

Altair ultraFluidX 2022

User Guide

Updated: 03/22/2022

Contents

Intellectual Property Rights Notice iii
Technical Support vii

1 User Guide 9

1.1 Overview 10
1.2 Use ultraFluidX 11
1.2.1 Additional Command Line Options 11
1.2.2 Workflow and Output 12
1.2.3 Multi-Node Usage 13
1.3 Simulation Parameters 15
1.3.1 <simulation> 15
1.3.2 <meshing> 20
1.3.3 <boundary_conditions> 23
1.3.4 <sources> 25
1.3.5 <output> 28
1.4 Expert Parameters 39
1.4.1 <simulation> 39
1.4.2 <meshing> 42

Index 44

Intellectual Property Rights Notice

Copyright © 1986-2022 Altair Engineering Inc. All Rights Reserved.

This Intellectual Property Rights Notice is exemplary, and therefore not exhaustive, of intellectual property rights held by Altair Engineering Inc. or its affiliates. Software, other products, and materials of Altair Engineering Inc. or its affiliates are protected under laws of the United States and laws of other jurisdictions. In addition to intellectual property rights indicated herein, such software, other products, and materials of Altair Engineering Inc. or its affiliates may be further protected by patents, additional copyrights, additional trademarks, trade secrets, and additional other intellectual property rights. For avoidance of doubt, copyright notice does not imply publication. Copyrights in the below are held by Altair Engineering Inc. except where otherwise explicitly stated. Additionally, all non-Altair marks are the property of their respective owners.

This Intellectual Property Rights Notice does not give you any right to any product, such as software, or underlying intellectual property rights of Altair Engineering Inc. or its affiliates. Usage, for example, of software of Altair Engineering Inc. or its affiliates is governed by and dependent on a valid license agreement.

Altair Simulation Products

Altair® AcuSolve® ©1997-2022

Altair Activate® ©1989-2022

Altair® BatteryDesigner™ ©2019-2022

Altair Compose® ©2007-2022

Altair® ConnectMe™ ©2014-2022

Altair® EDEM™ ©2005-2022 Altair Engineering Limited, ©2019-2022 Altair Engineering Inc.

Altair® ElectroFlo™ ©1992-2022

Altair Embed® ©1989-2022

Altair Embed® SE ©1989-2022

Altair Embed®/Digital Power Designer ©2012-2022

Altair Embed® Viewer ©1996-2022

Altair® ESAComp® ©1992-2022

Altair® Feko® ©1999-2022 Altair Development S.A. (Pty) Ltd., ©1999-2022 Altair Engineering Inc.

Altair® Flow Simulator™ ©2016-2022

Altair® Flux® ©1983-2022

Altair® FluxMotor® ©2017-2022

Altair® HyperCrash® ©2001-2022

Altair® HyperGraph® ©1995-2022

Altair® HyperLife® ©1990-2022

Altair® HyperMesh® ©1990-2022

Altair® HyperStudy® ©1999-2022

Altair® HyperView® ©1999-2022

Altair® HyperWorks® ©1990-2022

Altair® HyperXtrude® ©1999-2022

Altair® Inspire™ ©2009-2022

Altair® Inspire™ Cast ©2011-2022

Altair® Inspire™ Extrude Metal ©1996-2022

Altair® Inspire™ Extrude Polymer ©1996-2022

Altair® Inspire™ Form ©1998-2022

Altair® Inspire™ Friction Stir Welding ©1996-2022

Altair® Inspire™ Mold ©2009-2022

Altair® Inspire™ PolyFoam ©2009-2022

Altair® Inspire™ Play ©2009-2022

Altair® Inspire™ Print3D ©2022

Altair® Inspire™ Render ©1993-2016 Solid Iris Technologies Software Development One PLLC, ©2016-2022 Altair Engineering Inc

Altair® Inspire™ Resin Transfer Molding ©1990-2022

Altair® Inspire™ Studio ©1993-2022

Altair® Material Data Center™ ©2019-2022

Altair® MotionSolve® ©2002-2022

Altair® MotionView® ©1993-2022

Altair® Multiscale Designer® ©2011-2022

Altair® nanoFluidX® ©2013-2022 FluiDyna GmbH, ©2018-2022 Altair Engineering Inc.

Altair® OptiStruct® ©1996-2022

Altair® PollEx™ ©2003-2022

Altair® Pulse™ ©2020-2022

Altair® Radioss® ©1986-2022

Altair® SEAM® ©1985-2019 Cambridge Collaborative, Inc., ©2019-2022 Altair Engineering Inc.

Altair® SimLab® ©2004-2022

Altair® SimSolid® ©2015-2022

Altair® ultraFluidX® ©2010-2022 FluiDyna GmbH, ©2018-2022 Altair Engineering Inc.

Altair® Virtual Wind Tunnel™ ©2012-2022

Altair® WinProp™ ©2000-2022

Altair® WRAP™ ©1998-2022 Altair Engineering AB

Altair® S-FRAME® ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-STEEL™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-PAD™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-CONCRETE™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-LINE™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-TIMBER™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-FOUNDATION™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair® S-CALC™ ©1995-2022 Altair Engineering Canada, Ltd., ©2021-2022 Altair Engineering Inc.

Altair Packaged Solution Offerings (PSOs)

Altair® Automated Reporting Director™ ©2008-2022

Altair® e-Motor Director™ ©2019-2022

Altair® Geomechanics Director™ ©2011-2022

Altair® Impact Simulation Director™ ©2010-2022

Altair® Model Mesher Director™ ©2010-2022

Altair® NVH Director™ ©2010-2022

Altair® Squeak and Rattle Director™ ©2012-2022

Altair® Virtual Gauge Director™ ©2012-2022

Altair® Weld Certification Director™ ©2014-2022

Altair® Multi-Disciplinary Optimization Director™ ©2012-2022

Altair HPC & Cloud Products

Altair® PBS Professional® ©1994-2022

Altair® Control™ ©2008-2022

Altair ®Access™ ©2008-2022

Altair® Accelerator™ ©1995-2022

Altair® Accelerator™ Plus©1995-2022

Altair® FlowTracer™ ©1995-2022

Altair® Allocator™ ©1995-2022

Altair® Monitor™ ©1995-2022

Altair® Hero™ ©1995-2022

Altair® Software Asset Optimization (SAO) ©2007-2022

Altair Mistral™ ©2022

Altair Drive ©2021-2022

Altair® Grid Engine® ©2001, 2011-2022

Altair® DesignAI™ ©2022

Altair Breeze™ ©2022

Altair Data Analytics Products

Altair® Knowledge Studio® ©1994-2022 Altair Engineering Canada, Ltd., ©2018-2022 Altair Engineering Inc.

Altair® Knowledge Studio® for Apache Spark ©1994-2022 Altair Engineering Canada, Ltd., ©2018-2022 Altair Engineering Inc.

Altair® Knowledge Seeker™ ©1994-2022 Altair Engineering Canada, Ltd., ©2018-2022 Altair Engineering Inc.

Altair® Knowledge Hub™ ©2017-2022 Datawatch Corporation, ©2018-2022 Altair Engineering Inc.

Altair® Monarch® ©1996-2022 Datawatch Corporation, ©2018-2022 Altair Engineering Inc.

Altair® Panopticon™ ©2004-2022 Datawatch Corporation, ©2018-2022 Altair Engineering Inc.

Altair® SmartWorks™ ©2021-2022

Altair SmartCore™ ©2011-2022

Altair SmartEdge™ ©2011-2022

Altair SmartSight™ ©2011-2022

Altair One™ ©1994-2022

2022

January 10, 2022

Technical Support

Altair provides comprehensive software support via web FAQs, tutorials, training classes, telephone, and e-mail.

Altair One Customer Portal

Altair One (https://altairone.com/) is Altair's customer portal giving you access to product downloads, a Knowledge Base, and customer support. We recommend that all users create an Altair One account and use it as their primary portal for everything Altair.

When your Altair One account is set up, you can access the Altair support page via this link: www.altair.com/customer-support/

Altair Community

Participate in an online community where you can share insights, collaborate with colleagues and peers, and find more ways to take full advantage of Altair's products.

Visit the Altair Community (https://community.altair.com/community) where you can access online discussions, a knowledge base of product information, and an online form to contact Support. These valuable resources help you discover, learn and grow, all while having the opportunity to network with fellow explorers like yourself.

Altair Training Classes

Altair's in-person, online, and self-paced trainings provide hands-on introduction to our products, focusing on overall functionality. Trainings are conducted at our corporate and regional offices or at your facility.

For more information visit: https://learn.altair.com/

If you are interested in training at your facility, contact your account manager for more details. If you do not know who your account manager is, contact your local support office and they will connect you with your account manager.

