Disclaimer

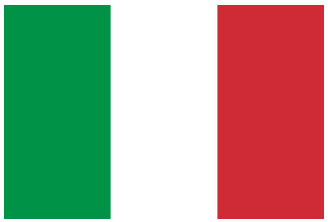
“This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM® and OpenCFD® trade marks.”

Introductory OpenFOAM® Course

From 8th to 12th July, 2013

University of Genoa, DICCA

Dipartimento di **I**ngegneria **C**ivile, **C**himica e **A**mbientale



UNIVERSITÀ DEGLI STUDI
DI GENOVA

Your Lecturer

Joel GUERRERO

joel.guerrero@unige.it

UNIVERSITÀ DEGLI STUDI
DI GENOVA

guerrero@wolfdynamics.com

wolf dynamics

Acknowledgements

These slides and the tutorials presented are based upon personal experience, OpenFOAM® source code, OpenFOAM® user guide, OpenFOAM® programmer's guide, and presentations from previous OpenFOAM® training sessions and OpenFOAM® workshops.

We gratefully acknowledge the following OpenFOAM® users for their consent to use their material:

- Hrvoje Jasak. Wikki Ltd.
- Hakan Nilsson. Department of Applied Mechanics, Chalmers University of Technology.
- Eric Paterson. Applied Research Laboratory Professor of Mechanical Engineering, Pennsylvania State University.

Today's lecture

1. Tips and tricks in OpenFOAM®

Tips and tricks in OpenFOAM®

Where and how should we
put more salt?

Tips and tricks in OpenFOAM®

OpenFOAM® shell environment

You can have as many versions of OpenFOAM® as you like living together. You just need to source the right environment variables. For instance, on my workstation I have nine versions living together. This is how my .bashrc file looks like,

```
## OPENFOAM ###

#     source $HOME/OpenFOAM/OpenFOAM-1.6-ext/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-1.6-ext.GPU/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-1.7.1/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-2.0.x/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-2.1.x/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-2.2.0/etc/bashrc
#     source $HOME/OpenFOAM/OpenFOAM-2.2.0-debug/etc/bashrc
      source $HOME/OpenFOAM/OpenFOAM-2.2.x/etc/bashrc

### OPENFOAM ###
```

Tips and tricks in OpenFOAM®

OpenFOAM® shell environment

- If at any time I want to use a different version of OpenFOAM®, I just source the right OpenFOAM® environment variables. For instance if I want to switch from version 2.2.x to version 2.1.1, I modify my .bashrc as follows,

Old .bashrc

```
## OPENFOAM ###

#    source $HOME/OpenFOAM/OpenFOAM-1.6-ext/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-1.6-ext.GPU/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-1.7.1/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.0.x/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.1.x/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.2.0/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.2.0-debug/etc/bashrc
     source $HOME/OpenFOAM/OpenFOAM-2.2.x/etc/bashrc

### OPENFOAM ###
```

New .bashrc

```
## OPENFOAM ###

#    source $HOME/OpenFOAM/OpenFOAM-1.6-ext/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-1.6-ext.GPU/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-1.7.1/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.0.x/etc/bashrc
     source $HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.1.x/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.2.0/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.2.0-debug/etc/bashrc
#    source $HOME/OpenFOAM/OpenFOAM-2.2.x/etc/bashrc

### OPENFOAM ###
```

- Then open a new terminal, and you can start using the new version. You can also have two different versions of OpenFOAM® running together at the same time in different terminals.
- You can also use aliases to source the OpenFOAM® version.

Tips and tricks in OpenFOAM®

OpenFOAM® shell environment

- When switching from one OpenFOAM® version to another one, it is highly advisable to clear as many OpenFOAM environment settings as possible, to do so you can use the script unset.sh. In the terminal type:
  - unset
- This script is located in the directory **$WM_PROJECT_DIR/etc/config.**
- In this directory, you will also find many useful scripts. For instance, you can take a look at the script aliases.sh, in this script you will find all the aliases defined in OpenFOAM®

Tips and tricks in OpenFOAM®

Looking for information in OpenFOAM® source code

- To locate files you can use the find command. For instance, if you want to locate a file that contains the string fvPatch in its name you can proceed as follows,
  - find **$FOAM_SOLVERS** -name "*fvPatch*"

  This command will locate the files containing the string "*fvPatch*" in the directory **$FOAM_SOLVERS**

- If you want to find a string inside a file, you can use the command grep
  - grep -r -n LES **$FOAM_SOLVERS**

  This command will look for the string LES in all files inside the directory **$FOAM_SOLVERS**. The option -r means recursive and -n will output the line number.

