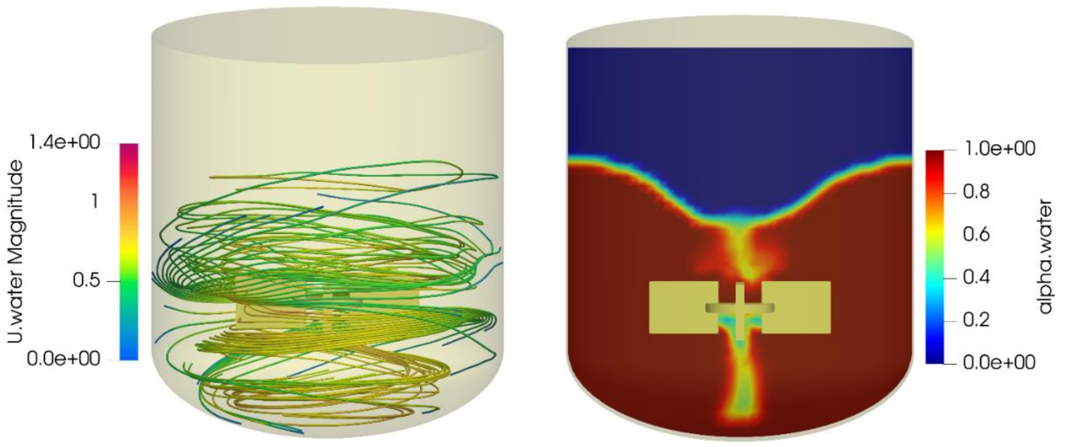
Mastering twoPhaseEulerFoam

Four: Two-Phase Stirred Tank Reactor



Simulating a two phase stirred tank reactor using OpenFOAM®

Compatible with	OpenFOAM® 7 OpenFOAM® v1912

Author

Hamidreza Norouzi

Amirkabir University of Technology

Center of Engineering and Multiscale Modeling of Fluid Flow

Document history

Revision	Description	Date
Rev1	The tutorial was reviewed and the simulation was performed using OpenFOAM® v1912	May 1, 2020
Rev0	First draft was prepared and run with OpenFOAM ® 7	April 25, 2020

Table of Contents

Prerequisites

- Basic simulation setup of a bubble column using twoPhaseEulerFoam solver.
- You need to know the basics of snappyHexMesh utility for creating the mesh.
- Most of the model setup settings and physical properties are similar to what is simulated in tutorial "Mastering twoPhaseEulerFoam, Three: Bubble Column," so you are referred to this document several times in this tutorial.

How to get simulation setup files?

You can get the simulation setup files from Website: setup files (a compressed file) are uploaded on www.cemf.ir alongside this tutorial file. You can find these files there.

1. Brief Description of twoPhaseEulerFoam

twoPhaseEulerFoam is a solver for a system of 2 non-reacting compressible fluid phases. One phase in this system is always dispersed. So it is a good candidate for simulating bubble columns in gas/liquid systems or fluidized beds and spouted beds in gas/solid systems. The solver also solves the energy equation (enthalpy or internal energy) for both phases and couple them by the one-film resistance heat transfer model.

In the case of gas-liquid systems, laminar and turbulence models (RAS and LES) models can be selected for both phases. In this case of gas-solid or liquid solid systems, these models can only be applied for the fluid phase. The solid phase momentum equation incorporates a model for the stress tensor. Two main approaches are possible: inviscid solid phase with a pressure model and a solid phase with stress tensor that is obtained by KTGF theory and frictional models.

Various sub-models for interphase coupling are also provided that make it possible to select a wide range of physical models for the system. It is also possible to extend the standard solver to include new sub-models in the simulation.

2. Problem definition

The stirred tank reactor in this tutorial has a cylindrical shell with inner diameter of 0.4 m and height of 0.35 m. The bottom cap is a 2:1 ellipse (radius1 is 0.2 m and radius2 is 0.1 m) and top of this cylinder is open to atmosphere. A four-bladed impeller is located in the middle of the reactor to provide the mixing force. This impeller rotates with at 120 RPM. The surface geometries of this mixer and the reactor vessel are provided in the form of stl files. The vessel is initially filled up to 0.15 m with water at 300 K. Air enters the reactor through a sparger located beneath the impeller at the bottom of the reactor.

