Tutorial 5: Free Surface Flow

Learning Outcomes:

Free surface calculations using Volume of Fluid, mapping values using `mapFields` and `setFields`, calculated boundary conditions

V.1 Flow over a broad crested wier

Free surface flow is a form of multiphase flow in which the phases occupy distinct regions of space separated by a macroscopic interface, whose location has to be determined. One commonly-used approach to simulating free surface flow is the Volume of Fluid or VOF method. In this, we solve for a 'mixture' fluid which can be either one of the phases (or at the interface, a blend) together with a transport equation for an *indicator function* α which takes the value 1 in one phase, 0 in the other. Here we will apply this to simulating flow of water over a broad-crested weir using the OpenFOAM code `interFoam`. Figure 1 shows the geometry; the weir is 1 m long and 1 m high; water is coming in from the left with velocity 1 m/s.

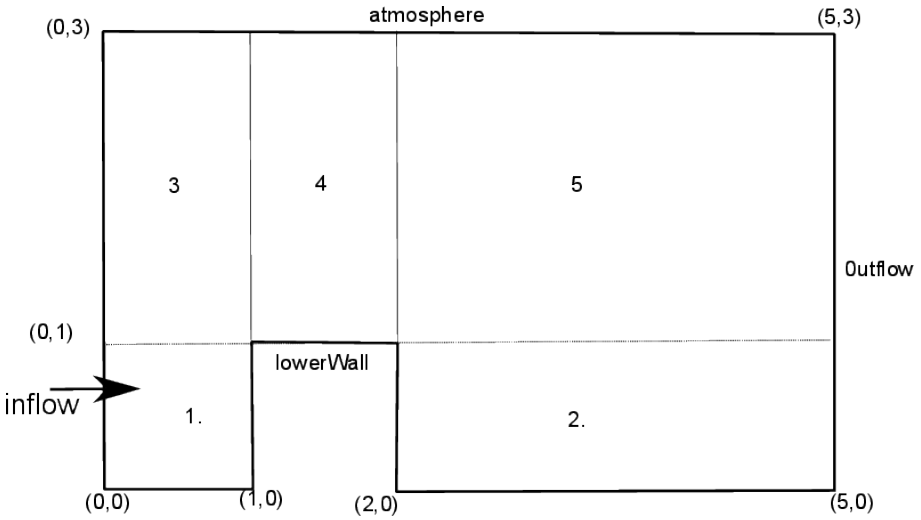

Figure 1: Geometry of the case.

A basic case is set up in the case directory `weir`. Note that the initial timestep directory is labelled `0.orig` for reasons which will be explained later. Run `blockMesh`, and have a look at the mesh. This is a 2d case with front and back patches type `empty`; the mesh consists of 5 blocks with boundary patches labelled as in figure 1.

V.1.1 Case setup

Free surface flows are often driven by gravity; hydrostatic effects are typically important (particularly in the water phase of an air/water system). `interFoam` solves for a modified pressure which factors out the hydrostatic term;

$$p^* = p - \rho g y \tag{V.1}$$

(with y being the direction gravity operates in). This is the variable `p_rgh` in the `0.orig` timestep directory. Similarly, the varying pressure with hydrostatic head must be correctly treated at outlets. The `outflow` patch in `p_rgh` makes use of the boundary condition `totalPressure` :

```
outflow
{
    type            totalPressure;
    p0              uniform 0;
    U               U;
    phi             phi;
    rho             rho;
    psi             none;
    gamma           1;
    value           uniform 0;
}
```

This links the outlet pressure patch to the other fields as necessary to correctly treat the pressures. The top boundary patch `atmosphere` is treated similarly.

Physical properties of the fluids are set in the dictionary `constant/transportProperties` for `water` and `air`. The `dimensionedScalar sigma` is the surface tension. The dictionary `g` defines gravity, whilst the type of turbulence modelling to be used is specified in dictionary `turbulenceProperties` (`interFoam` can be used in laminar, RANS or LES modes) in the usual way.

The field `alpha.water` is the indicator function : a value of 1 represents water, 0 air. If we were to initialise this field to 0 throughout, this would represent a completely empty channel, which would begin the simulation by being filled up from the inlet. This would be computationally very wasteful as this would take a significant number of timesteps to achieve as well as being prone to computational instability. It is best to initialise the `alpha.water` field so that those areas of the mesh which will be full of water start off so in the simulation; for this we use the `OpenFOAM` utility `setFields`. This takes a dictionary `system/setFieldsDict` which specifies a region of space the cells in which are to be set to a value of 1 – this is specified as a rectangular block of space by specifying opposite corners (0,0,0 and 1,1,1 in this case). This will rewrite the field, so it is a good idea to keep a copy safe to revert back to if necessary. The initial timestep directory is thus labelled `0.orig` and needs to be copied across;

```
cp -r 0.orig 0
setFields
```

This will fill the channel (block 1 in the mesh) to a height of 1 m.