Tips and tricks in OpenFOAM®

OpenFOAM® Doxygen documentation

- The best source of information is the Doxygen documentation.
- The **$WM_PROJECT_DIR/doc** directory contains the Doxygen documentation of OpenFOAM®.
- Before using the Doxygen documentation, you will need to compile it. To compile the Doxygen documentation, from the terminal:
  - cd **$WM_PROJECT_DIR**
  - ./Allwmake doc

Note: You will need to install doxygen and graphviz/dot

- After compiling the Doxygen documentation you can use it by typing:
  - firefox **file://$WM_PROJECT_DIR/doc/Doxygen/html/**index.html

Tips and tricks in OpenFOAM®

OpenFOAM® mesh quality and checkMesh utility

- Before running a simulation, remember to always check the mesh quality using the checkMesh utility.
- checkMesh will also check for topological errors.
- Remember, topological errors must be repaired.
- You will be able to run with mesh quality errors (such as skewness, aspect ratio, minimum face area, and non-orthogonality), but they will severely tamper the solution accuracy and eventually can made the solver blow-up.
- In my personal experience, I have found that OpenFOAM® is very picky when it comes to mesh quality. So if checkMesh is telling you that the mesh is bad, you better correct those errors.
- Unfortunately, to correct mesh quality errors you will have to remesh the geometry.

Tips and tricks in OpenFOAM®

OpenFOAM® mesh quality and checkMesh utility

- If for any reason checkMesh finds errors, it will give you a message and it will tell you what check failed.
- It will also write a set with the faulty cells, faces, points. These sets are saved in the directory **constant/polyMesh/sets/**
- If at any point you want to visualize the failed faces/cells/points you will need to proceed as follows:
  - foamToVTK -set_type name_of_sets

  where set_type is the type of sets (faceSet, cellSet, pointSet, surfaceFields) and name_of_sets is the name of the set in the directory **constant/polyMesh/sets** (highAspectRatioCells, nonOrthoFaces, wrongOrientedFaces, skewFaces, unusedPoints)

- At the end, foamToVTK will create a directory named **VTK**, where you will find the failed faces/cells/points in VTK format. At this point you can use paraFoam to visualize the failed sets.

Tips and tricks in OpenFOAM®

OpenFOAM® mesh quality and renumberMesh utility

- Before running a simulation, I highly recommend you to run the renumberMesh utility. This utility will renumber the mesh to minimize its bandwidth, in other words, it will make the solver run faster.

- Summarizing, before running a simulation remember to always check the mesh quality using checkMesh and then reduce the mesh bandwidth by using the renumberMesh. This is a good standard practice, I always do it (at least for big meshes).

- Before running a simulation, try to always get a good quality mesh. Remember, garbage in - garbage out.

Tips and tricks in OpenFOAM®

Initial conditions

- First at all, a good initial condition can improve the stability and convergence rate. On the other hand, unphysical boundary conditions can slow down the convergence rate or can cause divergence.
- Use potentialFoam to get an initial solution. It is computational inexpensive.
- You can also get a computational inexpensive solution on a coarse mesh and then interpolate it to a fine mesh. This will be a very good initial solution only if the solution in the coarse mesh is acceptable.
- Use mapFields to map the solution from a coarse mesh to a finer mesh (it also works the other way around).
- If you are running with a turbulence model, you can initialize the velocity and pressure fields from the solution obtained from a laminar case.
- If you are running an unsteady simulation and if the initial transient is of no interest, you can initialize your flow using the solution obtained from a steady simulation.
- I want to remind you again, initial conditions should be physically realistic.

Tips and tricks in OpenFOAM®

Starting and running a simulation

- Remember, you can change the parameters in the dictionaries controlDict, fvSchemes and fvSolution on-the-fly.
- When starting a simulation and if the initial transient is of no interest, you can start the simulation by using a high value of viscosity (low Reynolds number), and as the simulation runs you can change the value until you reach the desired value.
- You can also start the simulation by using a first order scheme and then switch to a second order scheme.
- You can also change the convergence criteria of the linear solvers on the fly. So you can start using a not so tight convergence criterion with a robust numerical scheme, and as the simulation runs you can tighten the convergence criterion to increase the accuracy.
- When the simulation starts I usually use more correctors steps (for piso and pimple schemes), and as the solution advance or stabilizes I decrease the number of corrector steps.
- Remember to always check the time step continuity errors. This number should be small (negative or positive), if it increases in time for sure something is wrong.

