Air-flow in bed-room with rotating fan explaining cyclicAMI in OpenFoam

Divyesh D. Variya
Gujarat Technological University, divyeshvariya7@gmail.com
Government Engineering College, Valsad
+91 7777908833

Abstract - **Aim of the case study is to simplify understanding of cyclicAMI in OpenFoam. Rotating, dynamic mesh gives clear and realistic solutions of real life problems. OpenFoam has wide range of possibility to solve engineering problems as per user criterion. Prior objective of this case study will be to explain rotating mesh feature in OpenFoam. A transient and RAS (turbulent) theory will be use while computing. By using simple geometry, snappyHexMesh and pimpleFoam solver, results to be analyzed.**

Index Terms – CyclicAMI, PimpleFoam, SnappyHexMesh

INTRODUCTION

In engineering, CFD of moving part/component is challenging work to perform, Specially with rotating geometry. Many object, in mechanical engineering are made of shaft, gear and so many rotating component. Yes, CFD can be perform with stationary part with desired angle position but, actual visualization gives only with rotating mesh. To compare two design with its shape optimization and flow design, moving mesh simulation gives best results. In this paper simulation of a rotating fan inside a bed room is taken as case.

In OpenFoam, three types of setup available to solve rotating mesh.

1. Single Rotating Frame (SRF)
2. Multiple Reference Frame (MRF)
3. Arbitrarily Mesh Interface (AMI)

To solve rotating mesh simulations, two fundamental question should we ask before begin with geometry creation.

(Domain type)
1. Does our model include some stationary regions or not?

(Time dependency)
2. Which do we want to obtain, a steady or transient solution?

	Only Rotating Region	**With Stationary Region**
Steady	SRFSimpleFoam	SimpleFoam with modification in fvOption
Transient	SRFPimpleFoam	pimpleFoam

Table 1 : Types of simulations for rotating mesh

In SRF simulations, Mesh will not move but, it will give actual simulation results. In this case if we remove room, bed, door and window, It will be a SRF simulation. Because, after removing stationary fields, there is no requirement of motion. Only rotational angular velocity is required to define.

The Multiple Reference Frame (MRF) model computes fluid flow using both the rotating and stationary reference frames.

- Rotating zone is solved in the rotating frame
- Stationary zone is solved in the stationary frame

In above both simulations, Mesh will not rotate. It gives as the same results as it is rotating. To solve case with rotating mesh, which should be visible in results, AMI is the best option.

In Arbitrarily Mesh Interface (AMI), dynamicMeshDict file should be present in case. In this dictionary, cellZone need to define for our AMI patch and Fan patch. SolidBodyMotionFunction defines type of motion in case. Various types of motion can be applied ti the cellZone, like:

1. Transitional Motion
linearMotion
oscillatingLinearMotion

2. Rotating Motion
rotatingMotion
axisRotationMotion
oscillatingRotatingMotion

3. Ship Design Analysis
SDA

4. Combination of the above motions
multiMotion

GEOMETRY

A room with one bed, one fan, one door and two windows is shown below. Dimensions of the geometry is in mm. So the room is 6m X 5m X 10m in dimensions.

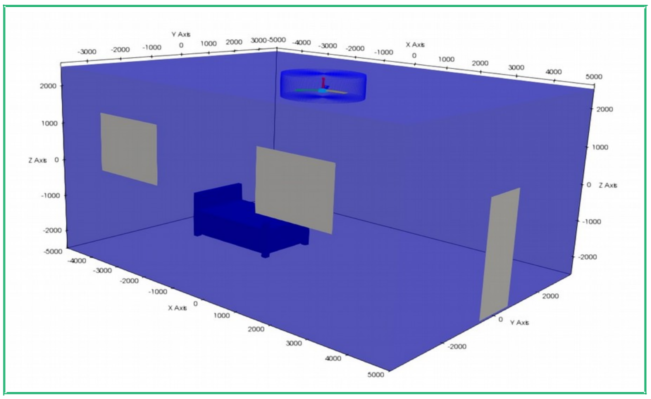

Figure 1 : Geometry

Note: One AMI boundary should be made while making geometries. This AMI should cover our rotating component. In this geometry, At the top of ceiling, a cylindrical part covers a Fan. That will be AMI patch in this case.

