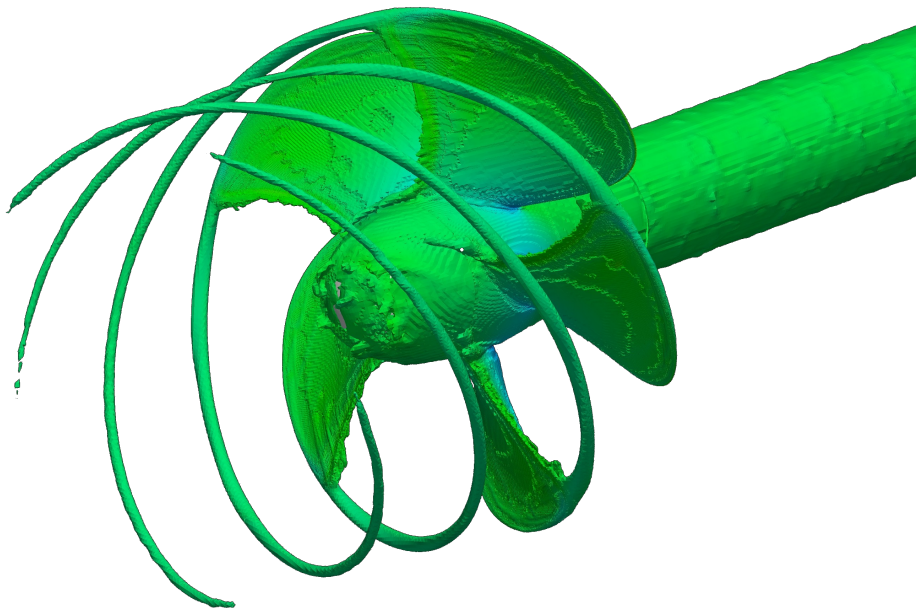
Solution techniques in OpenFOAM

Artur Lidtke

University of Southampton
akl1g09@soton.ac.uk

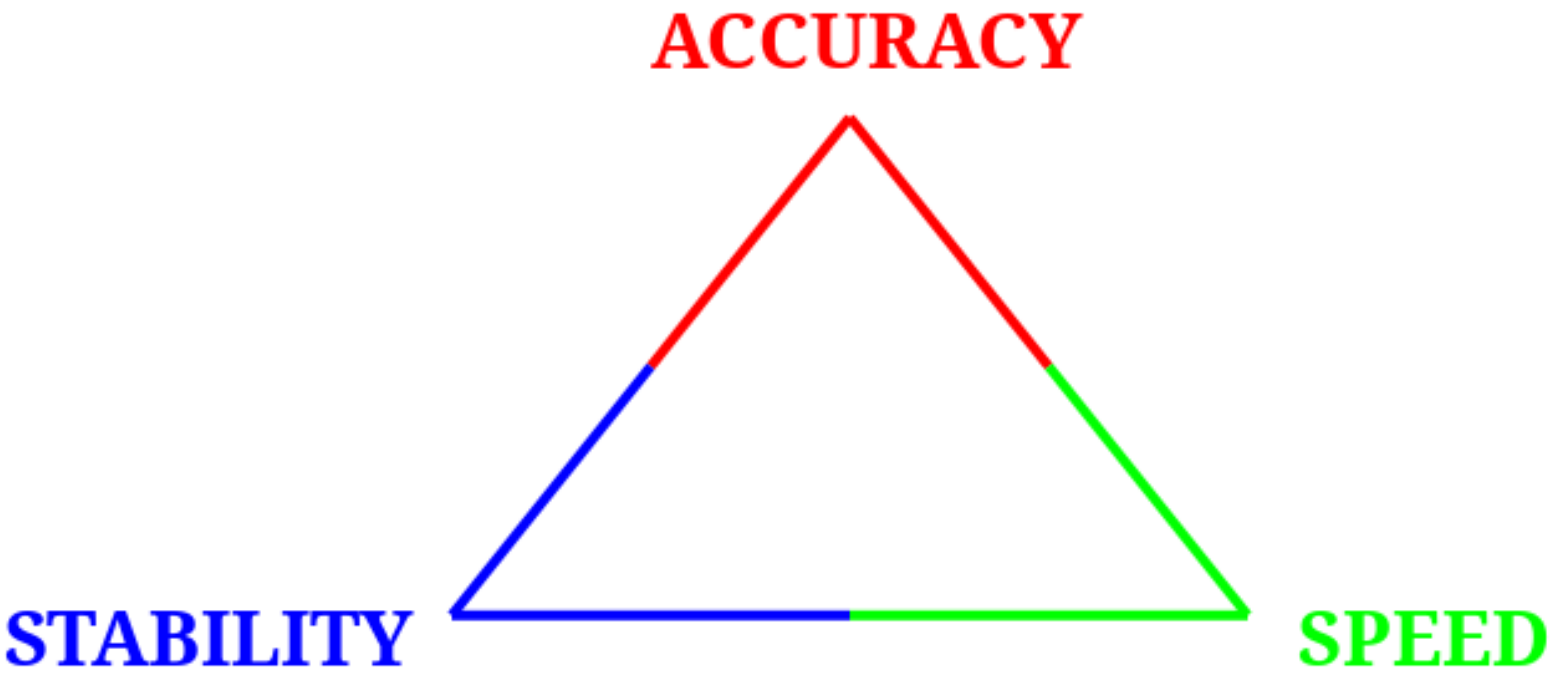
November 4, 2014

Content



Concepts

The balance



Choosing the correct set-up

- Cannot have maximum accuracy with rock-solid stability at a minimum cost
- Need to trade off the three drivers based on the application
- There is no definite answer that will always work for all problems, even vaguely similar ones
- Check forums, read papers and look at tutorials to find a starting point
- Then run a sensitivity study to see what works best for you
- Often useful to use a different set-up at various stages of a simulation - high stability first for an initial estimate and then move towards accuracy
- Stay up to date with code development to see what's new

How does this work?

- We use a set of small, finite volumes to describe our domain of interest
- Numerical **schemes** of various orders are used to construct linear equations approximating our problem
- The resulting matrices are then given to linear **solvers** which try to satisfy the equations
- An indication of how well an equation is satisfied is its **residual**
- This is done for all equations in turn in order to resolve the **pressure-velocity coupling**
- Need to repeat this at each time step (unsteady) or iteration (steady)

What's the point of all this stuff in the logs? - SIMPLE

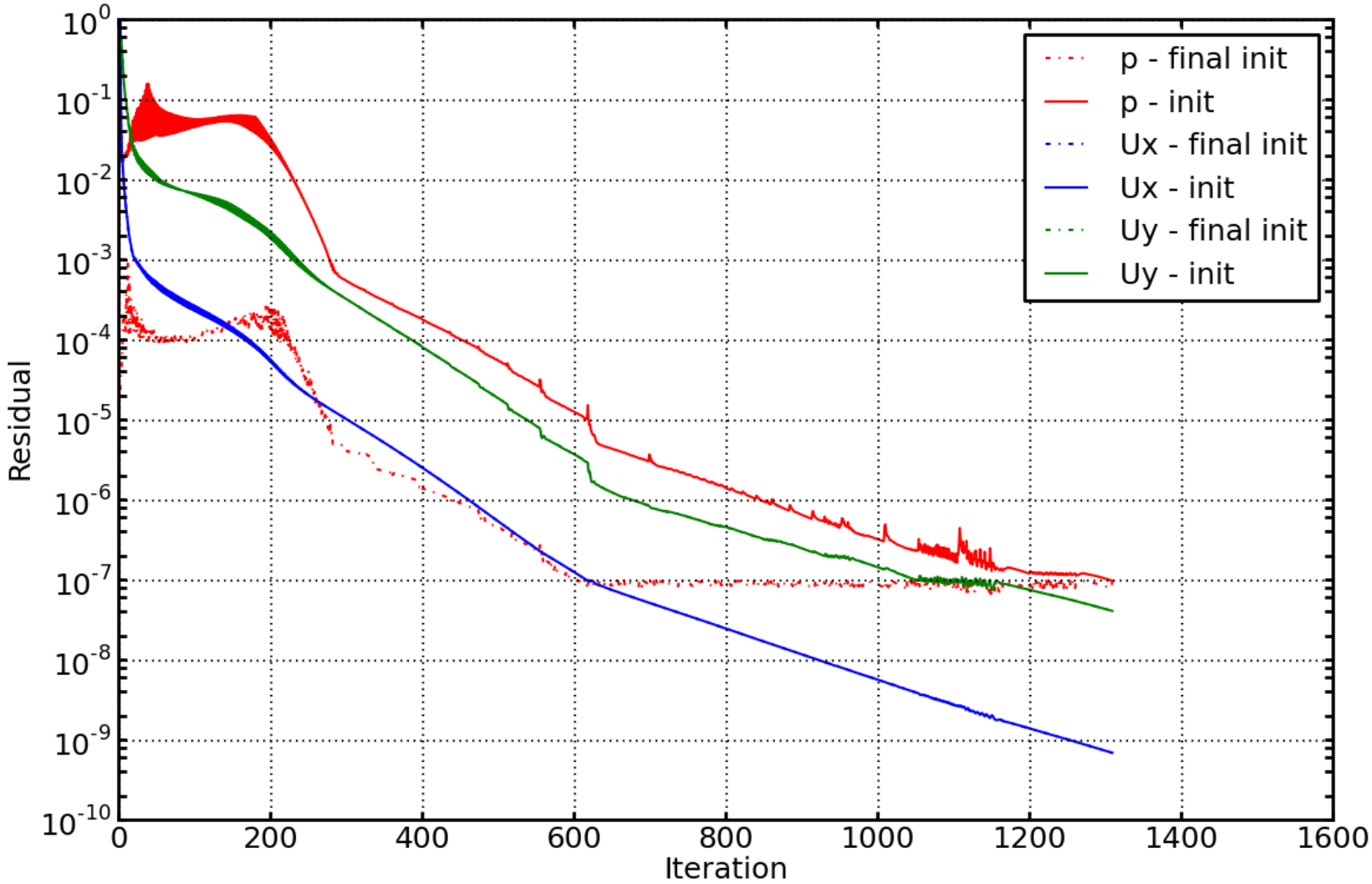
```
smoothSolver:  Solving for Ux, Initial residual = 9.69841e
    -08, Final residual = 9.69841e-08, No Iterations 0
smoothSolver:  Solving for Uy, Initial residual = 7.00898e
    -07, Final residual = 7.94086e-08, No Iterations 3

GAMG:  Solving for p, Initial residual = 0.0032684, Final
    residual = 2.99124e-06, No Iterations 26
GAMG:  Solving for p, Initial residual = 3.00459e-06, Final
    residual = 9.44095e-08, No Iterations 60
GAMG:  Solving for p, Initial residual = 9.44092e-08, Final
    residual = 9.44092e-08, No Iterations 0
time step continuity errors : sum local = 2.32813e-10,
    global = 3.76108e-12, cumulative = 8.16684e-12

smoothSolver:  Solving for omega, Initial residual = 9.81049
    e-08, Final residual = 9.81049e-08, No Iterations 0
smoothSolver:  Solving for k, Initial residual = 1.21105e-07
    , Final residual = 6.45996e-08, No Iterations 1
```

Steady-state solution (SIMPLE)

