|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **GRUNDFOS SIMULATION PROCEDURE** | | | | |  |
| Procedure ID  **CFD-0004** | SRL  **2** | | Procedure name  **Setup guide for sensitivity study of hydraulic components** | | |
| Release date  **2019-07-03** | | | Last update  **2019-07-03** | | |
| Created by  **GMATIR and GMAMINO** | | Reviewed by  **GMANSP, GMABMJ, GMAMSHA, GMALAUH and GMAPRA** | | Experienced users:  **GMATIR and GMAMINO** | |
| Example of study:  A close up of text on a black background  Description generated with high confidence | | | SRL details: | | |
| **PROCEDURE DATA** | | | | | |
| **DESCRIPTION OF PROCEDURE**  This simulation procedure is for sensitivity studies of hydraulic components. It is intended to feed sensitivity information into the development phase of a project and support the tolerance setting. | | | | | |

# Change history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Comments | Modified by | Reviewed by |
| 2019-07-02 | Simulations procedure documented and reviewed | GMATIR | GMANSP, GMABMJ, GMAMSHA, GMALAUH and GMAPRA |
|  |  |  |  |

Contents

[1 Change history 2](#_Toc13052397)

[4 Introduction 4](#_Toc13052398)

[4.1 Purpose 4](#_Toc13052399)

[4.2 Requirements 4](#_Toc13052400)

[4.3 Tool, macros and templates 4](#_Toc13052401)

[5 Simulation Procedure 4](#_Toc13052402)

[5.1 Structure and workflow 4](#_Toc13052403)

[5.2 Initial CAD work 5](#_Toc13052404)

[5.3 Setting up solver call for OptiSlang 10](#_Toc13052405)

[5.4 Setting up sensitivity study in OptiSlang 22](#_Toc13052406)

[5.5 Post-processing of OptiSlang results 28](#_Toc13052407)

# Introduction

## Purpose

To enable all hydraulic engineers to perform a sensitivity study for hydraulic components. For inspiration, the conference presentation from WOST 2019 about the work can be accessed in the SharePoint folder for this simulation procedure. The guide is a part of the outcome of HunT2.0.

## Requirements

* A working CFX simulation setup of the nominal geometry of interest.
* CAT-part containing the surfaces used to create the mesh of the component of interest.
* Access to euler and SVN.

## Tool, macros and templates

All tool and templates for the following can be found on the SVN: SVN\SimulationScripts\DesignTools\Sensitivity\SensitivityStudy\ , see Figure 1.

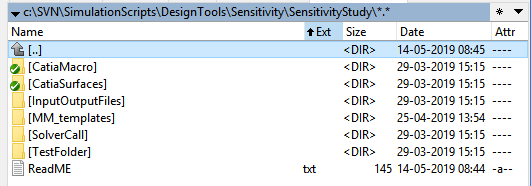


Figure 1: Screenshot of sensitivity SVN folder.

# Simulation Procedure

## Structure and workflow

The sensitivity study builds on perturbation of the nominal geometry using the mesh morpher in CFX. The mesh morphing feature in CFX does not have a user-friendly way of specifying the deformation points. The feature is designed for fluid-structure interaction, where the deformation input comes from a structural analysis. This is not the case when the morpher is used for perturbation for sensitivity studies. The deformation input needs to be created outside of CFX. Since many of the meshes at Grundfos are created with Catia surfaces as basis this is an obvious common starting point. Catia is, however, not robust to call in batch, which is needed when creating sensitivity studies with OptiSlang. This creates the need for only a one-time interaction with Catia prior to perturbation/sensitivity study. This problem is solved with the workflow in Figure 2.

