OpenFOAM guide/The PIMPLE algorithm in OpenFOAM

< OpenFOAM guide (/index.php/OpenFOAM_guide)

Contents

1 New reference

If you want to check out the latest work from Tobias Holzmann about the usage of the PIMPLE algorithm, you can checkout his book »Mathematics, Numerics, Derivations and OpenFOAM(R) (https://holzmann-cfd.de/)«. In the book you will find an own chapter which explains the algorithm in detail. In addition reference [21] on www.holzmann-cfd.de explains the pressure-velocity coupling very well. The following explanation is an old blog entry from Tobias Holzmann which is not anymore kept up to date.

2 The test case

The test case it the pitzDaily tutorial that comes with the standard OpenFOAM installation/compilation. The case can be found in the pimpleFoam folder of incompressible fluids. The case was modified in order to allow a maximum Courant number of 3. The whole procedure was done with OpenFOAM-2.3 and is not anymore up to date.

3 Case A

At the beginning, we are using the tutorial case and change the fvSolution file. Within the pimple control directory we remove all entries. The outcome is a so called dummy:

```
PIMPLE
{
}
```

After starting the simulation we realize that the solver does not crash and runs. However, based on the change of Co, the solver will crash after a while. The following questions arise now:

1. why is the solver working without complaining that keywords are missing
2. in which mode is the solver running when there are no keywords

Question 1:

If we don't insert any keyword to the PIMPLE control dictionary, the solver is using the default value settings. To understand this, we have to go to the source code and being able to analyze the C++ code. While starting the application (pimpleFoam) we generate a new object of the pimpleControl class. Analyzing this peace of source code, we can see that there are pre-defined (so called lookupOrDefault) values. To demonstrate this, a snippet of the *pimpleControl.C* file is given below:

```
121 // * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * //
122
123 Foam::pimpleControl::pimpleControl(fvMesh& mesh)
124  :
125     solutionControl(mesh, "PIMPLE"),
126     nCorrPIMPLE_(0),
127     nCorrPISO_(0),
128     corrPISO_(0),
129     turbOnFinalIterOnly_(true),
130     converged_(false)
131 {
132     read();
```

As we can see, there are several keywords that are initialized with default values. After the initializing phase, the constructor is calling the »read()« function which checks if the user defined the necessary keywords to overwrite the default values. If no keyword is set, the default values are set (see listing below):

```
 39 void Foam::pimpleControl::read()
 40 {
 41     solutionControl::read(false);
 42
 43     // Read solution controls
 44     const dictionary& pimpleDict = dict();
 45     nCorrPIMPLE_ = pimpleDict.lookupOrDefault<label> ("nOuterCorrectors", 1);
 46     nCorrPISO_ = pimpleDict.lookupOrDefault<label>("nCorrectors", 1);
 47     turbOnFinalIterOnly_ =
 48         pimpleDict.lookupOrDefault<Switch>("turbOnFinalIterOnly", true);
 49 }
```

As we can see, the read function is looking in the **fvSolutions** file for the corresponding entries in the PIMPLE control dictionary:

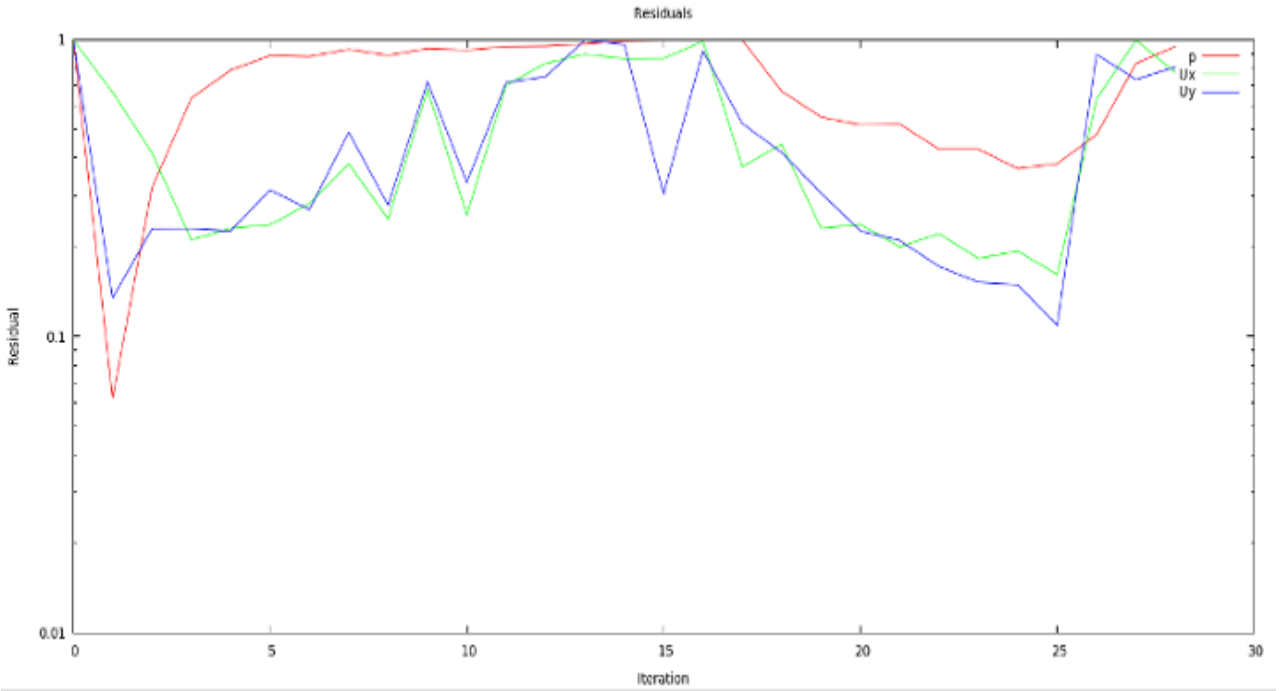
```
nOuterCorrctors (nCorrPIMPLE) is set by default to 1
nCorrectors (nCorrPISO) is set by default to 1
nNonOrthogonalCorrectors (corrPISO) is set by default to 0  (not shown here)
turbOnFinalIterOnly is set by default to true
Residual control is  set to false
```

Question 2:

If we are running the PIMPLE solver without any keywords, the solver operatings in the so called PISO mode. Therefore, we need to fulfill the stability criterion Co < 1. Based on the fact that we set CoMax to 3, this is the reason why the simulation will blow up. OpenFOAM outputs the operating mode of the solver after starting the solver:

```
134     if (nCorrPIMPLE_ > 1)
135     {
 .
 .
 .
156     }
157     else
158     {
159         Info<< nl << algorithmName_ << ": Operating solver in PISO mode" << nl
160             << endl;
161     }
```

Residuals: If we use an empty PIMPLE directory the residual plot looks as follow:



(/index.php/File:Case_1_Residual.png)

As we see, the residuals do not converge and the solution blow up after round about 27 time steps (each iteration = one time step). Again, the blow up is based on the limitation of the PISO algorithm which forces us to keep Co less than 1. There are two options to get the simulation stable. The first one is to limit the Courant number, which is in fact not the topic we are talking about here. The second one is to use the PIMPLE algorithm in the correct way. The PIMPLE algorithm combines the PISO with the SIMPLE. That means, the problem is transient but we use the SIMPLE (steady-state) treatment to find the steady-state solution for each time step. This allows us to increase the Courant number larger than one.

Legend:

> Iteration at the bottom means one PISO iteration; that means in each iteration we move on by delta t. In other words every iteration solves a new time step.

