CFD using OpenFOAM
Lecture 7: Essential Flow Governing Laws & OpenFOAM Implementation

Part III : OpenFOAM Implementation and Illustration

Instructor : Sumant R Morab (Ph.D Research Scholar)
Co-ordinator : Prof. Janani S Murallidharan
Indian Institute of Technology, Bombay

Outline

Recap of Part I and II

Incompressible Flow Solvers in OpenFOAM

OpenFOAM Implementation

OpenFOAM Illustration

- Mass Conservation (2D):

$$\text{Continuity: } \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

- Mass Conservation (2D):

$$\text{Continuity:} \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

- Momentum Conservation (2D):

$$x\text{-Momentum:} \quad \rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial p}{\partial x}$$

$$y\text{-Momentum:} \quad \rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial p}{\partial y}$$

- Mass Conservation (2D):

$$\text{Continuity: } \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

- Momentum Conservation (2D):

$$x\text{-Momentum: } \rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial p}{\partial x}$$

$$y\text{-Momentum: } \rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial p}{\partial y}$$

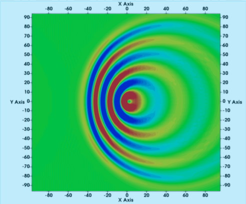
- Finite Volume Discretisation
- Challenges Faced & Solution Methodology

Division of contents for CFD & OpenFOAM Implementation

Part I : Governing Laws and PDE Format → $$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0$$

Part II : Discretisation & Solution Methodology → $$\frac{U_{i,j+1} - U_{i,j-1}}{\Delta x} = 0$$

Part III : OpenFOAM Implementation & Illustration →



Vector format of Navier Stokes Equation

- Since algorithms are implemented from vector notation-based equations, the N-S can be written as:

$$\nabla \cdot \overrightarrow{u} = 0 \tag{1}$$

Vector format of Navier Stokes Equation

- Since algorithms are implemented from vector notation-based equations, the N-S can be written as:

$$\nabla \cdot \overrightarrow{u} = 0 \tag{1}$$

$$\frac{\partial \rho \overrightarrow{u}}{\partial t} + \nabla \cdot (\overrightarrow{m}\,\overrightarrow{u}) = -\nabla p + \mu \Delta \overrightarrow{u} \tag{2}$$

where, $\overrightarrow{m}$ & $\overrightarrow{u}$ represents mass-flux and velocity vector respectively
∇ is the divergence vector, given by

$$\nabla = \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k}$$

- Since algorithms are implemented from vector notation-based equations, the N-S can be written as:

$$\nabla \cdot \overrightarrow{u} = 0 \tag{1}$$

$$\frac{\partial \rho \overrightarrow{u}}{\partial t} + \nabla \cdot (\overrightarrow{m}\,\overrightarrow{u}) = -\nabla p + \mu \Delta \overrightarrow{u} \tag{2}$$

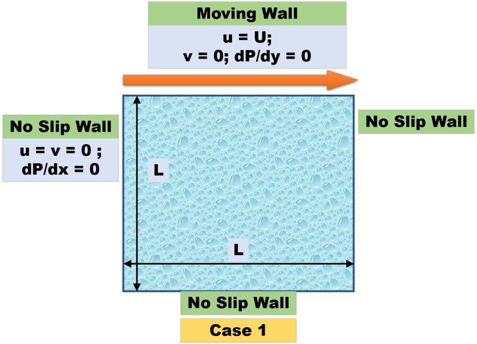
where, $\overrightarrow{m}$ & $\overrightarrow{u}$ represents mass-flux and velocity vector respectively
∇ is the divergence vector, given by

$$\nabla = \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k}$$

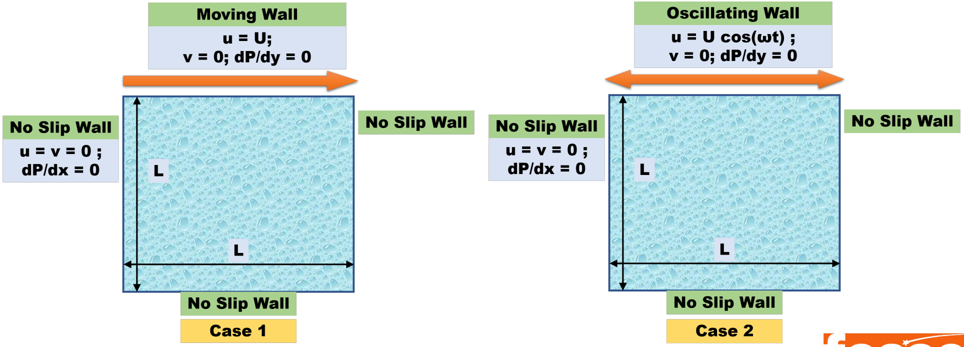
and Δ is the Laplacian, given by

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

- Before solving a problem, it is important to understand flow behavior so that correct algorithm is selected :
- Consider 2 variations for a lid-driven-cavity problem as given below

