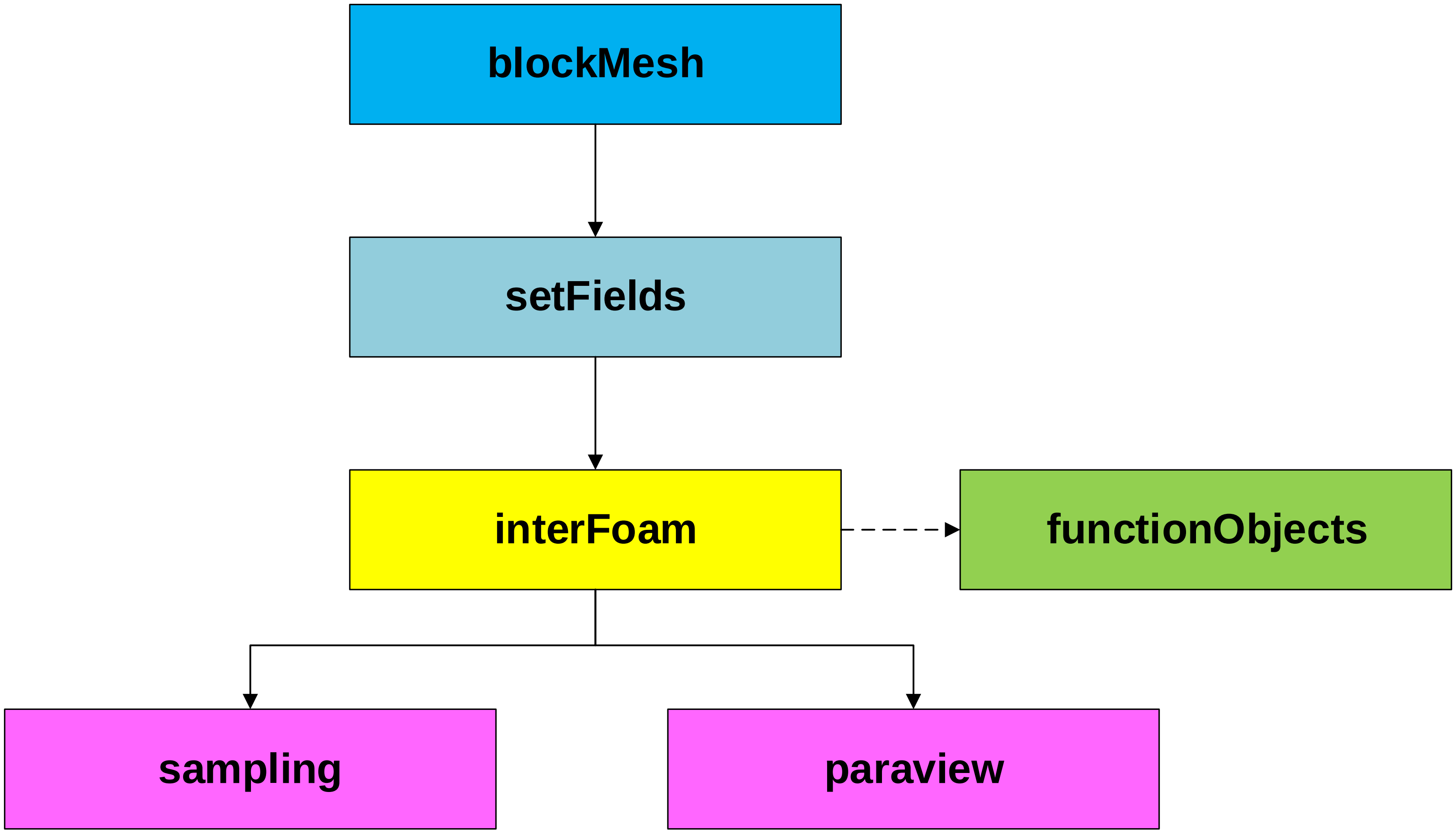


Physical and numerical side of the problem:

- In this case we are going to use the VOF method. This method solves the incompressible Navier-Stokes equations plus an additional equation to track the volume fraction (free surface location).
- We are going to work in a 2D domain but the problem can be extended to 3D easily.
- As this is a multiphase case, we need to define the physical properties for each phase involved (viscosity, density and surface tension).
- Additionally, we need to define the gravity vector and initialize the two flows.
- This is an unsteady case.