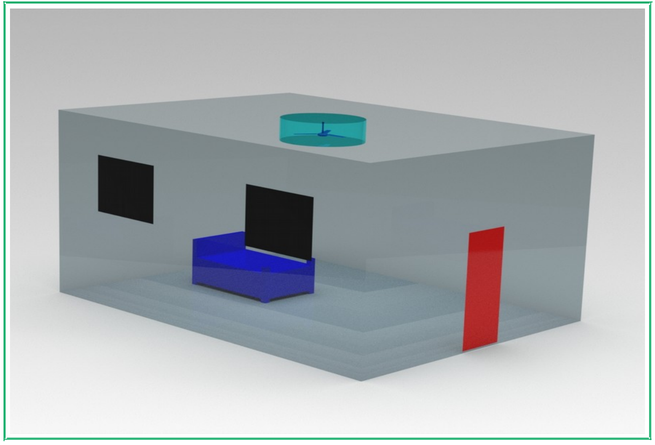

Figure 2 : Part file

This AMI part will not take part in any of the simulation process. It will just define boundary of stationary and rotating parts.

MESHING

In this case study all tools to be used for meshing are from openFoam itself.

I. blockMesh

BlockMesh defines domain of complete mesh. Geometry of the room should be covered by blockMesh. In this case, all patches defines from .stl files so no need to define in blockMeshDict.

Here is the blockMesh data:

Axis	Dimensions	Length	Cells
X	(-6,6)	12 meter	120
Y	(-4,4)	8 meter	80
Z	(-3,3)	6 meter	60
TOTAL			5,76,000

Table 2 : BlockMeshDict

II. surfaceFeatureExtract

extractionMethod and includedAngle should be define in surfaceFeatureExtractDict. In this case, all edges are selected by applying 180 angle and .objs are written.

III. SnappyHexMesh

SnappyHexMesh is in-build tool for internal and external meshing in OpenFoam. Here is source code for meshing,

```
//*********************************************************//
castellatedMesh                 true;
snap                            true;
addLayers               false;
//*********************************************************//
```

As shown in above code, castellated and snap options are ON for the meshing, for more accurate results, addLayers can be turn on too.

All Geometries are defined for meshing as below. If Geometry is made in mm and exported directly to .stl, it will automatically taken in METER UNIT. So, scaling is required as shown below.

```
//*********************************************************//
geometry
{

  AMI
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "AMI.stl";
  }
  door
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "door.stl";
  }
  fan
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "fan.stl";
  }
  sofa
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "sofa.stl";
  }
  room
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "room.stl";
  }
  window_1
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "window_1.stl";
  }
  window_2
  {
    type            triSurfaceMesh;
         scale      0.001;
    file            "window_2.stl";
  }

}
//*********************************************************//
```

In castellatedMeshControls, maxLocalCells and maxGlobalCells must be define. It controls the meshing. CastellatedMesh controls and refinement conditions are as below,

```
//*********************************************************//
castellatedMeshControls
{
  maxLocalCells 100000;
  maxGlobalCells 8000000;
  minRefinementCells 0;
  nCellsBetweenLevels 2;

  features
  (
    {
                  file "AMI.eMesh";
                  scale 0.001;
                  level 2;
         }
    {
                  file "fan.eMesh";
                  scale 0.001;
                  level 6;
         }
    {
                  file "door.eMesh";
                  scale 0.001;
                  level 0;
         }
    {
                  file "sofa.eMesh";
                  scale 0.001;
                  level 2;
         }
    {
                  file "room.eMesh";
                  scale 0.001;
                  level 0;
         }
    {
                  file "window_1.eMesh";
                  scale 0.001;
                  level 0;
         }
    {
                  file "window_2.eMesh";
                  scale 0.001;
                  level 0;
         }
  );

  refinementSurfaces
  {
    AMI
    {
      level (3 3);
      faceType boundary;
      cellZone rotatingZone;
      faceZone rotatingZone;
      cellZoneInside inside;
    }
    fan{ level (6 6);}
    door{ level (0 0);}
    window_1{ level (0 0);}
    window_2{ level (0 0);}
    room{ level (0 0);}
    sofa{ level (2 2);}
  }
```

```
    resolveFeatureAngle 30;

    refinementRegions
    {
        AMI{ mode inside; levels ((1E15 3));}
    }

    locationInMesh (0 0 0);
    allowFreeStandingZoneFaces false;
}
//*******************************************************//
```