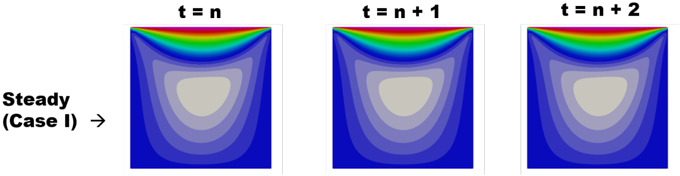


- Before solving a problem, it is important to understand flow behavior so that correct algorithm is selected :
- Consider 2 variations for a lid-driven-cavity problem as given below



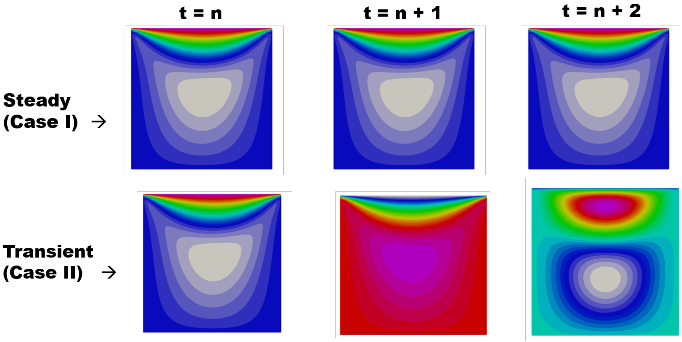
Differences in various flow conditions

- Let us see differences in the flow behavior in 2 cases presented previously:



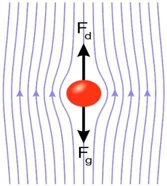
Differences in various flow conditions

- Let us see differences in the flow behavior in 2 cases presented previously:

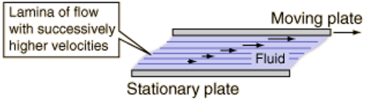


Differences in various flow conditions

Laminar →



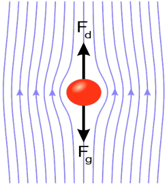
By Kraaiennest, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4115921



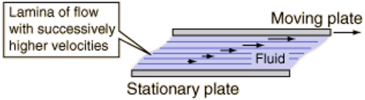
By R NaveGeorgia State University - HyperPhysicshhttp://hyperphysics.phy-astr.gsu.edu/hbase/pfric.html

Differences in various flow conditions

Laminar →

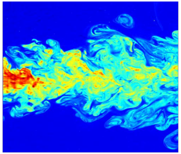


By Kraaiennest, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4115921



By R NaveGeorgia State University - HyperPhysicshhttp://hyperphysics.phy-astr.gsu.edu/hbase/pfric.html

Turbulent →



https://commons.wikimedia.org/w/index.php?curid=3082535

https://commons.wikimedia.org/w/index.php?curid=494937

- Let us look at important Incompressible flow solvers in OpenFOAM and their capabilities :

	Steady/Transient	Laminar/Turbulent	Viscosity
icoFoam	Transient	Laminar	Newtonian

- Let us look at important Incompressible flow solvers in OpenFOAM and their capabilities :

	Steady/Transient	Laminar/Turbulent	Viscosity
icoFoam	Transient	Laminar	Newtonian
nonNewtonianIcoFoam	Transient	Laminar	non-Newtonian

- Let us look at important Incompressible flow solvers in OpenFOAM and their capabilities :

	Steady/Transient	Laminar/Turbulent	Viscosity
icoFoam	Transient	Laminar	Newtonian
nonNewtonianIcoFoam	Transient	Laminar	non-Newtonian
simpleFoam	Steady	Turbulent	non-Newtonian

- Let us look at important Incompressible flow solvers in OpenFOAM and their capabilities :

	Steady/Transient	Laminar/Turbulent	Viscosity
icoFoam	Transient	Laminar	Newtonian
nonNewtonianIcoFoam	Transient	Laminar	non-Newtonian
simpleFoam	Steady	Turbulent	non-Newtonian
pisoFoam	Transient	Turbulent	non-Newtonian

- Let us look at important Incompressible flow solvers in OpenFOAM and their capabilities :

	Steady/Transient	Laminar/Turbulent	Viscosity
icoFoam	Transient	Laminar	Newtonian
nonNewtonianIcoFoam	Transient	Laminar	non-Newtonian
simpleFoam	Steady	Turbulent	non-Newtonian
pisoFoam	Transient	Turbulent	non-Newtonian

- What if problem is Laminar, Steady State ?

