051176 - Computational Techniques for Thermochemical Propulsion
Master of Science in Aeronautical Engineering

Post-Processing in OpenFOAM

Prof. **Federico Piscaglia**
Dept. of Aerospace Science and Technology (DAER)
Politecnico di Milano - Italy
federico.piscaglia@polimi.it

Bibliography

https://cfd.direct/openfoam/user-guide/v6-postprocessing/

Post-processing

The outcome of an OpenFOAM® run is a folder containing a large number of data:

- The case setup: `controlDict`, `fvSchemes`, `fvSolution`;
- The FV grid(s): `points`, `faces`, `owner`, `neighbour`, `boundaries`, `cell/face/pointZones`
- a series of time folders, where flow fields (p, U, k, ε, ...) are stored.

A single OpenFOAM case can be as large as 50 GB

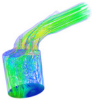
$\Downarrow$

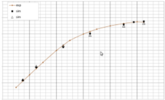
Need to extract only relevant information

$\Downarrow$

POST-PROCESSING

```
engine-1
|-- 0
|-- 500
|-- constant
`-- system
```





POST-PROCESSING – steps

1. **Data extraction**, using:
   - cutting planes
   - sampling lines
   - arbitrary probes
2. **Data reduction**,
   - domain integration or averaging
   - time- or ensemble- averaging
   - other statistics (e.g. RMS)
3. **Computation of derived quantities**, e.g.
   - vorticity
   - wall shear stress
   - drag coefficient
   - ...

---

4. **Visualization**, with different tools:
   - ParaView®, EnSight
   - `gnuplot`, `python` (matplotlib), Matlab, Excel
   - raw format (plain text)

In this lecture we will review only steps 1, 2 and 3

Introduction

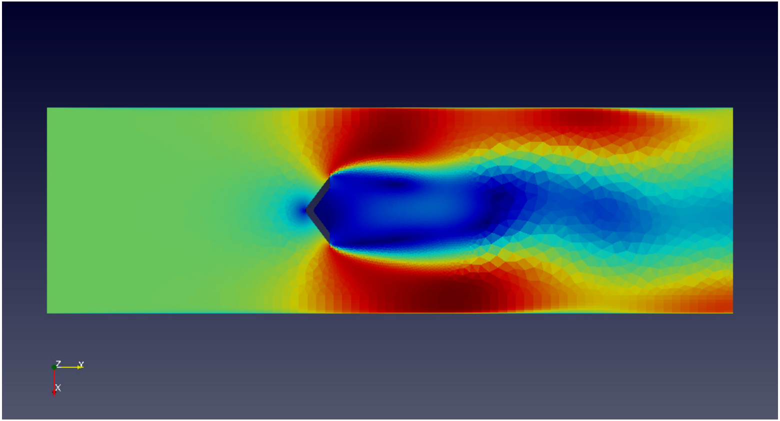
functionObjects

data format

Hands on

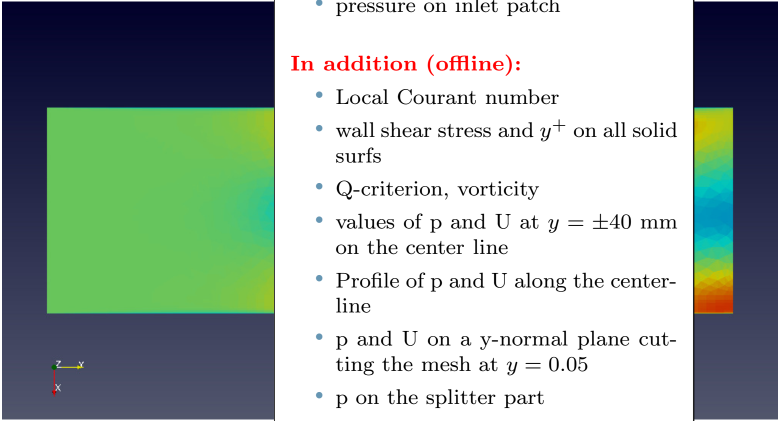
Postprocessing the results

This is the case to extract data from:



Postprocessing the results

This is the case to extract d



Online:

- residuals on p, U
- forces on splitter part
- pressure on inlet patch

In addition (offline):

- Local Courant number
- wall shear stress and y^+ on all solid surfs
- Q-criterion, vorticity
- values of p and U at $y = \pm 40$ mm on the center line
- Profile of p and U along the centerline
- p and U on a y-normal plane cutting the mesh at $y = 0.05$
- p on the splitter part
- iso-surfaces $p = 0$

Which tool should I use?

