FOAM@Iberia
2023
5th edition

Leveraging Function Objects

A Comprehensive Guide to OpenFOAM's functionObjects

Gabriel Magalhães / Sacha Mould

Computational Rheology @IPC

University of Minho
School of Engineering

INSTITUTE FOR POLYMERS AND COMPOSITES

Disclaimer

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

Contents

1

Introduction to functionObjects

Definition and purpose of functionObjects

Definition

- Function objects are **OpenFOAM utilities** to ease workflow configurations and enhance workflows by producing **additional user-requested data**;
- It can be used both during **runtime and postprocessing calculations**;
- Typically the results are showing in the form of **additional logging** to the screen, or **generating text, image and field files**.

[1]https://doc.openfoam.com/2306/tools/post-processing/function-objects/

Advantages

- **Function objects** eliminate the need to store all runtime generated data, hence **saving considerable resources**;
- Function objects are readily **applied to batch-driven processes,** improving reliability by standardising the sequence of operations and **reducing the amount of manual interaction**;
- The **output** of most function objects are **stored on the mesh database** to enable retrieval and chaining to other function objects and applications.

[1]https://doc.openfoam.com/2306/tools/post-processing/function-objects/

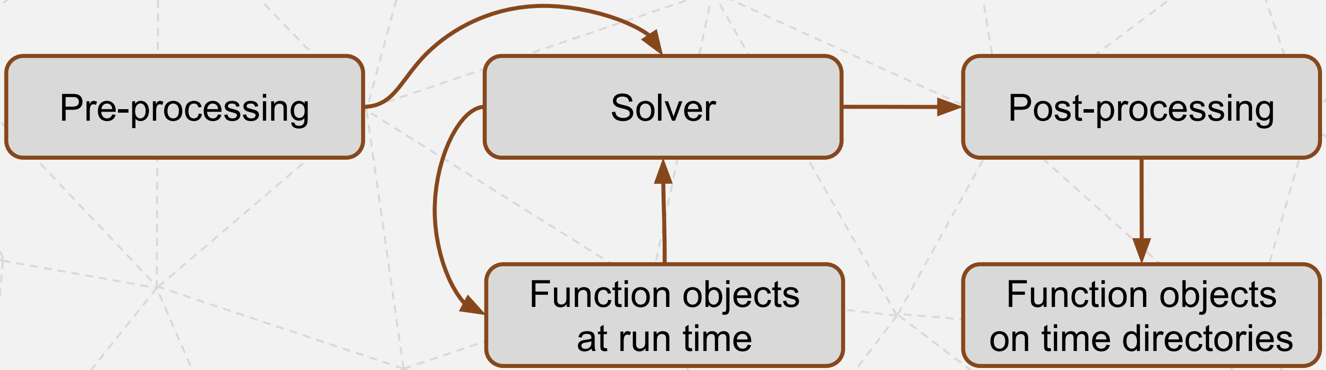
1

Introduction to functionObjects

How functionObjects fit into the OpenFOAM workflow

The function objects can be used in the OpenFOAM workflow using **two methods**:

- At **run-time**: via the functions sub-dictionary in the **controlDict** file;
- As **post-processing functions** on **time directories**.



[1]https://doc.openfoam.com/2306/tools/post-processing/function-objects/

1

Introduction to functionObjects

Types of functionObjects

CRheo@IPC

Categories

- The function objects are separated by **categories** according with the **type of operation** that it performs:
  - Field
  - Graphics
  - Lagrangian
  - Random Processes
  - Utilities
  - Forces
  - Initialisation
  - Phase Systems
  - Solvers

1

Introduction to functionObjects

Documentation

CRheo@IPC

Documentation

- **User Guide** of OpenFOAM `v23.06` (here);
- **Source codes** locally or online;
- **Annotated dictionaries** locally or online;
- **Examples**:
  - `Vorticity`;
  - `Continuity error`.

2

Commonly used functionObjects

probes functionObject

CRheo@IPC

probes functionObject

- **Definition and application:** `Probes` allows for the monitoring of specific points or regions in the domain during simulation.
- **Importance in data extraction:** provides critical data for validation, analysis, and further processing.
- **Example Scenario:** Simulating a flow with particles in a pipe it is possible to evaluate the velocity profile in a section using probes and compare the data with experimental results.

