# Computational Fluid Dynamics on HPC

Feng Chen
HPC User Services
LSU HPC & LONI
sys-help@loni.org

Louisiana State University
Baton Rouge
November 05, 2014

# Some CFD codes

ANSYS® FLUENT®

CD-adapco

OpenFOAM

SU2
The Open-Source CFD Code

# Things to be covered today

- Introduction to OpenFOAM
- Pre-processing OpenFOAM cases
- OpenFOAM case configuration
- Running OpenFOAM case
- Post-processing OpenFOAM cases
- Create your own solver/Develop with OpenFOAM

# Introduction to OpenFOAM

- Open **Field of Operation And Manipulation** (FOAM)
- **Free, open source CFD software package**
- **C++ programming language**
- **A set of libraries for continuum mechanics**
- **Based on Finite Volume Method (FVM)**

# Why consider OpenFOAM

- **Open architecture—will be detailed later**
- **Low(Zero)-cost CFD**
- **Problem-independent numerics and discretization**
- **Efficient environment for complex physics problems**

# OpenFOAM features overview

- **Physical Modeling Capability:**
  - Basic: Laplace, potential flow, passive scalar/vector/tensor transport
  - Incompressible and compressible flow: segregated pressure-based algorithms
  - Heat transfer: buoyancy-driven flows, conjugate heat transfer
  - Multiphase: Euler-Euler, VOF free surface capturing and surface tracking
  - Pre-mixed and Diesel combustion, spray and in-cylinder flows
  - Stress analysis, fluid-structure interaction, electromagnetics, MHD, etc.

LSU INFORMATION TECHNOLOGY SERVICES

# OpenFOAM features overview

- **Straightforward representation of partial differential equations (PDEs):**

$$\frac{\partial \rho U}{\partial t} + \nabla \bullet \rho U U - \nabla \bullet \mu \nabla U = -\nabla p$$

```
solve
    (
        fvm::ddt(rho, U)
      + fvm::div(phi, U)
      - fvm::laplacian(mu, U)
        ==
      - fvc::grad(p)
    );
```

# Introduction to OpenFOAM

- **History of OpenFOAM:**
  - Original development started in the late 1980s at Imperial College, London (FORTRAN)
  - Later changed to C++
  - OpenFOAM 1.0 released on 10/12/2004
  - Major releases: 1.4, 1.5, 1.6, 1.7.x, 2.0.x, 2.1.x, 2.2.x, ***2.3.x***
  - Wikki Ltd. Extend-Project: 1.4-dev, 1.5-dev, ***1.6-ext***

# Introduction to OpenFOAM

- **Theoretical background**
  - **Finite Volume Method (FVM)**
  - Unstructed grid
  - Pressure correction methods (SIMPLE, PISO, and their combination PIMPLE), for more information about FVM, see:

    - Partankar, S. V. (1980) *Numerical heat transfer and fluid flow*, McGraw-Hill.
    - H. Versteeg and W. Malalasekra, (2007) *An Introduction to Computational Fluid Dynamics: The Finite Volume Method Approach*
    - Ferziger, Joel H., Peric, Milovan, (2002) *Computational Methods for Fluid Dynamics*

# OpenFOAM toolbox overview

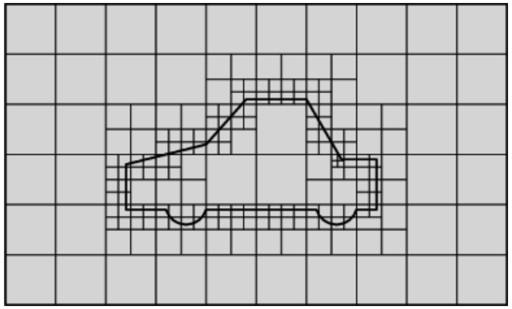
- **Applications:**
  - ***Utilities***: functional tools for pre- and post-processing, e.g. blockMesh, sampling tool
  - ***Solvers***: calculate the numerical solution of PDEs
- **Standard libraries**
  - ***General libraries***: those that provide general classes and associated functions;
  - ***Model libraries***: those that specify models used in computational continuum mechanics;

# OpenFOAM toolbox overview

- **Standard Solvers**
  - “Basic” CFD codes: e.g. laplacianFoam
  - Incompressible flow: e.g. icoFoam, simpleFoam
  - Compressible flow: e.g. rhoSimpleFoam, sonicFoam
  - Multiphase flow: e.g. interFoam
  - Direct numerical simulation (DNS) and large eddy simulation (LES)
  - Combustion
  - Particle-tracking flows (PIC)

# Mesh Generation

- **blockMesh**
  - For simple geometries, there is blockMesh, a multi-block mesh generator that generates meshes of hexahedra from a text configuration file.
  - Look at the OpenFOAM distribution files which contains numerous example configuration files for blockMesh to generate meshes for flows around simple geometries, e.g. a cylinder, a wedge, etc.
- **snappyHexMesh**
  - For complex geometries, meshes to surfaces from CAD
  - Can run in parallel
  - Automatic load balancing
- **Other mesh generation tools**
  - extrudeMesh
  - polyDualMesh



