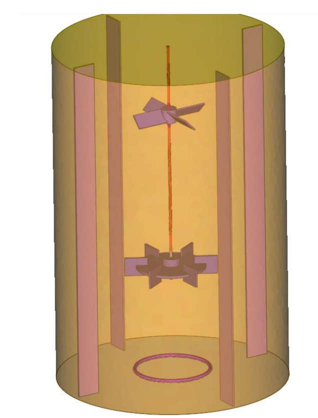
Multiphase Flow Project –
Multiphase Mixing in a Stirred Vessel
MRF vs Dynamic Mesh in OpenFOAM

10.27.2025

Furkan Erikci



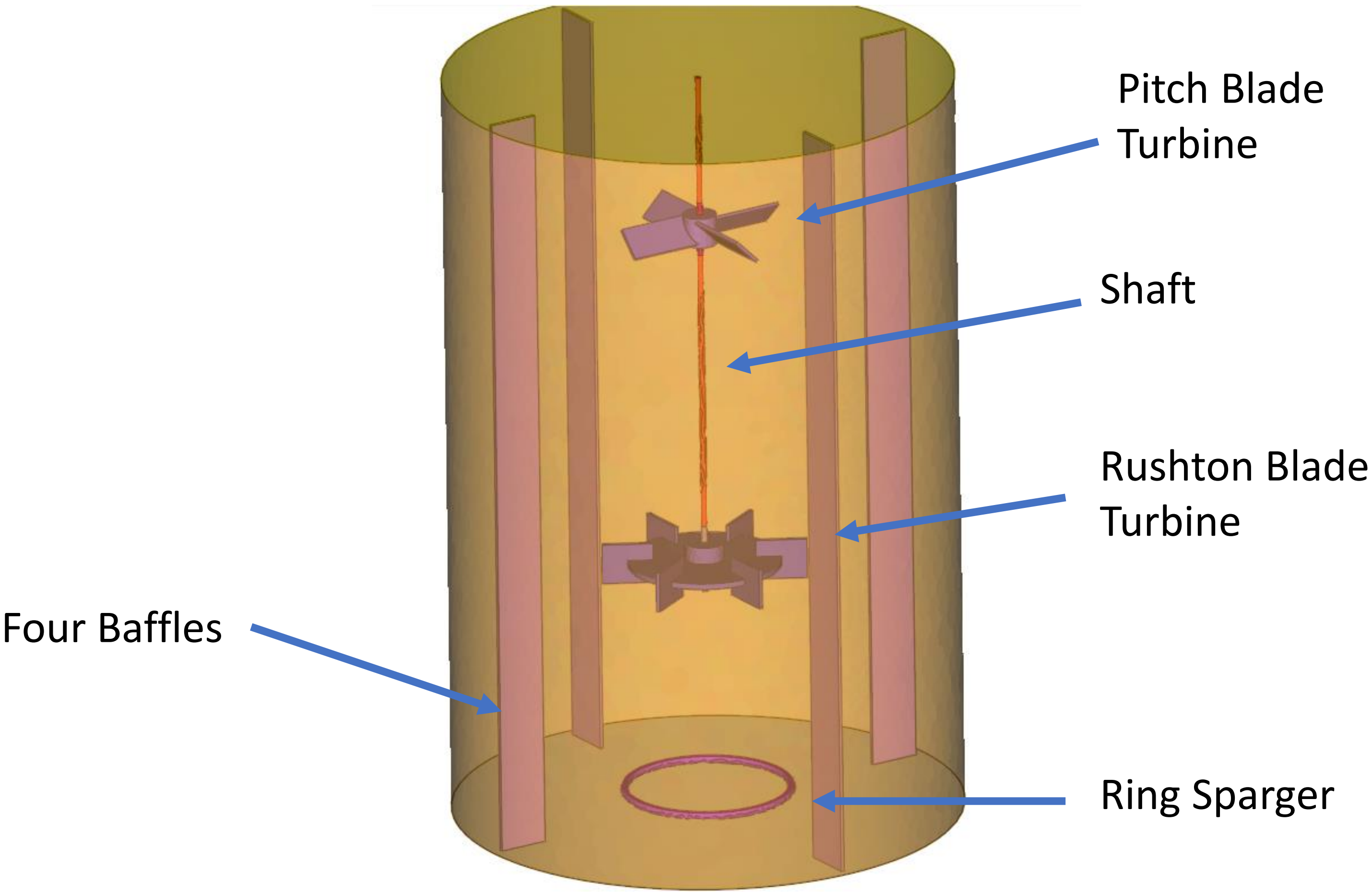
Objective

In industrial applications, mixing systems play a vital role in chemical and biotechnological processes, where efficient mixing and heat transfer are essential.
There are mainly three approaches in OpenFOAM in order to model the rotation body problem in Openfoam. This method are Single Rotating Frame (SRF), Multiple Rotating Frame (MRF), Dynamic Mesh (AMI – Sliding Mesh).
