The icoFoam “Cavity Tutorial”

Eric Paterson

Senior Scientist, Applied Research Laboratory
Professor of Mechanical Engineering
The Pennsylvania State University
University Park, PA 16802 USA

29-30 April 2010

Acknowledgements

These slides are based upon training slides from previous workshops.

- Håkan Nilsson, Department of Applied Mechanics, Chalmers University of Technology, OpenFOAM Workshop Training 2009
- Gianluca Montenegro, Department of Energy, Politecnico di Milano, OpenFOAM Workshop Training 2008

Learning outcome

You will learn ...

- how to run the `icoFoam cavity` tutorial
- how the `icoFoam cavity` tutorial is set up, and how to modify the set-up
- how to search for examples of how to use the utilities.

The slides are based on the `OpenFOAM-1.5-dev` distribution.

The icoFoam cavity tutorial

- We will use the `icoFoam cavity` tutorial as a general example of how to set up and run applications in OpenFOAM.
- We will copy the `icoFoam cavity` tutorial to our `run` directory and run it. Then we will check what we did.
- Start by copying the tutorial to your `run` directory:

```
cp -r $FOAM_TUTORIALS/icoFoam/cavity $FOAM_RUN
cd $FOAM_RUN/cavity
```

Run the icoFoam cavity tutorial

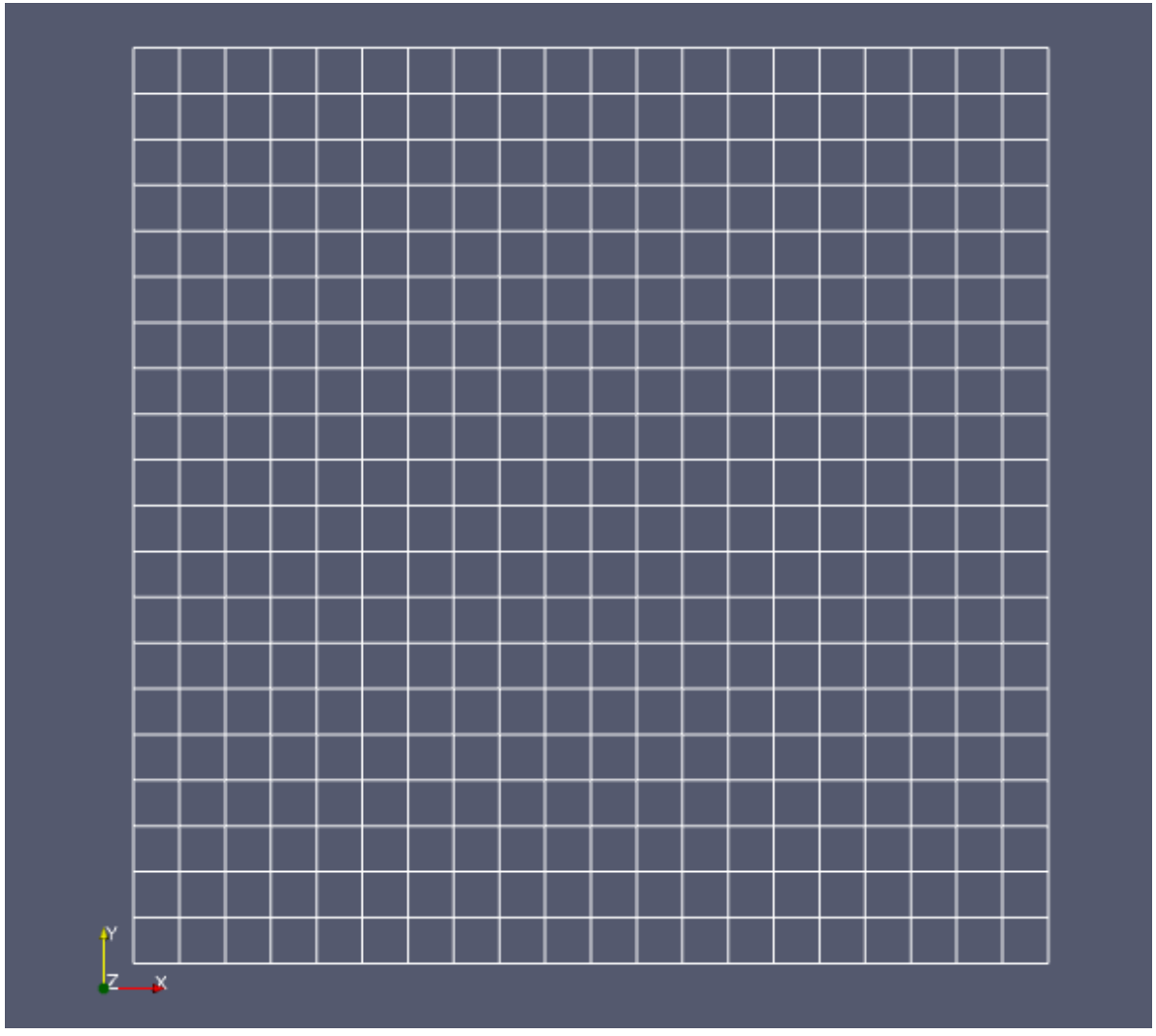
- The mesh is defined by a dictionary that is read by the `blockMesh` utility. Create the mesh by typing:
  `blockMesh`
  You have now generated the mesh in OpenFOAM format.
- Check the mesh by typing
  `checkMesh`
  You see the mesh size, the geometrical size and some mesh checks.
- This is a case for the `icoFoam` solver, so run
  `icoFoam >& log&`
  You now run the simulation in background using the settings in the case, and forward the output to the `log` file, where the Courant numbers and the residuals are shown.

