UNIVERSITY OF EXETER

REPORT

Multiphase flow and Multiple Reference Frame simulations

Andrej Vasilj
3/6/2015

Contents

List of pictures

List of videos

Multiphase flow

General

Multiphase flow is a simultaneous flow of multiple phases. A phase represents (Wallis, 1969) one of the states of matter, which can be gas, liquid or solid. Examples of multiphase flow can be easily found in our surroundings, i.e. rain, snowfall, water flowing out of a pipe etc. To be exact, most of the flows are multiphase flows but single phase flow assumption is usually used to evade multiple problems with formulation of multiphase flow. I.e. water contains different impurities and bubbles which can be neglected in most of the cases but there are phenomena like cavitations and boiling where they have a key role.

Classification of multiphase flows

General classification (Ishii, 1975) divides two phase flows into four groups depending on the constituents of the flow: gas-solid, gas-liquid, solid-liquid and flows of two immiscible fluids. Ishii also introduces a division based on the topology of the flow. Those categories are: separated flow, mixed flow and dispersed flow.

There are many ways, depending on averaging procedure and the closure laws, to formulate two phase flows. Out of them, there are two major approaches to simulate multiphase flow: Euler-Euler and Euler-Lagrangian approach.

Euler-Lagrangian approach

This approach is an alternative to Euler-Euler approach and is often used when second phase is very dispersed, in other words, in cases when volume ratios of the dispersed phase are lower than 5%. Euler-Lagrangian approach enables decoupling of volume ratio equations for dispersed and continuous phase which makes calculations very computationally efficient. On the other hand, it can result in too large number of equations for cases with many particles.

Euler-Euler approach

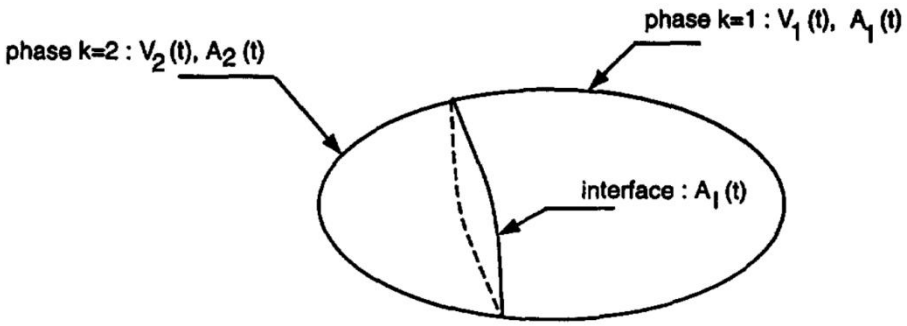
In Euler-Euler approach different phases are mathematically considered as interpenetrating types of continuum. Since the volume occupied by one phase can not be occupied by the other phase at the same time, concept of volume fractions is introduced. Volume fractions are continuous functions in space and time and their sum is equal to zero. Conservation laws are derived for every phase to get a set of equations that is similar for all phases. The set is closed with a definition of constitutive relations that are obtained from empirical information or, for granular flow, from kinetic theory.

The general idea is to first formulate integral balances for mass, momentum and energy for a fixed control volume containing both phases. This balance must be satisfied at any point in space and time. That enables us to diversify two types of local equations. One is instantaneous local equations for each phase ant the other represents local jump conditions at the interface. Generally speaking, these equations could be solved by a direct simulation if the mesh was finer than the smallest length scales and timestep is finer than the fastest system fluctuations. Since the number of particles or droplets is large, averaging procedures

are necessary to make the equations solvable. The most important averaging procedures are space, time and ensemble average.

For the Eulerian approach local instantaneous equations must be averaged in a suitable way, either in space, in time or as an ensemble. This allows a coarser mesh and a longer time step to be used in the numerical simulation, but also introduces more unknowns than the number of equations into the system, and therefore, it is necessary to include additional expressions to close the set of equations. The closure laws are of three types: topological, constitutive and transfer laws, where the first type describes the spatial distribution of phase-specific quantities, the second type describes physical properties of the phases and the third type describes different interactions between the phases.

Let's consider a general control volume fixed in space and shared by two phases with phase index *k*.

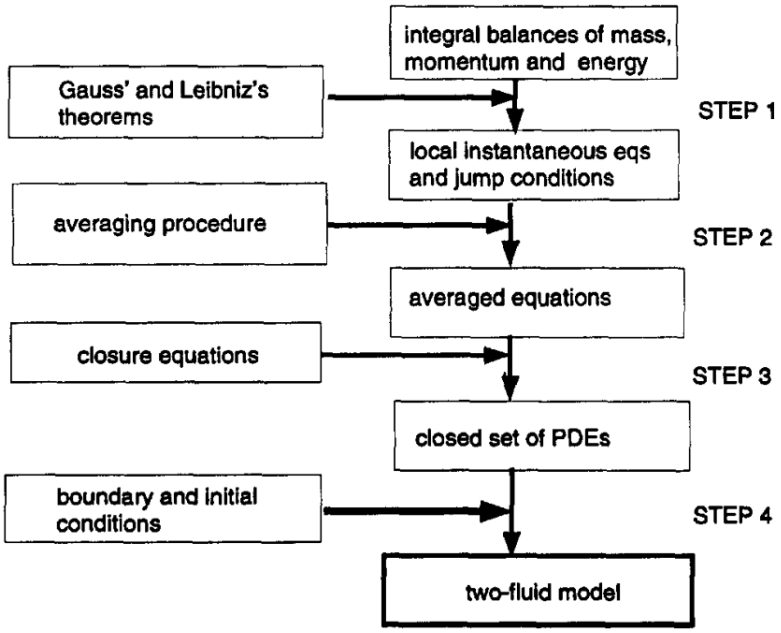


Picture 1 Control volume with two phases

The interface with the area between the phases $A_I(t)$ is moving with the velocity **u_I.** When a vector or scalar variable ψ_k belonging to phase *k* is to be transported through the control volume using a coordinate system which is fixed (Eulerian approach), the following integral balance can be written

$$\sum_{k=1}^{2}\left(\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_k(t)}\rho_k\psi_k\mathrm{d}V\right)=\int_{A_I(t)}\Phi_I\mathrm{d}A+\sum_{k=1}^{2}\left[-\int_{A_k(t)}\rho_k\psi_k(u_k\cdot n_k)\mathrm{d}A+\int_{V_k(t)}\rho_k\Phi_k\mathrm{d}V-\int_{A_k(t)}J_k\cdot n_k\mathrm{d}A\right]+\oint_l\sigma n_I\mathrm{d}l$$

In this notation, **n_k** is the outwardly directed normal unit vector to the interface of the volume occupied by phase k, d/dt is the ordinary time derivative, **u_k** is the velocity of phase k, ρ_k is the density, ψ_k is the conserved quantity, **J_k** is the molecular flux, Φ_k is the source term, Φ_I is the interfacial source term, σ is the surface stress tensor and n_I is the unit normal vector located in the tangent plane and directed outward the area $A_I(t)$.



Picture 2 Development of the Two Phase model

Applying Gauss' and Leibnitz's theorems the equation is transformed

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_k(t)}\rho_k\psi_k\mathrm{d}V=\int_{V_k(t)}\frac{\partial}{\partial t}(\rho_k\psi_k)\mathrm{d}V+\int_{A_I(t)}\rho_k\psi_k u_I\cdot n_k\mathrm{d}A+\int_{A_k(t)}\rho_k\psi_k u_{Bk}\cdot n_k\mathrm{d}A$$

$$\int_{V_k(t)}\nabla\cdot(\rho_k\psi_k u_k)\mathrm{d}V=\int_{A_k(t)}\rho_k\psi_k u_k\cdot n_k\mathrm{d}A+\int_{A_I(t)}\rho_k\psi_k u_k\cdot n_k\mathrm{d}A$$

$$\int_{V_k(t)}\nabla\cdot J_k\mathrm{d}V=\int_{A_k(t)}J_k\cdot n_k\mathrm{d}A+\int_{A_I(t)}J_k\cdot n_k\mathrm{d}A$$

When we rewrite the conservation equation with abovementioned terms in mind

$$\sum_{k=1}^{2}\int_{V_k(t)}\left[\frac{\partial}{\partial t}(\rho_k\psi_k)+\nabla\cdot(\rho_k\psi_k u_k)+\nabla\cdot J_k-\rho_k\Phi_k\right]\mathrm{d}V-\int_{A_I}\left[\sum_{k=1}^{2}(\dot{m}_k\psi_k+J_k n_k)+\Phi_I\right]\mathrm{d}A=0$$

Volume integral for volumes occupied by two phases

Jump conditions across the interface

$\dot{m}_k=\rho_k(u_k-u_I)\cdot n_k\rightarrow$ Mass flow through the interface

The surface tension term is left out in the above equation just to show the structure of the equation. For the complete formulation, please see Delhaye (1974).

Modelling two-phase flows is a difficult task, both from a mathematical and a physical point of view. The mathematical difficulty lies in the formulation of the two-phase flow as two single phases with moving boundaries, while the physical difficulty lies in the modelling of the interaction between the phases at the interface. If the number of particles of dispersed phase suspended in the main flow is large, an averaging operator acting on the local instantaneous equations is needed (the alternative would be to solve one ordinary differential equation for each particle using a Lagrangian approach). More on general averaging techniques can be found in Ishii's book (1975), publications of Delhaye and Achard (1977, 1978) and publication by Delhaye (1981).