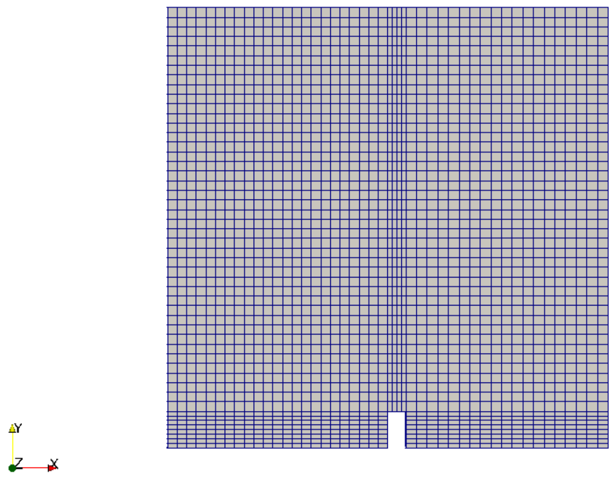
Dam break free surface flow

Workflow of the case

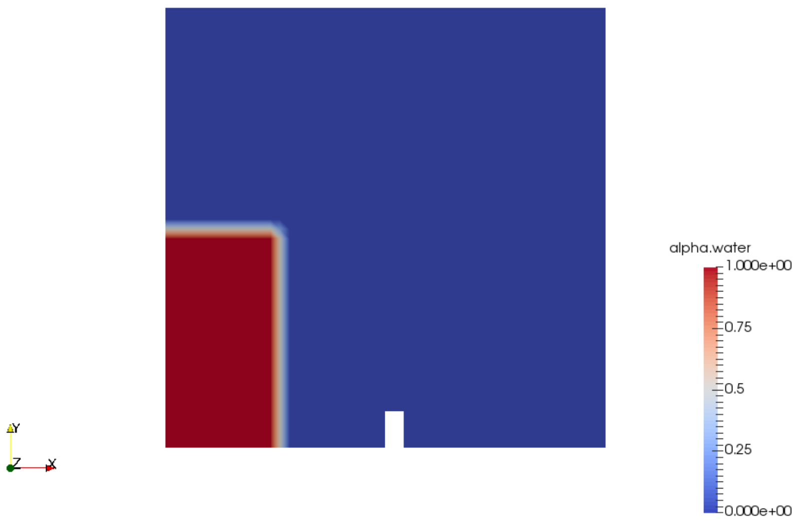


Dam break free surface flow

At the end of the day you should get something like this



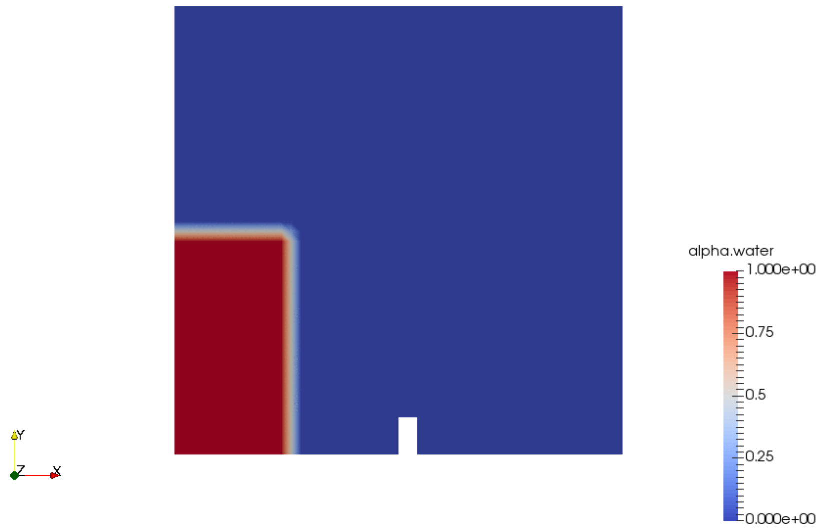
Mesh



Initial conditions

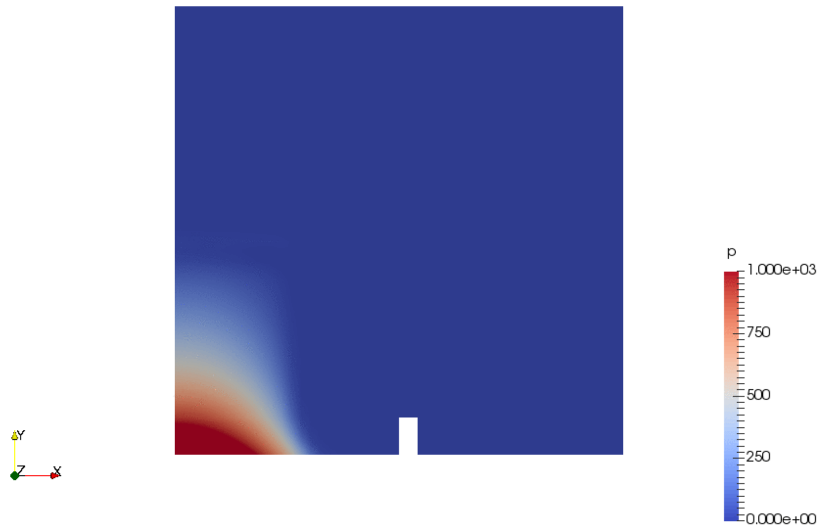
Dam break free surface flow

At the end of the day you should get something like this



VOF Fraction

www.wolfdynamics.com/wiki/dambreak/ani1.gif



Hydrostatic pressure

www.wolfdynamics.com/wiki/dambreak/ani2.gif

Dam break free surface flow

What are we going to do?

- We will use this case to introduce the multiphase solver `interFoam`.
- `interFoam` is a solver for 2 incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach
- We will define the physical properties of two phases and we are going to initialize these phases.
- We will define the gravity vector in the dictionary *g*.
- After finding the solution, we will visualize the results. This is an unsteady case so now we are going to see things moving.
- We are going to briefly address how to post-process multiphase flows.

Dam break free surface flow

Let’s explore the case directory

Dam break free surface flow

The *blockMeshDict* dictionary file

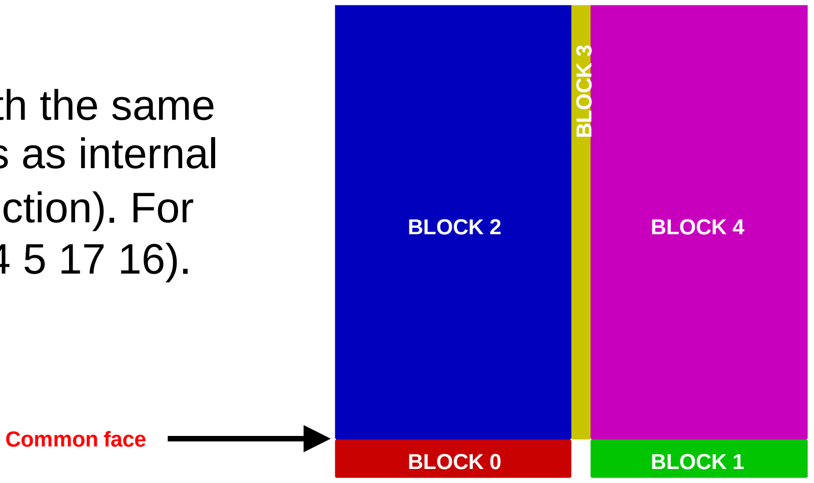
```
17      convertToMeters 0.146;
18
19      vertices
20      (
21          (0 0 0)                     //Vertex0
22          (2 0 0)
23          (2.16438 0 0)
24          (4 0 0)
25          (0 0.32876 0)
26          (2 0.32876 0)
27          (2.16438 0.32876 0)
28          (4 0.32876 0)
29          (0 4 0)
30          (2 4 0)
31          (2.16438 4 0)
32          (4 4 0)
33          (0 0 0.1)
34          (2 0 0.1)
35          (2.16438 0 0.1)
36          (4 0 0.1)
37          (0 0.32876 0.1)
38          (2 0.32876 0.1)
39          (2.16438 0.32876 0.1)
40          (4 0.32876 0.1)
41          (0 4 0.1)
42          (2 4 0.1)
43          (2.16438 4 0.1)
44          (4 4 0.1)                   //Vertex 23
45      );
```

- This dictionary is located in the **`system`** directory.
- We are using scaling (line 17).
- In lines 19-45, we define the vertices coordinates.

Dam break free surface flow

The *blockMeshDict* dictionary file

- In this case we are defining five blocks.
- In the common faces, the blocks share vertices with the same index number, `blockMesh` recognizes these faces as internal (we do not need to define them in the boundary section). For example, block 0 and block 2 share the vertices (4 5 17 16).
- We are using uniform grading in all blocks.
- All edges are straight lines by default.



```
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
);

edges
(
);
```

Dam break free surface flow

The *blockMeshDict* dictionary file

```
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)
        );
    }
```

- The boundary patches **leftWall, rightWall** and **lowerWall** are of **base type** wall.
- Notice that each boundary patch groups many faces.
- Remember, we assign the **primitive type** boundary conditions (numerical values), in the field files found in the directory ***0***

Dam break free surface flow

The *blockMeshDict* dictionary file

```
        atmosphere
        {
            type patch;
            faces
            (
                (8 20 21 9)
                (9 21 22 10)
                (10 22 23 11)
            );
        }
);

mergePatchPairs
(
);
```

- The boundary patch atmosphere is of **base type** patch.
- Notice that we do not define the front and back patches, these patches are automatically group in the boundary patch **defaultFaces** of **base type** empty.
- Remember, we assign the **primitive type** boundary conditions (numerical values), in the field files found in the directory ***0***
- We do not need to merge faces.