4 Case B

The second case sets the first keyword in order to change the operating mode of the solver. We change the dummy directory as follow:

```
PIMPLE
{
    nCorrectors         2;
}
```

Now we correct the pressure field twice which means that the following equation is calculated twice:

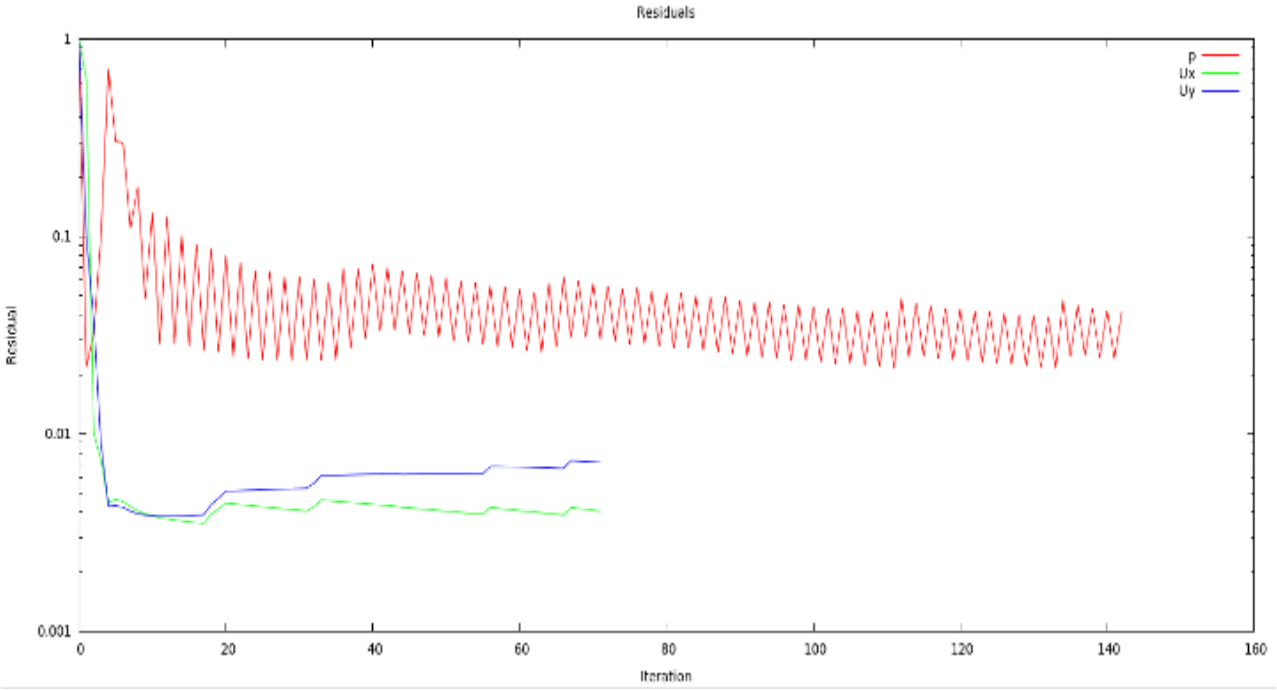
$$\nabla^2 p = f(U, \nabla p)$$

In our case B, the pressure - momentum coupling is now stable enough to produce a non-diverging solution. Thus, the solver is not blowing up.

Advantage

While using nCorrectors with a number higher than one, the application (pimpleFoam) is still working in PISO mode with the advantage that we re-calculate the pressure twice. This PISO loop option is always available in all transient FOAM solvers, if the application uses the PISO or PIMPLE control class.

Residuals: If we have a look at the new residual plots in which we have `nCorrectors 2`, we get the following:



(/index.php/File:Case_2_Residual.png)

The residuals of the pressure are much better than in case A and the solution is stable for the shown iterations - not tested if the solver will actually blow up, if we run the case for longer time.