# Mesh Conversion

- **From: http://www.openfoam.org/features/mesh-conversion.php**

| **Part of the mesh converters** | |
|---|---|
| *ansysToFoam* | Converts an *ANSYS* input mesh file, exported from *I-DEAS*, to OPENFOAM® format |
| *cfx4ToFoam* | Converts a *CFX* 4 mesh to OPENFOAM® format |
| *datToFoam* | Reads in a datToFoam mesh file and outputs a points file. Used in conjunction with blockMesh |
| *fluent3DMeshToFoam* | Converts a *Fluent* mesh to OPENFOAM® format |
| *fluentMeshToFoam* | Converts a *Fluent* mesh to OPENFOAM® format including multiple region and region boundary handling |
| *foamMeshToFluent* | Writes out the OPENFOAM® mesh in *Fluent* mesh format |
| *foamToStarMesh* | Reads an OPENFOAM® mesh and writes a *PROSTAR* (v4) bnd/cel/vrt format |
| *foamToSurface* | Reads an OPENFOAM® mesh and writes the boundaries in a surface format |
| *gambitToFoam* | Converts a *GAMBIT* mesh to OPENFOAM® format |
| *gmshToFoam* | Reads .msh file as written by Gmsh |
| See http://www.openfoam.org/features/mesh-conversion.php for complete list | |

# Changes to your .soft file

- **Add the following keys to ~/.soft and then `resoft`**
  - On Super Mike:

```
+Intel-13.0.0
+openmpi-1.6.3-Intel-13.0.0
+OpenFOAM-2.2.1-Intel-13.0-openmpi-1.6.3
```

  - On Eric:

```
+gcc-4.7.0
+openmpi-1.6.3-gcc-4.7.0
+OpenFOAM-2.2.2-gcc-4.7.0-openmpi-1.6.3
```

  - On SuperMIC or QB2:

```
module load openfoam/2.3.0/INTEL-140-MVAPICH2-2.0
```

- **Start an interactive session:**

```
qsub -I -l nodes=1:ppn=16,walltime=02:00:00 –A your_allocation_name
```

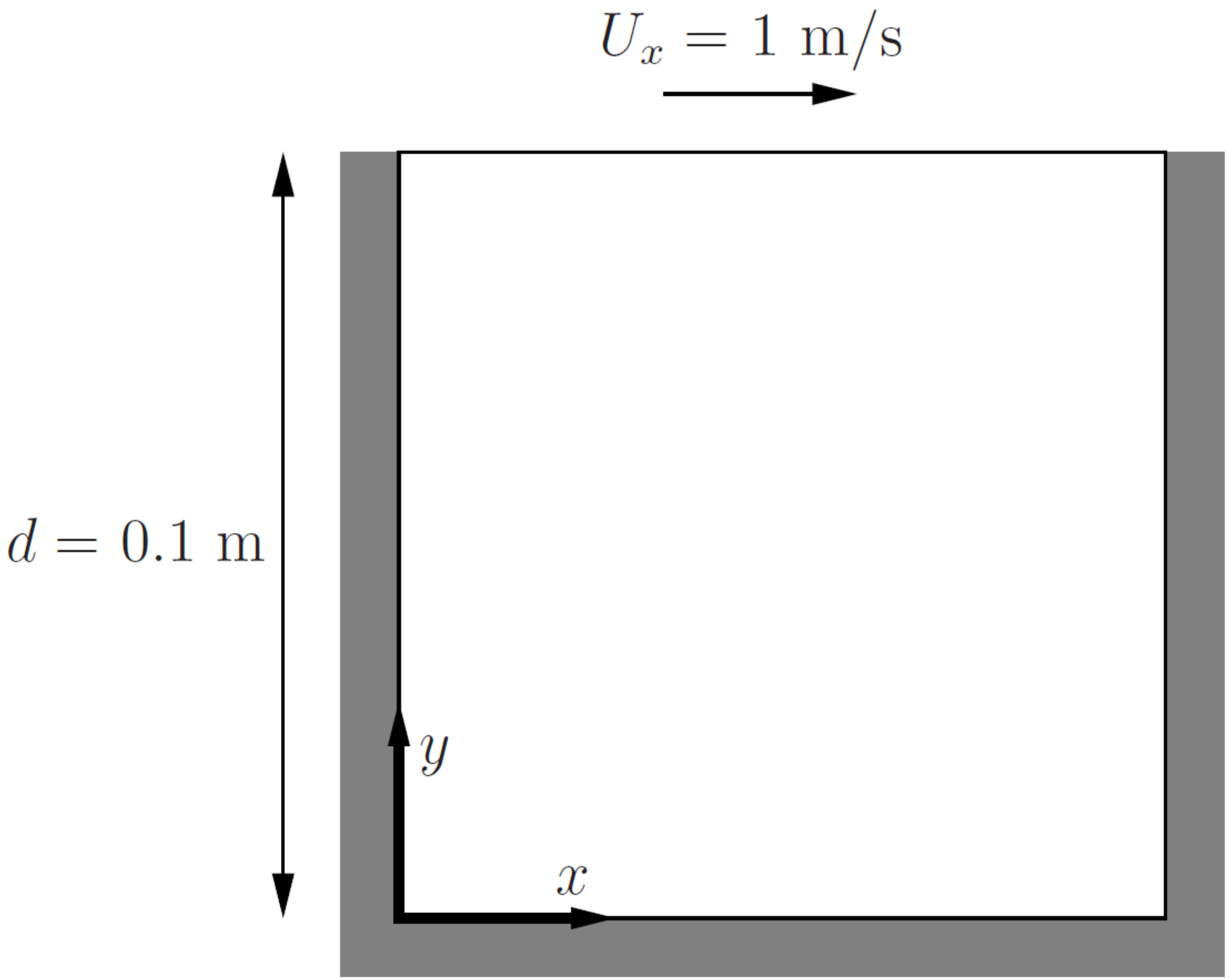
# Run First OpenFOAM case

- **Steps of running first OF case on Mike:**

```
$ mkdir -p /work/$USER/foam_run
$ cd /work/$USER/foam_run
$ wget
  https://tigerbytes2.lsu.edu/users/hpctraining/web/Downloads/intro_of.tar.gz
$ tar zxf intro_of.tar.gz
$ cd /work/$USER/foam_run/intro_of/cavity
$ blockMesh (generate mesh information)
$ icoFoam (running the PISO solver)
$ foamToVTK (convert to VTK format, optional)
$ paraFoam (post-processing)
```