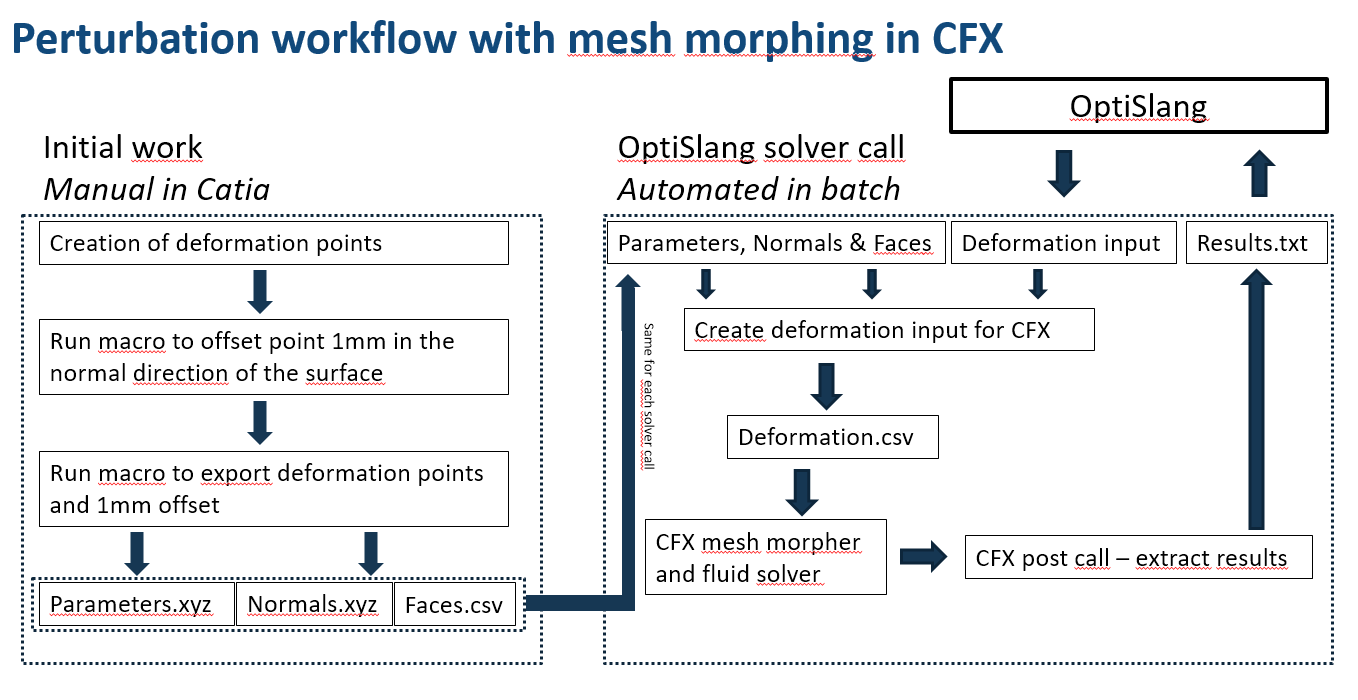


Figure 2: Perturbation workflow with mesh morphing in CFX.w

The following sections will go in details with the different steps in Figure 2.

## Initial CAD work

Start by copying the content of SVN\SimulationScripts\DesignTools\Sensitivity\SensitivityStudy\ to the folder on Euler where the sensitivity study is to be performed.

MD(**M**echanical **D**evelopment, POD, BD) macros for Catia should be installed. If this is not done, then follow the instructions by GMAKDS in Figure 3.