Legend:

For the momentum lines, each iteration represents »one time step«. Thus, we are solving around 65 time steps

For the pressure we solve the equation given above twice and therefore, it is not anymore possible to say that one iteration is one time step. Of course, it is obvious that two iterations represents one time step but it is not always the case.

5 Case C

Now we are going to introduce the real merged PISO - SIMPLE algorithm and therefore we are using the advantage of the SIMPLE mode for each time step (relaxation). What does that mean? If we are using the PISO mode, we are limited to the time step which fulfills the condition that Co < 1. Based on the definition of the Courant number one knows that this can lead to very expensive computational resources while solving a real time problem. To speedup the simulation we have to overcome the limit of Co < 1 which can be achieved by using the SIMPLE algorithm. The reason for that is based on the fact that the SIMPLE algorithm is developed for the final, steady-state, condition. To reach that flow behavior rapidly, we are actually not interested in the transient behavior and development of the flow (ddt terms = zero) and thus, dt should be as large as possible. Of course there is much more behind which is not mentioned right now. More details can be found in the book (http://www.holzmann-cfd.de/index.php/en/mathematics-numerics-derivations-and-openfoam) mentioned at the beginning or in other literature.

However, a very large time step or actually neglecting the temporal derivative changes the characteristic of the equations. For such cases the SIMPLE algorithm families (SIMPLE, SIMPLEC, SIMPLER...) were developed. Based on the fact that in the normal SIMPLE algorithm, a pressure term is neglected, we need to have under-relaxation (even with SIMPLC it is needed in most cases but the relaxation factors can be chosen larger). In addition it should be mentioned that there are two different relaxation methods available in OpenFOAM: Field and matrix relaxation. More about that is given in the book (http://www.holzmann-cfd.de/index.php/en/mathematics-numerics-derivations-and-openfoam) mentioned at the beginning.

Miss understanding

A lot of people miss understand the capability and usage of the PIMPLE algorithm. The main problem in the case we are analyzing right now is, that this is very simple and does not need special numerical treatments. However, in complex geometries and with different models which causes stiff problems, the PIMPLE algorithm is the method of choice. Imagine a very complex CFD simulation with small cell sizes and high velocities, it is straight forward to use small time steps in the PISO mode. However, If one wants to have bigger time steps one has to increase the Co number which break the criterion of Co < 1. At that moment we get a discrepancy because one one hand we want go on faster in time in order to reduce the computational effort but it is not possible because we are limited based on C o < 1. Thats why we use the PIMPLE mode then. Keep in mind that each time step can be much more expensive than a time step in the PISO calculation but the time step itself can be much larger. In addition it can occur that we loose significant information of the flow which changes the flow pattern.

5.1 Case C.1

For the C1 case we change the **fvSolutions** file as follow:

```
PIMPLE
{
    nCorrectors 1;
    nOuterCorrectors 2;
}
```

Compared to case B, we added the nOuterCorrectors keyword and set to two while changing the pressure correction back to one again. The new keyword and the value > 1 activates the PIMPLE loop and lead to a re-calculation of the pressure - momentum coupling (here two times) in one time step.

1. Construct momentum matrix

2. Construct the pressure matrix using the momentum matrix
3. Calculate pressure
4. Correct the velocities with the new pressure field
5. Reconstruct the momentum matrix with the new velocities
6. Construct the pressure matrix using the momentum matrix with the updated velocities
7. Calculate the pressure
8. Correct the velocities with the new pressure field

1. $\frac{U}{\partial t} + \nabla \cdot (UU) + \nabla \cdot R = -\nabla p$
2. $\nabla^2 p = f(U, \nabla p)$

$\longrightarrow p_{\text{new}}$

$\longrightarrow$ correct U with $p_{\text{new}} \longrightarrow U_{\text{corrected}}$

$\longrightarrow$ now we have the $U_{\text{corrected}}$ and p_{new} for the second loop

