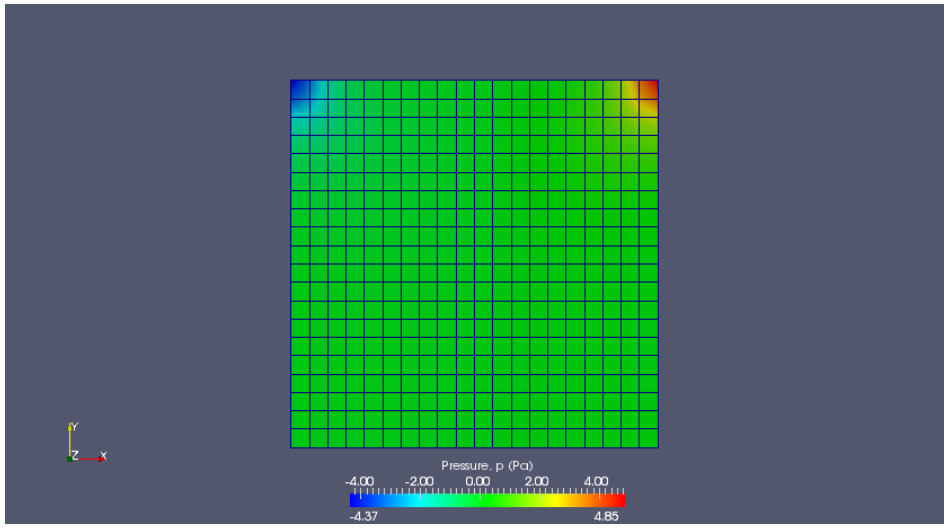
Assignment 1

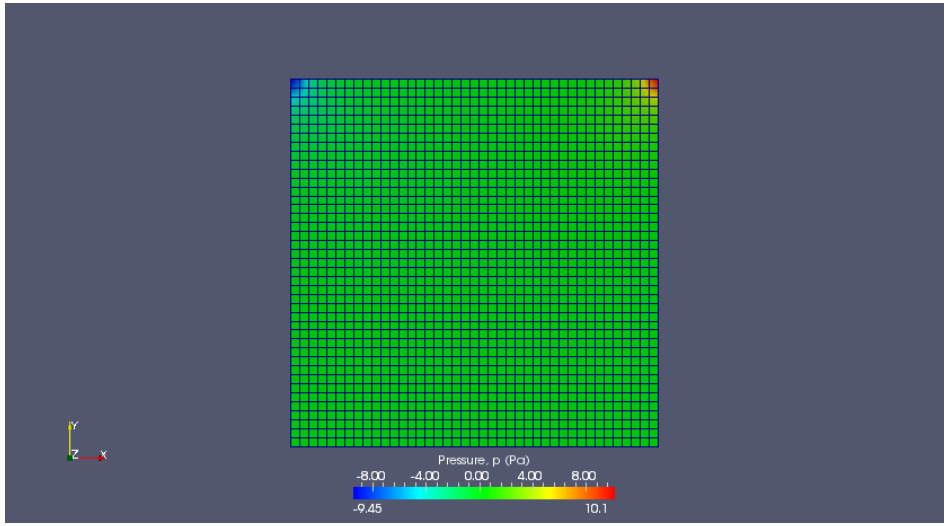
PhD course in CFD with OpenSource Software