Tips and tricks in OpenFOAM®

Discretization schemes

- Never execute production runs using first order schemes. They are too diffusive, which means they will under predict the forces and smear the gradients.
- You can start using a first order scheme, and then switch to a high order scheme. Remember, you can change the parameters on the fly.
- **This means, start robustly and end with accuracy.**
- If you have a good mesh and a good initial solution, you can start immediately with a high order scheme, but be careful, remember to check the stability of the solution.
- Try to always use a good quality mesh. Remember, garbage in - garbage out.
- In the next slides I will show you how I usually setup the discretization schemes.

Tips and tricks in OpenFOAM®

Discretization schemes - Convective terms

- A robust numerical scheme but diffusive,

```
gradSchemes
{
    default         cellMDLimited Gauss linear 1.0;
}
divSchemes
{
    div(phi,U)                      Gauss upwind;
    div(phi,omega)                  Gauss upwind;
    div(phi,k)                      Gauss upwind;
    div((nuEff*dev(T(grad(U)))))    Gauss linear;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes - Convective terms

- An accurate and stable numerical scheme,

```
gradSchemes
{
    default         cellMDLimited Gauss linear 0.5;
}
divSchemes
{
    div(phi,U)                      Gauss linearUpwind grad(U);
    div(phi,omega)                  Gauss linearUpwind grad(U);
    div(phi,k)                      Gauss linearUpwind grad(U);
    div((nuEff*dev(T(grad(U)))))    Gauss linear;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes - Convective terms

- An even more accurate but oscillatory scheme,

```
gradSchemes
{
    default         Gauss linear;
}
divSchemes
{
    div(phi,U)                      Gauss linear;
    div(phi,omega)                  Gauss linearUpwind grad(U);
    div(phi,k)                      Gauss linearUpwind grad(U);
    div((nuEff*dev(T(grad(U)))))    Gauss linear;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes - Diffusive terms

- An accurate numerical scheme on orthogonal meshes,

```
laplacianSchemes
{
        default         Gauss linear limited 1;
}

snGradSchemes
{
        default         limited 1;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes - Diffusive terms

- A less accurate numerical scheme valid on non-orthogonal meshes,

```
laplacianSchemes
{
        default         Gauss linear limited 0.5;
}

snGradSchemes
{
        default         limited 0.5;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes

- If after checking the mesh quality, the non-orthogonality is higher than 80, I prefer to redo the mesh an improve the quality.
- By the way, if you are running LES simulations, do not even think about running in highly non-orthogonal meshes. Better spend more time in getting a good quality mesh, maybe with non-orthogonality less than 60.
- Do not try to push too much the numerical scheme on highly non-orthogonal meshes. You already know that the quality is low, so this highly influence the accuracy and stability of the solution.
- I do not like to run simulations on highly orthogonal meshes, but if I have to, I often use the following numerical schemes.

Tips and tricks in OpenFOAM®

Discretization schemes

- Non-orthogonality more than 80. I do not waste my time. I prefer to go back and get a better mesh. But if you want, you can try something like this,

```
gradSchemes
{
    default      faceLimited leastSquares 1.0;
}
divSchemes
{
    div(phi,U)          Gauss linearUpwind grad(U);
    div(phi,omega)      Gauss linearUpwind;
    div(phi,k)          Gauss linearUpwind;
    div((nuEff*dev(T(grad(U)))))  Gauss linear;
}
```

```
laplacianSchemes
{
    default      Gauss linear limited 0.333;
}

snGradSchemes
{
    default      limited 0.333;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes

- Non-orthogonality between 70 and 80

```
gradSchemes
{
    default      cellMDLimited leastSquares 1.0;
}
divSchemes
{
    div(phi,U)          Gauss linearUpwind grad(U);
    div(phi,omega)      Gauss linearUpwind;
    div(phi,k)          Gauss linearUpwind;
    div((nuEff*dev(T(grad(U)))))  Gauss linear;
}
```

```
laplacianSchemes
{
    default      Gauss linear limited 0.5;
}

snGradSchemes
{
    default      limited 0.5;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes

- Non-orthogonality between 60 and 70

```
gradSchemes
{
    default       cellMDLimited Gauss linear 0.5;
}
divSchemes
{
    div(phi,U)          Gauss linearUpwind grad(U);
    div(phi,omega)      Gauss linearUpwind;
    div(phi,k)          Gauss linearUpwind;
    div((nuEff*dev(T(grad(U)))))  Gauss linear;
}
```