The other problem with setting up a VOF case is how to specify the inlet boundary. The `inflow` patch extends right to the top; if the whole patch is defined with $\alpha = 1$ then this will represent a huge wall of water which will come crashing into the domain, which is physically incorrect and also almost impossible to compute, numerically. Instead we will use a computed boundary condition called `codedFixedValue`, which incorporates and compiles some OpenFOAM code at runtime to calculate which faces of the inlet patch are to have the value 1.0.

The `inflow` b.c. for `alpha.water` is set as

```
inflow
{
    type            codedFixedValue;
    value           uniform 1.0;
    redirectType    inletWater;
    code
    #{
        const fvPatch& boundaryPatch = patch();
        const vectorField& Cf = boundaryPatch.Cf();
        scalarField& field = *this;

        const scalar waterHeight = 1.0;

        forAll(Cf, faceI)
        {
            const scalar y = Cf[faceI].y();
            scalar indValue = 0.0;

            if (y<waterHeight)
            {
                 indValue = 1.0;
            }

            field[faceI] = indValue;
         }
    #};
}
```

(best to refer back to the field in `0.orig` for this). The `redirectType` allows us to name this particular boundary type, whilst the section between `#{...#}` is any valud C++

code – here, getting access to the `boundaryPatch` and face centres `Cf`, and the field `field`, checking to see if the y-coordinate of the face is below the set height of the water `waterHeight`, and setting it to 0.0 or 1.0 accordingly. Note that the first time the field is read in by any OpenFOAM code (probably `setFields` in this case), this code will be compiled, creating a directory `dynamicCode` in the case directory.

Having set up the case, run `setFields` and checked the various files, we can now run `interFoam` to run the case. This may take some time to execute. VOF calculations are strongly sensitive to the Courant number and are usually run with variable timestepping – activated by the following lines in `controlDict` :

```
adjustTimeStep  on;
maxCo           0.2;
maxAlphaCo      0.2;
maxDeltaT       1;
```

This will automatically adjust the timestep to ensure the Courant number never rises above 0.2. This has the side effect that if the calculation goes wrong, instead of the Courant number rising (as is typical with PISO calculations failing) instead the timestep gets shorter and shorter, usually unrecoverably.

[Q.V.1] Plot the evolution of the `alpha1` field as a series of colour plots. What do you notice about the level of the free surface upstream of the weir? How long does the calculation take to settle into a dynamic steady state?
[Q.V.2] Calculate the upstream head H generated by this flow rate Q over the weir. Use the general weir equation with $C = 1.6$. Note that the problem is 2d, so all extensive quantities will be "per unit length in the z-direction".

V.2 Modified calculation

As you will have noticed, the inlet is still not completely satisfactory; its level is incorrect, and the mesh upstream of the weir is too short – it is probably better to have a longer section of the computational domain upstream of the weir to ensure that the inlet conditions do not interfere with the flow over the weir itself. Modify the mesh to lengthen this section (extend the inlet section, and do not forget to increase the number of cells in blocks 1 and 3 to compensate for their changed dimension), and adjust the inflow patch to provide the correct depth of water at the inlet, adjusting the velocity inlet to maintain the same volumetric flow at the inlet. Rerun the calculation.

The mesh used here is quite coarse, and since the quality of the interface resolution depends on the fineness of the mesh we may want to recalculate on a finer mesh. However a fine mesh will take longer to compute, wasting time over the initial part of the simulation. The best approach is to perform the initial calculation on a coarse mesh until this reaches a dynamic steady state (or more generally reaches whatever flow physics we are

actually interested in simulating) and then to transfer or `map` the solution onto a new, finer resolution mesh. Copy the existing case to a new case `weirFine` and delete the saved timesteps. Modify the `blockMeshDict` to increase the mesh resolution, and run `blockMesh`. Then from the `tutorial5` directory (i.e. above both `weir` and `weirFine`) use the `mapFields` utility :

```
mapFields weir -case weirFine -sourceTime latestTime -consistent
```

This will map the fields from the `latestTime` time directory of case `weir` onto the initial timestep (`0`) of the new case `weirFine`. The `-consistent` flag specifies that the two domains are the same, differing only in mesh resolution, and so the fields can just be mapped across (otherwise a `mapFieldsDict` would need to be provided to specify how to transfer data between the cases).

[Q.V.3] Plot the water surface and velocity vectors for the modified calculation(s). Is the weir operating correctly for a broad-crested weir?