In this study, the rotational motion of a stirred vessel is simulated using two different **transient CFD approaches**:

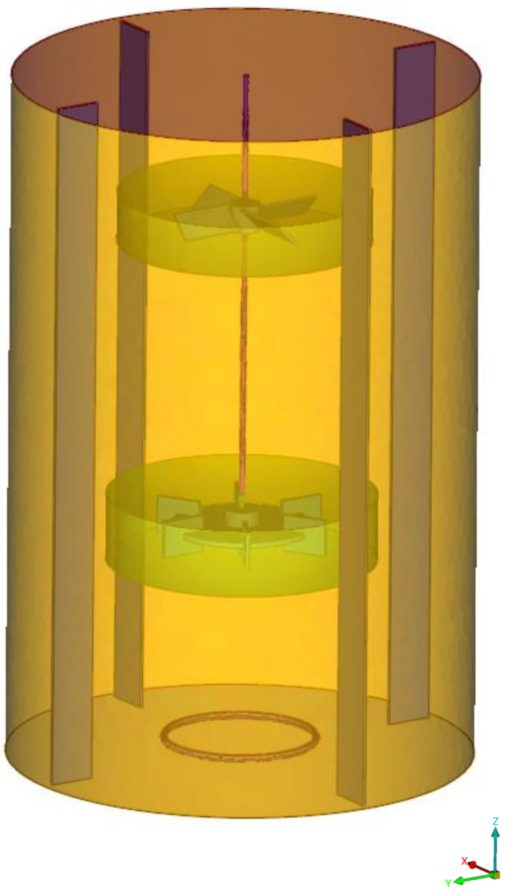
- **MRF (Multiple Reference Frame):** A quasi-steady approach that represents rotation through a rotating reference frame
- **Dynamic Mesh:** A fully transient approach that explicitly accounts for the real-time motion of the impeller

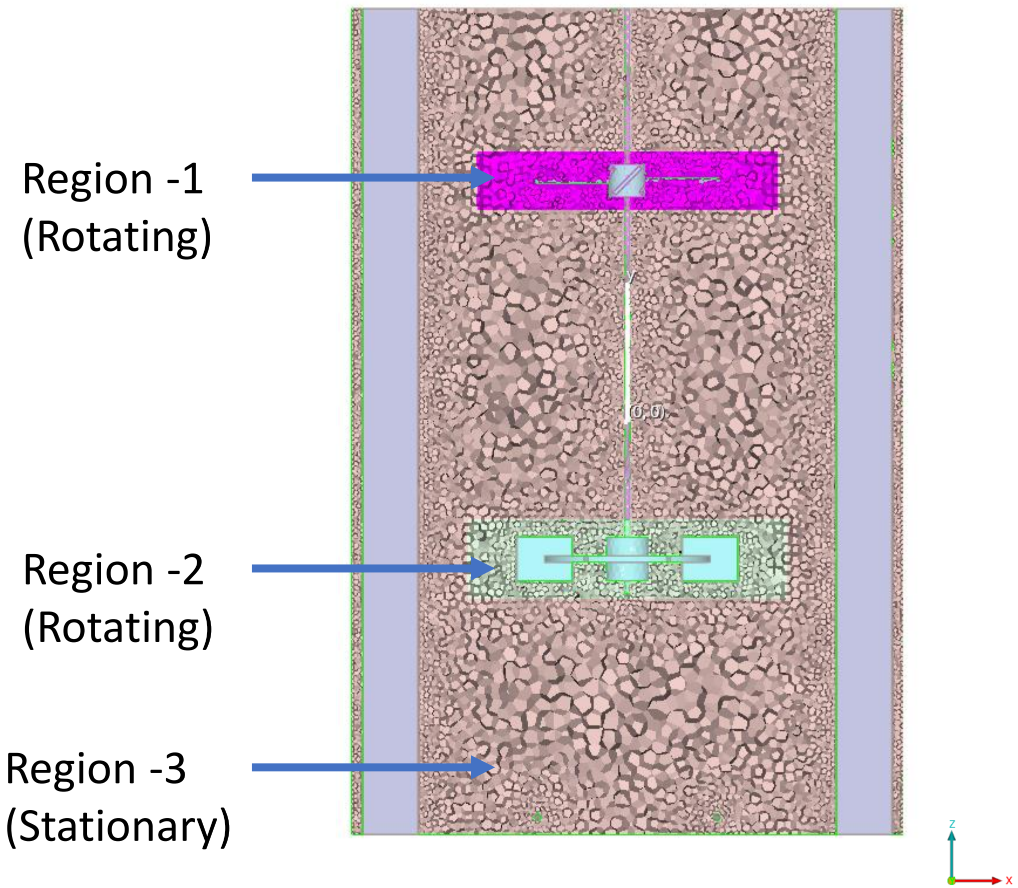
The objective is to compare these two methods in terms of **flow field characteristics, torque evolution, and computational efficiency**.