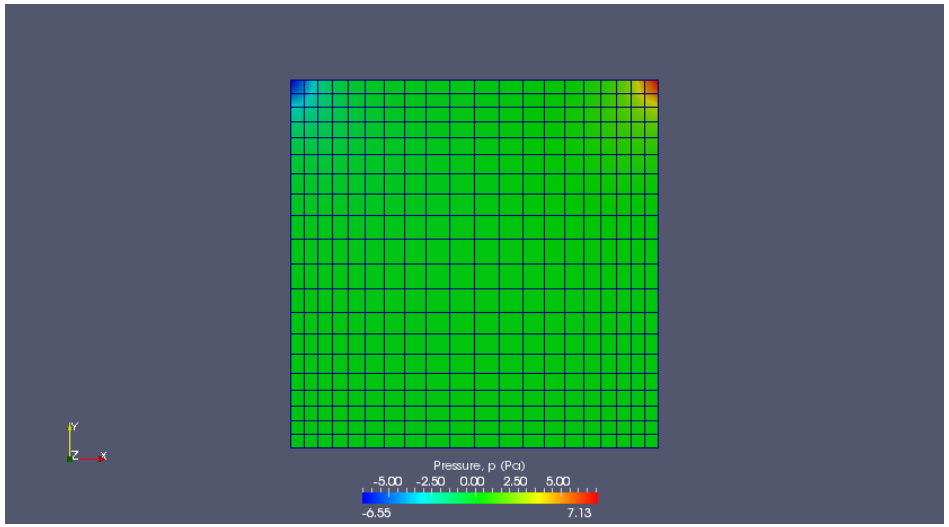
Tian Tang

2012-09-09

Different meshing: cavity, cavityFine, cavityGrade



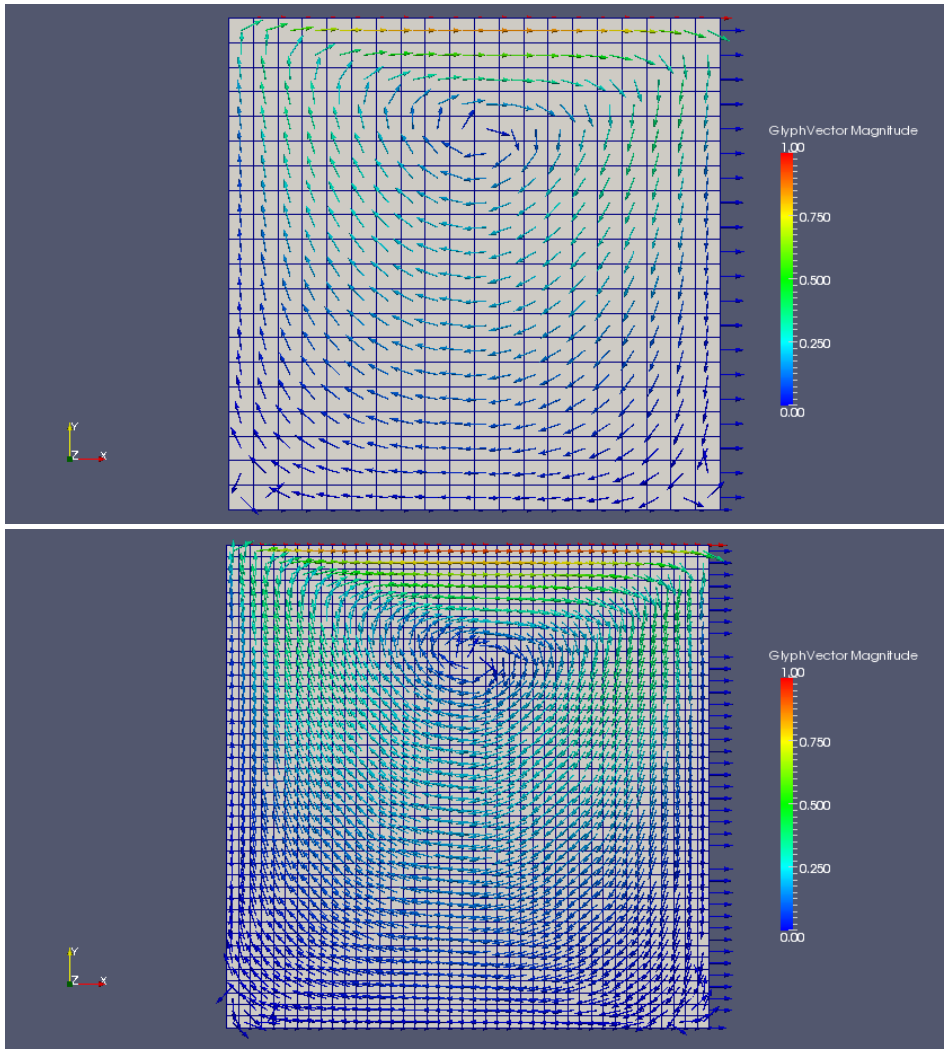