Post-process the icoFoam cavity tutorial

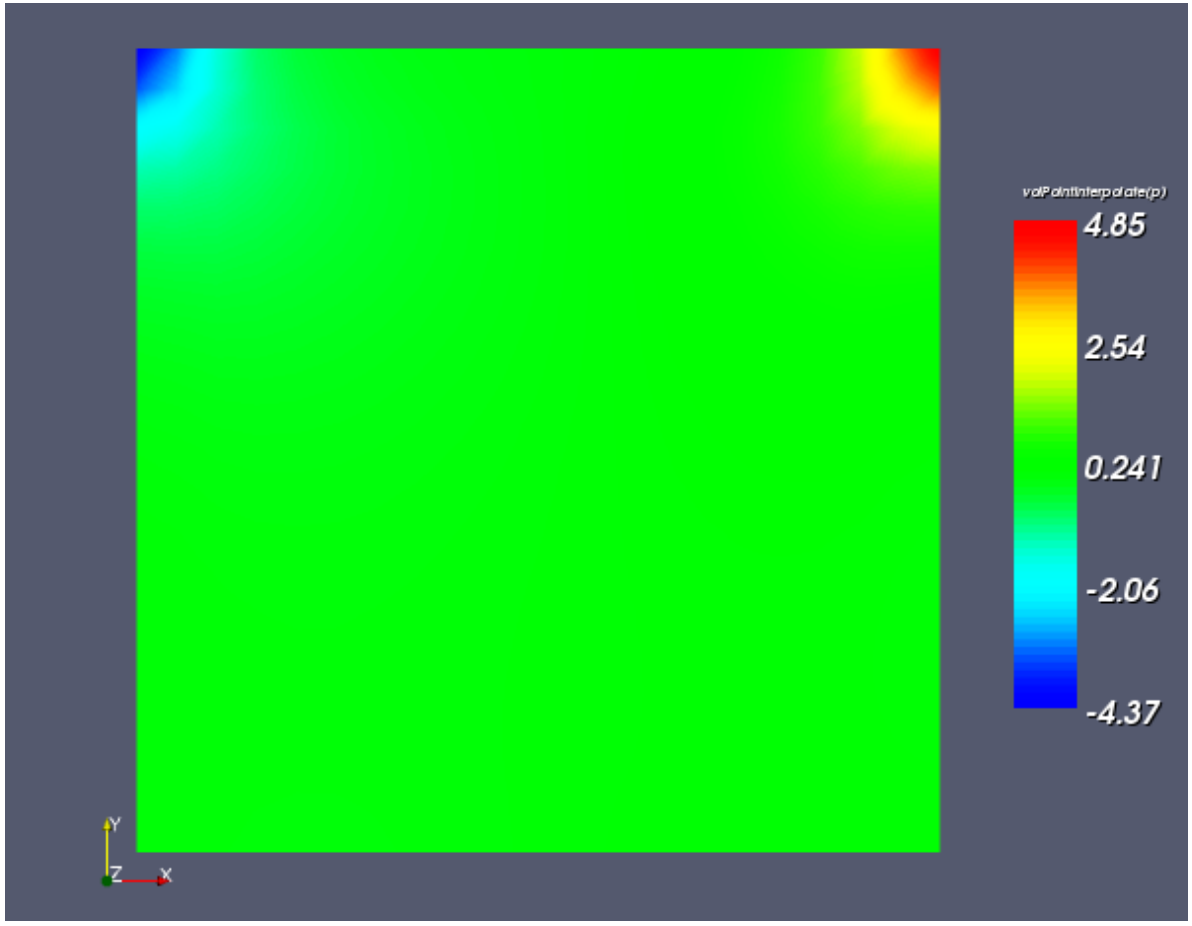
- View the results by typing:
  `paraFoam`
  Click Accept.
  Go to the final time step
  Choose which variable to color by with `Display/Color by`
  Move, rotate and scale the visualization using the mouse
- Find more instructions on the use of paraFoam in the UserGuide:
  `$WM_PROJECT_DIR/doc/Guides-a4/UserGuide.pdf`
- Exit paraFoam: `File/Exit`
- The results may also be viewed using third-party products

Later, we will show lots of more ways to use `paraFoam`.

Visualization of the mesh in paraFoam



Visualization of the static pressure in paraFoam



icoFoam cavity tutorial - What did we do?