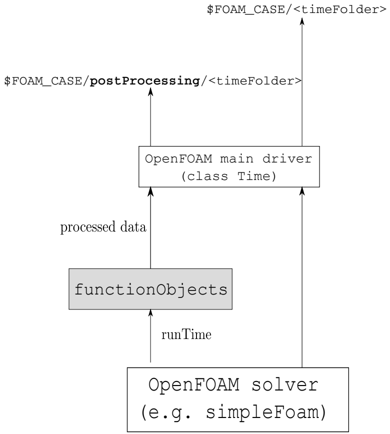
1) **Ad-hoc scripts/codes** (`python`, Matlab, etc.)
   - they can do nearly everything, but...
   - they have to be written from scratch
   - are they always needed? Check the code first!

2) **Visual (e.g. with a GUI) post-processing software**
   - easy to learn
   - automation is not easy, though feasible
   - parallel processing is not always available
   - co-processing is very difficult

3) **OpenFOAM® integrated functions and utilities**
   - already available for common CFD quantities
   - already working in parallel
   - automation is straightforward
   - co-processing is trivial
   - ...but they need external tools to visualize the results

In this lecture, we will review only the post-processing utilities available in OpenFOAM®

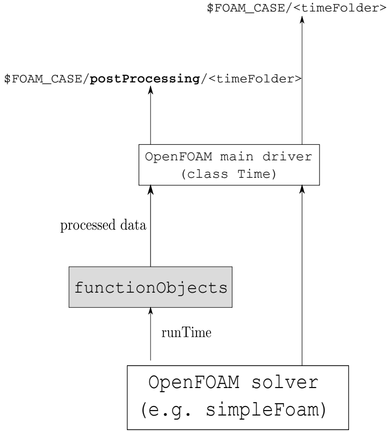
functionObjects in OpenFOAM®



`functionObjects` perform data processing while the simulation is running;

- needed variables/fields are read <u>`directly from the RAM`</u>, rather than from the disk;
- derived quantities and/or samples are generated run-time and stored on disk.

functionObjects in OpenFOAM®



Advantages:

- real-time computation
- reduced storage;
- no need for graphics facilities

Drawbacks

- computational overhead;
- might require frequent Input/Output

`functionObjects` in OpenFOAM®

`functionObjects`:

- are specified in the `controlDict` and executed every timestep (*).
- are run-time modifiable
- can run offline (at the end of the execution)
- can be temporarily disabled by the option `-noFunctionObjects`

```
functions
{
  probes1
  {
    type probes;

    enabled          true;
    writeControl   timeStep;
    writeInterval  1;

    probeLocations
    (
      (0.1778 0.0253 0.0)
    );

    fields
    (
      p
    );
  }
};
```

(*) Starting from OpenFOAM®-2.3.x, the user can specify an `evaluateControl`

`functionObjects` in OpenFOAM®

```
functions
(
  residuals            // arbitrary name
  {
    type residuals;    // type of functionObject
    enabled         true; // or false
    writeControl   timeStep; // see controlDict
    writeInterval  1;
    libs ("libutilityFunctionObjects.so");

    // ... type-dependent parameters ...
  }
);
```

All `functionObjects` require:

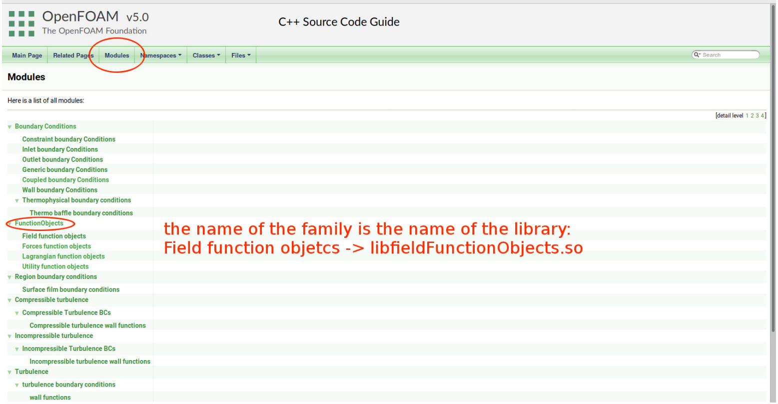
- an arbitrary name (e.g. 'probes1') – no spaces and no special characters
- an enable on/off switch (default: on)
- `writeControls`: outputTime, runTime, timeStep
- name of the library containing the function object to be executed.