- Start wtih no information about the flow (maybe a potential solution)
- Iterate over velocity and pressure interchangeably to drive the equations towards convergence
- Need to monitor the residuals of all equations to make sure the approximate solution satisfies them
- May control **underrelaxation** in order to "dampen" the solution process - for $\alpha < 1$ more stable but slower convergence

$$\phi^n = \phi^{n-1} + \alpha(\phi^{n*} - \phi^{n-1})$$

Graphically speaking



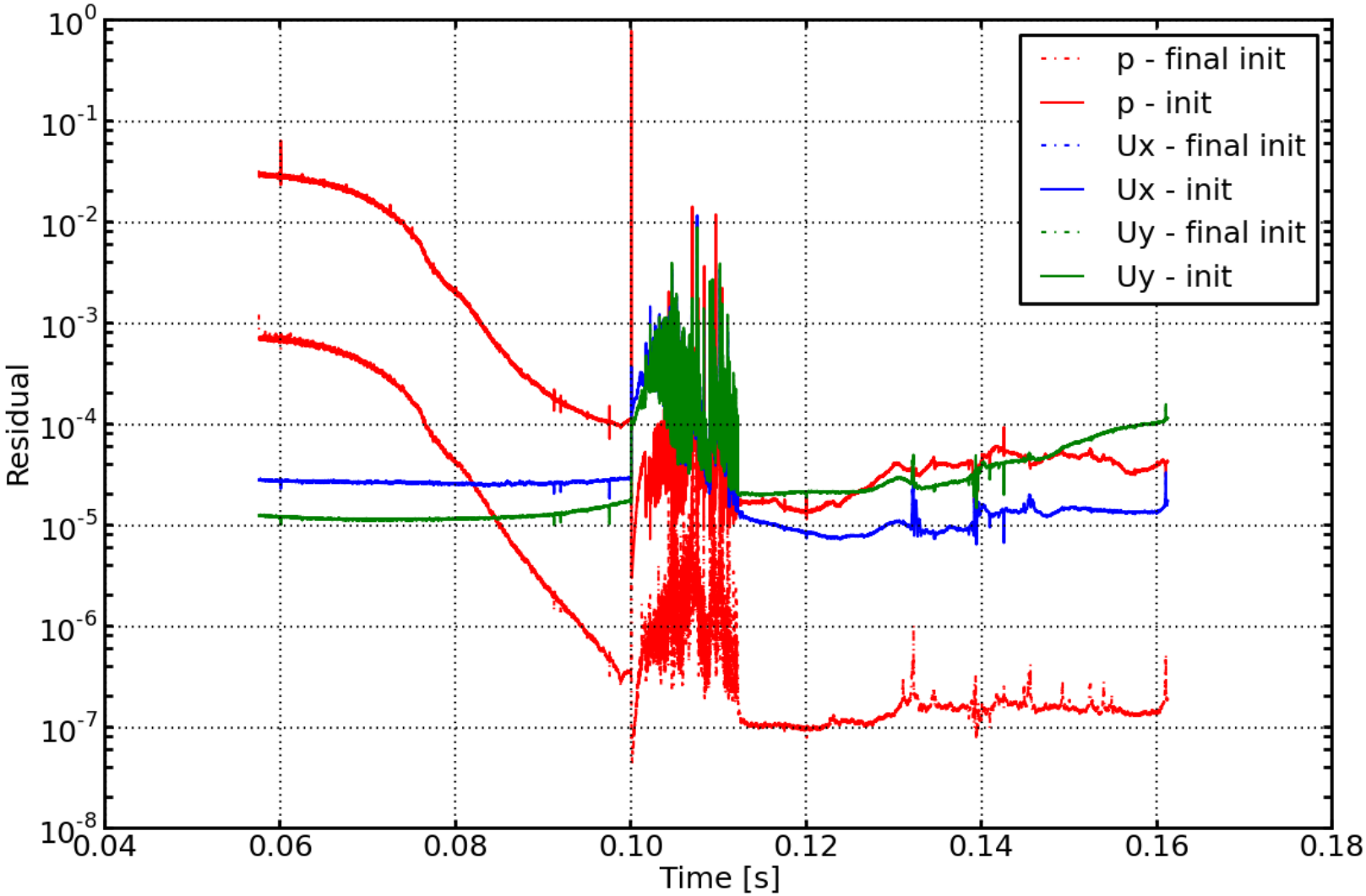
What's the point of all this stuff in the logs? - PISO

```
Courant Number mean: 0.0976805 max: 0.585722

DILUPBiCG:  Solving for Ux, Initial residual = 0.148584,
    Final residual = 7.15711e-06, No Iterations 6
DILUPBiCG:  Solving for Uy, Initial residual = 0.256618,
    Final residual = 8.94127e-06, No Iterations 6

DICPCG:  Solving for p, Initial residual = 0.379232, Final
    residual = 3.38648e-07, No Iterations 34
time step continuity errors : sum local = 3.15698e-09,
    global = 1.94222e-19, cumulative = 9.28841e-19

DICPCG:  Solving for p, Initial residual = 0.286937, Final
    residual = 5.99637e-07, No Iterations 33
time step continuity errors : sum local = 6.08774e-09,
    global = 5.80474e-19, cumulative = 1.50932e-18
```

Unsteady solution (PISO/PIMPLE)

- Need to reduce the residuals **at each time step** and not just in the long run as in SIMPLE
- May be very hard to maintain stability without a good initial condition
- Must monitor the entire solution, not just the residuals, to ensure convergence at each time step
- Check if the solvers reach maximum no. iterations before the residuals drop by the required amount
- If yes then
  - Reduce the time step
  - Increase the no. solver iterations
  - Use more loops over each equation (nCorrectors, nOuterCorrectors, etc. in fvSolution)

Graphically speaking



Courant number

- At each time the Courant number is computed for an unsteady problem

Courant number

- At each time the Courant number is computed for an unsteady problem

$$Co = \frac{u\Delta t}{\Delta x}$$

Δt - simulation time step
Δx - grid size
u - local flow velocity

OR

"How many cells does a fluid particle travel through in one time step"

Courant number

- At each time the Courant number is computed for an unsteady problem
- Should keep Co reasonable - depending on the schemes and methods there may be a theoretical limit for stability
- Generally, $Co < 1.0$ is a good rule of thumb, **will** need less for LES with high order schemes

Courant number

- At each time the Courant number is computed for an unsteady problem
- Should keep Co reasonable - depending on the schemes and methods there may be a theoretical limit for stability
- Generally, $Co < 1.0$ is a good rule of thumb, **will** need less for LES with high order schemes
- The higher the Reynolds number (u), the smaller the cells have to be to resolve the boundary layer

Courant number

- At each time the Courant number is computed for an unsteady problem
- Should keep Co reasonable - depending on the schemes and methods there may be a theoretical limit for stability
- Generally, $Co < 1.0$ is a good rule of thumb, **will** need less for LES with high order schemes
- The higher the Reynolds number (u), the smaller the cells have to be to resolve the boundary layer
- The smaller the cells and the higher the speed the smaller the time step that allows a stable solution

Courant number

- At each time the Courant number is computed for an unsteady problem
- Should keep Co reasonable - depending on the schemes and methods there may be a theoretical limit for stability
- Generally, $Co < 1.0$ is a good rule of thumb, **will** need less for LES with high order schemes
- The higher the Reynolds number (u), the smaller the cells have to be to resolve the boundary layer
- The smaller the cells and the higher the speed the smaller the time step that allows a stable solution
- **CFD of High-Re flows is still a huge challenge!**

Schemes - convective

- Full list of available options on
  http://www.openfoam.org/docs/user/fvSchemes.php
- If you are going to use upwind for LES then don't bother!
- Typical choices for RANS/URANS
  - upwind - stable but inaccurate
  - lienarUpwind - mixed 1^{st}-2^{nd} order, more accurate but less stable
- My suggestions for LES
  - linear - 2^{nd} order accuracy but very unstable (convective term)
  - filteredLinear/limitedLinear - special mixed schemes supposed to balance accuracy and stability (convective term)

Schemes - time

- Two main choices:
  - Euler - 1^{st} order time scheme, stable but not accurate - not a problems since URANS introduces even bigger inaccuracies!
  - backward/Crank-Nicholson - 2^{nd} order time schemes - a must for LES
- The time scheme will influence the maximum time step greatly
- Typical symptome for an LES case running at a too large time step for a high order time scheme is the last loop over pressure equation exploding (residuals > 1, pressure field becoming unphysical)

Other schemes

- These include
  - Gradient
  - Laplacian
  - Surface normal gradient
  - Interpolation
- Generally affect stability less than the time and convective scheme
- As a rule of thumb OK to use linear unless a specific problem requires otherwise - look at literature and tutorial cases to get an idea

Other schemes

- These include
  - Gradient
  - Laplacian
  - Surface normal gradient
  - Interpolation
- Generally affect stability less than the time and convective scheme
- As a rule of thumb OK to use linear unless a specific problem requires otherwise - look at literature and tutorial cases to get an idea
- Let us examine the flat plate test case with the default settings I specified

Other schemes

- These include
  - Gradient
  - Laplacian
  - Surface normal gradient
  - Interpolation
- Generally affect stability less than the time and convective scheme
- As a rule of thumb OK to use linear unless a specific problem requires otherwise - look at literature and tutorial cases to get an idea
- Let us examine the flat plate test case with the default settings I specified

Go to

```
cd $FOAM_RUN/flatPlate/ZPG_wallFunction_coarse
```

Other schemes

- These include
  - Gradient
  - Laplacian
  - Surface normal gradient
  - Interpolation
- Generally affect stability less than the time and convective scheme
- As a rule of thumb OK to use linear unless a specific problem requires otherwise - look at literature and tutorial cases to get an idea
- Let us examine the flat plate test case with the default settings I specified

Edit

```
gedit system/fvSchemes
```

Solvers

- Even less obvious choice than the schemes, here are some commonly used ones:
  - smoothSolver - fairly stable, good for steady RANS (U, k, ω)
  - GAMG + GaussSeidel - robust for the pressure equation
  - PBiCG + DILU - works OK for velocity for unsteady problems, can also use for scalar fields (like volume fraction, for instance)
- Factors affecting the optimum solver for a given problem:
  - Size and nature of the mesh (structured/unstructured)
  - Parallelisation strategy
  - Numerical schemes used
  - Solution strategy (no. loops per equation etc.)
- More info on
  `http://www.openfoam.org/docs/user/fvSolution.php`

Solvers

- Even less obvious choice than the schemes, here are some commonly used ones:
  - smoothSolver - fairly stable, good for steady RANS (U, k, ω)
  - GAMG + GaussSeidel - robust for the pressure equation
  - PBiCG + DILU - works OK for velocity for unsteady problems, can also use for scalar fields (like volume fraction, for instance)
- Factors affecting the optimum solver for a given problem:
  - Size and nature of the mesh (structured/unstructured)
  - Parallelisation strategy
  - Numerical schemes used
  - Solution strategy (no. loops per equation etc.)
- More info on
  http://www.openfoam.org/docs/user/fvSolution.php

