Project for ME7310

Yuxuan Liu #001230010

Problem of interest: Pipe flow is one of the most important branches of hydraulics and fluid mechanics and has been studied decades. Figuring out pressure loss of fluid passing through pipe with different shapes is essential for building pipe system for variety applications in industry. This report produces a type of 2D incompressible unstructured solver to solve N-S equation in complex geometry to study the pressure loss of pipe with curved shape.
Geometry:

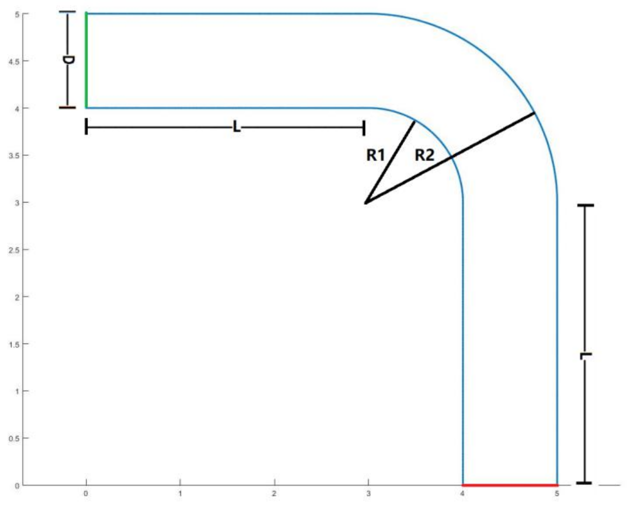


Fig 1. A type of joint, α = 90° ， R/D = 1.5

L	R1	R2	D
5.0	1.0	2.0	1.0

Table 1. Size of the mesh

Mesh: Unstructured mesh will be used to fit the curved pipe shape in the whole domain. Denser grid will be generated near the wall to get better accuracy in simulating the low-pressure area and reversed flow zone.

Boundary conditions: The green boundary will be an inlet boundary condition where a constant velocity of U is injected into the domain. The red boundary is a constant pressure outlet boundary condition. The blue boundary is wall boundary condition.

Type of analysis: I will collect the pressure from scatterpoints distributed in inflow region and outflow region of the geometry and compare them with the result from FLUENT.

Numerical Methods

In this part I will focus on explaining the interpolation methods I used to imply pressure correction, surface flux and boundary conditions for SIMPLE algorithm for unstructured mesh.

This solver deals with incompressible flow which governing equations are:

$$\frac{\partial u_i}{\partial t}+\frac{\partial u_i u_j}{\partial x_j}=-\frac{\partial p}{\partial x_i}+\upsilon\frac{\partial^2 u_i}{\partial x_j x_j},\frac{\partial u_i}{\partial x_i}=0 \tag{1}$$

Currently I have developed steady solver and fully-implicit transient solver. Both of them use same discretization method and interpolation methods come from [1]. Basically, the solver has cartesian velocity and pressure stored at cell center, surface normal flux stored at cell faces.

For the convection term, surface velocity is interpolated from:

$$\phi_{\text{face}}=\frac{\phi_{cv}+\phi_{nbr}}{2} \tag{2}$$

And assume the summation of all surface fluxes equal to zero to eliminate the cumulative error.

$\sum_{\text{faces}}\phi_{face}v_n A_f=0$,for each cell (3)

$V_{cv}\frac{d\phi_{cv}}{dt}+\sum_{faces}\frac{\phi_{nbr}}{2}v_n A_f=0$,for convection term (4)

Implementation of boundary conditions should developed from this formula strictly, if not, it might be possible to run the simulation but it is easily to blow up or get the false flow pattern.

Diffusive term is easily to implement because SIMPLE allows an explicit cross-diffusion term although the momentum equation is solved implicitly.

The pressure gradient term is harder to implement because it requires to do gradient reconstruction to minimize the error from interpolation. Same operation is also done after solve the pressure correction equation to reconstruct the pressure correction value at cell center.

It is possible to run the simulation without doing gradient reconstruction but simply interpolate the gradient from pressure at face center, for this situation reasonable solution can be observed but burrs and unphysical solution can be observed near boundaries.

The pressure gradient at cell center is reconstructed from pressure at cell and neighbour cells using Least-Square:

$$\text{Minimize:}\begin{bmatrix} N_{x1} & N_{y1} \\ N_{x2} & N_{y2} \\ N_{x3} & N_{y3} \end{bmatrix}\begin{bmatrix} \frac{\partial p}{\partial x}_{icv} \\ \frac{\partial p}{\partial y}_{icv} \end{bmatrix}A_f=\begin{bmatrix} \frac{\partial p}{\partial n}_{face1} \\ \frac{\partial p}{\partial n}_{face2} \\ \frac{\partial p}{\partial n}_{face3} \end{bmatrix}A_f \tag{5}$$

Capital “N” is used to denote the direction of face normal should point outward of the control volume.
The pressure gradient at cell face is interpolated as:

$$\frac{\partial \mathrm{p}}{\partial n} = \frac{p_{icv2} - p_{icv1}}{\delta_f} \qquad (6)$$

By specifying the storage of link_face_to_cell and only use 1 loop to go through all the faces instead of going through all faces for each cell will be much cheaper to do interpolation like this.
After solving the momentum equation, a guessed surface flux will be interpolated simply using:

$$\mathrm{v}_n = (\frac{u_i^{icv1} + u_i^{icv2}}{2}) n_i \qquad (7)$$

For pressure correction equation, same interpolation method ((5),(6)) is applied for pressure correction gradient at cell center an face center.

$$\sum_{\mathrm{faces}} a_p (\nabla p')_f \cdot N_f A_f = \sum_{faces} v_n^* A_f, \quad a_p = \frac{1}{2}(\frac{V_{cv}}{(a_{u,p})_{cv}} + \frac{V_{\mathrm{nb}r}}{(a_{u,p})_{\mathrm{nb}r}}) \qquad (8)$$

Once get the pressure correction, the guessed surface normal flux will be corrected using pressure correction without under-relaxation to achieve surface flux conservation and participate in the momentum equation at next iteration.

$$\mathrm{v}_n = v_n^* - a_p (\nabla p')_f \cdot n_f \qquad (9)$$

The pressure correction gradient at cell center will be reconstructed using Least-Square method to update cartesian velocity for next iteration.

Boundary Implementation

WALL

Assuming ϕ is general flow variable, for cell attaches to wall boundary,

$\phi_{\mathrm{face}} = 0$. from velocity interpolation method we got $\phi_{\mathrm{nbr}} = 2\phi_{face} - \phi_{cv}$.

For diffusive term it is easy to implement and the way how J_f is implemented explicitly for boundary has very small influence on final solution so it is feasible not to tell interior face from boundary face when dealing with J_f.
For pressure term it is unnessary to have a pressure boundary when using Least-Square. If the pressure gradient is interpolated from neighbors directly, to avoid unphysical solutions it is necessary to treat the pressure at boundary faces as unknown variables and add them to pressure correction equation to solve.

VELOCITY INLET

The way to implement inlet boundary is same as wall.

PRESSURE OUTLET

The implementation of outlet became a huge issue for me. Implementing a fully-developed outlet boundary will blow up the whole simulation and the reason is still unknown. But it is feasible to implement a constant pressure outlet to avoid this problem and the final solution is still accurate.

To implement a pressure outlet, neumann boundary will be implemented in momentum equation for fully-developed assumption:

$$\frac{\partial \phi}{\partial n}_{\text{boundary}face} = 0$$

Then modify pressure correction matrix for cells attach outlet faces.

Solver Structure

In this part I will explain the structure of my code and how to run it briefly.

Main functions

Main script	Sub functions	Description
IcoSolver.m	GEOMETRY.m	Read mesh file includes verts and cells to generate topological informations which will be used in unstructured solver.
	Preprocessing.m	Assign boundary conditions required by solver and initialization.
	IterSettings.m	User can modify iteration settings for convergence criterion, under-relaxation factor, max iteration,etc
	UnstructuredSolver.m	Main function to solve steady incompressible flow in unstructured mesh using SIMPLE
	Postprocessing.m	Post-process result from solver to plot velocity magnitude, quiver graph for surface flux and pressure distribution.

Main function	Sub functions	Description
Preprocessing.m	PhysicalSettings.m	User can modify physical properties here, Reynolds number, gauge pressure,etc.
	BoundarySettings.m	User can modify boundary conditions here. Some default settings have been put into the function for pre-defined meshes in the folder.
	GeometricInfo.m	Pre-compute some geomotry infomations will be used in solver. No need to modify.
	Nearest.m	Use to define which cell will be fixed in solving problems with the absence of a pressure boundary. Default setting is the one closest to origin.

There are 3 versions of solver function: UnstructuredSolver.m,UnstructuredSolverAdvance.m and UnstructuredSolverTransient.m. Basically, the first one can only solve steady problem and has not been improved for performance. Second one has been modified to improve performance via direct mounting the sparse matrix ,reduce unnecessary computing and parallelize children functions. The third one is the transient version based on the second function.

Generally, there is no difference between UnstructuredSolver.m and UnstructuredSolverAdvance.m except computing speed.

All results below are produced by UnstructuredSolverAdvance.m.

Main function	Sub functions	Description
UnstructuredSolver.m	Cell2Verts.m	Interpolation from neighbour cells to vert.
	CrossDiffusionTerm.m	Calculating cross-diffusion term Jf for all faces.
	SurfNormFlux.m	Interpolate guessed cartesian velosity to get guessed surface normal flux.

Main function	Sub functions	Description
UnstructuredSolverAdvance.m/UnstructuredSolverTransient.m	ParallelCell2Verts.m	Parallel version
	ParallelCrossDiffusionTerm.m	Parallel version
	ParallelSurfNormFlux.m	Parallel version

Other functions

Main function	Sub functions	Description
BoundarySettings.m	Catchboundary.m	Use to assign boundary faces to specific bounday classes.i.e NBC(North Boundary) see comments in mat file for more details.
ReadAutosave.m		Read autosave.m generated by solver in current folder.
.../mesh/GridGeneration.m		Generate grid points for mesh generation.
.../duct/FLUENTCMP.m		Compare my result to FLUENT.
.../old_version/IcoSolverUns_backup20171129.m		First version of the solver. More comments.

