Introduction to OpenFOAM

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

Louisiana State University
Baton Rouge
February 24, 2016

Selected CFD codes

ANSYS® FLUENT®

CD-adapco

OpenFOAM

SU2
The Open-Source CFD Code

Topics to be covered today

- **OpenFOAM general overview**
- **Lid driven cavity example walk through**
  - Pre-processing OpenFOAM cases
  - OpenFOAM case configuration
  - Running OpenFOAM case
  - Post-processing OpenFOAM cases
- **Stress analysis of plateHole example**
- **Create custom solver**
  - Adding a transport equation to icoFoam

Introduction to OpenFOAM

OpenFOAM General Overview

OpenFOAM overview

- **Open Field of Operation And Manipulation (FOAM)**
- **Free, open source CFD software package**
  - The GNU Public License (GPL) gives freedom to contribute to any or all of these projects.
  - Open architecture—will be detailed later
  - Low(Zero)-cost CFD
  - Problem-independent numerics and discretization
  - Efficient environment for complex physics problems
- **C++ programming language**
- **A set of libraries for continuum mechanics**
- **Based on Finite Volume Method (FVM)**

OpenFOAM History Review

- **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: 1.4-1.7.x, 2.0.x, 2.1.x-2.4.x, ***v3.0.(1), v3.0+***
  - Wikki Ltd. Extend-Project: 1.4-dev, 1.5-dev, 1.6-ext, ***foam-extend-3.2***

OpenFOAM and OpenFOAM+

- **How does it compare to OpenFOAM and OpenFOAM+?**
  - The OpenFOAM project managed by the OpenFOAM Foundation is similar to RHEL.
  - The OpenFOAM+ project managed by OpenCFD Limited (ESI Group) is similar to Fedora.
- **How are OpenFOAM and OpenFOAM+ similar?**
  - OpenFOAM Foundation (http://www.openfoam.org/) is aiming to maintain versions that are meant to be of high quality standards, future-proof and easy to maintain.
  - OpenCFD Limited (http://www.openfoam.com/) has its own development cycle and provides new features, while keeping as close as possible to the same standards, as announced: OpenCFD is pleased to announce the release of OpenFOAM-v3.0+OpenFOAM Foundation: (OpenFOAM-v3.0+)
- **Which version/fork/variant to choose?**
  - Check the release notes of each one and compare with your needs.
  - Or simply try them all without spending a single dime in licenses??

OpenFOAM theoretical background

- **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 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.

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)
);
```

OpenFOAM toolbox overview

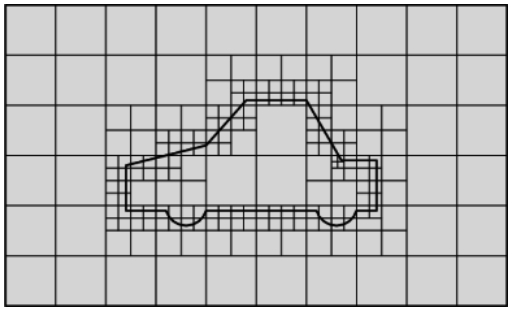
- **Applications:**
  - ***Utilities***: functional tools for pre- and post-processing, e.g. blockMesh, sampling tool
  - ***Solvers***: calculate the numerical solution of PDEs
- **Standard libraries**
  - ***General libraries***: those that provide general classes and associated functions;
  - ***Model libraries***: those that specify models used in computational continuum mechanics;
- **Use the following commands to check:**
  - `cd $FOAM_APPBIN && ls`
  - `cd $FOAM_LIBBIN && ls`

OpenFOAM toolbox overview

- **Standard Solvers**
  - "Basic" CFD codes: e.g. `laplacianFoam, potentialFoam`
  - 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): e.g. `DPMFoam, MPPICFoam`

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	

Introduction to OpenFOAM

Run OpenFOAM under HPC Environment

Changes to your .soft file

- **Add the following keys to ~/.soft and then `resoft`**
  - On SuperMike2:

```
+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:**

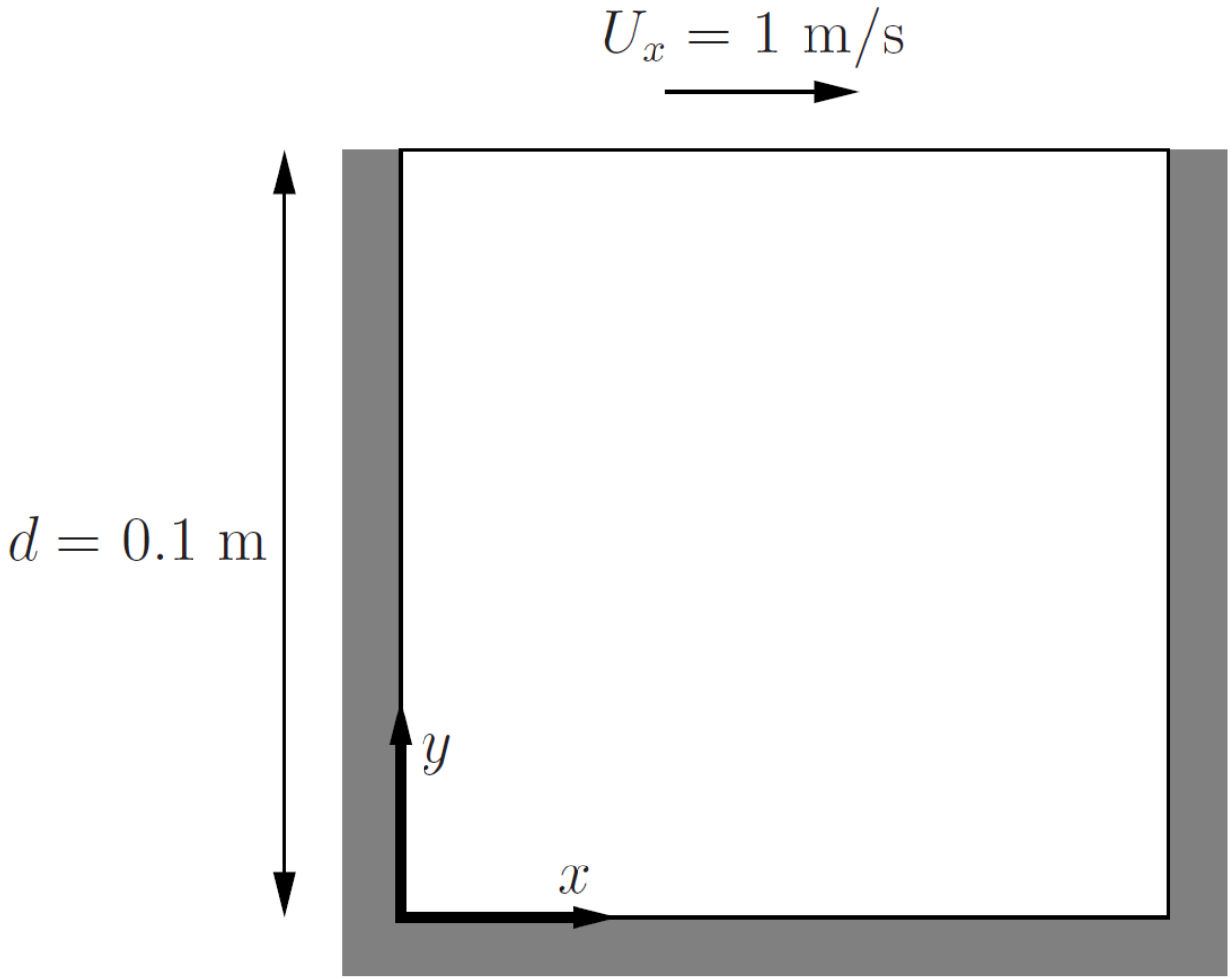
```
$  mkdir -p /work/$USER/foam_run
$  cd /work/$USER/foam_run
$  wget http://www.hpc.lsu.edu/training/weekly-materials/Downloads/intro_of.tar.gz
$  tar zxf intro_of.tar.gz
$  cd /work/$USER/foam_run/intro_of/cavity
$  blockMesh (generate mesh information)
$  icoFoam (running the PISO solver)
$  foamToVTK (convert to VTK format, optional)
```