In this tutorial you will learn how to simulate such a reactor system using multiple reference frame (MFR) method. Other properties and operating conditions are given in Table 1. In this tutorial it is tried to keep important details (not all details) of the such systems while keeping the whole geometry simple enough to obtain reasonable execution time for the final case.

Table 1: Physical properties and operating conditions

Water viscosity [Pa.s]	3.645×10^{-4}	Air viscosity [Pa.s]	1.84×10^{-8}
Water density [kg/m^3]	~1000	Air density	Ideal gas
Water heat capacity [J/kg/K]	4195	Air heat Capacity [J/kg/K]	1007
Water Prandtl number	2.289	Air Prandtl number	0.7
Water temperature [K]	300	Inlet air temperature [K]	300
Air-water surface tension [N/m]	0.072	Superficial gas velocity [m/s]	0.05
Rotation speed [RPM]	120	Bubble diameter [m]	0.003

3. MRF

Rotating walls can be simulated using multiple reference frame (MRF). In this method, the whole region is sub-divided into one or more rotating zones and one stationary zone. This approach considers the rotation of wall (impeller) without moving the mesh by making necessary changes in the momentum equation. The velocity field with respect to inertial frame (stationary zone) is $\vec{u}$ and the relative velocity in the rotating zone is $\vec{u}_r$. The absolute velocity and relative velocity in each zone are related as:

$$\vec{u}_r = \vec{u} - (\vec{\Omega} \times \vec{r})$$

where $\vec{\Omega}$ is the rotation speed and $\vec{r}$ is the vector pointing from rotation center to the point of evaluation of relative velocity.

The momentum equation is solved for absolute velocity with convective fluxes that accounts for the rotation of the rotating zones (by making fluxed relative to stationary frame in all zones). The left-hand side of momentum equation (for a single phase flow) becomes:

$$\frac{\partial}{\partial x}(\rho\vec{u}) + \nabla.(\rho\vec{u}_r\vec{u}) + \rho(\vec{\Omega} \times \vec{u}) = RHS$$

where $\rho(\vec{\Omega} \times \vec{u})$ is the Coriolis force. The face fluxed are made relative by the following equation:

$$phi_r = phi - \rho(\vec{\Omega} \times \vec{r})S$$

where the relative flux of cells and relative velocity of the cells located in the stationary zone ($\vec{\Omega}$=0) are reduce to absolute flux and velocity.

4. Simulation setup

Physical properties of both phases and phase coupling models are first specified and then you will learn how to generate the mesh, to define MRF in the simulation, and to define an injection region as the sparger.

4.1. Physical properties of phases

More description on the physical properties can be found in the previous tutorials of this series. The physical properties of air are specified in constant/thermophysicalProperties.air.

constant/thermophysicalProperties.air

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        molWeight   28.9;
    }
    thermodynamics
    {
        Cp          1007;
        Hf          0;
    }
    transport
    {
        mu          1.84e-05;
        Pr          0.7;
    }
}
```

And the properties of water phase are defined in constant/thermophysicalProperties.water.

constant/thermophysicalProperties.water

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          eConst;
    equationOfState perfectFluid;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        molWeight   18;
    }
    equationOfState
    {
        R           3000;
        rho0        1027;
    }
    thermodynamics
    {
        Cv          4195;
        Hf          0;
    }
    transport
    {
        mu          3.645e-4;
        Pr          2.289;
    }
}
```

4.2. Defining phases and interphase coupling parameters

For more description on the interphase coupling models, you are referred to previous tutorials of this series. Here, only the content of file constant/phaseProperties is given as a reference. Two phases are `air` and `water` in this system.