As shown above, **locationInMesh** should be add wisely. It should be inside of keeping data. If external flow should determine then, it should be inside domain but outside the .stl file.

In refinementSurfaces, AMI patch is desined as boundary and its cellZone defined as rotatingZone (any name can be given).

Solver iteration to refine and keeping of cells to removing of cells are define in Snap control portion. Mesh quality control will be define in meshqualitycontrols. Shifting of cells and restoring of cells from snap to castalleted is done here. Snap control data is as below,

```
//*******************************************************//
snapControls
{
    nSmoothPatch 3;
    tolerance 1.0;
    nSolveIter 300;
    nRelaxIter 5;
    nFeatureSnapIter 10;
    implicitFeatureSnap true;
    explicitFeatureSnap false;
    multiRegionFeatureSnap true;
}

meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality -1;
    minArea -1;
    minTwist 0.01;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    nSmoothScale 4;
    errorReduction 0.75;
    relaxed
    {
        maxNonOrtho 75;
    }
}
//*******************************************************//
```

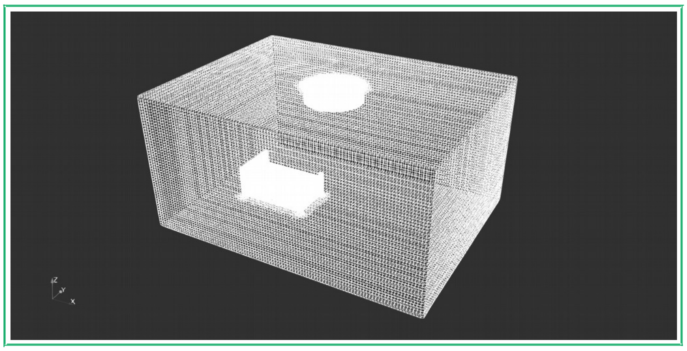

Figure 3 : Wire-frame of Mesh

Note: Not covering addLayers portion because not going use in this case study. If interested in that too, please refer INS_Vikramaditya case study.

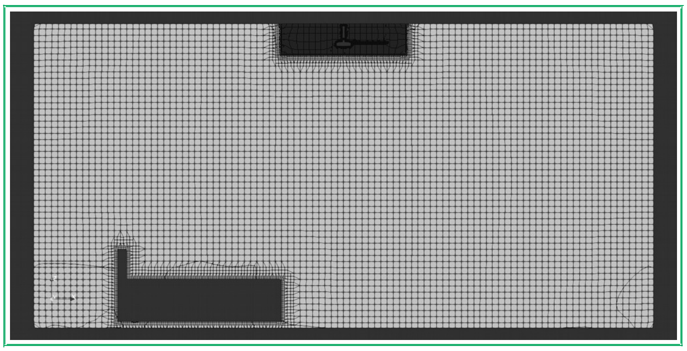

Figure 4 : 2D slice of Mesh

IV. RenumberMesh and checkMesh

It is suggested all time to perform renumberMesh to improve simulation calculation. Checkmesh gives, wheter mesh is ok or not. Here are few data for this case,

Number of cells	22,19,176
Max aspect ratio	7.383145
Max skewness	6.3611
Highly skew faces	5

Table 3: checkMesh results

One failed mesh is there but going with same results.