Telephone and E-mail

If you are unable to contact Altair support via the customer portal, you may reach out to technical support via phone or e-mail. Use the following table as a reference to locate the support office for your region.

When contacting Altair support, specify the product and version number you are using along with a detailed description of the problem. It is beneficial for the support engineer to know what type of workstation, operating system, RAM, and graphics board you have, so include that in your communication.

Location	Telephone	E-mail
Australia	+61 3 9866 5557	anzsupport@altair.com
Brazil	+55 113 884 0414	br_support@altair.com

Location	Telephone	E-mail
Canada	+1 416 447 6463	support@altairengineering.ca
China	+86 400 619 6186	support@altair.com.cn
France	+33 141 33 0992	francesupport@altair.com
Germany	+49 703 162 0822	hwsupport@altair.de
Greece	+30 231 047 3311	eesupport@altair.com
India	+91 806 629 4500 +1 800 425 0234 (toll free)	support@india.altair.com
Israel		israelsupport@altair.com
Italy	+39 800 905 595	support@altairengineering.it
Japan	+81 3 6225 5830	support@altairjp.co.jp
Malaysia	+60 32 742 7890	aseansupport@altair.com
Mexico	+52 55 5658 6808	mx-support@altair.com
New Zealand	+64 9 413 7981	anzsupport@altair.com
South Africa	+27 21 831 1500	support@altair.co.za
South Korea	+82 704 050 9200	support@altair.co.kr
Spain	+34 910 810 080	support-spain@altair.com
Sweden	+46 46 460 2828	support@altair.se
United Kingdom	+44 192 646 8600	support@uk.altair.com
United States	+1 248 614 2425	hwsupport@altair.com

If your company is being serviced by an Altair partner, you can find that information on our web site at https://www.altair.com/PartnerSearch/.

See www.altair.com for complete information on Altair, our team, and our products.

User Guide

1

View detailed information regarding the features, functionality, and simulation methods available in ultraFluidX.

This chapter covers the following:

- 1.1 Overview (p. 10)
- 1.2 Use ultraFluidX (p. 11)
- 1.3 Simulation Parameters (p. 15)
- 1.4 Expert Parameters (p. 39)

1.1 Overview

ultraFluidX is a simulation tool for the prediction of the aerodynamic properties of passenger and heavy-duty vehicles, as well as for the evaluation of building and environmental aerodynamics.

Its cutting-edge technology, based on the Lattice Boltzmann Method is optimized for GPUs to deliver extreme performance. Therefore, a CUDA-capable NVIDIA GPU device is always required to run ultraFluidX.

Main features of ultraFluidX 2022 include:

- D3Q27 Cumulant Lattice Boltzmann model for isothermal, unsteady 3D single-phase flows.
- CUDA-aware MPI support for multi-GPU and multi-node usage.
- Case Setup via Altair's Virtual Wind Tunnel (support for box-shaped simulation domains).
- Automated volume mesh generation with local grid refinement and support for baffle parts.
- LBM-consistent Smagorinsky LES turbulence model and wall modeling.
- Porous media model (pressure drop) for simulation of multiple heat exchangers.
- Handling of rotating geometries via wall-velocity boundary conditions, a Moving Reference Frame (MRF) model, or truly rotating overset grids (OSM).
- Support for moving floors (single and 5-belt system) and static floors with boundary layer suction.
- Advanced post-processing capabilities, for example, window averaging, spatial drag/lift contribution, probe output and section cut output.

1.2 Use ultraFluidX

Check system requirements and launch ultraFluidX.

Before using ultraFluidX, make sure that enough compatible GPU devices are available on your machine(s) for your desired simulation, especially in terms of available memory on the devices and their compute performance.

ultraFluidX requires at least one NVIDIA GPU device with compute capability 3.5 or higher and NVIDIA drivers version 450.80.02 or higher, and it is strongly recommended to use a minimum of four GPUs based on the NVIDIA Ampere Architecture, for example, NVIDIA A100, to allow for overnight simulations of typical passenger-car setups on a single machine.

ultraFluidX always needs MPI to run a simulation. The main rank, 0 handles the preprocessing and the I/O, whereas each additional secondary rank 1...n handles one GPU device each. Therefore, ultraFluidX must be started with a minimum of 2 MPI ranks to use at least 1 GPU device for the simulation.

To start ultraFluidX on 4 GPU devices for example, execute the BASH command,

```
mpirun -np 5 ultraFluidX case.xml
```

`case.xml` specifies the simulation parameters and needs to reference a tessellated surface mesh in `STL` format, a small example (`cube.xml`, `cube.stl`) is located in the installation folder `uFX/examples/cube/`.

If there are more GPUs available than needed for running the simulation, the number of GPUs that is accessible by ultraFluidX can be selected via the environment variable `CUDA_VISIBLE_DEVICES`, that is, by executing the BASH command,

```
export CUDA_VISIBLE_DEVICES=0,1,2,3
```

in your terminal to restrict the usable GPUs on the respective machine to devices 0 to 3.

> **CAUTION:** If ultraFluidX cannot find as many GPU devices as you expect to be available on your machine, verify that `CUDA_VISIBLE_DEVICES` has not been set unintentionally.

For convenience, ultraFluidX can also be launched via the `uFX` script in the `altair/scripts` folder:

```
ufx -np 5 -inpFile case.xml
```

The script sets the required environment variables, wraps the `mpirun` call, and redirects the output of ultraFluidX into a log file that is automatically tagged with the current time stamp.

1.2.1 Additional Command Line Options

Starting ultraFluidX with the parameter *-h (ultraFluidX -h)* exposes additional command line options and special ultraFluidX functions without license checkout in the terminal.

The special functions also work without using MPI.

A helpful option is the specification of a separate output folder or path via the option *-o*, where a location, such as *output*, will automatically be created if it will be a new folder in the current directory:

```
mpirun -np 5 ultraFluidX case.xml -o output
```

If an output path is not specified, ultraFluidX uses the current run directory for the output by default.

1.2.2 Workflow and Output

After ultraFluidX has been launched, it will read in the tessellated surface mesh, as well as the simulation parameters, and then start to generate the volume mesh based on the specified refinement zones.

When the volume mesh is ready and the surface mesh has been optimized according to the respective parameters, such as *<triangle_splitting>*, ultraFluidX will start the actual simulation on the GPUs and perform as many iterations on the coarsest refinement level as specified via *<num_coarsest_iterations>*. On higher refinement levels, the necessary number of sub-iterations will be performed according to the mesh size of the respective level.

Terminal Output

During the simulation, ultraFluidX will continuously write some output to the terminal, for example, the current coarsest iteration and the corresponding physical simulation time, the current drag, lift and side force coefficients, as well as the averaged computational performance and the presumable end of the simulation.

File Output

Additionally, ultraFluidX will create the following files and folders in the output path, if specified, or in the current run directory (if option `-o` was omitted, see below) for the output of various result types:

uFX_coefficientsData
: Folder including the instantaneous and averaged aerodynamic coefficients (`uFX_coefficients_Inst.txt` and `uFX_coefficients_Avg.txt`) and additional coefficient data per section or part, according to input in the *<output> - <aero_coefficients>* section.
This folder contains information on the instantaneous and averaged mass flow rates in the porous medium, if any, according to the *<sources> - <porous>* section.

uFX_fullData
: Folder including the full output dataset according to the specifications in the *<output> - <general>* section. This folder contains all the output in EnSight Gold format.

> **Note:** ultraFluidX outputs the data in a distributed manner, per MPI rank. Additionally, a so-called "Server of Servers" file (`uFX_output.sos`) will be created that refers to the `*.case` files, per `rank:uFX_output_#variable _RANK#rank_#timestep.data`. Files that contain the actual result data, per variable, per rank, and per time step.

uFX_output_RANK#rank.geo
: Files that contain the volumetric mesh data, per rank.

uFX_output_RANK#rank.case
: Individual case files that collect the available data, per rank.

uFX_output.sos
: "Server of Server" file that collects all the distributed information and links the individual `*.case` files. The `*.sos` file can be used by many visualization tools to handle the `*.case` files, as if they were one output file.

ultraFluidX also offers specific output files for processing the result data with AcuFieldView:

uFX_output_RANK#.bcf
: "Binary case file" that collects the available data, per rank.

uFX_output.layout
: Layout file that collects the distributed `*.bcf` files.

