Functionality

The code initialises a rectangle boundary and fluid particles.

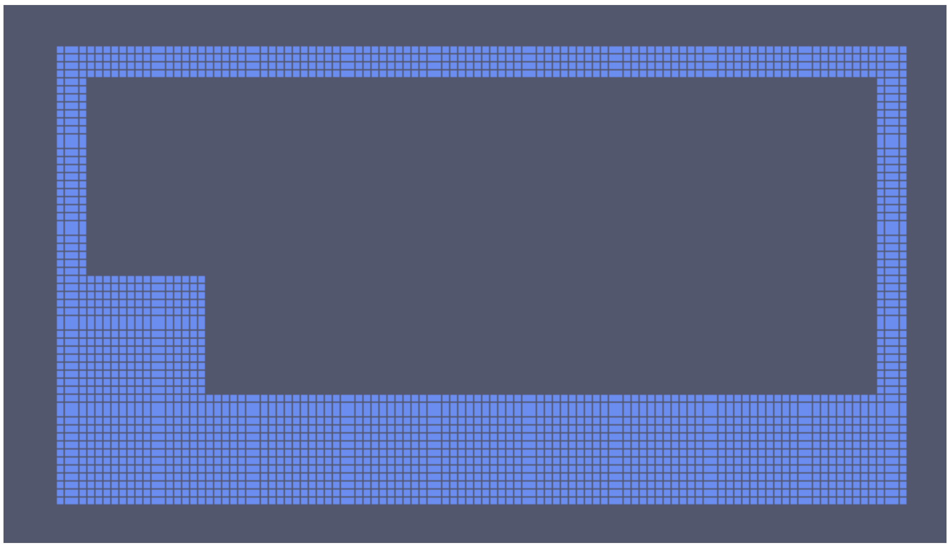

Fig.1 Initial fluid layout setup

The user is able to input time, particle distance dx, and numerical solver (Forward Euler / Predictor Corrector). It then computes the updated particles status based on the configuration.

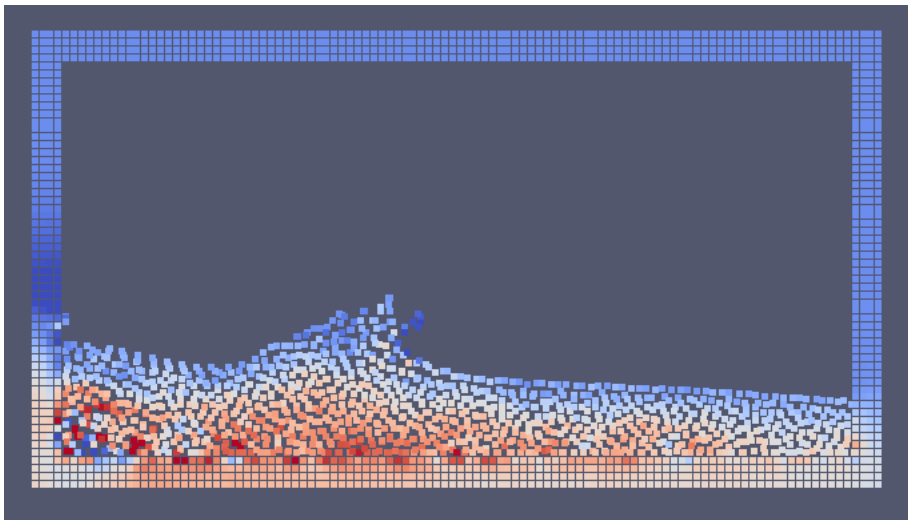

Fig.2 Fluid flow simulation after a few time steps. (Pressure profile)

Forward Euler scheme is realised to update the status of particles. The solver will smooth the densities among particles every 20 timesteps and checks every particles' neighbour before updating.

We implemented a stencil to make the 'neighbour_iterate' function more efficient. The original algorithm has a cost complexity of second order, while the stencil algorithm only has a complexity of first order. However, due to the boundary method we used(the pushback particle method), the simulation has leakage of particles, which causes allocate_to_grid function breaks down. To show our effort made, we still put the stencil option inside the program although it has unsolved bugs. If time allows, the bugs should be fixed by changing the boundary method(See the next section for boundary method details).

We wrote predictor corrector scheme to update the status of particles. The previous step density, velocity and positions are stored before the updates to perform two-step status update. Similar to Euler's method, it smooths the densities and checks neighbours before performing the update. The boundary method for this schme is the same as Euler's Method's.

Both solvers implemented is able to produce a 30-seconds simulation with output vtp files.

Boundary method

When the fluid particles are going to hit the wall (boundary particles), we applied the push back method to deal with problem of leak.

We firstly compute the updated the position of particles, then we check whether it will hit the wall. If the fluid particles hit the wall, we do not update the position for this update, namely we fix the portion of particles this process of update. Meanwhile, we change the corresponding direction of velocity, and add a lost rate of velocity to decrease the velocity. For example, if the particle is going to hit the left boundary, we do not change the position of particle for this process of update. And we only change the horizontal velocity of particle to opposite direction (namely right), and we multiply the original horizontal velocity with a lost rate.

Another un-implemented boundary method to deal with particles leakage is to apply artificial normal forces on the particles from the boundaries once the particles is closed to the boundary layer. The Lenard Jones potential theory may be applied for this method.

Parallel programming

We applied OpenMP to accelerate the computation within program. The time improvement is huge.

Domain Geometry

Currently the shape of the fluid container is rectangular. Because we applied push back boundary method, so it is difficult to add arbitrary boundary shape. If time allows, artificial forces should be applied to the acceleration term for the particles that are close to the wall. By this, arbitrary boundary shapes can be configured in the set up and the fluid will not leak out of the boundary.

Convergence Analysis

Due to time limit, our group have not yet compare the numerical results obtained with the analytical prediction. If time allows, the analysis can be conducted using the time of wave crest in each time step with different dx.

Testing

We use pytest and Travis to test if the program is properly written.

Post-Processing

The program can output the result files in vtp format. We the use Paraview to generate animation, or use python to generate the gif. We implemented crest velocities tracking in Jupyter Notebook.

Performance

A simple runtime performance is recorded in README.md Parallel Programming section.