Edit

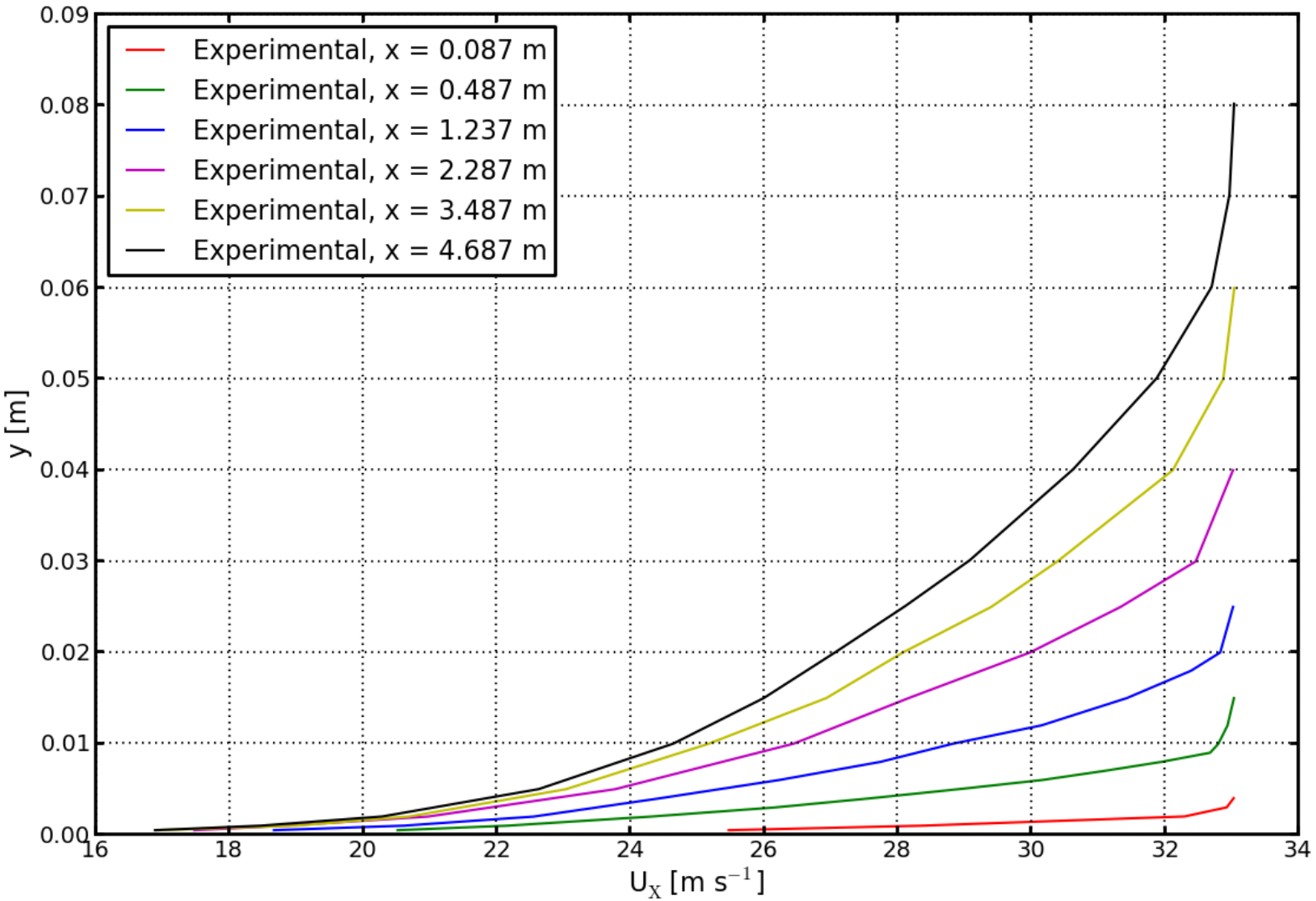
```
gedit system/fvSolution
```

Practice - flat plate

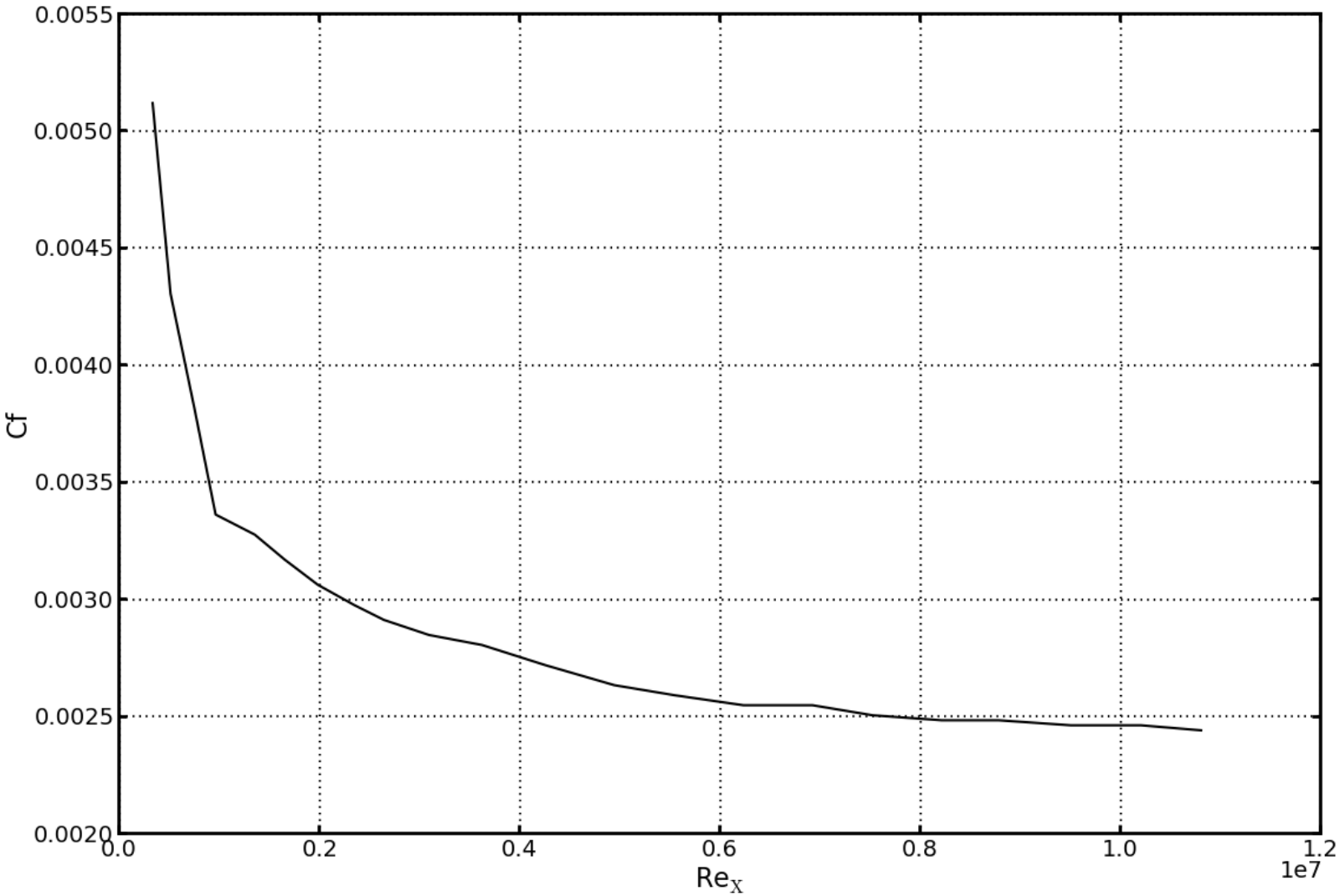
Test case

- Flat plate in zero pressure gradient condition
- Probably the most fundamental hydrodynamics problem
- Very relevant to all engineering applications
- Extensive amount of validation data in the public domain
  - Wieghardt et al. 1968
    http://www.grc.nasa.gov/WWW/wind/valid/fpturb/fpturb01/fpturb01.html
  - T3A transitional Re regime; see for instance Cutrone et al. 2007
    http://cfd.mace.manchester.ac.uk/twiki/pub/Main/ClareTurner/cutrone_bypass.pdf

Test case


Experimental, x = 0.087 m
Experimental, x = 0.487 m
Experimental, x = 1.237 m
Experimental, x = 2.287 m
Experimental, x = 3.487 m
Experimental, x = 4.687 m
y [m]
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
16
18
20
22
24
26
28
30
32
34
U_X [m s^-1]

Test case


0.0055
0.0050
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
Cf
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e7
Re$_X$

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified

Go to

```
cd $FOAM_RUN/flatPlate/ZPG_wallFunction_coarse
```

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified
- There is a ready script that should execute the solution, let's have a look

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified
- There is a ready script that should execute the solution, let's have a look

Edit

```
gedit Allrun
```

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified
- There is a ready script that should execute the solution, let's have a look
- Now, run the script. The case should converge quickly

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified
- There is a ready script that should execute the solution, let's have a look
- Now, run the script. The case should converge quickly

Use

```
./Allrun
```

Baseline simulation

- Let us examine the flat plate test case with the default settings I specified
- There is a ready script that should execute the solution, let's have a look
- Now, run the script. The case should converge quickly
- Let us take a look at the residuals and forces convergence

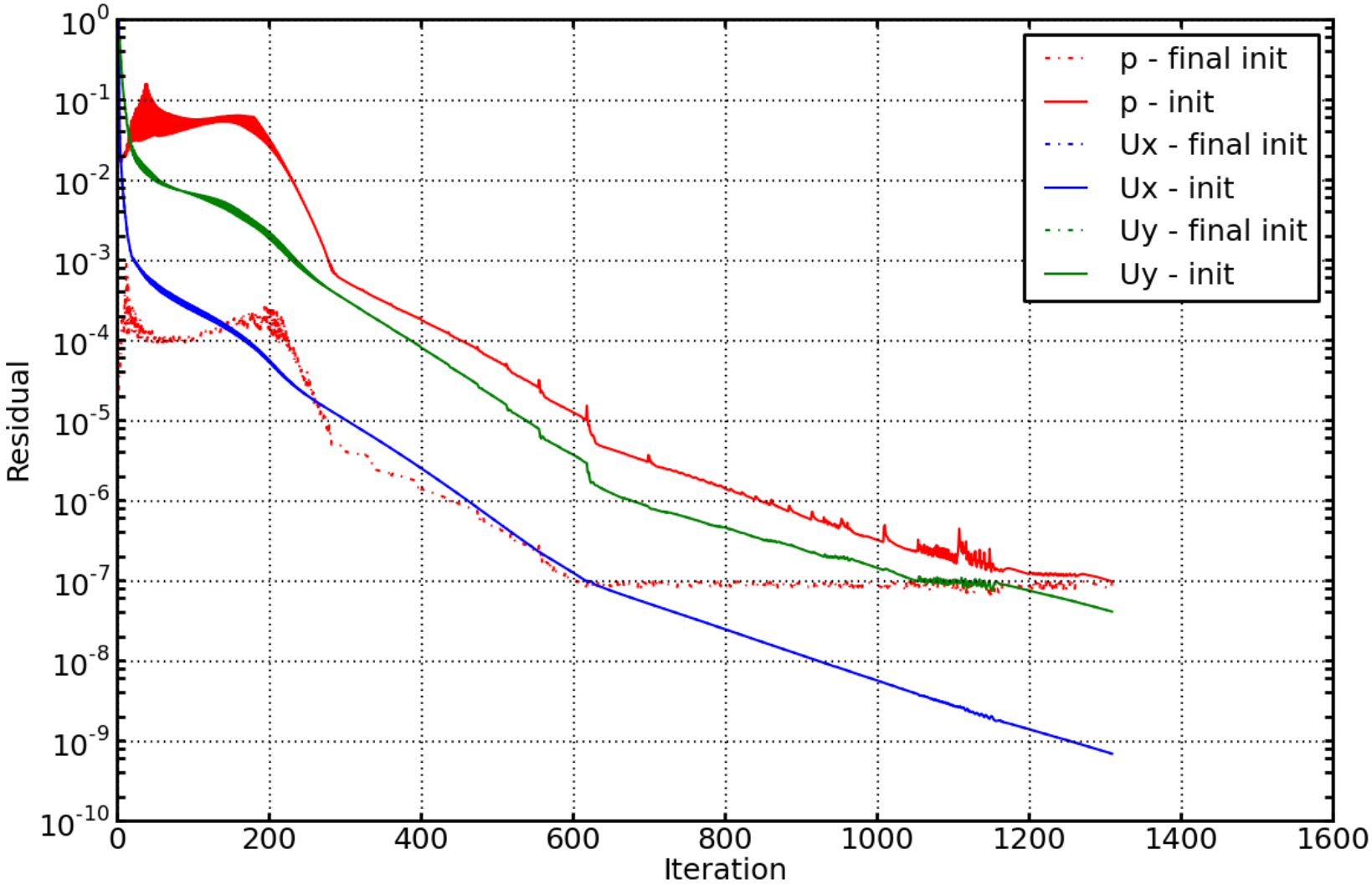
Baseline simulation

- Let us examine the flat plate test case with the default settings I specified
- There is a ready script that should execute the solution, let's have a look
- Now, run the script. The case should converge quickly
- Let us take a look at the residuals and forces convergence