Results after meshing done,

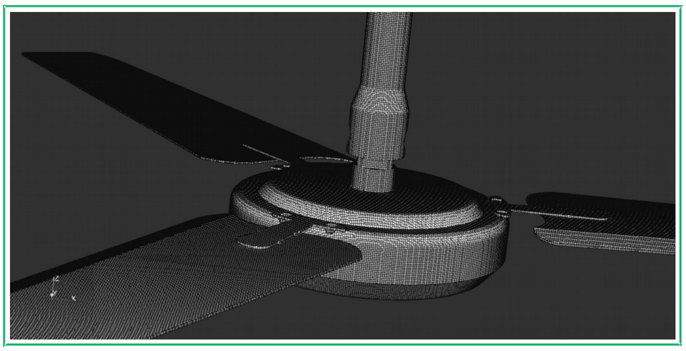

Figure 5 : Fan Mesh 1

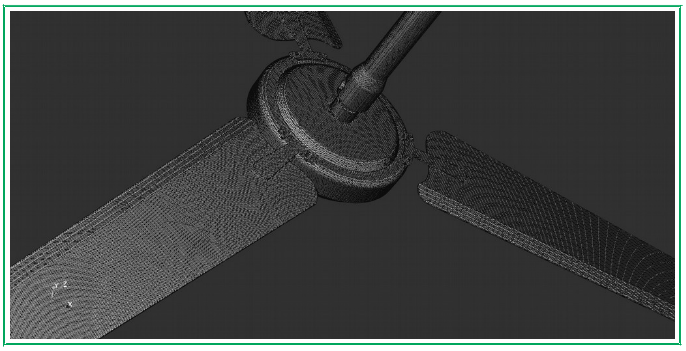

Figure 6 : Fan Mesh 2

V. CreatePatch

In the case of rotating region and stationary region, one boundary should be setup. That we have define as AMI, from one geometry, we need to give two boundary. That will be AMI1 and AMI2. Source code is as below,

```
//*******************************************************//
patches
(
 {
     //- Master side patch
     name          AMI1;
     patchInfo
     {
        type          cyclicAMI;
        matchTolerance  0.0001;
        neighbourPatch  AMI2;
        transform       noOrdering;
    }
    constructFrom patches;
    patches (AMI);
}

{
    //- Slave side patch
    name            AMI2;
    patchInfo
    {
        type            cyclicAMI;
        matchTolerance  0.0001;
        neighbourPatch  AMI1;
        transform       noOrdering;
    }
    constructFrom patches;
    patches (AMI_slave);
}
);
//************************************************************************//
```

VI. DynamicMeshDict

In Dynamic Mesh file cellzones should be our rotating zone that we have decided while sHM operaion. Origin and axis should be known otherwise it will start rotating whole zone wrong way. Omega is angular velocity in rad/s.

```
//************************************************************************//
dynamicFvMesh   dynamicMotionSolverFvMesh;

motionSolverLibs ("libfvMotionSolvers.so");

motionSolver    solidBody;

cellZone        rotatingZone;

solidBodyMotionFunction  rotatingMotion;

origin      (0 0 0);
axis        (0 0 1);
omega       10;
//*********************************************************************** //
```

SIMULATION

The CFD analysis of the airflow in a bedroom was done using the software OpenFOAM (v-6.0).

I. Boundary Conditions

Airflow enters with 8m/s fixed velocity by two windows. Airflow is in Normal to the window face and upper direction. Outlet is given to door, which is pressureInletOutletVelocity. Velocity of fan is defined in dynamicmeshdict and in boundary conditions, it is given as movingWallVelocity. Pressure is given to all boundaries as fixedFluxPressure and rest RAS conditions are stated in table below,

	K	Nut	Omega	p	U
AMI1	0.00341	1e-5	0.1	0	(0,0,0)
AMI2	0.00341	1e-5	0.1	FFP 0	(0,0,0)
fan	0.00341	1e-5	0.1	FFP 0	MovingWallVelocity (0,0,0)
window_1	0.00341	ZG	0.1	FFP 0	FV (0,8,8)
window_2	0.00341	ZG	0.1	FFP 0	FV (0,8,8)
door	ZG	ZG	ZG	FV 0	PressureInletOutlet Velocity (0,0,0)
room	0.00341	1e-5	0.1	FFP 0	noSlip
sofa	0.00341	1e-5	0.1	FFP 0	noSlip