> **Note:** ultraFluidX can also merge the individual files, per rank after the simulation is completed. Refer to the *<output> - <general> - <merge_output_files>* option.

ultraFluidX offers to merge files manually by specifying the *-merge* option.

uFX_meshData
: Folder with all mesh related output, including the mesh preview (if activated, see *<meshing> - <general> - <mesh_preview>*), the restart file (if activated, see *<meshing> - <general> - <mesh_export>*) and a list of void triangles that were removed from the specified tessellated surface mesh during meshing, if any.

uFX_surfaceData
: Folder containing the surface data of the simulation results (output with respect to the respective parameter *<output_frequency>*), including the output of partial surface, if activated in the *<output> - <partial_surface>* section.
The output has the same structure as the content of the `uFX_fullData` folder.

uFX_volumeData
: Folder that contains the partial volume output, if specified in the *<output> - <partial_volume>* section.
The output has the same structure as the content of the `uFX_fullData` folder.

1.2.3 Multi-Node Usage

ultraFluidX can generally be used on an arbitrary number of compute nodes.

However, be aware of typical strong scaling behavior, that is, for each problem size there is a maximum reasonable number of GPU devices. If the problem size per GPU device becomes too small, the communication between devices will start to dominate and the overall compute performance will not increase further.[1]

To start ultraFluidX on multiple compute nodes, it is recommended to use one of the two MPI options `--hostfile` or `--host`. Just like on a single machine, the main rank, 0 handles the preprocessing and the I/O, whereas each additional secondary rank 1...n handles one GPU device each. Therefore, if you

want to run ultraFluidX on, for example, three machines with 2 GPU devices each, the simulation must be started with a total number of 7 MPI ranks (main rank 0 + 3×2 GPU ranks).

If the names of the machines from the example above were `node1`, `node2` and `node3` with `node1` containing the main rank 0, the BASH command when using the `--host` option would be (`-np` is not needed in this case):

```
mpirun --host node1,node1,node1,node2,node2,node3,node3 ultraFluidX case.xml
```

Alternatively, the same behavior could be achieved when using a so-called "hostfile" named, for example, `uFX_hosts` containing the following lines:

```
node1 slots=3
node2 slots=2
node3 slots=2
```

The BASH command to start ultraFluidX with this option would then be:

```
mpirun --hostfile uFX_hosts -np 7 ultraFluidX case.xml
```

Note: ultraFluidX must be able to reach/use the same locations, folders, and so on, on all compute nodes, which is typically the case for a shared network file system.

For further documentation regarding multi-node usage and also general handling and run-time tuning of Open MPI, refer to the respective FAQ on the Open MPI web page: https://www.open-mpi.org/faq/

1. Niedermeier, C. A., Janßen, C. F., & Indinger, T. (2018). Massively-parallel multi-GPU simulations for fast and accurate automotive aerodynamics. In 7th European conference on computational fluid dynamics.

1.3 Simulation Parameters

Available parameters for defining simulations in ultraFluidX.

ultraFluidX Solver Input Deck

The format for the ultraFluidX solver input deck is XML. The file starts with the XML prolog specifying the XML version and the character encoding:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The prolog is followed by the root element,

```
<uFX_solver_deck name="ultraFluidX" xmlns="https://www.altair.com/ultrafluidx/http://
www.ultrafluidx.com">
```

which is parent to all other elements in the file, so the file must always end with *</uFX_solver_deck>* according to XML standards.

All categories and parameters must be specified in this way. The category or parameter, for example, always begins with a command of the format *<example>* and ends with the corresponding command *</example>*, with the respective child entries or the parameter specification in between.

The ultraFluidX solver input deck is grouped into five main categories: *<simulation>*, *<meshing>*, *<boundary_conditions>*, *<sources>* and *<output>*, each of them with several sub-categories and possible parameters as described below.

An example case (`roadster.xml`, `roadster.stl`) using the most important features including *<rotating>*, *<belt>*, *<porous>* and *<probe_file>* is located in the installation folder `uFX/examples/roadster/`.

1.3.1 <simulation>

The *<simulation>* category specifies the main parameters of the simulation like the number of iterations, and the file containing the surface mesh or the fluid material.

<simulation> - <general>

The *<general>* category contains the following parameters:

<num_coarsest_iterations>
: Defines the number of iterations at the coarsest refinement level. The physical time per coarsest iteration depends on the material parameters and the Mach factor.

<mach_factor>
: Specifies the scaling factor for the Mach number (1.0 = Mach-matched simulation).The time step increases linearly with this factor. With a value of 2.0, for example, the simulation runs approximately twice as fast for the same simulated physical time.

> **Note:** ultraFluidX is using a weak compressible LBM formulation, which is only valid up to approximately Ma = 0.4, so consider the maximum velocity that may occur in your simulation when setting this parameter to not exceed this limit.

Default = 1.0

<moving_ground>
The wind tunnel ground is a no-slip wall with zero velocity by default. If it should be moving with *<reference_velocity>* instead, the parameter must be set to true. If belts are specified (*<boundary_conditions>*), only the belts will move, not the whole ground.

<rotating_wheels>
By specifying this parameter, all rotating boundary conditions (*<boundary_conditions>*) can be switched on or off at once. If set to false, all entries in the *<rotating>* category will be effectively ignored.

<boundary_layer_suction>
Specifies if a part of the wind tunnel ground starting from the inlet to *<boundary_layer_suction_xpos>* should be treated as a slip wall, instead of a no-slip wall.

<boundary_layer_suction_xpos>
Specifies the x-coordinate (in $[\mathrm{m}]$), where the slip wall will end, if *<boundary_layer_suction>* is set to true.

<reference_velocity>
The reference velocity (specified in $\left[\frac{\mathrm{m}}{\mathrm{s}}\right]$) is used as velocity at the inlet. It is also used as wall velocity for any belts (if specified) or for the whole wind tunnel ground and for the calculation of the aerodynamic coefficients.

<simulation> - <geometry>

The *<geometry>* category contains the following parameters:

<source_file> *or* *<source_files>* - *<name>*
Specifies the path to the source file(s) that contain(s) the tessellated surface mesh. The source file(s) must be in ASCII STL format and all surface normals must point outwards, that is, from solid to fluid. ultraFluidX assumes that the STL file is specified in $[\mathrm{m}]$ and all other coordinates for bounding boxes, probe locations, etc. are based on the coordinate system of the STL.
While *<source_file>* (without an additional *<name>* parameter) is used for cases where the complete surface is available in a single STL file, the category *<source_files>* allows to specify more than one input file, each filename being indicated under the parameter *<name>*.

> **Warning:** If parameters, *<source_file>* and *<source_files>* are specified simultaneously, the run will stop.

<baffle_parts>
(Optional) Specifies all parts which are fully two-dimensional, each part with this property must be identified via the child parameter *<name>*. This property should be set for parts which do not

form a closed volume together with other parts and which are expected to be connected to fluid regions on both sides. ultraFluidX will then internally create a duplicate of each of the parts with the appendix "_inv" and inverted triangle normals to be able to assign proper pressure and wall shear stress values to both sides of the part.

> **CAUTION:** Only specify this property for the baffle part; otherwise, unwanted side effects might occur.

<surface_mesh_optimization> - <triangle_splitting>

Specifies the triangle splitting to increase the accuracy of the calculation of aerodynamics forces that act on the surfaces. Either global or local splitting can be selected. The default setting is global, controlled by the following child parameters:

<active>

(Global) Switch that activates the triangle splitting.
Default = True

<max_relative_edge_length>

This factor multiplies the voxel size connected to the geometric part to compute the splitting threshold. Setting this factor to zero makes the variable inactive. Negative values are corrected to zero. If this parameter and *<max_absolute_edge_length>* are both greater than zero, the smaller value per part has precedence.

<max_absolute_edge_length>

Absolute threshold (in [m]) of splitting. Default values is set to zero and hence inactive. Negative values are corrected to zero. If this parameter and *<max_relative_edge_length>* are both greater than zero, the smaller value per part has precedence.

Alternatively, the triangle splitting can be activated only on certain parts of the geometry with the *<triangle_splitting_instance>* category with the following parameters:

<name>

(Optional) Name of the instance.

<active>

Same as the global parameter but for local setting; has precedence over global setting.

<max_relative_edge_length>

Same as the global parameter but for local setting; has precedence over global setting.

<max_absolute_edge_length>

Same as the global parameter but for local setting; has precedence over global setting.

<parts>

Specify the parts on which the local settings of the splitting are applied via the child parameter *<name>*. Several parts can be specified via repeated usage of *<name>* to assign the local settings to all of them. Part names must be valid and not repeated in the same or other instances, or the run stops.

> **Note:** Old triangle splitting parameters are still supported. However, the new triangle splitting category has precedence over it.