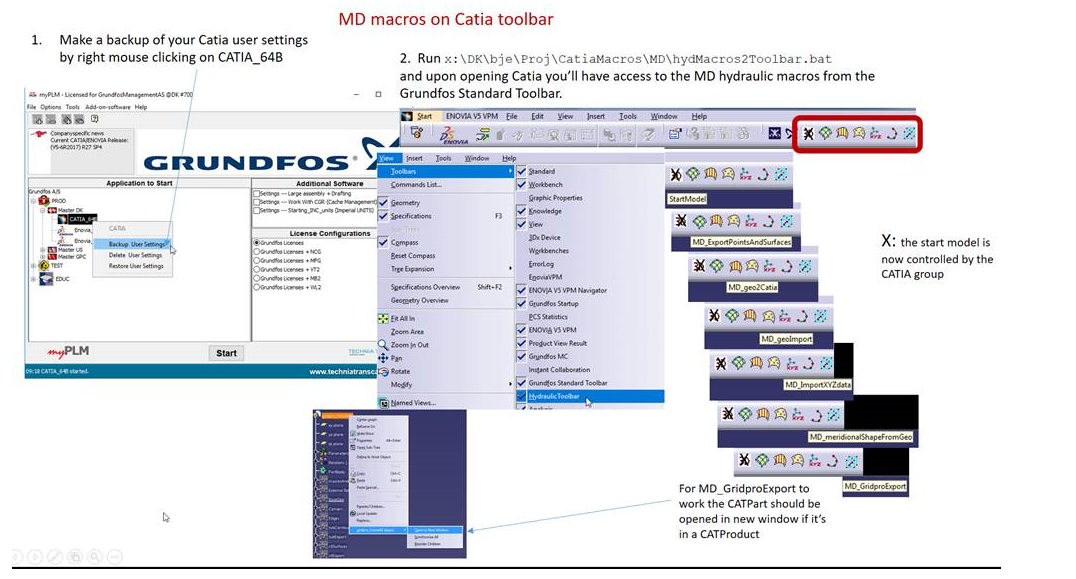


Figure 3: How to install MD macros

Create a Geometrical set named *“xyzExport”*. In this geometrical set, create a new geometrical set named *“Parameters”* . The first object in Parameters should be the surface where the deformation input is needed and it should be the exact same surface as is used in the mesh creation. The surface should be an Inverse named *“Inverse.1”*. The Inverse will later make it easier to control/change the direction on the surface.

If there is a sharp bend on the surface for example the rounded leading edge of a blade, then the blade should be separated in to a suction and pressure side with a separation line at the tip of the leading edge. This will prevent fault mapping of the deformation in CFX. In the case of an impeller the deformation should be specified individually on both the pressure and suction side.

Now Parameters should be marked as define work in object and point can be placed on Inverse.1. This will be the points where the deformation will be defined in CFX. It is advantageous to not place the points too close, as this will create too many parameters in OptiSLang and make it difficult to see tendencies.

In this case 26 points are places on the pressure side of the blade, see Figure 4.

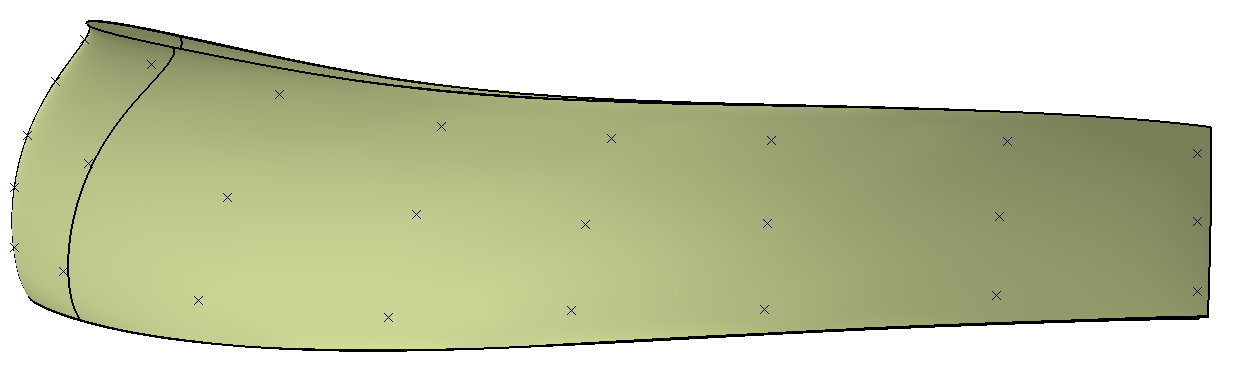


Figure 4: Placement of deformation point on the pressure side.

Two extra points are placed on the leading edge to make a smoother mesh morphing in this zone with high aspect ratio cells. Figure 5 shows an example of the tree structure in Catia. The names of the points in Parameters is not important.