- Let us look at important Incompressible flow solvers in OpenFOAM and their capabilities :

	Steady/Transient	Laminar/Turbulent	Viscosity
icoFoam	Transient	Laminar	Newtonian
nonNewtonianIcoFoam	Transient	Laminar	non-Newtonian
simpleFoam	Steady	Turbulent	non-Newtonian
pisoFoam	Transient	Turbulent	non-Newtonian

- What if problem is Laminar, Steady State ?
- 2 options : (1) modify simpleFoam for Laminar flow (2) run icoFoam only (for longer time)

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/icoFoam

icoFoam Implementation

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/icoFoam
- Contents : 2 files & 1 folder :
  1. Make folder : compiling the solver
  2. createFields.c → variable declaration sections
  3. icoFoam.c → definitions of equations to be solved

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/icoFoam
- Contents : 2 files & 1 folder :
  1. Make folder : compiling the solver
  2. createFields.c → variable declaration sections
  3. icoFoam.c → definitions of equations to be solved

$$\boldsymbol{UEqn} = \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla} \cdot \left(\vec{\boldsymbol{U}}\vec{\boldsymbol{U}}\right) - \boldsymbol{\nu}\boldsymbol{\nabla}^2\vec{\boldsymbol{U}}$$

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nu, U)
);
```

- Location of the solver to check implementation : /opt/openfoam7/applications/solvers/incompressible/icoFoam
- Contents : 2 files & 1 folder :
  1. Make folder : compiling the solver
  2. createFields.c → variable declaration sections
  3. icoFoam.c → definitions of equations to be solved

$$\boldsymbol{UEqn = -\nabla p}$$

(Obtain U) →

```
if (piso.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p));
}
```

- Location of the solver to check implementation : /opt/openfoam7/applications/solvers/incompressible/icoFoam
- Contents : 2 files & 1 folder :
  1. Make folder : compiling the solver
  2. createFields.c → variable declaration sections
  3. icoFoam.c → definitions of equations to be solved

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*$$

```
fvScalarMatrix pEqn
(
    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
);
```

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/simpleFoam

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/simpleFoam

- Location of the solver to check implementation : /opt/openfoam7/applications/solvers/incompressible/simpleFoam

$$\boldsymbol{UEqn} : \boldsymbol{\nabla} \cdot \left(\overrightarrow{\boldsymbol{U}}\overrightarrow{\boldsymbol{U}}\right) + \boldsymbol{MRF} + \boldsymbol{u'u'} - \boldsymbol{\nu}\boldsymbol{\nabla}^2\overrightarrow{\boldsymbol{U}} = \boldsymbol{S}$$

```
tmp<fvVectorMatrix> tUEqn
(
    fvm::div(phi, U)
  + MRF.DDt(U)
  + turbulence->divDevReff(U)
 ==
    fvOptions(U)
);
fvVectorMatrix& UEqn = tUEqn.ref();
```

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/simpleFoam

$$UEqn = -\nabla p$$

(Obtain U)

$\Rightarrow$

```
if (simple.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p));

    fvOptions.correct(U);
}
```

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/simpleFoam

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*$$

$\Rightarrow$

```
fvScalarMatrix pEqn
(
    fvm::laplacian(rAtU(), p) == fvc::div(phiHbyA)
);
```

simpleFoam Implementation

- Location of the solver to check implementation :
  /opt/openfoam7/applications/solvers/incompressible/simpleFoam

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*$$