Introduction to OpenFOAM

First case: Lid-driven cavity flow

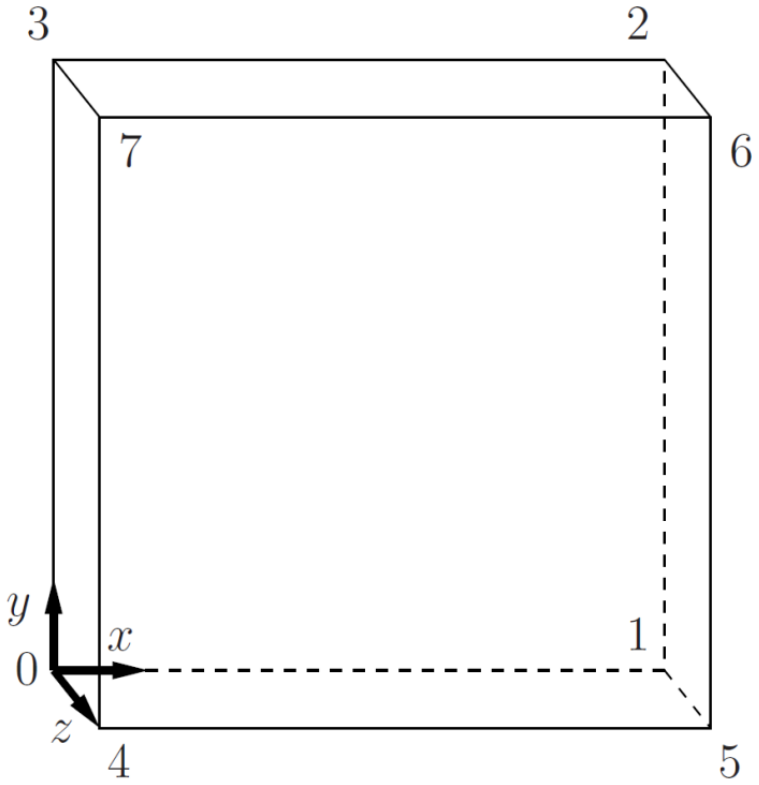
First OpenFOAM case overview

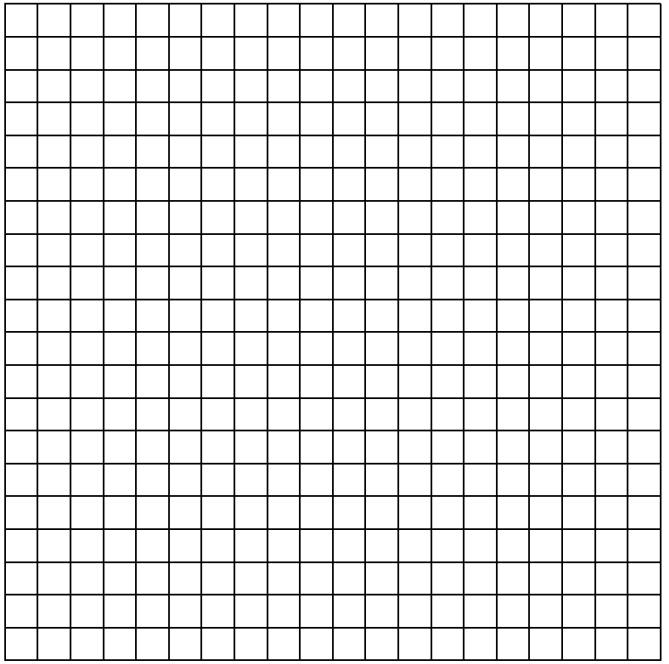
- **Lid-driven cavity flow using icoFoam**