```
laplacianSchemes
{
    default       Gauss linear limited 0.777;
}

snGradSchemes
{
    default       limited 0.777;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes

- Non-orthogonality between 40 and 60

```
gradSchemes
{
    default       cellMDLimited Gauss linear 0.5;
}
divSchemes
{
    div(phi,U)          Gauss linearUpwind grad(U);
    div(phi,omega)      Gauss linearUpwind;
    div(phi,k)          Gauss linearUpwind;
    div((nuEff*dev(T(grad(U)))))  Gauss linear;
}
```

```
laplacianSchemes
{
    default       Gauss linear limited 1.0;
}

snGradSchemes
{
    default       limited 1.0;
}
```

Tips and tricks in OpenFOAM®

Discretization schemes

- Additionally, I also change the number of non-orthogonal corrections.
  - Non-orthogonality between 70 and 80

    nNonOrthogonalCorrectors 4;

  - Non-orthogonality between 60 and 70

    nNonOrthogonalCorrectors 2;

  - Non-orthogonality between 40 and 60

    nNonOrthogonalCorrectors 1;

Tips and tricks in OpenFOAM®

Discretization schemes

- Let us talk about the gradSchemes, this entry defines the way we compute the gradients. The gradients can be computed using Gauss method or the least squares method.
- In practice, the least squares method is more accurate, however, I have found that it tends to be more oscillatory on tetrahedral meshes.
- You can also use gradient limiters, gradient limiters will avoid over and under shoots on the gradient computations. There are four available,

cellMDLimited
cellLimited
faceMDLimited
faceLimited

Less diffusive

More diffusive

- The following entry corresponds to the least squares method using gradient limiters

```
cellMDLimited leastSquares 1.0;
```

Tips and tricks in OpenFOAM®

Under-relaxation factors

- Because of the non-linearity of the equations being solved, it is necessary to control the change of ϕ. This is achieved by under-relaxation as follows,

$$\phi_P^n = \phi_P^{n-1} + \alpha\left(\phi_P^{n^*} - \phi_P^{n-1}\right)$$

here α is the relaxation factor,

$\alpha < 1$	Means under-relaxation. This will slow down the convergence rate but increases the stability.
$\alpha = 1$	Means no relaxation at all. We simple use the predicted value of ϕ.
$\alpha > 1$	Means over-relaxation. It can sometimes be used to accelerate convergence rate but will decrease the stability.

- In plain English, this means that the new value of the variable ϕ depends upon the old value, the computed change of ϕ, and the under-relaxation factor α.

Tips and tricks in OpenFOAM®

Under-relaxation factors

- To understand what are and how to change the under-relaxation factors requires experience and understanding.
- In general, under-relaxation factors are there to suppress oscillations.
- Small under-relaxation factors will significantly slow down convergence. Sometimes to the extent that the user thinks the solution is converged when it really is not.
- The recommendation is to always use under-relaxation factors that are as high as possible, without resulting in oscillations or divergence.
- I highly recommend you to use the default under-relaxation factors.
- If you have to change the under-relaxation factors, start by decreasing the values of the under-relaxation factors in order to improve the stability of the solution.
- In order to find the optimal under-relaxation factors for your case, it will take some trial and error.

Tips and tricks in OpenFOAM®

Under-relaxation factors

- These are the under-relaxation factors I often use (which by the way are the default values).

```
relaxationFactors
{
    p               0.3;
    U               0.7;
    k               0.7;
    omega           0.7;
}
```

- According to the physics involved you will need to add new under-relaxation factors.

Tips and tricks in OpenFOAM®

Linear solvers

- The linear solvers are defined in the fvSolution dictionary. You will need to define a solver for each variable you are solving.
- The solvers can be modified on-the-fly.
- The GAMG solver (generalized geometric-algebraic multigrid solver), can often be the optimal choice for solving the pressure equation. However, I have found that when using more than 1024 processors is better to use Newton-Krylov type solvers.
- When solving multiphase problems, I have found that the GAMG solver may give problems when running in parallel, so be careful. The problem is mainly related to nCoarsestCells keyword, so usually I have to set a high value of cells (in the order 1000).
- The utility renumberMesh can dramatically increase the speed of the linear solvers, specially during the first iterations.
- I usually start by using a not so tight convergence criterion and as the simulation runs, I tight the convergence criterion.