constant/phaseProperties

```
phases (air water);

air
{
    diameterModel   isothermal;
    isothermalCoeffs
    {
        d0              3e-3;
        p0              101325;
    }
    residualAlpha   1e-6;
}

water
{
    diameterModel   constant;
    constantCoeffs
    {
        d               1e-4;
    }

    residualAlpha   1e-6;
}

blending
{
    default
    {
        type            linear;
        maxFullyDispersedAlpha.air 0.3;
        maxPartlyDispersedAlpha.air 0.5;
        maxFullyDispersedAlpha.water 0.3;
        maxPartlyDispersedAlpha.water 0.5;
    }

    heatTransfer
    {
        type            linear;
        maxFullyDispersedAlpha.air 0.3;
        maxPartlyDispersedAlpha.air 0.7;
        maxFullyDispersedAlpha.water 0.3;
        maxPartlyDispersedAlpha.water 0.7;
    }
}

sigma
(
    (air and water)     0.072
);

aspectRatio
(
    (air in water)
    {
        type            Wellek;
    }

    (water in air)
    {
        type            constant;
        E0              1.0;
    }
);

drag
(
    (air in water)
    {
        type            IshiiZuber;
        residualRe      1e-3;
        swarmCorrection
        {
            type            none;
            residualAlpha   Tomiyama;
            l               1.75;
        }
    }

    (water in air)
    {
        type            SchillerNaumann;
        residualRe      1e-3;
        swarmCorrection
        {
            type        none;
        }
    }

    (air and water)
    {
        type            segregated;
        m               0.5;
        n               8;
        swarmCorrection
        {
            type        none;
        }
    }
);

virtualMass
(
    (air in water)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }

    (water in air)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }
);

heatTransfer
(
    (air in water)
    {
        type            RanzMarshall;
        residualAlpha   1e-4;
    }

    (water in air)
    {
        type            RanzMarshall;
        residualAlpha   1e-4;
    }
);


lift
(
    (air in water)
    {
        type constantCoefficient;
        Cl   0.25;
    }

    (water in air)
    {
        type none;
    }
);

wallLubrication
(
);

turbulentDispersion
(
);
```

4.3. Turbulence properties of phases

Laminar model is used for both phases for reducing the execution time. The physical model can be further improved by using proper turbulence models, knowing that the rotating blades create a turbulent flow in the vessel.

constant/turbulenceProperties.air

```
simulationType laminar;
```

constant/turbulenceProperties.water

```
simulationType  laminar;
```

4.4. Gravitational acceleration

The gravitational acceleration can be defined in constant/g file as follows:

constant/g

```
dimensions        [0 1 -2 0 0 0 0];
value             (0 -9.81 0);
```

4.5. Generating geometry and mesh

Mesh generation consists of two main steps:

- generating a background mesh using `blockMesh` utility; and
- refining, cutting and conforming the mesh on the triangulated surfaces that are provided to the program: here impeller blade and vessel surfaces.

4.5.1. blockMesh for creating the background mesh

The background mesh encompasses the whole surface of the vessel and mixer blade. The encompassing cuboid is meshed with hex cells with size of 0.01 m. It is recommended to produce background hex mesh whose edges have similar sizes to get the best result from `snappyHexMesh`. Two boundary patches are defined: `allBoundary` and `outlet`. The first patch will be automatically deleted after running `snappyHexMesh` and `outlet` patch will remain as the outlet of the vessel. Execute the following command to generate the background mesh:

```
> blockMesh
```

system/blockMeshDict

```
convertToMeters 1;

vertices
(
    ( -0.21 -0.16 -0.21)
    (  0.21 -0.16 -0.21)
    (  0.21  0.3  -0.21)
    ( -0.21  0.3  -0.21)
    ( -0.21 -0.16  0.21)
    (  0.21 -0.16  0.21)
    (  0.21  0.3   0.21)
    ( -0.21  0.3   0.21)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (42 46 42) simpleGrading (1 1 1)
);

edges
(
);
```

```
boundary
(
    allBoundary
    {
        type patch;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
            (0 3 2 1)
            (4 5 6 7)
        );
    }

    outlet
    {
        type patch;
        faces
        (
           (3 7 6 2)
        );
    }
);
```

4.5.2. snappyHexMesh for creating the final mesh

The surface geometry files should be placed in constant/triSurface folder. You can find the surface geometry of impeller and vessel in this folder. These surfaces can be visualized using Paraview®:

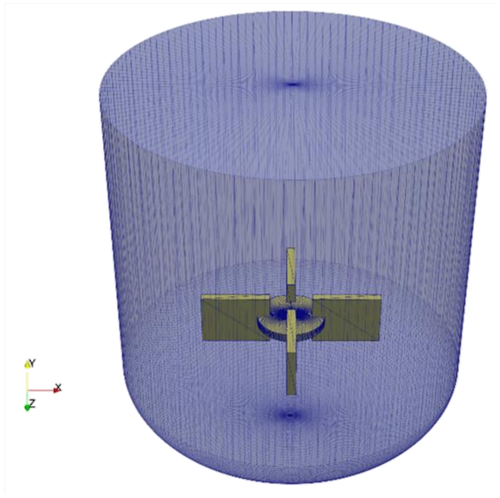


Figure 1: Visualization of impeller and vessel surface from stl files

Note

The axial shaft of the impeller system is not considered here to resolve the impeller surface with a less number of cells.

Splitting of cells starts with the edge features (eMesh file) that the user supplies in snappyHexMeshDict file. `sufraceFeatureExtractDict` utility is used to extract edge features from stl surfaces. The corresponding dictionary is as follows:

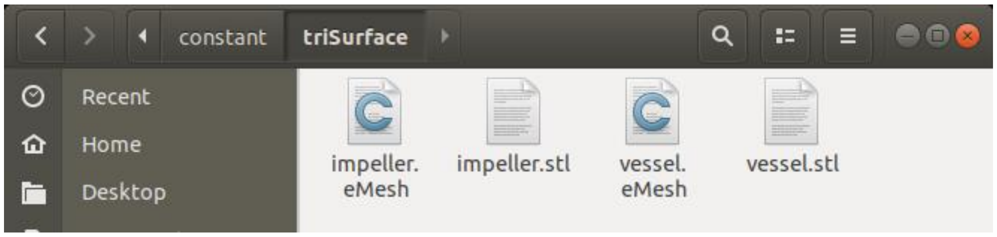
system/surfaceFeatrueExtractDict

```
impeller.stl
{
    extractionMethod    extractFromSurface;
    extractFromSurfaceCoeffs
    {
        includedAngle       150;
    }
}
vessel.stl
{
    extractionMethod    extractFromSurface;
    extractFromSurfaceCoeffs
    {
        includedAngle       150;
    }
}
```

Two separate files, impeller.stl and vessel.stl, are explored to find surface edges. Those edges whose angle are less than `includedAngle`, will be considered as the edges that will be refined in the mesh generation step. The result of this step is saved in an eMesh file. Execute the following command to create eMesh files for your surfaces:

```
> sufraceFeatureExtract
```

The eMesh files will be saved in constant/triSurface folder.



The settings for snappyHexMesh utility can be found in system/snappyHexMeshDict file. The first part of this file is as follows:

```
castellatedMesh true;
snap            true;
addLayers       false;

geometry
{
    impeller
    {
        type triSurfaceMesh;
        file "impeller.stl";
    }
    vessel
    {
        type triSurfaceMesh;
        file "vessel.stl";
    }
    //- Used to define MRF zone and to refine mesh round the impeller
    MRF
    {
        type searchableCylinder;
        point1  (0.0 -0.04 0.0);
        point2  (0.0  0.04 0.0);
        radius 0.11;
    }
}
```

The first three lines specify which step should be performed. Steps castellation and snapping will be performed and layering will be bypassed. In `geometry` sub-dictionray, all the geometry entities that will be used in the meshing process are defined. The first two, `impeller` and `vessel`, are created from stl files. The last one, `MRF`, is created using the built-in geometry entity, `searchableCylinder`. It only requires you to define the first and last point of the cylinder axis and its radius. This cylinder will be later utilized for defining the MRF zone.