List of pre-defined meshes(stored in main folder):

rectangle_denser.m
pipe.m
circular_-5_15_-5_5.m
naca0012.m
joint.m
joint_long.m (huge cells, directly load the geometry file saved in main folder instead of running GEOMETRY.m)

How to run the script

For pre-defined meshes:

1. copy the file name to IcoSolver.m
2. change physical settings in PhysicalSettings.m
3. change iteration settings in IterSettings.m
4. Choose which solver to use.(i.e choose UnstructuredSolverAdvance.m for steady problem)

For customized meshes, all information required customization are included in BoundarySettings.m,PhysicalSettings.m and IterSettings.m.

Accuracy
The interpolation method for cartesian velocity promises first-order accuracy, but in uniform grid it achieves second-order accuracy. For transient solver, fully-implicit time marching is first-order accuracy.

Solution
For problem stated before, I will do multiple simulations when Re = 10,100,400 ,700,1000 and 1500 to study how pressure loss changes with different Re number. To achieve relatively accurate result, the criterion of convergence is p_change<1e-7, u_inlet = 1.0.

Mesh
Mesh used in my solver is generated from *Mesh2d [2].* Structured mesh used in FLUENT is generated from ICEM and has been converted to unstructured mesh. ICEM mesh files are stored in .../ICEM.

	Mesh2D	**ICEM mesh**
Nodes	**6021**	**103600**
Elements	**11584**	**102883**
Minimum edge-length	**0.0234**	**0.002**
Maximum edge-length	**0.0707**	**0.056**

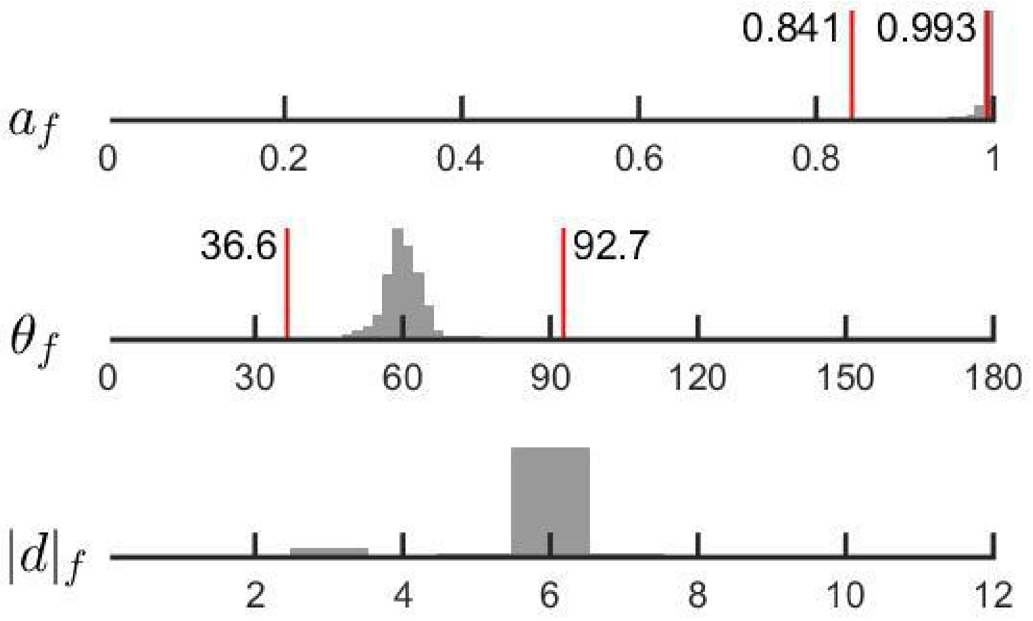


Fig 2:Mesh quality reported by *Mesh2d*

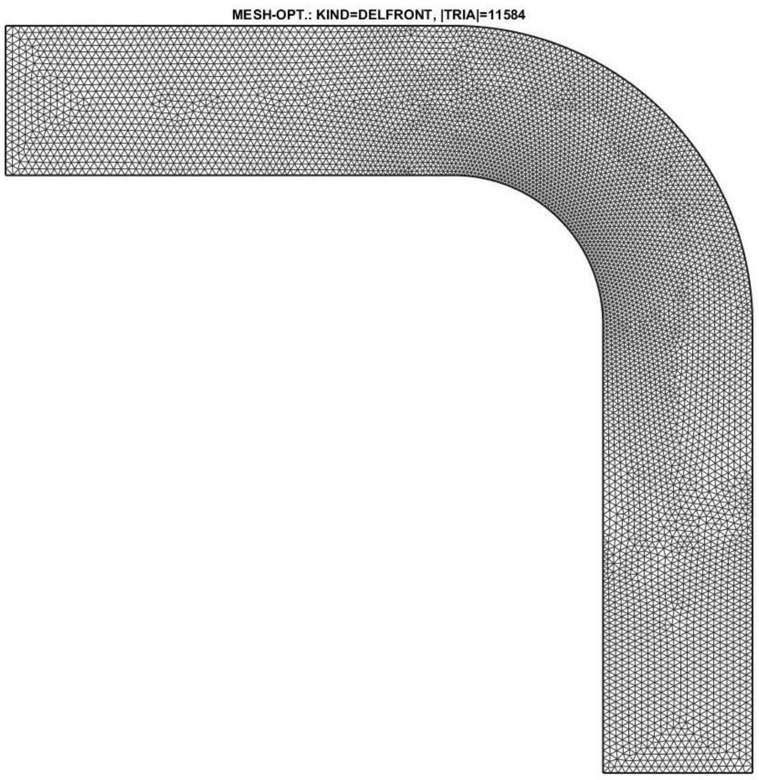
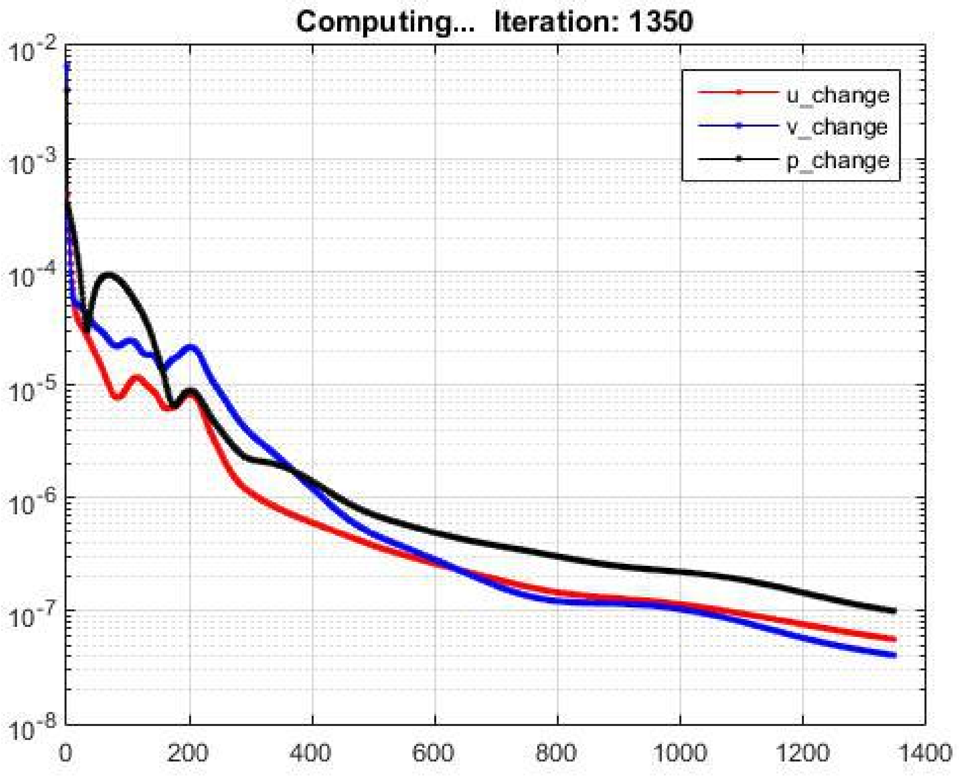
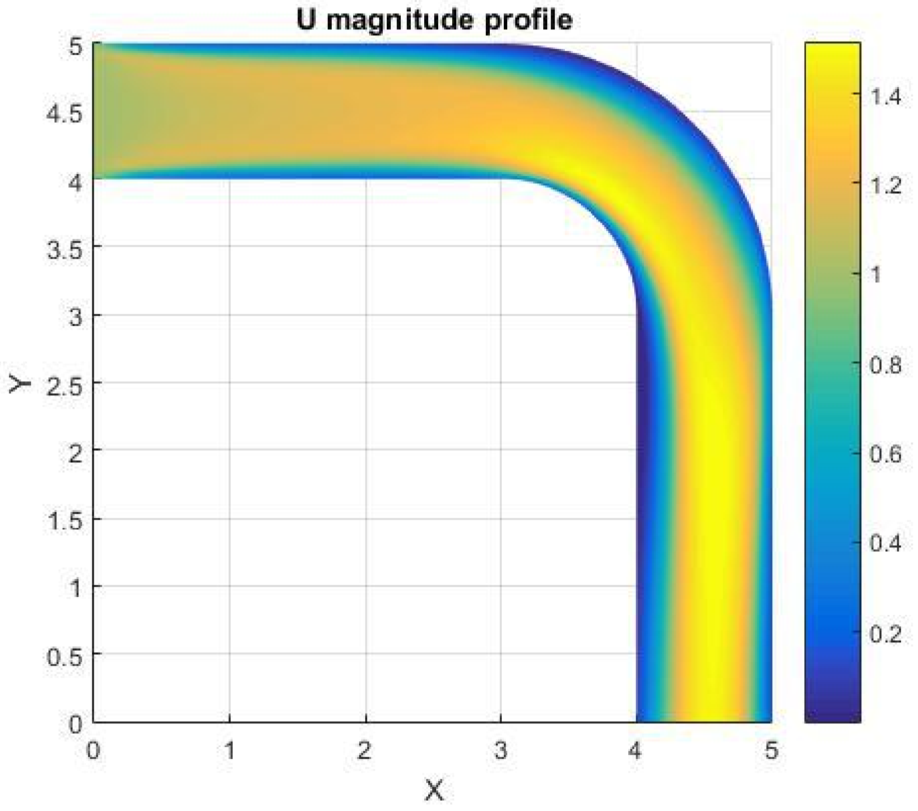