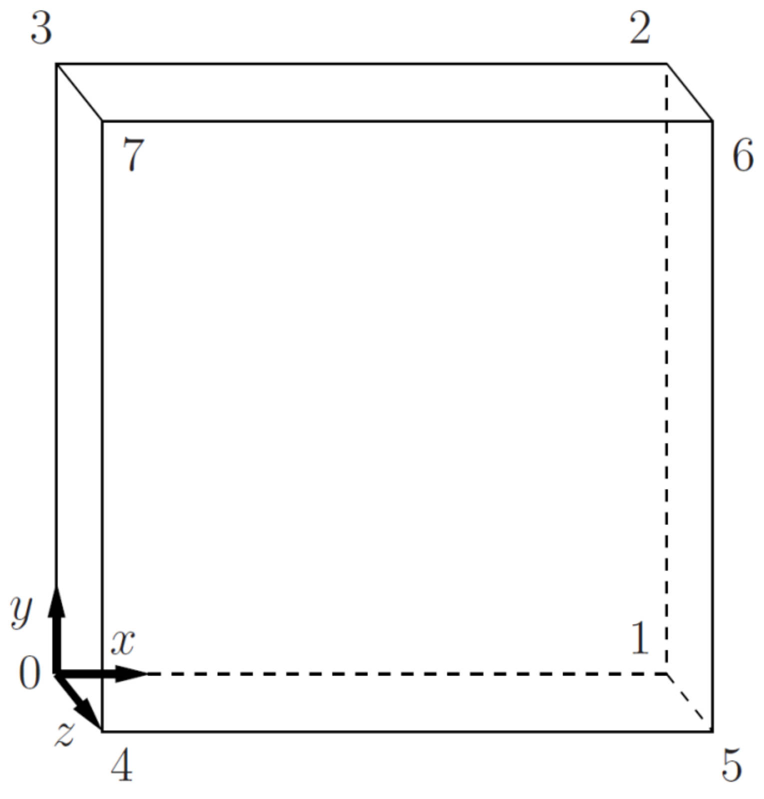
# Run First OpenFOAM case

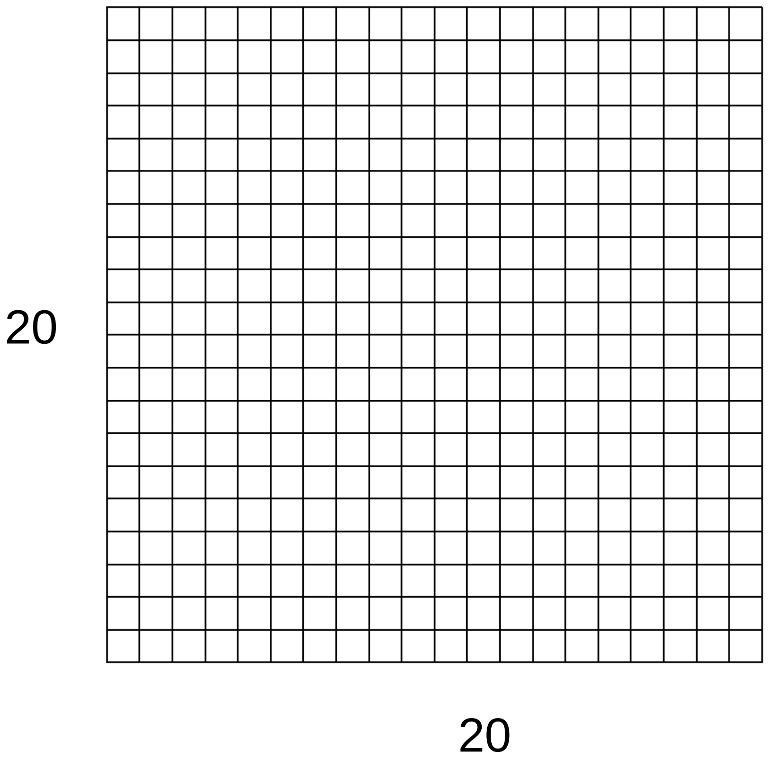
- **Lid-driven cavity flow using icoFoam**



# Lid-driven cavity flow

- **The cavity domain consists of a square of side length *d=0.1m* in the *x-y* plane. A uniform mesh of 20x20 cells will be used initially.**