Sub-dictionary `castellatedMeshControls` defines all the features (edges), surfaces and regions refinement parameters. In sub-dictionary `features`, extracted surface features (edges) are addressed to be split up to a defined level. Here, cells at `vessel.eMesh` surface features are split at level 0 (cells as large as background mesh) and cells at `impeller.eMesh` surface feature are split at level 1. Entry `resolveFeatureAngle` controls the local curvature of the surface features. If you use high values for `resolveFeatureAngle`, surface features (edges) with high curvature will not be captured in the final mesh.

```
castellatedMeshControls
{

    maxLocalCells 100000;

    maxGlobalCells 2000000;

    minRefinementCells 0;

    nCellsBetweenLevels 2;

    maxLoadUnbalance 0.10;

    features
    (
        {
            file "vessel.eMesh";
            level 0;
        }

        {
            file "impeller.eMesh";
            level 1;
        }

    );

    resolveFeatureAngle 30;

    refinementSurfaces
    {
        MRF
        {
            level (1 1);
            cellZone cellMRFzone;
            faceZone faceMRFzone;
            cellZoneInside inside;
        }

        impeller
        {
            level (1 1);
        }

        vessel
        {
            level (0 0);
        }
    }

    refinementRegions
    {
        MRF
        {
            mode inside;
            levels ((1E15 1));
        }

    }

    locationInMesh (0.0 0.1 0.0); // Inside point

    allowFreeStandingZoneFaces true;
}
```

Sub-dictionary `refinementSurfaces` defines the surfaces to be refined. Surface `MRF` is refined up to level (1 1). The first number shows the global refinement level on the surface and the second number shows the level of extra (local) refinement when local curvature of the surface exceeds `resolveFeatureAngle`. For `MRF` surface, a part of mesh (cells and faces) located inside the `MRF` surface is marked as `cellMRFzone` cell zone and `faceMRFzone` face zone. The cell zone `cellMRFzone` will be used later for specifying MRF region. Surfaces `impeller` and `vessel` are refined up to levels (1 1) and (0 0), respectively.

In sub-dictionary `refinementRegions`, the cells inside the `MRF` surface are selected to be refined at level 1. This part refines cells that are far from the `MRF` surface and surface features of `impeller` (it refines cells in the bulk of the region).

The typical settings of `snapControls` sub-dictionary and `meshQualityControlDict` are used in this tutorial. So the details are not presented here. Execute the following command to generate your final mesh:

```
> snappyHexMesh -overwrite
```

The final mesh and the cell zone `cellMRFZone` are illustrated in the figure below.

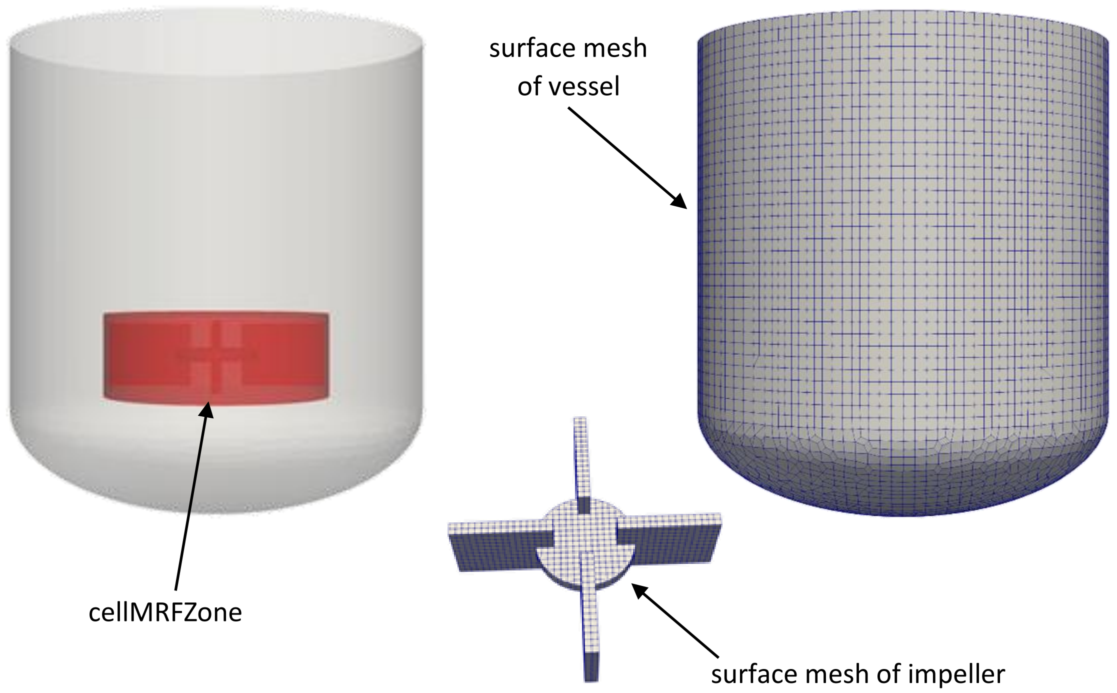


Figure 2: (left) vessel surface and MRF zone and (right) final mesh of the vessel and impeller.