Tips and tricks in OpenFOAM®

Linear solvers

- These are the linear solvers I often use.
- For the pressure equation I usually start with a tolerance equal to 1e-5 and relTol equal to 0.01. After a while I change these values to 1e-6 and 0.0 respectively.

```
p
{
    solver                  GAMG;
    tolerance               1e-5;
    relTol                  0.01;
    smoother                GaussSeidel;
    nPreSweeps              0;
    nPostSweeps             2;
    cacheAgglomeration      on;
    agglomerator            faceAreaPair;
    nCellsInCoarsestLevel   100;
    mergeLevels             1;
}
```

Tips and tricks in OpenFOAM®

Linear solvers

- These are the linear solvers I often use.
- For the pressure equation I usually start with a tolerance equal to 1e-5 and relTol equal to 0.01. After a while I change these values to 1e-6 and 0.0 respectively.
- If I do not use the GAMG solver for the pressure, I often use this solver.
- If the speed of the solver is too slow, I change the convergence criteria to 1e-4 and relTol equal to 0.01. I usually do this during the first iterations.

```
p
{
    solver          PCG;
    preconditioner  DIC;
    tolerance       1e-5;
    relTol          0.01;
}
```

Tips and tricks in OpenFOAM®

Linear solvers

- These are the linear solvers I often use.
- For the velocity equation I always use a tolerance equal to 1e-8 and relTol equal to 0.0.

```
U
{
    solver          PBiCG;
    preconditioner  DILU;
    tolerance       1e-8;
    relTol          0.0;
}
```

Tips and tricks in OpenFOAM®

Transient simulations and time-step selection

- Remember, the fact that we are using an implicit solver (unconditionally stable), does not mean that we can choose a time step of any size.
- The time-step must be chosen in such a way that it resolves the time-dependent features and it maintain the solver stability.
- In my personal experience, I have been able to go up to a CFL = 5.0. But to achieve this I had to increase the number of corrector steps in the pimple solver, decrease the under-relaxation factors and tightening the convergence criteria; and this translate into a higher computational cost.
- As I am often interested in the unsteadiness of the solution, I usually use a CFL number not larger than 1.5.
- If accuracy is the goal (e.g., predicting forces), definitively use a CFL less than 1.0.
- If you are doing LES simulations, I highly advice you to use a CFL less than 0.6 and if you can afford it, less than 0.3

Tips and tricks in OpenFOAM®

Transient simulations and time-step selection

- I usually use the pimple solver. In the pimple solver you have the option to limit your time-step to a maximum CFL number, so the solver automatically choose the time-step.
- In the piso solver you need to give the time-step size. So basically you need to choose your time-step so it does not exceed a target CFL value.
- You can also modify the piso solver so it automatically choose the time-step according to a maximum CFL number.
- Remember, setting the keyword nOuterCorrectors equal to 1 in the pimple solver is equivalent to use the piso solver.
- The pimple solver is a solver specially formulated for large time-steps. So in order to increase the stability you will need to add more corrector steps (nOuterCorrectors and nCorrectors).
- A smaller time-step may be needed in the first iterations to maintain solver stability. Also, have in mind that the first time steps may take longer to converge, do not be alarmed of this behavior, it is normal.