- We will have a look at what we did when running the `cavity` tutorial by looking at the case files.
- First of all it should be noted that `icoFoam` is a *Transient solver for incompressible, laminar flow of Newtonian fluids*
- The case directory originally contains the following sub-directories: `0`, `constant`, and `system`. After our run it also contains the output `0.1`, `0.2`, `0.3`, `0.4`, `0.5`, and `log`
- The `0*` directories contain the values of all the variables at those time steps. The `0` directory is thus the initial condition.
- The `constant` directory contains the mesh and a dictionary for the kinematic viscosity transportProperty.
- The `system` directory contains settings for the run, discretization schemes, and solution procedures.
- The `icoFoam` solver reads the files in the case directory and runs the case according to those settings.

icoFoam cavity tutorial - The constant directory

- The `transportProperties` file is a dictionary for the dimensioned scalar `nu`.
- The `polyMesh` directory originally contains the `blockMeshDict` dictionary for the `blockMesh` mesh generator, and now also the mesh in OpenFOAM format.
- We will now have a quick look at the `blockMeshDict` dictionary in order to understand what mesh we have used.

icoFoam cavity tutorial - blockMeshDict dictionary

- The `blockMeshDict` dictionary first of all contains a number of vertices:

```
convertToMeters 0.1;
vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);
```

- There are eight vertices defining a 3D block. OpenFOAM always uses 3D meshes, even if the simulation is 2D.
- `convertToMeters 0.1;` multiplies the coordinates by 0.1.

icoFoam cavity tutorial - blockMeshDict dictionary

- The `blockMeshDict` dictionary secondly defines a block and the mesh from the vertices:

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

- `hex` means that it is a structured hexahedral block.
- `(0 1 2 3 4 5 6 7)` is the vertices used to define the block. The order of these is important - they should form a right-hand system (read the UserGuide yourself).
- `(20 20 1)` is the number of mesh *cells* in each direction.
- `simpleGrading (1 1 1)` is the expansion ratio, in this case equidistant. The numbers are the ratios between the end cells along three edges. There are other grading schemes as well (read the UserGuide yourself).

icoFoam cavity tutorial - blockMeshDict dictionary

- The blockMeshDict dictionary finally defines three patches:

```
patches
(
    wall movingWall
    (
        (3 7 6 2)
    )
    wall fixedWalls
    (
        (0 4 7 3)
        (2 6 5 1)
        (1 5 4 0)
    )
    empty frontAndBack
    (
        (0 3 2 1)
        (4 5 6 7)
    )
);
```

icoFoam cavity tutorial - blockMeshDict dictionary

- Each patch defines a type, a name, and a list of boundary faces
- Let's have a look at the fixedWalls patch:

```
wall fixedWalls
(
    (0 4 7 3)
    (2 6 5 1)
    (1 5 4 0)
)
```

- `wall` is the type of the boundary.
- `fixedWalls` is the name of the patch.
- The patch is defined by three sides of the block according to the list, which refers to the vertex numbers. The order of the vertex numbers is such that they are marched clock-wise when looking from inside the block. This is important, and unfortunately `checkMesh` will not find such problems!

icoFoam cavity tutorial - blockMeshDict dictionary

- To sum up, the blockMeshDict dictionary generates a block with:
  x/y/z dimensions 0.1/0.1/0.01
  20×20×1 cells
  `wall fixedWalls` patch at three sides
  `wall movingWall` patch at one side
  `empty frontAndBack` patch at two sides
- The type `empty` tells OpenFOAM that it is a 2D case.

- Read more about `blockMesh` yourself in the UserGuide.
- You can also convert mesh files from third-party products, see the UserGuide.

icoFoam cavity tutorial - blockMeshDict dictionary

- `blockMesh` uses the `blockMeshDict` to generate some files in the `constant/polyMesh` directory:
  `boundary`
  `faces`
  `neighbour`
  `owner`
  `points`
- `boundary` shows the definitions of the pathches, for instance:

```
movingWall
{
    type wall;
    nFaces 20;
    startFace 760;
}
```

- The other files define the points, faces, and the relations between the cells.

icoFoam cavity tutorial - The system directory

- The `system` directory consists of three set-up files:

  `controlDict` `fvSchemes` `fvSolution`
- `controlDict` contains general instructions on how to run the case.
- `fvSchemes` contains instructions on which discretization schemes that should be used for different terms in the equations.
- `fvSolution` contains instructions on how to solve each discretized linear equation system. It also contains instructions for the PISO pressure-velocity coupling.