Fig 3:Mesh generated by *Mesh2d*

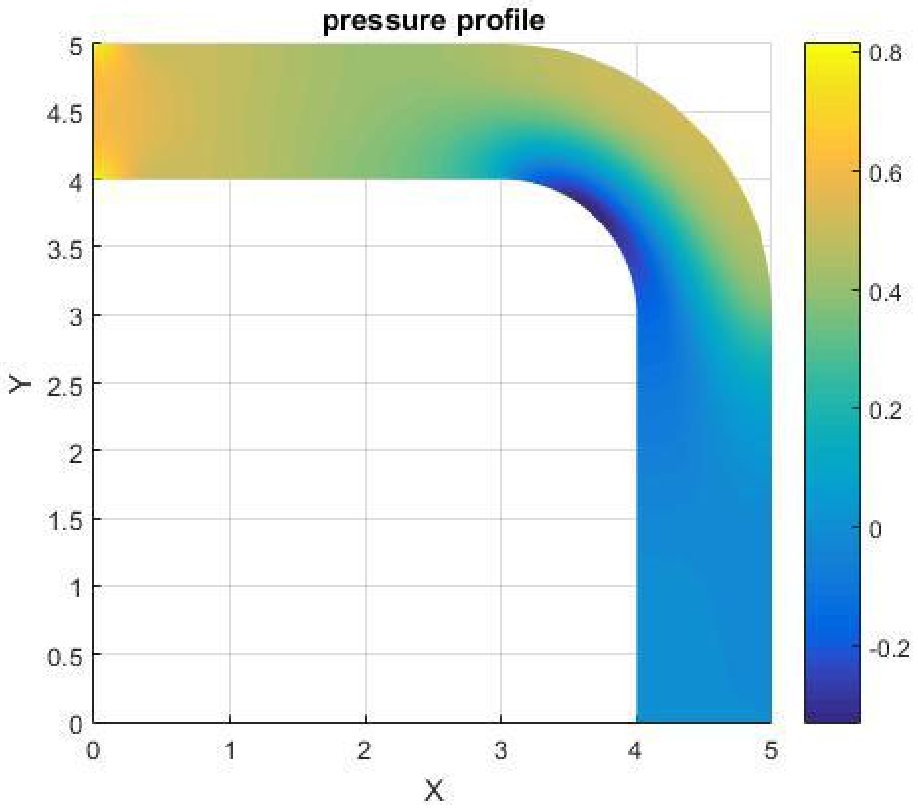
Result when Re = 400
Convergence



My result

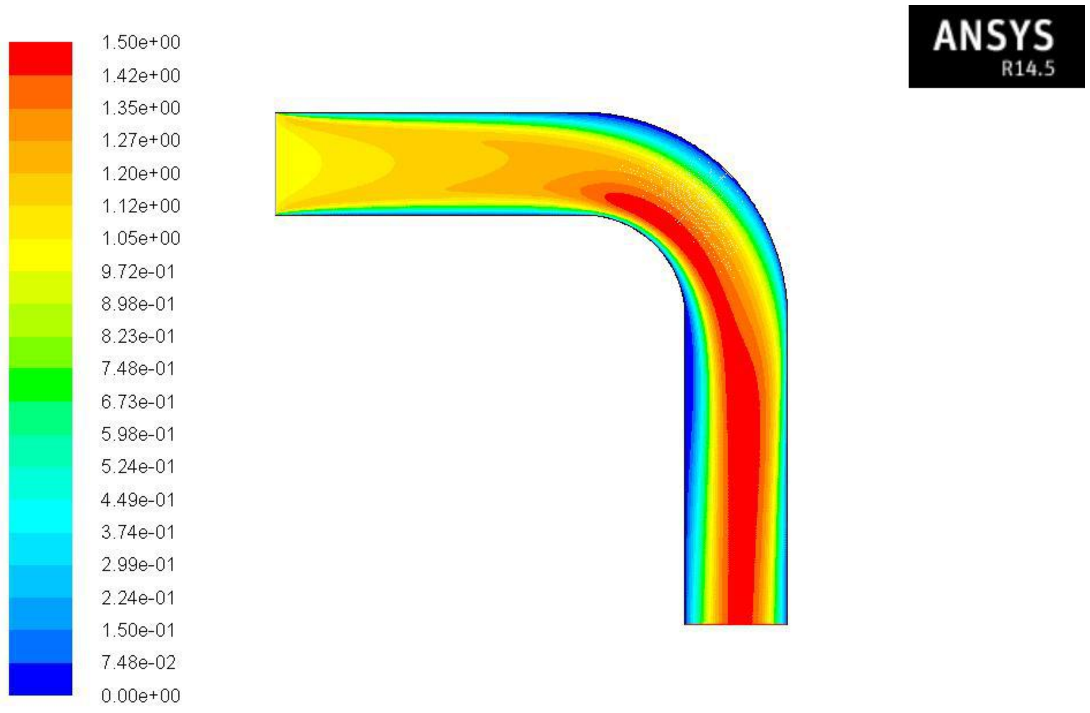


U magnitude profile
Y
X
0
1
2
3
4
5
0.5
1.5
2.5
3.5
4.5
0.2
0.4
0.6
0.8
1
1.2
1.4



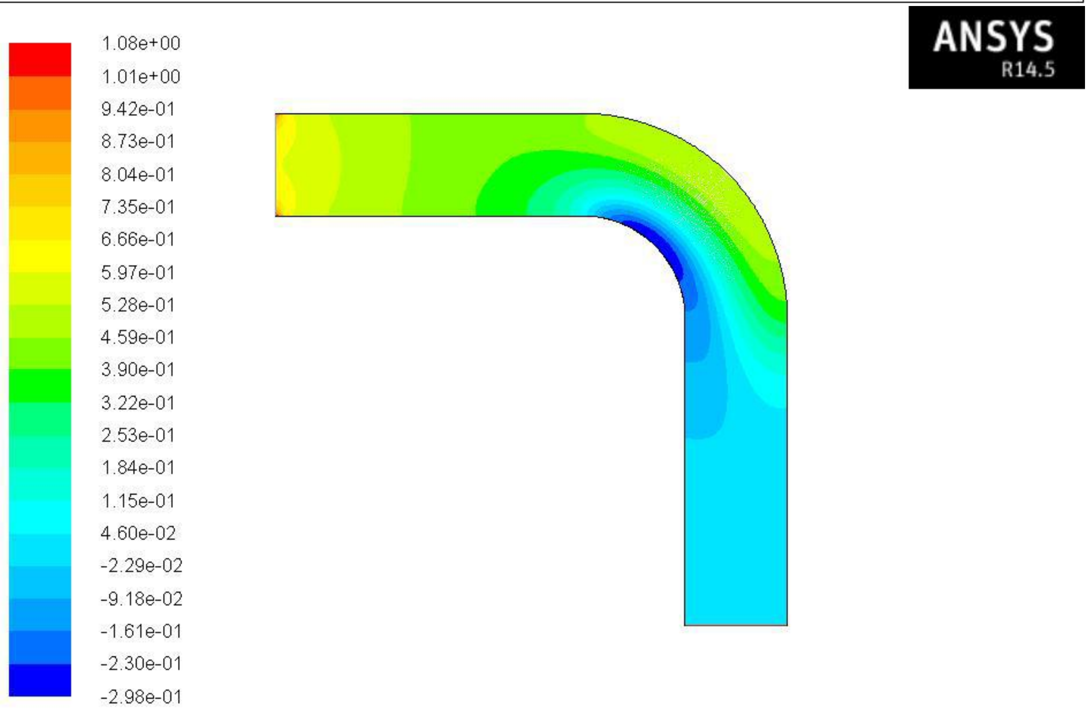
pressure profile
Y
X
0
1
2
3
4
5
0.5
1.5
2.5
3.5
4.5
-0.2
0
0.2
0.4
0.6
0.8

FLUENT



Contours of Velocity Magnitude (m/s)

Dec 05, 2017
ANSYS Fluent 14.5 (2d, dp, pbns, lam)



Contours of Static Pressure (pascal)

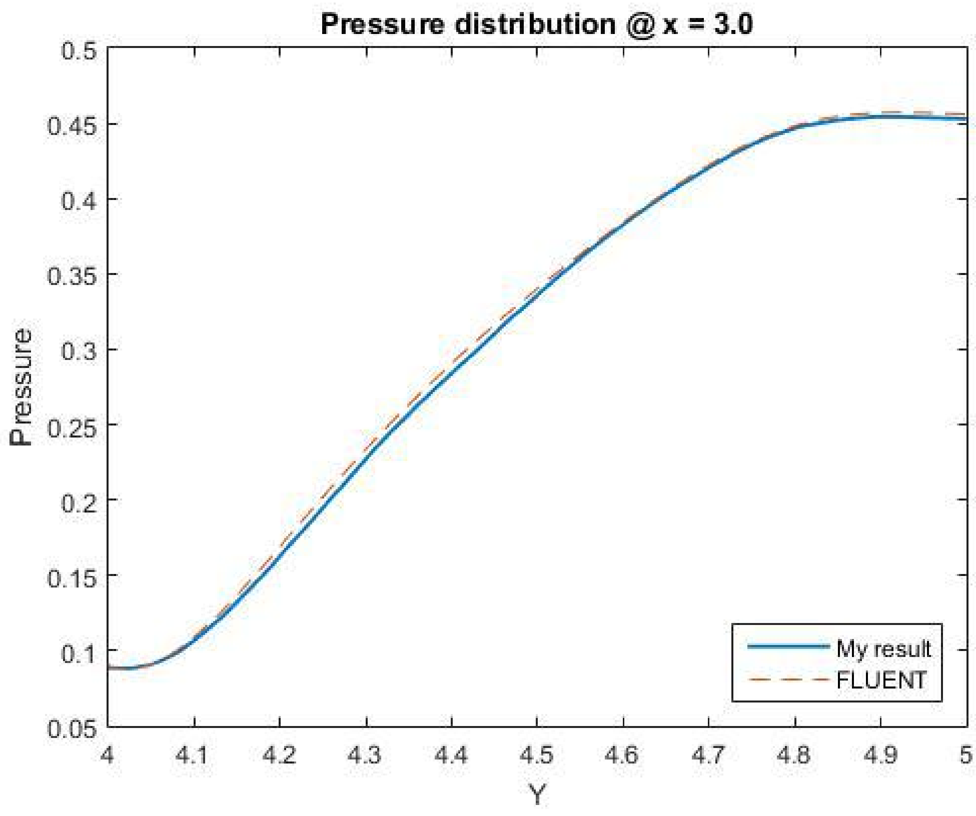
Dec 05, 2017
ANSYS Fluent 14.5 (2d, dp, pbns, lam)