2

Commonly used functionObjects

turbulenceField functionObject

CRheo@*IPC*

turbulenceFields functionObject

- **Definition and application:** The `turbulenceFields` function object computes various turbulence-related quantities that are not typically output during calculations.
- **Importance in data extraction:** Some of these properties play a crucial role in post-processing, helping to understand better critical points of the flow.
- **Example Scenario:** CFD analysis of a car's aerodynamics: analysis of the impact of the chosen turbulence model in the effective viscosity or analysis of turbulence intensity along the wake.

3

Advanced functionObjects

coded functionObjects

coded functionObjects

- When we are working on a **specific analysis** the existing function objects may not meet some demands in terms of post-processing or run-time analysis;
- The coded function objects can be seen as **on demand function objects** or **dynamic function objects**;
- Coded function objects can be valuable in many scenarios but it **requires** some domain of **C++ language and OpenFOAM architecture**.

coded functionObjects

Some key characteristics of `coded functionObjects`:

- **On-the-Fly Configuration:** `coded functionObjects` can be configured and initiated at runtime, enabling users to adapt their data extraction and processing needs during the simulation.
- **Adaptive Data Extraction:** They allow for the monitoring and analysis of specific data points or regions that may change over time or based on simulation conditions.
- **Flexible and Responsive:** `coded functionObjects` are highly flexible and responsive to changing simulation conditions, making them useful in scenarios where the areas of interest evolve or where specific data points need to be tracked dynamically.

coded functionObjects

Some key characteristics of `coded functionObjects`:

- **Range of Applications:** They are particularly valuable in simulations with evolving geometries, multi-phase flows, or complex boundary conditions where the regions of interest change over time.
- **Dynamic Data Processing:** `coded functionObjects` can perform computations and data processing operations that depend on evolving simulation conditions.
- **Efficient Resource Utilization:** They allow for more efficient use of computational resources by enabling the selective activation of `functionObjects` only when needed.

coded functionObjects

- The coded function object provides a general interface to enable **dynamic code compilation**;
- We have good examples of coded function objects working to:
  - **Pre process** using CLI (link);
  - **Change property** during the execution (link);
  - Generate and write **additional fields** (link);
  - **Compare** a field with an **analytical solution** (link);
  - **Post process**;

coded functionObjects

- The entries are:
  - ***codeInclude***: include files;
  - ***codeOptions***: include paths; inserted into `EXE_INC` in `Make/options`;
  - ***codeLibs***: link line; inserted into `LIB_LIBS` in `Make/options`;
  - ***codeData***: `C++`; local member data (`null` constructed);
  - ***localCode***: `C++`; local static functions;
  - ***codeRead***: `C++`; upon `functionObject::read()`;
  - ***codeExecute***: `C++`; upon `functionObject::execute()`;
  - ***codeWrite***: `C++`; upon `functionObject::write()`;
  - ***codeEnd***: `C++`; upon `functionObject::end()`;
  - ***codeContext***: additional dictionary context for the code.

[4] https://develop.openfoam.com/Development/openfoam/-/blob/master/src/functionObjects/utilities/codedFunctionObject/codedFunctionObject.H?ref_type=heads

4

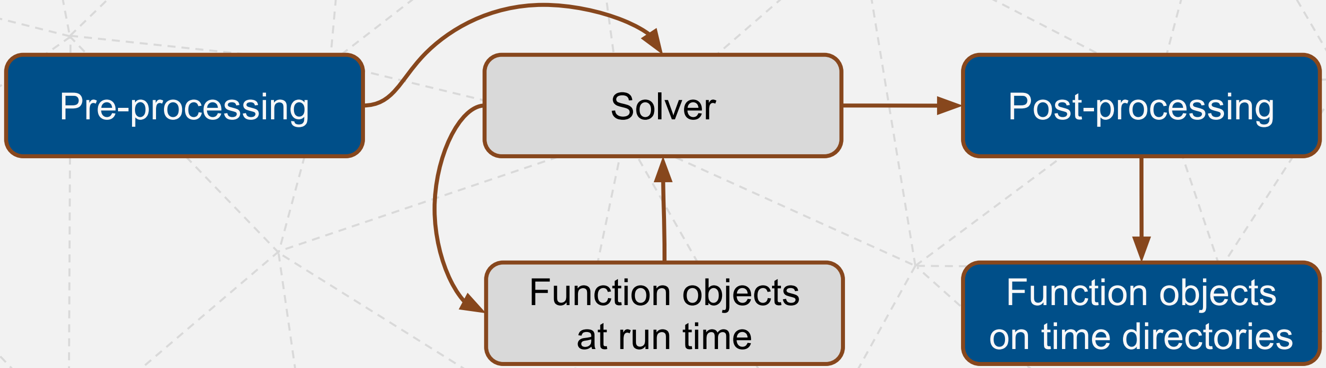
Examples of use

CLI (Command Line Interface)