The calculations were performed using OpenFOAM, an open source CFD tool. Meshing was done using blockMesh dictionary, where two structured grids were generated. One coarse ad one fine to be able to compare results and discern what kind of impact meshing has on data quality. Unfortunately during my stay in Exeter, I wasn't able to find a paper that has all experimental data disclosed to be able to validate my results. Usually the geometry, initial or boundary conditions were missing and the remaining data wasn't enough to create, at least, a similar case where the results would be comparable.

alpha_fine_kEps_250s.avi

Video 1 Mixing of two phases

This video shows flow patterns and distribution of phases in a bubble column. Red colour specifies cells with 100% air inside, while blue cells represent 100% water in them. Colours in between represent cells with volume ratios between 0 and 1. In other words, those are cells that contain both water and air. It is clear that the flow is unstable and it will affect mixing within the tank. To show it better, animations with velocity vectors are enclosed.

k_u1_u2_17_22_10fps.avi

Video 2 Flow patterns and turbulent kinetic energy

This particular animation shows velocity vectors for both phases and turbulent kinetic energy in the background. Strong current directed upwards due to rising of bubbles creates downflow next to tank walls. At certain regions, the downflow is strong enough to create downward motion of bubbles.

reverse_5fps.avi

Video 3 Reverse flow- reduced frame rate

Velocity vectors for air show the movement of bubble column. The shape is constantly changing and a slight downward motion of bubbles can be observed next to the walls due to strong downwards water current.

u2_17_22_downflow_5fps.avi

Video 4 Reverse flow- water velocity glyphs

The work done on bubble columns needs to be validated against experimental data which can be done easily as soon as full dataset and geometry is obtained. The most time consuming part of any simulation, setting up the workflow, is completed and optimized.

Multiple Reference Frame

General

A Moving Reference Frame (MRF) is a relatively simple, robust, and efficient steady-state, Computational Fluid Dynamics (CFD) modelling technique to simulate rotating machinery. For example, the impeller in a stirred tank can be modelled with MRFs. An MRF assumes that the assigned volume has a constant speed of rotation and the non-wall boundaries are surfaces of revolution (e.g., cylindrical, spherical, conical). In the case of the stirred tank test case the volumes around the impeller blades are designated as MRFs, assigned rotational speeds, and embedded within a multi-volume flow domain.

MRF is also known as "frozen rotor approach" since it is equivalent to running a rotational simulation and then observing the results only at the instants equivalent to the position of the rotor within the MRF. MRF assumes a weak interaction between the MRF volume and the surrounding stationary volumes. Therefore, MRF boundaries have to be placed sufficiently far away from the impeller blades.

An alternative to MRF is the moving/sliding mesh technique, which can deal with strong interactions between the moving volume and the surrounding stationary volumes. However, in practice the moving mesh technique often has robustness problems due its dependence on intersection calculations between the stationary and rotating volumes. Also the moving mesh technique is a transient (unsteady) simulation procedure which typically results in extremely long runtimes. Given the significant drawbacks of the moving mesh technique, the MRF is the preferred approach within its limitations.

Grid resolution

A CFD model of a stirred tank requires, among other things, an appropriate grid resolution, discretization scheme, impeller rotation model and turbulence model. That choice will have a drastic influence on both accuracy of the solution and computational time. That is particularly important when modelling detailed structures or large vessels since they require large meshes/fine spatial discretization. Since our mesh sizes are limited by existing computer performances, we can't increase our cell count as much as we would like. Furthermore, the resolution of the mesh is a key factor in any CFD calculation since it directly affects the computational cost and accuracy of solution. Therefore, CFD engineers always have to find a balance between accuracy and computational time. There are many papers where mesh sensitivity analysis was done (Ng et al., 1998; Montante et al., 2001; Wechsler et al., 1999). They show that successive refinement yields equivalent predictions of mean velocity profiles, which are in good agreement with experimental LDV data but turbulent kinetic energy was severely under-predicted on all grids, with the finer grids showing only marginal improvement. Relatively small improvement on finer grids led authors to conclude that further grid refinement wasn't likely to make larger improvements in turbulence predictions.

Discretization scheme

Some authors (Brucato et al., 1998; Aubin et al., 2004) have investigated the effects of different discretization schemes on accuracy of the solution and in most cases it was shown that the choice had little or no effect on the solution. In addition to this, it was shown that all of the discretization methods used under predict turbulence, this being most severe for first order upwind scheme.

Impeller rotation

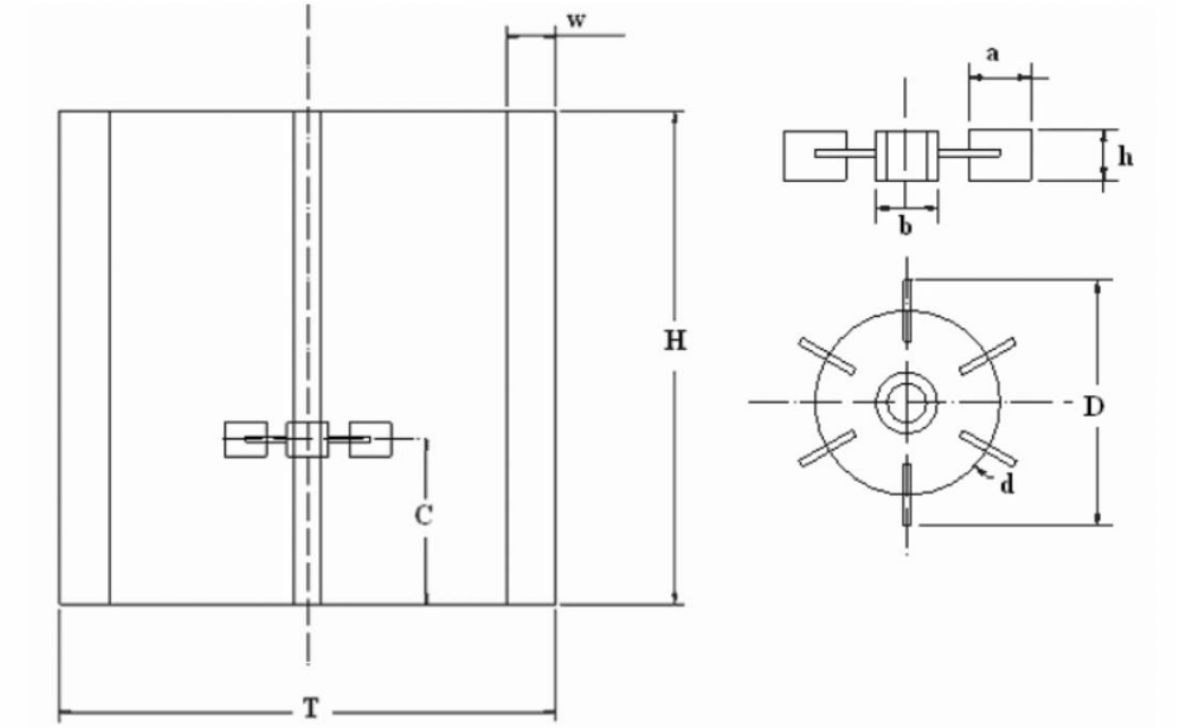
Modelling the impeller rotation is complex as the relative motion between the rotating impeller blades and the stationary baffles causes a cyclic variation of the solution domain. Two commonly used models are the Multiple Reference Frames and Sliding Mesh models. In these models, the solution domain is divided into an inner region containing the rotating impeller and an outer region containing the stationary baffles. While for MRF steady state calculations are performed with a rotating reference frame in the impeller region and a stationary reference frame in the outer region, where the effects of blade motion are accounted for by the definition of a moving reference frame, for Sliding mesh model the impeller region is allowed to slide relatively to the stationary outer region in discrete time steps. In general, Sliding mesh approach is more accurate since it's time dependency is a more accurate representation of actual phenomena but is more computationally expensive. In this particular case, it was shown that MRF calculations give similar results to more complex Sliding mesh model for steady state calculation of a stirred tank, suggesting that additional computational expense is unnecessary (Koh et al., 2003; Koh and Schwartz, 2005).

Turbulence modelling

When it comes to turbulence modelling, *k*-ò is the most widely used model for engineering purposes and is often used for stirred tanks. Poor prediction of this model is often attributed to deficiencies in the model, especially the inherent assumption of isotropic turbulence and limitations in predicting swirling or recirculating flow (Abujelala and Lilley, 1984; Armentante et al.,1997; Jenne and Reuss, 1999). Several authors compared results from *k*-ò to Chen-Kim and Renomalised Group (RNG) *k*-ò models (Ranade et al., 1989; Jenne and Reuss, 1999; Jaworski and Zakrzewska, 2002; Aubin et al., 2004). Different models gave only slight changes in turbulent predictions and, in some cases, standard model gave superior results. One of the suggestions was that a model that doesn't assume isotropic turbulence might yield better results. Armentante et al. (1997) found that the Algebraic Stress Model actually gives better results than standard model. However, in some published studies the Reynolds Stress Model based on non-isotropic turbulence was found to yield turbulent kinetic energy profiles showing a larger deviation from experimental values than those obtained using the standard *k*-ò model. (Montante et al., 2001; Jaworski and Zakrzewska, 2002). Large Eddy Simulations are said to give good alignment of numerical results and experimental data (Eggels, 1996; Derksen and Van Den Akker, 1999; Hartmann et al., 2004). However, since LES simulations require long computational time and very fine mesh, they are impractical for many research problems.