<triangle_plinth>
ultraFluidX automatically creates a plinth in regions where single voxels would be squeezed between the wind tunnel ground and, for example, the tire of a car by treating such single voxels as solid. If *<triangle_plinth>* is set to true, the corresponding surface triangles of these solid voxels will be created to properly visualize the behavior and to increase the accuracy of the calculation of aerodynamic forces in that region.

<domain_bounding_box>
The bounding box of the simulation domain, must be specified via the child parameters *<x_min>*, *<x_max>*, *<y_min>*, *<y_max>*, *<z_min>* and *<z_max>* in $[\mathrm{m}]$.
ultraFluidX currently needs the domain size in each direction to be a multiple of four of the *<coarsest_mesh_size>*. Therefore, the bounding box is automatically enlarged in +x-direction, +z-direction and symmetrically in y-direction if the requirement is not met.

<simulation> - <material>

The *<material>* category is used to describe the properties of the fluid material that should be assumed for the simulation, for example, air, and contains the following parameters:

<density>
Density ρ of the material $\left[\frac{\mathrm{kg}}{\mathrm{m}^3}\right]$.

<dynamic_viscosity>
Dynamic viscosity μ of the material $\left[\frac{\mathrm{kg}}{(\mathrm{s}\cdot\mathrm{m})}\right]$.

<temperature>
Temperature (T) of the material $[\mathrm{K}]$.

<specific_gas_constant>
Specific gas constant (R) of the material $\left[\frac{\mathrm{J}}{\mathrm{kg}\cdot\mathrm{K}}\right]$.

<simulation> - <wall_modeling>

ultraFluidX allows you to specify the handling of near wall regions of the flow using the following parameters:

<wall_model> *(GWF, WangMoin, off)*
This parameter specifies the way in which information (velocity, pressure) tracked at the donor position is processed to determine the wall shear stress. Available options:

GWF (Default) [2]

WangMoin [3]

off

<coupling> (two-way, one-way, off)

This parameter determines what is done with the information generated by the wall model.

two-way coupling (Default)	The wall shear stress determined using the wall model is used as an input value for the computation of the velocity at the first near wall voxel.
one-way coupling	The flow in the first near wall cell is derived not via the wall shear stress, but based on information of the donor position used by the wall model; while the wall shear stress, determined by wall model is only used for the force calculations near the wall.
off	Information from the wall model is not used in the flow field computations. Instead, a no-slip boundary condition is used and the wall shear stress is computed from flow field information.

<pressure_gradient> (favorable, adverse, full, off)

It is possible to specify the way in which pressure gradients along the wall should be considered by the wall model.

off	The wall model will not consider pressure gradients present in the mean flow field and determine the wall shear stress neglecting the respective pressure-gradient term.
favorable and adverse	Consider only negative and positive pressure gradients in direction of the flow, respectively.
full	Takes any pressure gradient near the wall into account.

<simulation> - <wall_model_variant>

You can specify different wall model settings than the global default (specified via *<simulation> - <wall_modeling>*) on different parts of the vehicle.

For this purpose, an arbitrary number of such different wall model settings can be specified as children of *<wall_model_variant>* via *<wall_model_variant_instance>*, each with the following parameters:

<name>

Name of the instance, so that it can be referenced in the respective *<rotating_instance>* or *<static_instance>* to which it should be applied (mandatory).

2. Shih, T. H., Povinelli, L. A., & Liu, N. S. (2003). Application of generalized wall function for complex turbulent flows. Journal of Turbulence, 4, 015
3. Wang, M., & Moin, P. (2002). Dynamic wall modeling for large-eddy simulation of complex turbulent flows. Physics of Fluids, 14(7), 2043-2051

<wall_model> (GWF, WangMoin, off)
See above in *<simulation> - <wall_modeling>*.

<coupling> (two-way, one-way, off)
See above in *<simulation> - <wall_modeling>*.

<pressure_gradient> (favorable, adverse, full, off)
See above in *<simulation> - <wall_modeling>*.

Note: The use of wall model variants is currently not supported in overset mesh regions. The wall model within the overset regions is globally specified in *<simulation> - <wall_modeling>*.

1.3.2 <meshing>

During meshing, ultraFluidX discretizes the volumetric simulation domain into cubic elements (“voxels”) for which the flow field is computed. The *<meshing>* category specifies the main parameters for the volume meshing and the refinement zones.

<meshing> - <general>

The *<general>* category contains the following parameters:

<coarsest_mesh_size>
Specifies the coarsest size (edge length) of the voxels in $[\mathrm{m}]$ that ultraFluidX uses to discretize the domain. Higher resolution than the coarsest mesh size in regions of interest is specified via the *<refinement>* category.

<mesh_preview>
Specifies if a preview of the volume mesh is exported prior to the actual simulation. If set to true, ultraFluidX writes an EnSight Gold file (`uFX_meshPreview.case` and corresponding `*.geo` and `*.data` files) into the `uFX_meshData` folder.
Information included in the mesh preview is the state of the voxels (solid: -1, fluid: 0), the applied boundary conditions (none and solid: -1, none and fluid: 0, moving/rotating: 1, no-slip: 2, slip: 3, outlet: 4, inlet: 5), the source nodes (none and solid: -1, none and fluid: 0, numbering from 1 onwards according to number of source regions) and the rotating nodes (none and solid: -1, none and fluid: 0, numbering from 1 onwards according to number of rotating regions).
Default = false

<mesh_export>
Specifies if the mesh is written to disk after the meshing is completed. If set to true, ultraFluidX will write a mesh source file `uFX_mesh.bin` to the `uFX_meshData` folder, that can then be used as input for a future simulation. Refer to the *<mesh_source_file>* option.
Default = false

<mesh_source_file>
(Optional) Specifies the path to a mesh source file. If specified, ultraFluidX skips the meshing step and read in the mesh topology from the specified mesh source file instead. Generally, only mesh source files that were written by the same version of ultraFluidX (by setting *<mesh_export>* to true) can be used.

<meshing> - <refinement>

Refinement zones are specified via the *<refinement>* category. Possible sub-categories are *<box>*, *<offset>* and *<custom>* and they all share the parameter *<refinement_level>* that specifies the refinement of the zone with regards to the coarsest level. With each additional level the coarsest mesh size (corresponding to level 0) is divided by 2. In general, the voxel size of the n^{th} refinement level is defined as:

$$\Delta x_n = \frac{\Delta x_0}{2^n} \tag{1}$$

Where Δx_0 is the voxel size at the coarsest refinement level.

<box>
This category contains all refinement zones with cuboid shape, it can have an arbitrary number of children named *<box_instance>* with the following parameters:

<name>	(Optional) Name of the instance.
<refinement_level>	Refinement level of the instance with regards to the coarsest level.
<bounding_box>	Bounding box of the instance, must be specified via the child parameters *<x_min>*, *<x_max>*, *<y_min>*, *<y_max>*, *<z_min>* and *<z_max>* in $[\mathrm{m}]$.

<offset>
This category contains all refinement zones that are an offset of the whole surface mesh or a specific part. It can have an arbitrary number of children named *<offset_instance>* with the following parameters:

<name>	Name of the instance (optional).
<refinement_level>	Refinement level of the instance with regards to the coarsest level.
<normal_distance>	Distance of the offset to the surface, that is, the effective "thickness" of the zone. Specified in $[\mathrm{m}]$.

> **Note:** A minimum number of 4 voxels will always be applied.

<parts> If the offset is not applied to the whole surface mesh, an arbitrary number of individual parts can be specified via the child parameter *<name>* within the *<parts>* category. The specified names must exactly match the respective part names in the STL file.

<custom>

This category contains all refinement zones that are specified via a volume with arbitrary shape contained in the STL file. Any volume that you want to use for custom refinement needs to be identified as a separate "solid" within the STL file, and the name of this solid needs to be passed to ultraFluidX in the *<parts>* - *<name>* section.
The volume must be closed and all normals of the volume must point outwards.

> **CAUTION:** If the volume is included in the STL, but not specified as a custom zone, it will be treated as a solid by ultraFluidX.

The category can have an arbitrary number of children named *<custom_instance>* with the following parameters:

<name>

Name of the instance (optional).

<refinement_level>

Refinement level of the instance with regards to the coarsest level.

<parts>

<name>

The closed volume(s) contained in the STL file must be specified via the child parameter *<name>* within the *<parts>* category, several volumes can be specified via repeated usage of *<name>* to assign the same refinement level to all of them. The specified names must exactly match the respective part names in the STL file.

<meshing> - <overset>

This category contains all parameters needed to define a moving overset grid region.

<rotating> - <rotating_instance>

In these grid regions, a local frame of reference is defined and the surface geometry and voxels are rotated each timestep according to the specified rotational speed.

> **Note:** The rotating volume must be axisymmetric.