4.6. Defining rotating zone

In file constant/MRFProperties, the settings for MRF region can be found. `cellMRFzone` is region for which the rotation is applied. The origin of rotation is (0 0 0) and its direction is (0 1 0), i.e. rotation around the y-axis. The rotation speed is 12.56 rad/s which is equivalent to 120 RPM.

constant/MRFProperties

```
MRFImpeller
{
    cellZone    cellMRFzone;
    active      yes;

    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches ();

    origin    (0.0 0.0 0.0);
    axis      (0 1 0);
    omega     constant 12.56; // ~120 RPM
}
```

4.7. Defining sparger

You can use fvOptions to simulate a sparger[1]. fvOptions is invoked in each conservation PDE that is solved in the solver. It is located at the right-hand side of equation to act as a source term. Each time that one PDE is solved, the solver looks into the system/fvOptions file (if such file is supplied) to evaluate the source term for the equation. In general, a conservation equation can be formulated as follows:

$$\frac{D}{Dt}\varphi = S(\varphi)$$

where φ is the dependent field variable for which the conservation equation is solved and $S(\varphi)$ is the source term that is decomposed (linearized) into two explicit and implicit parts:

$$S(\varphi) = S_u + S_p\varphi$$

where S_u is the explicit source term component and S_p is the implicit source term component.

The content of the file system/fvOptions is as follows:

system/fvOptions

```
injector1
{
    timeStart        0.0;
    duration         30;
    selectionMode    cellSet;
    cellSet          injection1;
}

options
{
    massSource1
    {
        type             scalarSemiImplicitSource;

        $injector1;

        volumeMode       absolute;
        injectionRateSuSp
        {
            thermo:rho.air      (3.0e-4 0); // kg/s
        }
    }
```

[1] Other applications of fvOptions

```
    momentumSource1
    {
        type            vectorSemiImplicitSource;

        $injector1;

        volumeMode      absolute;
        injectionRateSuSp
        {
            U.air           ((0 -1.0e-6 0) 0); // kg*m/s^2
        }
    }

    energySource1
    {
        type            scalarSemiImplicitSource;

        $injector1;

        volumeMode      absolute;
        injectionRateSuSp
        {
            e.air       (90 0); // kg*m^2/s^3
        }
    }
}
```

`injector1` is an inline sub-dictionary that defines the duration of source term (here from the start of simulation up to 30 s) and the collection of cells that the source term is applied for (here, cell set `injection1`). The process of creating a cell set `injection1` will be described later.

Three conservation equation are solved for the gas phase for which you must apply the source term: continuity, momentum and energy equation (here sensible internal energy based on the settings in file thermophysicalProperties.air). In table below you will see φ variable for each equation and the units of each source term component (if `volumeMode` is `absolute`[2]):

	φ variable	Unit in SI system for absolute volumeMode		
		φ	S_u	S_p
Continuity	$\alpha\rho$	kg/m^3	kg/s	m^3/s
Momentum	$\alpha\rho U$	kg/(m^2.s)	kg.m/s^2	m^3/s
Energy	$\alpha\rho.e$	kg/(m.s^2)	kg.m^2/s^3	m^3/s

[2] the other option for `volumeMode` is `specific` in which the source term is a volumetric term.

In the `injectionRateSuSp` sub-dictionaries, you must supply the name of primitive field of the conservation PDE. For continuity equation, the primitive field is `thermo:rho.air`; for momentum equation, `U.air`; and for energy equation `e.air`. In front of the primitive field name specify the source term components, the first value in the parentheses is S_u and the second one is S_p.

The cell set `injection1` is used to define a set of cells that the source term is applied for (here, it is the sparger). To create this cell set, you need to use `topoSet` utility. The settings can be found in system/topoSetDict file.