Geometry

The system investigated in this study consists of a 15 cm diameter cylindrical tank with four equally spaced baffles and agitated by a standard six-bladed Rushton turbine impeller. The tank is filled with water at 25°C. Since the configuration is standard and relatively small it is possible to test the case with both coarse and very fine meshes. The results are compared with the experimental data of Wu and Patterson (1987). The authors published phase-averaged experimental data obtained with LDV. They are presented in form of axial profiles of mean velocities profiles and RMS turbulent velocities at different radial distances from the impeller, measured in a plane mid way between the baffles. Wu and Patterson (1989) used a larger tank of equivalent geometry to conduct their testing. However, experimental studies have shown that for turbulent flow in geometrically similar vessels, velocity values normalized by the impeller tip speed are independent of the rotational speed of the impeller or the size of the stirred tank (Costes and Courdec, 1988; Wu and Patterson, 1989; Dyster et al., 1993). Therefore, results from this study are comparable to experimental data from Wu and Patterson (1989).

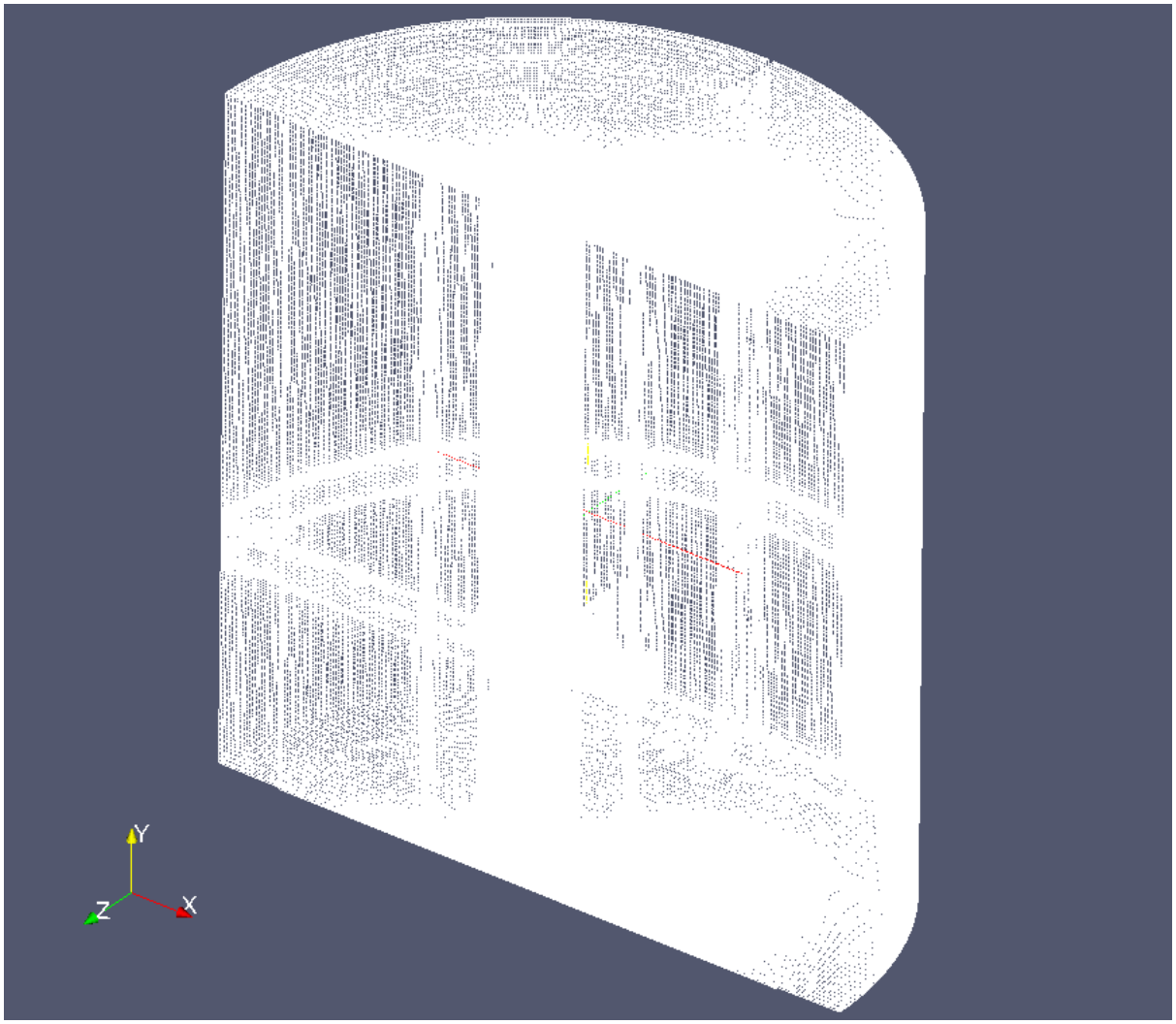


H/T	C/T	w/T	D/T	d/D	a/D	h/D	b/D
1	1/3	1/10	1/3	3/4	1/4	1/5	1/5

Picture 3 Geometry

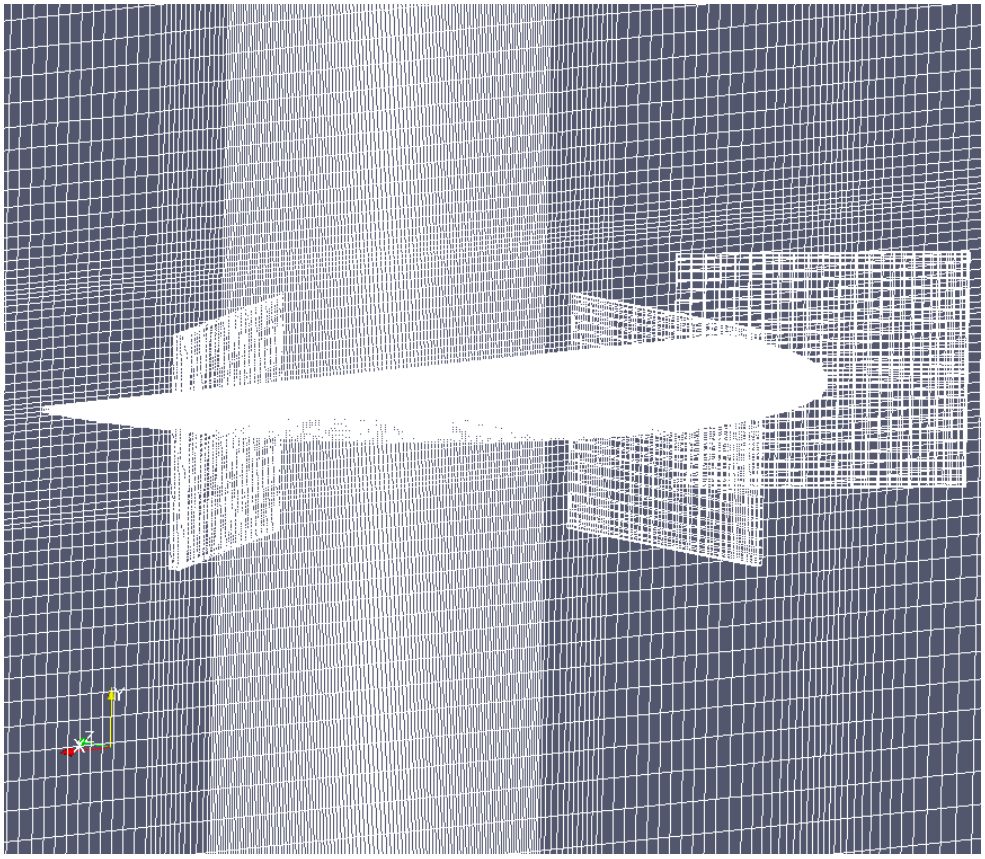
Mesh

Since the stirred tank shows rotational symmetry, only one half of the geometry was used as the computational domain using cyclic boundary condition. The mesh was built as structured in Pointwise with a cell count of approximately 6M, as can be seen in Picture 4.