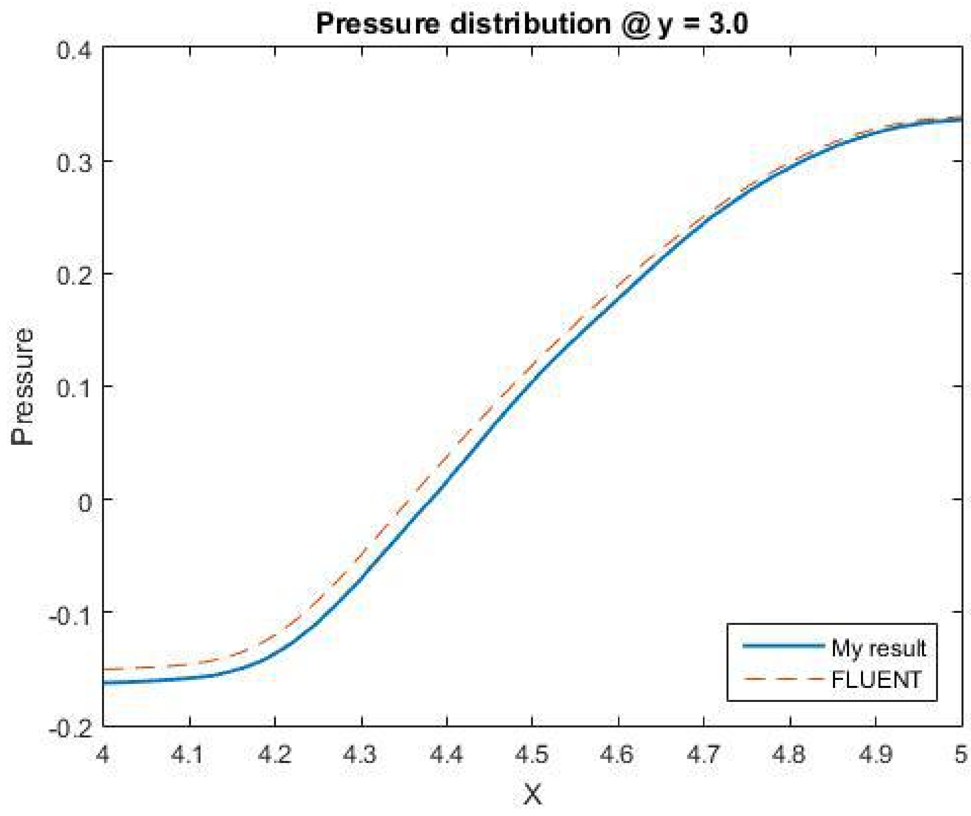
Pressure Loss Evaluation
To evaluate the pressure loss, I will distribute some scatter points uniformly along the vertical line (x = 3) located in inflow zone and horizontal line (y = 3) located in outflow zone, interpolating values at nearby cells and draw the graph. Average pressure value will be used to count the pressure loss.

	My solver	FLUENT	Absolute Error	Relative Error
Inflow	0.30868	0.30173	0.006959	2.3064%
Outflow	0.088359	0.097108	-0.0087489	-9.0094%
Net	0.22033	0.20462	0.015708	7.6767%

Table 2: Head loss, Re = 400, all values are produced by .../duct/FLUENTCMP.m

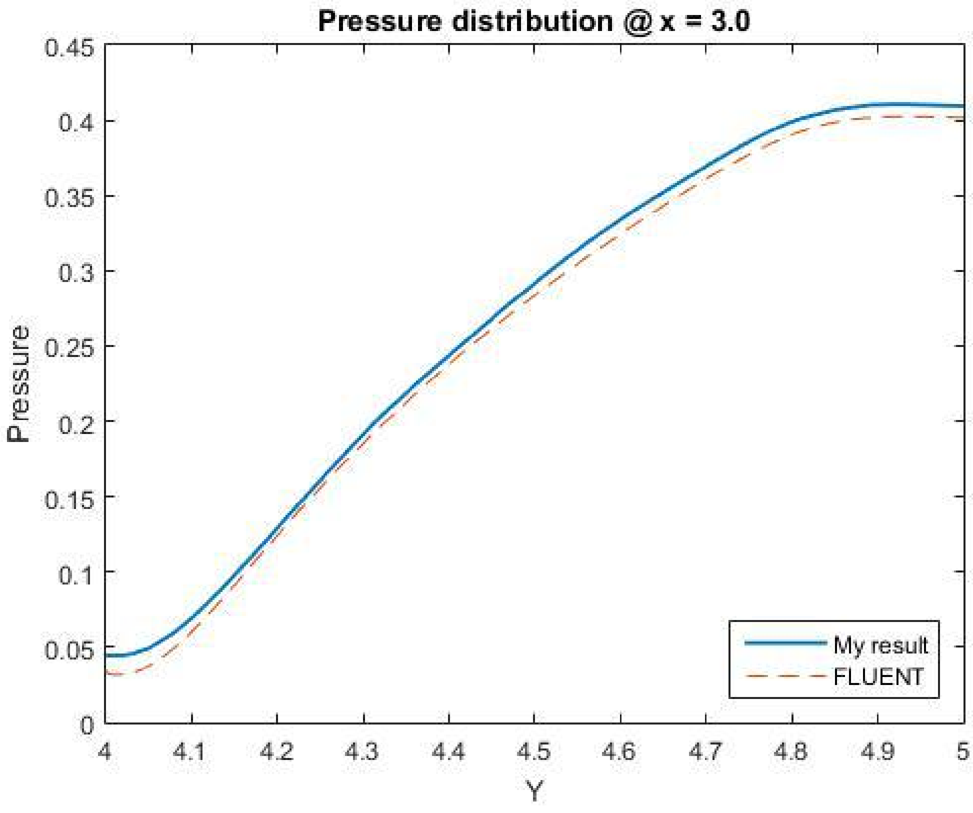
Since I use different mesh for my solver and FLUENT, the main source of error is considered to be mesh.