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.**

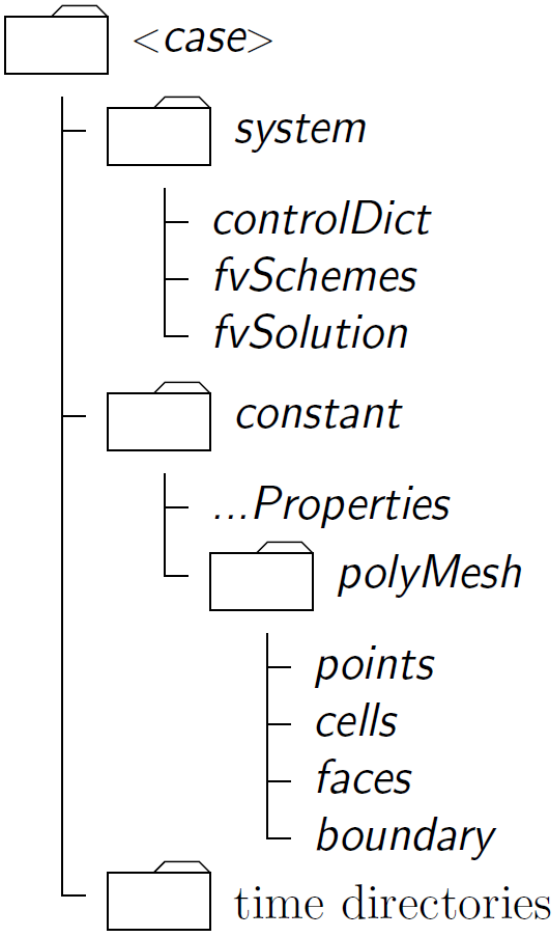




Inside case configuration

- **File structure of OpenFOAM cases**

```
$ ls -R /work/$USER/foam_run/intro_of/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 next few slides***
    - 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

  can be generated using blockMeshDict

  - transportProperties
    - Physical/material properties: e.g. viscosity

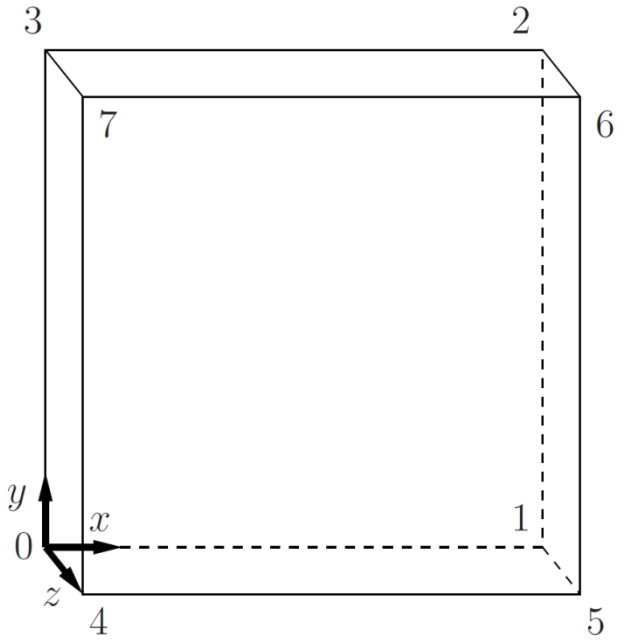
Edit blockMeshDict file (0)

- **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)

```
17  convertToMeters 0.1;
18
19  vertices
20  (
21      (0 0 0)    //0
22      (1 0 0)    //1
23      (1 1 0)    //2
24      (0 1 0)    //3
25      (0 0 0.1)  //4
26      (1 0 0.1)  //5
27      (1 1 0.1)  //6
28      (0 1 0.1)  //7
29  );
```

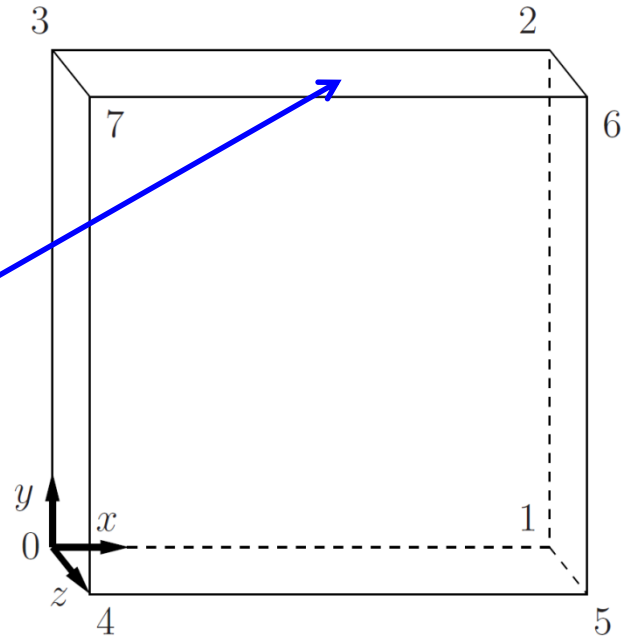


Edit blockMeshDict file (2)

```
31  blocks
32  (
33      hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
34  );
35
36  edges
37  (
38  );
```

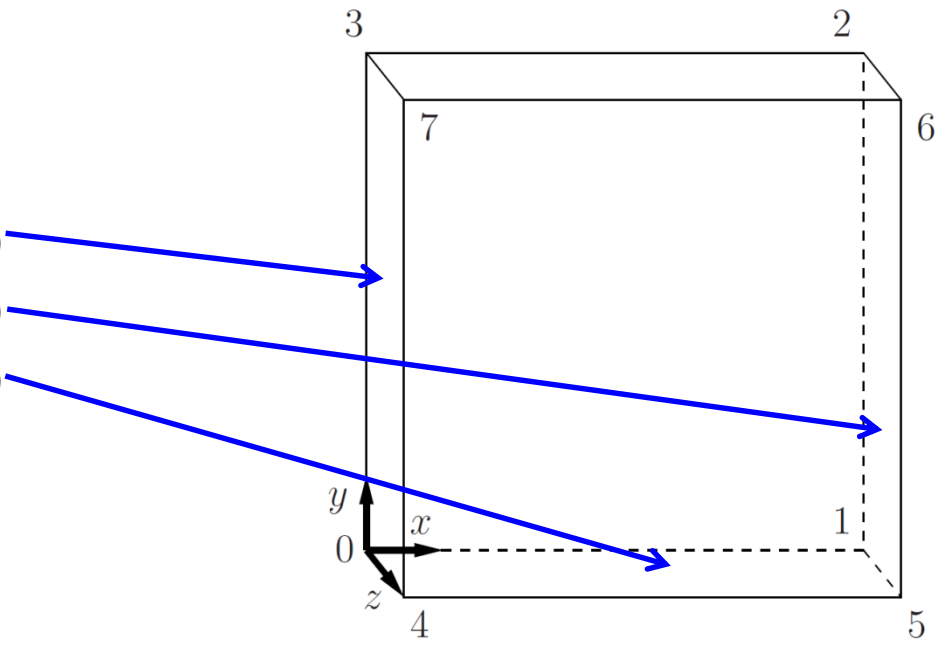
Edit blockMeshDict file (3)