Geometry


Pitch Blade Turbine
Shaft
Rushton Blade Turbine
Four Baffles
Ring Sparger

Mesh Display





OpenFOAM Folder

Transport Properties =

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2506                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases          (water air);

water
{
    transportModel  Newtonian;
    nu              1e-06;
    rho             1000;
}

air
{
    transportModel  Newtonian;
    nu              1.48e-05;
    rho             1;
}

sigma           0.07;


// ************************************************************************* //
```

Turbulence Properties =

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2506                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType      RAS;

RAS
{
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}


// ************************************************************************* //
```

OpenFOAM Folder

- In this study, the rotational motion of a stirred vessel is simulated using MRF and Dynamic Mesh method.
- The dynamicMultiMotionSolverFvMesh method was used in the Dynamic Mesh approach since the model includes two rotating regions.

DynamicMesh Properties =

```
FoamFile
{
    version 2.0;
    format  ascii;
    0 references
    class   dictionary;
    object  dynamicMeshDict;
}

dynamicFvMesh   dynamicMultiMotionSolverFvMesh;
motionSolverLibs ("libfvMotionSolvers.so");

dynamicMultiMotionSolverFvMeshCoeffs
{
    MRF_1
    {
        solver                  solidBody;
        cellZone                fluid_mrf_1-1;

        solidBodyMotionFunction rotatingMotion;
        rotatingMotionCoeffs
        {
    origin    (0.0 0.0 0.08);
    axis      (0 0 1);
    omega     constant 12; //
        }
    }

    MRF_2
    {
        solver                  solidBody;
        cellZone                fluid_mrf_2-0;

        solidBodyMotionFunction rotatingMotion;
        rotatingMotionCoeffs
        {
    origin    (0.0 0.0 0.190);
    axis      (0 0 1);
omega constant 12; //
        }
    }
}
```


FlowThermoLab

OpenFOAM Folder

MRF Properties =

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      MRFProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

MRFImpeller
{
    cellZone    fluid_mrf_1-1;
    active      yes;

    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches (ns-imp2-internal_SHADOW  ns-imp2-internal);

    origin    (0.0 0.0 0.08);
    axis      (0 0 1);
    omega     constant 12; // ~120 RPM
}


MRFImpeller1
{
    cellZone    fluid_mrf_2-0;
    active      yes;

