Disclaimer

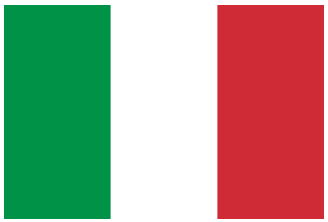
“This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM® and OpenCFD® trade marks.”

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

guerrero@wolfdynamics.com

UNIVERSITÀ DEGLI STUDI DI GENOVA

wolf dynamics

Damiano NATALI

damiano.natali@unige.it

natali@wolfdynamics.com

UNIVERSITÀ DEGLI STUDI DI GENOVA

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.

Post-processing

Big spoiler alert for those who like cool and fantastic looking CFD pictures.

Post-processing

As we are going to do some post-processing, I want to remind you that CFD does not stand for

- **C**olorful **F**luid **D**ynamics
- **C**areful **F**it of **D**ata

I highly advise you to always be skeptical with the results and use common sense when interpreting the results.

Cool and fantastic looking pictures do not imply that the results are correct.

Post-processing

Always remember, the solution not only depends on the mesh resolution, but also on the numerical scheme. So every change in the mesh or numerical scheme can give you very different results.

There are several different factors that can affect the solution, namely:

- Round-off errors.
- Iteration errors.
- Discretization error.
- Model errors.
- Programming erros.
- User errors.

So, you need to know what are you doing and need to use common sense when interpreting the results.

Post-processing

Remember, what you have done could be completely physically wrong, so colorful pictures do not mean physically correct results.

There is no way to know if your solution is good without knowing the physics involve and the theoretical background. Specially, if you did not take part of the solution process.

Always be skeptical with the results (do not believe any simulation results).

If experimental data is available, compare it with the numerical solution, maybe this is the best way of validating.

But again, the experimental data can be wrong or biased due to measurements errors.

Today's lecture

1. Data visualization with paraFoam
2. Data visualization with VISIT
3. Data manipulation and conversion
4. The sample utility
5. Probes
6. functionObjects
7. foamLog
8. More post-processing utilities
9. Hands-on session

Today's lecture

1. **Data visualization with paraFoam**
2. Data visualization with VISIT
3. Data manipulation and conversion
4. The sample utility
5. Probes
6. functionObjects
7. foamLog
8. More post-processing utilities
9. Hands-on session

Data visualization with paraFoam

- paraFoam is the main post-processor distributed with OpenFOAM®. However, you can use other alternatives (commercial or open source).

- paraFoam is a wrapper of a third-party open source product named Paraview (www.paraview.org).

- Paraview is based on VTK, the visualization toolkit (www.vtk.org).

- paraFoam comes with many built-in filters. By using these filters you can manipulate your data in order to create vectors, streamlines, iso-surfaces, cut-planes, plot over a lines, and so on.

Data visualization with paraFoam

We will now do some post-processing using paraFoam. From now on follow me.

- Go to the **postprocessing/cavity3d/** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/cavity3d/c1**
  - blockMesh
  - icoFoam
  - paraFoam
- paraFoam generates a file named case_name.OpenFOAM (where case_name is the name of the case, cavity3d in this tutorial), which is necessary since Paraview needs a file to be specified.
- If you want to read multiple cases, you will need to generate the case_name.OpenFOAM file manually. In the terminal,
  - touch new_case_name.OpenFOAM

Data visualization with paraFoam

We will now do some post-processing using paraFoam. From now on follow me.

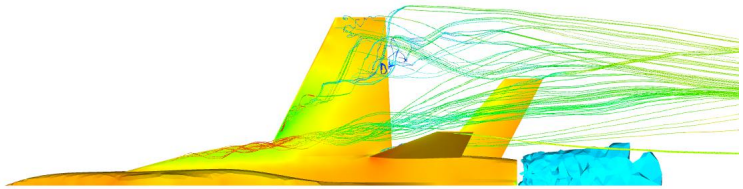
- Go to the **postprocessing/yf17/** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/yf17**
  - cd **c1**
  - cp -r **fs_0.20 0.20**
  - paraFoam
- We do not need to run this case, the solution is in the directory **postprocessing/yf17/fs_0.20.**
- The mesh is in the folder **postprocessing/yf17/mesh**.
- At this point, try to get familiar with the user interface and most important, try to use all the features available in paraFoam.