Examples of use

CLI

- The Command Line Interface (CLI) is the first and **simplest way** to use the existing function objects;
- The CLI is used mainly in **pre and post processing** operations **before or after to run** the simulation;



- It consists in use **commands on the terminal** to apply pre or post processing functions. Commonly used for **fast analysis**;
- postProcess utility works in the way of **re-reading the result folders**.

Examples of use

- The built-in post-processing **utilities** can be **listed** by:

`$ postProcess -list`

- The **dictionary examples** are located in

`$FOAM_ETC/caseDicts/postProcessing`

- The **source codes** of the `functionObjects` are located in

`$FOAM_SRC/functionObjects`

Examples of use

CLI

- **Get a copy** of the function object in the case folder

```
$ foamGetDict probes
```

- On the **header** we have a **description** of the function object:

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Version:  v2306
    \\  /    A nd           | Website:  www.openfoam.com
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Writes out values of fields from cells nearest to specified locations.

\*---------------------------------------------------------------------------*/

#includeEtc "caseDicts/postProcessing/probes/probes.cfg"

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

// ************************************************************************* //
```

Examples of use

CLI

➢ **Options** for postProcess by CLI

```
$ postProcess -help-full
```

➢ **Calling** a functionObject by CLI:

```
$ postProcess -func "fieldMinMax(p)" [OPTIONS]
```

➢ **Calling multiple** functionObjects by CLI:

```
$ postProcess -funcs "(fieldMinMax(p) Q)" [OPTIONS]
```

Example (go to a case)

- Inside an example case we will get the probe dictionary

```
$ foamGetDict probes
```

- We need to check the configurations on the included file:

```
$ vim $FOAM_ETC/caseDicts/postProcessing/probes/probes.cfg
```

- Edit the file in system folder:

```
$ vim system/probes
```

- Execute the inline call:

```
$ postProcess -func probes
```

- The folder `postProcessing/probes` is created and the data is written.

4

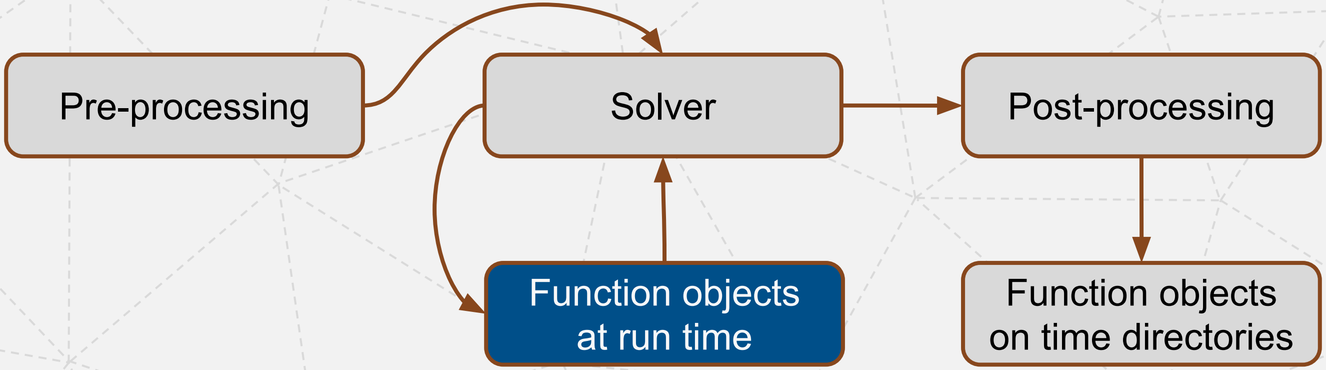
Examples of use

Function objects at run time

CRheo@IPC

Run time data process

- The other way to use the function objects can be named **run time data process**;
- The defined function objects will be executed in run time. It means that we can **control the functions by time step**, for example.



Run time data process

When applied at **run-time**, the objects are **controlled** according to the optional **two time-based** entries:

- **`executeControl`:** when the object is updated (for updating calculations or for management tasks),
- **`writeControl`:** when the object output is written (for writing the calculated data to disk);

If **neither entries** are present the **object will execute and write every time step** which can create much more data than intended!

[1]https://doc.openfoam.com/2306/tools/post-processing/function-objects/

Examples of use

Run time data process

Option	Description
`none`	Trigger is disabled
`timeStep`	Trigger every 'Interval' time-steps, e.g. every x time steps
`writeTime`	Trigger every 'Interval' output times, i.e. alongside standard field output
`runTime`	Trigger every 'Interval' run time period, e.g. every x seconds of calculation time
`adjustableRunTime`	Currently identical to "`runTime`"
`clockTime`	Trigger every 'Interval' clock time period
`cpuTime`	Trigger every 'Interval' CPU time period
`onEnd`	Trigger on end of simulation run

[1]https://doc.openfoam.com/2306/tools/post-processing/function-objects/

Run time data process

- To perform the **run time data processing** we need to put the **functions into** `system/controlDict` on the sub-dictionary `functions;`

- We can **write** the functions **directly or include other files** on `system` folder using:

```
#includeFunc [FileName]
```

or

```
#include [FileName]
```

Run time data process

- To perform the **run time data processing** we need to put the **functions into** `system/controlDict` on the dictionary `functions;`

- We can **write** the functions **directly or include other files** on `system` folder using:

```
#includeFunc [FileName]
```

or

```
#include [FileName]
```

What is the difference?

CRheo@IPC

Examples of use

Run time data process

- The use of `#includeFunc [FileName];`
- Example in tutorial case [backwardFaceStep2D]() in `simpleFoam`;

```
writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    #includeFunc "stressComponents"
    #includeFunc "pressureCoefficient"
    #includeFunc "sample"
    #includeFunc "sampleCp"
    #includeFunc "writeCellCentres"
    #includeFunc "wallShearStress"
}


// ************************************************************************* //
```

```
type            patchProbes;
libs            (sampling);
writeControl    writeTime;

patch           lowerWall;
probeLocations  ();
fields          (cp);
```

CRheo@IPC

Examples of use

Run time data process

- The use of `#include [FileName];`
- Example in tutorial case

motorBike in `interFoam`;

```
timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.5;

maxAlphaCo      0.5;

maxDeltaT       1;

functions
{
    #include "minMax"
}


// ************************************************
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      minMax;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

minMax
{
    // Type of functionObject
    type            fieldMinMax;

    // Where to load it from (if not already in solver)
    libs            (fieldFunctionObjects);

    // Log to output (default: false)
    log             true;

    // Fields to be monitored - runTime modifiable
    fields
    (
        U
        p
    );
}

// ************************************************************************* //
```

Examples of use

Run time data process

- Composed use;
- Example in tutorial case

eletrostaticDeposition in `interFoam`;

```
adjustTimeStep  yes;

maxCo           10;

maxAlphaCo      20;

maxDeltaT       0.05;

functions
{
    #include    "FOelectricPotential"

    fieldMinMax1
    {
        type            fieldMinMax;
        libs            (fieldFunctionObjects);
        fields          ("electricPotential:V");
    }
}


// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      FOelectricPotential;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
electricPotential
{
    // Mandatory entries
    type            electricPotential;
    libs            (solverFunctionObjects);
    phases
    {
        alpha.air
        {
            epsilonr      1.12940906737;
            sigma         1e-10;
        }
        alpha.water
        {
            epsilonr      3.38822720212;
            sigma         0.14;
        }
    }

    // Optional entries
    nCorr               1;
    writeDerivedFields  true;

    // Inherited entries
    region          region0;
    enabled         true;
    log             true;
    timeStart       0;
    timeEnd         100;
    executeControl  timeStep;
    executeInterval 1;
    writeControl    writeTime;
    writeInterval   -1;
}
```

CRheo@IPC

5

Custom functionObjects

coded: Implementing and using

CRheo@IPC

- Generating a **new field** for the **non-dimensional velocity** U*

$$U^* = \frac{||U||}{||U_{inlet}||}$$

- We will use the airFoil2D tutorial as the base, a 2D flow around an airfoil;
- Just to **check** in the implementing code, the inlet velocity is

$$U_{inlet} = (25.75, 3.62, 0)\, m/s$$

which leads to a **magnitude** of

$$||U_{inlet}|| \approx 26\, m/s$$

Custom functionObjects

coded - Example