Dam break free surface flow

The *boundary* dictionary file

- This dictionary is located in the **constant/polyMesh** directory.
- This file is automatically created when generating or converting the mesh.
- In this case, we do not need to modify this file. All the **base type** boundary conditions and **name** of the patches were assigned in the *blockMeshDict* file.
- The **defaultFaces** boundary patch contains all patches that we did not define in the boundary section.
- If you change the **name** or the **base type** of a boundary patch, you will need to modify the field files in the directory **0**.

```
defaultFaces
{
    type            empty;
    inGroups        1(empty);
    nFaces          4563;
    startFace       4640;
}
```

Dam break free surface flow

🗀 The **`constant`** directory

- In this directory, we will find the following compulsory dictionary files:

  - *`g`*
  - *`transportProperties`*
  - *`turbulenceProperties`*

- *`g`* contains the definition of the gravity vector.
- *`transportProperties`* contains the definition of the physical properties of each phase.
- *`turbulenceProperties`* contains the definition of the turbulence model to use.

Dam break free surface flow

The g dictionary file

```
8       FoamFile
9       {
10          version     2.0;
11          format      ascii;
12          class       uniformDimensionedVectorField;
13          location    "constant";
14          object      g;
15      }
17
18      dimensions        [0 1 -2 0 0 0 0];
19      value             (0 -9.81 0);
```

- This dictionary file is located in the directory **`constant`**.
- For multiphase flows, this dictionary is compulsory.
- In this dictionary we define the gravity vector (line 19).
- Pay attention to the **class** type (line 12).

Dam break free surface flow

The *transportProperties* dictionary file

Primary phase

```
18    phases (water air);
19
20    water
21    {
22        transportModel  Newtonian;
23        nu              [0 2 -1 0 0 0 0] 1e-06;
24        rho             [1 -3 0 0 0 0 0] 1000;
25    }
26
27    air
28    {
29        transportModel  Newtonian;
30        nu              [0 2 -1 0 0 0 0] 1.48e-05;
31        rho             [1 -3 0 0 0 0 0] 1;
32    }
33
34    sigma           [1 0 -2 0 0 0 0] 0.07;
```

- This dictionary file is located in the directory **`constant`**.
- We first define the name of the phases (line 18). In this case we are defining the names **water** and **air**. The first entry in this list is the primary phase (**water**).
- The name of the phases is given by the user.
- In this file we set the kinematic viscosity (**nu**), density (**rho**) and transport model (**transportModel**) of the phases.
- We also define the surface tension (**sigma**).

Dam break free surface flow

The *turbulenceProperties* dictionary file

- In this dictionary file we select what model we would like to use (laminar or turbulent).
- This dictionary is compulsory.
- As we do not want to model turbulence, the dictionary is defined as follows,

```
simulationType    laminar;
```

Dam break free surface flow

🗀 The **0** directory

- In this directory, we will find the dictionary files that contain the boundary and initial conditions for all the primitive variables.
- As we are solving the incompressible laminar Navier-Stokes equations using the VOF method, we will find the following field files:

• *`alpha.water`*	(volume fraction of water phase)
• *`p_rgh`*	(pressure field minus hydrostatic component)
• *`U`*	(velocity field)

Dam break free surface flow

The file *0/alpha.water*

```
17    dimensions       [0 0 0 0 0 0 0];
18
19    internalField    uniform 0;
20
21    boundaryField
22    {
23        leftWall
24        {
25            type             zeroGradient;
26        }
27
28        rightWall
29        {
30            type             zeroGradient;
31        }
32
33        lowerWall
34        {
35            type             zeroGradient;
36        }
37
38        atmosphere
39        {
40            type             inletOutlet;
41            inletValue       uniform 0;
42            value            uniform 0;
43        }
44
45        defaultFaces
46        {
47            type             empty;
48        }
49    }
```

- This file contains the boundary and initial conditions for the non-dimensional scalar field **alpha.water**
- This file is named `alpha.water`, because the primary phase is water (we defined the primary phase in the `transportProperties` dictionary).
- Initially, this field is initialize as 0 in the whole domain (line 19). This means that there is no water in the domain at time 0. Later, we will initialize the water column and this file will be overwritten with a non-uniform field for the **internalField**.
- For the **leftWall**, **rightWall,** and **lowerWall** patches we are using a **zeroGradient** boundary condition (we are just extrapolating the internal values to the boundary face).
- For the **atmosphere** patch we are using an **inletOutlet** boundary condition**.** This boundary condition avoids backflow into the domain. If the flow is going out it will use **zeroGradient** and if the flow is coming back it will assign the value set in the keyword **inletValue** (line 41).
- The **defaultFaces** patch is of **primitive type empty**.

Dam break free surface flow

The file *0/p_rgh*

```
17    dimensions      [1 -1 -2 0 0 0 0];
18
19    internalField   uniform 0;
20
21    boundaryField
22    {
23        leftWall
24        {
25            type            fixedFluxPressure;
26            value           uniform 0;
27        }
29        rightWall
30        {
31            type            fixedFluxPressure;
32            value           uniform 0;
33        }
35        lowerWall
36        {
37            type            fixedFluxPressure;
38            value           uniform 0;
39        }
41        atmosphere
42        {
43            type            totalPressure;
44            p0              uniform 0;
45            U               U;
46            phi             phi;
47            rho             rho;
48            psi             none;
49            gamma           1;
50            value           uniform 0;
51        }
53        defaultFaces
54        {
55            type            empty;
56        }
57    }
```

- This file contains the boundary and initial conditions for the dimensional field **p_rgh**. The dimensions of this field are given in Pascal (line 17)
- This scalar field contains the value of the static pressure field minus the hydrostatic component.
- This field is initialize as 0 in the whole domain (line 19).
- For the **leftWall**, **rightWall,** and **lowerWall** patches we are using a **fixedFluxPressure** boundary condition (refer to the source code or doxygen documentation to know more about this boundary condition).
- For the **atmosphere** patch we are using the **totalPressure** boundary condition (refer to the source code or doxygen documentation to know more about this boundary condition).
- The **defaultFaces** patch is of **primitive type empty**.

Dam break free surface flow

The file *0/U*

```
17    dimensions      [0 1 -1 0 0 0 0];
18
19    internalField   uniform (0 0 0);
20
21    boundaryField
22    {
23        leftWall
24        {
25            type            fixedValue;
26            value           uniform (0 0 0);
27        }
28        rightWall
29        {
30            type            fixedValue;
31            value           uniform (0 0 0);
32        }
33        lowerWall
34        {
35            type            fixedValue;
36            value           uniform (0 0 0);
37        }
38        atmosphere
39        {
40            type            pressureInletOutletVelocity;
41            value           uniform (0 0 0);
42        }
43        defaultFaces
44        {
45            type            empty;
46        }
47    }
```

- This file contains the boundary and initial conditions for the dimensional vector field **U**.
- We are using uniform initial conditions and the numerical value is **(0 0 0)** (keyword **internalField** in line 19).
- The **leftWall**, **rightWall,** and **lowerWall** patches are no-slip walls, therefore we impose a **fixedValue** boundary condition with a value of **(0 0 0)** at the wall.
- For the **outlet** patch we are using a **zeroGradient** boundary condition (we are just extrapolating the internal values to the boundary face).
- For the **atmosphere** patch we are using the **pressureInlterOutletVelocity** boundary condition (refer to the source code or doxygen documentation to know more about this boundary condition).
- The **defaultFaces** patch is of **primitive type empty**.

Dam break free surface flow

🗀 The **`system`** directory

- The **`system`** directory consists of the following compulsory dictionary files:
    - *`controlDict`*
    - *`fvSchemes`*
    - *`fvSolution`*

- *`controlDict`* contains general instructions on how to run the case.
- *`fvSchemes`* contains instructions for the discretization schemes that will be used for the different terms in the equations.
- *`fvSolution`* contains instructions on how to solve each discretized linear equation system.