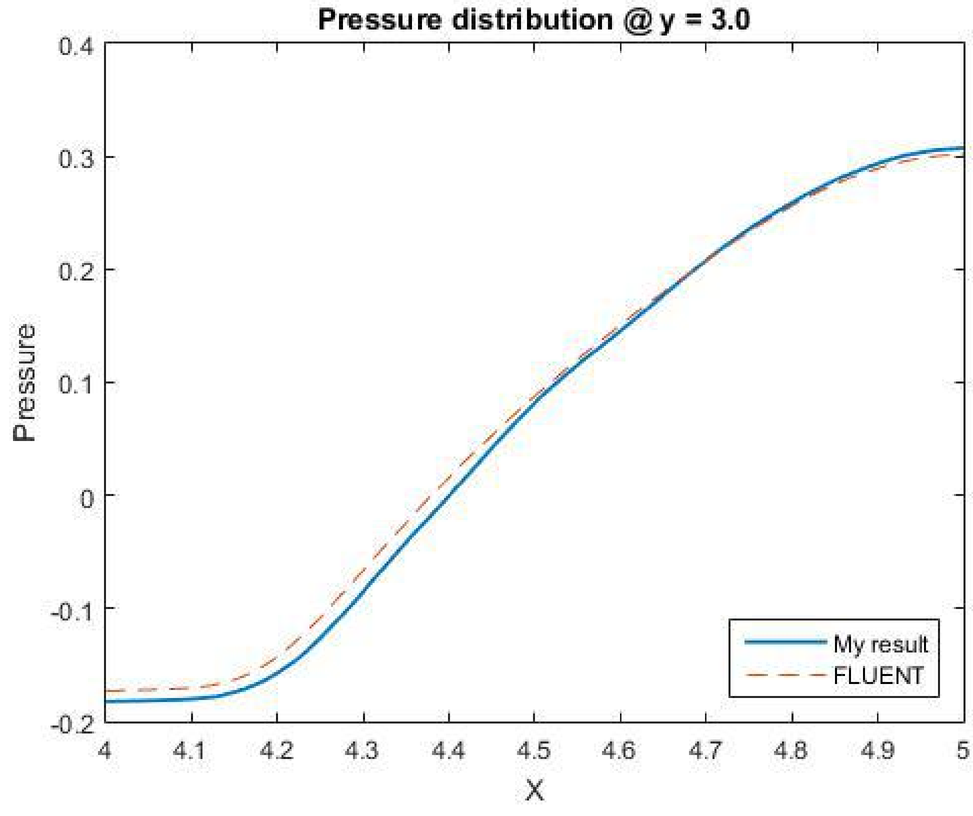




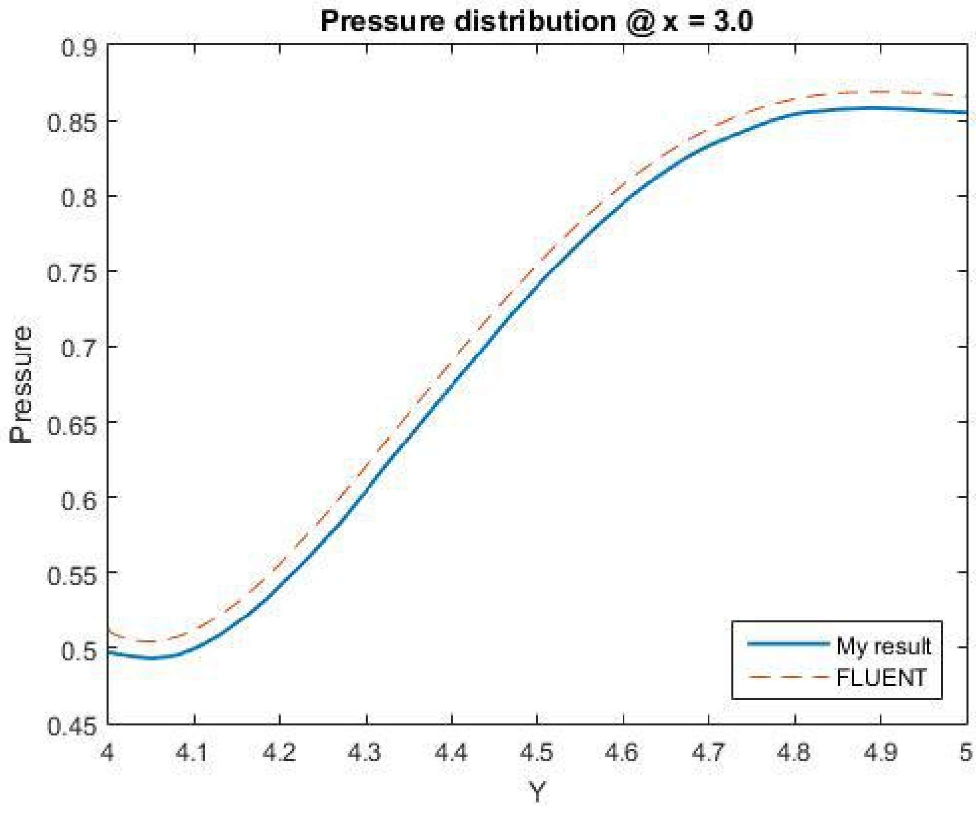
Re = 700

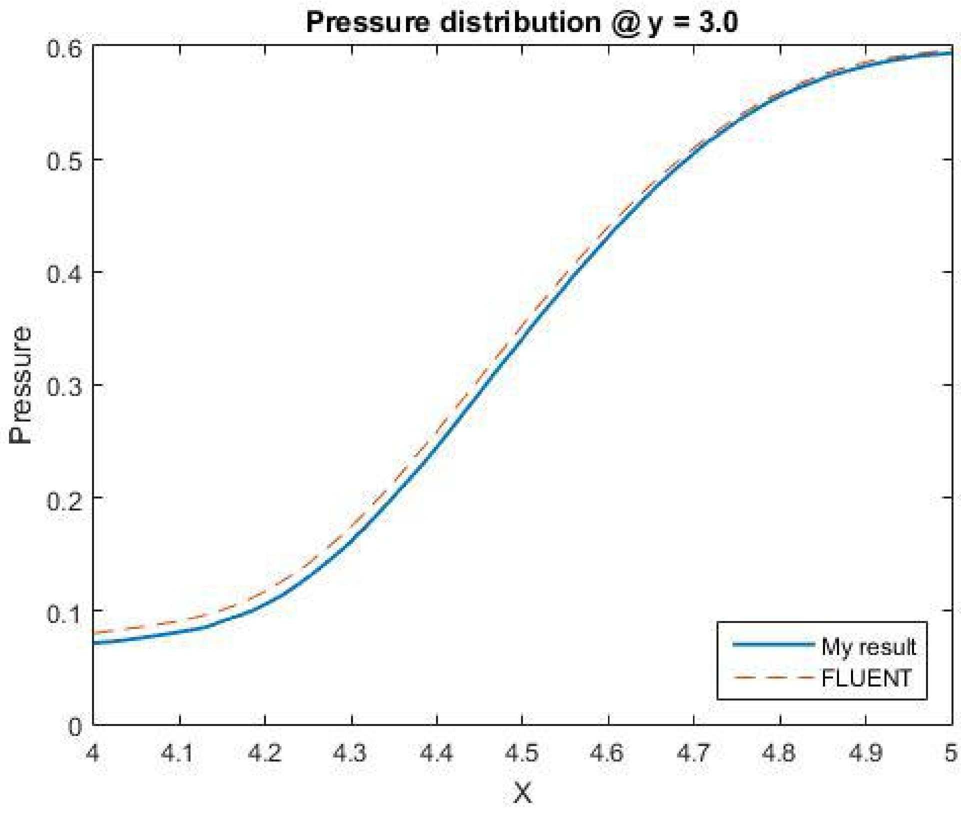
	My solver	FLUENT	Absolute Error	Relative Error
Inflow	0.266	0.24813	0.017873	7.2029%
Outflow	0.062839	0.066052	-0.0032129	-4.8642%
Net	0.20316	0.18208	0.021085	11.5804%


Pressure distribution @ x = 3.0
Pressure
Y
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4
4.1
4.2
4.3
4.4
4.5
4.6
4.7
4.8
4.9
5
My result
FLUENT


Pressure distribution @ y = 3.0
Pressure
X
0.4
0.3
0.2
0.1
0
-0.1
-0.2
4
4.1
4.2
4.3
4.4
4.5
4.6
4.7
4.8
4.9
5
My result
FLUENT

Re = 100

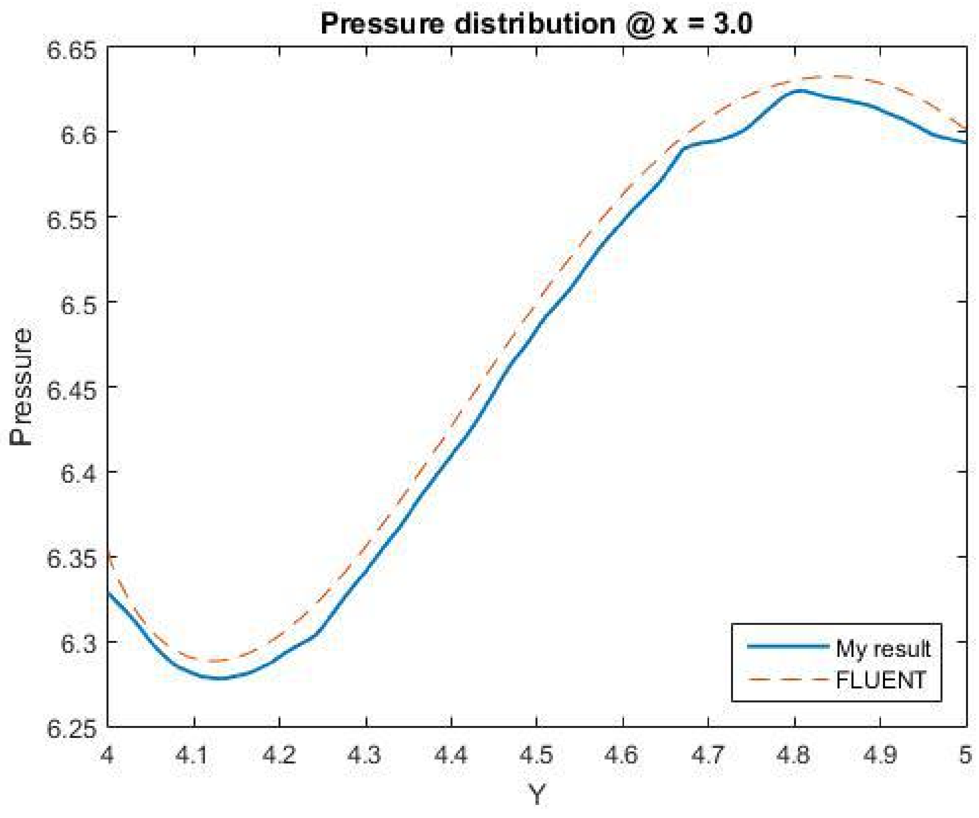

Pressure distribution @ x = 3.0
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
Pressure
4
4.1
4.2
4.3
4.4
4.5
4.6
4.7
4.8
4.9
5
Y
My result
FLUENT

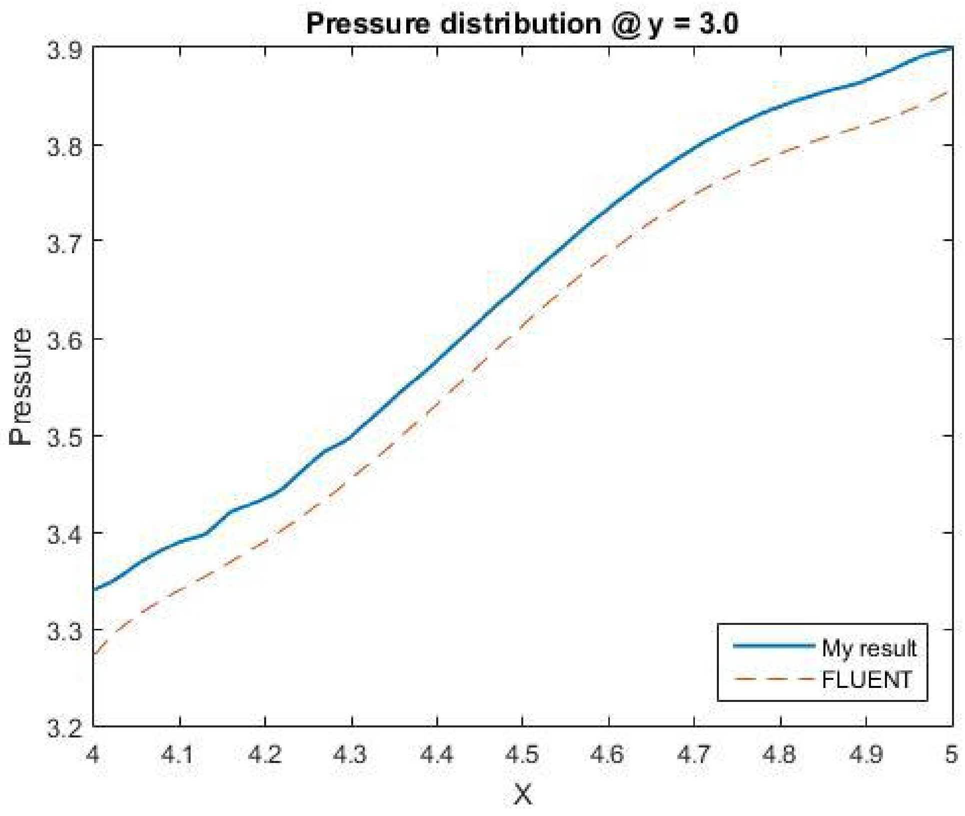

Pressure distribution @ y = 3.0
0.6
0.5
0.4
0.3
0.2
0.1
0
Pressure
4
4.1
4.2
4.3
4.4
4.5
4.6
4.7
4.8
4.9
5
X
My result
FLUENT

	My solver	FLUENT	Absolute Error	Relative Error
Inflow	0.70703	0.70984	-0.0028113	-0.39604%
Outflow	0.33424	0.3412	-0.0069534	-2.038%
Net	0.37279	0.36864	0.0041422	1.1236%

Re = 10

	My solver	FLUENT	Absolute Error	Relative Error
Inflow	6.4636	6.4726	-0.0089582	-0.1384%
Outflow	3.6417	3.5883	0.053375	1.4875%
Net	2.8219	2.8842	-0.062333	-2.1612%





Re = 1000

Flow becomes unstable and the outlet is too close to the reversed flow region which causes instability. I will change to another geometry which uses a double length of inlet and outlet part and leave other sizes unchanged. I will do fully-implicit time marching to t = 30s and compare to the result from FLUENT.