```
40  boundary
41  (
42      movingWall
43      {
44          type wall;
45          faces
46          (
47              (3 7 6 2)
48          );
49      }
```



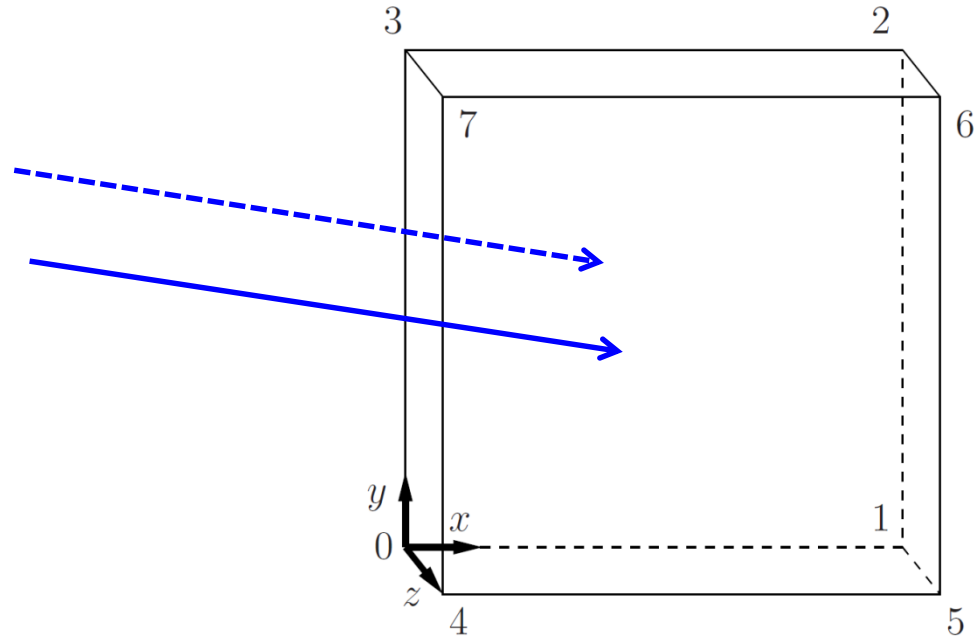
Edit blockMeshDict file (4)

```
50      fixedWalls
51      {
52          type wall;
53          faces
54          (
55              (0 4 7 3)
56              (2 6 5 1)
57              (1 5 4 0)
58          );
59      }
```



Edit blockMeshDict file (5)

```
53      frontAndBack
54      {
55          type empty;
56          faces
57          (
58              (0 3 2 1)
59              (4 5 6 7)
60          );
61  }
62  );
63
64  mergePatchPairs
65  (
66  );
```



Solver settings

- **constant directory may also contain:**
  - Files which define some mesh properties, e.g. `dynamicMeshDict`
  - Files which defines turbulent properties `RASProperties`

Solver settings – constant directory

- **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
```

Solver settings – system directory

- **system directory contains:**
  - Files concerning solver parameters, as well as definition files for utility tools e.g.
    - `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
    - `decomposeParDict` – for parallel domain decomposition

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-decomposeParDict

- **decomposeParDict: (Parameters for breaking up the geometry and fields for parallel processing)**

```
numberOfSubdomains 20;

method          simple;

simpleCoeffs
{
    n               ( 4 5 1 );
    delta           0.001;
}
```

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

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

- cd /work/fchen14/foam_run/intro_of/cavity_parallel_is
- 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:**

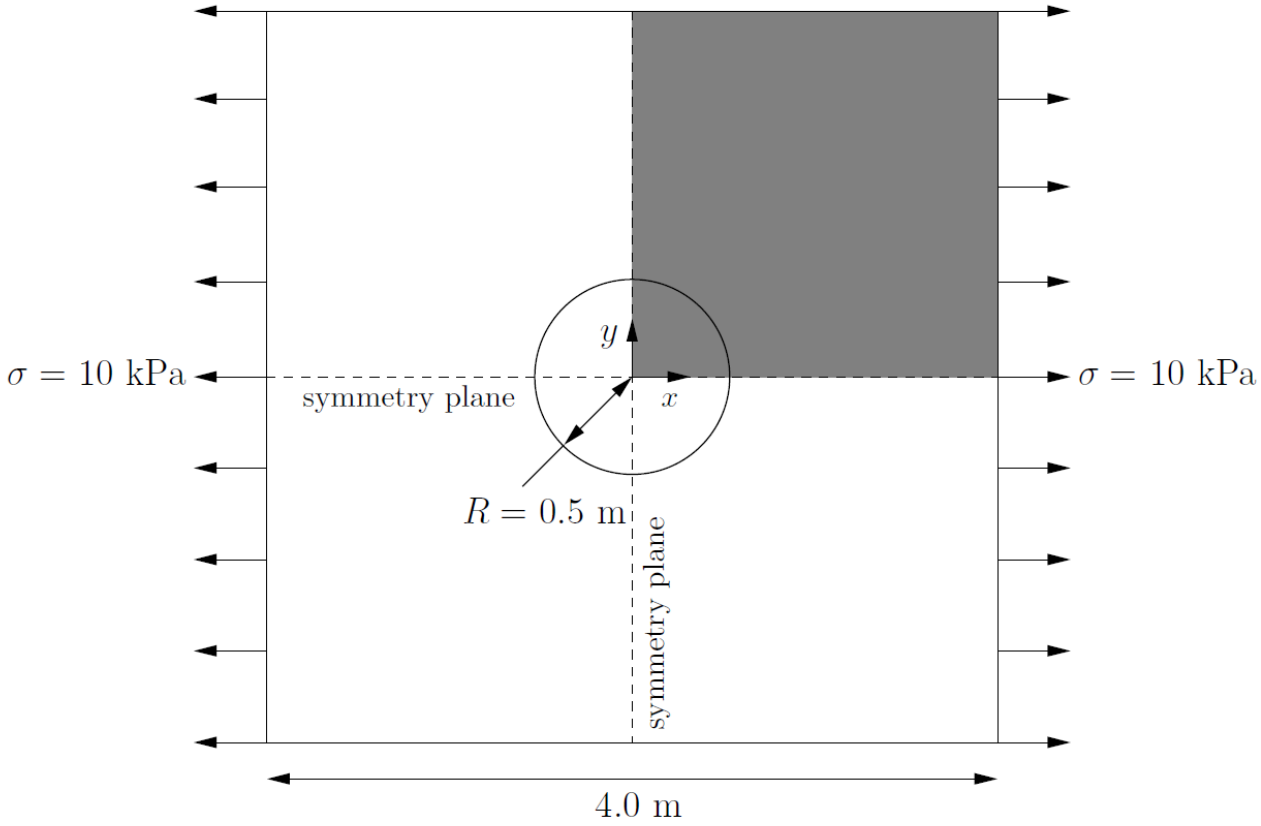
```
$ cd /work/$USER/foam_run/intro_of/cavity_parallel_is
$ blockMesh
$ vi system/decomposeParDict
$ decomposePar
$ mpirun --hostfile $PBS_NODEFILE -np 16 icoFoam -parallel
$ reconstructPar
```