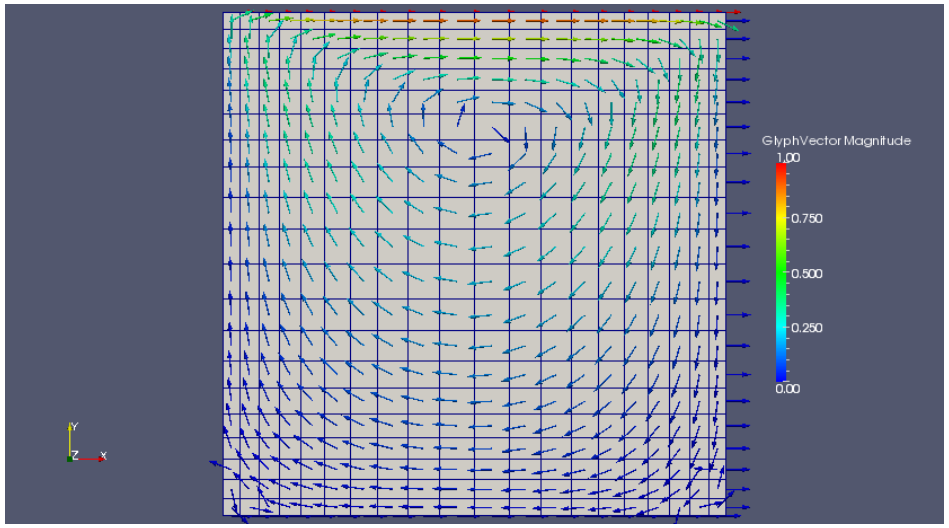


The pressure plot for the cases of cavity(upper left), cavityFine(lower left), and cavityGrade(right).

The representation is *surface with Edges*.