Mesh

	Mesh2D	**ICEM mesh**
Nodes	**67705**	**26400**
Elements	**132317**	**25823**
Minimum edge-length	**0.0077**	**0.002**
Maximum edge-length	**0.1545**	**0.17**
Δt	**0.02s**	**0.02s**
T_final	**30s**	**30s**
p_tolerance	**1e-8**	**1e-8**

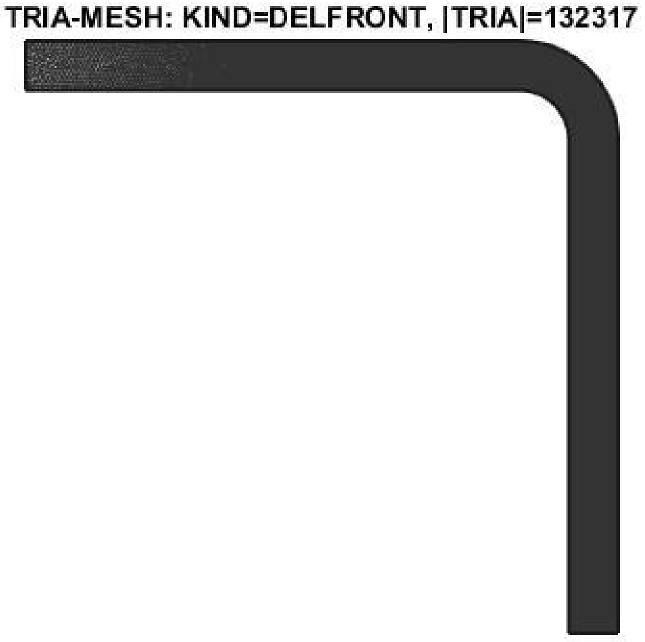


Fig 4:Mesh generated by
Mesh2d

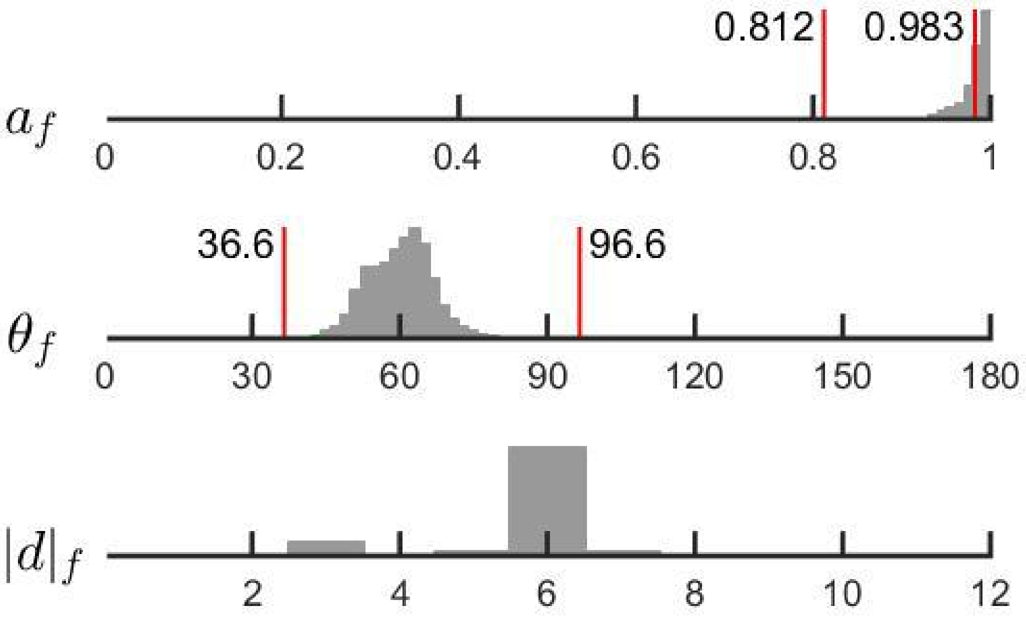
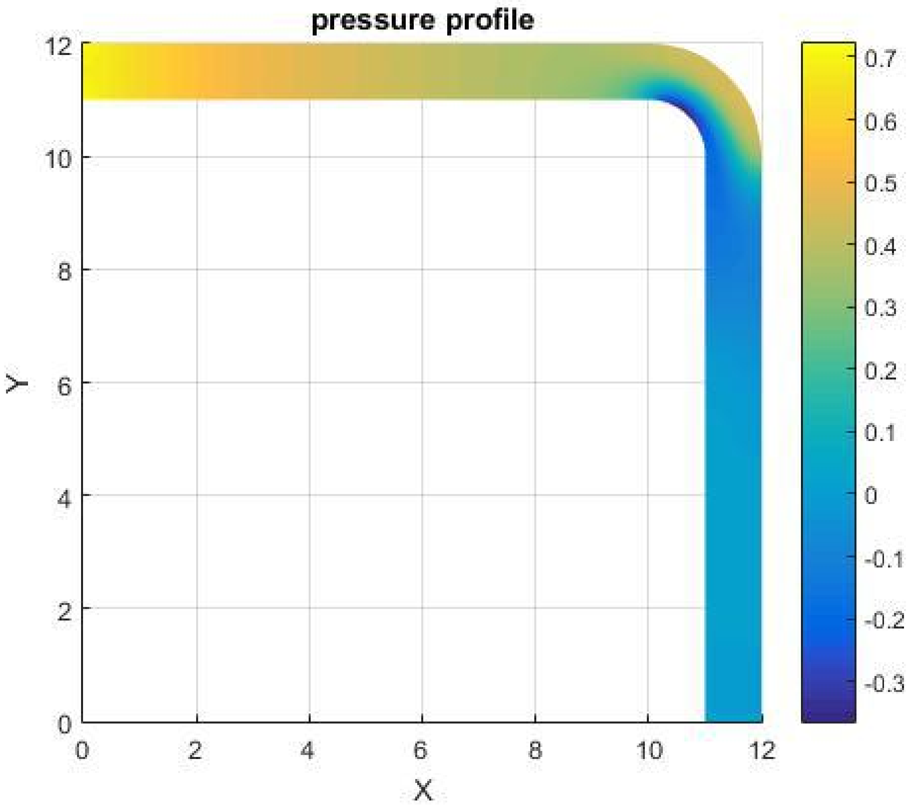
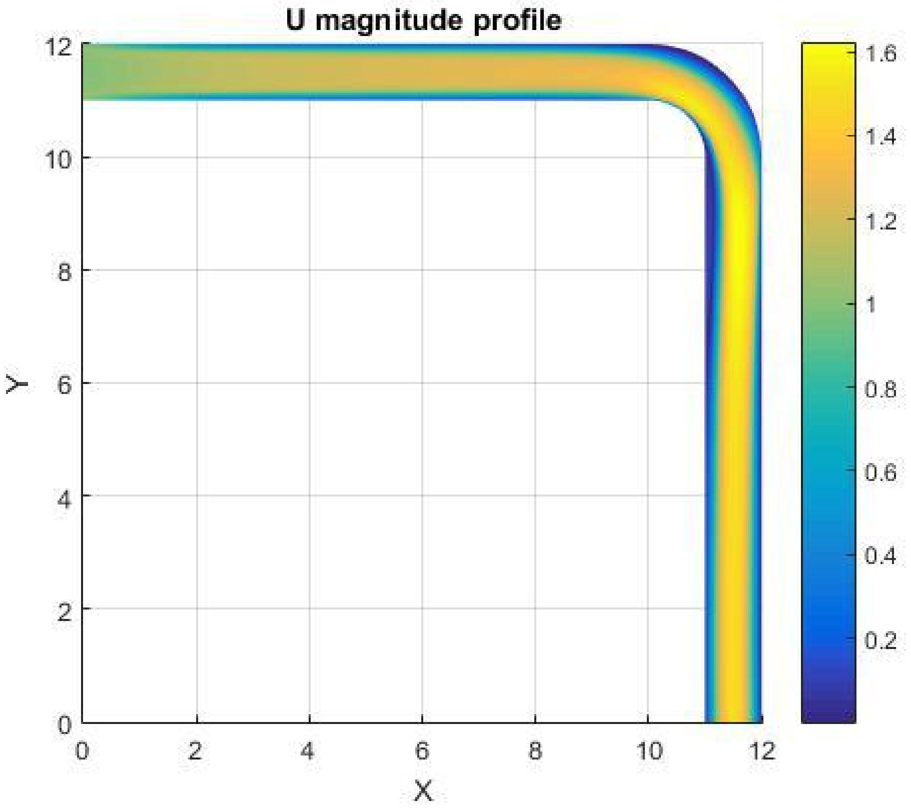