- **Via batch mode:**

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

Second case: stress analysis of plateHole

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 (Results can also be used by Paraview)**

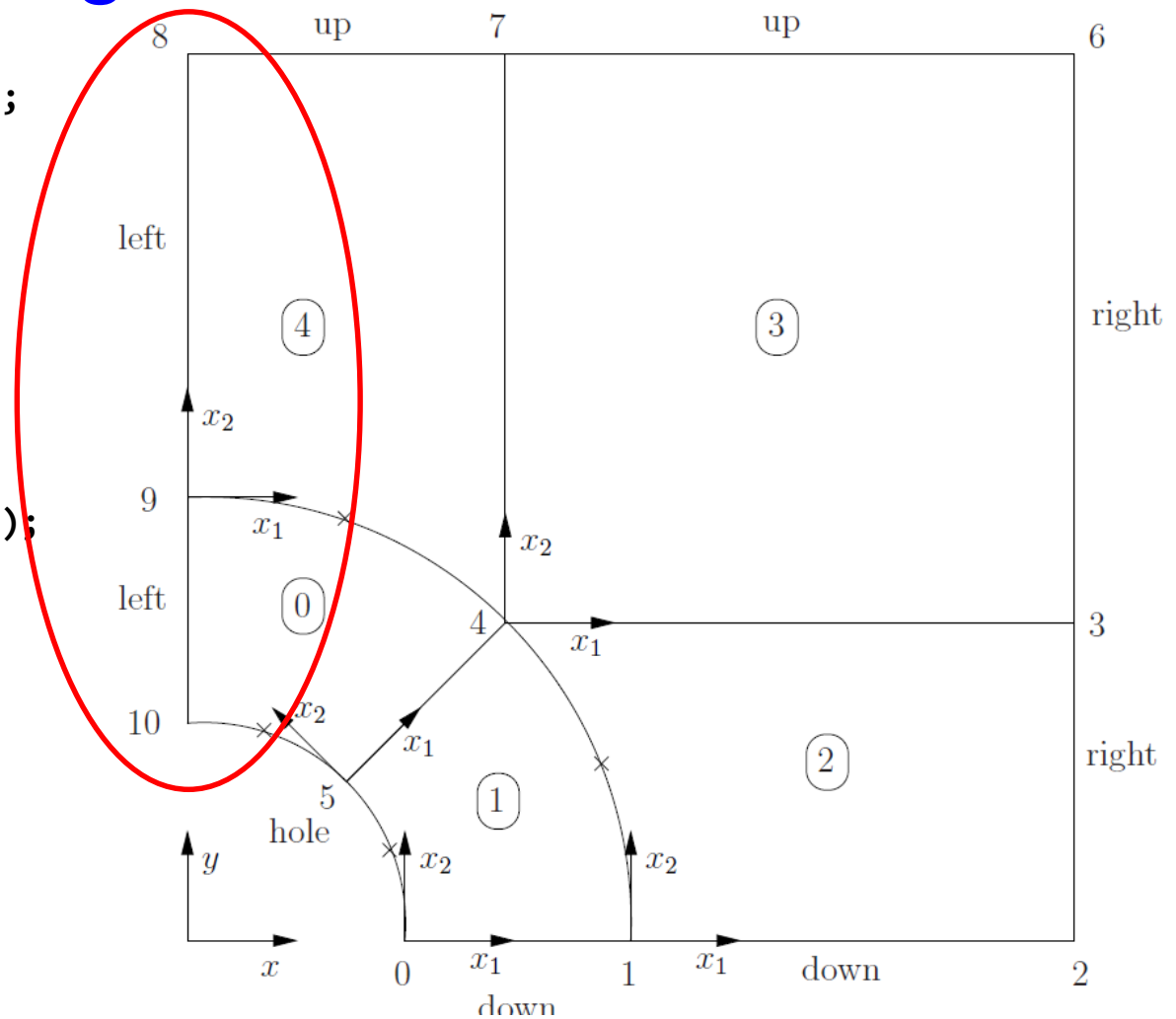
Post-processing

- ***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
```

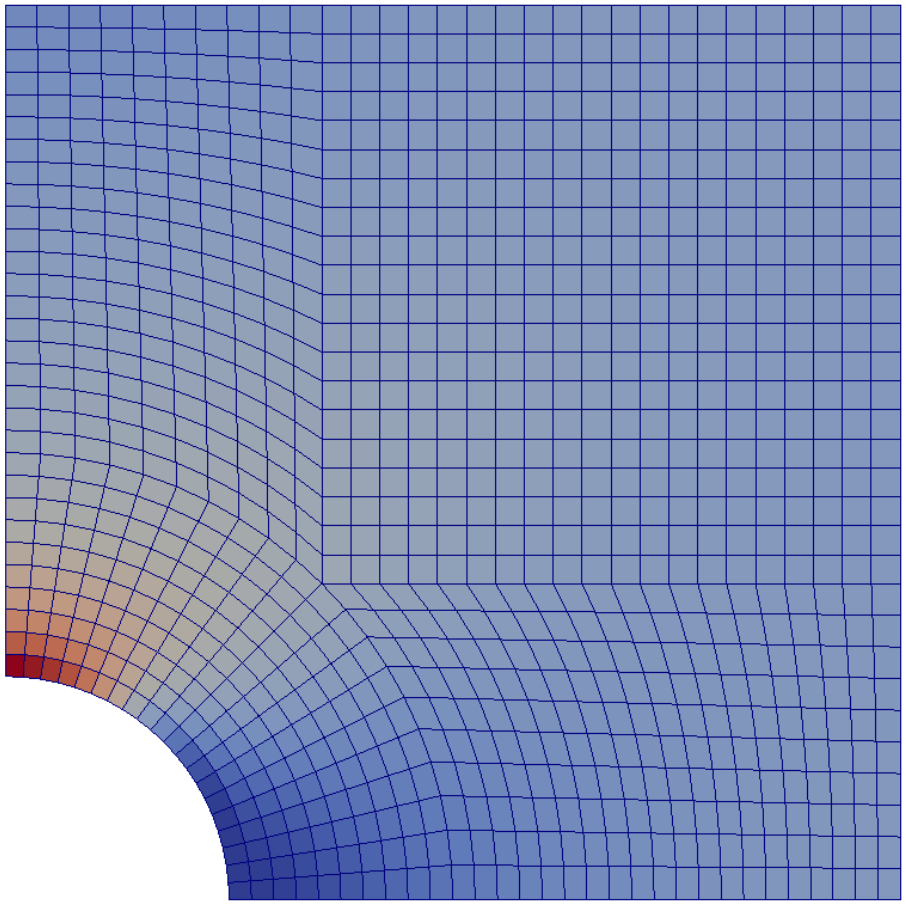
sampleDict for the plateHole case along the left line

```
/*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

Introduction to OpenFOAM

Create customized OpenFOAM solver

Before trying to develop your own solver...

- **Understand the background, e.g. go through commented icoFoam PISO solver, see http://openfoamwiki.net/index.php/IcoFoam**

```
//set up the linear algebra for the momentum equation.  The flux of U, phi, is treated explicity
//using the last known value of U.

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

// solve using the last known value of p on the RHS.  This gives us a velocity field that is
// not divergence free, but approximately satisfies momentum.  See Eqn. 7.31 of Ferziger & Peric
        solve(UEqn == -fvc::grad(p));

        // --- PISO loop---- take nCorr corrector steps

        for (int corr=0; corr<nCorr; corr++)
        {

// 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).
```

Simple tutorial on customizing the solver

- Overview of the OpenFOAM code structure
- A look at icoFoam
- Customizing an application
- Implementing a transport equation in a new application

OpenFOAM Code Structure

- **The OpenFOAM code is structures as follows (type `foam` and then `ls`).**
  - applications: source files of all the executables:
    - solvers
    - utilities
    - bin
    - test
  - bin: basic executable scripts.
  - doc: pdf and Doxygen documentation.
    - Doxygen
    - Guides-a4
  - lib: compiled libraries.
  - src: source library files.
  - test: library test source files.
  - tutorials: tutorial cases.
  - wmake: compiler settings.

Navigating the OpenFOAM source code

- **Some useful commands to navigate inside the OpenFOAM sources:**
  - **app** = $WM_PROJECT_DIR/applications
  - **sol** = $WM_PROJECT_DIR/applications/solvers
  - **util** = $WM_PROJECT_DIR/applications/utilities
  - **src** = $WM_PROJECT_DIR/src
- **Environment variables:**
  - $FOAM_APP = $WM_PROJECT_DIR/applications
  - $FOAM_SOLVERS = $WM_PROJECT_DIR/applications/solvers
  - $FOAM_UTILITIES = $WM_PROJECT_DIR/applications/utilities