Table 4 : Boundary Conditions

II. PimpleFoam Solver

In this case, we need to analyze Turbulent, Transient flow for an in-compressible fluid, we have used pimpleFoam solver. Here, no need to solve energy equation due to the in-compressibility. PimpleFoam is Transient solver for incompressible, turbulent flow of Newtonian fluids, with optional mesh motion and mesh topology changes. The PIMPLE Algorithm is a combination of PISO (Pressure Implicit with Splitting of Operator) and SIMPLE (Semi-Implicit Method for Pressure-Linked Equations). All these algorithms are iterative solvers but PISO and PIMPLE are both used for transient cases whereas SIMPLE is used for steady-state cases.

Continuity equation:

$$\nabla\cdot U=0 \qquad (1)$$

and momentum equation:

$$\partial/\partial t(U)+\nabla\cdot(UU)-\nabla\cdot R=-\nabla p+S_U \qquad (2)$$

III. Results

The simulation is run with deltaT is 0.0002 and writecontrol of adjustableRunTime. WritePrecision taken to 7 and timePrecision is 6. Max cornant number is 1 and maxDeltaT is 1. Simulation run for 0.555 seconds so, its flow is not fully developed for complete simulation it can be run further. These data are simulated in intel CORE i5 8th Gen processor and 8 GB RAM. Time taken for these simulation is 3,45,000 seconds with 4 cores in MPI parallel.

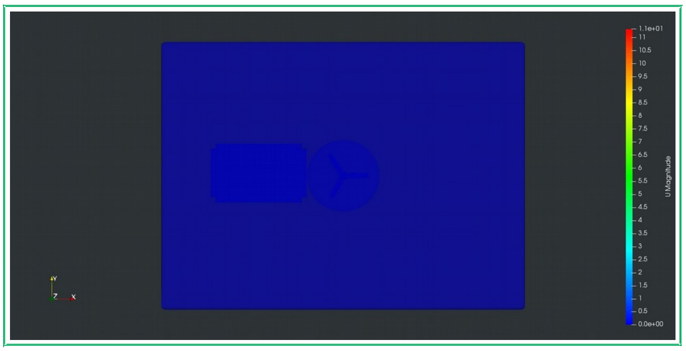

Figure 7 : T=0 Sec.

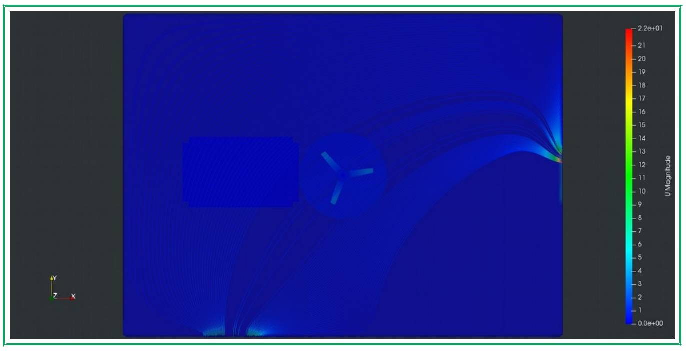

Figure 8 : T=0.015 Sec.

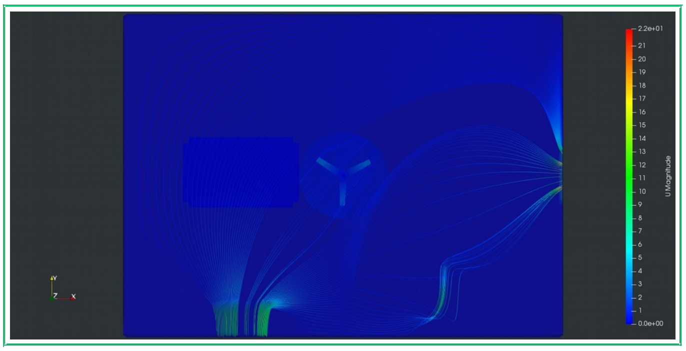

Figure 9 : T=0.255 Sec.