> **CAUTION:** All geometry parts inside the moving mesh region will rotate with the mesh. Care is needed at the interface when setting up a rotating volume in the vicinity of geometry parts. Refer to separate setup recommendations.

<name>
(Optional) Name of the rotating instance. If not specified, a generic name is attributed.

<rpm>
This parameter indicates the rotational speed of the moving overset grid region in revolutions per minute.

* *

<x_pos>, <y_pos>, <z_pos>
The center point around which the moving region rotates.

<axis>

<x_dir>, <y_dir>, <z_dir>
The axis of rotation of the overset mesh zone.

<parts>

<name>
This section contains the `STL` parts that enclose the rotating region.

> **Note:** The parts must form a closed volume, and normals must point outwards.

The rotating region needs to be axisymmetric, such as cylindrical or toroid shape.

1.3.3 <boundary_conditions>

This category specifies parameters for special types of boundaries.

<boundary_conditions> - <rotating>

This category contains all regions where a rotating wall velocity should be applied to the surface. It can have an arbitrary number of children named *<rotating_instance>*, each with the following parameters:

<name>
Name of the instance (optional).

<rpm>
Rotational speed of the instance in [rpm] (revolutions per minute).

* *
Center of the rotation, specified via the child parameters *<x_pos>*, *<y_pos>* and *<z_pos>* in $[\mathrm{m}]$.

<axis>
Rotational axis of the instance, its direction (right-hand rule applies) must be specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*.

<parts>
: The parts to which the rotating wall velocity should be applied must be specified via the child parameter *<name>* within the *<parts>* category, several parts can be specified via repeated usage of *<name>* to assign the rotating wall velocity to all of them. The specified names must exactly match the respective part names in the `STL` file.

<boundary_conditions> - <belt>

This category contains all regions where a moving wall velocity should be applied to the wind tunnel ground (if *<moving_ground>* is set to true, see above), it can have an arbitrary number of children named *<belt_instance>*, each with the following parameters:

<name>
: Name of the instance (mandatory).

<include_force>
: If set to true, aerodynamic forces that act on the belt instance are included in the force evaluation.
Default = false

<bounding_box>
: Bounding box of the instance, must be specified via the child parameters *<x_min>*, *<x_max>*, *<y_min>* and *<y_max>* in $[\mathrm{m}]$. The z-coordinate is omitted because the belt is always located on the wind tunnel ground.

<boundary_conditions> - <periodicity>

Via the child parameters *<x_dir>*, *<y_dir>*, and *<z_dir>*, periodic *<boundary_conditions>* can be activated.

> **Note:** At least eight voxels in periodic direction have to be used.

The default value for all three children is *false*.

<boundary_conditions> - <static>

This category contains all regions where a no-slip *<boundary_condition>* with non-default wall model settings should be applied to the surface. It can have an arbitrary number of children named *<static_instance>*, each with the following parameters:

<name>
: Name of the instance (optional).

<wall_model_variant>
: Name of the non-default wall model variant to be applied, must be specified in *<simulation> - <wall_model_variant>*.

<parts>

The parts to which the properties should be applied must be specified via the child parameter *<name>* within the *<parts>* category. Several parts can be specified via repeated usage of *<name>* to assign the properties to all of them. The specified names must match the respective part names in the `STL` file.

> **Note:** Static parts will use the globally specified wall model settings by default. They must be specified separately, if settings other than global settings are applied locally.

1.3.4 <sources>

This category specifies parameters for volumetric sources (and sinks) of the following kinds: *<porous>*, *<turbulence>*, *<mrf>*, and *<acoustic>*.

<sources> - <porous>

This category contains all porous regions where a volumetric body force should be applied to the fluid according to:

$$\frac{-\partial p}{\partial x} = \rho R_v u + \rho R_i u^2 \qquad (2)$$

Where,

R_v	Viscous coefficient.
R_i	Inertial resistance coefficient.

These regions must be contained in the `STL` file as a closed volume and all normals of the volume must point outwards.

> **CAUTION:** If the volume is included in the `STL` and not specified as porous region, it is treated as a solid by ultraFluidX.

For each porous medium, the text files `uFX_porousMassFlow_Inst.txt` and `uFX_porousMassFlow_Avg.txt` in the `uFX_coefficientsData` folder contain the instantaneous and averaged mass flow rates and the corresponding cross sections per porous medium (output at every coarsest iteration in $\left[\frac{\mathrm{kg}}{\mathrm{s}}\right]$).

If a porous region is defined as isotropic (*<porous_axis>* is set to zero), the region is included in the file, but the mass flow rate cannot currently be evaluated. It is reported as zero.

The category can have an arbitrary number of children named *<porous_instance>*, each with the following parameters:

<name>	Name of the instance (optional).

<inertial_resistance>	Inertial resistance (see Equation 2) of the instance in $\left[\frac{1}{\mathrm{m}}\right]$.
<viscous_resistance>	Viscous resistance (see Equation 2) of the instance in $\left[\frac{1}{\mathrm{s}}\right]$.
<porous_axis>	Flow direction through the instance. Crossflow that is not aligned with the porous axis is blocked. Its components are specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*. If all components are set to zero, no preferred direction is used and the instance acts as an isotropic porous region, for example, bug screens.
<parts>	The closed volume(s) to which the volumetric body force should be applied and which are contained in the `STL` file must be specified via the child parameter *<name>* within the *<parts>* category, several volumes can be specified via repeated usage of *<name>* to assign the same properties to all of them. Additionally, if a closed volume consists of separate parts, they can also be specified via repeated usage of *<name>* to effectively form the closed volume. The specified names must exactly match the respective part names in the `STL` file.

<sources> - <turbulence>

This category contains all areas where synthetic turbulence is generated according to the so-called "*vortex method*." [4]

Currently, the turbulence generator adds turbulent structures for flows in positive x-direction only. It is set up by an x-position and the extents in y- and z-direction.

ultraFluidX automatically selects the corresponding voxel layer with a thickness of one voxel where the synthetic turbulence is added. All voxels of each turbulence instance are refined to the finest voxel size that is found in the instance.

The category can have an arbitrary number of children named *<turbulence_instance>*, each with the following parameters:

<name>
: (Optional) Name of the instance.

<point>

: *<x_pos>*
: Global x-position of the turbulence generation zone.

<bounding_box>

: *<y_min>*, *<z_min>*, *<y_max>*, *<z_max>*
: Extents of the turbulence generation zone in the global y- and z-direction.

<num_eddies>
: Number of eddies that will be synthesized within the region in a pseudo-random manner. Should be in the order of 100-1000.

<length_scale>
: Characteristic length scale for the synthetic turbulence in $[\mathrm{m}]$. Should be in the order of (or identical to) the voxel size in the turbulence generator region.

<turbulence_intensity>
: Intensity of the synthetic turbulence, that is, ratio between root-mean-square of the turbulent velocity fluctuations and the mean velocity.

<sources> - <mrf>

This category contains all the regions in which ultraFluidX applies body forces to account for a moving reference frame.

> **Note:** The ultraFluidX MRF model is applied on top of the wall velocity *<boundary_conditions>*, which must be specified separately in the *<boundary_conditions>* section of the input deck.

The MRF regions must be contained in the `STL` file as a closed volume and all normals of the volume must point outwards.

> **Note:** If the volume is included in the `STL` and not specified as an MRF region, it will be treated as a solid by ultraFluidX.

The *<mrf>* category can have an arbitrary number of children named *<mrf_instance>*, each with the following parameters:

<name>
: (Optional) Name of the instance.

<rpm>
: Rotational speed of the instance in [rpm] (revolutions per minute).

* *
: Center of rotation for each instance. Specified via the child parameters *<x_pos>*, *<y_pos>* and *<z_pos>* in $[\mathrm{m}]$.

<axis>
: Rotational axis of the instance. Its direction (right-hand rule applies) must be specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*.

4. Mathey, F., Cokljat, D., Bertoglio, J. P., & Sergent, E. (2006). Assessment of the vortex method for large eddy simulation inlet conditions. Progress in Computational Fluid Dynamics, An International Journal, 6(1-3), 58-67.

<parts>

The closed volume(s) in which the MRF body force terms should be applied and which are contained in the STL file must be specified via the child parameter *<name>* within the *<parts>* category.
Several volumes can be specified via repeated usage of *<name>* to assign the same properties to all of them.
Additionally, if a closed volume consists of separate parts, they can also be specified via repeated usage of *<name>* to effectively form the closed volume. The specified names must exactly match the respective part names in the STL file.

<sources> - <acoustic>

This category specifies acoustic sources to create monopole sources. The category can have an arbitrary number of children named *<acoustic_instance>* with the following parameters:

<amplitude>

Amplitude of the monochromatic acoustic wave in $[\mathrm{Pa}]$.