icoFoam cavity tutorial - The controlDict dictionary

- The `controlDict` dictionary consists of the following lines:

```
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    uncompressed;
timeFormat          general;
timePrecision       6;
runTimeModifiable yes;
```

icoFoam cavity tutorial - The controlDict dictionary

- `application icoFoam;` Names the application the tutorial is set up for
- The following lines tells `icoFoam` to start at `startTime`=0, and stop at `endTime`=0.5, with a time step `deltaT`=0.005:

```
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.5;
deltaT          0.005;
```

icoFoam cavity tutorial - The controlDict dictionary

- The following lines tells `icoFoam` to write out results in separate directories (`purgeWrite 0;`) every 20 `timeStep`, and that they should be written in `uncompressed ascii` format with `writePrecision` 6. `timeFormat` and `timePrecision` are instructions for the names of the time directories.

```
writeControl      timeStep;
writeInterval     20;
purgeWrite        0;
writeFormat       ascii;
writePrecision    6;
writeCompression  uncompressed;
timeFormat        general;
timePrecision     6;
```

- `runTimeModifiable yes;` allows you to make modifications to the case while it is running.

icoFoam cavity tutorial - A dictionary hint

- If you don't know which entries are available for a specific key word in a dictionary, just use a dummy and the solver will list the alternatives, for instance:

```
stopAt          dummy;
```

When running icoFoam you will get the message:

```
dummy is not in enumeration
4
(
nextWrite
writeNow
noWriteNow
endTime
)
```

and you will know the alternatives.

icoFoam cavity tutorial - A dictionary hint

- Note that

```
startFrom               dummy;
```

only gives the following message without stopping the simulation:

```
--> FOAM Warning :
    From function Time::setControls()
    in file db/Time/Time.C at line 132
        expected startTime, firstTime or latestTime \
                 found 'dummy' in dictionary controlDict
    Setting time to 0
```

and the simulation will start from time 0.

- You may also use C++ commenting in the dictionaries:

```
// This is my comment
/* My comments, line 1
   My comments, line 2 */
```

icoFoam cavity tutorial - The fvSchemes dictionary

- The `fvSchemes` dictionary defines the discretization schemes, in particular the time marching scheme and the convections schemes:

```
ddtSchemes
{
    default         Euler;
}
divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}
```

- Here we use the `Euler` implicit temoral discretization, and the `linear` (central-difference) scheme for convection.
- `default none;` means that schemes must be explicitly specified.
- Find the available convection schemes using a 'dummy' dictionary entry. There are 50 alternatives, and the number of alternatives are increasing!

icoFoam cavity tutorial - The fvSolution dictionary

- The `fvSolution` dictionary defines the solution procedure.
- The solutions of the *p* linear equation systems is defined by:

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

- The *p* linear equation system is solved using the Conjugate Gradient solver `PCG`, with the preconditioner `DIC`.
- The solution is considered converged when the residual has reached the `tolerance`, or if it has been reduced by `relTol` at each time step.
- `relTol` is here set to zero since we use the PISO algorithm. The PISO algorithm only solves each equation once per time step, and we should thus solve the equations to `tolerance 1e-06` at each time step. `relTol 0;` disables `relTol`.

icoFoam cavity tutorial - The fvSolution dictionary

- The solutions of the *U* linear equation systems is defined by:

```
U PBiCG
{
    preconditioner   DILU;
    tolerance        1e-05;
    relTol           0;
};
```

- The *U* linear equation system in solved using the Conjugate Gradient solver `PBiCG`, with the preconditioner `DILU`.
- The solution is considered converged when the residual has reached the `tolerance 1e-05` for each time step.

icoFoam cavity tutorial - The fvSolution dictionary

- The settings for the PISO algorithm are specified in the `PISO` entry:

```
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

- `nCorrectors` is the number of PISO correctors. You can see this in the log file since the *p* equation is solved twice, and the pressure-velocity coupling is thus done twice.
- `nNonOrthogonalCorrectors` adds corrections for non-orthogonal meshes, which may sometimes influence the solution.
- The pressure is set to `pRefValue 0` in cell number `pRefCell 0`. This is over-ridden if a constant pressure boundary condition is used for the pressure.

icoFoam cavity tutorial - The 0 directory

- The `0` directory contains the dimensions, and the initial and boundary conditions for all primary variables, in this case `p` and `U`.
  `U`-example:

```
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;
    }}
```

icoFoam cavity tutorial - The 0 directory

- `dimensions [0 1 -1 0 0 0 0];` states that the dimension of `U` is m/s.
- `internalField uniform (0 0 0);` sets `U` to zero internally.
- The boundary patches `movingWall` and `fixedWalls` are given the `type fixedValue; value uniform (1 0 0);` and `(0 0 0)` respectively, i.e. $U_x = 1 m/s$, and $U = 0 m/s$ respectively.
- The `frontAndBack` patch is given `type empty;`, indicating that no solution is required in that direction since the case is 2D.
- You should now be able to understand `0/p` also.
- The resulting 0.* directories are similar but the `internalField` is now a `nonuniform List<scalar>` containing the results. There is also a `phi` file, containing the resulting face fluxes that are needed to yield a perfect restart. There is also some time information in `0.*/uniform/time`. The `0.*/uniform` directory can be used for uniform information in a parallel simulation.

icoFoam cavity tutorial - The log file

- If you followed the earlier instructions you should now have a `log` file. That file contains mainly the Courant numbers and residuals at all time steps:

```
Time = 0.09

Courant Number mean: 0.116099 max: 0.851428 velocity magnitude: 0.851428
PBiCG:  Solving for Ux, Initial residual = 0.000443324,
                    Final residual = 8.45728e-06, No Iterations 2
PBiCG:  Solving for Uy, Initial residual = 0.000964881,
                    Final residual = 4.30053e-06, No Iterations 3
PCG:  Solving for p, Initial residual = 0.000987921,
                    Final residual = 5.57037e-07, No Iterations 26
time step continuity errors : sum local = 4.60522e-09,
                    global = -4.21779e-19, cumulative = 2.97797e-18
PCG:  Solving for p, Initial residual = 0.000757589,
                    Final residual = 3.40873e-07, No Iterations 26
time step continuity errors : sum local = 2.81602e-09,
                    global = -2.29294e-19, cumulative = 2.74868e-18
ExecutionTime = 0.11 s  ClockTime = 1 s
```

icoFoam cavity tutorial - The log file

- Looking at the `Ux` residuals

```
PBiCG:  Solving for Ux, Initial residual = 0.000443324,
          Final residual = 8.45728e-06, No Iterations 2
```

- We see that we used the `PBiCG` solver
- The `Initial residual` is calculated before the linear equation system is solved, and the `Final residual` is calculated afterwards.
- We see that the `Final residual` is less than our `tolerance` in `fvSolution` (`tolerance 1e-05;`).
- The `PBiCG` solver used 2 iterations to reach convergence.
- We could also see in the log file that the pressure residuals and continuity errors were reported twice each time step. That is because we specified `nCorrectors 2;` for the `PISO` entry in `fvSolution`.
- The `ExecutionTime` is the elapsed CPU time, and the `ClockTime` is the elapsed wall clock time for the latest time step.

icoFoam cavity tutorial - foamLog

- It is of interest to have a graphical representation of the residual development.
- The `foamLog` utility is basically a script using `grep`, `awk` and `sed` to extract values from a log file.
- `foamLog` uses a database (`foamLog.db`) to know what to extract. The `foamLog.db` database can be modified if you want to extract any other values that `foamLog` doesn't extract by default.