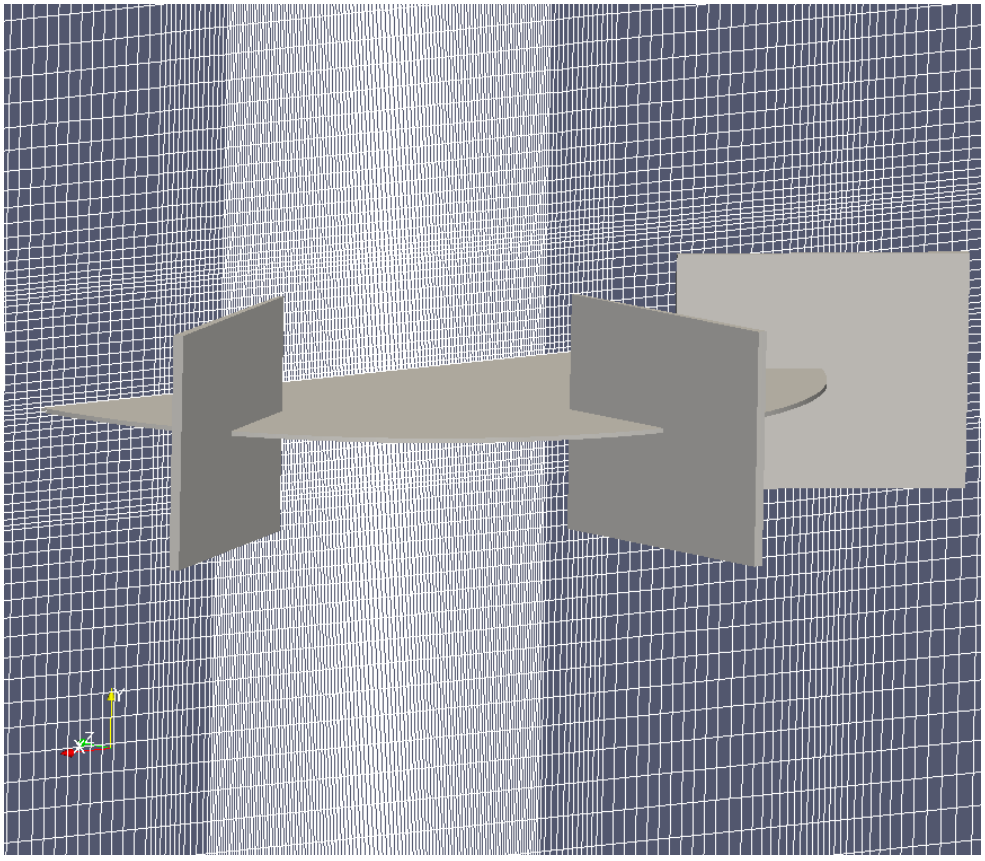


Picture 4 Mesh

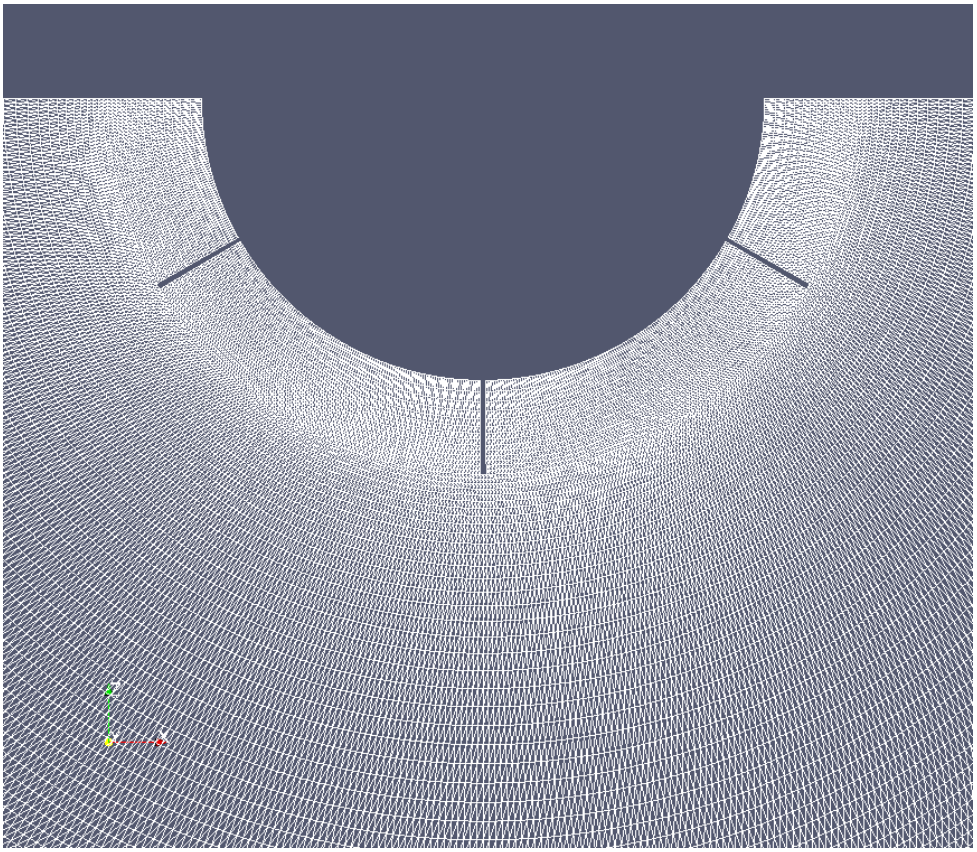
When building the mesh, it is preferable to set planes where the tank was divided as *Periodic* since any change done on one side will automatically be reflected on the other side. That way those faces will remain completely the same, which is important for setting up cyclic boundary conditions.