  - $FOAM_SRC = $WM_PROJECT_DIR/src
- **OpenFOAM source code serves two functions:**
  - Efficient and customized top-level solver for class of physics. Ready to run in a manner of commercial CFD software

    ```
    ls $WM_PROJECT_DIR/applications/solvers
    ```
  - Examples showing the usage of OpenFOAM classes and library functions

    ```
    ls $WM_PROJECT_DIR/applications/test
    ```

Simple solver walk-through: icoFoam

- **Types of files**
  - Header files
    - Located before the entry line of the executable

      ```
      int main(int argc, char* argv[])
      ```
    - Contain various class definitions
    - Grouped together for easier use
  - Include files
    - Often repeated code snippets, e.g. mesh creation, Courant number
    - calculation and similar
    - Held centrally for easier maintenance
    - Enforce consistent naming between executables, e.g. `mesh, runTime`
  - Local implementation files
    - Main code, named consistently with the executable, e.g.:

      ```
      createFields.H
      ```

Walk through icoFoam: file organization

- `sol -> cd incompressible -> cd icoFoam`
- **The icoFoam directory consists of what follows (type `ls`):**

  `createFields.H icoFoam.C icoFoam.dep Make/`
- **The `Make` directory contains instructions for the `wmake` compilation command.**
- **`icoFoam.C` is the main file, while `createFields.H` is included by `icoFoam.C`.**
- **The file `fvCFD.H`, included by `icoFoam.C`, contains all the class definitions which are needed by icoFoam. See the file `Make/options` to understand where fvCFD.H is included from:**

  `$FOAM_SRC/finiteVolume/lnInclude/fvCFD.H, symbolic link to:`

  `$FOAM_SRC/finiteVolume/cfdTools/general/include/fvCFD.H`
- **Use the command `find PATH -iname "*LETTERSINFILENAME*"` to find where in PATH a file name containing `LETTERSFILENAME` in its file name is located.**
- **Example: `find $WM_PROJECT_DIR -iname "*fvCFD.H*"`**

Walk through icoFoam: A look into icoFoam.C

- **case setup and variable initialization**
  - icoFoam starts with

```
int main(int argc, char *argv[])
// where int argc and char *argv[] are the number of parameters
// and the actual parameters used when running icoFoam.
```

  - The case is initialized by:

```
# include "setRootCase.H"
# include "createTime.H"
# include "createMesh.H"
# include "createFields.H"
# include "initContinuityErrs.H"
```

    where all the included files except **createFields.H** are in `$FOAM_SRC/finiteVolume/lnInclude`.