```
    turbulenceProperties:epsilon

Using dynamicCode for functionObject Ustar at line 100 in "/home/gmarcos/Documents/foamIberia/foCourse/airFoi
l2D/system/controlDict.functions.Ustar"
Could not load "/home/gmarcos/Documents/foamIberia/foCourse/airFoil2D/dynamicCode/platforms/linux64GccDPInt32
Opt/lib/libUstar_7c3e079f2355efb344ddaafc66cd7456da0815b6.so"
/home/gmarcos/Documents/foamIberia/foCourse/airFoil2D/dynamicCode/platforms/linux64GccDPInt32Opt/lib/libUstar
_7c3e079f2355efb344ddaafc66cd7456da0815b6.so: cannot open shared object file: No such file or directory
Creating new library in "dynamicCode/Ustar/platforms/linux64GccDPInt32Opt/lib/libUstar_7c3e079f2355efb344ddaa
fc66cd7456da0815b6.so"
Invoking wmake libso /home/gmarcos/Documents/foamIberia/foCourse/airFoil2D/dynamicCode/Ustar
wmake libso /home/gmarcos/Documents/foamIberia/foCourse/airFoil2D/dynamicCode/Ustar
    ln: ./lnInclude
    dep: functionObjectTemplate.C
    Ctoo: functionObjectTemplate.C
    link: /home/gmarcos/Documents/foamIberia/foCourse/airFoil2D/dynamicCode/Ustar/../platforms/linux64GccDPIn
t32Opt/lib/libUstar_7c3e079f2355efb344ddaafc66cd7456da0815b6.so
Time = 1

smoothSolver:  Solving for Ux, Initial residual = 1, Final residual = 0.0010401, No Iterations 2
smoothSolver:  Solving for Uy, Initial residual = 1, Final residual = 0.00104017, No Iterations 2
GAMG:  Solving for p, Initial residual = 1, Final residual = 0.0924376, No Iterations 6
time step continuity errors : sum local = 0.000538058, global = 4.06251e-05, cumulative = 4.06251e-05
smoothSolver:  Solving for nuTilda, Initial residual = 1, Final residual = 0.0493018, No Iterations 4
ExecutionTime = 0.08 s  ClockTime = 5 s
```

Compilation of the coded function object **at the begin**. **If it doesn't compile** successfully, **the solver will not be executed**.

CRheo@*IPC*

Custom functionObjects

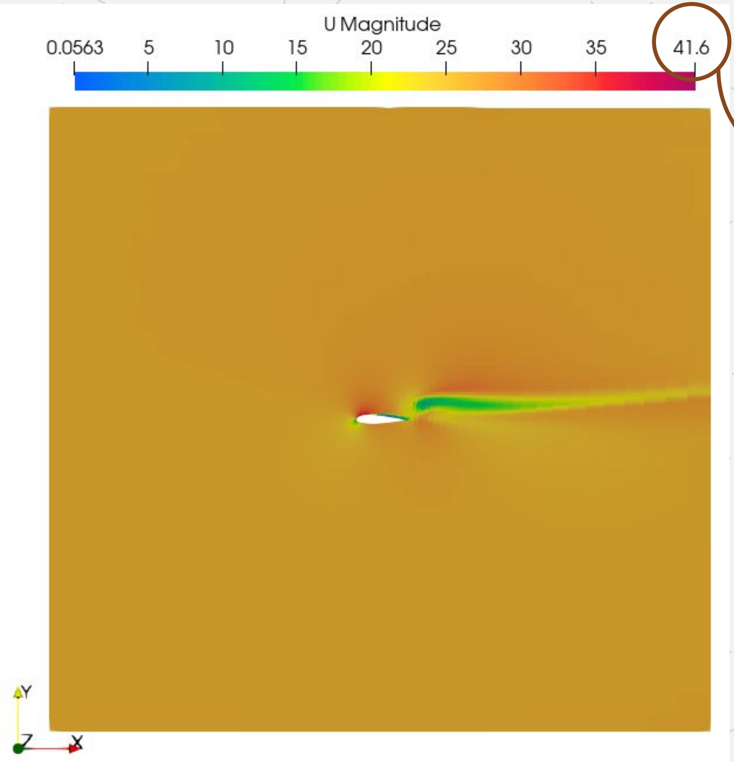
coded - Example