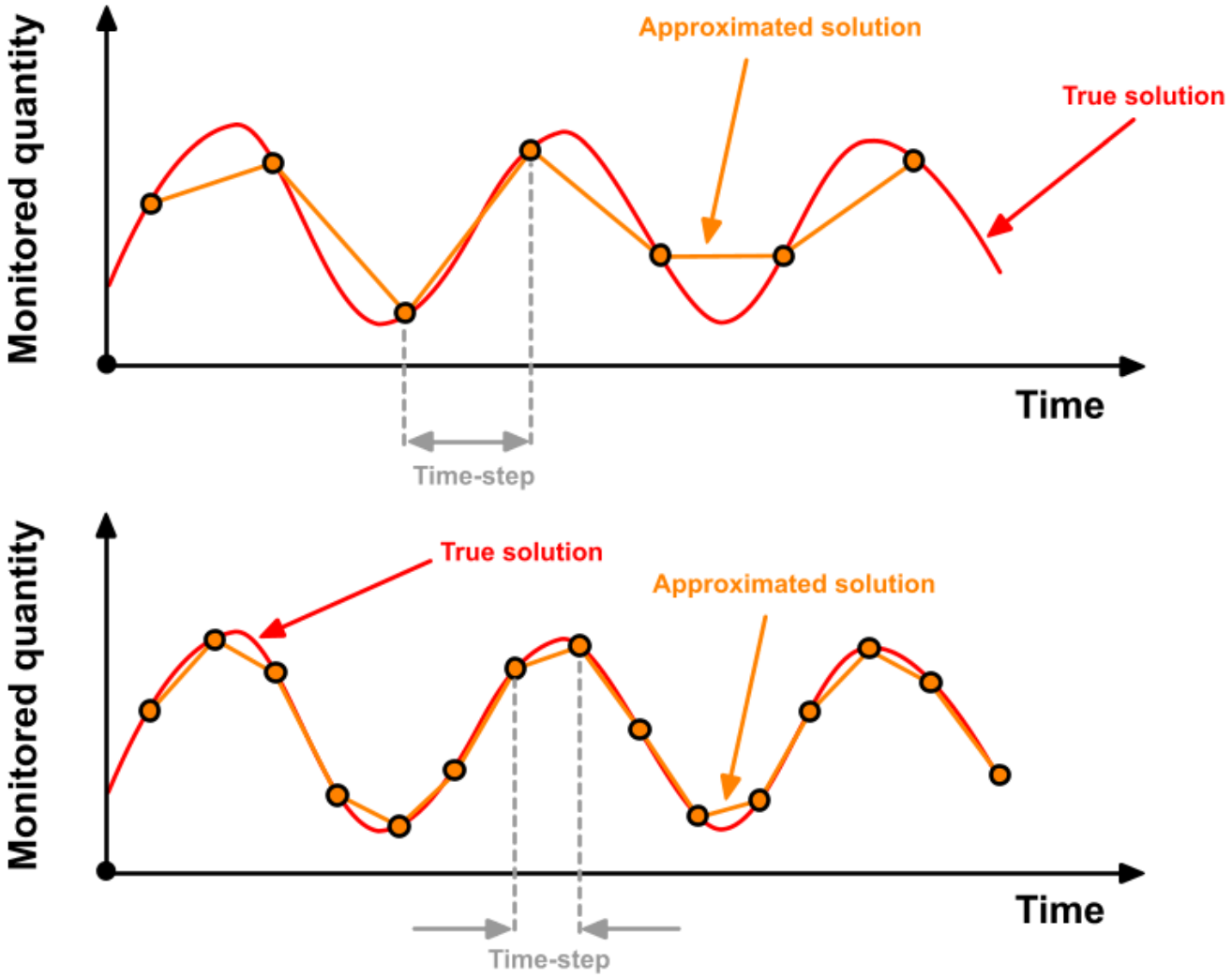
Tips and tricks in OpenFOAM®

Transient simulations and time-step selection

- If you are interested in the initial transient, you should start using a high order discretization scheme, with a tight convergence criterion, and the right flow properties.
- If you start from an approximate initial solution obtained from an steady simulation, the initial transient will not be accurate.
- If the solution is not converging or is taking too long to converge, try to reduce your time-step.
- If you use the first order Euler scheme, try to use a CFL number less than 1.0 and preferably in the order of 0.5, this is in order to keep temporal diffusion to a minimum. Yes, diffusion also applies to time.
- Remember, you can change the temporal discretization on-the-fly. As for the spatial discretization, first order schemes are robust and second order schemes are accurate.
- Euler is a first order implicit scheme (bounded), backward is a second order implicit scheme (unbounded) and Crank-Nicolson is a second order implicit scheme (bounded).

Tips and tricks in OpenFOAM®

Transient simulations and time-step selection



Tips and tricks in OpenFOAM®

Notes on convergence

- Determining when the solution is converged can be really difficult.
- Solutions can be considered converged when the flow field and scalar fields are no longer changing, but usually this not the case for unsteady flows. Most flows in nature and industrial applications are highly unsteady.
- It is a good practice to monitor the residuals. But be careful, residuals are not always indicative of a converged solution. The residuals may reach the tolerance criterion, but the flow field may be continuously changing from instant to instant.
- In order to properly assess convergence, it is also recommended to monitor a physical quantity. If this physical quantity does not change in time you may say that the solution is converge.
- But be careful, it can be the case that the monitored quantity exhibit a periodic oscillatory behavior, in this case you may say that you have reached a converged periodic solution.
- Remember, for unsteady flows you will need to analyze the solution in a given time window. Do not take the last solution as the final answer.