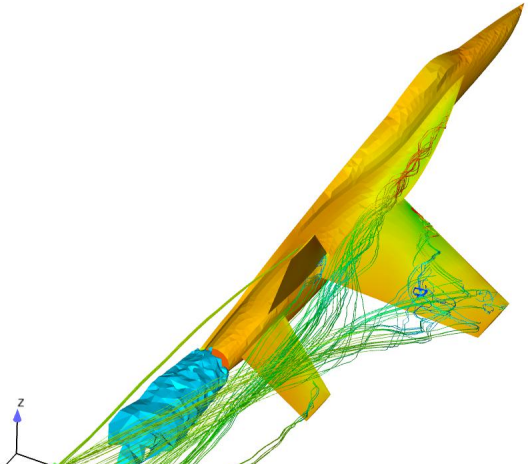
Data visualization with paraFoam

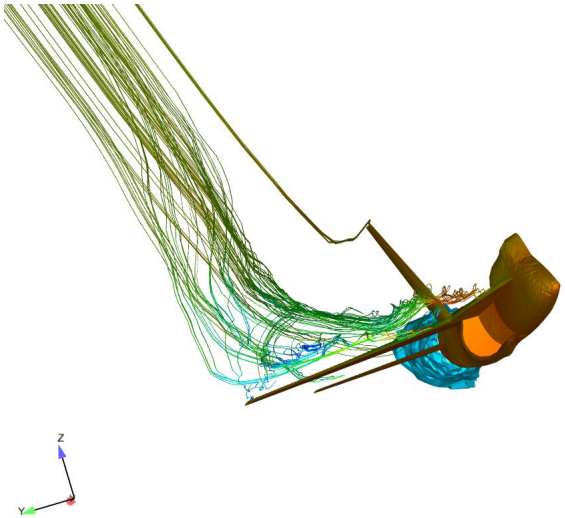
Try to capture the vortex

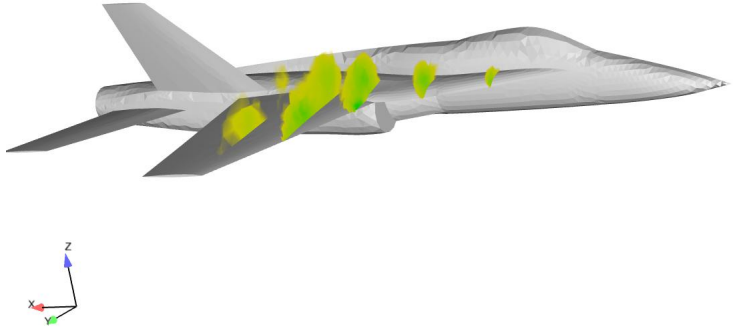
Data visualization with paraFoam

Try to capture the vortex



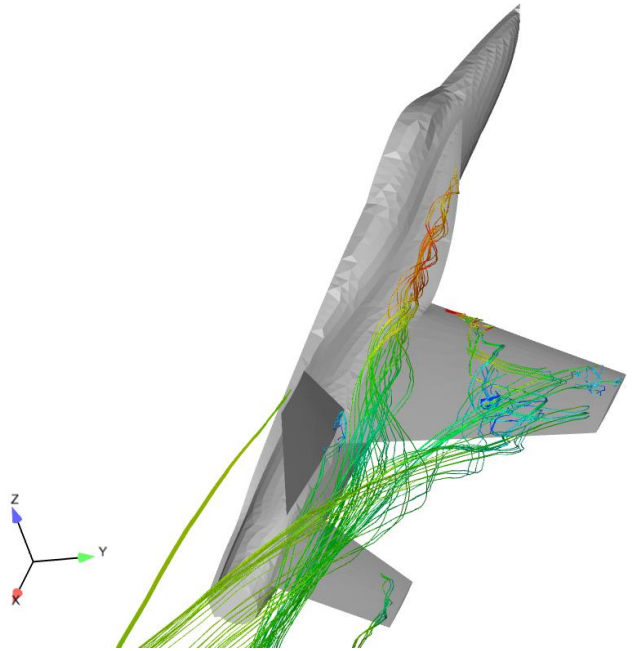






Data visualization with paraFoam

Try to capture the vortex



Today's lecture

1. ~~Data visualization with paraFoam~~
2. **Data visualization with VISIT**
3. Data manipulation and conversion
4. The sample utility
5. Probes
6. functionObjects
7. foamLog
8. More post-processing utilities
9. Hands-on session

Data visualization with VISIT

- VISIT is an open source interactive parallel visualization and graphical analysis tool.
- VISIT was developed by the U.S. Department of Energy (DOE) Advanced Simulation and Computing Initiative (ASCI) to visualize and analyze the results of terascale simulations
- VISIT contains a rich set of visualization filters for data manipulation.
- VISIT can read OpenFOAM® native format.
- VISIT can also read other formats, such as VTK format. To convert the OpenFOAM® results to VTK format you need to use the foamToVTK utility.
- VISIT can be downloaded from the following site: https://wci.llnl.gov/codes/visit/home.html
- Personally speaking, for large datasets, I do prefer to use VISIT rather than paraFoam.

Data visualization with VISIT

Let us do some post-processing using VISIT.

From now on follow me.

- Go to the **postprocessing/damBreak** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/damBreak/c1**
  - blockMesh
  - cp **0/**alpha1.org **0/**alpha1
  - setFields
  - interFoam
- To do post-processing with VISIT, let us first convert the case to VTK format. In the terminal type:
  - foamToVTK
  - cd **VTK**
  - visit (I added this alias to my .bashrc file)

Data visualization with VISIT

Let us do some post-processing using VISIT.

From now on follow me.

- Go to the **postprocessing/vespacoarse** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/vespacoarse/c1**
- We do not need to run this case, the solution is in the directory **postprocessing/vespacoarse/c1/fs_500**
- To do post-processing with VISIT, let us first convert the case to VTK format. In the terminal type:
  - cp -r **fs_500 500**
  - foamToVTK -latestTime
  - cd **VTK**
  - visit (I added this alias to my .bashrc file)

Data visualization with VISIT

Let us do some post-processing using VISIT.

From now on follow me.