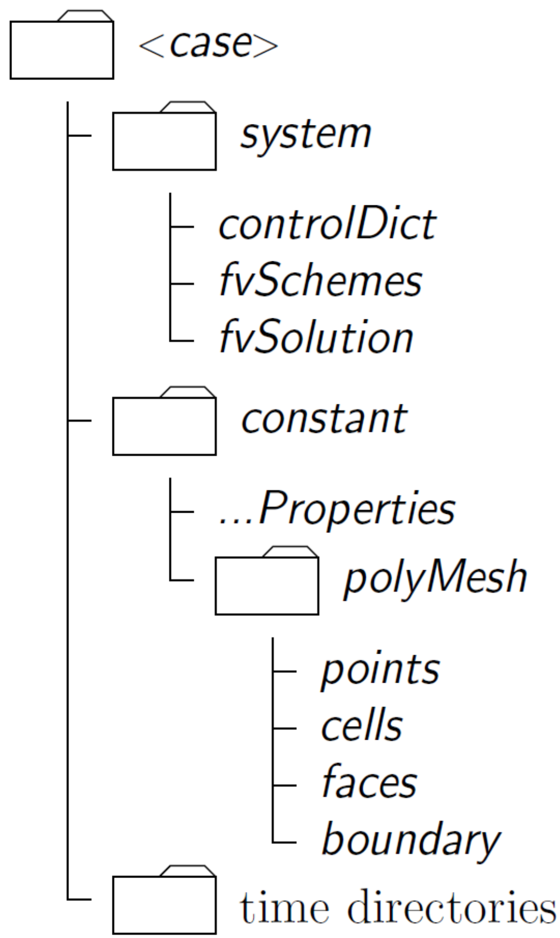


# Inside case configuration

- **File structure of OpenFOAM cases**

  $ ls -R $FOAM_RUN/tutorials/incompressible/icoFoam/cavity



# Inside case configuration

- **The minimum set of files required to run an OpenFOAM case**
  - constant directory:
    - description of the case mesh (geometry): e.g. ***polyMesh***
    - physical properties files: e.g. ***transportProperties***
  - system directory: solution procedure settings
    - ***controlDict***
    - ***fvSchemes***
    - ***fvSolution***
  - "time" directories: ***U, p***
    - initial conditions (I.C.)
    - boundary conditions (B.C.)
    - Future result files (typically determined by controlDict)

# Inside case configuration

- **constant directory:**
  - polyMesh
    - ***blockMeshDict: mesh description, will be detailed later***
    - boundary: list of patches with BCs definition
    - faces: list of mesh faces (list of points)
    - neighbour: list of neighboring cell labels
    - owner: list of owning cell labels
    - points: list of mesh points with their coordinates
  - transportProperties

# Edit blockMeshDict file (0)

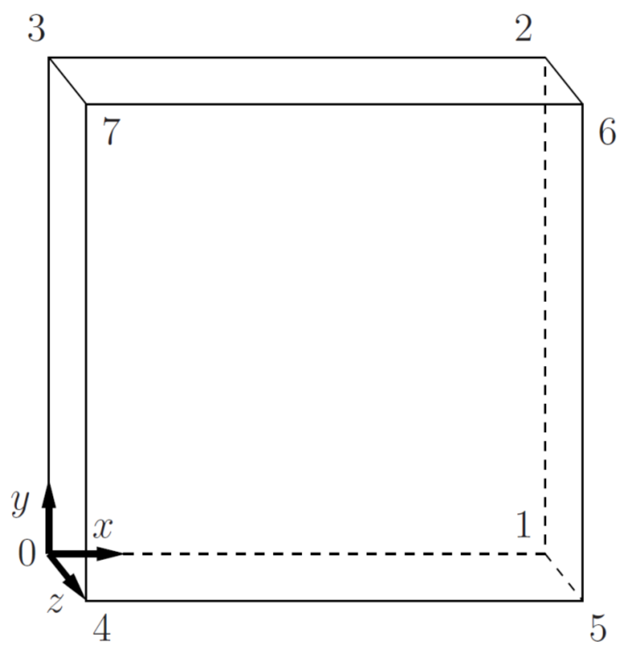
- **OpenFOAM file header:**

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
```

# Edit blockMeshDict file (1)

```
convertToMeters 0.1;

vertices
(
     (0 0 0)   //0
     (1 0 0)   //1
     (1 1 0)   //2
     (0 1 0)   //3
     (0 0 0.1) //4
     (1 0 0.1) //5
     (1 1 0.1) //6
     (0 1 0.1) //7
);
```



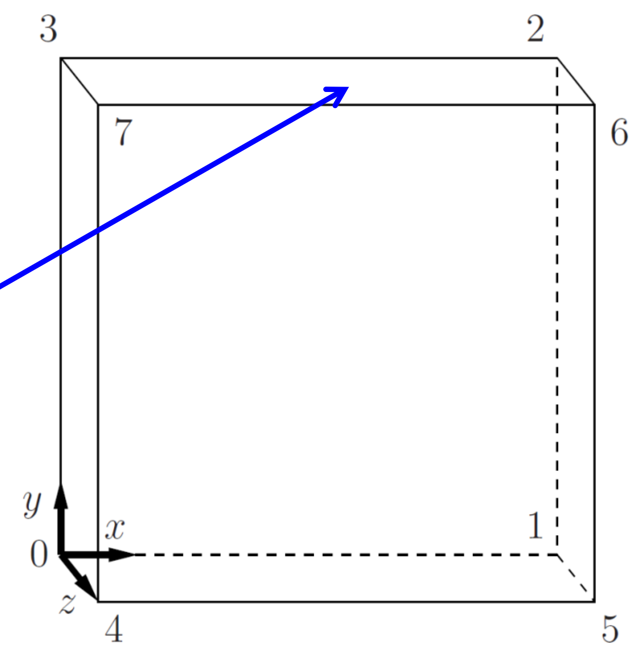
# Edit blockMeshDict file (2)

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

edges
(
);
```

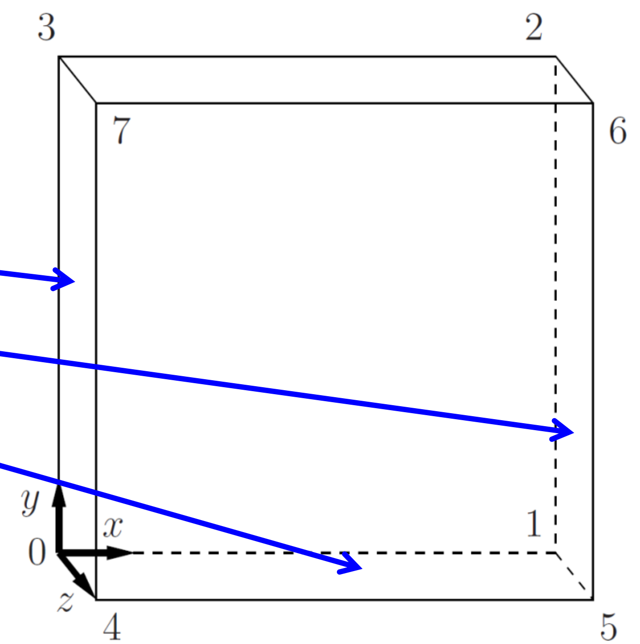
# Edit blockMeshDict file (3)