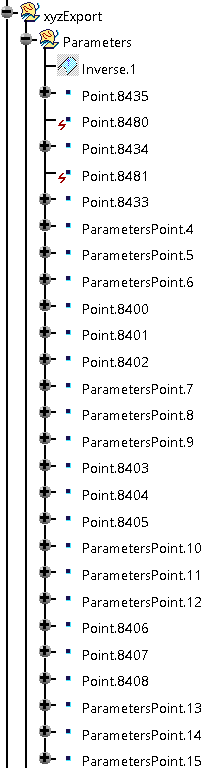


Figure 5: Structure of surface and point in geometrical sets.

Now the Deform macro in Catia can be used, see Figure 6. This is located in SVN\SimulationScripts\DesignTools\Sensitivity\SensitivityStudy\CatiaMacro\CreateNormalPoints.catvba.

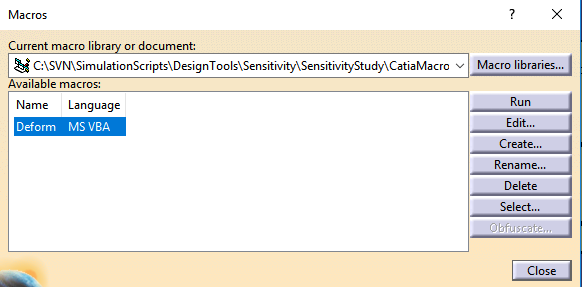


Figure 6: Execution of deform macro.

This macro is creating another geometrical set inside xyzExport named Normals. In this geometrical set 1mm lines normal to the surface are created together with end-points.

It should be noted that this is not desirable on the leading edge since it will prevent any angular change of the leading edge. The normal points on the leading edge are consequently changed to allow for movement normal to the chamber surface also on the leading edge. Note that the order of the points in the geometrical set matters. The order of the points in Parameters should match with the order of points in Normals.

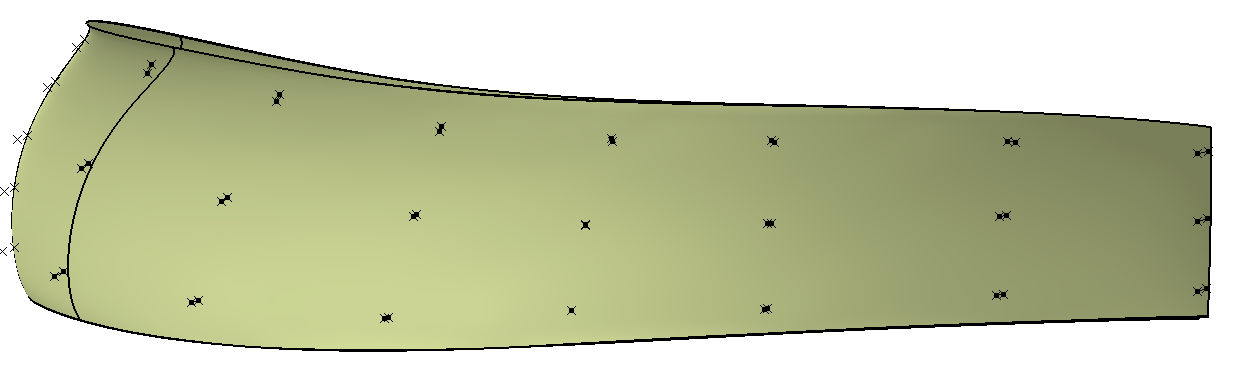


Figure 7: Deformation input together with point 1mm in the movement direction.

Now MD\_ExportPointsAndSurfaces macro can be executed. Press “no” to reopen part and enter the path to MM\_templates.

Now Normals.xyz and Parameters.xyz are written to the folder.

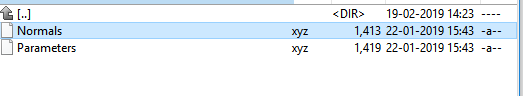


Figure 8: Output from macro execution.

In the current case the previous steps are repeated for the suction side of the blade. But this would not necessarily be the case for other geometries. Examples of the CAT parts can be seen in SVN\SimulationScripts\DesignTools\Sensitivity\SensitivityStudy\CatiaSurfaces\

The face connection needs now to be specified, this make a smoother mapping of the deformation in CFX.