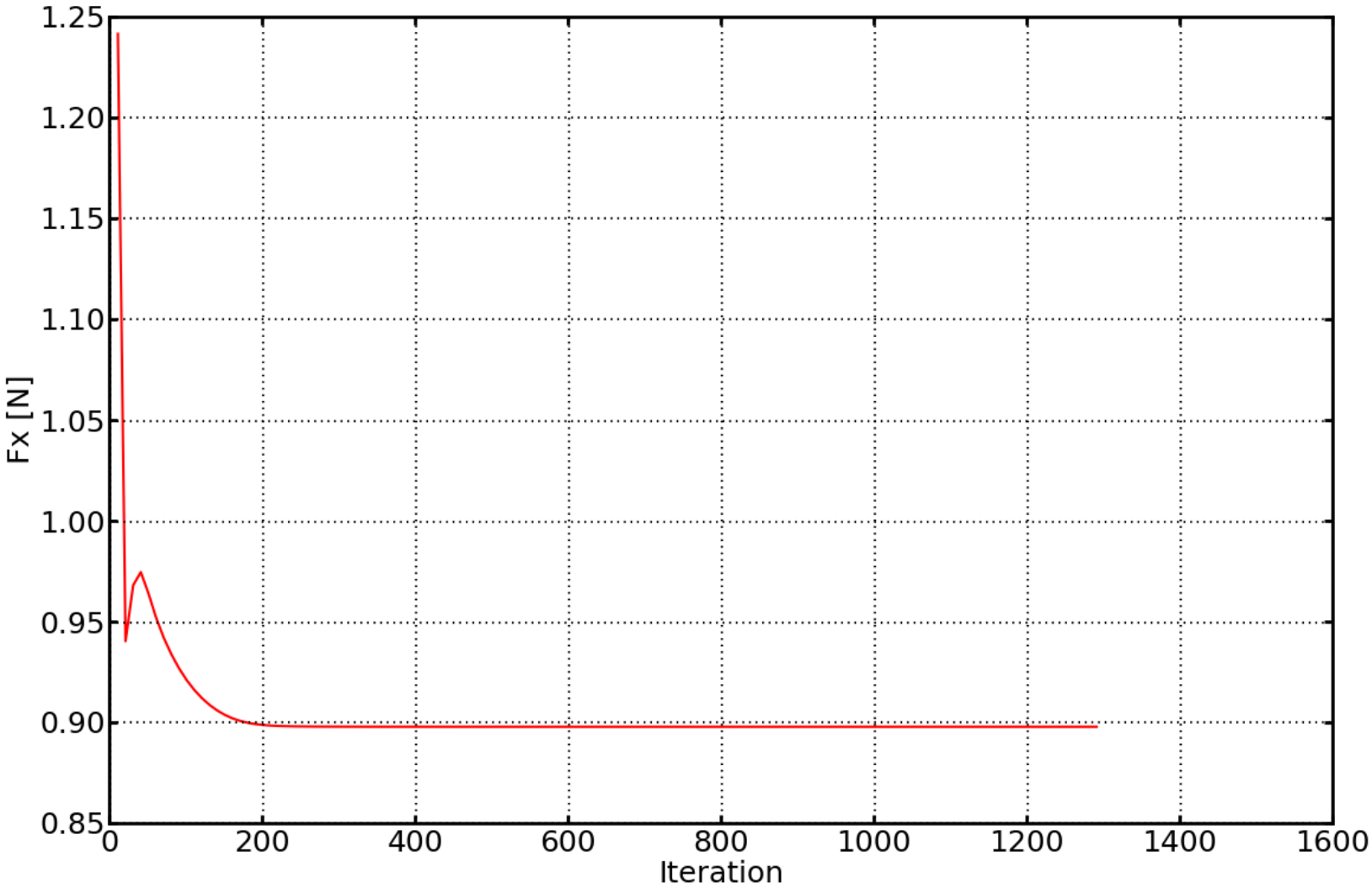
Use

- `cd ..`
- `python plotResiduals.py`
- `python plotForces.py`

Baseline simulation - residuals



Baseline simulation - drag force



What does this tell us?

- See both the pressure and velocity equations steadily progressing towards convergence
- Drag force settles quite quickly - the wall function makes convergence of the flow on the boundary easy but not necessarily very accurate!
- The remaining time of the simulation is spent on sorting out flow features
- Homework: we also collected boundary layer profile data, have a look at how these change as the solution progresses

Refining the mesh

- Let us use the coarse mesh solution to initialise one for a refined grid

Refining the mesh

- Let us use the coarse mesh solution to initialise one for a refined grid

Go to

```
cd ZPG_wallFunction_fine
```

Refining the mesh

- Let us use the coarse mesh solution to initialise one for a refined grid
- Note how we use the `mapFields` utility in the Allrun script to map the final fields from the coarse mesh onto the fine one

Refining the mesh

- Let us use the coarse mesh solution to initialise one for a refined grid
- Note how we use the `mapFields` utility in the Allrun script to map the final fields from the coarse mesh onto the fine one

Note

```
mapFields -sourceTime 2000 -consistent
    ../ZPG_wallFunction_coarse
```

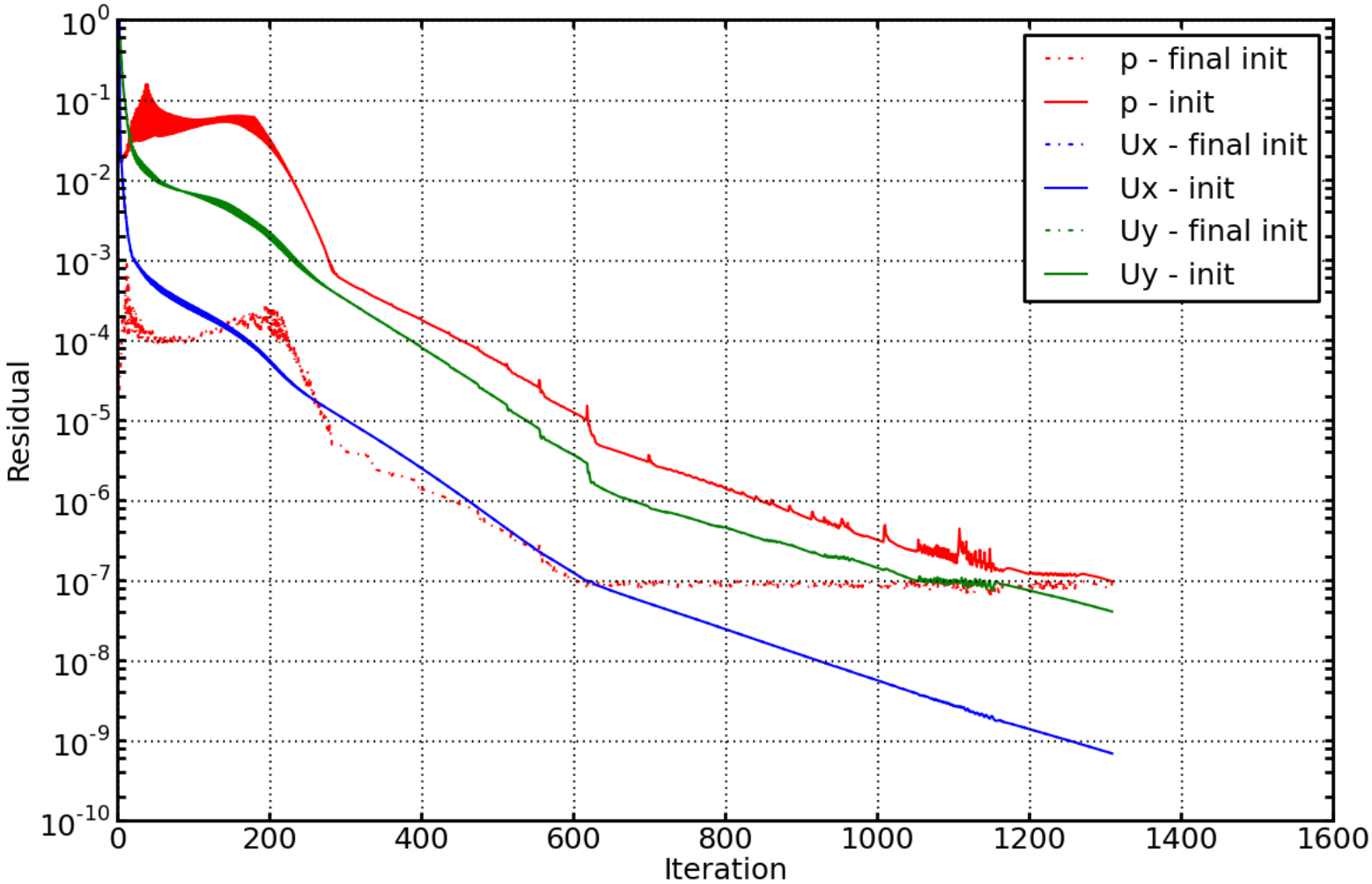
Refining the mesh

- Let us use the coarse mesh solution to initialise one for a refined grid
- Note how we use the `mapFields` utility in the Allrun script to map the final fields from the coarse mesh onto the fine one