Dam break free surface flow

The *controlDict* dictionary

```
18      application     interFoam;
19
20      startFrom       startTime;
21
22      startTime       0;
23
24      stopAt          endTime;
25
26      endTime         1;
27
28      deltaT          0.001;
29
30      writeControl    adjustableRunTime;
31
32      writeInterval   0.05;
33
34      purgeWrite      0;
35
36      writeFormat     ascii;
37
38      writePrecision  8;
39
40      writeCompression uncompressed;
41
42      timeFormat      general;
43
44      timePrecision   8;
45
46      runTimeModifiable yes;
47
48      adjustTimeStep  yes;
49
50      maxCo           1;
51      maxAlphaCo      1;
52      maxDeltaT       1;
```

- This case starts from time 0 (**startTime**).
- It will run up to 1 second (**endTime**).
- The initial time step of the simulation is 0.001 seconds (**deltaT**).
- It will write the solution every 0.05 seconds (**writeInterval**) of simulation time (**runTime**). It will automatically adjust the time step (**adjustableRunTime**), in order to save the solution at the precise write interval.
- It will keep all the solution directories (**purgeWrite**).
- It will save the solution in ascii format (**writeFormat**).
- The write precision is 8 digits (**writePrecision**). It will only save eight digits in the output files.
- And as the option **runTimeModifiable** is on, we can modify all these entries while we are running the simulation.
- In line 48 we turn on the option **adjustTimeStep**. This option will automatically adjust the time step to achieve the maximum desired courant number (lines 50-51). We also set a maximum time step in line 52.
- Remember, the first time step of the simulation is done using the value set in line 28 and then it is automatically scaled to achieve the desired maximum values (lines 50-51).

Dam break free surface flow

The *controlDict* dictionary

```
58     functions
59     {

62
63     minmaxdomain
64     {
65         type fieldMinMax;
66
67         functionObjectLibs ("libfieldFunctionObjects.so");
68
69         enabled true; //true or false
70
71         mode component;
72
73         outputControl timeStep;
74         outputInterval 1;
75
76         log true;
77
78         fields (p U alpha.water);
79     }

109    };
```

- Let’s take a look at the **functionObjects** definitions.
- In lines 63-79 we define the **fieldMinMax functionObject** which computes the minimum and maximum values of the field variables (**p U alpha.water**).

Dam break free surface flow

The *controlDict* dictionary

```
58      functions
59      {

84      water_in_domain
85      {
86          type                cellSource;
87          functionObjectLibs ("libfieldFunctionObjects.so");
88          enabled             true;
89
90          //outputControl     outputTime;
91          outputControl      timeStep;
92          outputInterval     1;
93
94          log                 true;
95
96          valueOutput         false;
97
98          source              all;
99
100         operation           volIntegrate;
101         fields
102         (
103             alpha.water
104         );
105     }

109     };
```

- Let’s take a look at the **functionObjects** definitions.
- In lines 84-105 we define the **cellSource functionObject** which computes the volume integral (**volIntegrate**) of the field variable **alpha.water** in all the domain.
- Basically, we are monitoring the quantity of water in the domain.

Dam break free surface flow

The *fvSchemes* dictionary

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)  Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha)  Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}
```

- In this case, for time discretization (**ddtSchemes**) we are using the **Euler** method.
- For gradient discretization (**gradSchemes**) we are using the **Gauss linear** method.
- For the discretization of the convective terms (**divSchemes**) we are using **linearUpwind** interpolation method for the term **div(rhoPhi,U)**.
- For the term **div(phi,alpha)** we are using **vanLeer** interpolation. For the term **div(phirb,alpha)** we are using **linear** interpolation. These terms are related to the volume fraction equation.
- For the term **div(((rho*nuEff)*dev2(T(grad(U)))))** we are using **linear** interpolation (this term is related to the turbulence modeling).
- For the discretization of the Laplacian (**laplacianSchemes** and **snGradSchemes**) we are using the **Gauss linear corrected** method
- This method is second order accurate but oscillatory.
- Remember, at the end of the day we want a solution that is second order accurate.

Dam break free surface flow

The *fvSolution* dictionary

```
18   solvers
19   {
20       "alpha.water.*"
21       {
22           nAlphaCorr      2;
23           nAlphaSubCycles 1;
24           cAlpha          1;
25
26           MULESCorr       yes;
27           nLimiterIter    3;
28
29           solver          smoothSolver;
30           smoother        symGaussSeidel;
31           tolerance       1e-8;
32           relTol          0;
33       }
34
35       pcorr
36       {
37           solver          PCG;
38           preconditioner  DIC;
39           tolerance       1e-8;
40           relTol          0;
41       }
42
43       p_rgh
44       {
45           solver          PCG;
46           preconditioner  DIC;
47           tolerance       1e-06;
48           relTol          0.01;
49       }
```

- To solve the volume fraction or **alpha.water** (lines 20-33) we are using the **smoothSolver** method.
- In line 26 we turn on the semi-implicit method MULES. The keyword **nLimiterIter** controls the number of MULES iterations over the limiter.
- To have more stability it is possible to increase the number of loops and corrections used to solve **alpha.water** (lines 22-23).
- The keyword **cAlpha** (line 24) controls the sharpness of the interface (1 is usually fine for most cases).
- In lines 35-41 we setup the solver for **pcorr** (pressure correction).
- In lines 43-49 we setup the solver for **p_rgh**.
- FYI, in this case **pcorr** is solved only one time at the beginning of the computation.

Dam break free surface flow

The *fvSolution* dictionary