Figure 9: Deformation input points and their numbering.

By looking at Figure 9, faces.csv can be created as seen in Figure 10 . This adds the information of neighboring deformation points. This creates a much smoother deformation.

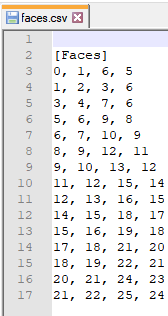


Figure 10: Faces.csv created from figure Figure 9.

The above operations should be repeated for the suction side of the blade as well.

Now all the initial work shown in the left side of Figure 2 is done.

## Setting up solver call for OptiSlang

The following section shows the steps needed to setup the solver call for OptiSlang.

The basic idea is that all needed inputs are in MM\_templates and OptiSlang only write a single input file containing the wanted deformation at the input points.

For the sake of trouble shooting the setup a DeformationInput.txt should be placed in PS folder. This is containing the wanted deformation that OptiSlang will write later on. The file should contain as many points as specified in the parameters file. An example can be seen in Figure 11.

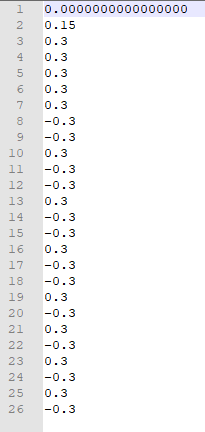


Figure 11: Example of deformation input.

MM\_templates contains a folder named Macros. This folder has two scripts that is executed RunSetup.pre and post.cse. RunSetup.pre is a pre call that will setup the simulation, map the deformation on the simulation mesh and write the new def file. Post.cse is a post call that will extract results and pictures from the simulation result. These files need minor changes when a new setup is done. But before doing this a tested def file with the nominal geometry should be copied to the MM\_templates folder, this def file should be named Deform.def. The rest of this section will cover how to setup the pre and post call.

In SVN\SimulationScripts\DesignTools\Sensitivity\SensitivityStudy\TestFolder\ a script called SolverCall\_.py can be found.

This is a simplified version of the python solver call needed for OptiSlang and is to be used in the setup phase for troubleshooting and testing. The following is a walk through the code and includes explanations of what is needed to be set up for each step. Furthermore, the comments in the code should be used as help.

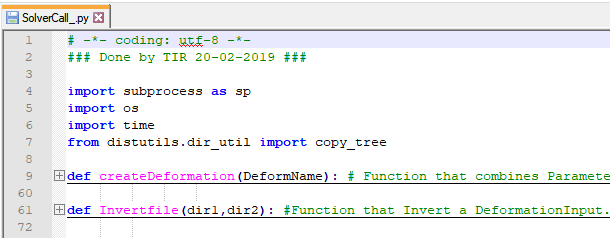


Figure 12: First part of batch solver call.

The first 72 lines of code contains two functions; createDeformation and Invertfile. CreateDeformation combines Parameters.xyz, Normals.xyz and DeformationInput.txt in to a readable deformation for CFX.

Invertfile is designed for sheet metal, where a constant plate thickness is needed. The function requires the deformation only to be created on one side of the blade, and will then invert the deformation to the other side of the blade to keep a constant blade thickness.

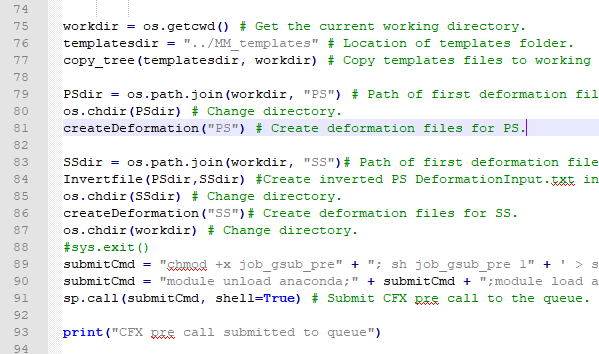


Figure 13: Middle part of batch solver call.

