Wave Generator

In this tutorial, we shall model a planar wavemaker in shallow water. The problem involves simulating a multiphase, incompressible, fluid subjected to the influence of a horizontal wall velocity to replicate the generation of waves on the water surface. To represent a planar wavemaker in a simulation, the tutorial will focus on explaining setting fields, boundary conditions, and dynamic mesh techniques to capture the transient flow of a shallow water wave.

The test setup consists of a bath of water across a given volume. At the start of the simulation the water is acted on by a velocity which creates a wave. The test setup consists of an incompressible fluid domain across the tank representing shallow water. A vertical boundary sends a horizontal velocity to the water, creating a wave on the surface of the water. The geometry and initial setup is shown in *Figure 01.*

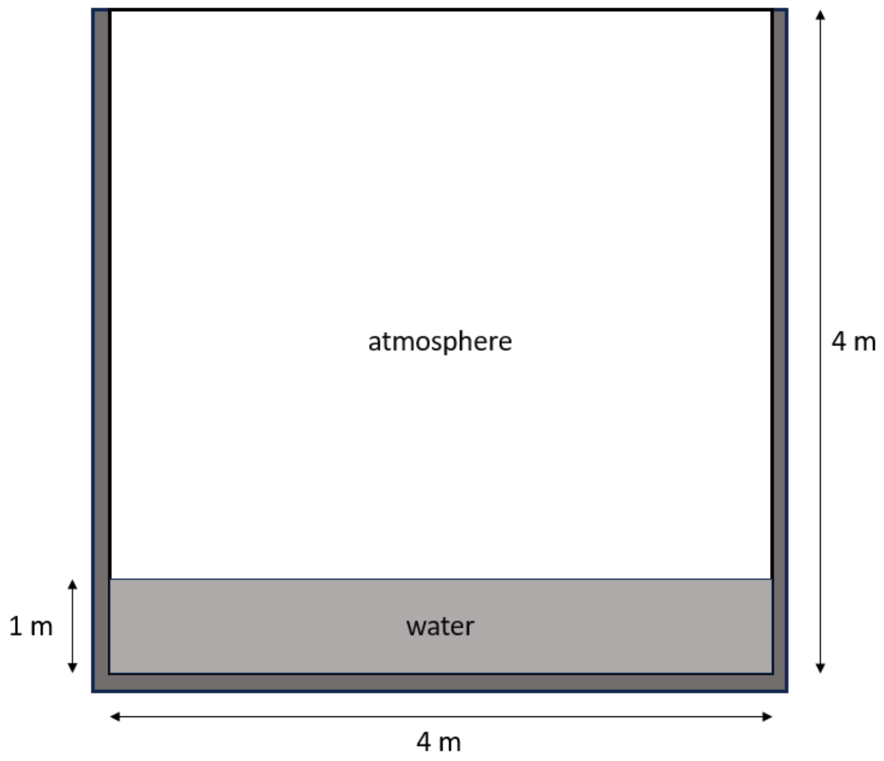

Figure 01: Geometry of wave generator.

Block Mesh

We shall translate the wave generator geometry using a block mesh dictionary. To simulate the domain of a 2D rectangle for our use case the user shall use the blockMeshDict file from the *PROJ011* file, which is a modified version of the “blockMeshDict” file from the OpenFOAM Dam Break tutorial[1]. The changes made were to ignore the obstacle geometry and convert the mesh dimensions to meters. To use the tutorial file provided(PROJ011), extract the *.zip* file to the users OpenFoam run directory. The user shall find the block mesh dictionary provided in the system folder of PROJ011. Exploring the *blockMeshDict* file is encouraged for those who are not familiar with block meshing.

To use the block mesh, make a copy of the project file and change to the newly created project directory (cd *$FOAM_RUN/PROJ/wavemaker/PROJ11* in my case). The practice of running a copy and keeping the original file is done to keep the original version of the setup unchanged by the effect of running commands. With the new directory created, execute the *blockMesh* command within the CFD terminal. The *blockMesh* command will generate the mesh provided by the block mesh dictionary. As a check, in paraview the user shall be presented with a geometry similar to *Figure 01*.

Boundary Conditions

Ensure the appropriate boundary conditions are applied to the mesh boundaries. Navigate to the 0 folder where the cell arrays are defined. There shall be three files in this folder: *alpha.water, p_rgh,* and *U.* The three files included in the 0 folder are necessary for the *interFoam* solver to be applied to the simulation. Notice the variety of boundary conditions within the U file:

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    leftWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    rightWall
    {
        type            noSlip;
    }
    lowerWall
    {
        type            noSlip;
    }
    atmosphere
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
    defaultFaces
    {
        type            empty;
    }
}
```

The *fixedValue* boundary condition creates a vector value for a velocity from the specified boundary. In this case the velocity is equal to 1 m/s in the horizontal direction from the left wall. The value along the left wall acts as a way to push the fluid from one direction of the box, which creates a propagating wave similar to one we would see in a planar wavemaker.

Set Fields

Once the boundaries have been constrained for the mesh, the water shall be added to create a multiphase condition. Set fields can be defined with different geometry types such as faces, points, and boxes. The bounds of the set fields are in the *setFieldsDict* file, the simulation for this tutorial uses a box. To achieve the desired dimensions of the set fields, the setfields shall be:

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

regions
(
    boxToCell
    {
        box (0 0 -1) (4 .5 1);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);
```

The variable for the *volScalarFieldValue* represents the phase of water determined in the *alpha.water* file and the domain of the water is established in the box region. Use the *setFields* command in the terminal to add the phase to the setup. The simulation is now a multiphase simulation.

Solver

The interfoam solver shall be used for this simulation. Interfoam requires a minimum of two inputs: pressure(p [Pa]), hydrostatic pressure(p_rgh [Pa]), and velocity(U [m/s]). The interfoam inputs are determined in the projects *0* folder, which was also utilized for setting the boundary conditions. Interfoam is an appropriate solver for this simulation since it can solve for time dependent, incompressible, multiphase flows such as the one set up in this tutorial. Execute the *interFoam* command in the terminal to solve the current setup, this should take about ten minutes. *For more information on interfoam see source 7.*

Post Processing

Use the *paraFoam* command once interfoam finishes solving to observe the simulation results. Within paraview apply the properties without changing anything and the user should see an interface similar to *Figure 02.* Select alpha.water in the visual representations tab to observe the water volume.

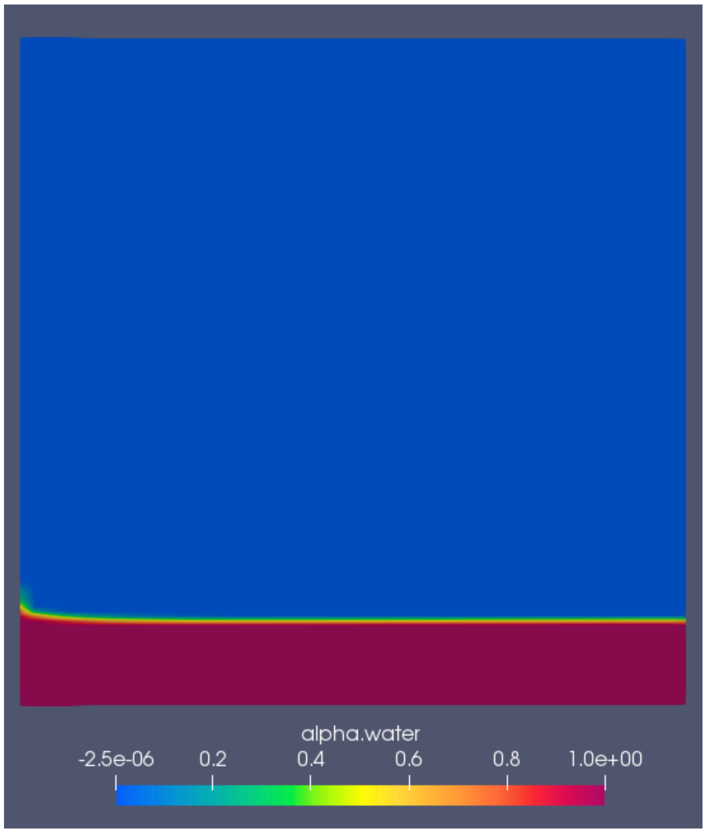

Figure 02: Initial paraview model representing water.

Play through the simulation using the time controls and observe the progressive wave created(*Figure 03)*.