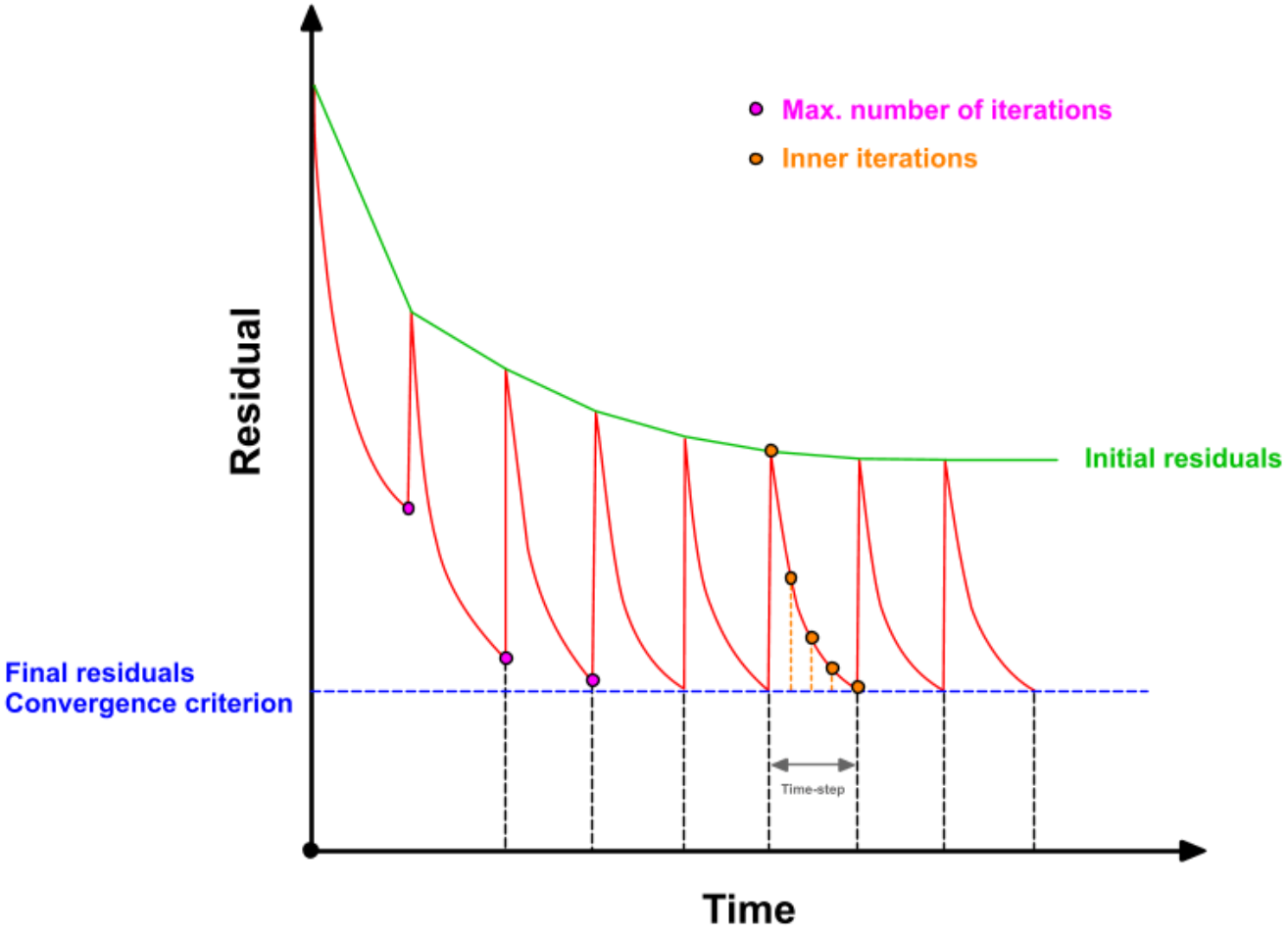
Tips and tricks in OpenFOAM®

Notes on convergence

- If your goal is predicting forces (e.g., drag prediction). You should monitor the force, and use it as your convergence criterion.
- In general, overall mass balance should be satisfied. Remember, the method is conservative.
- Residuals are not your solution. Low residuals do not automatically mean a correct solution, and high residuals do not automatically mean a wrong solution.
- Final residuals are often higher with higher order discretization schemes than with first order discretization. That does not mean that the first order solution is better.
- Always ensure proper convergence before using a solution. A not converged solution can be misleading when interpreting the results.