From line 74 to 77 all the content of MM\_templates is copied to the execution folder. From line 78 to 82 the createDeformation function is called for the pressure side of the blade(PS). The same is done from line 82 to 86 for the suction side except that the Invertfile also is called, to secure that the blade thickness is kept constant. Line 88 contains an argument to only run the above part of the code if the hash is removed. This makes sense when setting up the case, since all pre and post calls are for the default case and not fitted to the current. From line 89 to 91 a pre call to CFX is done. This is done by submitting the job\_gsub\_pre to the queue system. This makes a batch call to CFX pre with the Macros/RunSetup.pre

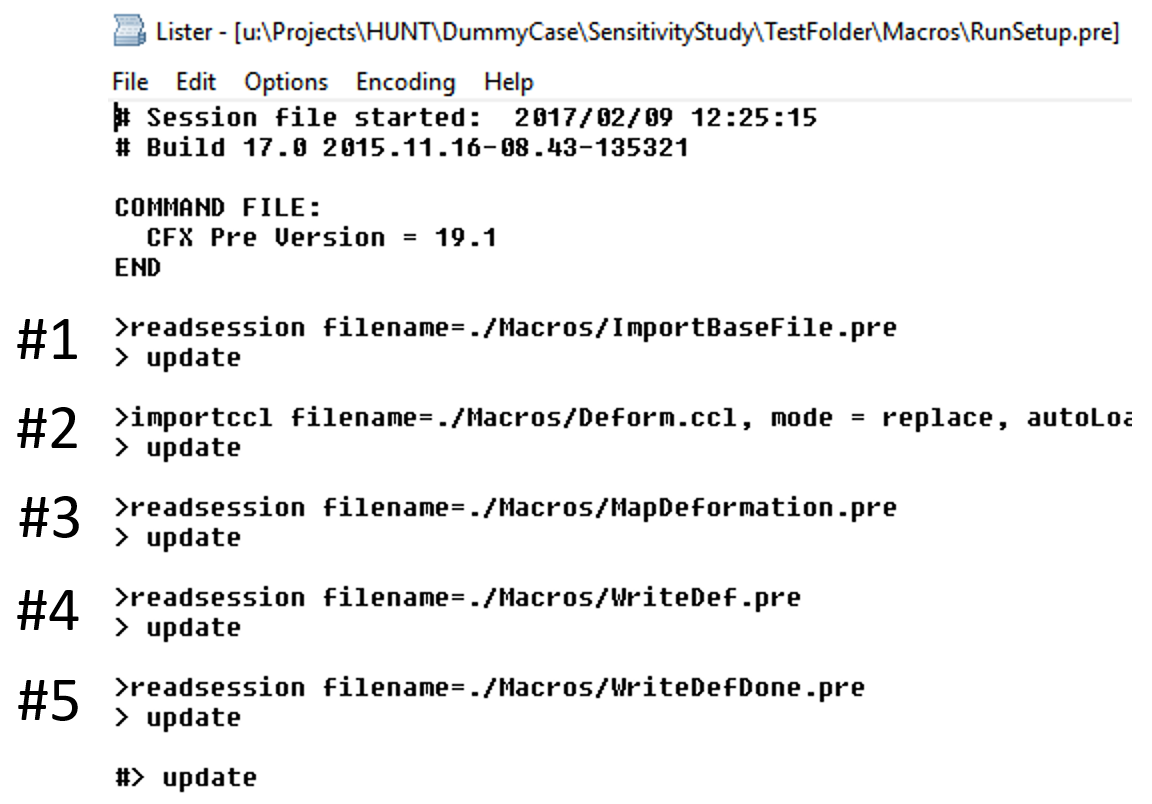


Figure 14: CFX pre setup script.

Step 1 in this file imports the original def file. Step 2 appends the setup to the deformation setup with relative paths to deformation files. Step 3 is mapping the predefined deformation on to the correct mesh surface in the CFX setup for smoother deformation. Step 4 writes the new def file to be deformed. Step 5 writes dummy file to let the setup know that the def file is finished and ready to be submitted to the cluster.