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
```



# Edit blockMeshDict file (4)

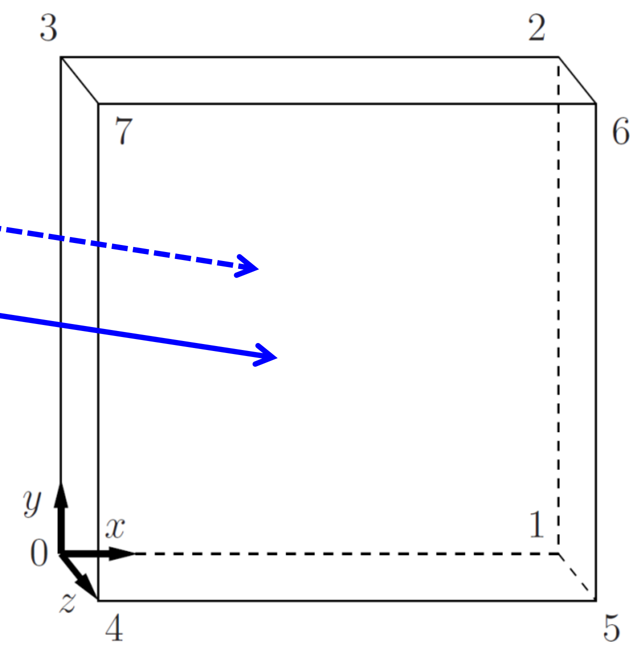
```
fixedWalls
{
    type wall;
    faces
    (
        (0 4 7 3)
        (2 6 5 1)
        (1 5 4 0)
    );
}
```



# Edit blockMeshDict file (5)

```
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
}
);

mergePatchPairs
(
);
```



# Solver settings

- **constant directory also contains:**
  - File which defines physical/material properties, transportProperties
  - Files which define some mesh properties, e.g. dynamicMeshDict
  - Files which defines turbulent properties RASProperties

# Solver settings

- **dimensions/units in OpenFOAM**
  - Representation of SI system
    - //dimensions [kg m sec K mol A cd ];
    - dimensions [0 2 -1 0 0 0 0];
- **Note: for incompressible solvers it is not needed to specify density. Pressure is then represented as *p/ρ***
- **transportProperties-representation of SI system**

```
transportModel  Newtonian; //viscosity options: newtonian/non-newtonian
nu nu [ 0 2 -1 0 0 0 0 ] 0.01;  // kinematic viscosity
```

# Selected codes in icoFoam

```
while (runTime.loop())
{
    Info<< "Time = " << runTime.timeName() << nl << endl;
    #include "readPISOControls.H"
    #include "CourantNo.H"
    fvVectorMatrix UEqn // Note the equation representation
    (
        fvm::ddt(U)
      + fvm::div(phi, U)
      - fvm::laplacian(nu, U)
    );
    solve(UEqn == -fvc::grad(p));
    // --- PISO loop
    for (int corr=0; corr<nCorr; corr++)
    {
        volScalarField rAU(1.0/UEqn.A());
    ...
    }
}
```

# Solver settings

- **system directory contains:**
  - Files concerning solver parameters, as well as definition files for utility tools e.g. decomposeParDict
    - controlDict – simulation control and parameters, additional libraries to load and extra functions
    - fvSchemes – definition of discretization schemes
    - fvSolution – definitions of solver type, tolerances, relaxation factors

# Solver settings-controlDict

- **controlDict: (basic time step control, how your results are written, etc.)**

```
application       icoFoam;
startFrom         startTime;
startTime         0;
stopAt            endTime;
endTime           0.5;
deltaT            0.005;
writeControl      timeStep;
writeInterval     20;
purgeWrite        0;
writeFormat       ascii;
writePrecision    6;
writeCompression  off;
timeFormat        general;
timePrecision     6;
runTimeModifiable true;
```

# Solver settings-fvSchemes

- **fvSchemes:**

```
// time schemes (Euler , CrankNicholson,
backward, steadyState )
ddtSchemes
{
    default         Euler;
}
// gradient schemes (Gauss , leastSquares,
fourth, cellLimited, faceLimited )
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}
// convection and divergence schemes (
interpolation schemes used: linear,
skewLinear,  cubicCorrected, upwind,
linearUpwind, QUICK, TVD, SFCD, NVD)
divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}
```

```
laplacianSchemes
{
    default         none;
    laplacian(nu,U) Gauss linear orthogonal;
    laplacian((1|A(U)),p) Gauss linear
orthogonal;
}
```

# Solver settings-fvSchemes

- **fvSchemes:**

```
// interpolation schemes to calculate
values on the faces (linear,
cubicCorrection, midPoint , upwind,
linearUpwind, skewLinear , QUICK, TVD,
limitedLinear , vanLeer , MUSCL,
limitedCubic, NVD, SFCD, Gamma )
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
// schemes for surface normal gradient on
the faces ( corrected, uncorrected,
limited, bounded, fourth )
snGradSchemes
{
    default         orthogonal;
}
// lists the fields for which the flux is
generated in the application
fluxRequired
{
    default         no;
    p;
}
```

# Solver settings-solution control

- **fvSolution:**

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }
}
// pressure - velocity coupling
// SIMPLE (Semi - Implicit Method for
Pressure - Linked Equations )
```

```
// PISO ( Pressure Implicit with Splitting
of Operators )
// PIMPLE ( Combination of SIMPLE and PISO
)
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell          0;
    pRefValue       0;
}
```

http://www.openfoam.org/docs/user/fvSolution.php

# Solver settings-time directory

- **Time directories contain field files (e.g. U, p, k, epsilon, omega, T etc.)**
- **Fields files store field solution values on all cells and boundary conditions on the computational domain**
- **0 time directory is initial directory containing field files with initial field values and boundary conditions**
- **Common parts for all field files are:**
  - header
  - dimensions
  - internalField
  - boundaryField