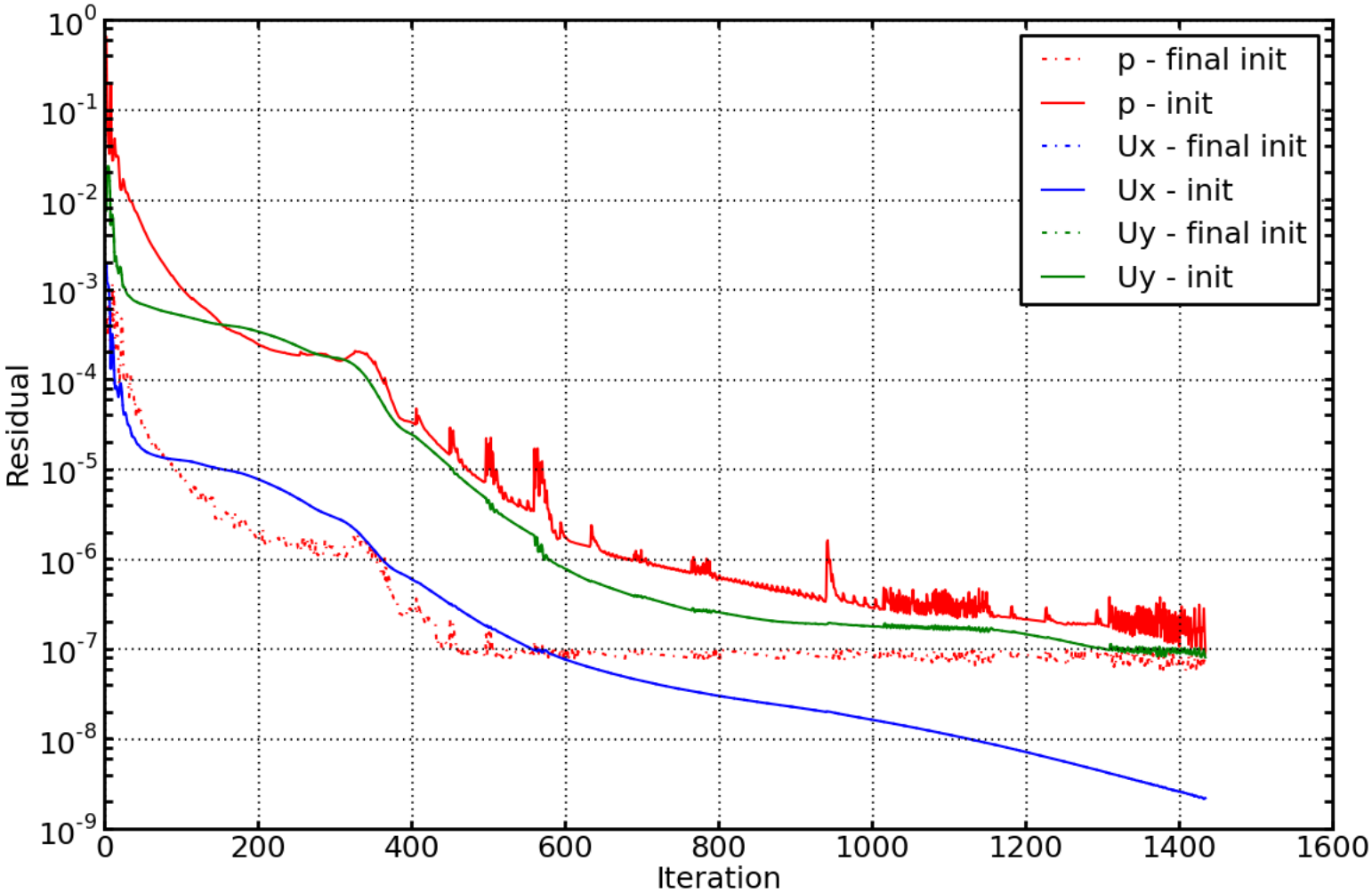
Do

- Run the simulation
- Look at the forces and residuals

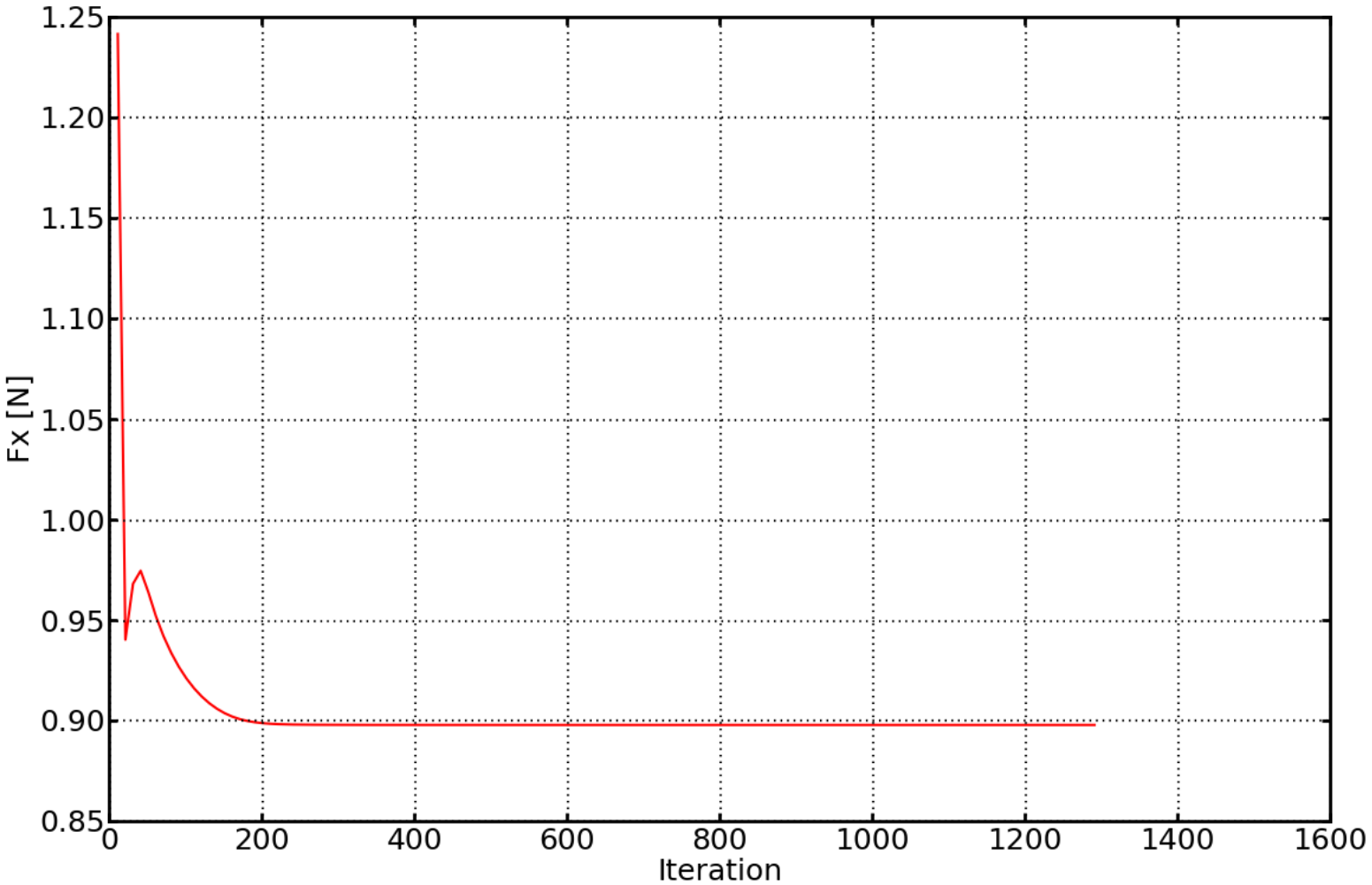
Original - residuals



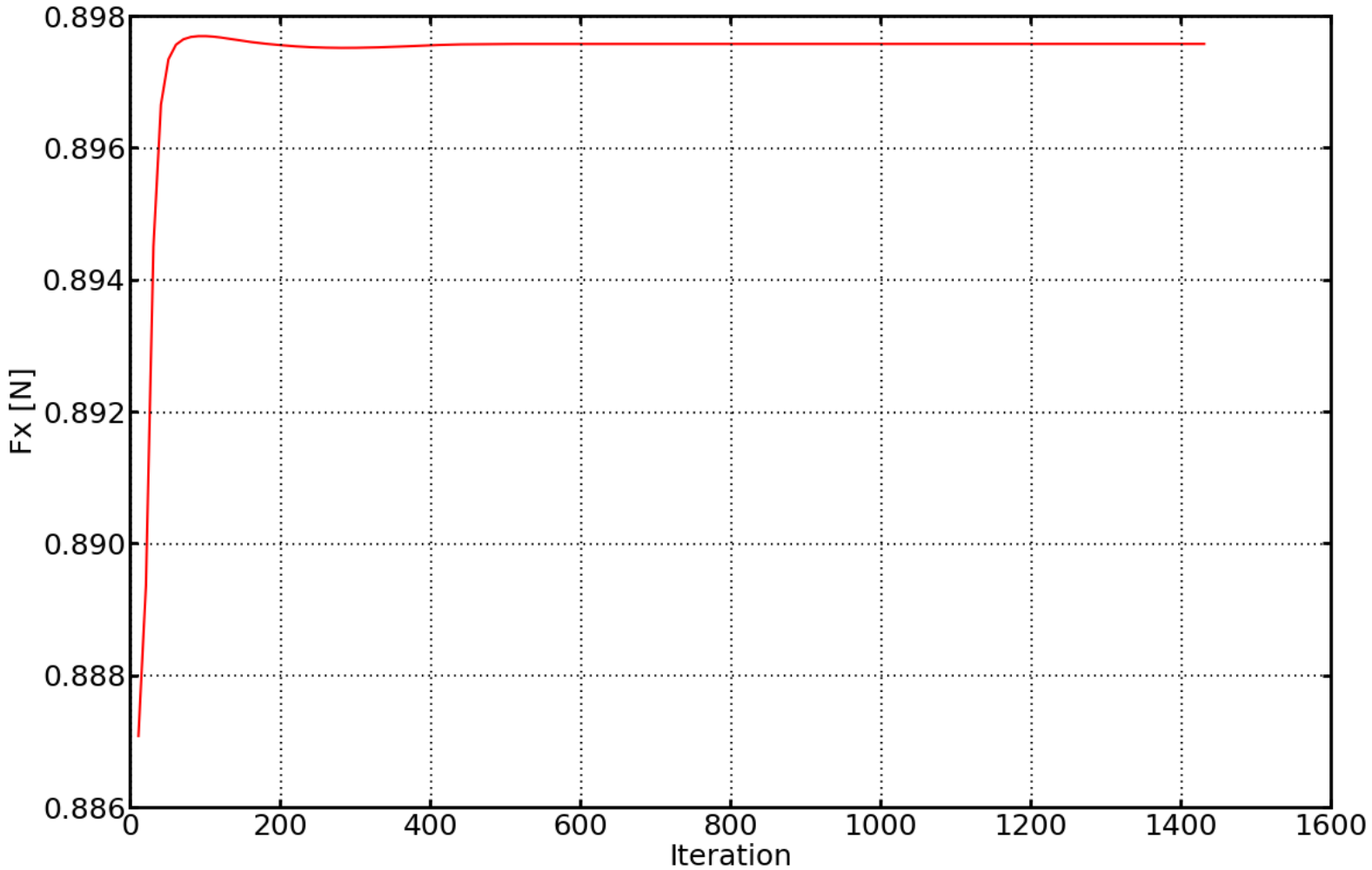
Refined grid - residuals



Original - drag force



Refined grid - drag force



What does this tell us?

- Many iterations spent on convergin the solution
- The drag force didn't change much - again, wall function is determining the friction resistance
- Since the initial condition is already pretty converged, relative to the initial state, the same change in residuals does not correspond to the same change in the drag
- Could have probably stopped the simulation earlier, obtaining the same result but saving some time!