- The mesh was generated using snappyHexMesh, so feel free to take a look at the snappyHexMesh dictionary and the .stl file containing the geometry.
- I really like this geometry, it is an actual 3D scan of a Vespa scooter. I would like to thanks Paul McIntosh for sharing the geometry and the case setup (http://www.vespalabs.org).
- At this point, try to get familiar with the user interface and most important, try to use all the features available in VISIT.

Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. **Data manipulation and conversion**
4. The sample utility
5. Probes
6. functionObjects
7. foamLog
8. More post-processing utilities
9. Hands-on session

Data manipulation and conversion

- OpenFOAM® comes with many data manipulation utilities. These utilities can be found in the **$FOAM_UTILITIES** folder (use alias util to go there).

- OpenFOAM® utilities can also be cloned and customized to create user defined utilities. Always remember to develop your own utilities in your user folder (**WM_PROJECT_USR_DIR**) and to keep the same directory structure as in the original OpenFOAM® installation.

- In the **postProcessing** folder, you will find the data manipulation utilities for post-processing. For example, in the sub-directory **postProcessing/velocityField** you will find the following utilities source code directories: **Co**, **Pe**, **enstrophy**, **Q**, **flowType**, **streamFunction**, **Lambda**, **uprime**, **Mach**, **vorticity.**

Data manipulation and conversion

- Inside each utility directory you will find a *.C file with the same name as the directory. This is the main file, where you will find the top-level source code and a short description of the utility.

- For instance, in the directory **Q**, you will find the file Q.C, which is the source code of the utility Q. In the source code you will find the following description:

  Calculates and writes the second invariant of the velocity gradient tensor.

- **Take your time and dig into each directory to get a complete description of each utility.**

Data manipulation and conversion

- It is also possible to export data to be visualized with other post-processing tools:
  - foamDataToFluent: converts OpenFOAM® data to Fluent format.
  - foamToEnsight: converts OpenFOAM® data to EnSight format.
  - foamToGMV: converts OpenFOAM® data to GMV.
  - foamToVTK: converts OpenFOAM® data to VTK format.
  - foamToTecplot360: converts OpenFOAM® data to tecplot format

In the User Guide you will find the complete listing

Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. ~~Data manipulation and conversion~~
4. **The sample utility**
5. Probes
6. functionObjects
7. foamLog
8. More post-processing utilities
9. Hands-on session

The sample utility

- OpenFOAM® provides the sample utility to sample field data for plotting on graphs.
- The sampling locations are specified for a case through a sampleDict dictionary located in the case **system** directory.
- Data can be written in a range of formats including well-known graphing packages such as: grace/xmgr, gnuplot and jPlot.
- The sampling can be executed by running the utility sample in the case folder and according to the application syntax.
- A final word, this utility does not do the sampling while the solver is running. It does the sampling after you finish the simulation.

For more information about the sample utility, refer to the user guide.

The sample utility

We will do now some post-processing using the sample utility. From now on follow me.

- Go to the **postprocessing/turb_backstep** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/turb_backstep/c1**
  - blockmesh
  - setDiscreteFields (if you do not have this tool, copy the files located in the directory **0init** to the directory **0**)
  - simpleFoam
  - sample -latestTime
- At this point take a look at the case folder. You will notice that the sample utility creates a new folder named **postProcessing**, which contains the folder **sets** where you will find **n** time folders (one for each time step sampled).

The sample utility

You will find in the case folder a small shell script named script. From the terminal type:

- ./script

This script will generate two output files in .eps format using the gnuplot scripts located in the case folder (profilek.gp and profileU.gp). Take a look at these files, this is a part of the post-processing tutorials.

To visualize the .eps files, from the terminal type:

- evince profilek.eps
- evince profileU.eps

or whatever application you use to view .eps files

- Let us now study the sampleDict dictionary. Go to the case **system** folder and open the file sampleDict.

The sample utility

- Let us now study the sampleDict dictionary. The probesDict dictionary contains the following entries:

```
// Set output format
        setFormat raw;
// Surface output format.
        surfaceFormat foamFile;
// interpolationScheme.
        interpolationScheme cellPointFace;
// Fields to sample.
        fields
        (
            U
        );
// Set sampling definition:
        sets
        (
            x=0H
            {
                type          midPoint;
                axis          z;
                start         (0 0.5 1);
                end           (0 0.5 4);
            }
        );
// Surface sampling definition:
        surfaces
        (
        );
```

For more information about the sample utility, refer to the user guide.

The sample utility

The sampleDict file

- The sampleDict file contains several entries to be set according to the user needs. There you can set,
  - The choice for the interpolationScheme.
  - The format of line data output.
  - The format of surface data output.
  - The fields to be sample.
  - The sub-dictionaries that controls each sampling operation. In these sub-dictionaries you set the name, type and geometrical information of the sampling operation.

For more information about the sample utility, refer to the user guide.

Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. ~~Data manipulation and conversion~~
4. ~~The sample utility~~
5. **Probes**
6. functionObjects
7. foamLog
8. More post-processing utilities
9. Hands-on session

Probes

- OpenFOAM® provides the probeLocations utility to sample field data in some discrete points.
- Probes can be created by using the probesDict dictionary, which must be contained in the **system** folder.
- During the job execution, inside the case directory a new folder named **postProcessing/probes**, will be created. In this folder, the probed values are stored.
- The probed values are saved in a ascii file, with the name of the probed field.
- The probing can be executed by running the utility probeLocations in the case folder and according to the application syntax.