```
Ustar
{
    name    Ustar;
    type    coded;
    libs    (utilityFunctionObjects);

    executeControl  writeTime;
    writeControl    writeTime;

    codeExecute
    #{
        // Name of the patch with inlet velocity
        word inletPatch = "inlet";
        // Lookup U
        Info<< "\n\tLooking up field U" << endl;
        const auto& U = mesh().lookupObject<volVectorField>("U");

        Info<< "\tReading inlet velocity uInlet" << endl;

        // Get the label of the patch within inlet velocity
        label inletI = mesh().boundaryMesh().findPatchID(inletPatch);
        // Get velocity in patch inletI (just in the processor)
        const auto& UInlet = U.boundaryField()[inletI];
        // Get the magnitude of inlet velocity as it is uniform
        scalar magUInlet = 0.0;
        if (fvp.size())
        {
            magUInlet = mag(fvp[0]);
        }
        // Get the maximum value through the processors
        // OBS: this operation is extremely importante because in some cases
        // we don't have cells for inlet patch at processor 0
        reduce(magUInlet, maxOp<scalar>());

        // Creates a dimensional scalar to make possible the operation with U
        dimensionedScalar uIn("uIn", dimVelocity, magUInlet);

        Info<< "\tMagnitude of U at inlet = " << uIn.value() << " m/s" << endl;

        // Create a volScalar field for the non-dimensional velocity
        // U* = mag(U)/mag(Uin)
        volScalarField UStar
        (
            IOobject
            (
                "UStar",
                mesh().time().timeName(),
                mesh(),
                IOobject::NO_READ,
                IOobject::AUTO_WRITE
            ),
            mag(U)/uIn
        );

        // Writing of UStar
        Info<< "\n\tfunctionObjects::coded UStar writing field: UStar\n" << endl
;       UStar.write();
    #};
}
```

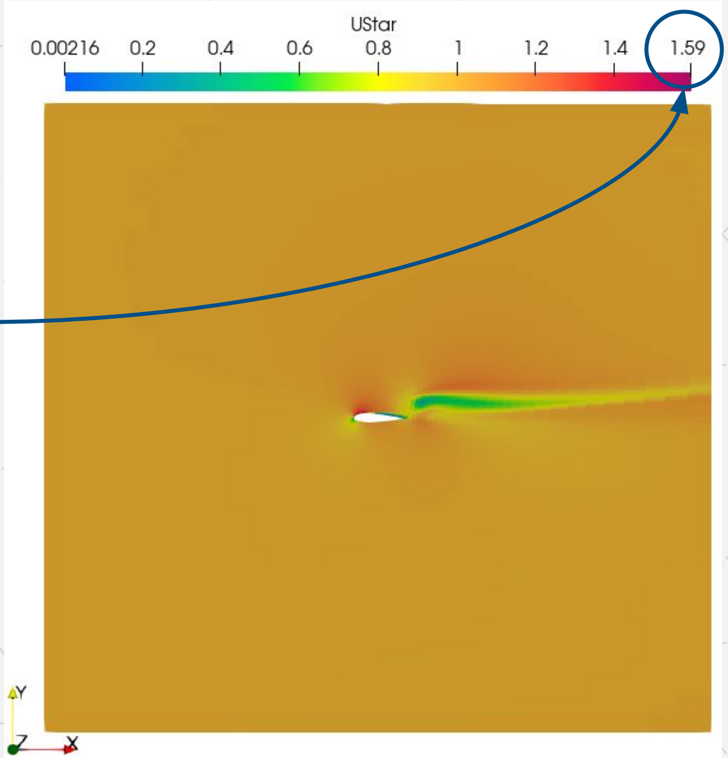
CRheo@IPC

Custom functionObjects

coded - Example



$$U^* = \frac{||U||}{||U_{inlet}||}$$



$$||U_{inlet}|| \approx 26\, m/s$$

Acknowledgment

This work has been part of the exaFOAM Project (https://www.exafoam.eu), which has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 956416. The JU receives support from the European Unions Horizon 2020 research and innovation programme and France, Germany, Italy, Croatia, Spain, Greece, and Portugal. The authors acknowledge the support of the Portuguese National Funds through FCT - Portuguese Foundation for Science and Technology, in the framework of projects UID/CTM/50025/2019 and UIDB/04436/2020.

END

gabrielmarcosmag@gmail.com (Gabriel)
d4224@dep.uminho.pt (Sacha)

CRheo@*IPC*