Under-relaxation

- Copy the fine mesh flat plate case

Under-relaxation

- Copy the fine mesh flat plate case

Use

```
cp -r ZPG_wallFunction_fine
ZPG_wallFunction_fine_lessUnderRel
```

Under-relaxation

- Copy the fine mesh flat plate case
- In fvSolution, change the relaxation factor for pressure from 0.3 to 0.55

Under-relaxation

- Copy the fine mesh flat plate case
- In fvSolution, change the relaxation factor for pressure from 0.3 to 0.55

Edit

```
gedit system/fvSolution
```

Under-relaxation

- Copy the fine mesh flat plate case
- In fvSolution, change the relaxation factor for pressure from 0.3 to 0.55

Amend

From this:

```
relaxationFactors
{
    fields
    {
        p    0.3;
    }
    ...
}
```

To This:

```
relaxationFactors
{
    fields
    {
        p    0.55;
    }
    ...
}
```

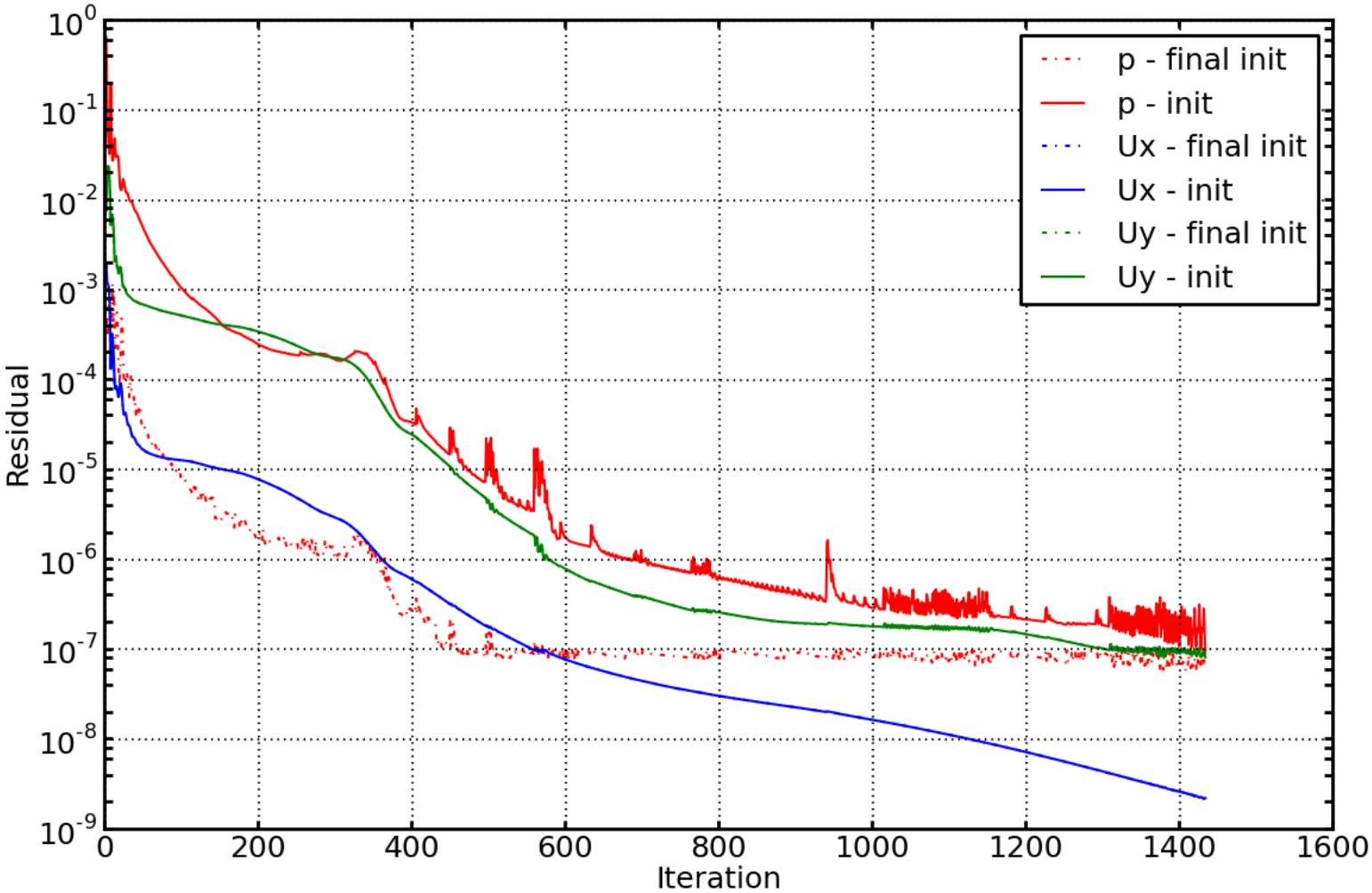
Under-relaxation

- Copy the fine mesh flat plate case
- In fvSolution, change the relaxation factor for pressure from 0.3 to 0.55