    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches (ns-imp1-internal_SHADOW  ns-imp1-internal);

    origin    (0.0 0.0 0.190);
    axis      (0 0 1);
    omega     constant 12; // ~120 RPM
}
```

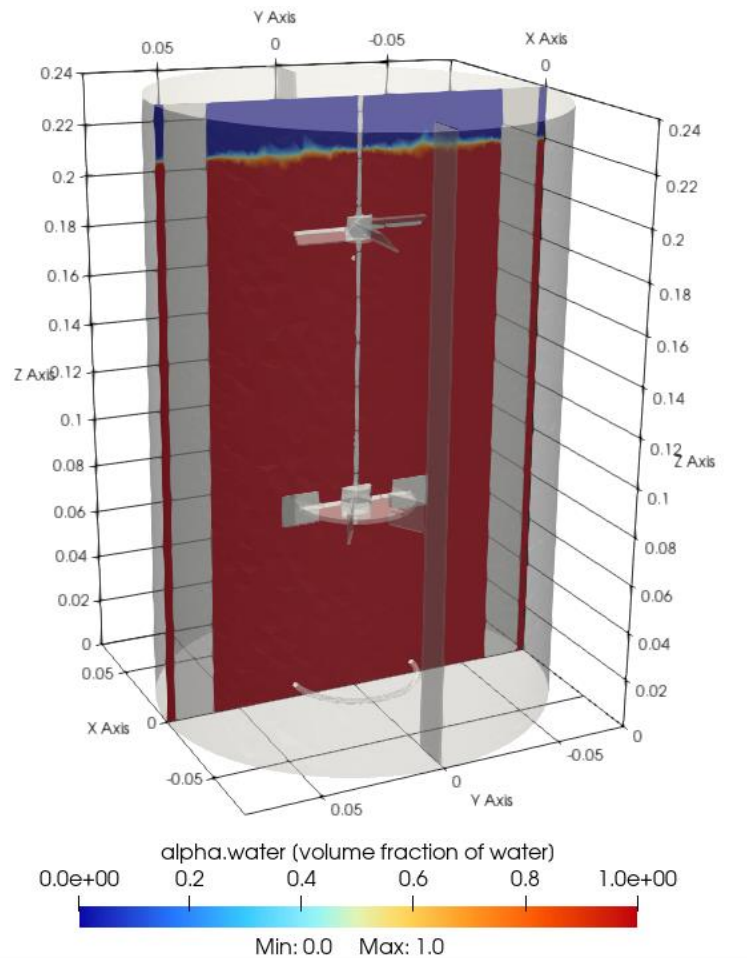
DynamicMesh Properties =

```
FoamFile
{
    version 2.0;
    format  ascii;
    0 references
    class   dictionary;
    object  dynamicMeshDict;
}

dynamicFvMesh   dynamicMultiMotionSolverFvMesh;
motionSolverLibs ("libfvMotionSolvers.so");

dynamicMultiMotionSolverFvMeshCoeffs
{
    MRF_1
    {
        solver                  solidBody;
        cellZone                fluid_mrf_1-1;

        solidBodyMotionFunction rotatingMotion;
        rotatingMotionCoeffs
        {

    origin    (0.0 0.0 0.08);
    axis      (0 0 1);
    omega     constant 12; //
        }
    }

    MRF_2
    {
        solver                  solidBody;
        cellZone                fluid_mrf_2-0;

        solidBodyMotionFunction rotatingMotion;
        rotatingMotionCoeffs
        {
    origin    (0.0 0.0 0.190);
    axis      (0 0 1);
omega constant 12; //
        }
    }
}
```

OpenFOAM Folder

- **setFields** is important because it defines the initial phase distribution or region-specific conditions, ensuring accurate initialization for multiphase or dynamic simulations.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2506                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    0 references
    class       dictionary;
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha.water 0
);

regions
(
    boxToCell
    {
        box (-0.80 -0.80 0) (0.80 0.80 0.22);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);


// ************************************************************************* //
```