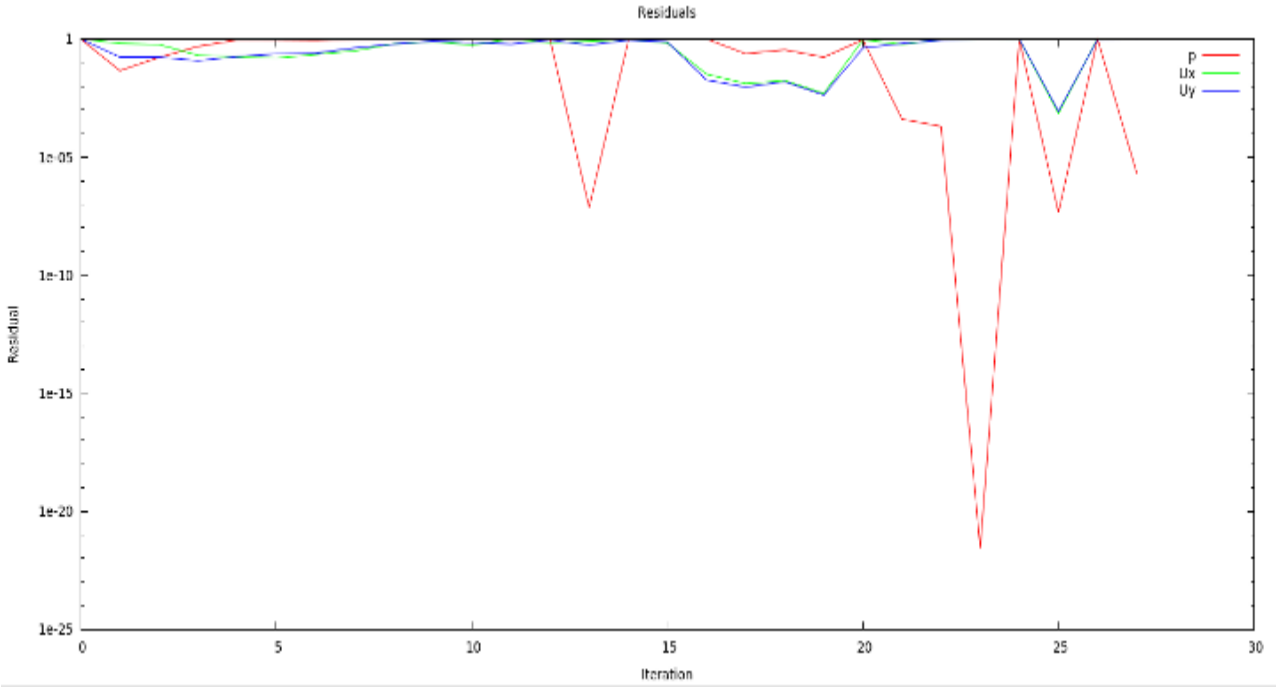
3. $\frac{U}{\partial t} + \nabla \cdot (UU) + \nabla \cdot R = -\nabla p$
4. $\nabla^2 p = f(U, \nabla p)$

$\longrightarrow p_{\text{new}}$

$\longrightarrow$ correct U with $p_{\text{new}} \longrightarrow U_{\text{corrected}}$

Case C1 does not has big advantages and is mostly like case A with the difference that we accelerate error accumulation and calculate the turbulence equations only at the end of the PIMPLE loop (here after two pimple iterations based on the fact that *turbOnFinalIterOnly* is set to true as default.

Residuals: Checking the residual plots give us:



(/index.php/File:Case_3a_Residual.png)

As already mentioned, it is amplifying the errors and the divergence occur earlier compared to case A. Keep in mind that the legend here represents the outer loops. Thus, the simulation diverges already after

half of the time compared to case A. The reason is that we use the PIMPLE algorithm in a wrong way.