```
fvScalarMatrix pEqn
(
    fvm::laplacian(rAtU(), p) == fvc::div(phiHbyA)
);
```

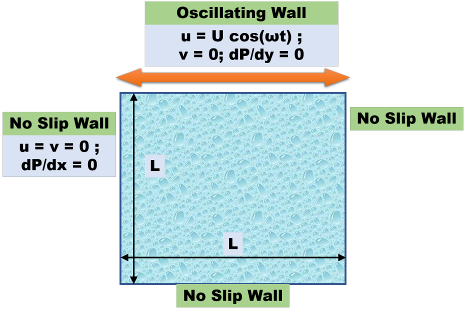
- To modify a solver : (1) make a copy of existing solver (2) change code (UEqn mostly) (3) compile using ‘wmake’ command

Oscillating LDC Problem Statement

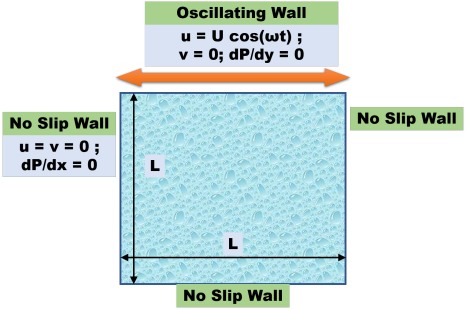
- Let us consider a simple oscillating Lid-Driven cavity problem [2].

Oscillating LDC Problem Statement

- Let us consider a simple oscillating Lid-Driven cavity problem [2].
- The domain consists of oscillating wall at the top of fluid-filled cavity as shown in Figure.



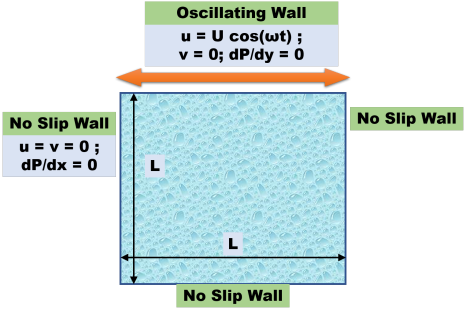
- Let us consider a simple oscillating Lid-Driven cavity problem [2].
- The domain consists of oscillating wall at the top of fluid-filled cavity as shown in Figure.



- Since the flow is periodic, icoFoam solver is used.

Oscillating LDC Problem Statement

- Let us consider a simple oscillating Lid-Driven cavity problem [2].
- The domain consists of oscillating wall at the top of fluid-filled cavity as shown in Figure.



- Since the flow is periodic, icoFoam solver is used.
- Domain Size $(L) = 1$
- Frequency $(\omega) = 2\pi/6$
- Maximum Velocity $(U) = 1$
- Grid Size : 100×100
- $Re = \frac{UL}{\nu} = 100$

Steps to Execute the Solver

- Go to the Folder: /opt/openfoam7/tutorials/incompressible/icoFoam/cavity

- Go to the Folder: /opt/openfoam7/tutorials/incompressible/icoFoam/cavity
- Copy ‘cavity’ tutorials to a local drive of your choice.

Steps to Execute the Solver

- Go to the Folder: /opt/openfoam7/tutorials/incompressible/icoFoam/cavity
- Copy 'cavity' tutorials to a local drive of your choice.
- We have to specify time-varying boundary conditions. open '0/U' file. Add the following in moving wall

Steps to Execute the Solver

- Go to the Folder: /opt/openfoam7/tutorials/incompressible/icoFoam/cavity
- Copy 'cavity' tutorials to a local drive of your choice.
- We have to specify time-varying boundary conditions. open '0/U' file. Add the following in moving wall

```
movingWall
{
    type            codedFixedValue;
    value           uniform (1.0 0 0);

    name    parabolicVelocity;
    code
    #{
            const vectorField& Cf = patch().Cf();
            const scalar t = this->db().time().value();

            vectorField& field = *this;
            const scalar Umax = 1.0;

            forAll(Cf, faceI)
            {
                    field[faceI] = vector( Umax*cos(2.0*3.142*t/6.0) , 0, 0);
            }
    #};
}
```

- In the 'system/blockMeshDict' file, change number of grid points as follows:

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (60 60 1) simpleGrading (1 1 1)
);
```

- In the 'system/blockMeshDict' file, change number of grid points as follows:

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (60 60 1) simpleGrading (1 1 1)
);
```

- In the 'system/controlDict' file, enter 'endTime' as 18.0 & 'delT' as 0.001.

- In the 'system/blockMeshDict' file, change number of grid points as follows:

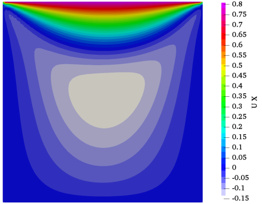
```
blocks
(
    hex (0 1 2 3 4 5 6 7) (60 60 1) simpleGrading (1 1 1)
);
```

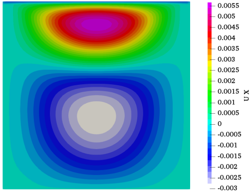
- In the 'system/controlDict' file, enter 'endTime' as 18.0 & 'delT' as 0.001.
- In the terminal, enter 'blockMesh' (to generate mesh) and then 'icoFoam' (to run the algorithm).

- In the 'system/blockMeshDict' file, change number of grid points as follows:

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (60 60 1) simpleGrading (1 1 1)
);
```