Probes

We will now run a simulation using some probes. From now on follow me.

- Go to the **postprocessing/pitzDaily** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/pitzDaily/c1**
  - blockMesh
  - pisoFoam
  - probeLocations
- Let us explore the case folder. You will see that now we have a new folder named **postProceing/probes,** which contains the **0** time folder, this is the folder from which we started to sample the probes.
- A final word, this utility does not probe while the solver is running. It does the probing after you finish the simulation.

Probes

- Let us now study the probesDict dictionary. Go to the case **system** folder and open the file probesDict. The probesDict dictionary contains the following entries:

```
// Fields to be probed. runTime modifiable!
    fields
    (
        p
    );

// Locations to be probed. runTime modifiable!
// You can add as many as you like
    probeLocations
    (
        ( 0.0254 0.0253 0 )
        ( 0.0508 0.0253 0 )
        ( 0.0762 0.0253 0 )
        ( 0.1016 0.0253 0 )
        ( 0.127 0.0253 0 )
        ( 0.1524 0.0253 0 )
        ( 0.1778 0.0253 0 )

    );
```

Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. ~~Data manipulation and conversion~~
4. ~~The sample utility~~
5. ~~Probes~~
6. **functionObjects**
7. foamLog
8. More post-processing utilities
9. Hands-on session

functionObjects

- It is possible to perform some data extraction/manipulation operations while the simulation is running by using the functionObjects feature.
- functionObjects are small pieces of code executed at a regular interval without explicitly being linked to the application.
- Using functionObjects, files of sampled data can be written for graph plotting and post processing.
- functionObjects are specified in the controlDict dictionary and executed every timestep (or regular intervals).
- All functionObjects are runtime modifiable.

functionObjects

- Below is a list of some of the available functionObjects that can be called at run-time:
  - fieldAverage - temporal averaging of fields.
  - forces - calculates pressure/viscous forces and moments.
  - fieldValue - averaging/integration across sets of faces/cells.
  - sets - data sampling along lines, e.g. for graph plotting.
  - probes - data probing at point locations.
  - streamlines - generates streamlines in one of the sample formats.
  - isoSurface - generation of an isoSurface of given fields in one of the standard sample formats.
  - cuttingPlane - generation of a cuttingPlane with field data in one of the sample formats.
  - fieldMinMax - writes min/max values of fields.

functionObjects

We will now run a simulation using functionObjects. From now on follow me.

- Go to the **postprocessing/2d_cylinder** folder. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/2d_cylinder/c1**
  - blockMesh
  - icoFoam > log &
  - tail -f log
- Let us explore the case folder. You will see that we now have a new directory named **postProcessing**. Inside this directory, you will find many subdirectories pointing to the functionObject used.

- In this case you will find the following directories: **forceCoeffsCyl, forcesCyl, minmaxdomainCyl, probes1, probes2** and **surfaceSampling,** each one containing the **0** time folder, this is the folder from which we started to sample using functionObjects.

functionObjects

We will now run a simulation using functionObjects. From now on follow me.

- Let us now study the controlDict dictionary. Go to the case system folder and open the file controlDict.

- At the end of the controlDict dictionary, you will find a new sub-dictionary entry, here all the functionObjects are defined.

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
    fieldAverageCyl
    {
        type            fieldAverage;

        functionObjectLibs ("libfieldFunctionObjects.so");

        enabled         true;

        outputControl   outputTime;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }

        );
    }
```

Name of functionObject

functionObject to use

Library to use

Keywords and sub-dictionaries specific to the functionObject

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
fieldAverageCyl
  {
     type            fieldAverage;
     functionObjectLibs ("libfieldFunctionObjects.so");
     enabled         true;
     outputControl   outputTime;

     fields
     (
        U
        {
           mean        on;
           prime2Mean  off;
           base        time;
        }

        p
        {
           mean        on;
           prime2Mean  off;
           base        time;
        }
     );
  }
```

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
forcesCyl
{
        type forces;
        functionObjectLibs ("libforces.so");

        outputControl   timeStep;
        outputInterval  1;

        patches ("cylinder");

        pName p;
        Uname U;

        rhoName rhoInf;
        rhoInf 1.225;

        CofR (0 0 0);
}
```

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
forceCoeffsCyl
{
        type forceCoeffs;
        functionObjectLibs ("libforces.so");
        patches ("cylinder");

        pName p;
        Uname U;
        rhoName rhoInf;
        rhoInf 1.0;

        log true;

        CofR (0.0 0 0);
        liftDir (0 1 0);
        dragDir (1 0 0);
        pitchAxis (0 0 1);
        magUInf 1.0;
        lRef 1.0;
        Aref 1.0;

      outputControl   timeStep;
      outputInterval  1;
}
```

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
minmaxdomainCyl
{
        type fieldMinMax;

        functionObjectLibs ("libfieldFunctionObjects.so");

        enabled true;

        mode component;

        outputControl timeStep;
        outputInterval 1;

        log true;

        fields (p U);
}
```

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
probes1
{
        type probes;
        functionObjectLibs ("libsampling.so");

        probeLocations
        (
        (2 1 0)
        );
        fields
        (p U);

       outputControl      timeStep;
       outputInterval     1;
}
```