5.2 Case C.2

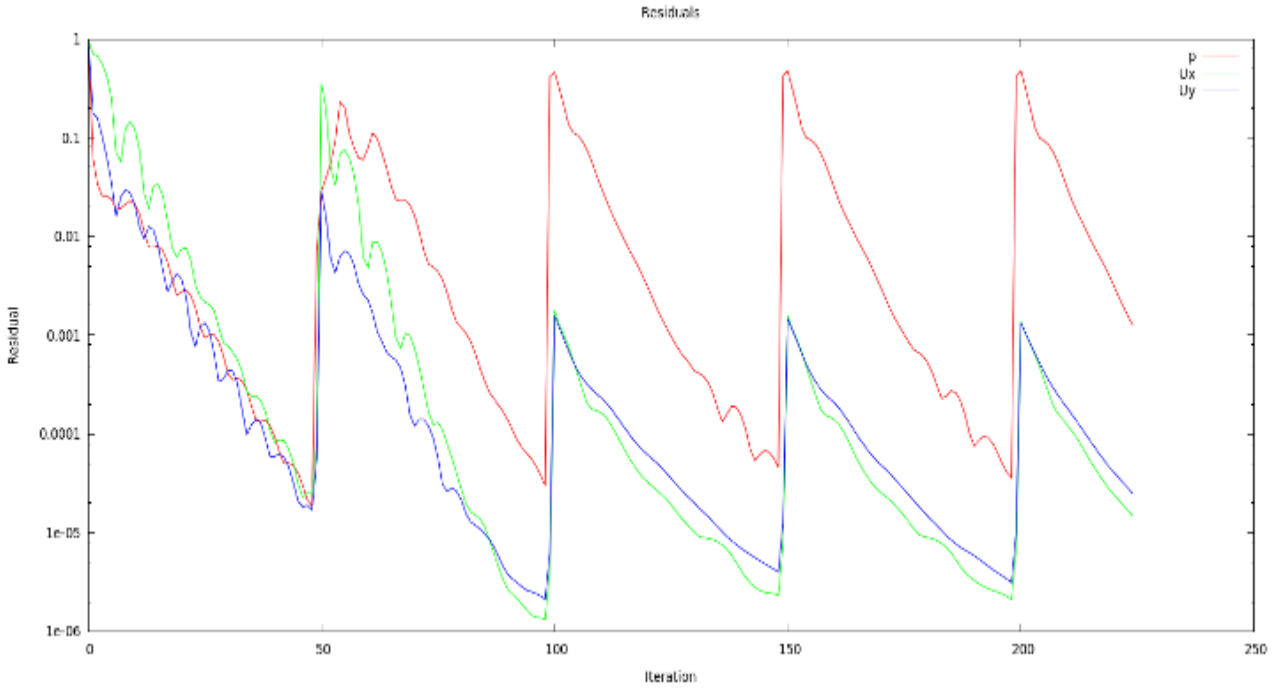
Case C.2 will set-up the complete PIMPLE mode. Therefore, we have to add the relaxationFators dictionary for the fields and equations. Thus, the **fvSolutions** file looks as follow:

```
PIMPLE
{
    nNonOrthogonalCorrectors 0;
    nCorrectors          1;
    nOuterCorrectors    50;
}

relaxationFactors
{
    fields
    {
        p      0.3;
        pFinal   1;
    }
    equations
    {
        "U|k|epsilon"     0.3;
        "(U|k|epsilon)Final"   1;
    }
}
```

We introduced the relaxation factors for the SIMPLE calculation in between the single time steps to obtain a smooth convergence rate. The application relaxes the fields and equations for each pimple loop as given above. After we reach the 50 outer corrections, the final flag is set and the relaxation factors are set to one again. A discussion about that is given in [1]. However, in case C.2 we calculate the pressure - momentum coupling now for 50 times (49 times with relaxation and the last time without relaxation - if the .*Final flag is set to 1 or neglected). For stiff problems or/and weak described systems (boundary conditions etc.) this method allows us to get a smooth convergence rate.