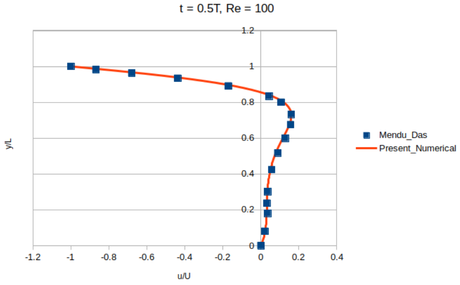
- In the 'system/controlDict' file, enter 'endTime' as 18.0 & 'delT' as 0.001.
- In the terminal, enter 'blockMesh' (to generate mesh) and then 'icoFoam' (to run the algorithm).
- The contours at t/T = 0.1, 0.25 look as follows :

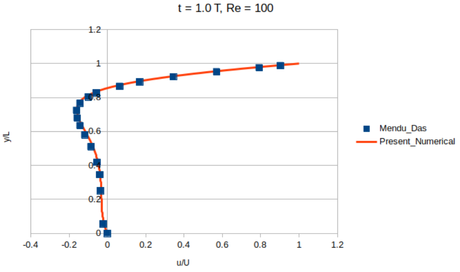




Validation of Results

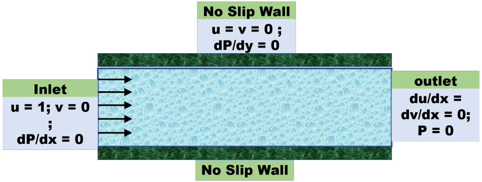
- In order to check whether correct results are obtained or not, X-velocity along vertical centerline at X = 0.5 is compared with literature [2] at different time-instants.



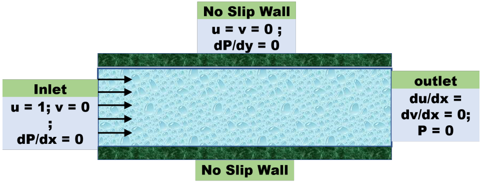


Problems to Try Out ... !

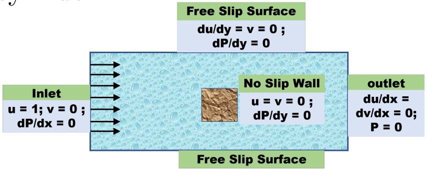
- To understand the implementation of grid generation and boundary conditions, following examples can be tried out:
- Flow inside a Channel:



- To understand the implementation of grid generation and boundary conditions, following examples can be tried out:
- Flow inside a Channel:



- Flow across square cylinder:



1. Vector Format of Conservation Equations

1. Vector Format of Conservation Equations
2. Steady & Unsteady State, Laminar & Turbulent flow

Summary

1. Vector Format of Conservation Equations
2. Steady & Unsteady State, Laminar & Turbulent flow
3. Comparison of Algorithms

Summary

1. Vector Format of Conservation Equations
2. Steady & Unsteady State, Laminar & Turbulent flow
3. Comparison of Algorithms
4. OpenFOAM implementation of Oscillating LDC test-case

Summary

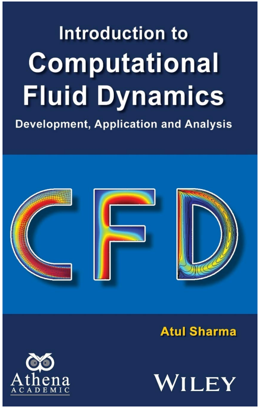
1. Vector Format of Conservation Equations
2. Steady & Unsteady State, Laminar & Turbulent flow
3. Comparison of Algorithms
4. OpenFOAM implementation of Oscillating LDC test-case
5. Sample problems to try

1. Vector Format of Conservation Equations
2. Steady & Unsteady State, Laminar & Turbulent flow
3. Comparison of Algorithms
4. OpenFOAM implementation of Oscillating LDC test-case
5. Sample problems to try

In the next lecture, we shall look into the problems involving complex geometry.

References

1. Sharma, A. (2016). Introduction to computational fluid dynamics: development, application and analysis. John Wiley & Sons.
2. Mendu, S. S., & Das, P. K. (2013). Fluid flow in a cavity driven by an oscillating lid—A simulation by lattice Boltzmann method. European Journal of Mechanics-B/Fluids, 39, 59-70.
3. https://www.openfoam.com/



Thank you for listening!

Sumant R Morab