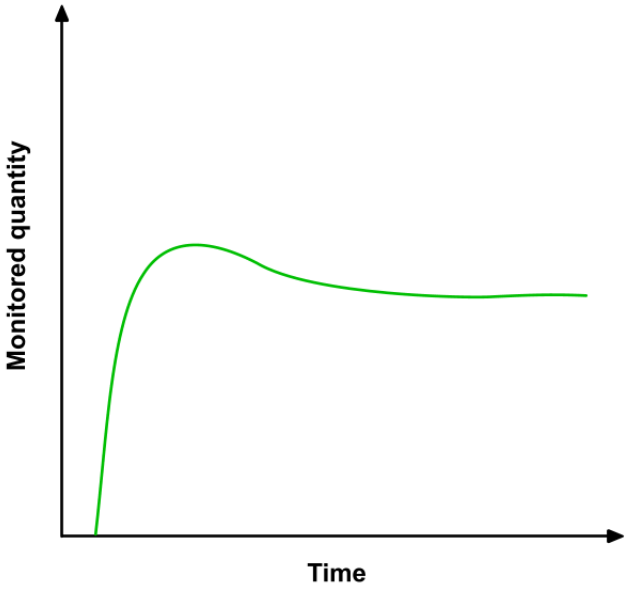
Tips and tricks in OpenFOAM®

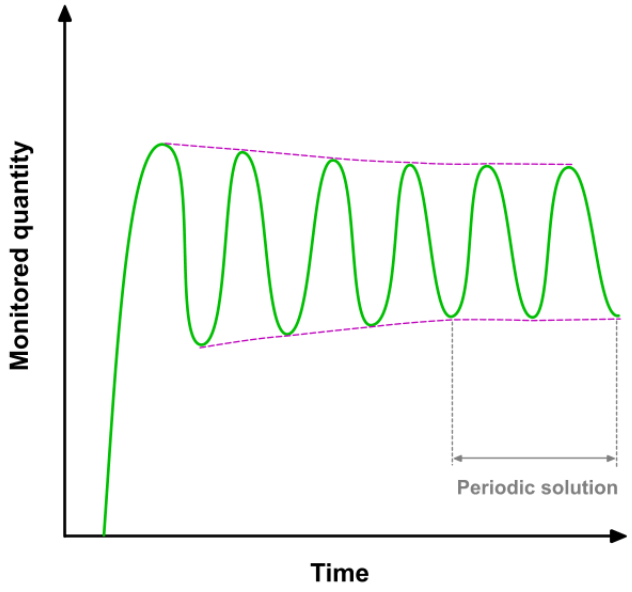
Notes on convergence

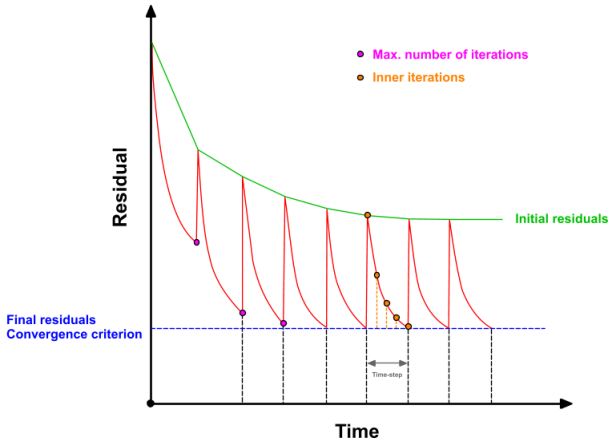


Tips and tricks in OpenFOAM®

Notes on convergence







Tips and tricks in OpenFOAM®

Mesh quality

- **I do not know how much to stress this, but try to always use a good quality mesh. Remember, garbage in - garbage out.**
- Use hexahedral meshes whenever is possible, specially if high accuracy in predicting forces is your goal (e.g., drag prediction).
- For the same cell count, hexahedral meshes will give more accurate solutions, especially if the grid lines are aligned with the flow.
- The mesh density should be high enough to capture all relevant flow features.
- Increasing the cells count will likely improve the solution accuracy, but at the cost of a higher computational cost.
- Change in cell size should be smooth.
- In boundary layers, quad, hex, and prism/wedge cells are preferred over triangles, tetrahedras, or pyramids.

Tips and tricks in OpenFOAM®

On the incompressible solvers

- For those using the incompressible solvers (simpleFoam, pisoFoam, pimpleFoam), I want to remind you that the pressure used by these solvers is the modified pressure or

$$P = \frac{p}{\rho}$$

So do not forget to multiply the pressure by the density in order to get the physical pressure. This will not made much different if you are working with air, but is you are working with water, you will notice the difference.

Thank you for your attention