Different meshing: cavity, cavityFine, cavityGrade - continued

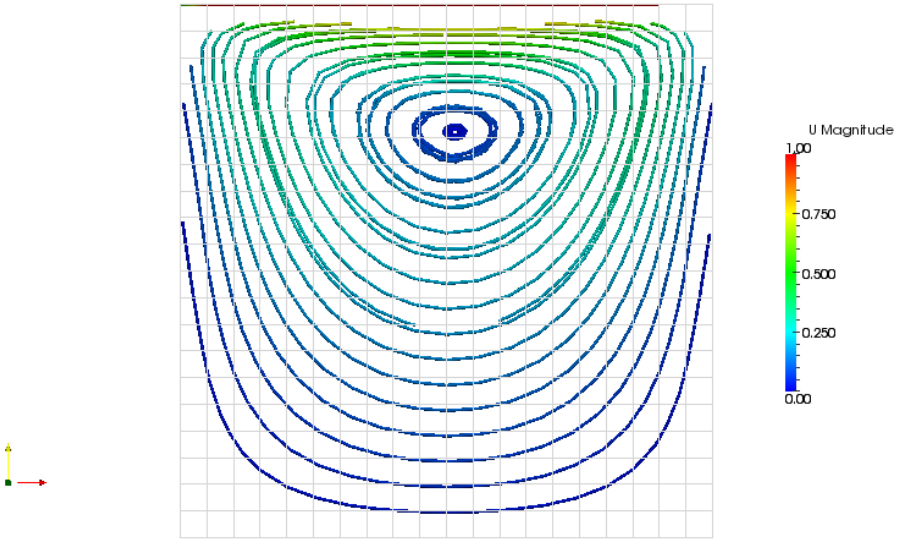


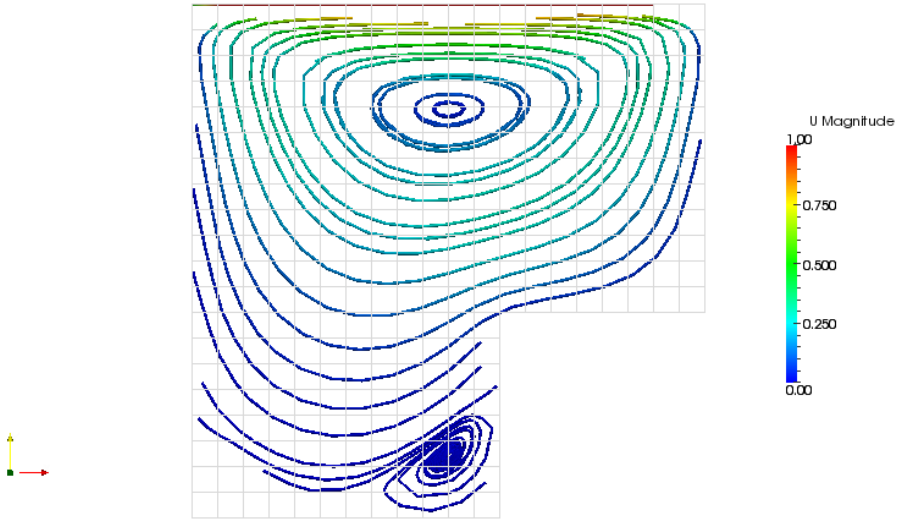


The velocity plot for the cases of cavity(upper left), cavityFine(lower left), and cavityGrade(right).

The *cell centre* and *Glyph* filters have been used.

