Hands-on session, steps to follow

Text in red indicates interaction with a GUI
Text in brown are commands to type/paste in a terminal
Text in blue is to be added to the file indicated

1. Prepare

1.1. Start VNC session

We are going to use a VNC session to run OpenFOAM and ParaView in this tutorial. OpenFOAM itself can also be run in the portal shell or using ssh. But ParaView requires rendering, so we'll use VNC here.

In a web browser, go to https://portal-grace.hprc.tamu.edu
Interactive Apps > VNC
Select:
- 2 hours
- 3 cores
- 5 GB of memory
- Node Type : CPU Only

Launch

Once the VNC is started, open it and you will get a terminal. Commands can be pasted into clipboard and then to terminal with Shift + Insert

1.2. Load OpenFOAM v.10 and ParaView v.5.11

```
module purge
module load GCC/11.3.0  OpenMPI/4.1.4 OpenFOAM/10 ParaView/5.11.0-mpi-cuda
source $FOAM_BASH
```

1.3. Create a place to run OpenFOAM cases

```
mkdir $SCRATCH/FOAM_RUN
cd $SCRATCH/FOAM_RUN
```

2. Run windAroundBuildings tutorial

These are the basic steps outlined in the Allrun script of the tutorial. Each OpenFOAM tutorial has an Allrun script that gives you an idea what tools are involved. You can simply run the Allrun

script as bash Allrun. But we'll do it step by step here so we have time to talk about what is going on.

2.1. Copy tutorial case

```
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/windAroundBuildings .
chmod --recursive +w windAroundBuildings
cd windAroundBuildings
```

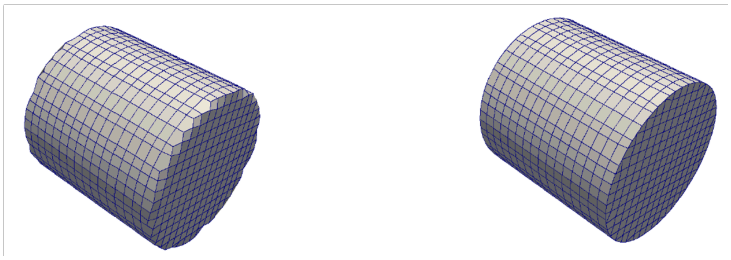
2.2. Create background mesh

```
blockMesh
```

2.3. Extract surface features/edges (optional)

snappyHexMesh can detect edges and refine around them, as well as snap the mesh to wrap nicely around corners. There seems to be a problem with the surfaceFeatures utility on Grace so we'll have to disable it for now. We'll do this by removing the line that picks out the edges in system/snappyHexMeshDict.

```
surfaceFeatures
OR
sed -i '/buildings.eMesh/d' system/snappyHexMeshDict
```