<frequency>

Frequency of the monochromatic acoustic wave in $[\mathrm{Hz}]$.

<position>

Position of the acoustic source with the child parameters *<x_pos>*, *<y_pos>* and *<z_pos>* in $[\mathrm{m}]$.

1.3.5 <output>

The *<output>* category specifies the main parameters for various types of simulation output.

Table 1: Overview of Available Output Options for Section Cut, Volume Data, and Surface Data

	Full Data	**Surface**	**Section Cut**	**Partial Surface**	**Partial Volume**
pressure	Default	Default	Default	Default	Default
time_avg_pressure	Default	Default	Default	Optional[2]	Optional[2]
window_avg_pressure	Optional	Optional	Optional	Optional[2]	Optional[2]
velocity	Default	Default[1]	Default	Optional[1]	Default
time_avg_velocity	Default	N.A.	Default	N.A.	Optional[2]

5. Mathey, F., Cokljat, D., Bertoglio, J. P., & Sergent, E. (2006). Assessment of the vortex method for large eddy simulation inlet conditions. Progress in Computational Fluid Dynamics, An International Journal, 6(1-3), 58-67.

	Full Data	Surface	Section Cut	Partial Surface	Partial Volume
window_avg_velocity	Optional	N.A.	Optional	N.A.	Optional[2]
wall_shear_stress	Default	Default	N.A.	Default	N.A.
time_avg_wall_shear_stres	Default	Default	N.A.	Optional[2]	N.A.
window_avg_wall_shear_sti	Optional	Optional	N.A.	Optional[2]	N.A.
surface_normal	Default	Default	N.A.	Optional	N.A.
mesh_data	Default	N.A.	N.A.	N.A.	N.A.

[1]The surface velocity is 0 on static surfaces and non-zero if a wall velocity is prescribed using the *<boundary_condition> - <rotating>* section.

[2]Section cut, partial surface, and partial volume allow for sub-iteration output (for example, output frequency < 1).

Note: If time- or window-averaged quantities for partial surface and partial volume are requested, the subiteration output for the respective instance is deactivated.

<output> - <general>

The *<general>* category contains the following parameters:

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs of the volume and the surface data in EnSight Gold format.

<output_start_iteration>
Specifies after which coarsest iteration the output will start.
Default = 0

<merge_output_files>
Specifies if the output files written in a partitioned way per MPI rank are merged into single files after the simulation is complete. The individual files per MPI rank are not deleted by this operation, so the data size of the output is effectively doubled.
Default = false

<saved_states>
Specifies the number of consecutive output files to be kept, that is, if set to 2, the latest two output time steps are kept while the rest are removed. If set to zero, the feature is deactivated and all output time steps are kept.
Default = 0

<avg_start_coarsest_iteration>
Specifies after which coarsest iteration the continuous averaging of variables and force coefficients starts. The averaging itself also uses information from subiterations on the finer refinement levels.

<avg_window_size>
Specifies the number of consecutive coarsest iterations that will be used for window averaging. This value should be a factor of *<output_frequency>*.

<output_variables_full>
In this section, you can specify which of the variables listed in Table 1 should or should not be included in the full data output by setting *<variable_name>* to true or false.

> **Note:** The default case is to export all variables, thus it is not required to explicitly set the variables to "true"; if not mentioned in this section, the variables are still written. To omit a variable from being written, you have to specifically set the corresponding solver deck entry to "false".

<output_variables_surface>
In this section, you can specify which of the variables listed in Table 1 should be included and which should be excluded in the surface data output by setting *<variable_name>* to true or false.

> **Note:** The default case is to export all variables, thus it is not required to explicitly set the variables to "true"; if not mentioned in this section, the variables are still written. To omit a variable from being written, you have to specifically set the corresponding solver deck entry to "false".

<output> - <moment_reference_system>

This category specifies the properties of the moment reference system that is used to calculate the aerodynamic forces in a body-fixed coordinate system and also to calculate the pitch moment to be able to split the lift into front and rear part. The moment reference system is defined according to automotive aerodynamics conventions and needs to be placed in the center of the four support points defined by the wheel positions (in general on the ground plane).

As reference length, you need to indicate the wheel base (the distance between front and rear axle).

It contains the following parameters:

<origin>
The origin of the moment reference system, must be specified via the child parameters *<x_pos>*, *<y_pos>* and *<z_pos>* in $[\mathrm{m}]$.
Default values = 0.0, 0.0, 0.0

<roll_axis>
The roll axis of the moment reference system, must be specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*.
Default values = 1.0, 0.0, 0.0

<pitch_axis>
The pitch axis of the moment reference system, must be specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*.
Default values = 0.0, 1.0, 0.0

<yaw_axis>
The yaw axis of the moment reference system, must be specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*.
Default values = 0.0, 0.0, 1.0

<output> - <aero_coefficients>

This category specifies the properties that are connected to the output of the aerodynamic coefficients.

By default, the instantaneous and averaged aerodynamic coefficients are recorded at every coarsest iteration in the files `uFX_coefficients_Inst.txt` and `uFX_coefficients_Avg.txt` in the `uFX_coefficientsData` folder.

The category contains the following parameters:

<reference_area>
The reference area used to calculate the aerodynamic coefficients (drag, lift, side force) in $[\mathrm{m}^2]$.

<reference_area_auto>
Specifies if the value of *<reference_area>* should be used (false) or if the reference area (the frontal projection area) should be automatically calculated by ultraFluidX itself (true).
Default = true

<reference_length>
The reference length (in $[\mathrm{m}]$) that is used to calculate the pitch moment to be able to subsequently calculate the front and rear lift coefficients. See <output> - <moment_reference_system>.

<passive_parts>
If certain parts of the geometry should be excluded from the calculation of the aerodynamic coefficients, they can be specified via the child parameter *<name>* within this category. The specified names must match the respective part names in the `STL` file.

<coefficients_parts>
Specifies if the aerodynamic coefficients should also be output by part. If set to true (default value), ultraFluidX will create the files `uFX_coefficientsPerPart_Inst.txt` and `uFX_coefficientsPerPart_Avg.txt` in the output folder containing the whole list of parts per time step and a subfolder named `uFX_coefficientsPerPart`, where one individual file per part (`uFX_coefficients_#PARTNAME_Inst.txt`, `uFX_coefficients_#PARTNAME_Avg.txt`) will be recorded.

<coefficients_along_axis>
Specifies the number of sections along the x-, y- and z-direction of the object that should be used for the sectional coefficient output.
Has three children: *<num_sections_x>*, *<num_sections_y>* and *<num_sections_z>* that allow you to specify the number of sections per direction.

Creates the files `uFX_coefficientsPerSection{X,Y,Z}_Inst.txt` and `uFX_coefficientsPerSection{X,Y,Z}_Avg.txt` for the instantaneous and averaged aerodynamic coefficients per section in x-, y- and z-direction of the object, respectively (output at every coarsest iteration). Set the number of sections to 0 to deactivate the output.

<forces_and_moments_export>
If set to true, the aerodynamic forces and moments will be exported in SI units. For each file `uFX_coefficients*.txt`, a corresponding file `uFX_forcesAndMoments*.txt` will be written. This includes the respective files for *<coefficients_parts>* and *<coefficients_along_axis>* if they are activated. The default value for *<forces_and_moments_export>* is false.

<output_start_iteration>
(Optional) Specifies when to start the coefficient output (coarsest iteration). The default value is 0.

<output> - <section_cut>

This category specifies section cuts on which field variables should be output independent from the general variable output. The same conventions apply for the EnSight Gold files and ultraFluidX will create a subfolder named `uFX_sectionCuts`, where the output of each section cut will be stored in a separate folder.

The category can have an arbitrary number of children named *<section_cut_instance>*, each with the following parameters:

<name>
Name of the instance, used to create the respective subfolder in `uFX_sectionCuts` (mandatory).

<axis>
Normal axis of the instance, its direction must be specified via the child parameters *<x_dir>*, *<y_dir>* and *<z_dir>*. Currently, only section cuts normal to any global axis are supported.

<point>
Point on the instance, its coordinates must be specified via the child parameters *<x_pos>*, *<y_pos>* and *<z_pos>* in $[\mathrm{m}]$.

<bounding_box>
Bounding box of the instance, must be specified via the child parameters *<x_min>*, *<x_max>*, *<y_min>*, *<y_max>*, *<z_min>* and *<z_max>* in $[\mathrm{m}]$.

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs. Fractions of 1 are allowed to get an output at the desired sub-time step level.