When setting up a new setup only step 2 and 3 need to be changed.

Step 2 is changed by first removing # in line 88 of SolverCall\_.py and then run the script. This will only run the first part of the script and the deformation input is created. Remember to do “module load anaconda” and “module load cfx” in the terminal before executing “python SolverCall\_.py”Now the def file should be opened in CFX pre.

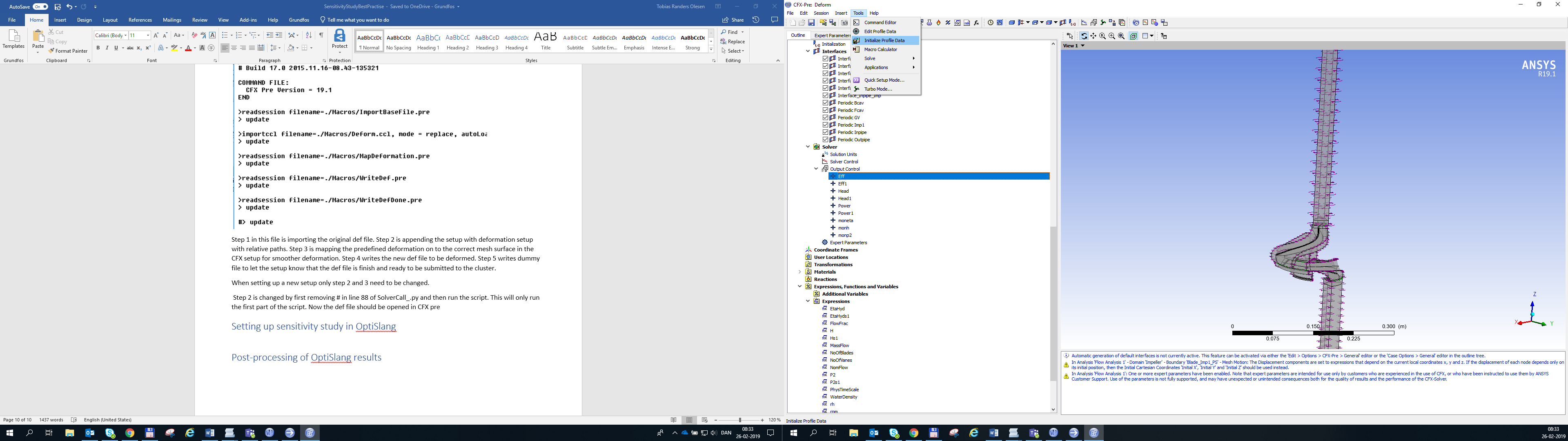


Figure 15: Loading of deformation input.

The deformation input can now be loaded into CFX by going to Tools -> Initialize Profile Data and selecting DeformationPS.csv. This is the deformation input for the PS of the blade. Do the same for the SS. They could have other names when doing other cases. Now the deformation can be seen under User Functions with the name specified in the csv file.

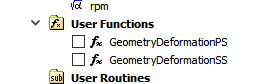


Figure 16: Deformation input

Now the deformation details can be setup. In the domain to be deformed (in this case impeller) the following needs to be applied in basic settings.

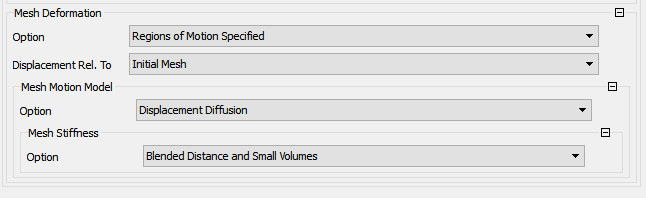


Figure 17: Specify mesh deformation in domain settings.

Now each of the boundary conditions need to have mesh deformation details specified. This is done in boundary details where the following three options can be selected:

* **Specified Displacement:** Displacement is specified in the input file.
* **Unspecified:** Soft constraint. Mesh can be moved to make overall nice mesh, but no displacement specified.
* **Stationary:** Hard constraint. No modifications to mesh are done.