  - `createFields.H` is located in the `icoFoam` directory. It initializes all the variables used in `icoFoam`. Have a look inside it and see how variables are created.

Walk through icoFoam:
A look into icoFoam.C – the time loop code

```
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());
    ...
    }
}
```

Walk through icoFoam: A look into icoFoam.C

- **time-loop code**
  - The time-loop starts by:

```
while (runTime.loop()) // and the rest is done at each time-step
```

  - The fvSolution subdictionary PISO is read, and the Courant Number is calculated and written to the screen by (use the find command):

```
#include "readPISOControls.H"
#include "CourantNo.H"
```

  - The momentum equations are defined and a velocity predictor is solved by:

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nu, U)
);
solve(UEqn == -fvc::grad(p));
```

Walk through icoFoam: A look into icoFoam.C

- **the PISO loop**
  - A PISO corrector loop is initialized by:

```
// --- PISO for loop
for (int corr=0; corr<nCorr; corr++)
// Or a while loop (newer versions)
while (runTime.loop())
```

  - The PISO algorithm uses these member functions:
    - `A()` returns the central coefficients of an `fvVectorMatrix`
    - `H()` returns the H operation source of an `fvVectorMatrix`
    - `Sf()` returns cell face area vector of an `fvMesh`
    - `flux()` returns the face flux field from an `fvScalarMatrix`
    - `correctBoundaryConditions()` corrects the boundary fields of a `volVectorField`
  - Identify the object types (classes) and use the OpenFOAM Doxygen (http://foam.sourceforge.net/doc/Doxygen/html) or the OpenFOAM wiki to better understand what they do.

Walk through icoFoam: A look into icoFoam.C

- **write statements**
  - At the end of icoFoam there are some write statements

```
runTime.write();

Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
    << "  ClockTime = " << runTime.elapsedClockTime() << " s"
    << nl << endl;
```

  - `write()` makes sure that all the variables that were defined as an loobject with `IOobject::AUTO_WRITE` are written to the time directory according to the settings in the `$FOAM_CASE/system/controlDict` file.
  - `elapsedCPUTime()` is the elapsed CPU time.
  - `elapsedClockTime()` is the elapsed wall clock time.

OpenFOAM work space General information

- **OpenFOAM is a library of tools, not a monolithic single-executable**
- **Most changes do not require surgery on the library level: code is developed in local work space for results and custom executables**
- **Environment variables and library structure control the location of the library, external packages (e.g. gcc, Paraview) and work space**
- **For model development, start by copying a model and changing its name: library functionality is unaffected**
- **Local workspace:**
  - Run directory: `$FOAM_RUN`. Ready-to-run cases and results, test loop etc. May contain case-specific setup tools, solvers and utilities.
  - Local work space: `$WM_PROJECT_USER_DIR`. Contains applications, libraries and personal library and executable space.

```
[fchen14@qb1 ~]$ echo $FOAM_RUN
/home/fchen14/OpenFOAM/fchen14-2.3.0/run
[fchen14@qb1 ~]$ echo $WM_PROJECT_USER_DIR
/home/fchen14/OpenFOAM/fchen14-2.3.0
```

Creating your own OpenFOAM applications

- **General steps:**
  1. Find appropriate code in OpenFOAM which is closest to the new use or provides a starting point
  2. Copy into local work space and rename
  3. Change file name and location of library/executable: Make/files
  4. Environment variables point to local work space applications and libraries:

     `$FOAM_PROJECT_USER_DIR, $FOAM_USER_APPBIN and $FOAM_USER_LIBBIN`
  5. Change the code to fit your needs

Creating your OpenFOAM applications

- **Creating the application icoScalarTransportFoam. It is an incompressible solver with a scalar transport equation (species mass fraction, temperature, . . .).**
- **To do this, we need to create a new application based on the icoFoam code.**

Setting up new solver directory and compile

- **The applications are located in** `$WM_PROJECT_DIR/applications`
  - `cd $WM_PROJECT_DIR/applications/solvers/incompressible`
- **Copy the icoFoam solver and put it in the** `$WM_PROJECT_USER_DIR/applications` **directory**
  - `cp -r icoFoam $WM_PROJECT_USER_DIR/applications`
- **Rename the directory and the source file name, clean all the dependencies and**
  - `mv icoFoam icoScalarTransportFoam`
  - `cd icoFoam`
  - `mv icoFoam.C icoScalarTransportFoam.C`
  - `wclean`
- **Go the Make directory and change files as follows:**

```
icoScalarTransportFoam.C
EXE = $(FOAM_USER_APPBIN)/icoScalarTransportFoam
```

- **Now compile the application with** `wmake` **in the** `icoScalarTransportFoam` **directory.**

icoScalarTransportFoam

- **Physical/numerical model modeling**
  - We want to solve the following transport equation for the scalar field *T*
  - It is an unsteady, convection-diffusion transport equation. ν is the thermal diffusion constant.

$$\frac{\partial T}{\partial t} + \nabla \cdot \left(UT \right) - \nabla \cdot \left(\nu \nabla T \right) = 0$$

  - What to do:
    - Create the geometric field *T* in the `createFields.H` file
    - Solve the transport equation for *T* in the `icoScalarTransportFoam.C` file after the PISO correction loop.

icoScalarTransportFoam: Creating field *T*

- **Modify createFields.H adding this volScalarField constructor for T:**

```
// Adding the Temperature field ...
Info<< "Reading field T\n" <<endl;
    volScalarField T
    (
        IOobject
        (
            "T",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
```

icoScalarTransportFoam: Creating field *T*

- **Modify createFields.H, adding the thermal diffusion constant DT after nu:**