<output_start_iteration>
(Optional) Specifies when to start the section cut output (coarsest iteration) for each section cut instance. Can be used to generate high-frequency output for a specific part of the domain once the initial transient phase is completed. The default value is 0.

<triangulation>
: By default, the value is false and all section cuts are exported using “nsided” elements. If set to true, the respective section cut instance is exported using “tria3” elements instead.

<output_variables>
: (Optional) In this section, you can specify which variables should be included in the section cut output by setting *<variable_name>* to true or false. The possible options are listed in Table 1.

<output> - <partial_surface>

This category specifies parts on which surface variables should be output independent from the general variable output. The same conventions apply for the EnSight Gold files and ultraFluidX will create a separate folder in the subfolder named `uFX_surfaceData`, where the output of each part will be stored.

The category can have an arbitrary number of children named *<partial_surface_instance>*, each with the following parameters:

<name>
: (Optional) Name of the instance.

<parts>
: The part(s) on which the surface variables should be output, must be specified with the parameter *<name>* within the *<parts>* category. Several parts with the same output behavior can be specified via repeated usage of *<name>*. The specified names must exactly match the respective part names in the `STL` file.

<output_frequency>
: Specifies the number of coarsest iterations between consecutive outputs. Fractions of 1 are allowed to get an output at the desired sub-time step level, unless time- or window-averaged quantities are requested. If this is the case, the output frequency will be corrected to the next higher integer number.

<output_start_iteration>
: (Optional) Specifies when to start the partial surface output (coarsest iteration) for each partial surface instance. Can be used to generate high-frequency output for a specific part of the domain, once the initial transient phase is completed. The default value is 0.

<output_variables>
: In this section, you can specify which variables should be included in the section cut output by setting *<variable_name>* to true or false. The possible options are listed in Table 1.

<output> - <partial_volume>

This category specifies a volumetric part of the domain in which field variables should be output independent from the general variable output. The same conventions apply for the EnSight Gold files and ultraFluidX will create a separate folder where the output will be stored. The category can have an arbitrary number of children named *<partial_volume_instance>*, each with the following parameters:

<name>
Name of the instance, used to create the respective folder (mandatory).

<bounding_box>
Bounding box of the instance, must be specified via the child parameters *<x_min>*, *<x_max>*, *<y_min>*, *<y_max>*, *<z_min>* and *<z_max>* in $[\mathrm{m}]$.

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs. Fractions of 1 are allowed to get an output at the desired sub-time step level, unless time- or window-averaged quantities are requested. If this is the case, the output frequency will be corrected to the next higher integer number.

<output_start_iteration>
Specifies when to start the partial volume output (coarsest iteration) for each partial volume instance. Can be used to generate high-frequency output for a specific part of the domain, once the initial transient phase is completed. The default value is 0.

<output_variables>
In this section, you can specify which variables should be included in the section cut output by setting *<variable_name>* to true or false. The possible options are listed in Table 1.

> **Note:** If no variables are chosen explicitly, the full set will be exported; otherwise, only the variables that are indicated in this section are written.

<output> - <single_probe>

This category specifies single probes, where instantaneous surface data or volume data should be output independent from the general variable output. The files are stored in the `uFX_probeData` folder. Each probe creates one `.csv` file per output variable, plus an additional file that contains the probe locations (`uFX_volumeProbes_***_location.csv` and `uFX_surfaceProbes_***_location.csv`).

The category can have an arbitrary number of children named *<single_probe_instance>*, each with the following parameters:

<name>
(Mandatory) Name of the instance used to create the respective file or subfolder.

<x_pos>, *<y_pos>*, *<z_pos>*
Respective coordinate of the probe in $[\mathrm{m}]$.

<type>
Specifies if data from the nearest voxel (*volume*) or from the nearest surface triangle (*surface*) should be output.

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs. Fractions of 1 are allowed to get an output at the desired sub-time step level.

<scientific_notation>
If set to true, the output files are written in scientific notation. If set to false, the output files are written in conventional decimal notation instead.
Default = true

<output_precision>
In combination with scientific notation (default), the output includes eight significant digits. Using this parameter, you can choose an arbitrary value.
Default = 7

<output_variables>
In this section, you can specify which variables should be included in the probe output by setting *<variable_name>* to true or false. The possible options depend on the probe type:
A default volume probe has five output variables (pressure, velocity (x,y,z) and velocity magnitude). You can optionally request the output of Cp and time/window averaged pressure and velocity.

> **Note:** Volume probes do not have access to wall shear stress.

A surface probe has one default output (Cp). You can optionally request pressure, velocity (x, y, z), velocity magnitude, wall shear stress (tau_wall x, y, z), and time/window averaged pressure and wall shear stress.

<output_start_iteration>
(Optional) Specifies when to start the output (coarsest iteration) for each single probe instance.
Default = 0

<output> - <probe_file>

This category specifies probe files containing multiple locations where instantaneous surface data or volume data should be output independent from the general variable output.

The files are stored in the `uFX_probeData` folder. Each probe file instance will create one `.csv` file per output variable, plus an additional file that contains the probe locations (`uFX_volumeProbes_***_location.csv` and `uFX_surfaceProbes_***_location.csv`).

Depending on the type of the probe file, ultraFluidX will either create a single `*.csv` file containing the transient Cp data (*surface probes*, `uFX_surfaceProbes_#NAME.csv`) or multiple `*.csv` files containing pressure, velocity magnitude and velocity component data (*volume probes*, *uFX_volumeProbes_#NAME_Variable_Name*) for recording the data.

The category can have an arbitrary number of children named *<probe_file_instance>*, each with the following parameters:

<name>
Name of the instance, used to create the respective file(s) or subfolder (mandatory).

<source_file>
Source file containing the probe locations in `*.csv` format without header, each line defines a probe with the coordinates in $[\mathrm{m}]$, which must be separated by semicolons:

- `x_pos;y_pos;z_pos;Description_A`
- `x_pos;y_pos;z_pos;Description_B`

<type>
Specifies if data from the nearest voxel (*volume*) or from the nearest surface triangle (*surface*) should be output for each probe location.

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs. Fractions of 1 are allowed to get an output at the desired sub-time step level.

<scientific_notation>
If set to true, the output files are written in scientific notation. If set to false, the output files are written in conventional decimal notation instead.
Default = true

<output_precision>
Using this parameter, you can choose an arbitrary integer value.
Default = 7

<output_variables>
In this section, you can specify which variables should be included in the probe output by setting *<variable_name>* to true or false. The possible options depend on the probe type:
A default volume probe has five output variables (pressure, velocity (x,y,z) and velocity magnitude). You can optionally request the output of Cp, and time/window avgeraged pressure and velocity.

> **Note:** Volume probes do not have access to wall shear stress.

A surface probe has one default output (Cp). You can optionally request pressure, velocity (x, y, z), velocity magnitude, wall shear stress (tau_wall x, y, z), and time/window averaged pressure and wall shear stress.

<output_start_iteration>
(Optional) Specifies when to start the probe output (coarsest iteration) for each probe file instance. The default value is 0.

<output> - <monitoring_surface>

This category specifies monitoring surfaces, which are arbitrarily shaped, user-defined surfaces for additional post-processing. Monitoring surfaces can be used for the additional visual output of flow field quantities on the respective surface, or for the output of integral values in a summary file.

The files are stored in the `uFX_monitoringSurface` folder.

The category can have an arbitrary number of children named *<monitoring_surface_instance>*, each with the following parameters:

<name>
Name of the instance (mandatory).

<parts>
The part(s) which define the monitoring surface. Must be specified with the parameter *<name>* within the *<parts>* category. Several parts can be specified via repeated usage of *<name>*. The specified names must exactly match the respective part names in the STL file.

<visual_output>

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs.

<output_start_iteration>
Specifies when to start the visual output (coarsest iteration) for each instance. The default value is 0.

<output_variables>
In this section, you can specify which variables should be included in the visual output by setting *<variable_name>* to true or false. The possible options include:

- *pressure*
- *time_avg_pressure*
- *window_avg_pressure*
- *velocity*
- *time_avg_velocity*
- *window_avg_velocity*
- *mass_flow*
- *time_avg_mass_flow*
- *window_avg_mass_flow*
- *normal_velocity*
- *time_avg_normal_velocity*
- *window_avg_normal_velocity*

<summary_output>

<output_frequency>
Specifies the number of coarsest iterations between consecutive outputs.

<output_start_iteration>
Specifies when to start the summary output (coarsest iteration) for each instance. The default value is 0.