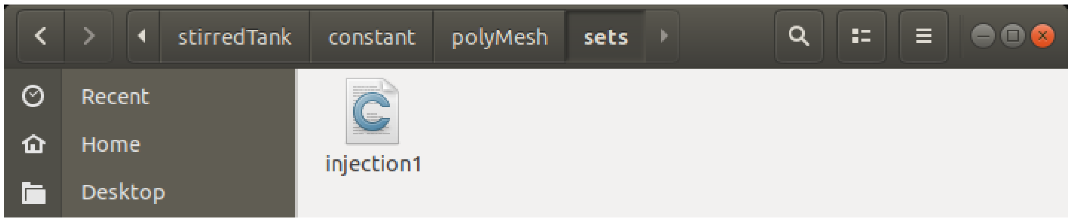
system/topoSetDict

```
actions
(
  {
    name    injection1;
    type    cellSet;
    action  new;
    source  boxToCell;
    sourceInfo
    {
      box (-0.02 -0.13 -0.02) (0.02 -0.12 0.02);
    }
  }
);
```

A `new` cell set with name `injection1` will be created whose cells are inside the `box` with minimax points `(-0.02 -0.13 -0.02)` and `(0.02 -0.12 0.02)`. Execute the following command to create this cell set:

```
> topoSet
```

this cell set will be saved in constant/polyMesh/sets/injection1.



4.8. Boundary and initial conditions

Initial and boundary conditions of the filed variables are defined in folder 0. In table below, a brief overview of the important boundary conditions is given. Note that the exact syntax for boundary conditions can be found in the simulation setup files.

Field name	internalFiled	Impeller condition	Outlet condition	vessel condition

alpha.air	0	zeroGradient	inletOutlet
			phi = phi.air
			value = 1
				zeroGradient

alpha.water	1	zeroGradient	inletOutlet
			phi = phi.water
			value = 0
				zeroGradient

p_rgh	101325	fixedFluxPressure
		value = $internalField
			prghPressure
			p = $internalField
			value = $internalField
				fixedFluxPressure
				value = $internalField

U.air	(0 0 0)	fixedValue
		value = (0 0 0)
			pressureInletOutletVelocity
			phi = phi.air
			value = $internalField
				fixedValue
				value = (0 0 0)

U.water	(0 0 0)	fixedValue
		value = $internalField
			pressureInletOutletVelocity
			phi = phi.water
			value = $internalField
				fixedValue
				value = (0 0 0)

T.air	300	zeroGradient;	inletOutlet;
			phi = phi.air;
			value = $internalField;
				zeroGradient;

T.water	300	zeroGradient;	inletOutlet;
			phi = phi.water;
			value = $internalField;
				zeroGradient;

`inletOutlet` condition is applied for outlet patch to prevent reverse flow into the tank. The impeller is treated as a fixed wall, as we know that in MRF method, the impeller is assumed to be fixed with no-slip condition on the surface. For volume fraction fields `zeroGradient` is applied at walls and `fixedFluxPressure`, for p_rgh field.

The reactor should be initially filled with water up to 0.15 m. `setFields` utility is used to create an initial (non-uniform) patch for phase volume fraction. The settings can be found in system/setFieldsDict.

system/setFieldsDict

```
defaultFieldValues
(
    volScalarFieldValue alpha.air 1
    volScalarFieldValue alpha.water 0
);

regions
(
    boxToCell
    {
        box (-0.21 -0.2 -0.21) (0.21 0.15 0.21);
        fieldValues
        (
            volScalarFieldValue alpha.air 0
            volScalarFieldValue alpha.water 1
        );
    }
);
```

The default settings are first applied for all cells and then the values specified in the `regions` entry overwrite the default values. Here, the default settings are α_{air} = 1 and α_{water} = 0, which create and an empty tank. Then, volume fractions of the cells reside in the cuboid `box` defined by two points `(-0.21 -0.2 -0.21)` and `(0.21 0.15 0.21)` are set to α_{air} = 0.0 and α_{water} = 1.0. Execute the following command to apply these initial conditions:

```
> mkdir 0
> cp 0.orig/* 0/
> setFields
```

5. Running the simulation

If all steps are done correctly, you are ready to run the solver. Execute the following commands from the case directory:

```
> twoPhaseEulerFoam
```

A summary of all commands to run this simulation is listed below for your reference.

```
> blockMesh
> snappyHexMesh -overwrite
> topoSet
> mkdir 0
> cp 0.orig/* 0/
> setFields
> twoPhaseEulerFoam
```

6. Results

Some snapshots of the bubble column simulation are presented here.