Mesh with/without edge snapping

2.4. Refine around object and snap mesh

```
snappyHexMesh -overwrite
```

2.5. Run incompressible solver

```
simpleFoam
```

2.6. Look at the results with ParaView

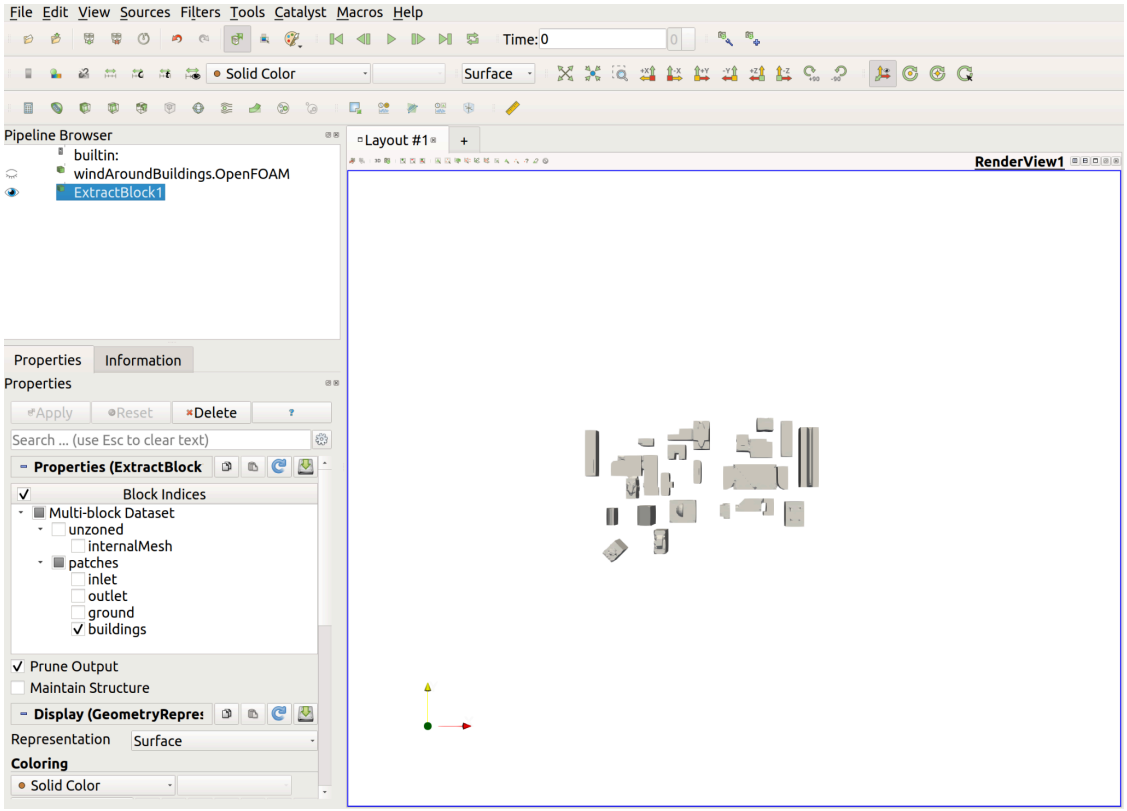
paraFoam -builtin

In ParaView

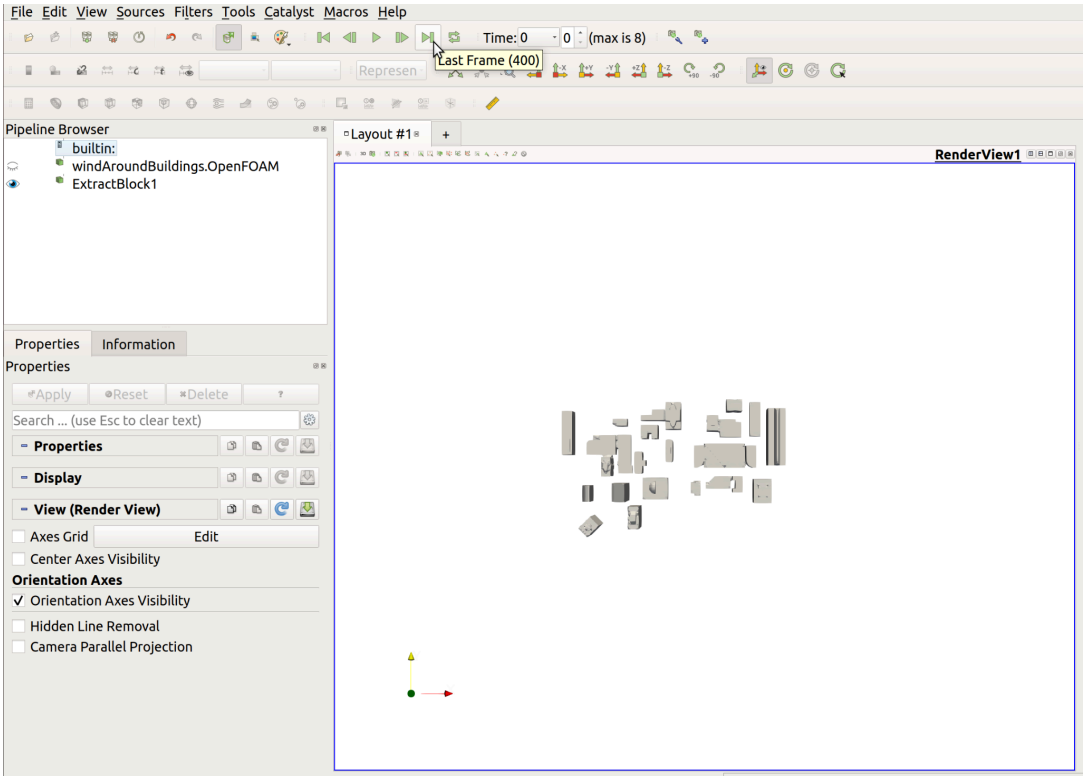
Click “Apply” when ParaView finishes loading. If not already selected, select patch/buildings under “Mesh Regions” (Scroll down from green “Apply” button”)