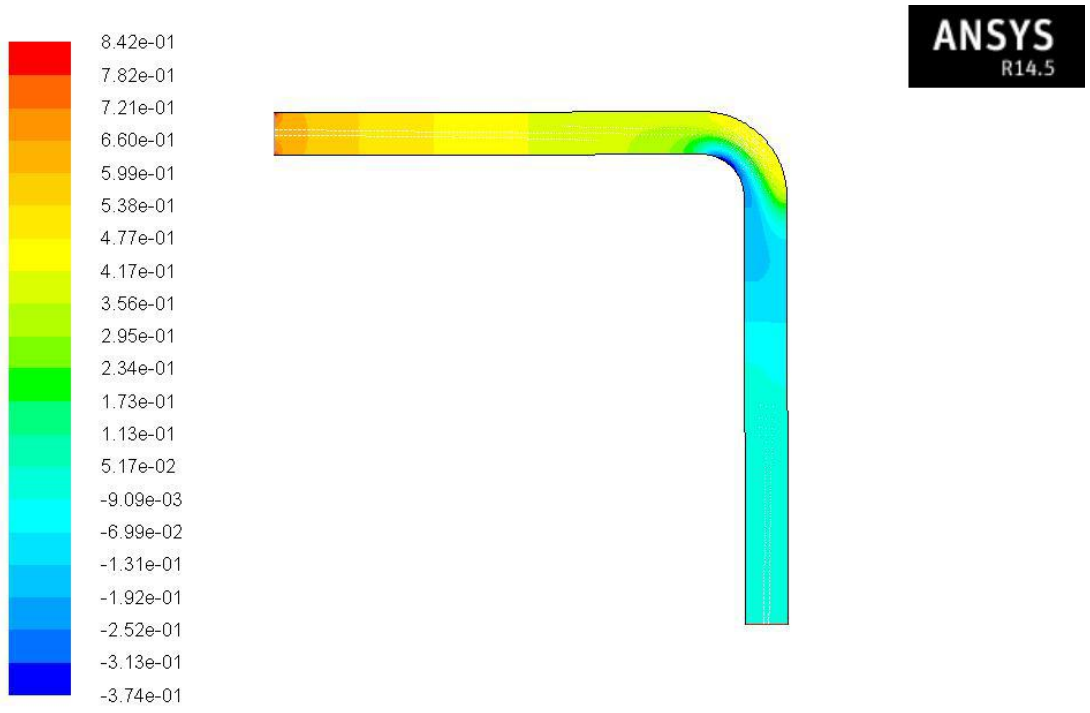
Fig 5:Mesh quality reported by *Mesh2d*

My result


pressure profile
Y
X
0
2
4
6
8
10
12
0
2
4
6
8
10
12
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3

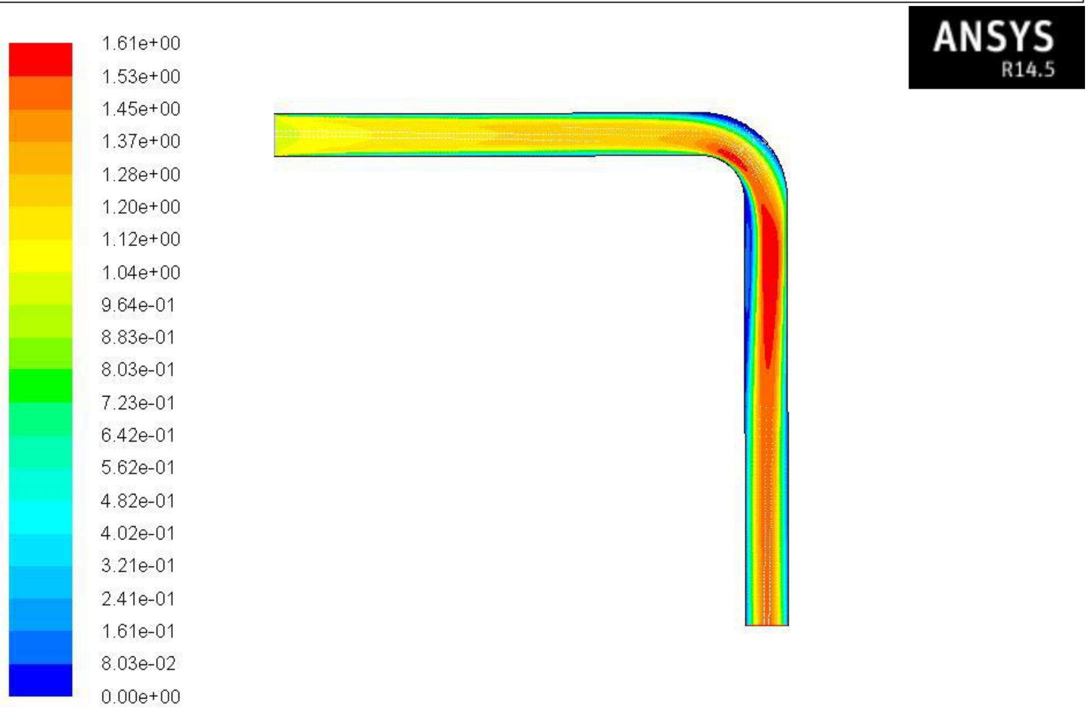

U magnitude profile
Y
X
0
2
4
6
8
10
12
0
2
4
6
8
10
12
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2

FLUENT



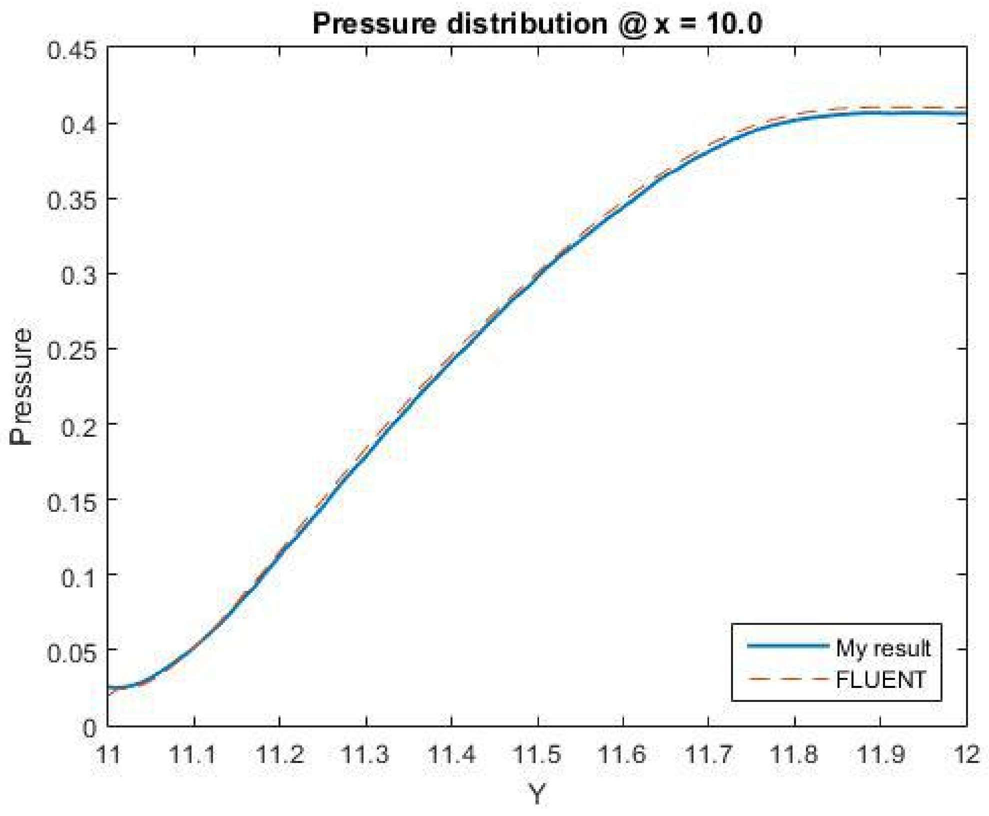
Contours of Static Pressure (pascal) (Time=3.0000e+01)

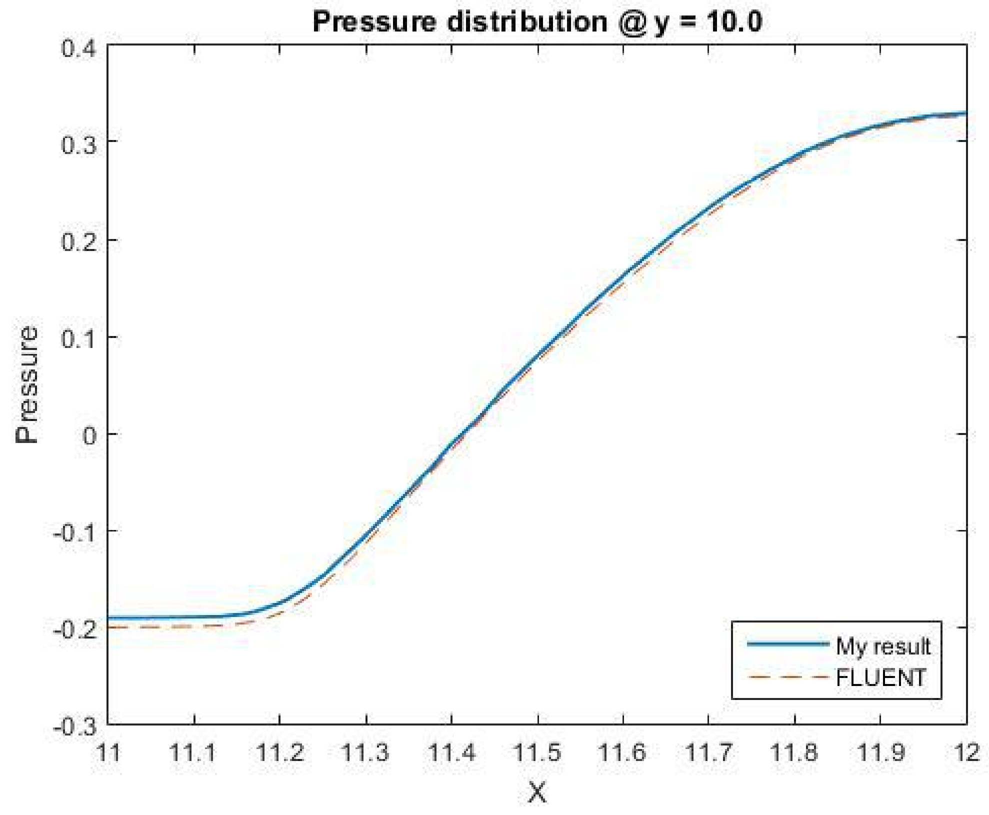
Dec 11, 2017
ANSYS Fluent 14.5 (2d, dp, pbns, lam, transient)