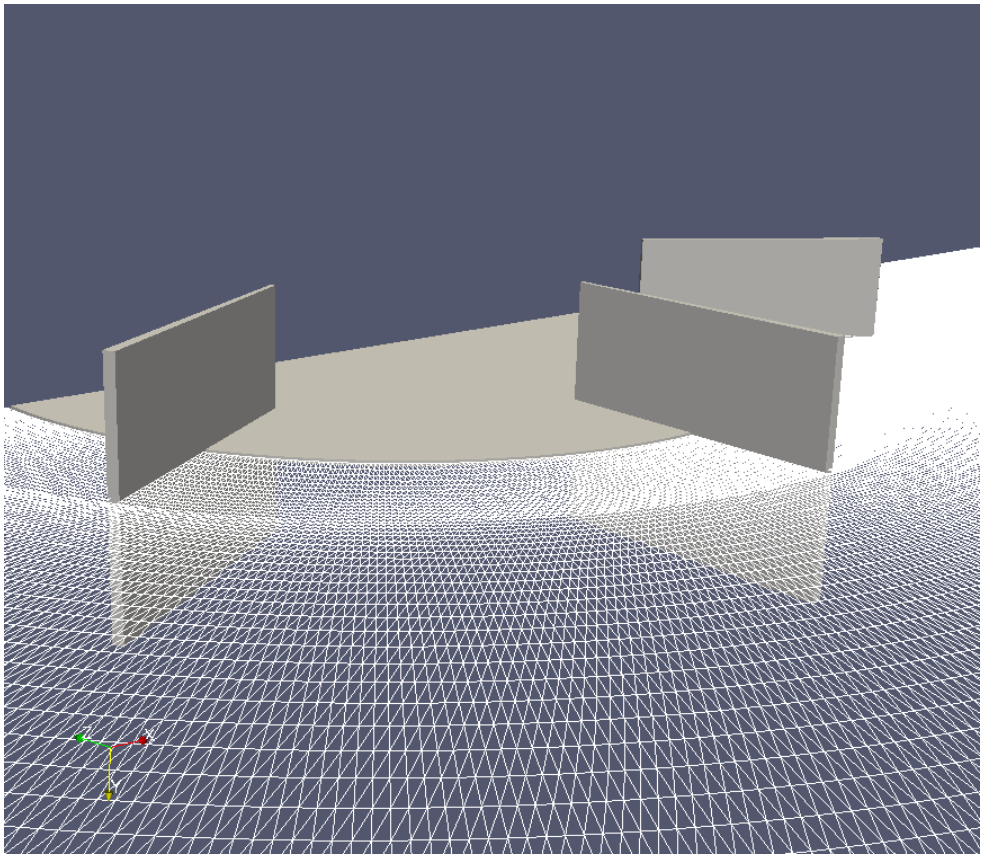
Picture 5 Impeller surface mesh



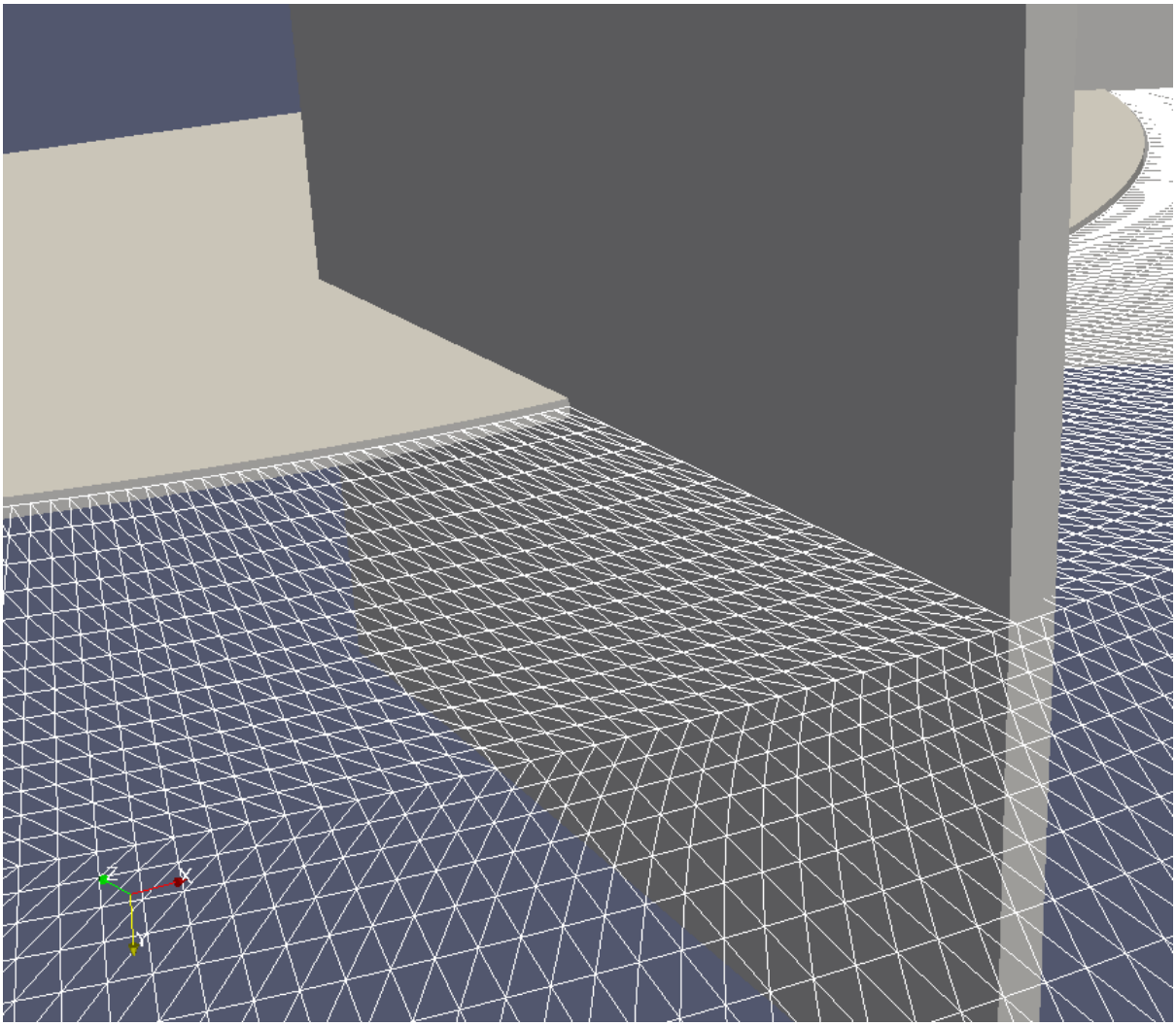
Picture 6 Impeller



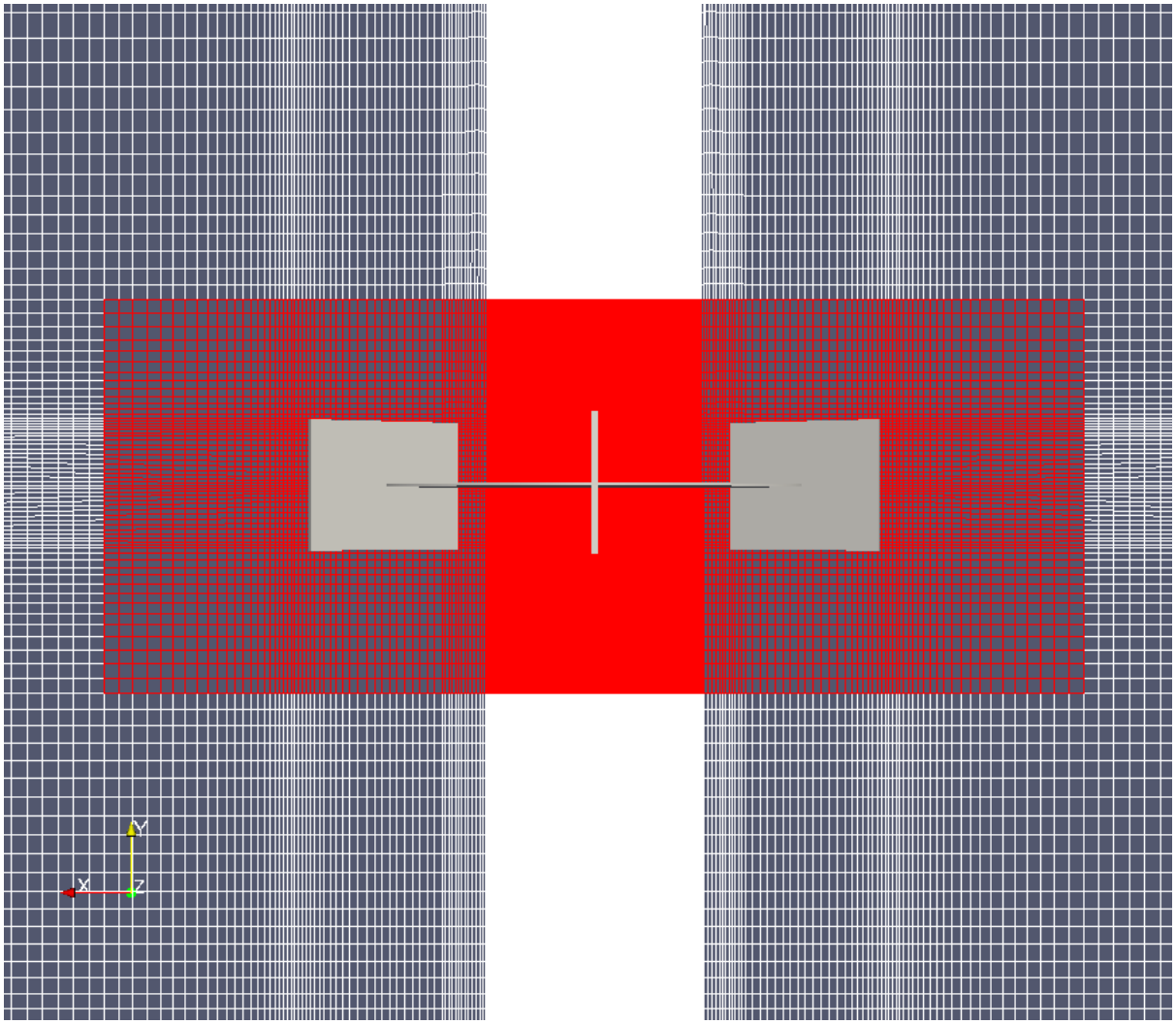
Picture 7 Mesh distribution around the impeller



Picture 8 Mesh surrounding the impeller



Picture 9 Mesh detail next to the impeller



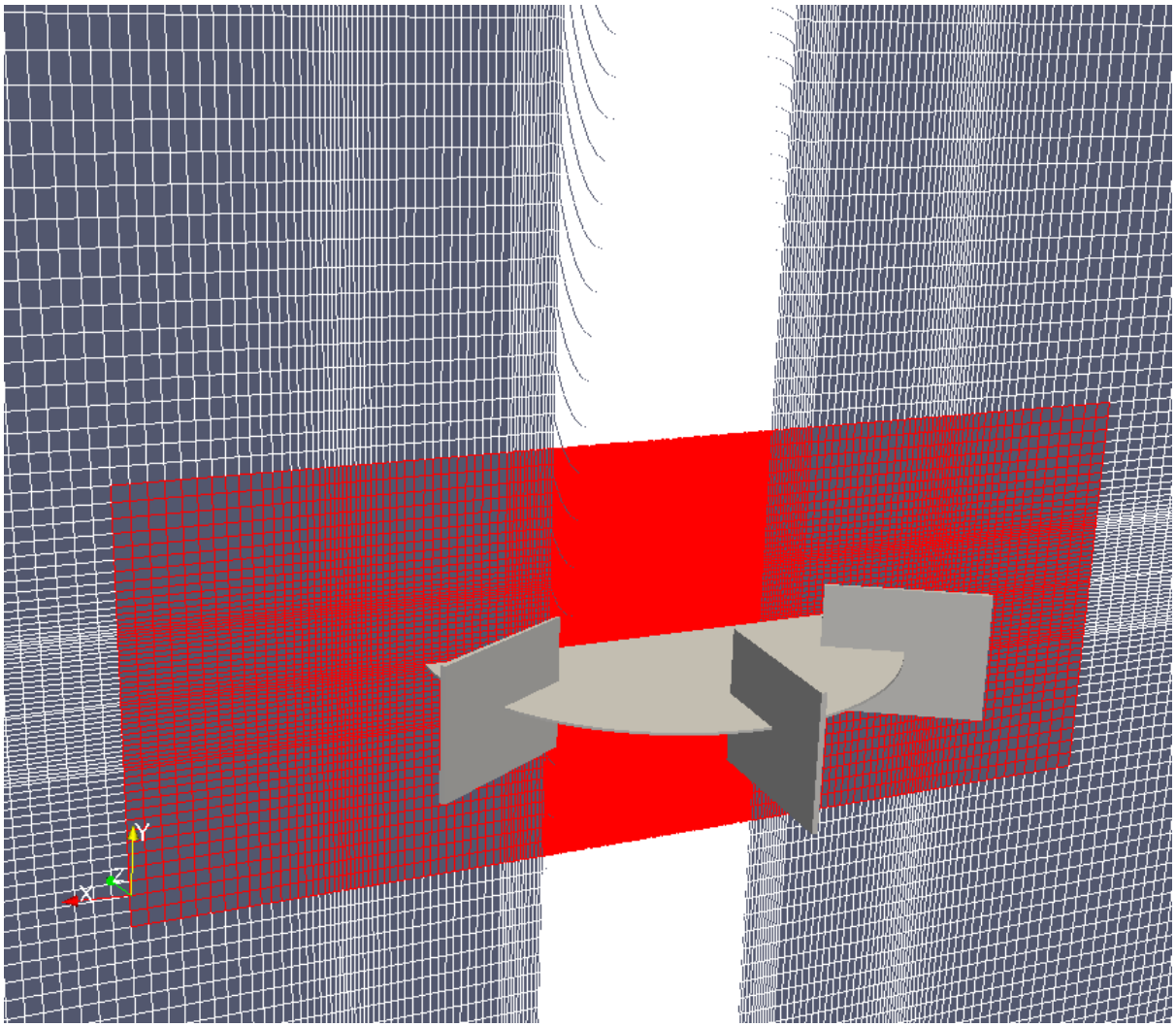
Picture 10 Periodic boundaries

Unfortunately, I was unable to make the simulation run with *Cyclic* boundary condition in OpenFOAM. The solver repeatedly complained about the interfaces not being completely identical. To test that claim, mesh was exported and imported to several different CFD tools where it was confirmed that the interfaces are identical. Not to waste too much time, those interfaces were set as *cyclicAMI* (Cyclic Arbitrary Mesh Interface). *CyclicAMI* is usually used when mesh on periodic boundaries is not completely symmetric. Therefore, a certain type of interpolation is used between mismatched faces. The general assumption, when I decided to use this approach, was that since the interfaces are actually identical in Pointwise, *cyclicAMI* won't use interpolation. If it does, it won't have a significant impact. But if some of the faces become mismatched when exporting the mesh from Pointwise, *cyclicAMI* will take care of the error. For more information: http://www.openfoam.org/version2.3.0/ami.php