Extract building geometry

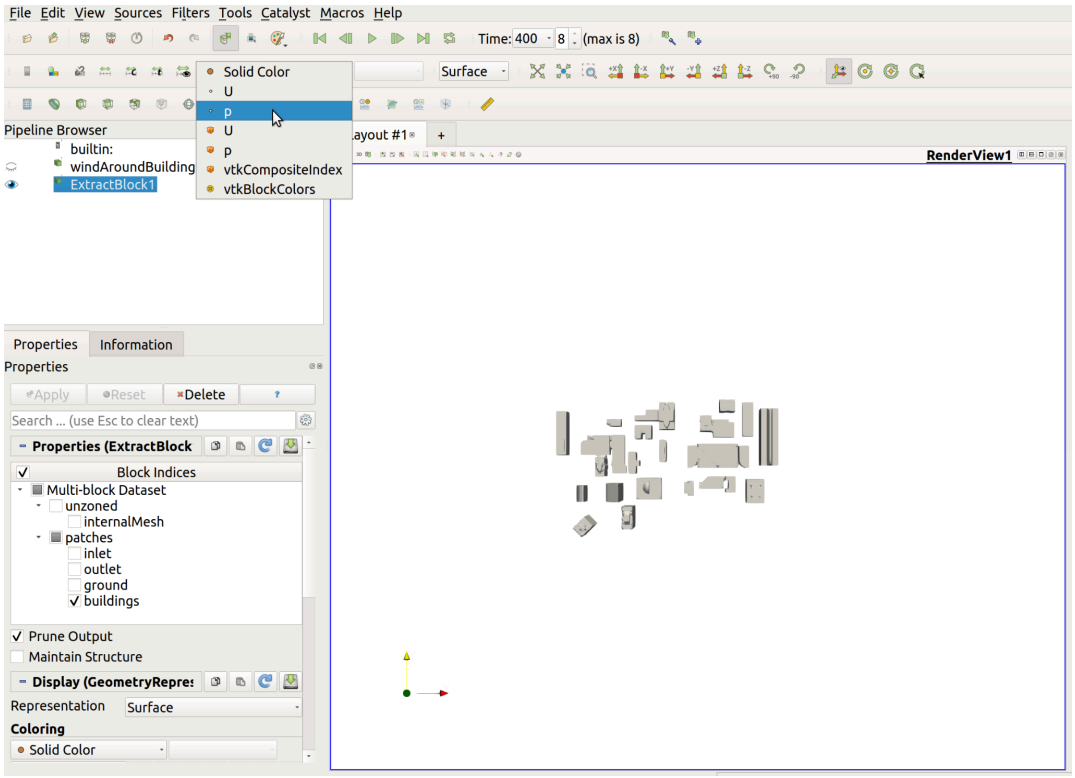
- In the top menu, Filters -> Alphabetical -> Extract block
- Select “buildings” patch



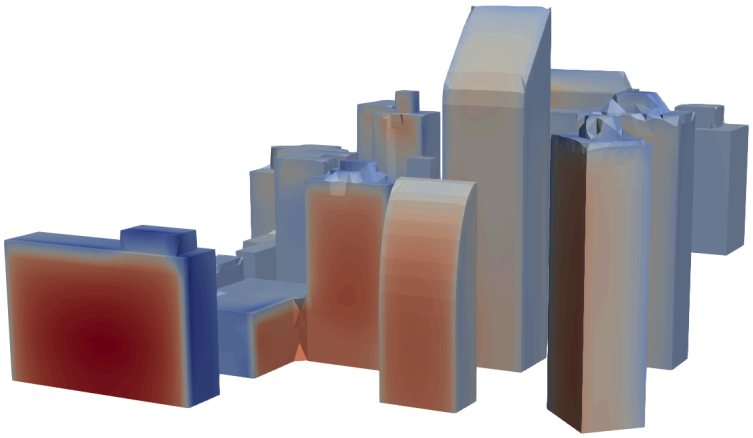
- Advance time to last time step (400) by clicking ▸| in the case menu