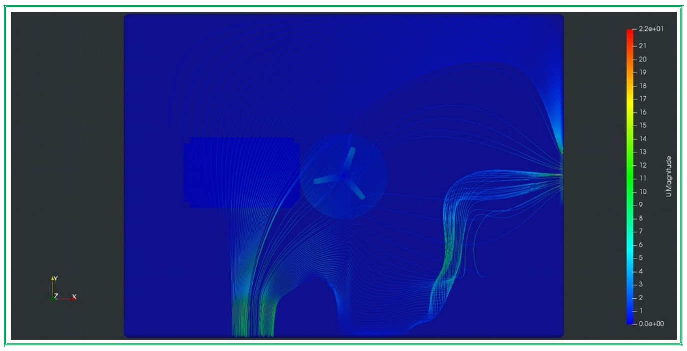

Figure 10 : T=0.54 Sec.

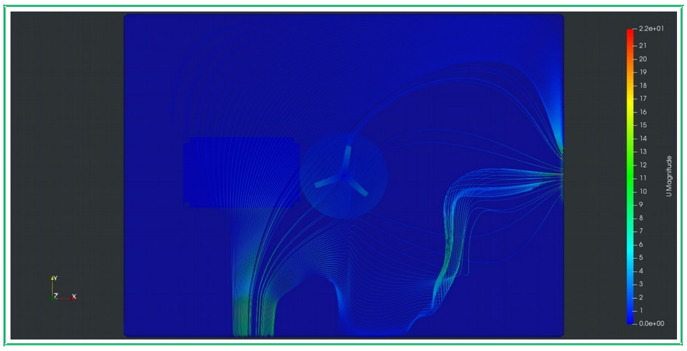

Figure 11 : T=0.555 Sec.

CONCLUSION

Yet, flow in results shown above is not fully developed but main objective of dynamic mesh is completely satisfy. As visuals from paraview shows, fan inside room is rotating and streamline shows flow through Y-Axis on point of fan center. Simulation can further run and better results can be visualize.

ACKNOWLEDGMENT

It is always a pleasure to remind the fine people in the Indian Institute of Technology (IIT), Bombay and Government Engineering college, Valsad for their the sincere guidance I received to uphold my theoretical as well as CAE skills in Computational fluid dynamics.

First of all, thanks to my parent for giving encouragement, enthusiasm and invaluable assistance to me. Without all this, I might not be able to complete this subject properly.

Second, I would like to thanks to Professor Shivasubramanian Gopalakrishnan (Department of Mechanical Engineering) forgive me the opportunity to do the marvelous project study. He also gives me their guidance and support.

Thirdly, I also want to express my deepest thanks to Mr Rohit Panday as an industry professional advisory for CAE Department that has helped me a lot in dealing with the industrial project. He had supported me by showing a different method of information collection about the CAE. He helped all the time when I needed and He gave the right direction toward completion of the project.

Besides, I would like to thank Mr Sathish kanniappan, Miss. Deepa Vedartham for extending their friendship towards me and making a pleasure-training environment in the IIT, Bombay during the internship.

A paper is not enough for me to express the support and guidance I received from them almost for all the work I did there.

Finally, I apologize for all other unnamed who helped me in various ways to have a good training.

REFERENCES

[1] https://www.slideshare.net/fumiyanozaki96/cfd-for-rotating-machinery-using-openfoam

[2] https://www.slideshare.net/fumiyanozaki96/openfoam

AUTHOR INFORMATION

Divyesh Variya (M’97) received the B.E. degree in mechanical engineering from the gujarat technological university, in 2018 and also worked as an intern under Prof. Shivasubramanian Gopalakrishnan for FOSSEE (Free and Opensource Software for Education) Project on OpenFOAM in Indian Institute of Technology, Bombay. His work interests include all CAE projects with linear/non linear structural analysis, computational fluid dynamics, dynamic robotics analysis, failure analysis and prevention, and design development.