- `foamLog` is executed on the `cavity` case with log-file `log` by:

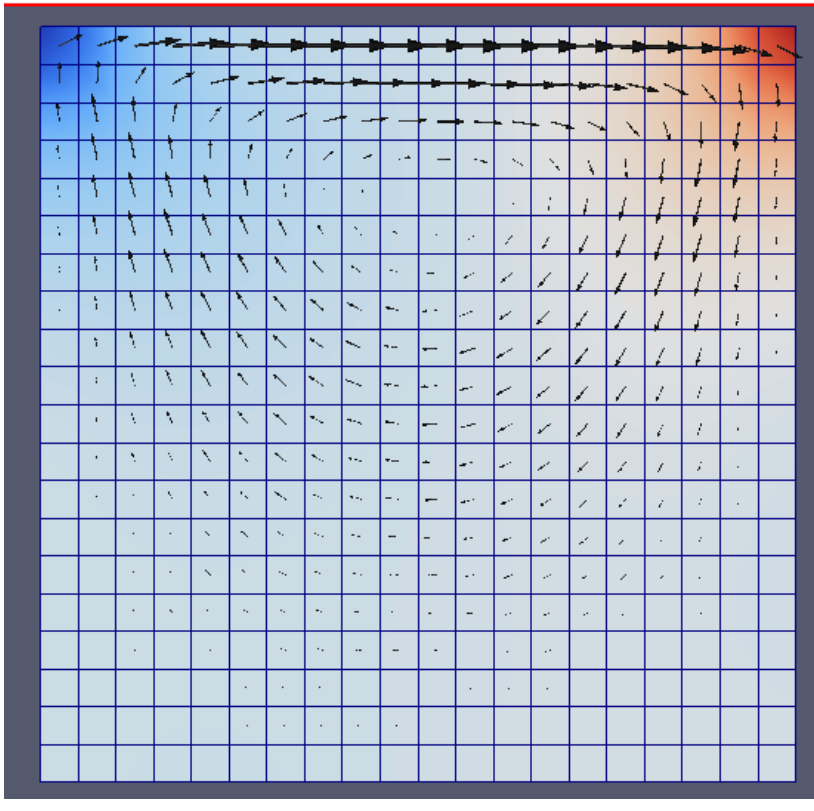
  ```
  foamLog log
  ```

- A directory `logs` has now been generated, with extracted values in ascii format in two columns. The first column is the `Time`, and the second column is the value at that time.
- Type `foamLog -h` for more information.
- The graphical representation is then given by Matlab, `xmgrace -log y Ux_0 p_0` or `gnuplot: set logscale y, plot "Ux_0","Uy_0","p_0"`.

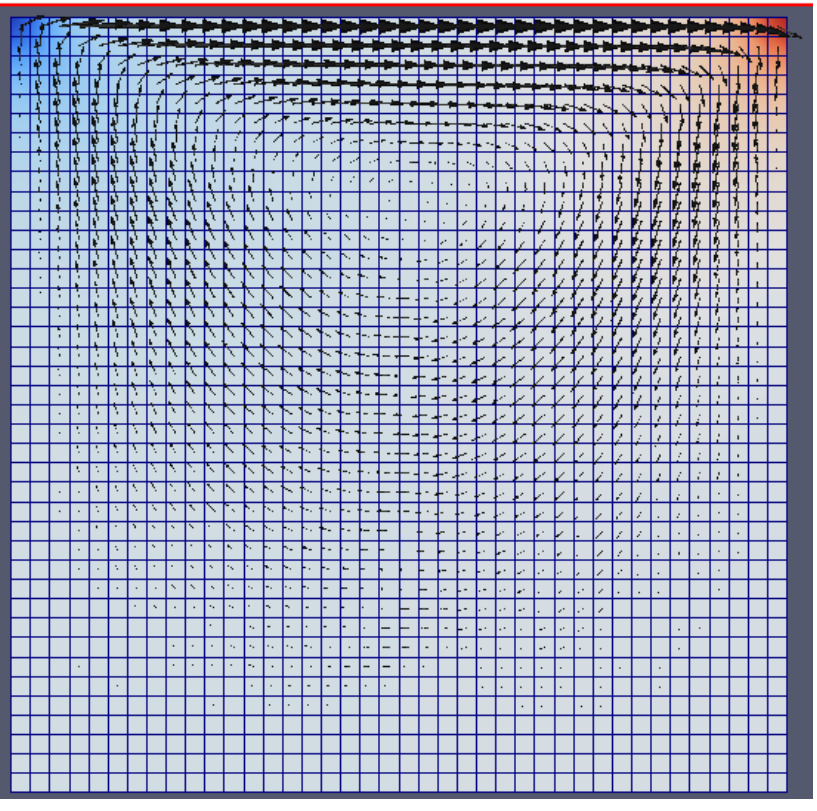
Run the icoFoam cavity tutorials using the Allrun script (1/8)

We will now run through the `icoFoam/cavity` tutorials

cavity

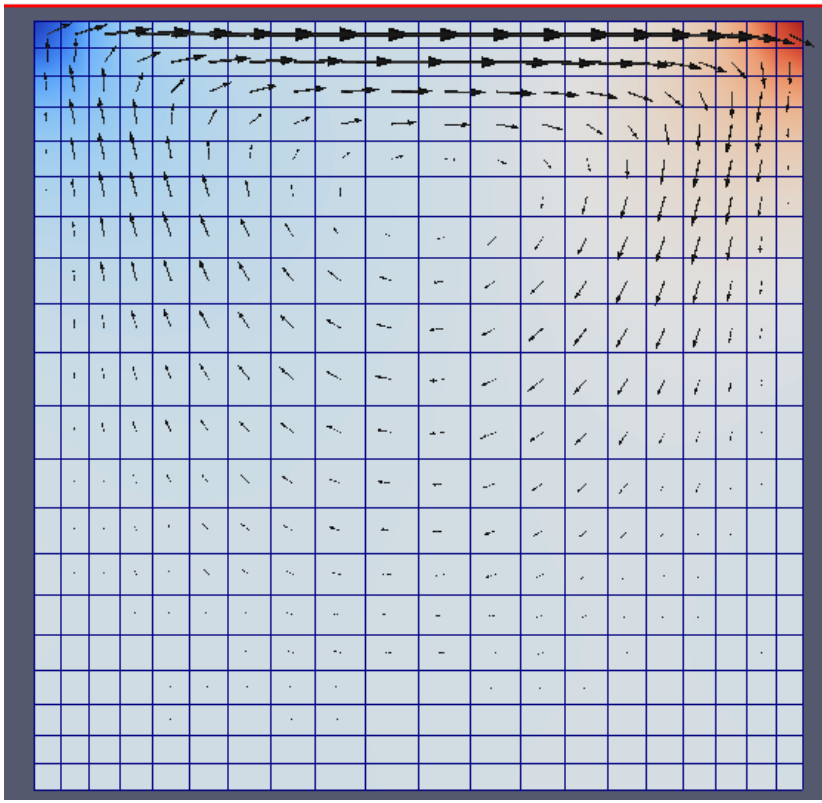


cavityFine

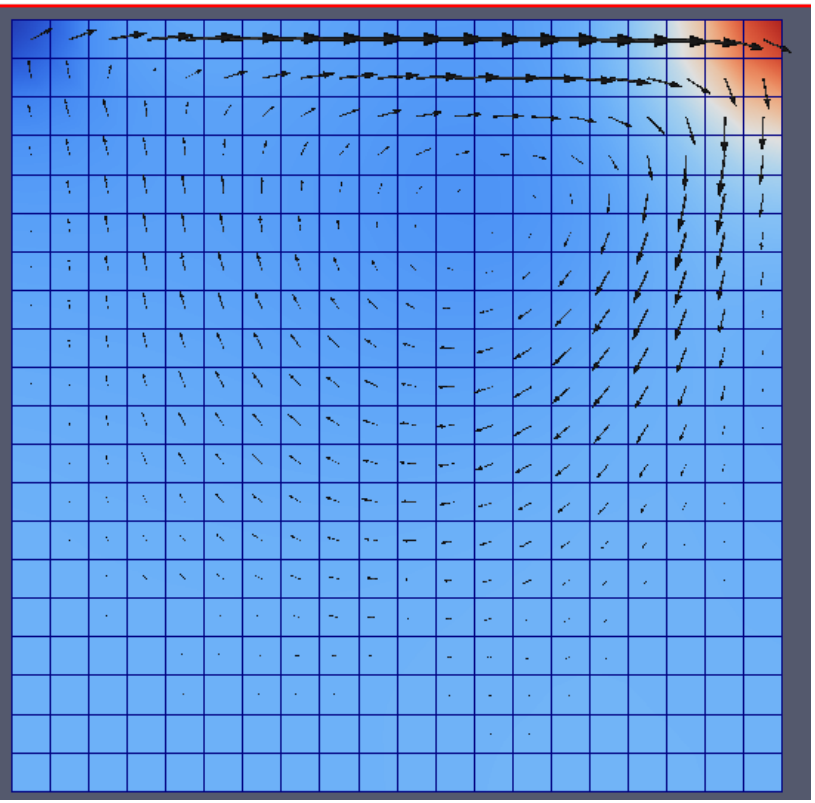


Run the icoFoam cavity tutorials using the Allrun script (2/8)

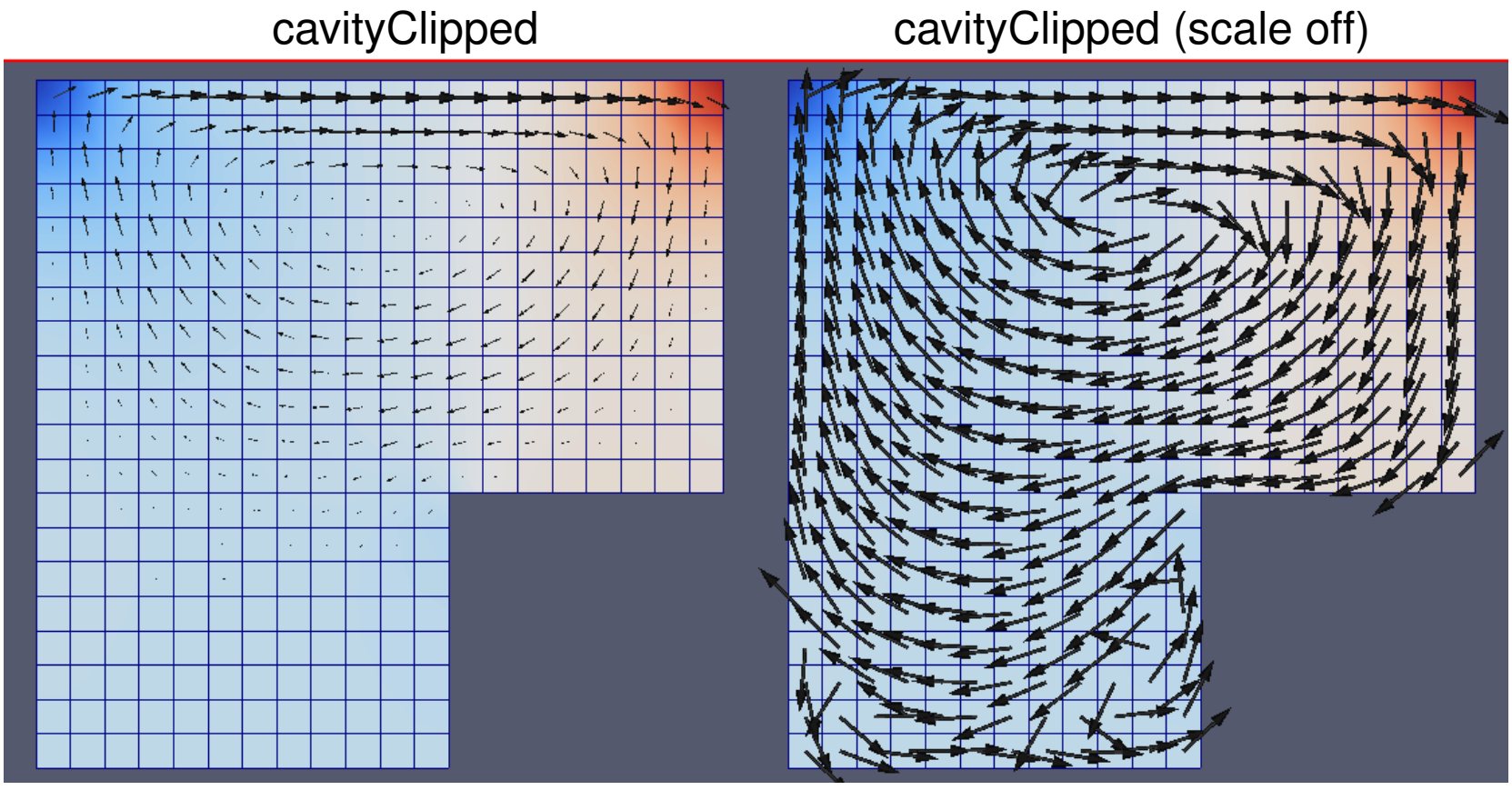
cavityGrade



cavityHighRe



Run the icoFoam cavity tutorials using the Allrun script (3/8)



Run the icoFoam cavity tutorials using the Allrun script (4/8)

First, copy the `icoFoam` tutorials directory:

```
cp -r $FOAM_TUTORIALS/icoFoam $FOAM_RUN
cd $FOAM_RUN/icoFoam
```

If you run the `Allrun` script for the icoFoam cavity tutorials you actually **first run the cavity case**

```
#Running blockMesh on cavity:
blockMesh -case cavity
#Running icoFoam on cavity:
icoFoam -case cavity
```

Run the icoFoam cavity tutorials using the Allrun script (5/8)

then run the cavityFine case:

```
#Cloning cavityFine case from cavity:
mkdir cavityFine
cp -r cavity/{0,system,constant} cavityFine
    [change "20 20 1" in blockMeshDict to "41 41 1"]
    [set startTime in controlDict to 0.5]
    [set endTime in controlDict to 0.7]
    [set deltaT in controlDict to 0.0025]
    [set writeControl in controlDict to runTime]
    [set writeInterval in controlDict to 0.1]
#Running blockMesh on cavityFine
blockMesh -case cavityFine
#Running mapFields from cavity to cavityFine (UserGuide, 6.5)
mapFields cavity -case cavityFine -sourceTime latestTime \
                 -consistent
#Running icoFoam on cavityFine
icoFoam -case cavityFine
```

Run the icoFoam cavity tutorials using the Allrun script (6/8)

then run the cavityGrade case:

```
#Running blockMesh on cavityGrade
blockMesh -case cavityGrade
#Running mapFields from cavityFine to cavityGrade
mapFields cavityFine -case cavityGrade \
                  -sourceTime latestTime -consistent
#Running icoFoam on cavityGrade
icoFoam -case cavityGrade
```

Run the icoFoam cavity tutorials using the Allrun script (7/8)

then run the cavityHighRe case:

```
#Cloning cavityHighRe case from cavity
mkdir cavityHighRe
cp -r cavity/{0,system,constant} cavityHighRe
#Setting cavityHighRe to generate a secondary vortex
    [set startFrom in controlDict to latestTime;]
    [set endTime in controlDict to 2.0;]
    [change 0.01 in transportProperties to 0.001]
#Copying cavity/0* directory to cavityHighRe
cp -r cavity/0* cavityHighRe
#Running blockMesh on cavityHighRe
blockMesh -case cavityHighRe
#Running icoFoam on cavityHighRe
icoFoam -case cavityHighRe
```

Run the icoFoam cavity tutorials using the Allrun script (8/8)

then run the cavityClipped case:

```
#Running blockMesh on cavityClipped
blockMesh -case cavityClipped
#Running mapFields from cavity to cavityClipped
cp -r cavityClipped/0 cavityClipped/0.5
mapFields cavity -case cavityClipped -sourceTime latestTime
(no longer consistent, so it uses system/mapFieldsDict)
    [Reset the boundary condition for fixedWalls to:]
    [        type            fixedValue;            ]
    [        value           uniform (0 0 0);       ]
    [      We do this since the fixedWalls got      ]
    [   interpolated values by cutting the domain   ]
#Running icoFoam on cavityClipped
icoFoam -case cavityClipped
```

Then there is also the Fluent elbow case, which we will not discuss now.

Run all the tutorials using the Allrun scripts

- You can run a similar script, located in the tutorials directory, and also named `Allrun`. This script will run through all the tutorials (calls `Allrun` in each solver directory).
- You can use this script as a tutorial of how to generate the meshes, how to run the solvers, how to clone cases, how to map the results between different cases etc.

Finding tutorials for the utilities in OpenFOAM

- There are no tutorials for the utilities, but we can search for examples:

```
find $WM_PROJECT_DIR -name \*Dict | \
     grep -v blockMeshDict | grep -v controlDict
```

  You will get a list of example dictionaries for some of the utilities.

- Most utilities take arguments. Find the alternatives by typing (for `foamToVTK`):

```
foamToVTK -help
```

  yielding:

```
Usage: foamToVTK [-noZero] [-surfaceFields] [-ascii]
[-region name] [-faceSet faceSet name] [-nearCellValue]
[-pointSet pointSet name] [-noLinks] [-case dir]
[-excludePatches patches to exclude] [-allPatches]
[-cellSet cellSet name] [-parallel] [-noFaceZones]
[-fields fields] [-constant] [-noPointValues] [-latestTime]
[-noInternal] [-time time]  [-help] [-doc] [-srcDoc]
```

Now you should be ready to go on exploring the applications by yourself.