```
51        p_rghFinal
52        {
53            $p_rgh;
54            relTol          0;
55        }
56
57        "(U|Ufinal)"
58        {
59            solver          smoothSolver;
60            smoother        symGaussSeidel;
61            tolerance       1e-06;
62            relTol          0;
70        }
71    }
72
73    PIMPLE
74    {
75        momentumPredictor   yes;
76        nOuterCorrectors    1;
77        nCorrectors         3;
78        nNonOrthogonalCorrectors 0;
79    }
80
81    relaxationFactors
82    {
83        fields
84        {
85            ".*" 1;
86        }
87        equations
88        {
89            ".*" 1;
90        }
91    }
```

- In lines 51-55 we setup the solver for **p_rghFinal**. This correspond to the last iteration in the loop (we can use a tighter convergence criteria to get more accuracy without increasing the computational cost)
- In lines 57-70 we setup the solver for **U**.
- In lines 73-79 we setup the entries related to the pressure-velocity coupling method used (**PIMPLE** in this case). Setting the keyword **nOuterCorrectors** to 1 is equivalent to running using the **PISO** method.
- To gain more stability we can increase the number of correctors (lines 76-78), however this will increase the computational cost.
- In lines 81-91 we setup the under relaxation factors related to te PIMPLE method. By setting the coefficients to one we are not under-relaxing.

Dam break free surface flow

The **`system`** directory

- In the **`system`** directory you will find the following optional dictionary files:
    - *`decomposeParDict`*
    - *`setFieldsDict`*
    - *`probesDict`*

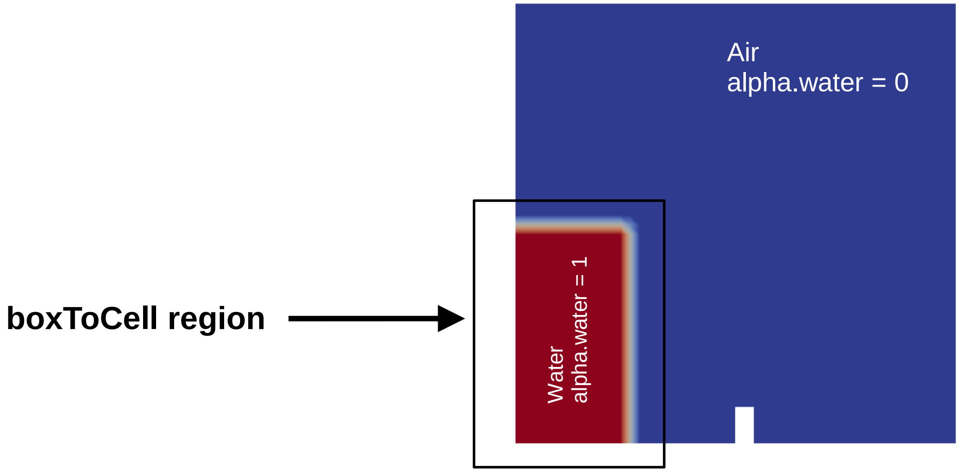
- *`decomposeParDict`* is read by the utility `decomposePar`. This dictionary file contains information related to the mesh partitioning. This is used when running in parallel.
- *`setFieldsDict`* is read by the utility `setFields`. This utility set values on selected cells/faces.
- *`probesDict`* is read by the utility `probeLocations`. This utility sample field values at a given location.

Dam break free surface flow

The *setFieldsDict* dictionary

```
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    );
```

- This dictionary file is located in the directory **system**.
- In lines 18-21 we set the default value to be 0 in the whole domain (no water).
- In lines 25-32, we initialize a rectangular region (**box**) containing water (**alpha.water 1**).
- In this case, `setFields` will look for the dictionary file *alpha.water* and it will overwrite the original values according to the regions defined in *setFieldsDict*.
- If you are interested in initializing the vector field **U**, you can proceed as follows **volVectorFieldValue U (0 0 0)**



Dam break free surface flow

The *probesDict* dictionary

```
fields
(
    alpha.water
    U
    p_rgh
    p
);

probeLocations
(
    (0.292 0 0)
    (0.292 0.0240 0)
    (0.292 0.0480 0)
    (0.316 0.0480 0)
    (0.316 0.0240 0)
);
```

Fields to sample.

Points location.

The sampled information is always saved in the directory **postProcessing/probes**

As we are sampling starting from time 0, the sampled data will be located in the directory:

postProcessing/probes/0

The files *alpha.water*, *p_rgh*, *p*, and *U* located in the directory **postProcessing/probes/0** contain the sampled data. Feel free to open them using your favorite text editor.

Dam break free surface flow

Running the case

- You will find this tutorial in the directory **`$PTOFC/101OF/damBreak`**
- In the terminal window type:

```
1. $> foamCleanTutorials
2. $> blockMesh
3. $> checkMesh
4. $> cp 0/alpha.water.org 0/alpha.water
5. $> setFields
6. $> paraFoam
7. $> interFoam > log.interFoam | tail -f log.interFoam
8. $> probeLocations
9. $> paraFoam
```

Dam break free surface flow

Running the case

- In step 2 we generate the mesh.
- In step 3 we check the mesh quality.
- In step 4 we copy the information of the backup file `alpha.water.org` to the file `alpha.water`. We do this because in the next step the utility `setFields` will overwrite the file `alpha.water`, so it is a good idea to keep a backup.
- In step 5 we initialize the solution using the utility `setFields`. This utility reads the dictionary *`setFieldsDict`* located in the **`system`** directory.
- In step 6 we use `paraFoam` to visualize the initialization. Remember to select the field `alpha.water` in `paraFoam`.
- In step 7 we run the simulation.
- In step 8 we use the utility `probeLocations` to sample field values at given locations. This utility reads the dictionary *`probesDict`*.
- Finally, in step 9 we visualize the solution.

Dam break free surface flow

- To plot the sampled data using gnuplot you can proceed as follows. To enter to the gnuplot prompt type in the terminal:

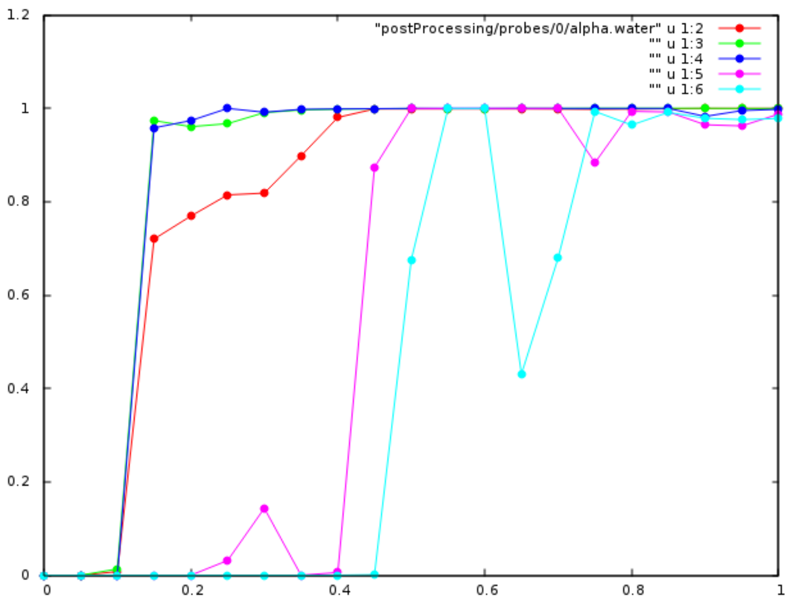
1.
```
$> gnuplot
```

- Now that we are inside the gnuplot prompt, we can type,

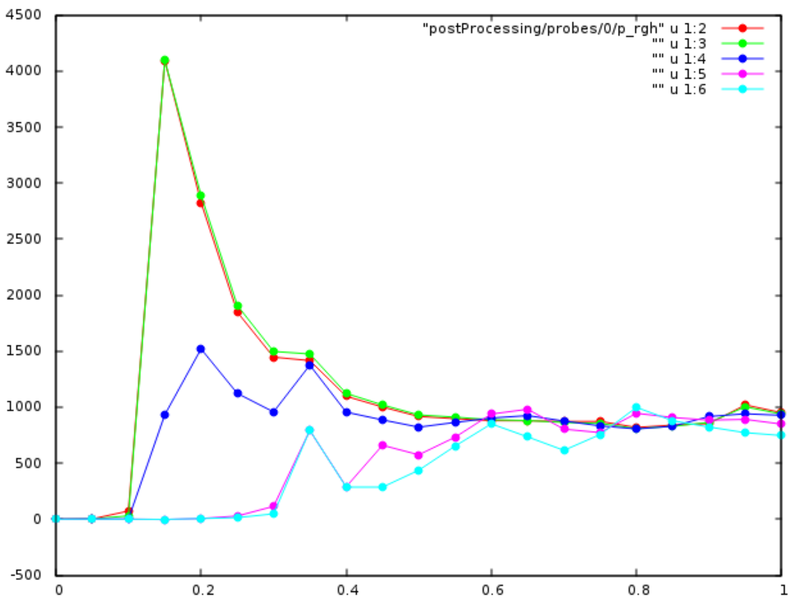
1.
```
gnuplot> plot [][0:1.2] “postProcessing/probes/0/alpha.water” u 1:2 pt 7 w lp,
        “ ” u 1:3 pt 7 w lp, “ ” u 1:4 pt 7 w lp,
        “ ” u 1:5 pt 7 w lp, “ ” u 1:6 pt 7 w lp
```
2.
```
gnuplot> plot [][] “postProcessing/probes/0/p_rgh” u 1:2 pt 7 w lp,
        “ ” u 1:3 pt 7 w lp, “ ” u 1:4 pt 7 w lp,
        “ ” u 1:5 pt 7 w lp, “ ” u 1:6 pt 7 w lp
```
3.
```
gnuplot> plot [][] “postProcessing/probes/0.05/p” u 1:2 pt 7 w lp,
        “ ” u 1:3 pt 7 w lp, “ ” u 1:4 pt 7 w lp,
        “ ” u 1:5 pt 7 w lp, “ ” u 1:6 pt 7 w lp
```
4.
```
gnuplot> exit
```
To exit gnuplot

Dam break free surface flow

- The output of steps 2 and 3 is the following:



alpha.water vs. time

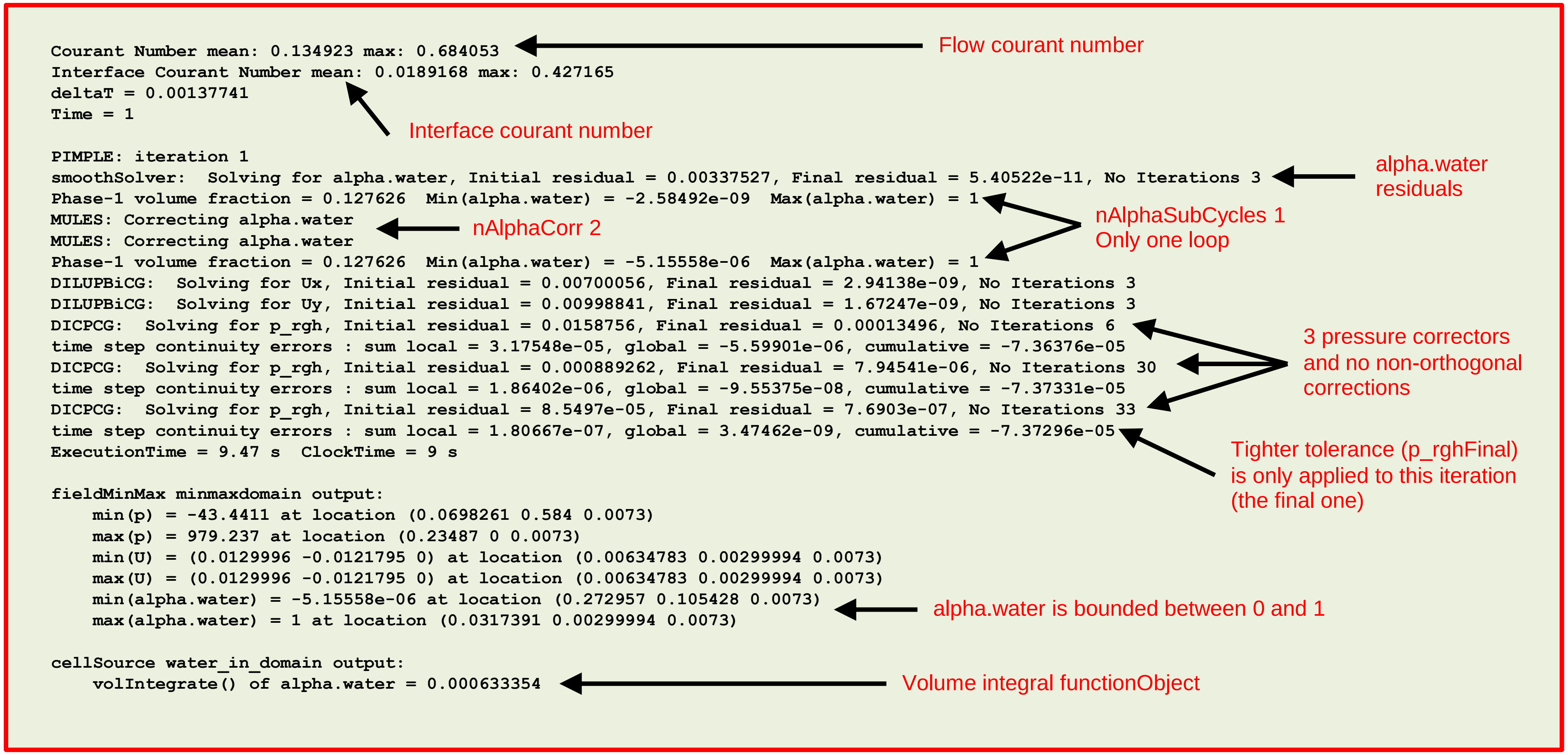


p_rgh vs. time

Dam break free surface flow

The output screen

- This is the output screen of the `interFoam` solver.
- The interface courant number is more restrictive than the flow courant number.
- When solving multiphase flows, is always desirable to keep the interface courant number less than 1.



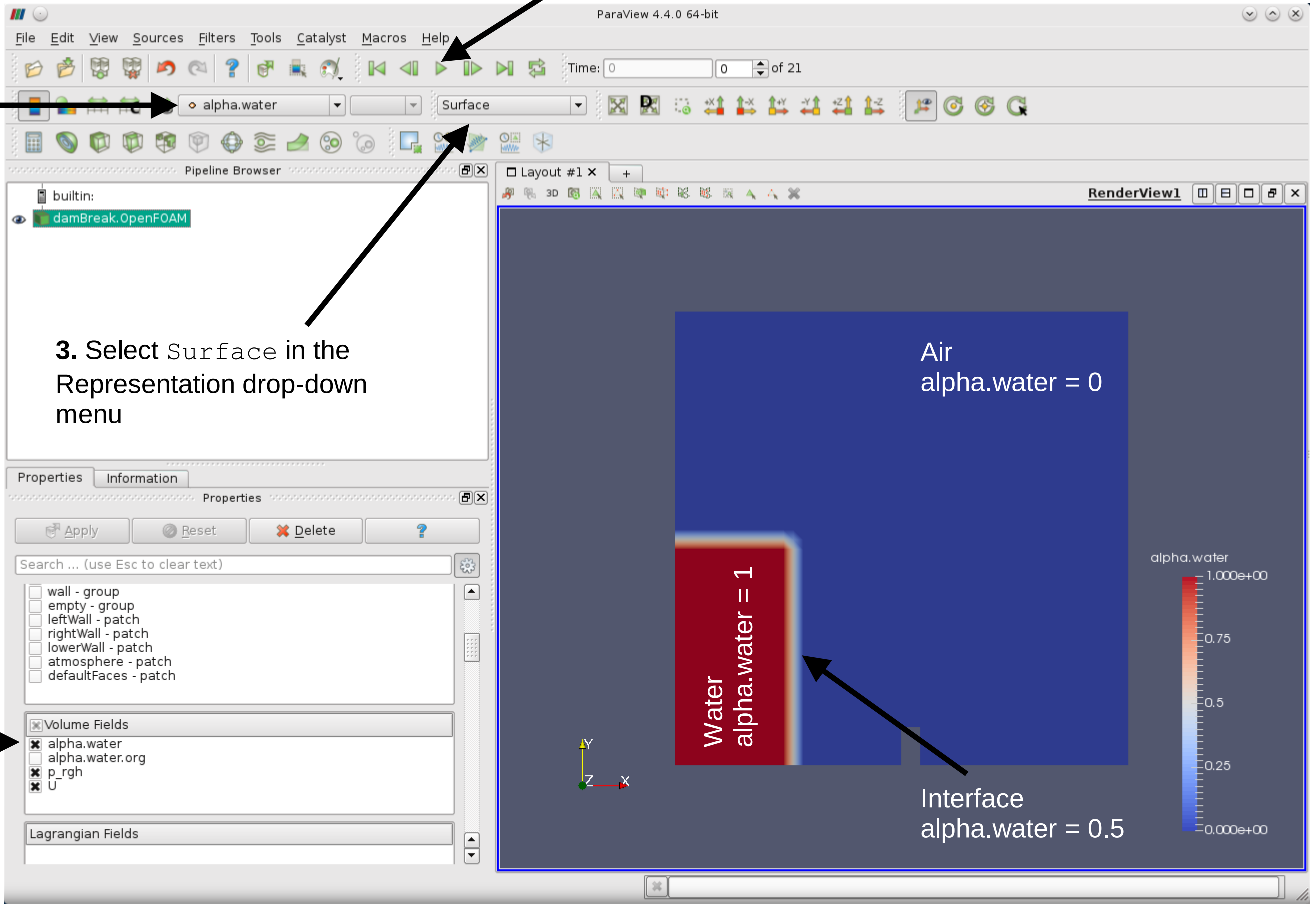
```
Courant Number mean: 0.134923 max: 0.684053
Interface Courant Number mean: 0.0189168 max: 0.427165
deltaT = 0.00137741
Time = 1