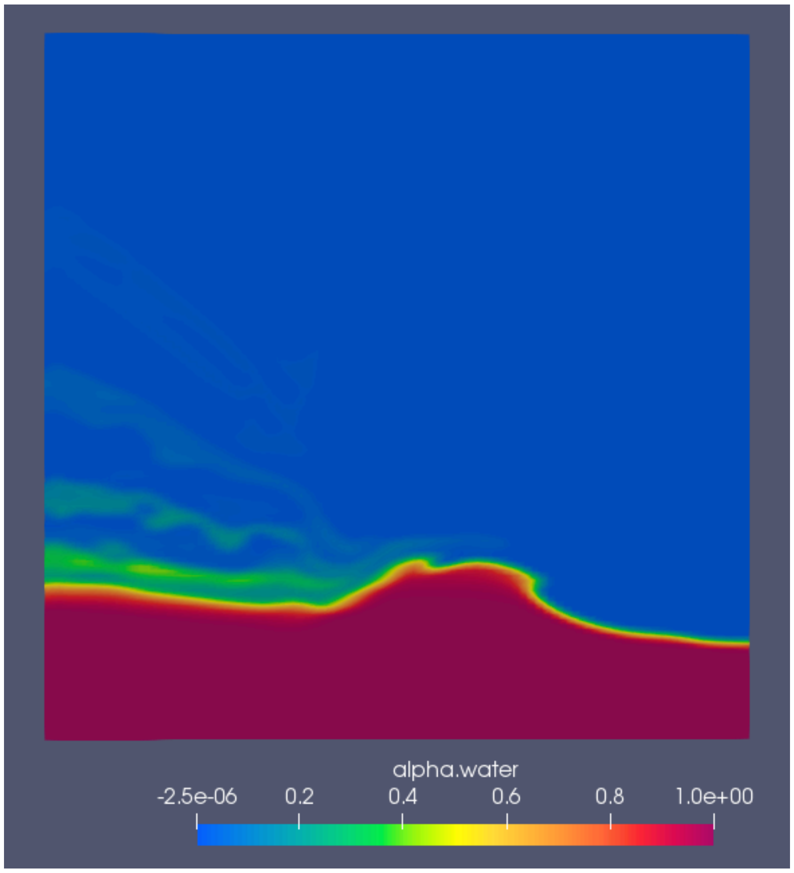

Figure 03: Progressive wave from simulation.

Use this tutorial to experiment with altering the test geometry, run time, and wave velocities to get different wave results such as the wave shown in *Figure 04.* Compare the results of the new simulation to the original. A 5 second run time over a larger domain is more computationally expensive. Compare the 10 minutes it takes to run the first simulation to the 1 hour 10 minutes it takes to run the second simulation on my system.

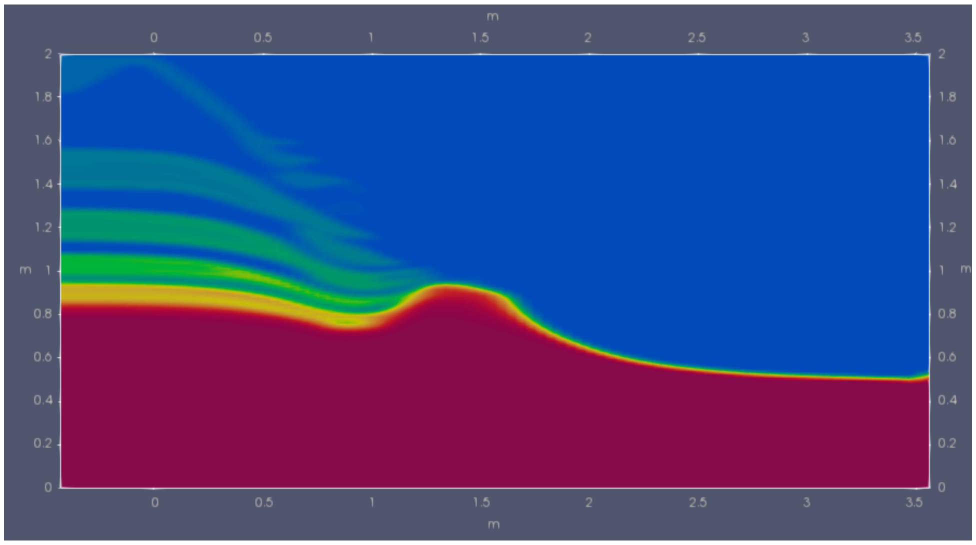

Figure 04: Simulation with altered mesh size and run time.

Future plans

Due to the nature of this problem setup the simulation will only produce one progressive wave, like the one you would see in an actual wavemaker. The problem is caused by the velocity from the left wall constantly acting on the fluid. To solve this I would like to be able to implement Dynamic Meshing. The dynamic mesh would allow for linear oscillatory motion and the left face could be defined by a topological mesh dictionary so that it's the only oscillating face.

<u>Sources</u>:

1. Greenshields, Chris. "OpenFOAM V8 User Guide." *CFD Direct*, 7 July 2023, doc.cfd.direct/openfoam/user-guide-v8/dambreak.
2. "Technical Documents." *XSim*, www.xsim.info/articles/OpenFOAM/en-US/tutorials/incompressible-pimpleDyMFoam-oscillatingInletACMI2D.html.
3. "Technical Documents." *XSim*, www.xsim.info/articles/OpenFOAM/en-US/tutorials/multiphase-potentialFreeSurfaceDyMFoam-oscillatingBox.html.
4. Dean, Robert G., and Robert A. Dalrymple. *"water Wave Mechanics for Engineers and Scientists (Advanced Series on Ocean Engineering ; V. 2)."* World Scientific, 1991.
5. "Dynamicmeshdict." *OpenFOAMWiki*, openfoamwiki.net/index.php/DynamicMeshDict.
6. "Toposet." *OpenFOAMWiki*, openfoamwiki.net/index.php/TopoSet.
7. *Description and Utilization of Interfoam Multiphase Solver - Cfdyna.Com*, www.cfdyna.com/Home/OpenFOAM/of_Tut_Web/of_Suites_Description.pdf.