Different geometry: cavity, cavityClipped

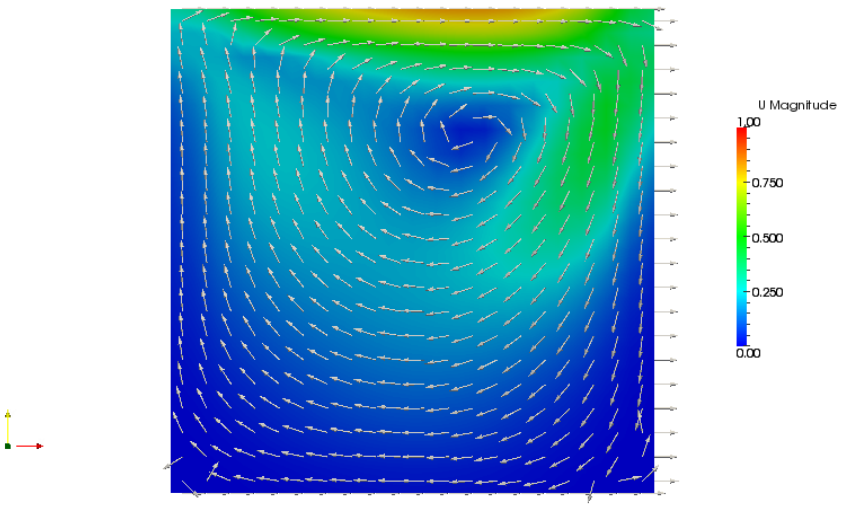


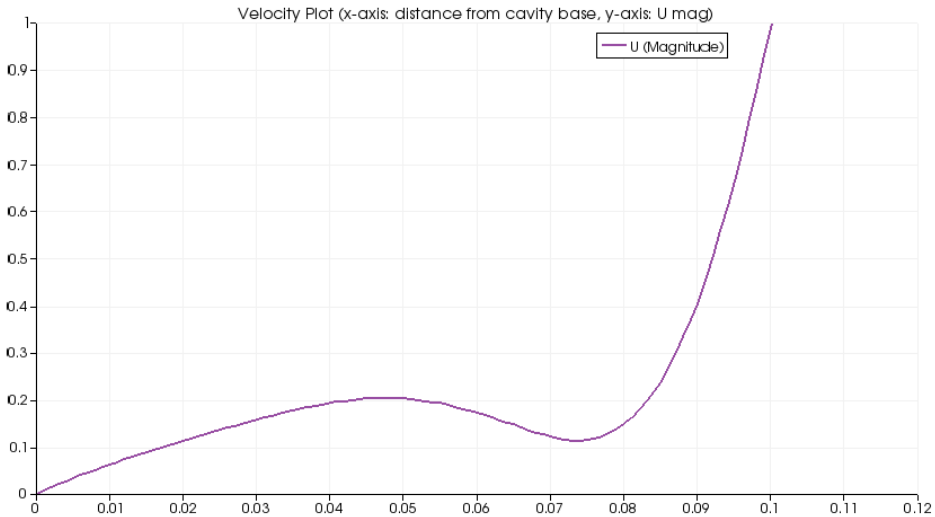


The different velocity stream plots for the cases of cavity(left), and cavityClipped(right).

The *streamTrace* and *tube* filters have been used.

Different Reynolds number: cavityHighRe

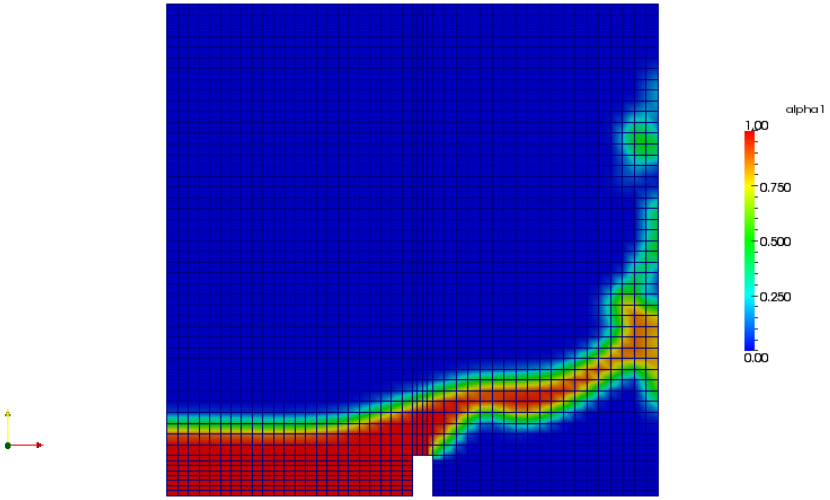


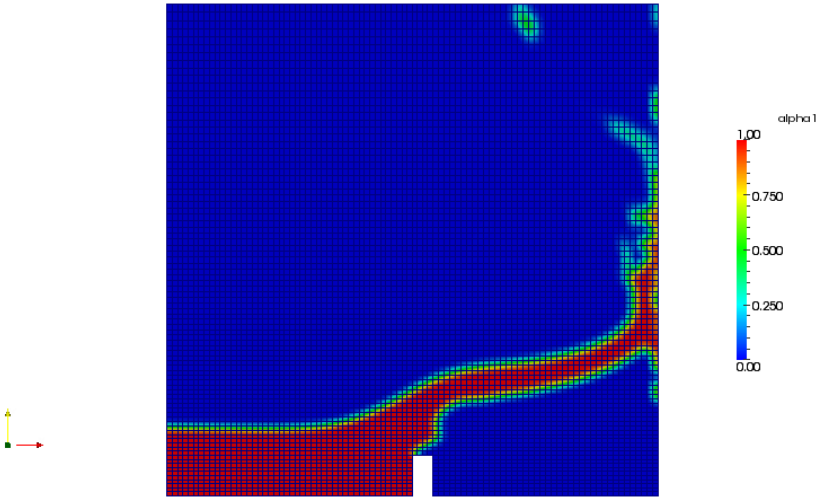


The velocity plots for the case of High Reynolds number cavity case.

The left graph used the *Glyph* filter.
The right graph applied the *Plot Over Line* filter.