All parameters can be modified run-time.

For a complete list, type: `postProcess -list`

How do I find all functionObjects?

- source guide `https://cpp.openfoam.org/v5/index.html`
- Modules → Function Objects → Family



Executing `functionObjects` off-line

- functionObjects can be executed **off-line**, i.e. on data saved on disk, when the simulation has completed
- Two alternatives:
  - Using the `postProcess` utility;
  - Using the solver name with the option "-postProcess":
- By default, `functionObjects` are read from the `controlDict` file;
- the user can specify an alternate dictionary or a single functionObject, thus overriding any `controlDict` entry

`postProcess` utility

Now the user can run execute post-processing functions with `postProcess`. The `-help` option provides a summary of its use.

```
postProcess -help
```

- `postProcess` is used if FO does not require the loading of any physical model (e.g. sampling, probing, simple derived quantities)
- Simple functions can be executed using the `-func` option; text on the command line generally needs to be quoted ("...") if it contains punctuation characters. Example:

```
postProcess -func "mag(U)"

postProcess -func "totalPressureIncompressible(p,U)"

postProcess -fields "(p U)" -func totalPressureIncompressible
```

Solver post-processing: `-postProcess` option

There is a situation where we need to post-process (as opposed to run-time process) and we need to have solver modelling available for the post-processing function needs. In this case we need to the `-postProcess` option. Help for this operation can be printed with the following command.

```
simpleFoam -postProcess -help
```

- `simpleFoam -postProcess` is used if the functionObject requires any physical model (e.g. turbulence, chemistry, thermodynamic properties. . .). We can monitor quantities like y^+, τ_w, forces, Q_w, etc. . .

```
simpleFoam -postProcess -func wallShearStress
```

Data conversion

It is also possible to export data to be visualized with other post-processing tools:

OpenFOAM application	source data	resulting data format
`foamDataToFluent`	OpenFOAM	Fluent
`foamToEnsightParts` `foamToEnsight`	OpenFOAM	EnSight
`foamToGMV`	OpenFOAM	GMV (General Mesh Viewer)
`smapToFoam`	STAR-CD	OpenFOAM
`foamToFieldview9`	OpenFOAM	FieldView UNS
`foamToVTK`	OpenFOAM	VTK
`foamToTecplot360`	OpenFOAM	Tecplot binary

foamToVTK utility

OpenFOAM data can be converted into ParaView native format with the command:

```
Usage: foamToVTK [-allPatches] [-ascii]  \
       [-excludePatches <wordReList>] \
       [-faceSet <name>] [-fields <wordList>] \
       [-nearCellValue] [-noFaceZones]  \
       [-noInternal] [-noLinks] [-noPointValues] [-noZero] \
       [-pointSet <name>] \
       [-poly] [-region <name>] [-surfaceFields]
```

Advantages

- More powerful, allowing for conversion of pointSets, faceSets, cellSets
- You can choose to convert only selected times / mesh regions / patches;
- Data reading by ParaView is faster
- Data are easier to handle

Drawbacks

- Conversion of the whole case can be very time-consuming
- For parallel runs, reconstruction is needed prior to convert
- Extra disk space is needed
- Cannot perform automatic-update of the results (like in paraFoam)

foamLog

- During execution, OpenFOAM writes values of residuals, number of iterations, etc. . . on the standard output
- It is possible to extract these information with the `foamLog` utility, provided the stdout has been written to a file.