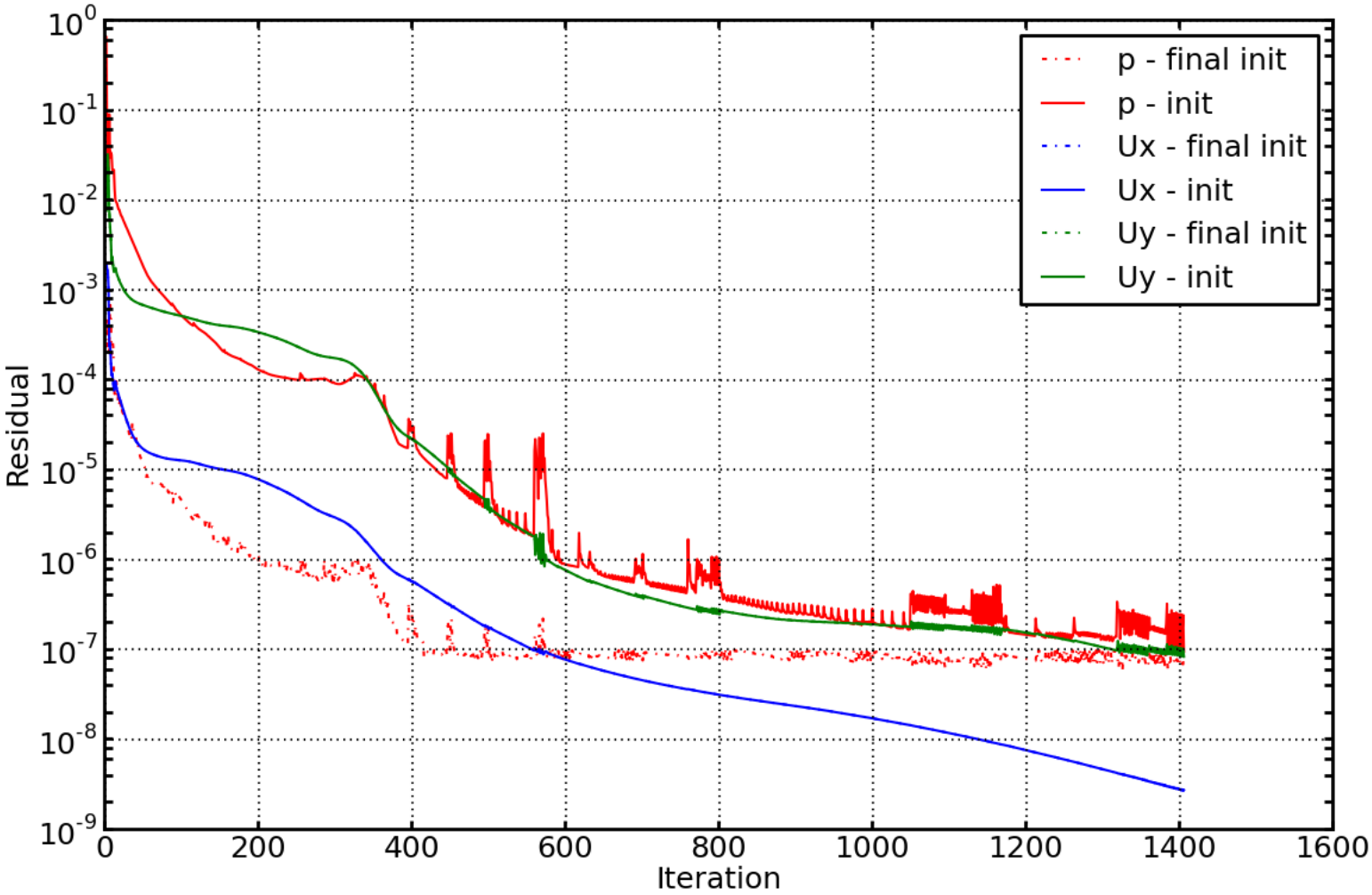
Do

- Run the simulation
- Look at the forces and residuals

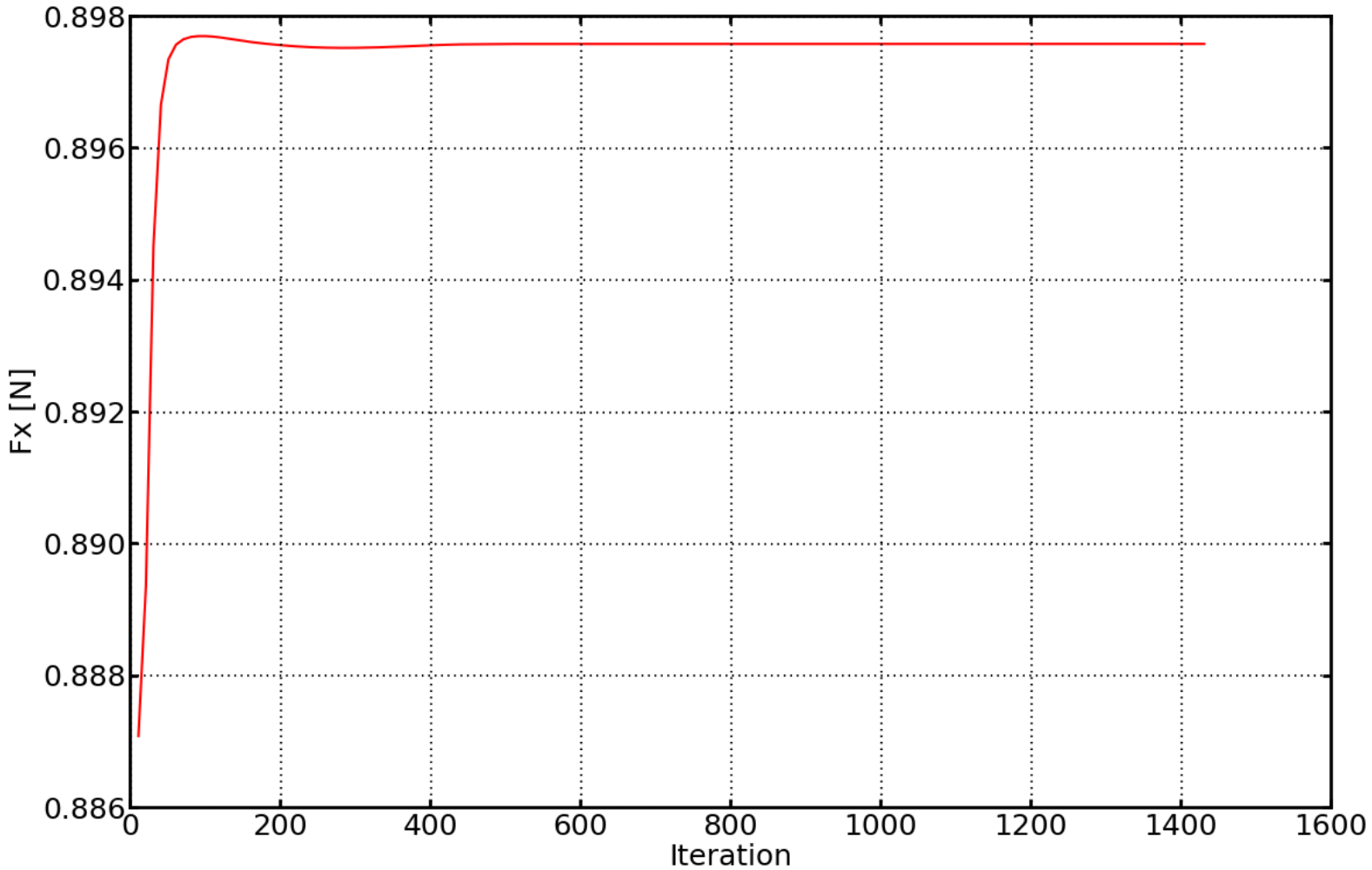
Original (fine mesh) - residuals



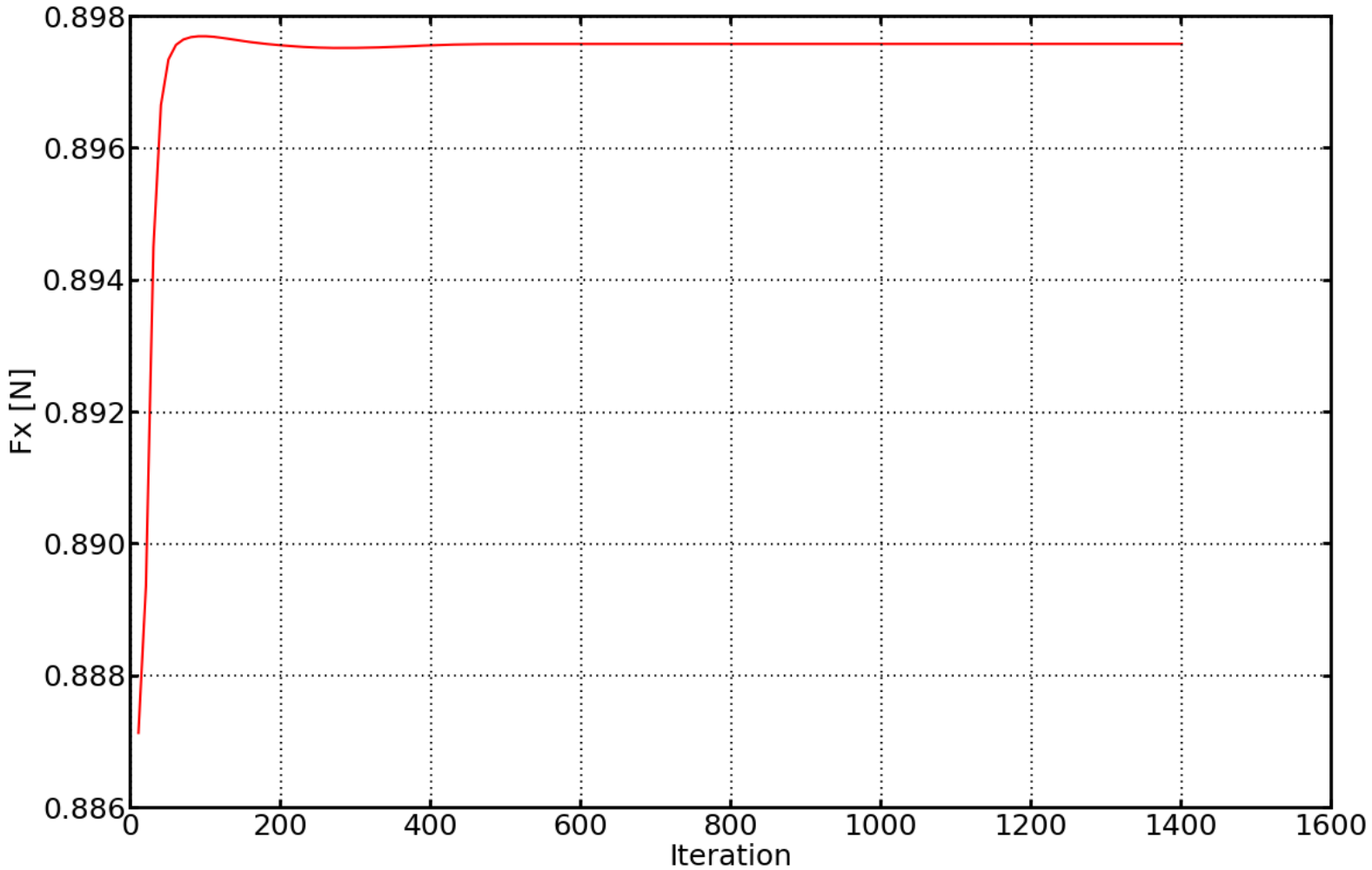
Less under-relaxation - residuals



Original (fine mesh) - drag force



Less under-relaxation - drag force



What does this tell us?

- We removed the lag in convergence between pressure and wall-normal velocity
- This means that we are wasting less time looping over already converged Uy equation when p is still changing
- Very powerful parameter in terms of optimising a solution procedure
- Cannot push it to far without losing stability
- Note the more prominent local fluctuations in the pressure equation - they will only get worse the less we under-ralax, ultimately causing a crash

High order schemes

- Again, copy the fine mesh flat plate case

High order schemes

- Again, copy the fine mesh flat plate case

Use

```
cp -r ZPG_wallFunction_fine
ZPG_wallFunction_fine_linear
```

High order schemes

- Again, copy the fine mesh flat plate case
- In fvSchemes, change the convection scheme from upwind to lienar

Edit

```
gedit system/fvSchemes
```

High order schemes

- Again, copy the fine mesh flat plate case
- In fvSchemes, change the convection scheme from upwind to lienar

Amend

- From this:

```
div(phi,U) bounded Gauss upwind;
```

- To this:

```
div(phi,U) bounded Gauss linear;
```

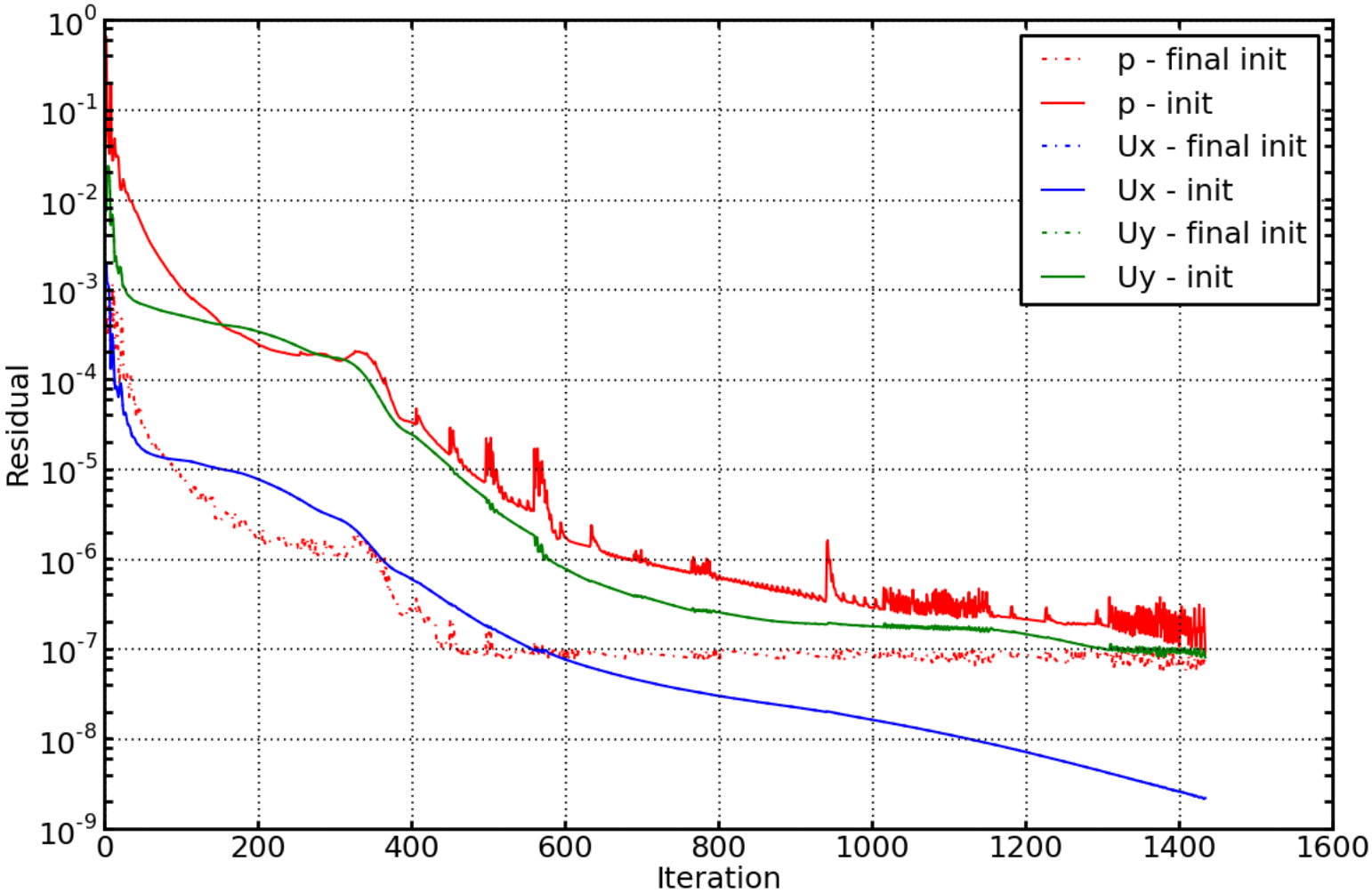
High order schemes

- Again, copy the fine mesh flat plate case
- In fvSchemes, change the convection scheme from upwind to lienar