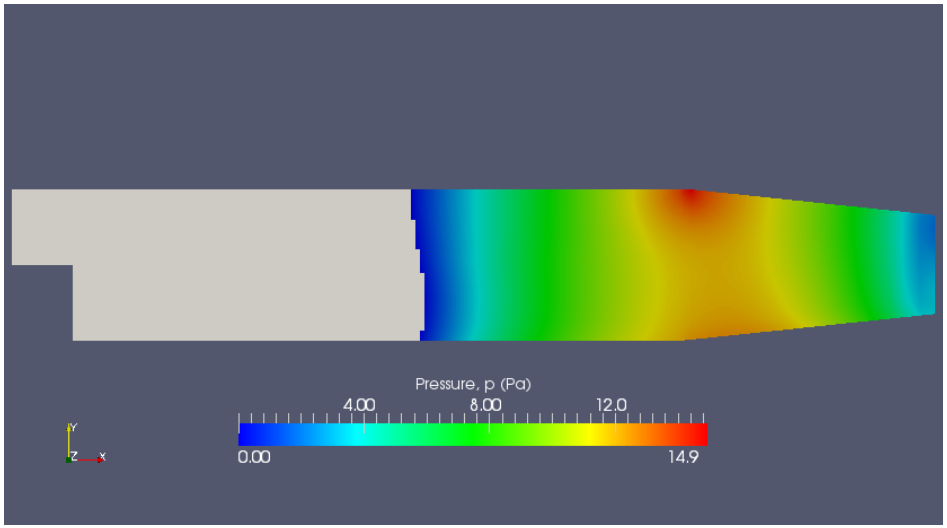
Different meshing: damBreak & damBreakFine

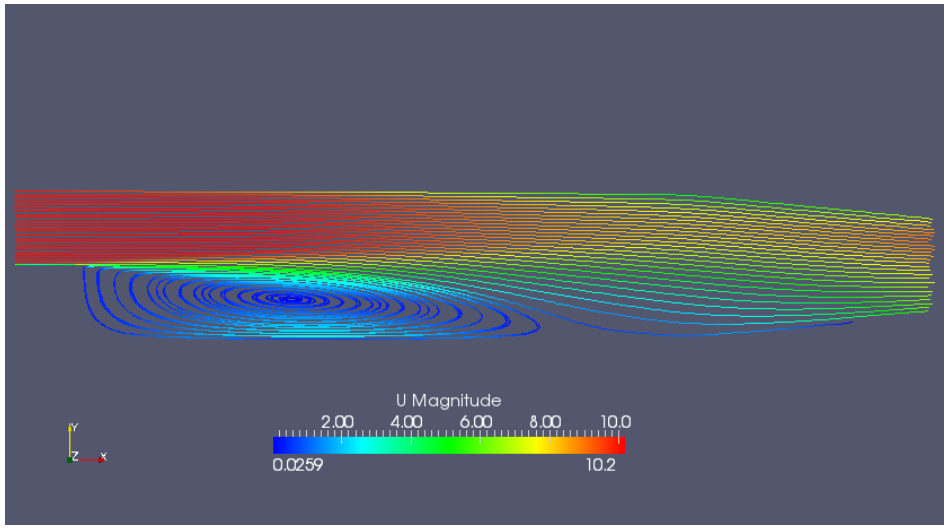




The plottng of phase fraction $\alpha 1$ in the cases of damBreak(left) and damBreakFine(right) both at time $t = 0.4s$.

pitzDaily





The left graph indicats the distribution of positive pressures after 1000 iterations, using the *threshold* filter with $p > 0$.

The right graph showes the streamline of velocity using *streamTrace* filter, after 1000 iterations.

Utilities - which has been used in the OpenFOAM tutorial cases

- *mapFields*
  This utility of mapping data from availble case to new case has been used in cavityFine, cavityGrade, cavityClipped and cavityHighRe cases.
- *setFields*
  This utility of setting values to specific regions has been used in the damBreak case.
- *decomposePar*
  This utility of decomposing domains for parallel computation has been applied in the damBreakFine case.
- *sample*
  This utility of helping to create graphs has been used in the plateHole case.

Utilities - 'new tutorial'

The *solidStress* utility

In some solid mechanic cases, it is often needed to determine the stress states from the displacements. Therefore, this *solidStress* utility is a postProcessing tool, which helps to calculate the stresses from displacements based on the following equation:

$$\boldsymbol{\sigma} = 2\mu\nabla\mathbf{U} + \lambda\mathbf{I}tr(\nabla\mathbf{U})$$

where,
σ : stress tensor;
U : displacement vector;
∇ : gradient operator;
I : identity tensor;
$tr()$: trace of a tensor;
μ, λ : material parameters for elasticity.