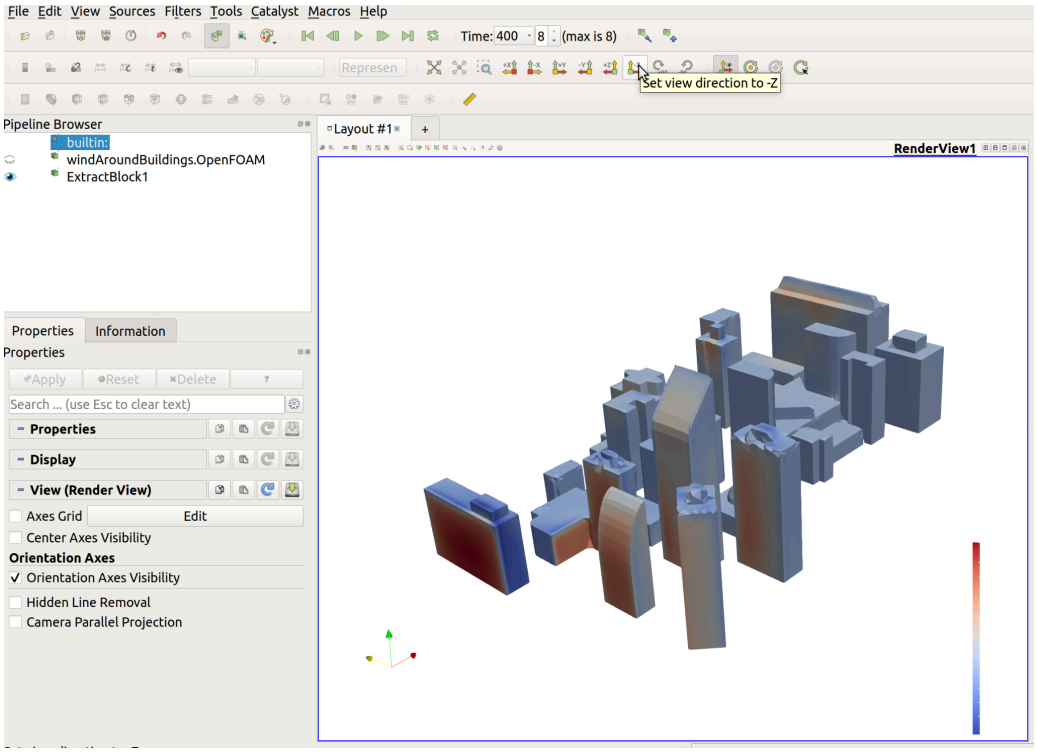
- Color by pressure by clicking “Solid Color” in the display menu and change to *p*