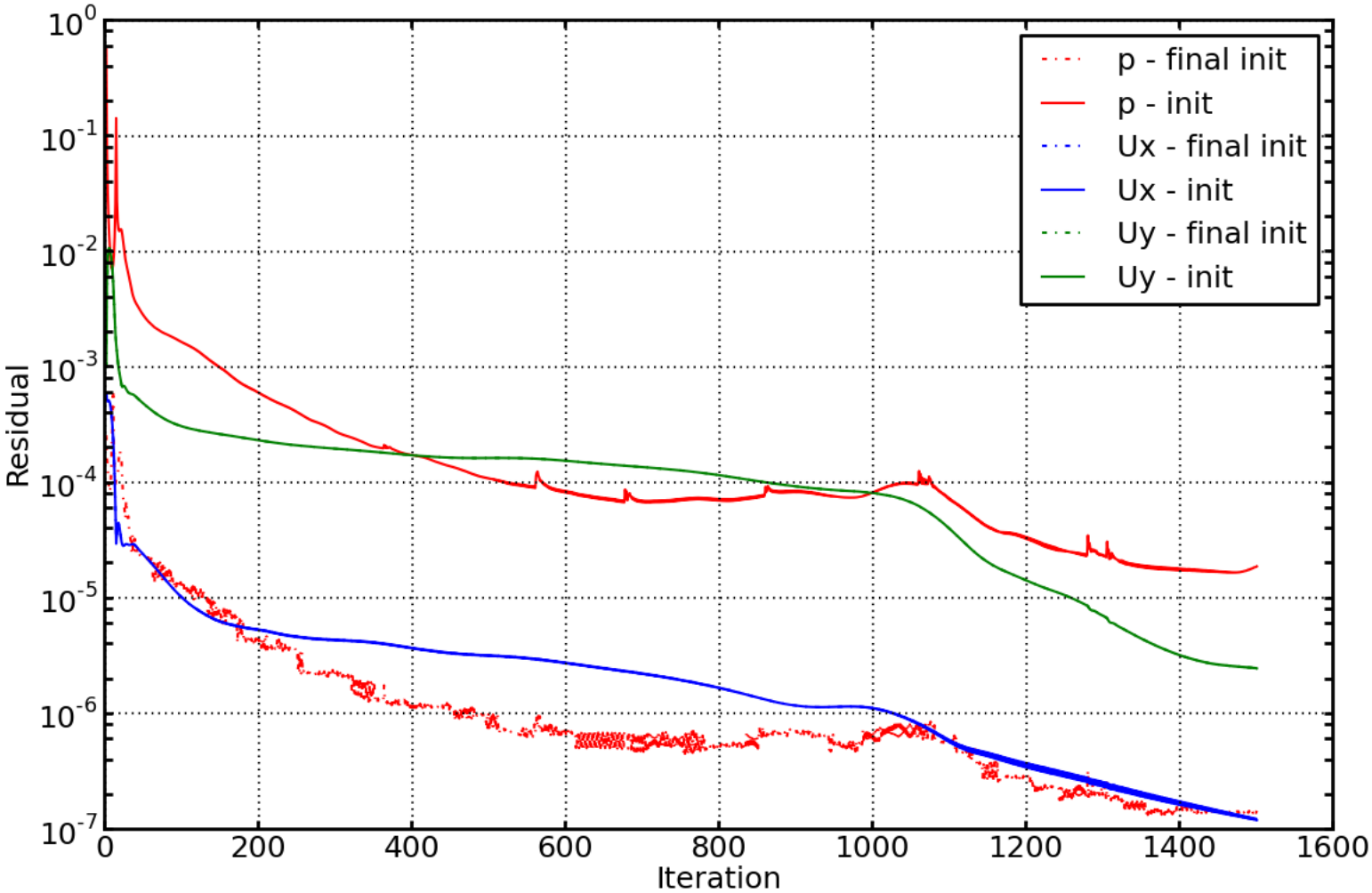
Do

- Run the simulation
- Look at the forces and residuals

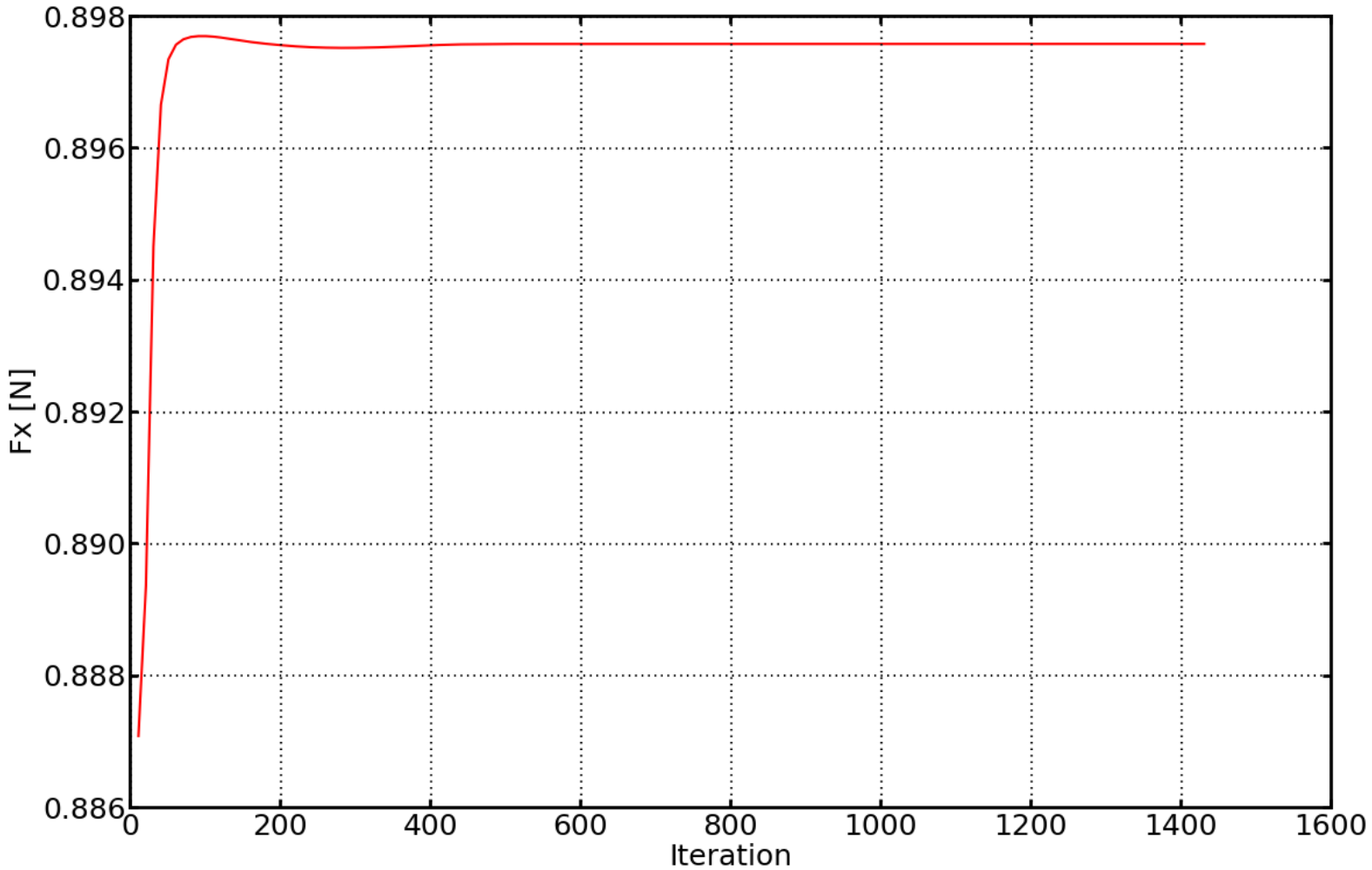
Original (fine mesh) - residuals



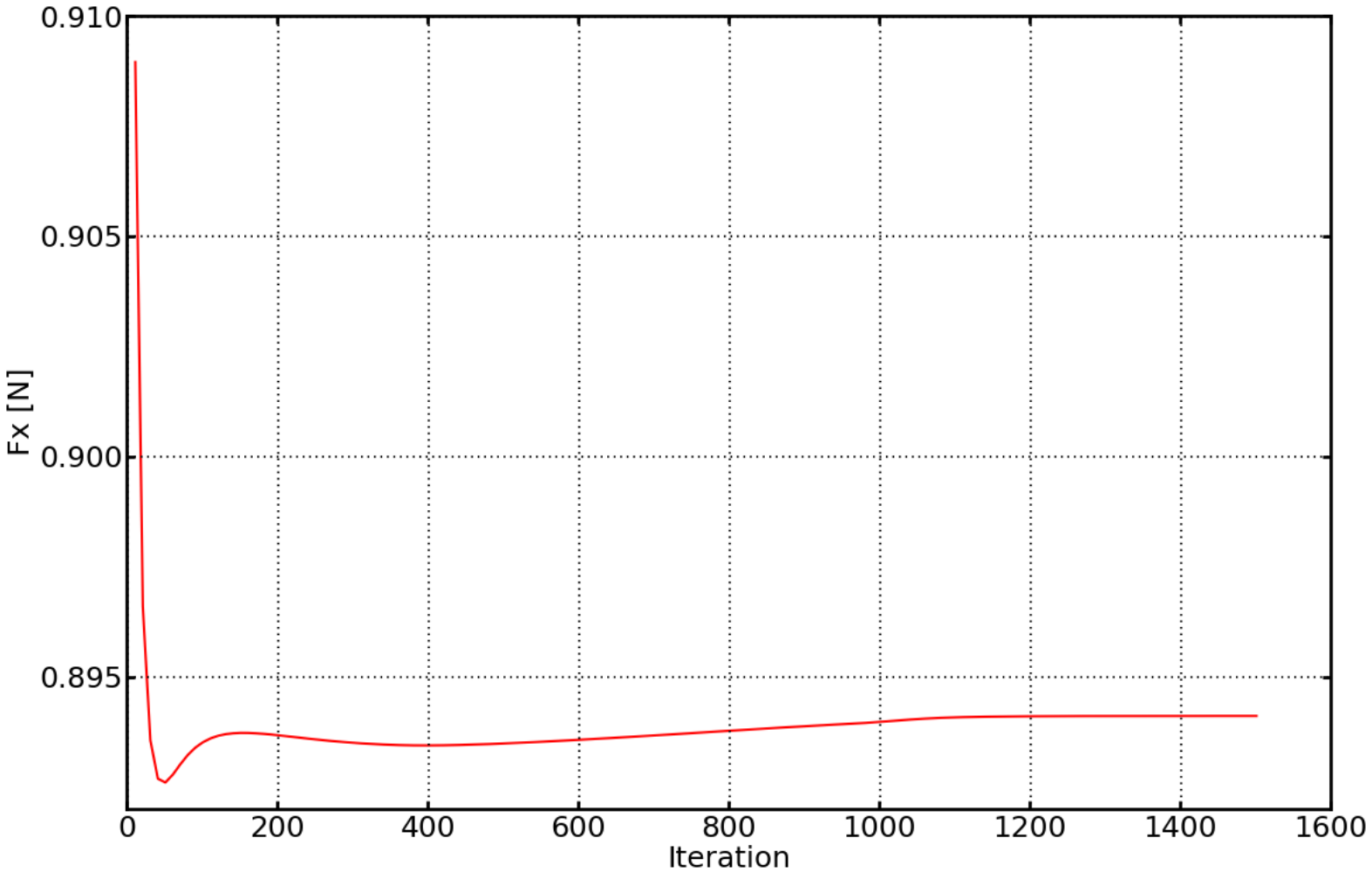
Linear convection scheme - residuals



Original (fine mesh) - drag force



Linear convection scheme - drag force



What does this tell us?

- Significantly slower convergence, both in terms of iterations and total computational time
- After the maximum no. iterations the solution is still far from convergence, would need to run for much longer - no guarantee that it will work at all!
- This shows the difficulties in using pure high order schemes even for very simple problems in finite volume frameworks

Summary

Conclusions

- We touched upon how CFD actually works, useful to understand the basics in order to troubleshoot problems
- The use of high order schemes, mesh refinement and under-relaxation on a steady-state problem has been shown to affect the solution
- The convergence process for an unsteady problem has also been discussed briefly

Further reading

- Tips and tricks to choosing solvers and schemes in OpenFOAM by Joel Guerrero
  http://www.dicat.unige.it/guerrero/of2014b/12tipsandtricks.pdf
- Implicit vs Explicit time schemes explained
  http://www.cfd-online.com/Forums/main/5982-implicit-vs-explicit-method.html
- Brief introduction to the PISO algorithm
  http://openfoamwiki.net/index.php/OpenFOAM_guide/The_PISO_algorithm_in_OpenFOAM
- Brief introduction to the SIMPLE algorithm
  https://openfoamwiki.net/index.php/OpenFOAM_guide/The_SIMPLE_algorithm_in_OpenFOAM

The End