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
probes2
{
        type probes;
        functionObjectLibs ("libsampling.so");

        probeLocations
        (
        (5 0.5 0)
        (5 0 0)
        (5 -0.5 0)
        (10 0.5 0)
        (10 0 0)
        (10 -0.5 0)
        );
        fields
        (p U);

       outputControl      timeStep;
       outputInterval     1;
}
```

functionObjects

- The functionObject entry in the controlDict dictionary, contains the following information:

```
surfaceSampling
  {
      type surfaces;
      functionObjectLibs ("libsampling.so");
      enabled         true;
      outputControl   outputTime;
      interpolationScheme cellPoint;
      surfaceFormat raw;

      // Fields to be sampled
      fields
      (
          p
      );

      surfaces
      (
          nearWall
          {
              type            patch;
              patches         ( "cylinder" );
              interpolate     true;
              triangulate     false;
          }
      );
  }
```

functionObjects

- In the previous example we only used a few of the functionObjects available in OpenFOAM®, namely:

fieldAverage
fieldMinMax
forceCoeffs
forces
probes
sets
surfaces

For more information about functionObjects, refer to the user guide.

Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. ~~Data manipulation and conversion~~
4. ~~The sample utility~~
5. ~~Probes~~
6. ~~functionObjects~~
7. **foamLog**
8. More post-processing utilities
9. Hands-on session

foamLog

- During execution, OpenFOAM® writes values of residuals, number of iterations, and so on, on the standard output (terminal).
- It is possible to extract this information with the foamLog utility, provided the standard output has been written to a file (logfile).
  - solver_name > logfile
  - foamLog logfile
- By default, the files are presented in two-column format of time and the extracted values. You can plot each file by using gnuplot or your favorite plotting tool. Depending of your case setup, the following files might be present:

  courant_0, contCumulative_0, contGlobal_0, contLocal_0

  foamLog.awk, executionTime_0, Separator_0, Time_0

  p_0 , pFinalRes_0, pIters_0

  Ux_0, UxFinalRes_0, UxIters_0

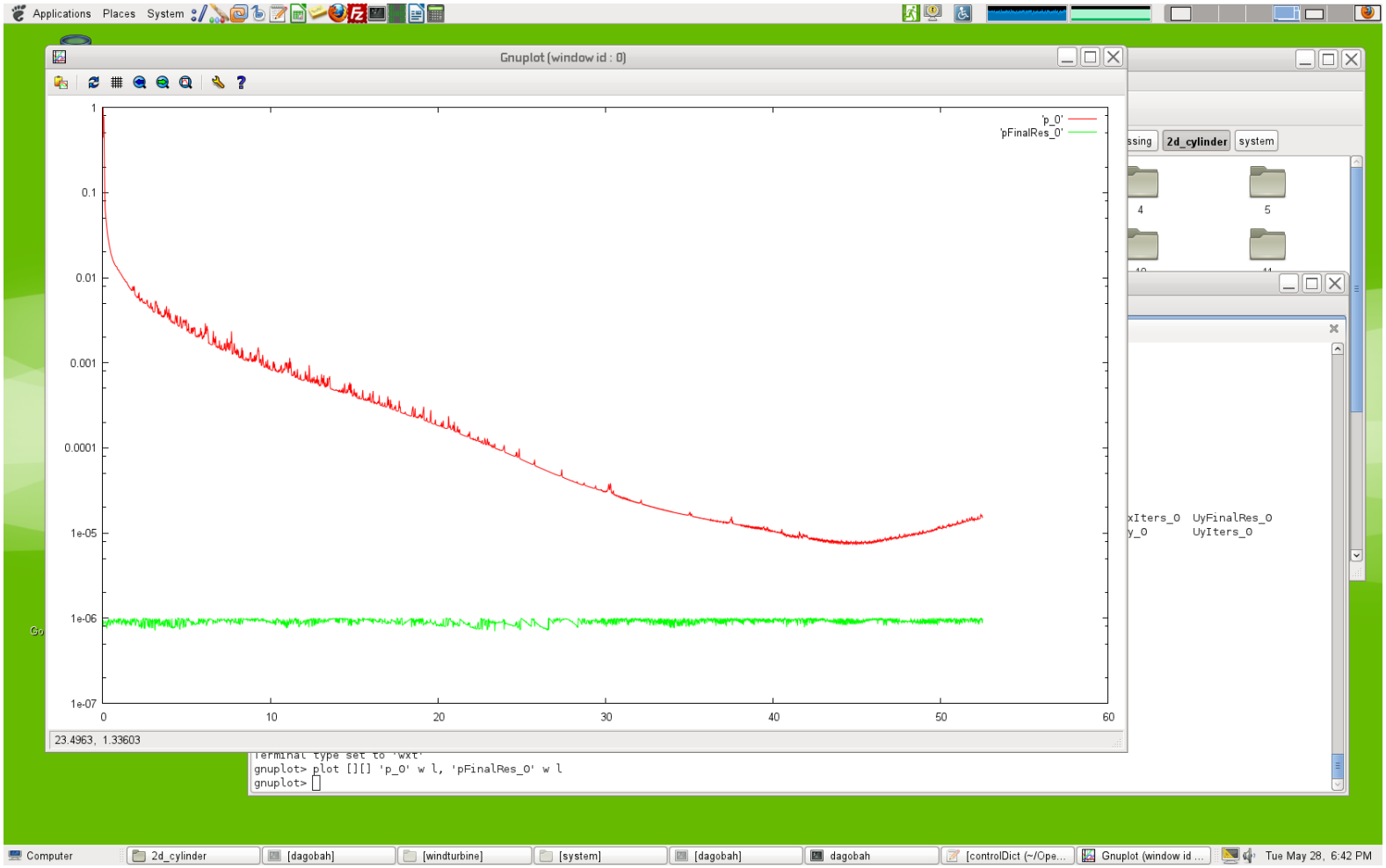
  Uy_0, UyFinalRes_0 , UyIters_0

foamLog

- All the information extracted from the log file is saved in the directory **logs.**
- To plot this info we can use gnuplot, grace, xmgr, octave and/or jPlot. (just to name a few).
- Let us analyze the log of the case **postprocessing/2d_cylinder**. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/2d_cylinder/c1**
  - foamLog log
  - cd **logs**
  - gnuplot
    - plot ‘CourantMean_0’ w l
    - plot ‘p_0’ w l, ‘pFinalRes_0’ w l

foamLog