# Boundary Conditions (BCs)

- **base type (described purely in terms of geometry):**
  - patch, wall, empty, symmetry, cyclic
- **primitive type (base numerical patch condition assigned to a field variable on the patch):**
  - fixedValue, fixedGradient, zeroGradient, mixed, directionMixed, calculated
- **derived type (complex patch condition, derived from the primitive type, assigned to a field variable on the patch):**
  - inletOutlet

LSU INFORMATION TECHNOLOGY SERVICES

# Initial and Boundary conditions: Velocity

- **U**

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}
```

# Initial and Boundary conditions: Pressure

- **p**

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

# Running cavity in parallel

- **decomposePar //specify the parallel run params**
- **mpirun --hostfile <machines> -np <nProcs>**
  **<foamExec> <otherArgs> -parallel > log**
  - Examples on Mike:

    **mpirun --hostfile $PBS_NODEFILE -np 16 icoFoam -parallel > log**
  - Examples on Eric:

    **mpirun --hostfile $PBS_NODEFILE -np 8 icoFoam -parallel > log**

- **reconstructPar //merge time directories sets from each processor**
- **See:**

  /work/$USER/foam_run/intro_of/cavity_parallel_is

  /work/$USER/foam_run/intro_of/cavity_parallel

# Running cavity in parallel

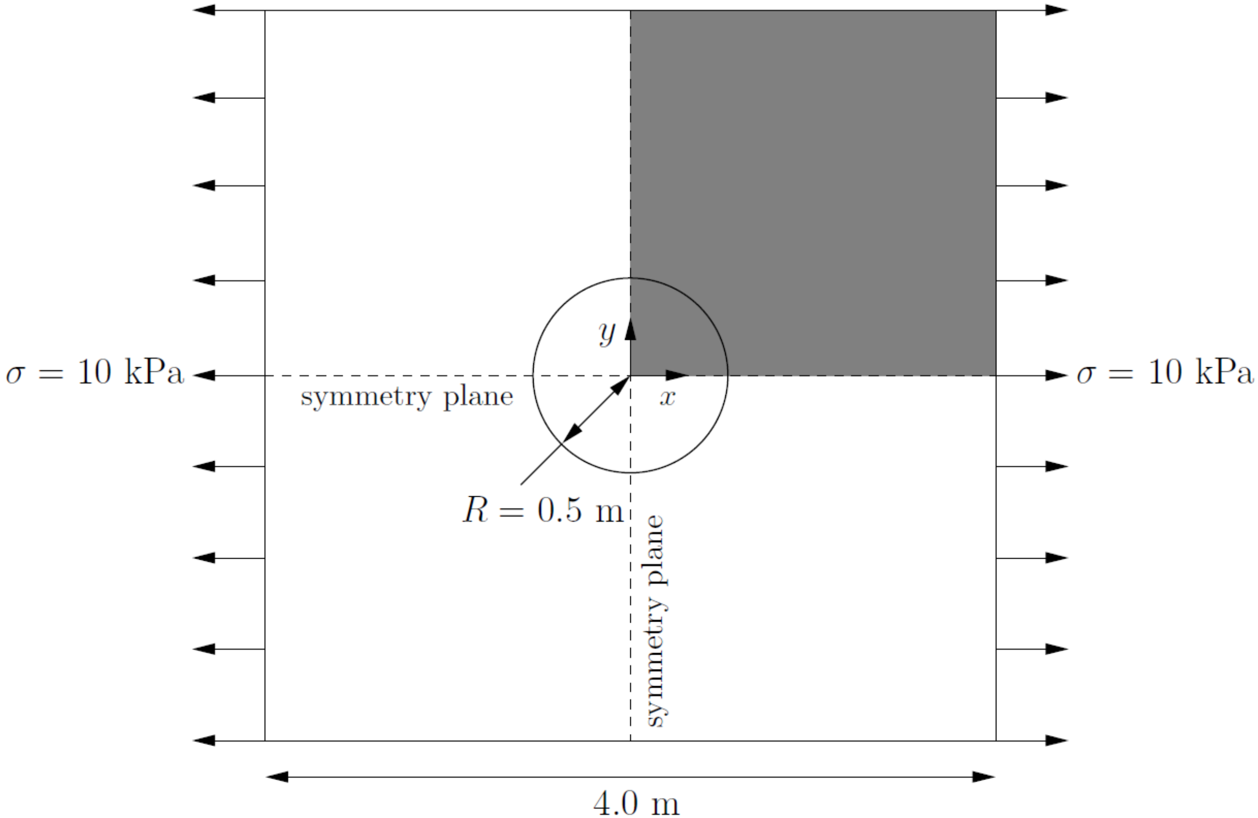
- **On interactive session:**

```
$ cd /work/$USER/foam_run/intro_of/cavity_parallel_is
$ blockMesh
$ vi system/decomposeParDict
$ decomposePar
$ mpirun --hostfile $PBS_NODEFILE -np 16 icoFoam -parallel
$ reconstructPar
```

- **Via batch mode:**