Pointwise has a very specific way of numbering points, which in some cases conflicts with OpenFOAM and makes the pressure equation diverge. Therefore, it is advised to use *renumberMesh* function to renumber mesh to OpenFOAM's liking. It takes a couple of minutes to run it on very large meshes, but it saves up a lot of time on pressure equation during run time.

MRF zones

After finishing the mesh, it is advisable to set MRF zones in Pointwise. It is a useful new utility in Pointwise which allows the user to directly specify zones for OpenFOAM and save time. When doing a stirred tank simulation, it is advisable to select as small as possible box around the impeller as the rotational zone since the size of rotational zone directly affects computational time. One should be careful to select the boundaries of rotational zone far enough from the impeller, not to affect the discharge stream too much. Modelling the impeller shaft doesn't mean that it should be included in the MRF zone. As previously mentioned, the rotational zone box should be as small as possible without affecting the flow and the part of the shaft that's left in the stationary zone can be given a rotating boundary condition. One should be careful to make rotational zone and the shaft rotate in the same direction. In Picture 11, red colour represents cylindrical rotational zone *rotor* while stationary zone *stator* is coloured white.



Picture 11 MRF zones/periodic bundaries

Initial and boundary conditions

Regarding the initial and boundary conditions, all of the cylinder walls, baffles and the impeller are set as *wall* boundary condition while the faces which represent where the cylinder was divided are set as c*yclicAMI*. Wherever there are walls, there are wall functions for *k, epsilon, nut.* For *p* at all wall boundaries *zeroGradient* is set, while for *U* velocity at wall boundaries is set to *fixedValue* with the value of 0. At *cyclicAMI* boundaries, all initial conditions are set as *cyclicAMI*. All faces that are set as *cyclicAMI* have additional definitions which define pairs (*neighbourPatch*), state their tolerance and the way data is supposed to be mapped between them (*transform, rotationAxis, rotationCentre*). Therefore, to match everything, *cellZones* file has two separate entities: zone one is *rotor* and zone two is *stator*.

MRFZones file is used to define the number of rotating zones and the patches that connect stationary and seemingly rotational part of the mesh. The origin was set at (0,0,0) with a *y* axis of rotation (0,1,0) and angular velocity of 200 [1/s]. Therefore, rotational zone *rotor* will rotate with angular velocity of 200 [1/s] around the *y* axis that passes through point (0,0,0).

Simulation settings

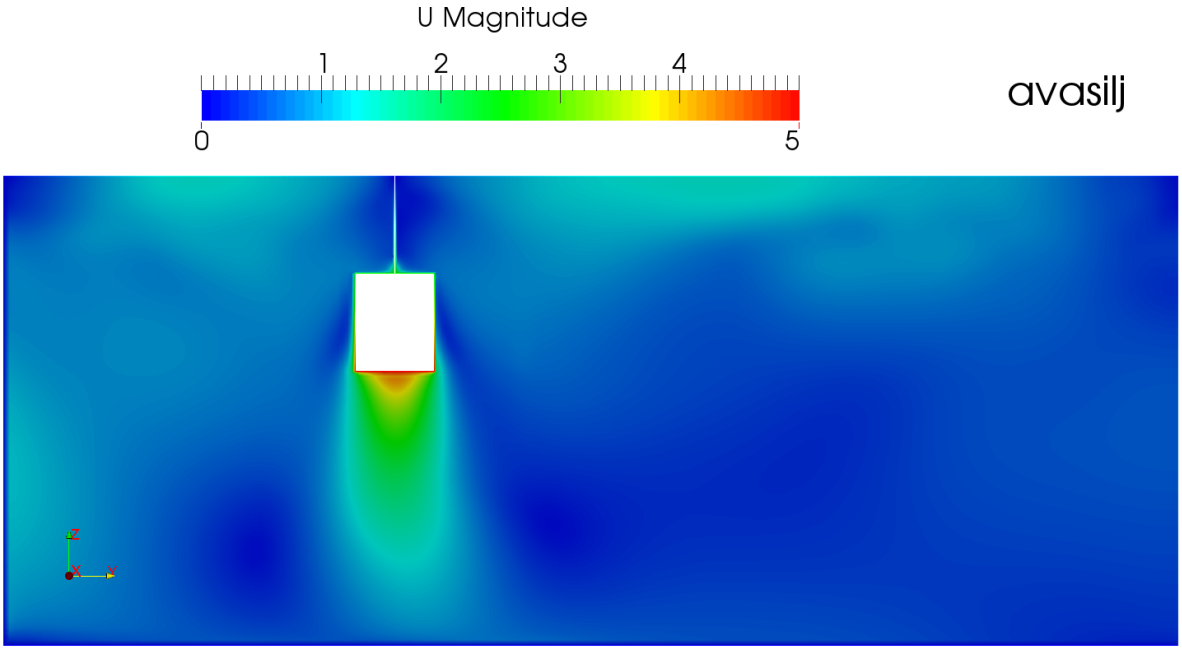
The simulation was set as a steady state one with *deltaT* equal to one. With that, time step becomes an iteration counter. The results are written every 50 time steps to save on disk space and overall simulation speed.

For the pressure equation *PCG* solver is used, with *GAMG* as a preconditioner. Number of *preSweeps* is set to zero and number of *postSweeps* to two to reduce computational time. Since the number of *postSweeps* has a bigger impact on error reduction and *GAMG* convergence, it is given advantage to over number of *preSweeps*. For all other equations, *PBiCG* solver is used.

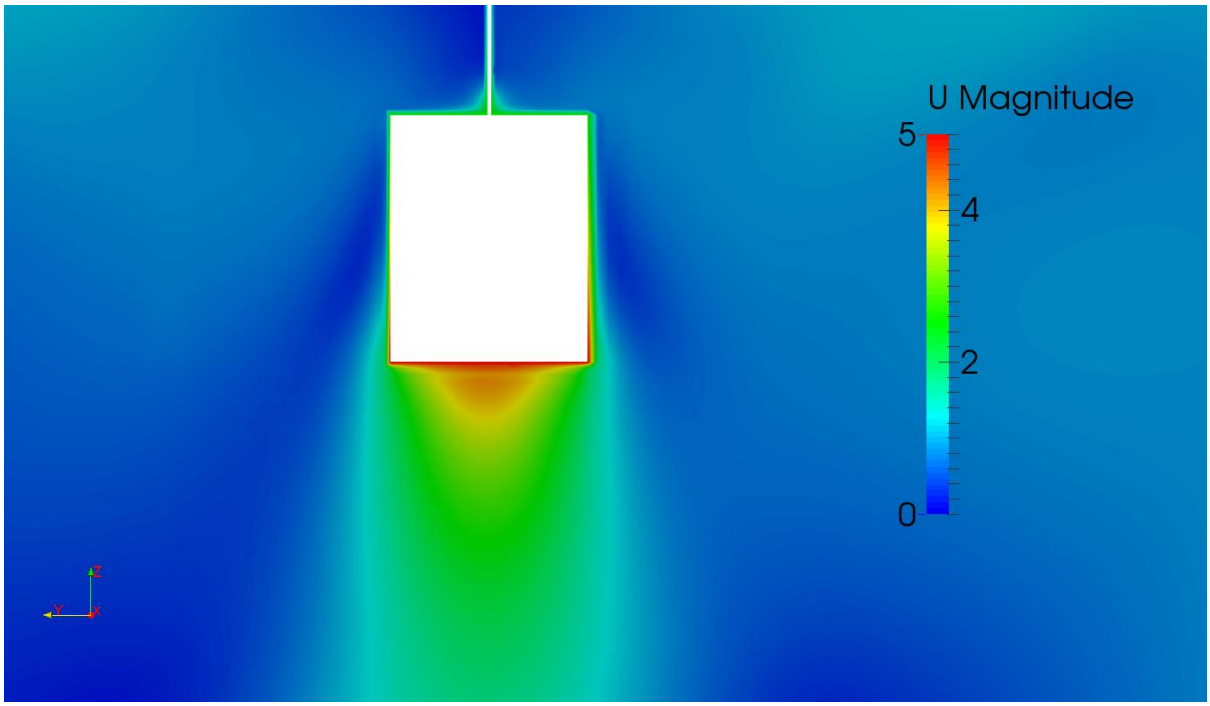
The simulation was ran for 4520 time steps and the results are provided in the following section.