OpenFOAM Folder

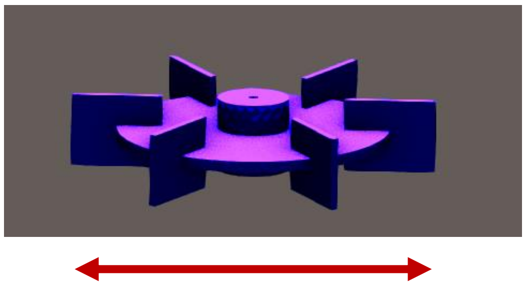
- The **controlDict** file is essential because it controls the **simulation setup**, including time settings, output frequency, and function objects.
- **interFoam** is a multiphase flow solver in OpenFOAM used to simulate the **interaction between two immiscible fluids** using the **VOF (Volume of Fluid)** method.

```
FoamFile
{
    version     2.0;                // File format version
    format      ascii;              // Human-readable text format
    0 references
    class       dictionary;         // OpenFOAM dictionary type
    object      controlDict;        // Object name (control settings)
}
// ************************************************************************* //

application     interFoam;          // Solver used (two-phase flow solver)

startFrom       latestTime;         // Start simulation from the latest saved time
startTime       0;                  // Start time (0 seconds)
stopAt          endTime;            // Stop simulation at the end time
endTime         10;                 // Total simulation time (10 seconds)

deltaT          0.001;              // Time step size (seconds)
writeControl    adjustable;         // Output frequency is adjustable
writeInterval   0.02;               // Write results every 0.02 simulated seconds
purgeWrite      0;                  // Keep all time directories (no deletion)

writeFormat     ascii;              // Output data format (text)
writePrecision  6;                  // Output precision (6 decimal places)
writeCompression off;               // Disable file compression

timeFormat      general;            // Time output format (general notation)
timePrecision   6;                  // Display time with 6 decimal digits

runTimeModifiable yes;              // Allows editing setup while running
adjustTimeStep  off;                // Fixed time step (no automatic adjustment)

maxCo           10;                 // Maximum Courant number
maxAlphaCo      5;                  // Max Courant number for phase fraction
maxDeltaT       1;                  // Maximum allowed time step size
```

```
functions
{
    forces
    {
        type            forces;             // Function type: calculates forces and moments
        libs            ("libforces.so");   // Library that provides the 'forces' function

        writeControl    runTime;            // Output control based on run time
        writeInterval   0.01;               // Write every 0.01 seconds
        log             yes;                // Enable log output in terminal
        outputControl   timeStep;           // Output results at each time step
        outputInterval  1;                  // Output frequency (every 1 step)

        patches         (wall_impeller_1);  // Target boundary (first impeller surface)

        rho             rhoInf;             // Use constant density (incompressible)
        rhoInf          1000;               // Density of water [kg/m3]
        CofR            (0.0 0.0 0.190);    // Center of rotation for impeller 1
        pitchAxis       (0 0 1);            // Rotation axis direction (Z-axis)
    }

    forces_1
    {
        type            forces;             // Second force function for another impeller
        libs            ("libforces.so");   // Load same library for forces calculation

        writeControl    runTime;            // Output control based on simulation time
        writeInterval   0.01;               // Write every 0.01 seconds
        log             yes;                // Enable log messages
        outputControl   timeStep;           // Output every time step
        outputInterval  1;                  // Frequency of output

        patches         (wall_impeller_2);  // Target boundary (second impeller)

        rho             rhoInf;             // Constant density assumption
        rhoInf          1000;               // Density value for fluid
        CofR            (0.0 0.0 0.08);     // Center of rotation for impeller 2
        pitchAxis       (0 0 1);            // Rotation axis (Z-axis)
    }
}
```

Verification of Results through Tangential Velocity Calculation

- To ensure the accuracy of rotational motion parameters, the tangential velocity was calculated from the given angular velocity and fan diameter.
  This verification step confirms that the rotational speed defined in the simulation corresponds to the expected physical motion, ensuring the model's reliability.



D = 0.057 m

```python
# Fan Tangential Velocity Calculator
# Description: Calculates tangential velocity and RPM from angular velocity.

import math

# Given data
D = 0.057           # Diameter [m]
omega = 12          # Angular velocity [rad/s]

# Calculations
r = D / 2                                   # Radius [m]
v_t = omega * r                             # Tangential velocity [m/s]
N = (omega * 60) / (2 * math.pi)            # Convert angular velocity to RPM

# Output
print("Fan Tangential Velocity Calculation")
print("-----------------------------------")
print(f"Diameter (D): {D:.3f} m")
print(f"Angular Velocity (ω): {omega:.2f} rad/s")
print(f"Radius (r): {r:.3f} m")
print(f"Tangential Velocity (vₜ): {v_t:.3f} m/s")
print(f"Rotational Speed (N): {N:.1f} RPM")
```

```
Fan Tangential Velocity Calculation
-----------------------------------
Diameter (D): 0.057 m
Angular Velocity (ω): 12.00 rad/s
Radius (r): 0.029 m
Tangential Velocity (vₜ): 0.342 m/s
Rotational Speed (N): 114.6 RPM
```

OpenFOAM Folder