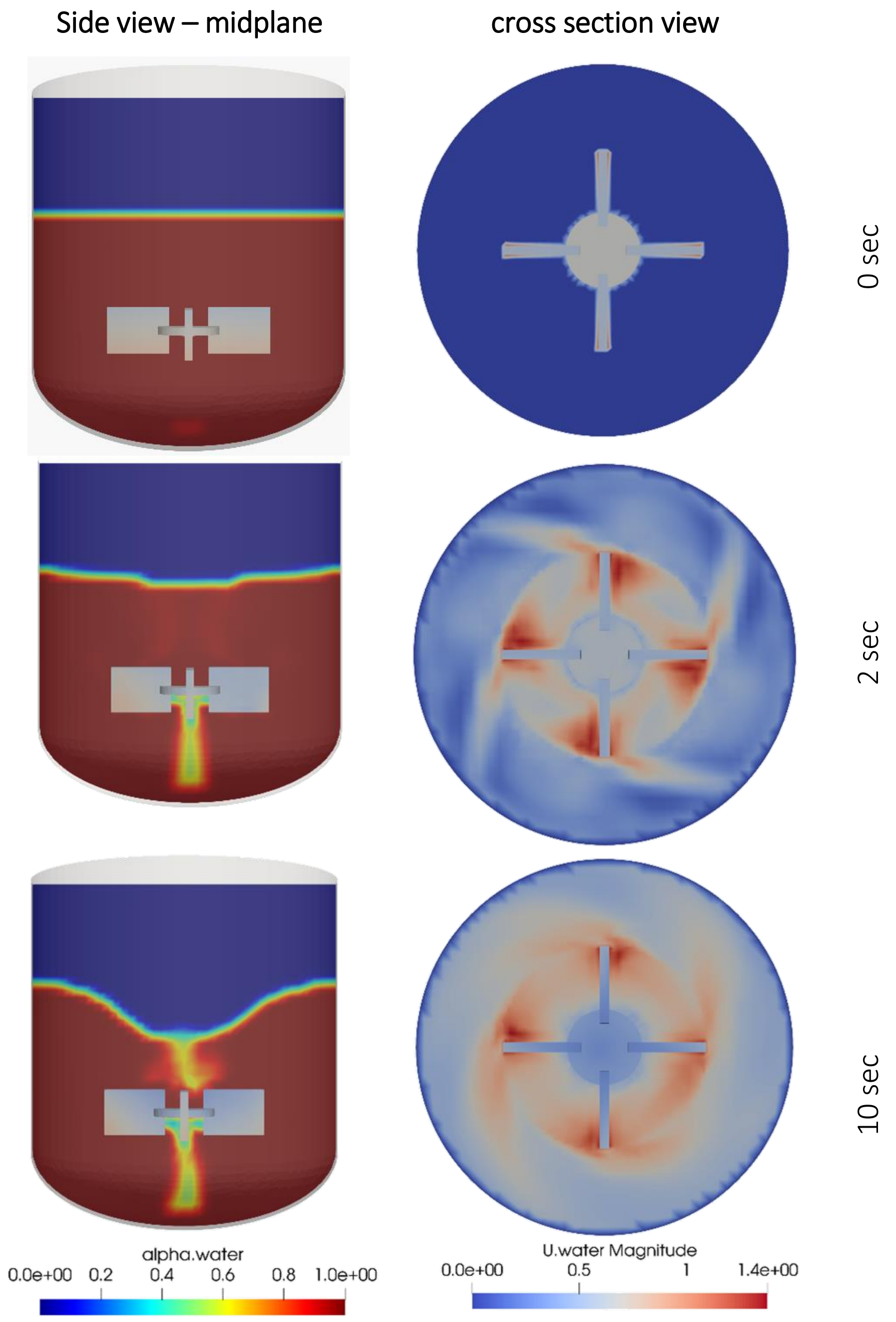


Figure 3: (left) water volume fraction contour on a plate at the center of the vessel and (right) the velocity contour on a cross-sectional plane near the impeller.