Results

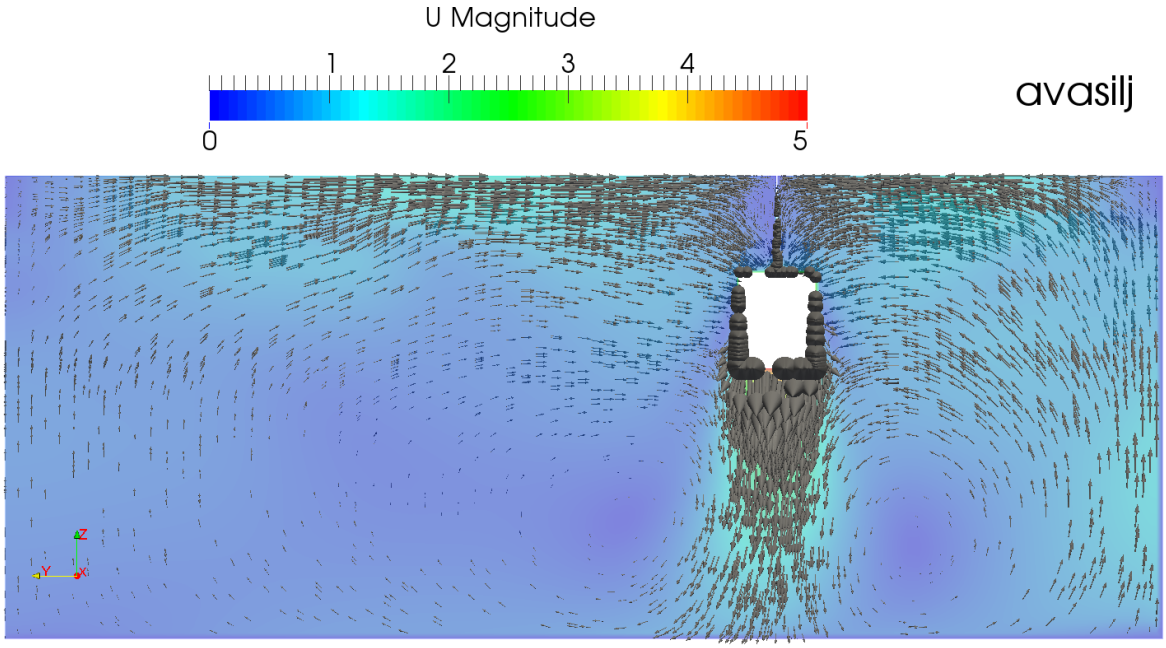
An accurate CFD model should be able to predict general features of a flow field in a stirred tank as well as the subtle phenomena. As can be seen in Picture 12 and Picture 13, the stream ejected from the impeller blades flowing toward the tank wall behaves like a jet. Region of the discharge stream has the largest velocities reaching up to blade tip velocity of 5 [m/s] right next to it. Movement of discharge stream flow in the radial direction creates two large swirls, one in the region above the impeller and the other in the region below, as can be seen in the Picture 14, due to the division of stream in the near wall region.



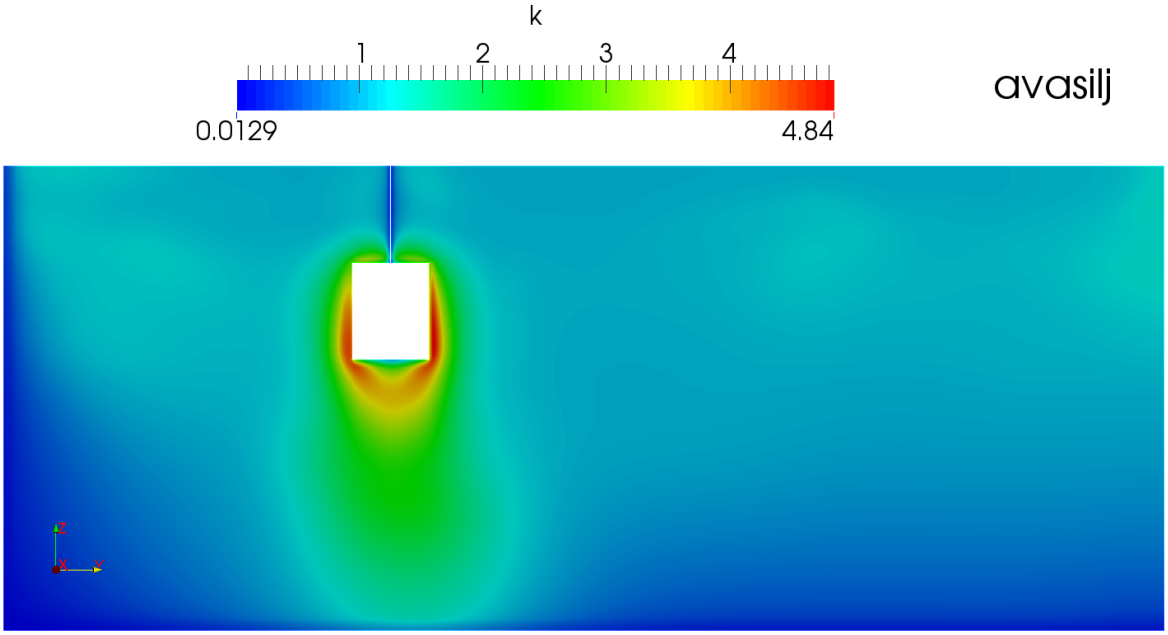
Picture 12 Velocity magnitude field



Picture 13 Velocity magnitude distribution near the blade



Picture 14 Velocity magnitude glyphs



Picture 15 Turbulent kinetic energy distribution

Normalized radial and tangential velocities were compared to the experimental results (Deglon, Meyer, 2006). Unfortunately, radial location where the data was collected was not disclosed in the paper nor was I able to contact the authors. Therefore, simulation results were taken at $r = 2,66\,\text{cm}$ and compared with the experimental curve. This approximation was taken after reading various papers and checking their radial sampling points.

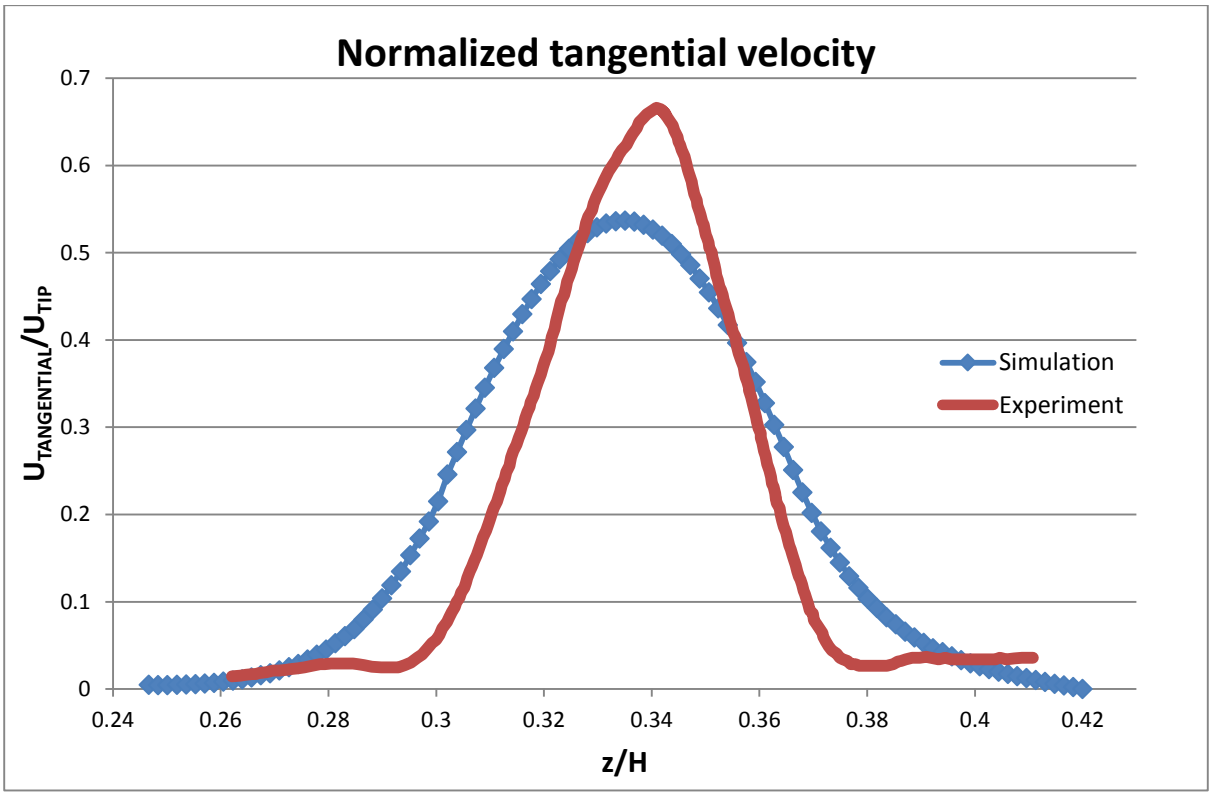
Experimental curve was provided in the paper without any points or error bars and therefore taken as such.