- Rotate to view result

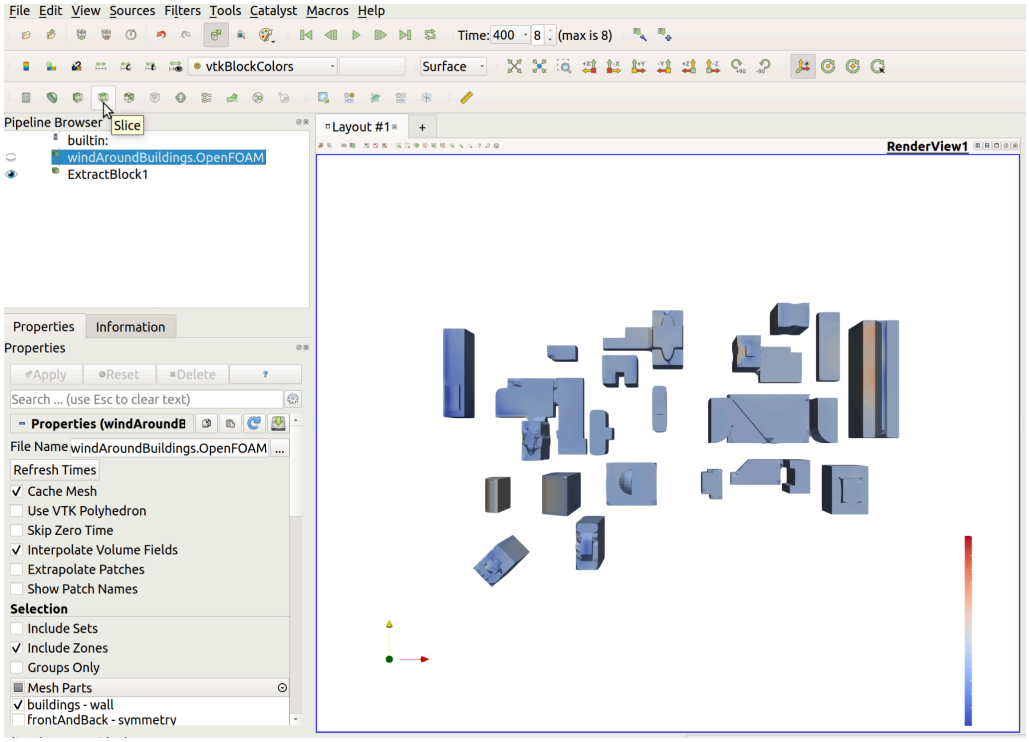


- Return to top view by clicking -z in the display menu

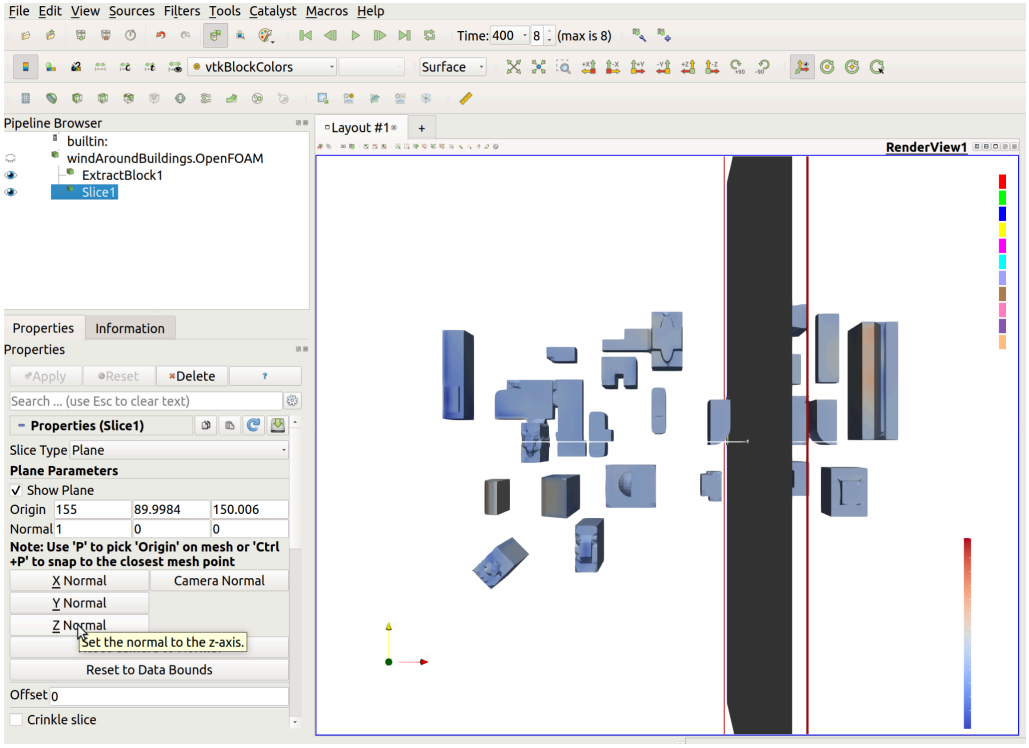


Make a slice to view velocity profile

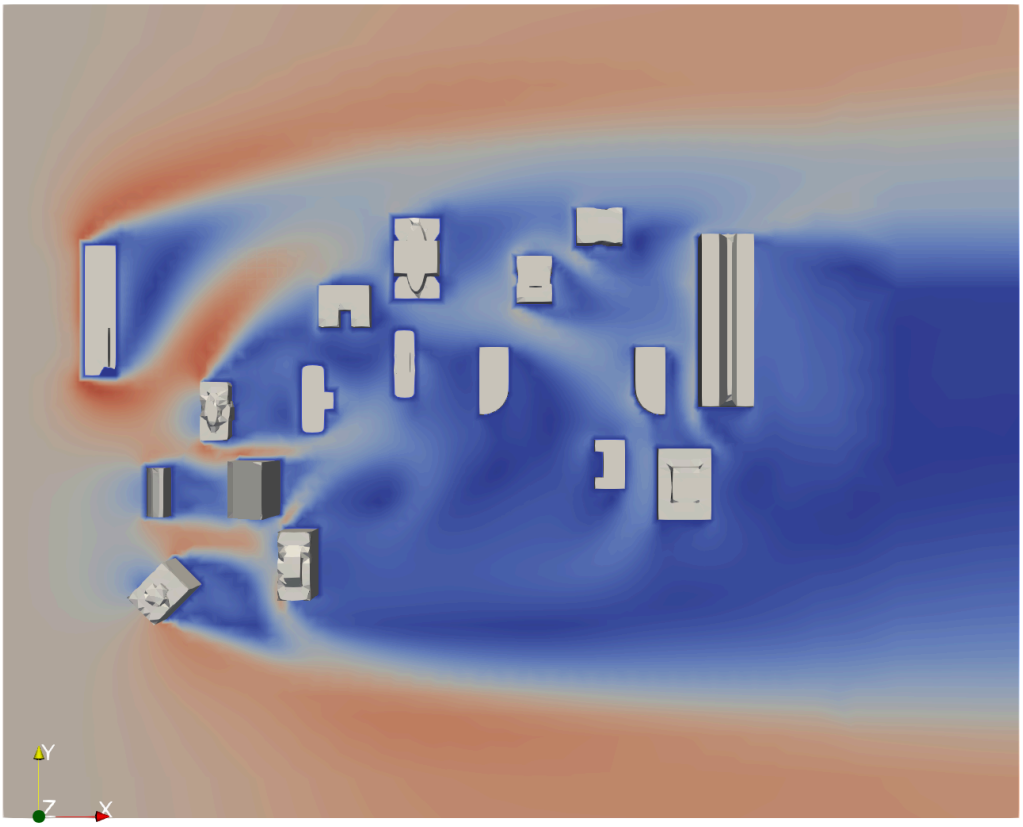
- Click windAroundBuildings.OpenFOAM in the left pipeline browser and click “Slice” in the filters menu



- Click “Z-Normal” in the properties of the slice and change Origin z to 30m (x and y do not matter)



- Color slice by *U* (Same method as coloring buildings by *p*) and view results



3. Modify windAroundBuildings tutorial

Since you are unlikely to find a tutorial for exactly your problem. The usual way to get started in OpenFOAM is:

- Find the tutorial that is closest to your problem. Meaning it has the key features you are looking for. Not necessarily the closest geometry. Geometry is “easy” to change.
- Modify the geometry, boundary conditions and/or solver settings to suit your needs.
- Success?

In this example, we will add a new building to the existing geometry from the tutorial and measure forces on it. The new building will be a cylindrical tower with a height of 225m and diameter of 50m.

3.1. Copy the case again

```
cd $SCRATCH/FOAM_RUN
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/windAroundBuildings
./windAroundNewBuilding
```

```
chmod --recursive +w windAroundNewBuilding

cd windAroundNewBuilding
```

3.2. Add the geometry for the new building

Let's make a simple cylindrical tower. Using the searchableCylinder geometry tool in OpenFOAM. We'll place it at x=150m, y=30m and have it extend from z=0m to z=225m. The radius of the building is 25m.

In snappyHexMeshDict:

Add in geometry

```
newBuilding
{
        type searchableCylinder;
        point1  (  150   30   0);
        point2  (150 30  225);
        radius 25;
}
```

Add in refinementSurfaces

```
newBuilding
{
        level (3 3);                    //Specify refinement level 3 on new bldg.
        patchInfo { type wall; } //Specify boundary type "wall"
}
```