```
$ cd /work/$USER/foam_run/intro_of
$ ./cavity_parallel_run.sh
```

# Stress analysis of plateHole




**Analytical Solution:**

$$\left(\sigma_{xx}\right)_{x=0} = \begin{cases} \sigma\left(1+\dfrac{R^2}{2y^2}+\dfrac{3R^4}{2y^4}\right) & \text{for } |y| \geq R \\ 0 & \text{for } |y| < R \end{cases}$$

# Part of the solidDisplacementFoam code

```
do // loop for residual and iterations
        {
            if (thermalStress)
            {
                volScalarField& T = Tptr();
                solve
                (
                    fvm::ddt(T) == fvm::laplacian(DT, T)
                );
            }

            {
                fvVectorMatrix DEqn // not a N-S equation
                (
                    fvm::d2dt2(D)
                 ==
                    fvm::laplacian(2*mu + lambda, D, "laplacian(DD,D)")
                  + divSigmaExp
                );
...
} while (initialResidual > convergenceTolerance && ++iCorr < nCorr);
```

# Run stress analysis case

- **Steps of running plateHole on Mike:**

```
$ cd /work/$USER/foam_run/intro_of/plateHole
$ blockMesh #generate geometry
$ checkMesh #tool for checking the mesh quality
$ solidDisplacementFoam #running the stress analysis solver
$ foamToVTK #convert VTK format, optional
$ paraFoam  #post-processing
```

# Post-processing

- **Most used post-processing software for OpenFOAM data visualization is *Paraview***
- **paraFoam – script for automatic import of OpenFOAM results into Paraview**
- **OpenFOAM Data transformation in other formats: e.g. foamToVTK (also used by Paraview)**

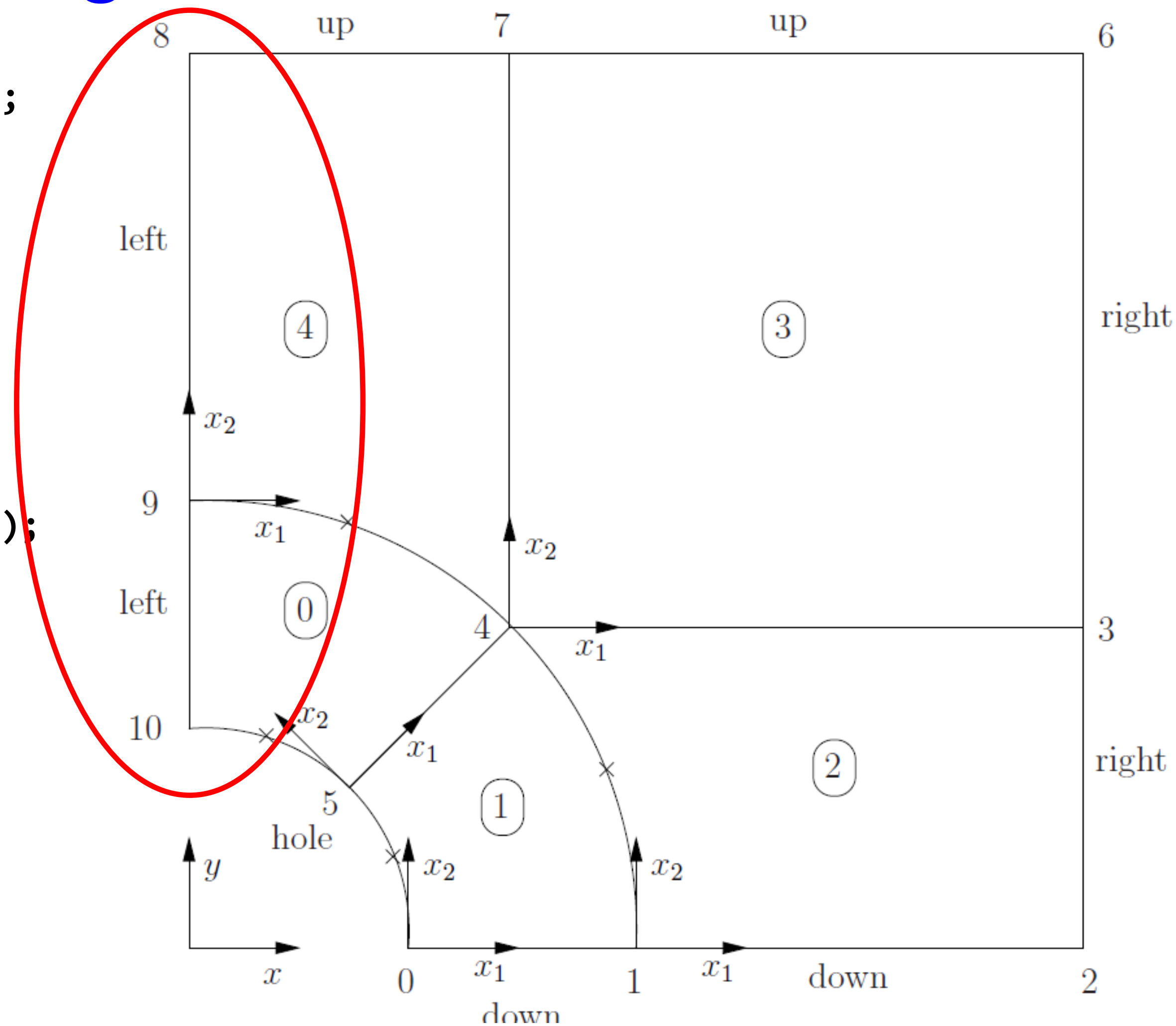
# Post-processing

- ***sample*** **– utility used for sampling**
- **Sample setups are defined in system/sampleDict**
- **Sample data are stored in the new (automatically) created subdirectory sets**
- **Example:**