- This is a screenshot on my computer. This a plot of the pressure initial and final residual using gnuplot. You can plot any of the output files generated by foamLog.



foamLog

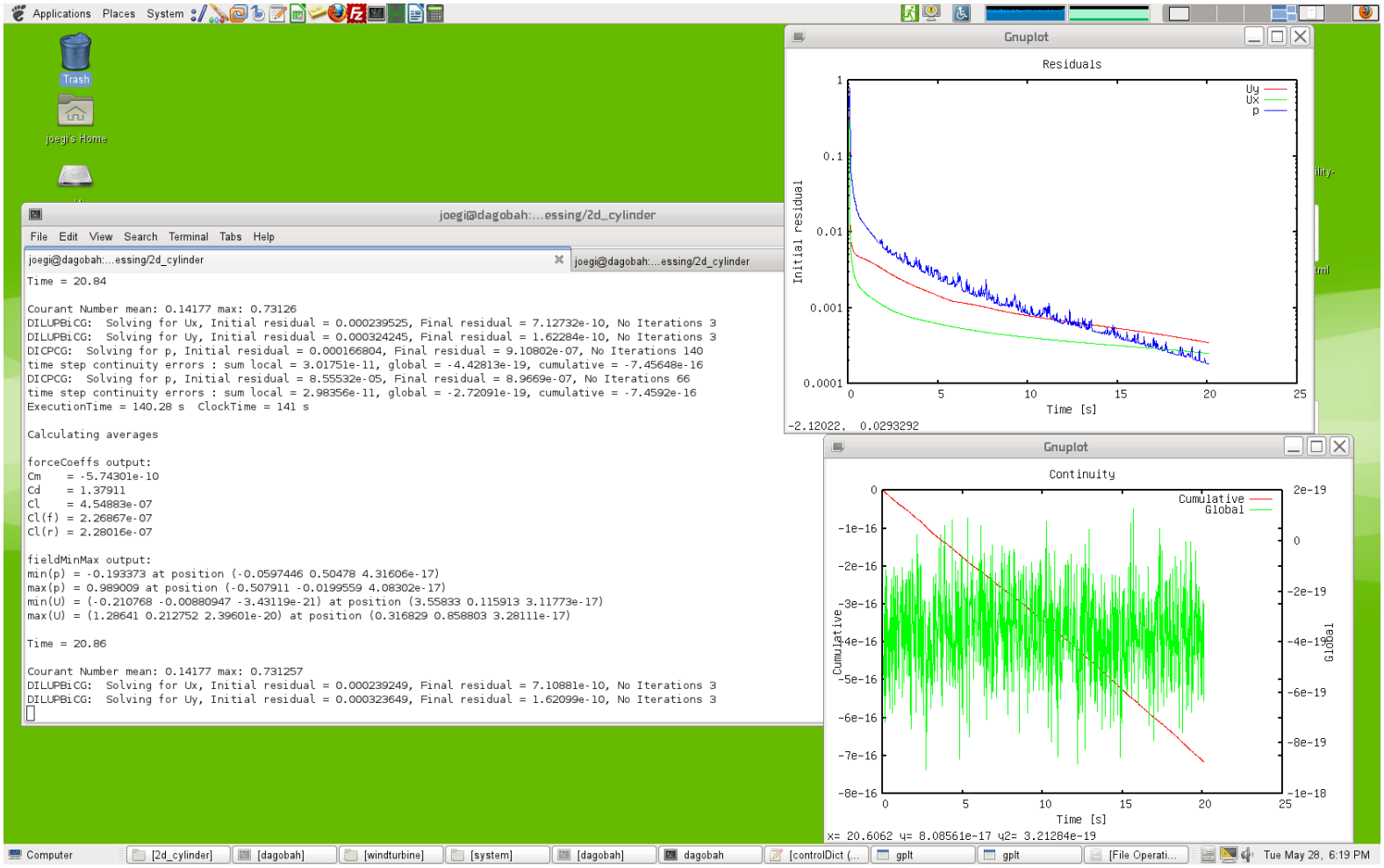
- It is also possible to plot this information on-the-fly. To do this you will need to install PyFoam and use the following utility:
  - pyFoamPlotRunner.py [options] <foamApplication>

- Let us run again the case **postprocessing/2d_cylinder** but using pyFoamPlotRunner.py this time. In the terminal type:
  - cd **$path_to_openfoamcourse/postprocessing/2d_cylinder**
  - pyFoamPlotRunner.py icoFoam

- If you need help or want to know all the options available, type in the terminal,
  - pyFoamPlotRunner.py --help

foamLog

- This is a screenshot on my computer. In this case, pyFoamPlotRunner is showing the initial residuals and continuity errors on the fly



Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. ~~Data manipulation and conversion~~
4. ~~The sample utility~~
5. ~~Probes~~
6. ~~functionObjects~~
7. ~~foamLog~~
8. **More post-processing utilities**
9. Hands-on session

More post-processing utilities

- In OpenFOAM®, you can output on the fly streamlines, cutting planes, iso-surfaces, wall bounded stream lines, forces and force coefficients written into data bins.