Calling this utility is by simply typing the following commands in the case directory:

```
solidStress
```

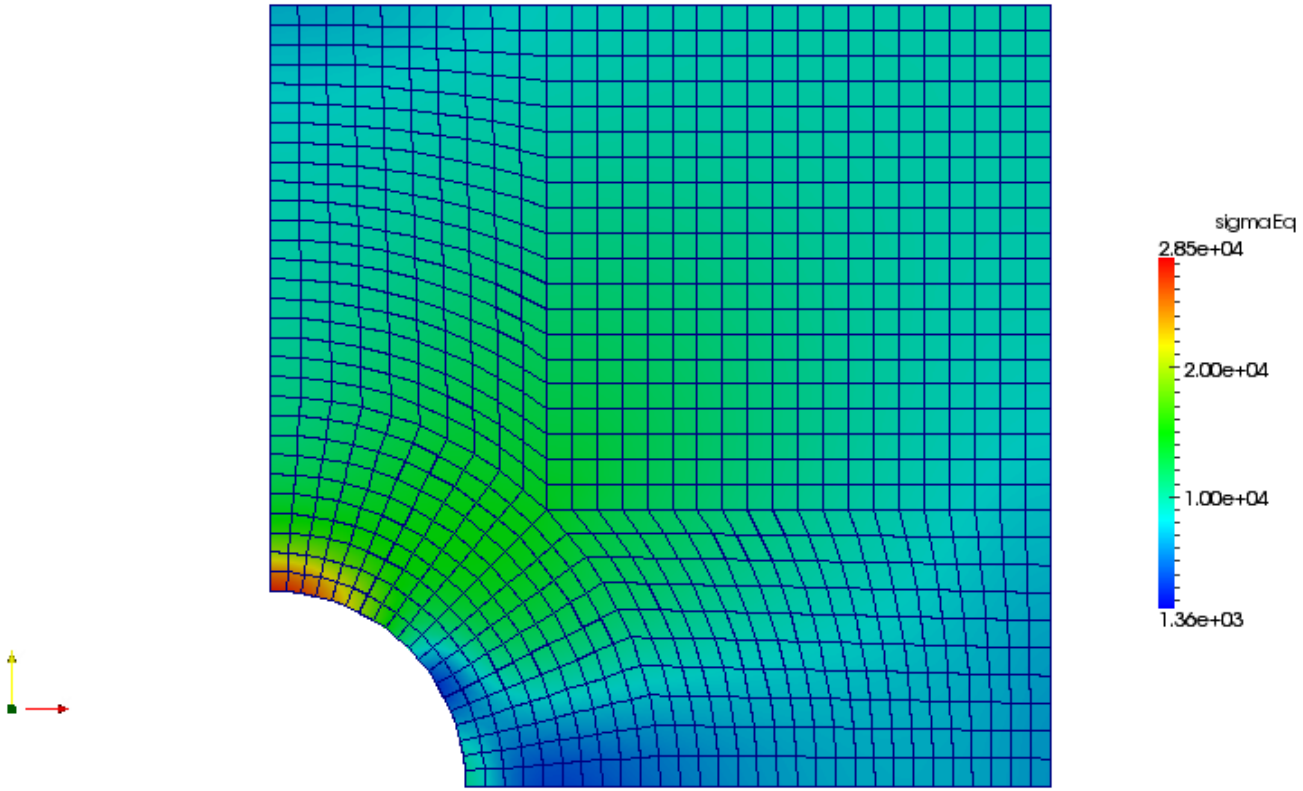
it will then calculate and output the six stress components and one stress invariant (σ^{eq}) into each time step file.