Contours of Velocity Magnitude (m/s) (Time=3.0000e+01)

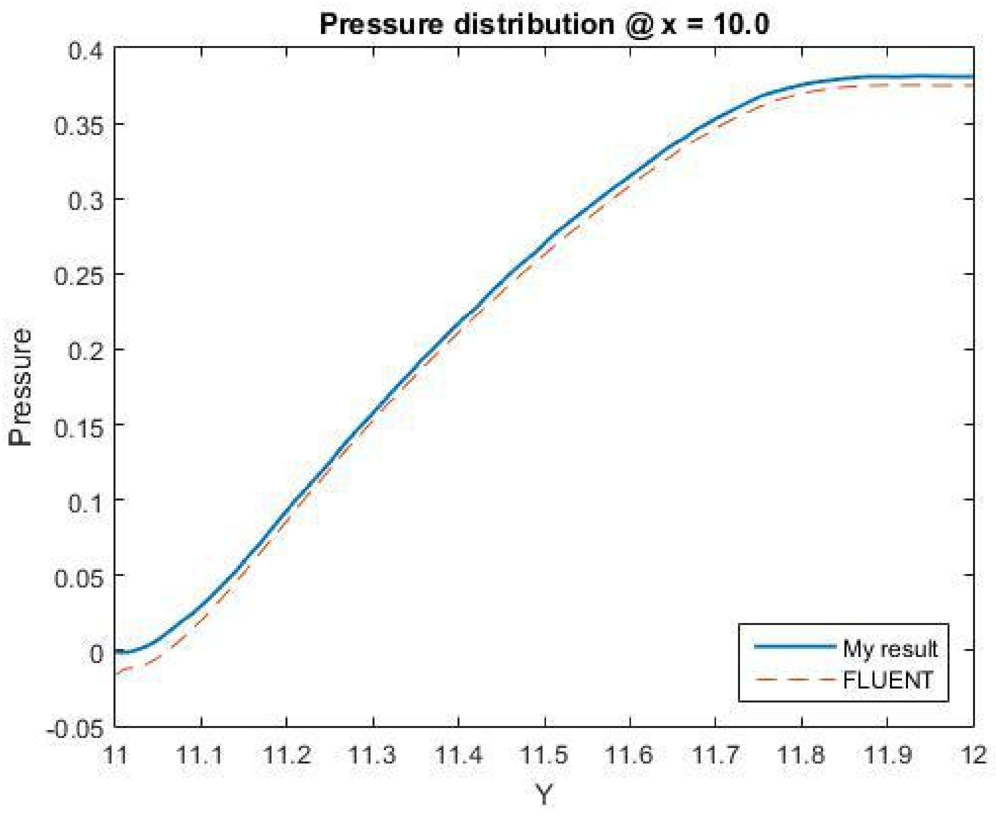
Dec 11, 2017
ANSYS Fluent 14.5 (2d, dp, pbns, lam, transient)

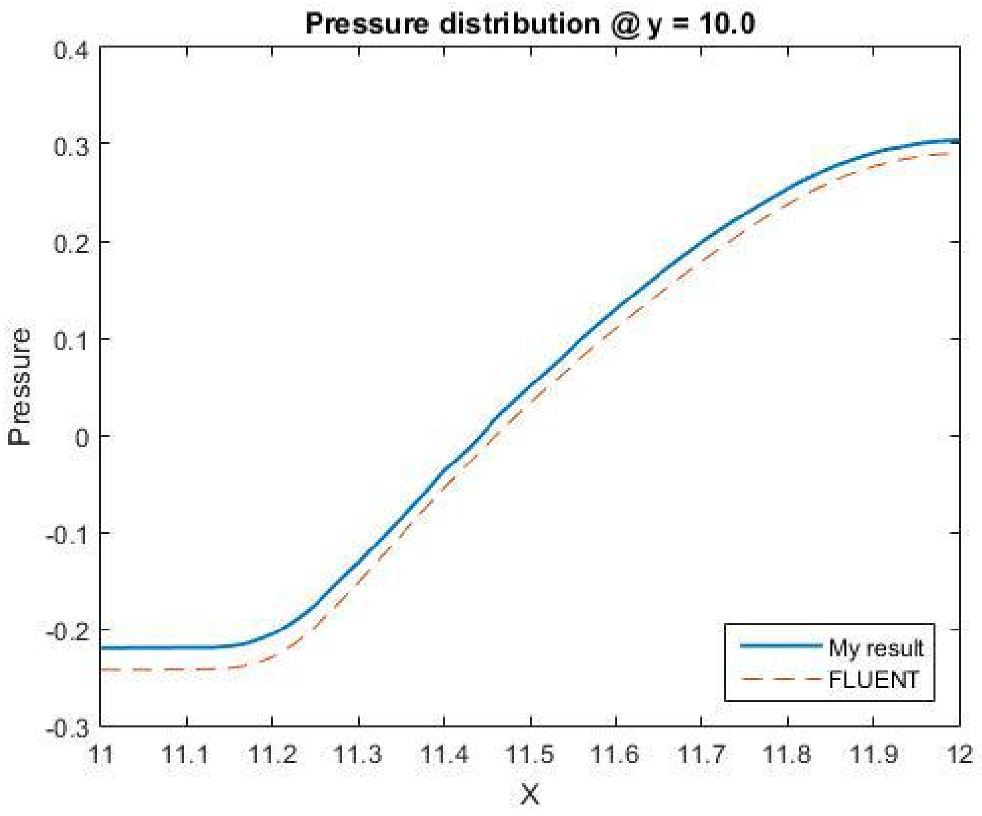

Pressure distribution @ x = 10.0
Pressure
Y
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
11
11.1
11.2
11.3
11.4
11.5
11.6
11.7
11.8
11.9
12
My result
FLUENT


Pressure distribution @ y = 10.0
Pressure
X
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
11
11.1
11.2
11.3
11.4
11.5
11.6
11.7
11.8
11.9
12
My result
FLUENT

	My solver	FLUENT	Absolute Error	Relative Error
Inflow	0.26254	0.25443	0.0081178	3.1906%
Outflow	0.066069	0.05866	0.0074084	12.6293%
Net	0.19647	0.19576	0.00070936	0.36235%

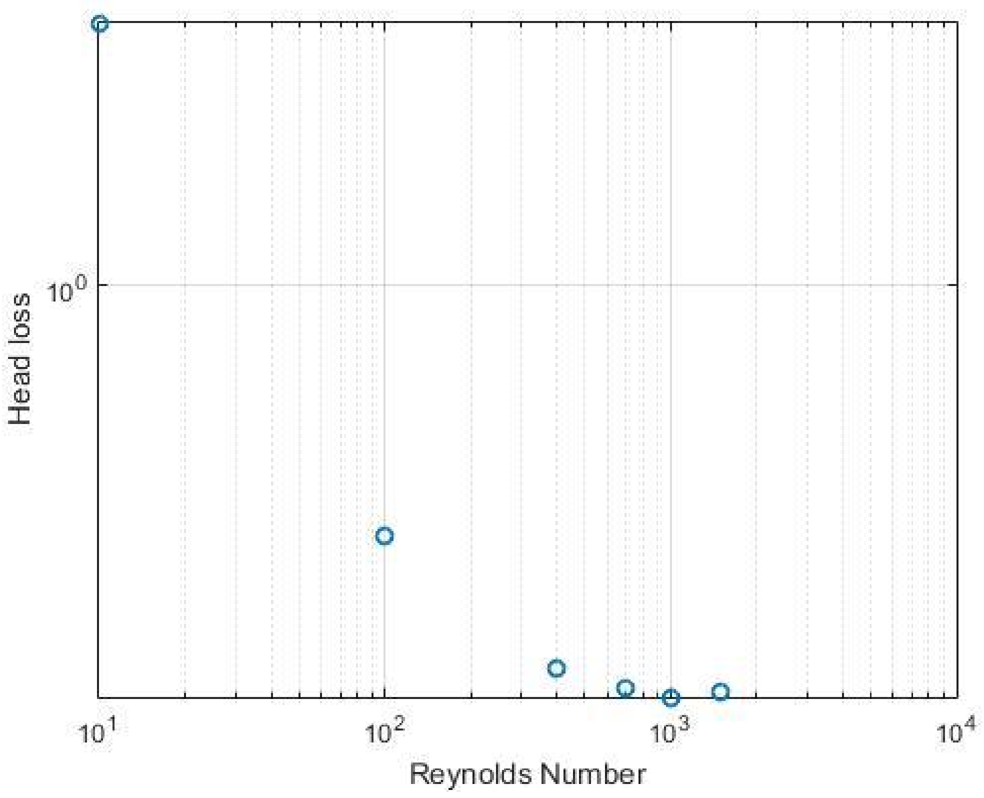
Re = 1500





	My solver	FLUENT	Absolute Error	Relative Error
Inflow	0.23766	0.21992	0.017737	8.0654%
Outflow	0.036842	0.017729	0.019114	107.8123%
Net	0.20081	0.20219	-0.0013764	-0.68074%

Summary



Head loss of flow passing through straight pipe inversely propotional to Reynolds number. However the pressure loss in curved pipe seems not to fulfill the relationship. The head loss decreases obviously slower when Re > 100 and reversed flow region forms close to inner face of outflow zone between Re = 100 and Re = 400.

Reference

[1] Mahesh K, Constantinescu G, Moin P. A numerical method for large-eddy simulation in complex geometries[J]. Journal of Computational Physics, 2004, 197(1): 215-240.
[2] Darren Engwirda, Locally-optimal Delaunay-refinement and optimisation-based mesh generation, Ph.D. Thesis, School of Mathematics and Statistics, The University of Sydney, September 2014.