```
$ cd /work/$USER/foam_run/intro_of/plateHole
$ foamCalc components sigma #calculates new fields from existing ones.
$ sample
```

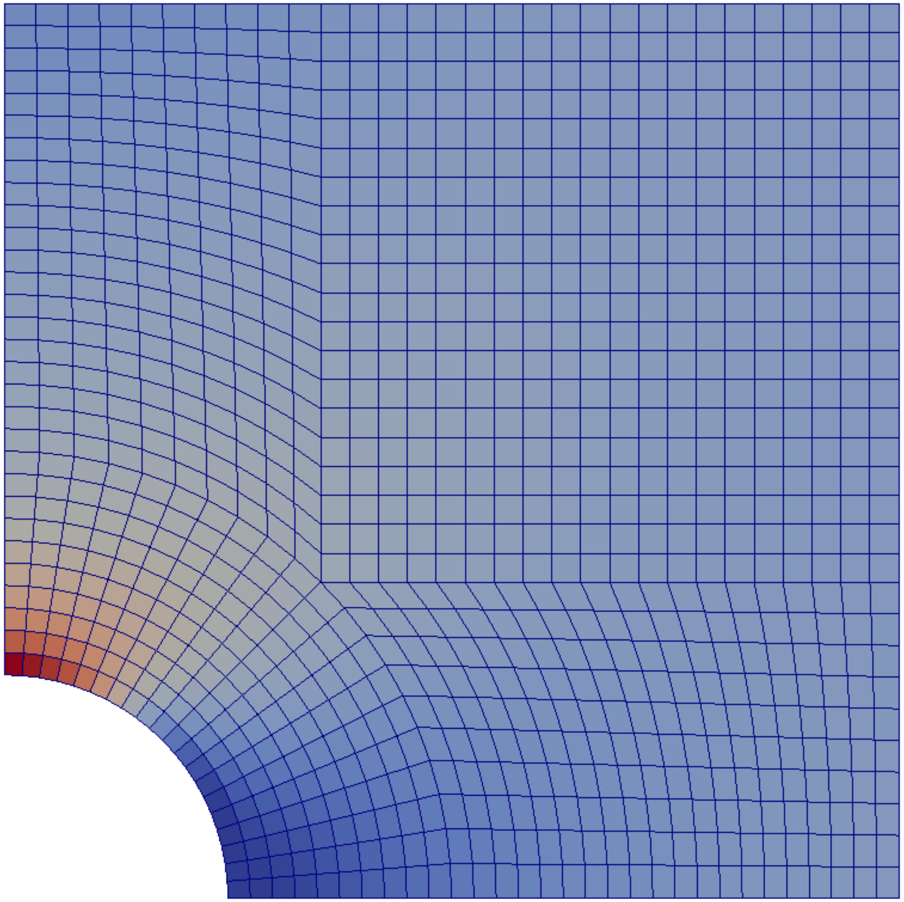
# sampleDict for the plateHole case along the left line

```
/*OpenFOAM file header*/
interpolationScheme cellPoint;
setFormat       raw;
sets
(
    leftPatch
    {
        type    uniform;
        axis    y;
        start   ( 0 0.5 0.25 );
        end     ( 0 2 0.25 );
        nPoints 100;
    }
);
fields          ( sigmaxx );
```



# sampleDict for the plateHole case for the entire surface

```
/*OpenFOAM file header*/
interpolationScheme cellPoint;
surfaceFormat   vtk;
surfaces
(
    sigmaxx
    {
        type plane;
        basePoint ( 0 0 0.25 );
        normalVector ( 0 0 1 );
    }
);
fields          ( sigmaxx );
```



# Exercise

- **Run the cavity case and sample:**
  1. along middle-y axis
  2. surface

# Develop your own solver

- **A simple commented overview of the icoFoam PISO solver, see [http://openfoamwiki.net/index.php/IcoFoam](http://openfoamwiki.net/index.php/IcoFoam)**

```
// from the last solution of velocity, extract the diag. term from the matrix and store the reciprocal
// note that the matrix coefficients are functions of U due to the non-linearity of convection.
            volScalarField rUA = 1.0/UEqn.A();

// take a Jacobi pass and update U.  See Hrv Jasak's thesis eqn. 3.137 and Henrik Rusche's thesis, eqn. 2.43
// UEqn.H is the right-hand side of the UEqn minus the product of (the off-diagonal terms and U).
// Note that since the pressure gradient is not included in the UEqn. above, this gives us U without
// the pressure gradient.  Also note that UEqn.H() is a function of U.

            U = rUA*UEqn.H();

// calculate the fluxes by dotting the interpolated velocity (to cell faces) with face normals
// The ddtPhiCorr term accounts for the divergence of the face velocity field by taking out the
//  difference between the interpolated velocity and the flux.
            phi = (fvc::interpolate(U) & mesh.Sf())
                + fvc::ddtPhiCorr(rUA, U, phi);
```

- **Create your own solver by copy (on SuperMikeII):**
  **`/usr/local/packages/OpenFOAM/2.2.1/Intel-13.0-openmpi-1.6.3/OpenFOAM-2.2.1/applications/solvers/incompressible/icoFoam`**

# Documentation and Help

- **OpenFOAM course: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD/**
- **OpenFOAM homepage: www.openfoam.com**
- **OpenFOAM User Guide, Programmers Guide, Doxygen**
- **OpenFOAM Wiki: www.openfoamwiki.net**
- **OpenFOAM-extend:**
  - http://sourceforge.net/projects/openfoam-extend/,
  - http://www.foam-extend.org
  - http://extend-project.de
- **OpenFOAM Forum: http://www.cfd-online.com/Forums/openfoam/**
- **OpenFOAM workshop: www.openfoamworkshop.org**
- **CoCoons project: http://www.cocoons-project.org/**

- **User forum:**
  - http://www.cfd-online.com/Forums/openfoam/

# Thank you for your attention!

# Any questions?