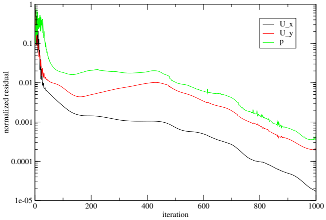
```
foamLog [-n] [-s] <log>
```

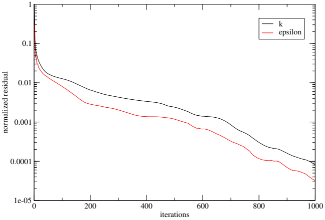
- By default, the files are presented in two-column format of time and the extracted values:

```
contCumulative_0  epsilonIters_0  kIters_0     Time_0
contGlobal_0      executionTime_0 p_0          Ux_0
contLocal_0       foamLog.awk     pFinalRes_0  UxFinalRes_0
epsilon_0         k_0             pIters_0     UxIters_0
epsilonFinalRes_0 kFinalRes_0     Separator_0  Uy_0
UyFinalRes_0      UyIters_0
```

Introduction
functionObjects
data format
Hands on

Fast residual extraction





- With long simulations, the `log` file can be large and extracting residuals can be very time-consuming;
- a quick-and-dirty solution is available for the following common variables:
  - Velocity and pressure → `foamGraphResUVWP log`
  - Turbulent kinetic energy and dissipation → `foamGraphResKE log`
- Result is a text file that can be plotted e.g. with xmgr:

```
xmgrace -log y residualUVWP.dat
```

Hands on

Quantities to be extracted:

- Online:
  - residuals on p, U
  - forces on splitter part
  - pressure on inlet patch
- **In addition (offline):**
  - Local Courant number
  - wall shear stress and y^+ on all solid surfs
  - Q-criterion, vorticity
  - values of p and U at $y = \pm 40$ mm on the center line
  - Profile of p and U along the centerline
  - p and U on a y-normal plane cutting the mesh at $y = 0.05$
  - p on the splitter part
  - iso-surfaces $p = 0$

Hands on: solution...

Calculation of the residuals

```
residuals
{
    type residuals;
    enabled yes;
    fields (p U k epsilon);
    libs ("libutilityFunctionObjects.so");
}
```

Introduction

functionObjects

data format

Hands on

Average p on inlet

```
pInlet
{
    type surfaceFieldValue;
    enabled yes;
    log true;
    writeControl timeStep;
    writeInterval   1;
    regionType patch;
    name inlet;
    operation average;
    fields (p);
    writeFields false;
    libs ("libfieldFunctionObjects.so");
}
```

Forces on the splitter

```
force
{
    type forces;
    enabled yes;
    libs ("libforces.so");
    log yes;
    writeControl timeStep;
    writeInterval 1;

    patches (splitter back fuel_inlet);
    rho rhoInf;
    rhoInf 1.18;
    liftDir (1 0 0);
    dragDir         (0 1 0);
    CofR    (0 0 0);
}
```

Derived quantities

- Local Co: `postProcess -func CourantNo -noZero`
- Q-criterion: `postProcess -func Q`
- vorticity: `postProcess -func vorticity`
- Wall shear stress: `pimpleFoam -postProcess -func wallShearStress`
- y^+: `pimpleFoam -postProcess -func yPlus`

Probes

Probe p and U at fixed location:

```
probe
{
    type    probes;
    enabled yes;
    writeControl timeStep;
    writeInterval 1;

    fields (p U);
    probeLocations
    (
       (0 40e-3 5e-4)
       (0 -40e-3 5e-4)
    );
}
```

Sampled set

Extract p and U along a line:

```
axialLine
{
    type    sets;
    enabled yes;
    writeControl timeStep;
    writeInterval 1;

    fields (p U);
    interpolationScheme cell;
    setFormat raw;

    sets
    (
        line1
        {
            type    uniform;
            axis y;
            nPoints 200;
            start (0 -0.3 5e-4);
            end (0 0.5 5e-4);
        }
    );
}
```

Sampled surfaces


```
surf
{
    type    surfaces;
    enabled yes;
    writeControl timeStep;
    writeInterval 1;
    fields (p U);
    interpolationScheme cellPoint;
    surfaceFormat vtk;

    surfaces
    (
        sec1
        {
            type            cuttingPlane;
            planeType       pointAndNormal;
            pointAndNormalDict
            {
                point   (0 0.05 5e-4);
                normal  (0 1 0);
            }
            interpolate     true;
        }
    );
}
```

Introduction

functionObjects

data format

Hands on

Sampled surfaces

```
  // ...
surfaces
(
    splitter
    {
        type            patch;
        patches ("splitter" "back" "fuel_inlet");
        interpolate     true;
    }
    pRef
    {
        type     isoSurface;
        isoField p;
        isoValue 0;
        interpolate true;
    }
);
}
```

Thank you for your attention!

contact: federico.piscaglia@polimi.it