PIMPLE: iteration 1
smoothSolver:  Solving for alpha.water, Initial residual = 0.00337527, Final residual = 5.40522e-11, No Iterations 3
Phase-1 volume fraction = 0.127626  Min(alpha.water) = -2.58492e-09  Max(alpha.water) = 1
MULES: Correcting alpha.water
MULES: Correcting alpha.water
Phase-1 volume fraction = 0.127626  Min(alpha.water) = -5.15558e-06  Max(alpha.water) = 1
DILUPBiCG:  Solving for Ux, Initial residual = 0.00700056, Final residual = 2.94138e-09, No Iterations 3
DILUPBiCG:  Solving for Uy, Initial residual = 0.00998841, Final residual = 1.67247e-09, No Iterations 3
DICPCG:  Solving for p_rgh, Initial residual = 0.0158756, Final residual = 0.00013496, No Iterations 6
time step continuity errors : sum local = 3.17548e-05, global = -5.59901e-06, cumulative = -7.36376e-05
DICPCG:  Solving for p_rgh, Initial residual = 0.000889262, Final residual = 7.94541e-06, No Iterations 30
time step continuity errors : sum local = 1.86402e-06, global = -9.55375e-08, cumulative = -7.37331e-05
DICPCG:  Solving for p_rgh, Initial residual = 8.5497e-05, Final residual = 7.6903e-07, No Iterations 33
time step continuity errors : sum local = 1.80667e-07, global = 3.47462e-09, cumulative = -7.37296e-05
ExecutionTime = 9.47 s  ClockTime = 9 s

fieldMinMax minmaxdomain output:
    min(p) = -43.4411 at location (0.0698261 0.584 0.0073)
    max(p) = 979.237 at location (0.23487 0 0.0073)
    min(U) = (0.0129996 -0.0121795 0) at location (0.00634783 0.00299994 0.0073)
    max(U) = (0.0129996 -0.0121795 0) at location (0.00634783 0.00299994 0.0073)
    min(alpha.water) = -5.15558e-06 at location (0.272957 0.105428 0.0073)
    max(alpha.water) = 1 at location (0.0317391 0.00299994 0.0073)