```
//Add here...
dimensionedScalar DT
(
  transportProperties.lookup("DT")
);
//Done for now...
```

icoScalarTransportFoam: Creating field *T*

- **We have created a volScalarField object called T.**
  - `T` is created by reading a file (`IOobject::MUST_READ`) called T in the `runTime.timeName()` directory. At the beginning of the simulation, `runTime.timename()` is the `startTime` value specified in the `controlDict` file.
  - `T` will be automatically written (`IOobject::AUTO_WRITE`) in the `runTime.timeName()` directory according to what is specified in the `controlDict` file of the case.
  - `T` is defined on the computational mesh (`mesh` object):
    - It has as many internal values (`internalField`) as the number of mesh cells
    - It needs as many boundary conditions (`boundaryField`) as the mesh boundaries specified in the `constant/polyMesh/boundary` file of the case.

icoScalarTransportFoam
Solving the transport equation for T

- **Create a new empty file, TEqn.H:**

```
echo > TEqn.H
```

- **Include it in `icoScalarTransportFoam.C` at the end of the PISO loop:**

```
...
#include "continuityErrs.H"
    U = HbyA - rAU*fvc::grad(p);
    U.correctBoundaryConditions();
}
//add TEqn to the time loop
#include "TEqn.H"
runTime.write();
```

- **Now we will implement the scalar transport equation for *T* in icoScalarTransportFoam...**

icoScalarTransportFoam
Solving the transport equation for T

- **This the transport equation:**

$$\frac{\partial T}{\partial t} + \nabla \cdot \left(UT \right) - \nabla \cdot \left(\nu \nabla T \right) = 0$$

- **This is how we implement and solve it in `TEqn.H` after the PISO correction loop:**

```
fvScalarMatrix TEqn
(
    fvm::ddt(T)
    + fvm::div(phi, T)
    - fvm::laplacian(DT, T)
);
```

- **Now compile the application with `wmake` in the `icoScalarTransportFoam` directory.**

icoScalarTransportFoam: setting up the case

- **Copy the cavity tutorial case in your `$FOAM_RUN` directory and rename it:**

```
cp -r $FOAM_TUTORIALS/icoFoam/cavity $FOAM_RUN
mv cavity cavityScalarTransport
```

- **Introduce the field `T` in `cavityScalarTransport/0` directory:**

```
cp p T
```

icoScalarTransportFoam: case setup

- **startTime**
  - modify T as follows:

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 300;
boundaryField {
    movingWall
    {
        type        fixedValue;
        value       uniform 350;
    }
    fixedWalls
    {
        type        fixedValue;
        value       uniform 300;
    }
    frontAndBack
    {
        type        empty;
    }
}
```

icoScalarTransportFoam case setup

- **system/fvSchemes**
  - Modify the subdictionary divSchemes, introducing the discretization scheme for `div(phi,T)`

```
divSchemes
{
    default     none;
    div(phi,U) Gauss linear;
    div(phi,T) Gauss upwind;
} //NOTICE: there is no space between the comma and the variables
```

icoScalarTransportFoam case setup

- **system/fvSolution**
  - Introduce the settings for T in the solvers subdictionary

```
T
{
    solver          BICCG;
    preconditioner  DILU;
    tolerance       1e-7;
    relTol          0;
}
```

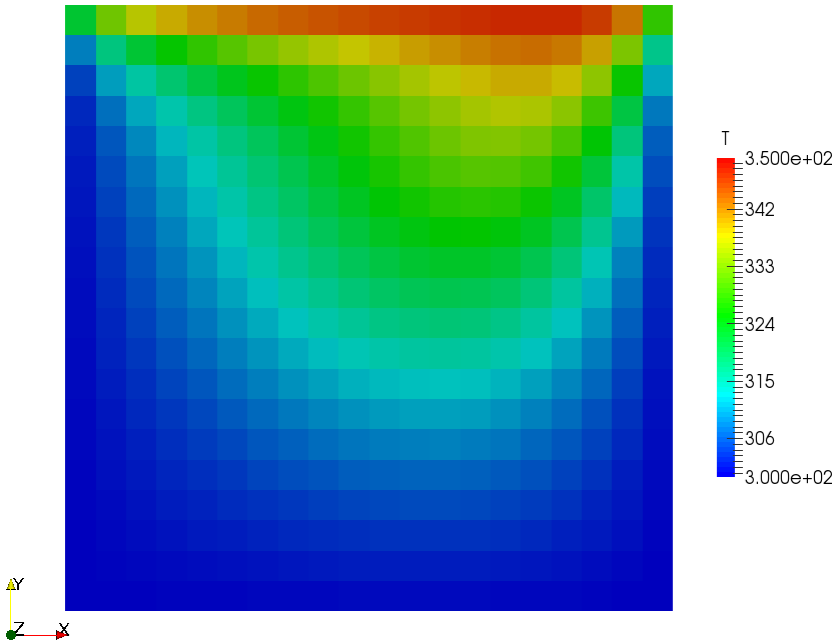
icoScalarTransportFoam case setup

- **constant/transportProperties**
  - add the following line under the definition of nu:

```
DT              DT [0 2 -1 0 0 0 0] 0.002;
```

icoScalarTransportFoam Run and Post-processing

- **Run the case:**
  - `icoScalarTransportFoam -case cavityScalarTranport`
- **Post-processing**
  - `foamToVTK -case cavityScalarTranport`
  - Render results in Paraview (on local machine)



icoScalarTransportFoam Run in Parallel

- **The modified icoScalarTransportFoam is capable of running in parallel without additional change to the source code**
- **need to edit** `cavityScalarTransport/system/decomposeParDict`**:**

```
//part of decomposeParDict
numberOfSubdomains 20;
method          simple;
simpleCoeffs
{
    n                   ( 4 5 1 );
    delta               0.001;
}
```

- **Then run the solver by using the following commands:**
  - `cd cavityScalarTransport`
  - `decomposePar`
  - `mpirun -np 20 icoScalarTransportFoam –parallel`
  - `reconstructPar`

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?