3.3. Extend the domain to accommodate taller building

In blockMeshDict

Modify:

```
zMax   300;
```

3.4. Calculate forces and moments on the new building

In controlDict

Add in functions

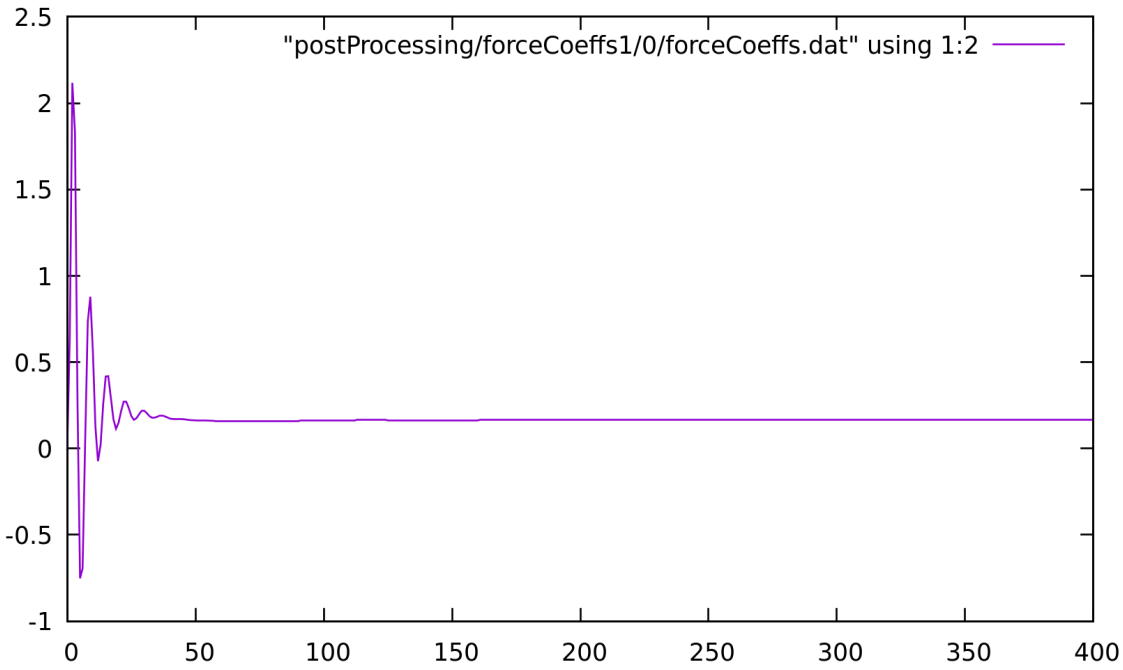
```
forceCoeffs1
{
        type            forceCoeffs;
        libs            ("libforces.so");

        writeControl    timeStep;
        timeInterval    1;      //Write forces every time step
        log             yes;

        patches         (newBuilding); //What object to calculate forces on
        rho             rhoInf;         // Indicates incompressible
        rhoInf          1;      // Redundant for incompressible
        liftDir         (0 0 1);
        dragDir         (1 0 0);
        CofR            (150 30 0);  // Moment calculated around this point.
        pitchAxis       (0 1 0);
        magUInf         10;
        lRef            50;     // Diameter
        Aref            11250;          // Projected
}
```

3.5. Repeat steps from tutorial

```
blockMesh
sed -i '/buildings.eMesh/d' system/snappyHexMeshDict
snappyHexMesh -overwrite
simpleFoam
```

3.6. Plot forces over time

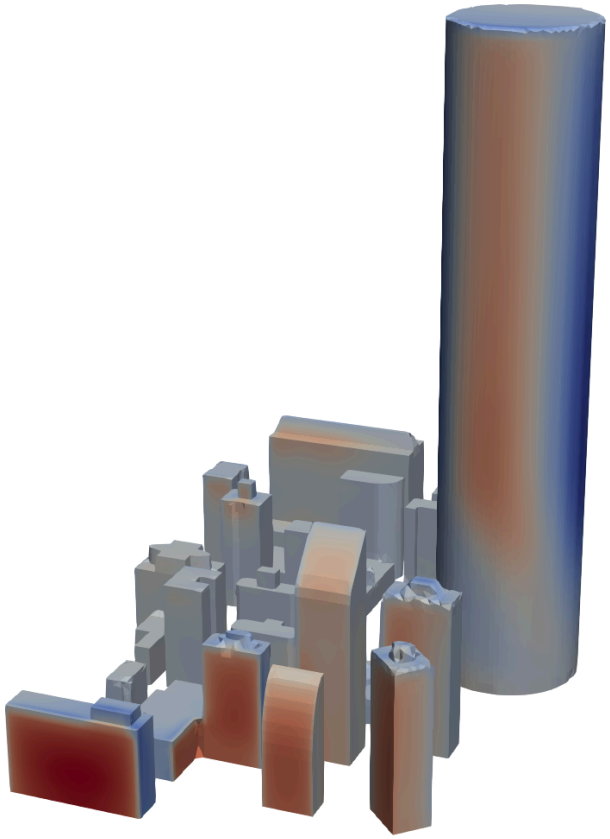
```
gnuplot
plot "postProcessing/forceCoeffs1/0/forceCoeffs.dat" using 1:2 with lines
exit
```



3.7. Look at the results with ParaView

```
paraFoam -builtin
```

Follow the same steps as in the tutorial. But when doing “Extract Block”, select both *buildings* and *newBuilding* blocks.