Both radial and tangential velocities are divided by the blade tip velocity, which is calculated as

$$v_{tip} = \omega \cdot r = 200 \frac{1}{\mathrm{s}} \cdot 0,025\,\mathrm{m} = 5 \frac{\mathrm{m}}{\mathrm{s}}$$

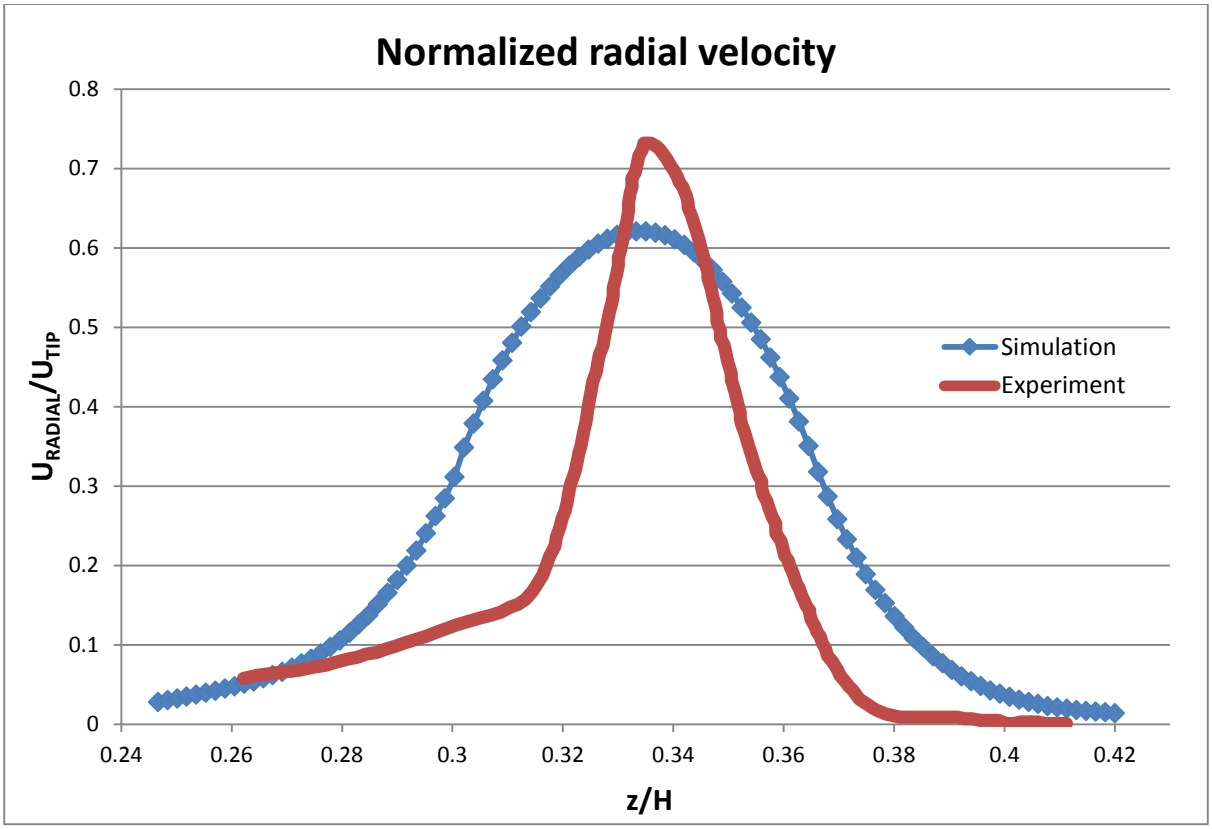
while height is divided by the tank height.

Simulation obtained tangential velocity profile is wider than the experimentaly obtained one and with a lower peak magnitude. It has a more symmetric shape which would suggest weaker influence from the bottom swirl while it can be seen that the experimental curve favours upper part of the discharge stream since local lower velocities from the top swirl are having weaker influence compared to the larger local bottom swirl velocities. We can conclude that the local momentum transfer in the *z* direction is larger in the simulation and therefore, we are experiencing earlier flattening of the velocity profile compared to the experimental data, as shown in Picture 16.



Picture 16 Normalized tangential velocity

Simillar effects can be seen in the Picture 17, where numerical and experimental radial velocity profiles are compared. It is noticable how the experimental velocity profile is not symmetric about $z = 0$ plane, but shifts upwards as the wall is approached (Wu and Patterson 1988). It is due to imperfections in positioning the impeller and the top of the tank is a free surface.



Picture 17 Normalized radial velocity

Conclusion

Usage of OpenFOAM simulation software was tested against multiphase flow and MRF test case. Effects of grid resolution, discretization schemes and turbulence models were investigated, but need to be validated against experimental results and compared to other available simulation software.

For Multiphase flow test case, basic equations are derived and explained which give insight into physical phenomena occurring within the simulation. Furthermore, case building process is established and explained to help speed up any future work within the field. In the near future, I would recommend finding an experimental paper with full experimental setup data disclosed, to be able to validate OpenFOAM simulation results.

Multiple Reference Frame basic concepts were introduced and results of an extensive literature review are given. With those findings in mind, Pointwise mesh was created and OpenFOAM framework established. The simulation was successfully performed, with 4520 iterations and results were compared to experimental data (Wu and Patterson 1988). The accuracy of CFD model was investigated using flow field predictions and normalized velocity components in redial and tangential directions. From these results one can conclude that flow field is well predicted and can be established using even relatively coarse grids. On the other hand, subtle phenomena like trailing vortices require very fine grids. Normalized velocities capture the trend but fail to fit the jet width. Such smearing of the velocity profile needs to be further investigated. It is advisable to compare turbulent kinetic energy results with experimental data since the accuracy of turbulent kinetic energy predictions strongly depends on discretization schemes and grid resolution. In addition to that, turbulent kinetic energy overprediction or underprediction when using $k-\varepsilon$ turbulence model comes from the deficiencies in the model rather than from numerical issues related to grid discretization or numerical schemes.

Appendix

Boundary file

```
9
(
    baffles
    {
        type            wall;
        nFaces          7620;
        startFace       18421811;
    }
    bottom
    {
        type            wall;
        nFaces          48904;
        startFace       18429431;
    }
    cyclic1r
    {
        type            cyclicAMI;
        nFaces          8677;
        startFace       18478335;
        matchTolerance  0.0001;
        neighbourPatch  cyclic2r;
        transform       rotational;
        rotationAxis    (0 1 0);
        rotationCentre  (0 0 0);
    }
    cyclic1s
    {
        type            cyclicAMI;
        nFaces          12418;
        startFace       18487012;
        matchTolerance  0.0001;
        neighbourPatch  cyclic2s;
        transform       rotational;
        rotationAxis    (0 1 0);
        rotationCentre  (0 0 0);
    }
    cyclic2r
    {
        type            cyclicAMI;
        nFaces          8677;
        startFace       18499430;
        matchTolerance  0.0001;
        neighbourPatch  cyclic1r;
        transform       rotational;
        rotationAxis    (0 1 0);
        rotationCentre  (0 0 0);
    }
    cyclic2s
    {
        type            cyclicAMI;
        nFaces          12418;
        startFace       18508107;
        matchTolerance  0.0001;
        neighbourPatch  cyclic1s;
```

```
        transform       rotational;
        rotationAxis    (0 1 0);
        rotationCentre  (0 0 0);
    }
    cylinder
    {
        type            wall;
        nFaces          45212;
        startFace       18520525;
    }
    impeller
    {
        type            wall;
        nFaces          54660;
        startFace       18565737;
    }
    top
    {
        type            wall;
        nFaces          48904;
        startFace       18620397;
    }
)
```

MRF Zones

```
1
(
    rotor
    {
        // Fixed patches (by default they 'move' with the MRF zone)
        nonRotatingPatches (cyclic1r cyclic2r);

        origin    origin [0 1 0 0 0 0 0]  (0 0 0);
        axis      axis   [0 0 0 0 0 0 0]  (0 1 0);
        omega     omega  [0 0 -1 0 0 0 0] 200;
    }
)
```

controlDict

```
application     MRFSimpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         10000;

deltaT          1;
```

```
writeControl    timeStep;

writeInterval   50;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


functions
{
    forces
    {
        type forces;
        functionObjectLibs ("libforces.so");
        outputControl timeStep;
        outputInterval 5;
        patches (impeller baffles);
        rhoName rhoInf;
        rhoInf 1000;      // Reference density, fluid
        CofR (0 0 0);  // Origin for moment calculations
    }
}
```

fvSolution

```
solvers
{
    p
    {
        solver          PCG;
    preconditioner
      {
        preconditioner  GAMG;
        smoother        DICGaussSeidel;
        nPreSweeps  0;
        nPostSweeps 2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 4;
        agglomerator    faceAreaPair;
        mergeLevels     1;
      }
        tolerance       1e-08;
    relTol          0.05;
    }

    U
    {
```

```
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.0;
    }

    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.0;
    }

    epsilon
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 2;
    pRefCell        0;
    pRefValue       0;
}

relaxationFactors
{
    fields
    {
        p               0.15;
    }
    equations
    {
        U               0.3;
        k               0.3;
        epsilon         0.3;
    }
}
```

fvSchemes

```
ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         cellMDLimited leastSquares 1;
}
```

```
divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}
```