- In the folder **$FOAM_UTILITIES/postProcessing, $FOAM_SRC/postProcessing, $FOAM_SRC/sampling,** you will find the directories containing the source code for many post-processing utilities available in OpenFOAM®.

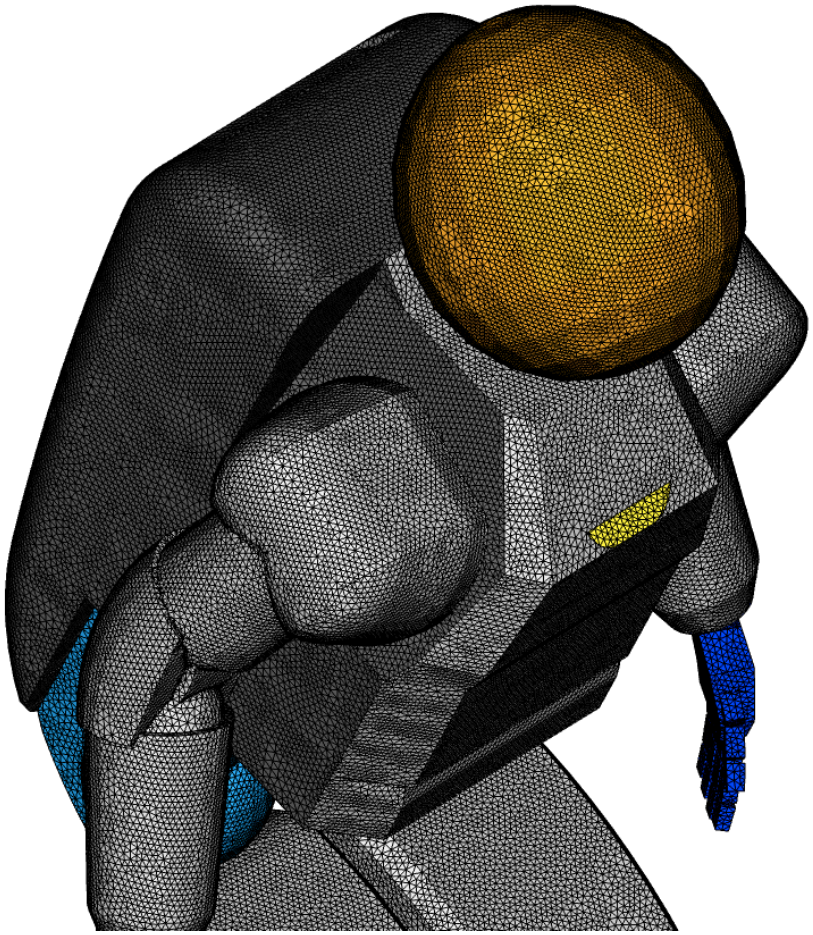
- So take your time and explore these folders. You can also take a look at the tutorial contained in the folder **$path_to_openfoamcourse/postprocessing/motorBike**

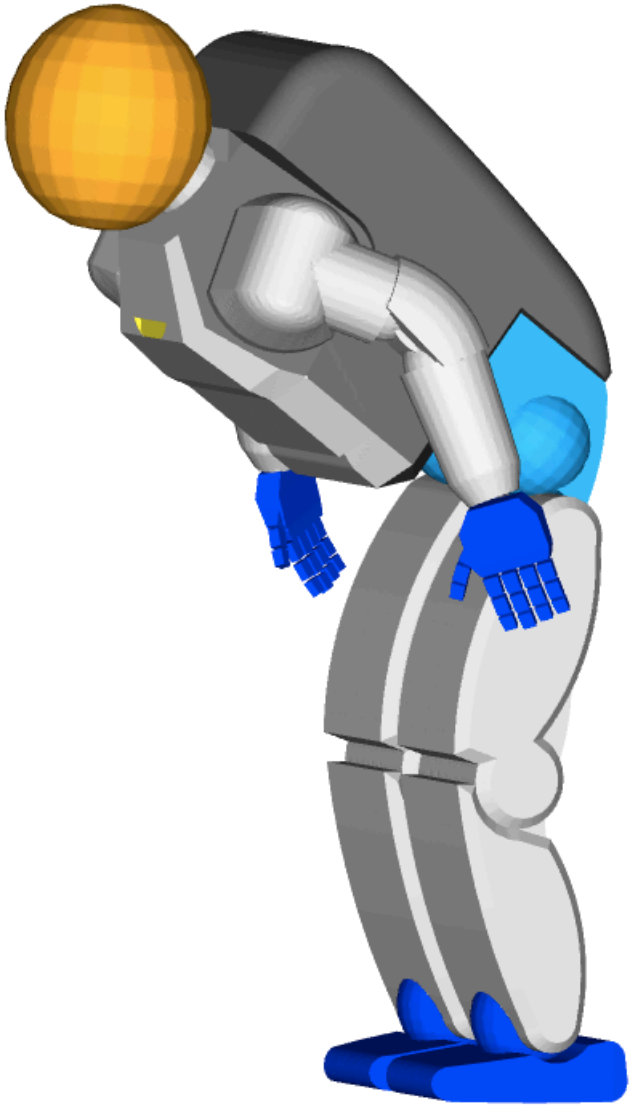
Mesh generation using open source tools

Additional tutorials

In the folder **$path_to_openfoamcourse/postprocessing** you will find many tutorials, try to go through each one to understand and get functional using the post processing tools.

Thank you for your attention

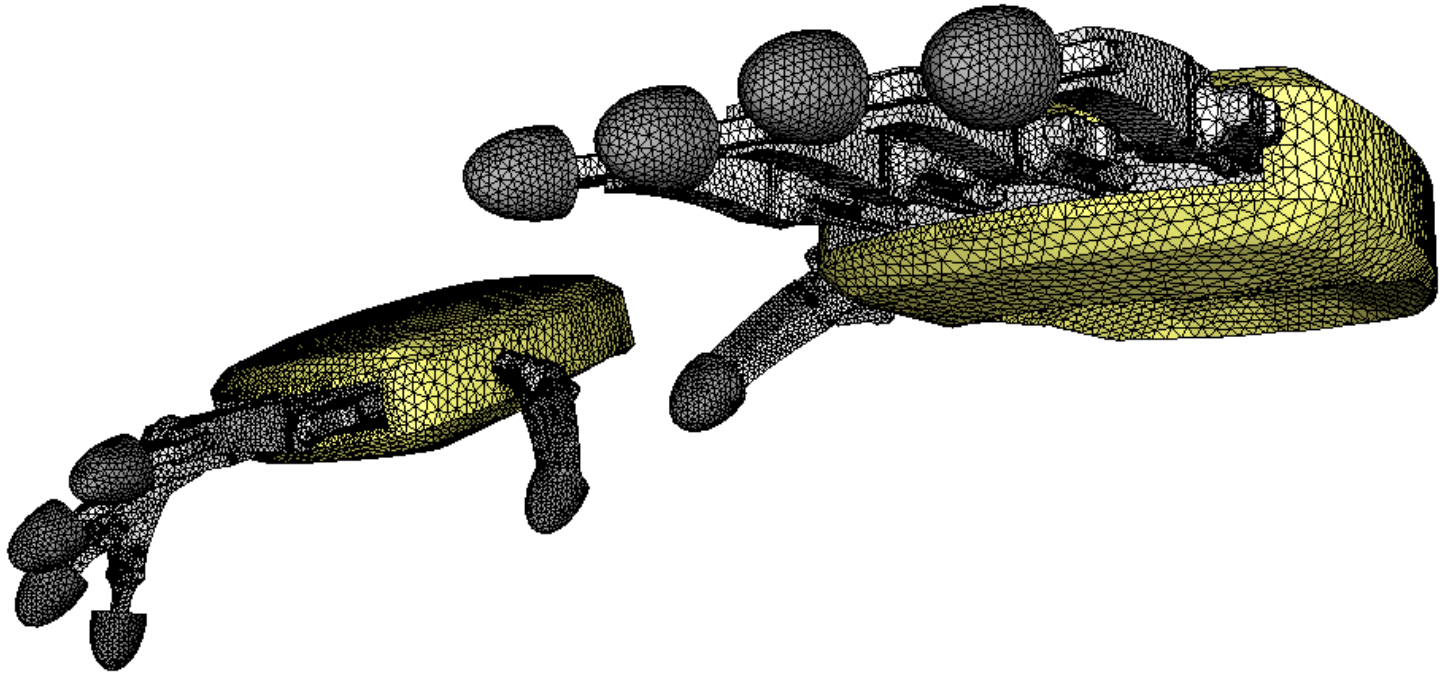




Today's lecture

1. ~~Data visualization with paraFoam~~
2. ~~Data visualization with VISIT~~
3. ~~Data manipulation and conversion~~
4. ~~The sample utility~~
5. ~~Probes~~
6. ~~functionObjects~~
7. ~~foamLog~~
8. ~~More post-processing utilities~~
9. **Hands-on session**

Hands-on session



In the course's directory **($path_to_openfoamcourse)** you will find many tutorials (which are different from those that come with the OpenFOAM® installation), let us try to go through each one to understand and get functional using OpenFOAM®.

If you have a case of your own, let me know and I will try to do my best to help you to setup your case. But remember, the physics is yours.