4. Running in parallel

4.1. Decompose the mesh

The windAroundBuildings tutorial doesn't have a dictionary for decomposing the mesh, so we'll copy one from another tutorial.

```
cp $FOAM_TUTORIALS/incompressible/simpleFoam/motorBike/system/decomposeParDict system
```

```
chmod +w system/decomposeParDict
```

We'll edit the decomposeParDict to decompose into 3 domains.

In decomposeParDict

Modify:

```
numberOfSubdomains   3;
n                    (3 1 1);
```

Then we can decompose the mesh for parallel processing using the decomposePar command.

Clean up from previous run, remove all but zero time

```
rm -r [1-9]*
```

Decompose the case

```
decomposePar
```

4.2. Run simpleFoam in parallel

```
mpirun -np 3 simpleFoam -parallel
```

Running in serial took about 180 seconds and this took about 60 seconds (180/3). So it scales pretty ideally in this case!

5. Modify solver

5.1. Prepare a directory for custom applications

```
mkdir $SCRATCH/FOAM_APPS
cd $SCRATCH/FOAM_APPS
```

5.2. Copy an existing application

```
cp -r $WM_PROJECT_DIR/applications/solvers/incompressible/icoFoam my_icoFoam

chmod --recursive +w my_icoFoam

cd my_icoFoam

mv icoFoam.C my_icoFoam.C

sed -i 's/icoFoam/my_icoFoam/g' Make/files
```

```
sed -i 's/FOAM_APPBIN/FOAM_USER_APPBIN/g' Make/files
```

The last two commands are very important not to skip! Or you may overwrite the default solver.

5.3. Define the new field and make the solver read/write it.

In createFields.H

Add, just after "createPhi" :

```
#include "createPhi.H"

Info<< "Reading field myField\n" << endl;
volScalarField myField
(
    IOobject
    (
        "myField",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

label pRefCell = 0;
```

5.4. Create a transport equation for the new field

We want to create a scalar transport equation for myField. Let's call the field *F*:

$$\frac{\partial F}{\partial t} + \nabla \cdot (\overline{U}F) - \nabla \cdot (\nu \nabla F) = 0$$

Let's put the equation in its own file for better readability.

```
touch myFieldEqn.H
```

In myFieldEqn.H

Add :

```
solve
(
        fvm::ddt(myField)
        + fvm::div(phi, myField)
        - fvm::laplacian(nu, myField)
);
```

In my_icoFoam.C

Add, just after while(piso.correct()) :

```
while(piso.correct())
{
        #include "myFieldEqn.H"
        volScalarField rAU(1.0/UEqn.A())
```

5.5. Compile the new solver

```
wmake
```

Note: If you are compiling a library rather than an executable, the command is `wmake libso`

5.6. Test the new solver

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity .
chmod --recursive +w cavity
cd cavity
```

Create the mesh

```
blockMesh
```

Make Boundary Condition for myField

Copy the boundary conditions from p

```
cp 0/p 0/myField
```

In 0/myField

Modify :

Make the field non-dimensional.

```
dimensions      [0 0 0 0 0 0 0];
```

Set a fixed value of 1 at the top boundary. Set to zero on walls.

```
movingWall
{
        type fixedValue;
        value uniform 1;
}
fixedWalls
{
        type fixedValue;
        value uniform 0;
}
```

Define matrix solvers for myField

Now we need to define solvers and schemes for myField

<u>In system/fvSolution</u>

Add, in "solvers" sub-dictionary :

```
myField
{
        solver PBiCG;
        preconditioner DILU;
        tolerance 1e-5;
        relTol 0;
}

myFieldFinal {$myField;}
```

<u>In system/fvSchemes</u>

Add, in "divSchemes" sub-dictionary :

```
div(phi,myField) Gauss linear;
```

Run the new solver!

my_icoFoam

Look at the results

paraFoam -builtin