<output_variables>
In this section, you can specify which variables should be included in the summary output by setting *<variable_name>* to true or false. The possible options include:

- *pressure*
- *time_avg_pressure*
- *window_avg_pressure*
- *mass_flow*
- *time_avg_mass_flow*

- *window_avg_mass_flow*
- *normal_velocity*
- *time_avg_normal_velocity*
- *window_avg_normal_velocity*

1.4 Expert Parameters

Additional expert parameters can be used to fine-tune simulations.

When these parameters are not specified, established default values are applied and the expert parameters do not appear in the effective solver deck written by the solver.

When you specify values for the expert parameters, they are used instead of the default values. The parameters are also included in the effective solver deck written to disk by ultraFluidX.

1.4.1 <simulation>

<simulation> - <general> - <num_ramp_up_iterations>

The inlet velocity is increased from zero to the value requested by *<reference_velocity>* over a certain amount of ramp-up iterations and physical time. By default, the number of ramp-up iterations is computed to achieve an average acceleration of 200 m/s^2. For a case with reference velocity 40 m/s, the ramp-up phase covers the first 0.2 s of the simulation, and the number of ramp-up iterations is calculated accordingly.
Alternatively, you can explicitly specify a fixed number of coarsest iterations for the ramp up of a simulation.
Default = 0

<simulation> - <general> - <convergence_detection>

Activates the ultraFluidX convergence detection algorithm output (still in beta mode). The algorithm analyzes the transient drag and lift signals and estimates if and when the initial transient phase has ended and when the overall simulation has converged; the respective iteration steps are reported in the `uFX_summary.txt` file, together with averaged force coefficients that are determined between these two points in time.

<simulation> - <geometry> - <triangle_mesh_export>

true The triangulated surface used in the code (after triangle_splitting, if applicable) is exported to `uFX_meshData/uFX_surfaceMesh.stl`.

This surface mesh contains all "physical" surfaces including passive parts, while surfaces needed for the mesh setup, for example, those defining sources or custom zones, are not displayed.

false (default)

<simulation> - <wall_modeling>

<wall_model_intensity>

The wall model intensity parameter is used together with the *<coupling>* "one-way" option and determines the strength of the link between the velocity taken as input to the wall model and the information fed to the first near wall cell. For a value of 0.0, the wall model is effectively switched off and a standard no-slip boundary condition is applied.

<donor_interpolation>

The donor position is the point where the wall model is coupled to the flow field, in a sufficiently well resolved part of the boundary layer. This donor position does not necessarily coincide with a voxel center; the information required at this position will be interpolated/ derived from the data available in the 27 voxels (3x3x3) around the donor position.

<donor_position_factor>

This factor is applied to a reference length (see below) to determine the exact distance to the wall where the wall model is extracting information from the flow. The default value of 3.5 together with the default reference length corresponding to the near wall voxel size leads to a setup where the donor position is placed in the third layer of voxels above the wall.

<donor_position_use_local_reference_length>

The reference length is coupled to the voxel size at the wall. This parameter controls if a local reference length per refinement level or a global reference length is used.
If set to true (default), the reference length is evaluated per refinement level, which moves the donor position closer to the wall in regions of finer grid.
If set to false, in case the wall touches voxels of different refinement levels, the coarsest voxel size at the wall will be used as a reference length, also in regions of finer grid.

<donor_position_reference_length>

Using this parameter, you can also set the reference length to a specific value in $[\mathrm{m}]$, which should, however, be of the order of magnitude of the near wall voxel size. If set to 0.0 (default), the parameter is effectively switched off.

> **Note:** This parameter is only effective in the case that you request a global reference length at the same time (see above). Otherwise, it will be ignored.

<simulation> - <initialization> - <seeding>

To accelerate convergence, you can initialize ("seed") the ultraFluidX simulation with velocity information from a previous run or from fallback velocity. Standard ultraFluidX output files (*.`geo` and *.`data`) with transient or time-averaged velocity information can be used for the initialization.
The following parameters are available for this feature:

<method>

(String)

velocity	Initialization with velocity field.
uniform	Uniform initialization with *<fallback_velocity>*, no ramp-up.
off (default, fallback)	Uniform initialization with 0.0, subsequent ramp-up to the *<reference_velocity>*.

<scale_factor>

(Double) A user-defined factor with which the initialization velocity from the EnSight file can be scaled in the entire domain. For example, you can rescale a velocity field based on an inlet velocity of 20 m/s to initialize a simulation with inlet velocity 25 m/s using a *<scale_factor>* of 1.25.
The scale factor is only applied to the Ensight initialization values, not to the *<fallback_velocity>*.
Default = 1.0

<fallback_velocity>

<x_dir>, <y_dir>, <z_dir>

(3x double) The velocity used for all voxels without valid initialization data in the EnSight data sets specified in the *<source_files>* category. This velocity is also used for the initialization if the *<method>* is set to uniform. The default for the *<fallback_velocity>* is the reference velocity of the simulation in x-direction and zero in y- and z-direction.
Default = Value of *<simulation> <general> <reference_velocity>* (X), 0.0 (Y), 0.0 (Z)

<source_files>

<default>

<file_set>

<geometry>

(String)

<data>

(String)

You can specify one or more sets of files. Each file includes one *.geo and one *.data file that must be specified via the *<geometry>* and *<data>* tags. The files must contain the same number of elements and the *.data file must contain a three-dimensional velocity field.

<overset>

<file_set>

<geometry>

(String)

<data>

(String)

In case of an overset simulation, the files defined in the *<default>* category are used for initialization of the background mesh. The files defined in the <overset> category are used for initialization of the foreground mesh.
If the source and target overset mesh voxels are not fully compatible due to a mismatch in the angle of rotation, the *<fallback_velocity>* is applied for the affected voxels.
In case of non-overset simulations, only the *<default>* files are used.

1.4.2 <meshing>

<meshing> - <general> - <seeding_point>

<x_pos>, <y_pos>, <z_pos>

The seeding point is the position in the domain from with the flow region is filled with "fluid" voxels. By default, this position is placed at one of the corners between the inlet plane and the ground.
If the geometric setup is such that a solid is placed at this position, you can indicate another arbitrary point (inside the fluid region) from which the process of filling the fluid region with voxels should start.

<meshing> - <overset> - <rotating>

<uniform_refinement>

(Bool) Global switch that controls the behavior of the overset mesh refinement.

true (default)	OSM regions are automatically refined to the finest refinement level present in each region.
false	Multiple refinement levels in each overset grid region are tolerated.

The interface region that connects the overset mesh region to the background mesh must be uniformly refined in any case.
Several features, such as global averaging and globally positioned probes, are not yet supported for multi-resolution overset mesh simulations. They are automatically deactivated.

<automatic_interface_refinement>

(Bool) Switch that controls the automatic refinement of the interface regions.

true	The interface regions are automatically refined to the finest refinement level present in each region. If set to true, *<uniform_refinement>* must be set to false.
false (default)	Refinement at the interface regions is not automatically performed.

<min_interface_refinement_level>

(Optional) Desired refinement level at the interface region with regards to the coarsest level. If not specified, the highest refinement level found in the interface region is used.

<rotating_instance>

<uniform_refinement>

(Bool) Local switch available in each overset mesh zone that overrides the global refinement behavior specified by *<uniform_refinement>*.

<automatic_interface_refinement>

(Bool) Local switch available in each interface region that overrides the global refinement behavior specified by *<automatic_interface_refinement>*.

<min_interface_refinement_level>
Desired local refinement level in each interface region that overrides the global refinement behavior specified by *<min_interface_refinement_level>*.

Index

Special Characters

<boundary_conditions> *23*
<boundary_conditions> - <belt> *24*
<boundary_conditions> - <periodicity> *24*
<boundary_conditions> - <rotating> *23*
<boundary_conditions> - <static> *24*
<meshing> *20*
<meshing> - <general> *20*
<meshing> - <overset> *22*
<meshing> - <refinement> *21*
<meshing>, expert parameters *42*
<output> *28*
<output> - <aero_coefficients> *31*
<output> - <general> *29*
<output> - <moment_reference_system> *30*
<output> - <monitoring_surface> *36*
<output> - <partial_surface> *33*
<output> - <partial_volume> *33*
<output> - <probe_file> *35*
<output> - <section_cut> *32*
<output> - <single_probe> *34*
<simulation> *15*
<simulation> - <general> *15*
<simulation> - <geometry> *16*
<simulation> - <wall_model_variant> *19*
<simulation> - <wall_modeling> *18*
<simulation>, expert parameters *39*
<sources> *25*
<sources> - <acoustic> *28*
<sources> - <mrf> *27*
<sources> - <porous> *25*
<sources> - <turbulence> *26*

C

command line options *11*

E

expert parameters *39*

M

multi-node usage *13*

O

overview *10*

S

simulation parameters *15*

U

using ultraFluidX *11*

W

workflow and output *12*