cellSource water_in_domain output:
    volIntegrate() of alpha.water = 0.000633354
```

Flow courant number
Interface courant number
alpha.water residuals
nAlphaSubCycles 1 Only one loop
nAlphaCorr 2
3 pressure correctors and no non-orthogonal corrections
Tighter tolerance (p_rghFinal) is only applied to this iteration (the final one)
alpha.water is bounded between 0 and 1
Volume integral functionObject

Dam break free surface flow

Post-processing multiphase flows in paraFoam

- To visualize the volume fraction, proceed as follows,



1. In the Properties tab select **alpha.water** in Volume Fields

2. Select **alpha.water** in the Active Variable drop-down menu

3. Select `Surface` in the Representation drop-down menu

4. To animate the solution, press `Play` in the VCR Controls

wall - group
empty - group
leftWall - patch
rightWall - patch
lowerWall - patch
atmosphere - patch
defaultFaces - patch

Volume Fields
- [x] alpha.water
- [ ] alpha.water.org
- [x] p_rgh
- [x] U

Lagrangian Fields

Air
alpha.water = 0

Water
alpha.water = 1

Interface
alpha.water = 0.5

Dam break free surface flow

Post-processing multiphase flows in paraFoam

- To visualize a surface representing the interface, proceed as follows,

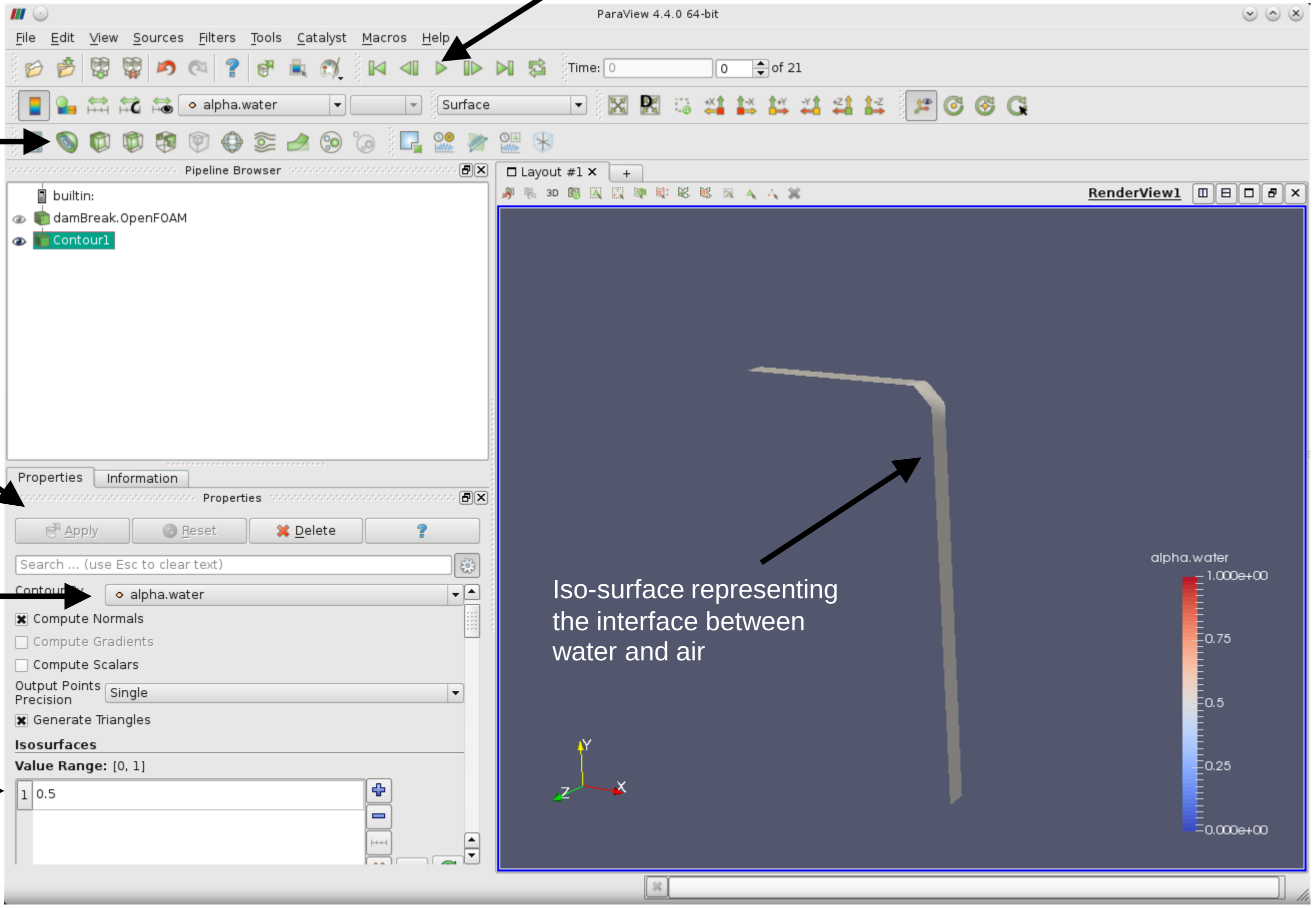
5. To animate the solution, press `Play` in the VCR Controls

1. Select the filter `Contour`

4. Press apply

2. Select **alpha.water** or the field you want to use to plot the iso-surface (it has to be a scalar)

3. Enter the value 0.5 which corresponds to the interface between water and air



Time: 0 | 0 of 21

alpha.water | Surface

builtin:
damBreak.OpenFOAM
Contour1

Search ... (use Esc to clear text)

Contour By: alpha.water
[✓] Compute Normals
[ ] Compute Gradients
[ ] Compute Scalars
Output Points Precision: Single
[✓] Generate Triangles

Isosurfaces
Value Range: [0, 1]
1 | 0.5

Iso-surface representing the interface between water and air

Dam break free surface flow

Post-processing multiphase flows in paraFoam

- To visualize all the cells representing the water fraction, proceed as follows,

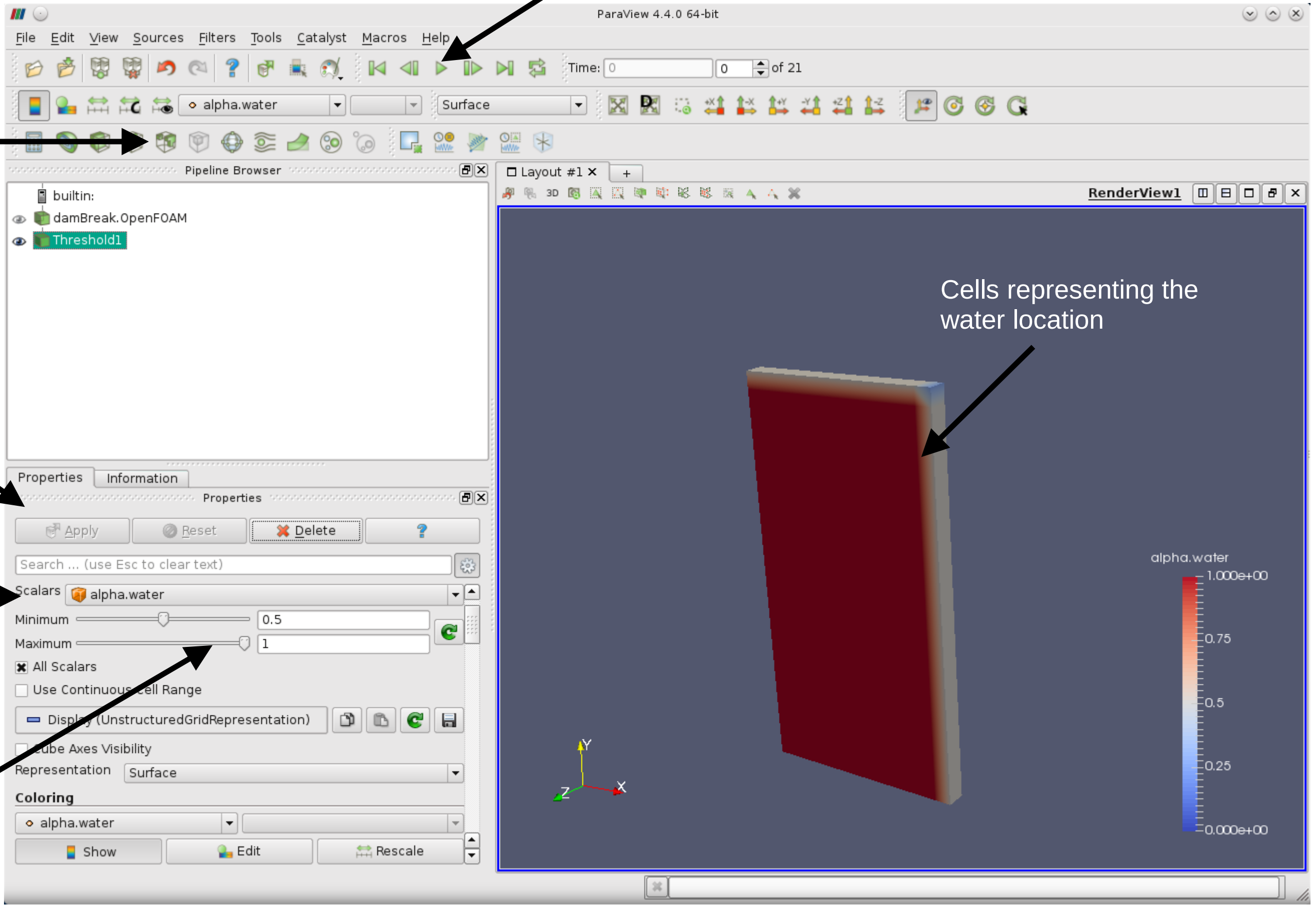
5. To animate the solution, press `Play` in the VCR Controls

1. Select the filter `Threshold`

4. Press apply

2. Select **alpha.water** or the field you want to use to visualize the cells (it has to be a scalar)

3. Select the range you want to visualize. To visualize the water select `Minimum` 0.5 and `Maximum` 1.



Scalars: alpha.water
Minimum: 0.5
Maximum: 1
[✓] All Scalars
[ ] Use Continuous Cell Range
Representation: Surface
Coloring: alpha.water