Residuals: Analyzing the residual plots again, we get the following:



(/index.php/File:Case_3b_Residual.png)

We see that the solution is stable and the convergence rate is very good. Furthermore, we see the steady-state behavior between the single time steps; here from zero to fifty iteration, 100 to 150 and so on. After finishing the pimple loop, we move on in time and do the same calculation procedure again. In addition we could set the *nCorrectors > 1* which is briefly discussed/mentioned later on.

Never forget

Right now we solve 50 corrections (momentum - pressure coupling) for each time step and therefore the computational effort could be much larger than before. To avoid this, we always should add the residual control in order to leave the pimple loop after we reach the residual control of each quantity we specify » case C.3

5.3 Case C.3

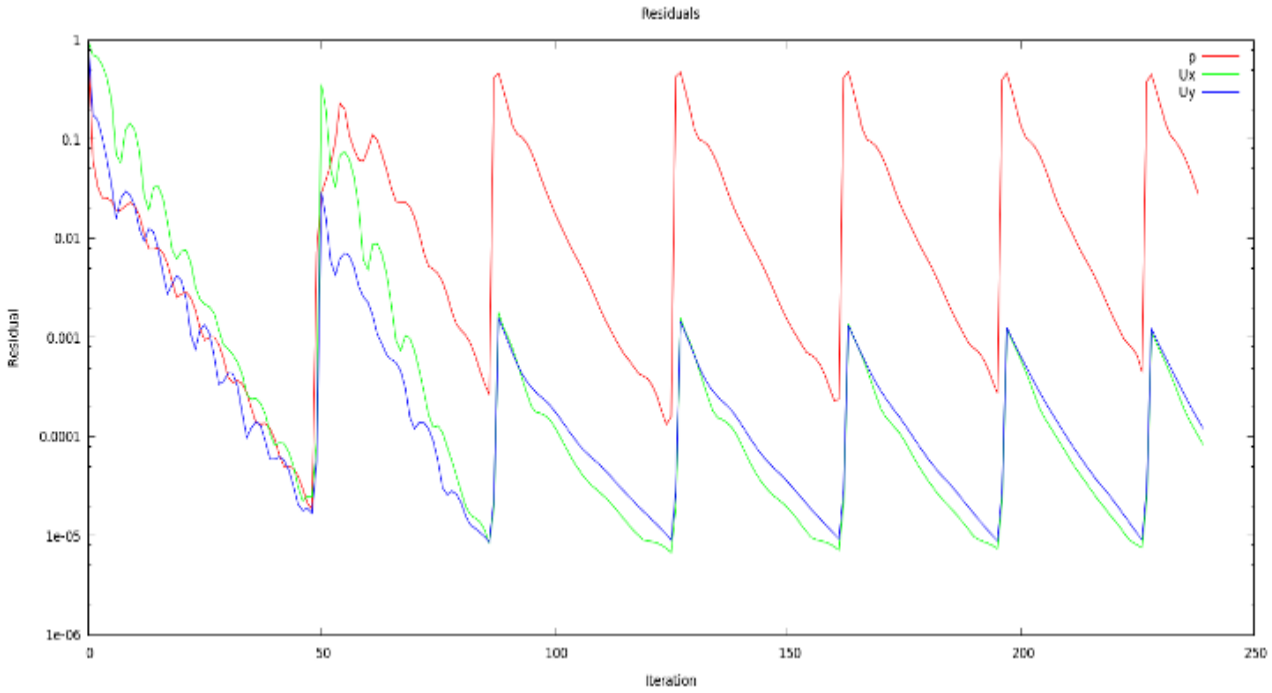
At the end we add the residual control section to the fvSolutions file in order to leave the momentum-pressure coupling loops after everything is converged to speedup the calculation and use all advantages from the PIMPLE algorithm:

```
PIMPLE
{
    nNonOrthogonalCorrectors 0;
    nCorrectors          1;
    nOuterCorrectors    50;

    residualControl
    {
        U
        {
                tolerance  1e-5;
                relTol      0;
        }
        p
        {
                tolerance  5e-4;
                relTol      0;
        }
     }
}

relaxationFactors
{
    fields
    {
        p      0.3;
        pFinal   1;
    }
    equations
    {
        "U|k|epsilon"     0.3;
        "(U|k|epsilon)Final"   1;
    }
}
```