More details can be found in the following path:

```
$FOAM_UTILITIES/postProcessing/stressField/solidStress/solidStress.C
```

Utitlies - 'new tutorial'

- The *solidStress* utility



This graph shows the calculated stress invariant σ^{eq} in the plateHole case using *solidStress* utility. Where:

$$\sigma^{eq} = \sqrt{\frac{3}{2}\mathbf{s} : \mathbf{s}} \qquad \mathbf{s} = \boldsymbol{\sigma} - \frac{1}{3}tr(\boldsymbol{\sigma})$$

Utilities - 'new tutorial'

- *funkySetFields* & *groovyBC*
  In some cases of having non-uniform initial condition and boundary conditions, this two utilities can be quite helpful.

  - *funkySetFields* $\rightarrow$ non-uniform initial condition
  - *groovyBC* $\rightarrow$ non-uniform boundary condition

Utilities - 'new tutorial' - *funkySetFields*

How to install *funkySetFields*? Downloading + Compiling.

- Downloading (here I took the *funkySetFields* for openfoam-1.6 as an example, you need to download the right one acording to your OpenFOAM version), type:

```
svn checkout https://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/-Breeder_1.6/
utilities/postProcessing/FunkySetFields
```

- Compiling (make sure you are in the *funkySetFields* directory, where the *insertGenerated.sh is*), type:

```
./insertGenerated.sh
wmake
```

How to apply *funkySetFields*? Use command line or *funkySetFieldsDict*.

- using command line

```
funkySetFields [-case][-fields][-keepPatches][-expression][-time]
```

An example of command could be:

```
funkySetFields -case cavity -field p -keepPatches -expression "10.0*pos().y()" -time 0
```

Here, the pressure field has been set to have a linear increase along the vertical direction at the initial condition. And with the option *-keepPatches*, patches are preserved as they are previously defined. If you would like to modify the patches as well, simply ignore this option.

Utilities - 'new tutorial' - *funkySetFields*, continued

- *funkySetFieldsDict*

  In case of modifying multiple fields, you could alternatively gather all the commands into a dictionary named as *funkySetFieldsDict* in the *system* dictionary. An example of *funkySetFieldsDict* could be shown in the following:

```
expressions
(
            displacement1
            {
                field U;
                expression "mag(pos()-vector(0.05,0.05,0))/0.05";
            }
            pressure1
            {
                field p;
                keepPatches;
                expression "10.0*pos().y()";
            }
)
```

More details can be found in the link:

`http://openfoamwiki.net/index.php/Contrib_funkySetFields`

Utilities - 'new tutorial' - *groovyBC*

The *groovyBC* is a library for boundary-conditions, it helps to set the non-uniform boundary values and/or gradients simply by expressions instead of patchfields.

How to install *groovyBC*? Downloading + Compiling.

- Downloading (same as *funkySetFields*, download the right version for your own sake), type:

```
svn checkout https://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/
Breeder_1.6/libraries/groovyBC
```

- Compiling, in the *groovyBC* directory, type:

```
wmake libso
```

How to apply *groovyBC*? Quite simple...

- Use it as a boundary condtion just like those availble in OpenFOAM: *fixedValue*, *zeroGradient*, *fixedGradient*, etc.
- The only extra thing you need to do is, adding one line in the *system/controlDict* file:

```
libs ( "libgroovyBC.so" ) ;
```

Utilities - 'new tutorial' - *groovyBC*, continued

An example of using *groovyBC* for an inlet boundary:

```
inlet
   {
       type            groovyBC;
       valueExpression "(pos().z<=A*cos(-w*time())+0.5*k*A*A*cos(2*(-w*time()))) ? 1 : 0";
       variables       "l=5;A=0.1;g=vector(0,0,-9.81);k=2*pi/l;w=sqrt(k*mag(g));";
       timelines       ();
   }
```

- In this case we defined a Dirichlet-boundary condition with specified *value*. You can also change this option to *gradientExpression* to define a Neumann boundary condition with specified *gradient*.
- The option of *variables* defines a list with temporary variables separated by a semicolon.
- The *timelines* option in our case is empty, however it can have a list with subdictionaries that specify interpolation tables over time.

More details can be found in the link:

`http://openfoamwiki.net/index.php/Contrib_groovyBC`