Residuals: A last time we are analyzing the residual plots of the calculation with relaxation and residual control:



(/index.php/File:Case_3c_Residual.png)

We see that after the first time step - which requires almost 50 outer corrections - the pimple loop is stopped earlier and therefore the simulation is faster (each time step needs less outer corrections). If the PIMPLE algorithm will converge with the settings one is using can be checked out after a new time step is started:

```
.
.
.
smoothSolver:  Solving for k, Initial residual = 0.0660137, Final residual = 1.08191e-06, No Iterations 3
'''PIMPLE: converged in 28 iterations'''
ExecutionTime = 13.84 s  ClockTime = 14 s
```

Although we set the total amount of 50 outer corrections, the calculation will exit the loop after the residual control settings are fulfilled. In the particular case it stops after 28.

6 Some hints

the pitzDaily tutorial is an too easy case to show the advantage of PIMPLE in a good manner (another case is given in the book by Tobias Holzmann mentioned at the beginning)

The PIMPLE algorithm is very effective for stiff and large cases, for really bad initialized cases, weak boundary setup and if one wants to increase the Co number

The PIMPLE algorithm speeds up your simulation and allow us to reach nice convergence rates while using under relaxation in between the time steps

For high turbulent flow fields we should add the keyword turbOnFinalIterOnly = false to calculate the turbulent properties each outer loop too

Always set the nOuterCorrectors to a very high number (~ 50 to 1000) and control your pimple loop with the residual control (make sure that explizit terms converged)

Set nCorrectors to a small number (1 to 3)

If one sets nCorrectors to a high value it could be possible to leave the pimple loop earlier and prevent extra calculation but can be vice versa too

nNonOrthogonalCorr means that you correct the pressure field more often with the new calculated one. This is needed for high non-orthogonal meshes because we get an explizit correction term (nOuterCorrectors will do the same but with more computational effort):

$$\nabla^2 p = f(U, \nabla p) \rightarrow p_{\text{new}} \rightarrow \nabla^2 p_{\text{new}} = f(U, \nabla p_{\text{new}}) \rightarrow$$ till nNonOrtogonalCorr is reached

The optimum relaxation factors depend on the simulation case we are running and if we use the SIMPLE or SIMPLEC (for corrected) algorithm
Due to the fact that the PIMPLE algorithm allows us to use bigger time steps, it could happen that we loose important information which lead to different results
Right at the start of a transient simulation it is common that the application does not converge the first time steps within the given outer corrections based on the bad initial fields. This should change after a few time steps
The CHT solver needs a lot of PIMPLE loops but can be reduced by tweaking the numerics (if you know how - a tutorial can be found on the website of Tobias Holzmann)

Updated 21.08.2017

Recommend Tweet +1

(http://www.heise.de/ct/artikel/2-Klicks-fuer-mehr-Datenschutz-1333879.html)

Category (/index.php/Special:Categories): Numerical methods (/index.php/Category:Numerical_methods)

This page was last modified on 27 August 2018, at 20:51.
This page has been accessed 114,915 times.
Content is available under GNU Free Documentation License 1.3 (http://www.gnu.org/copyleft/fdl.html) unless otherwise noted.

(http://www.gnu.org/copyleft/fdl.html) (//www.mediawiki.org/)

(https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki)

(http://openfoamwiki.net/hypotheticaljuicy.php)