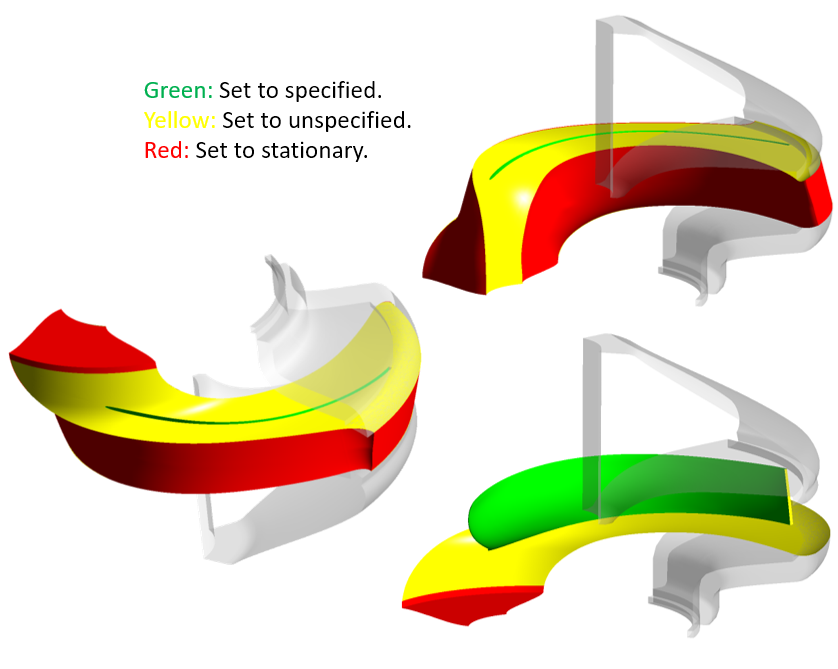


Figure 18: Deformation details for the sheet metal impeller example.

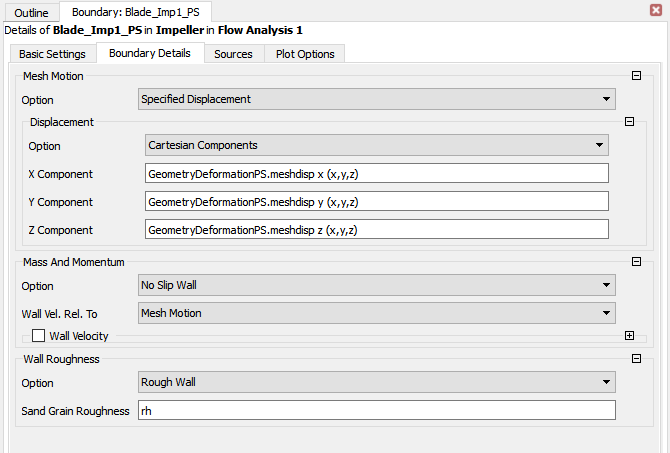
The mesh surfaces with specified displacement needs to be defined the following way: 

Figure 19: Mesh motion specification for blade PS.

The mesh surfaces with unspecified displacement needs to be defined in the following way:

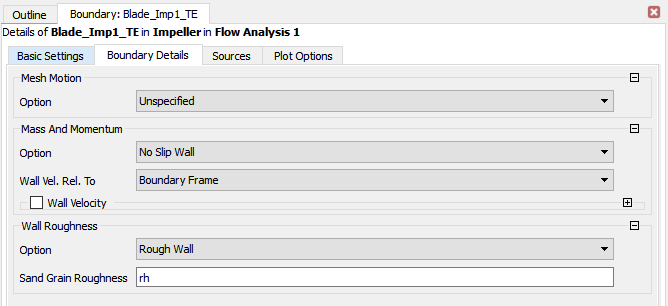


Figure 20: Mesh motion specification for the impeller trailing edge.

Stationary mesh morphing conditions are setup in an similar way.

After the mesh details for the deformation has been setup for all the surfaces a ccl of the setup is saved in MM\_templates/Macro/Deform.ccl

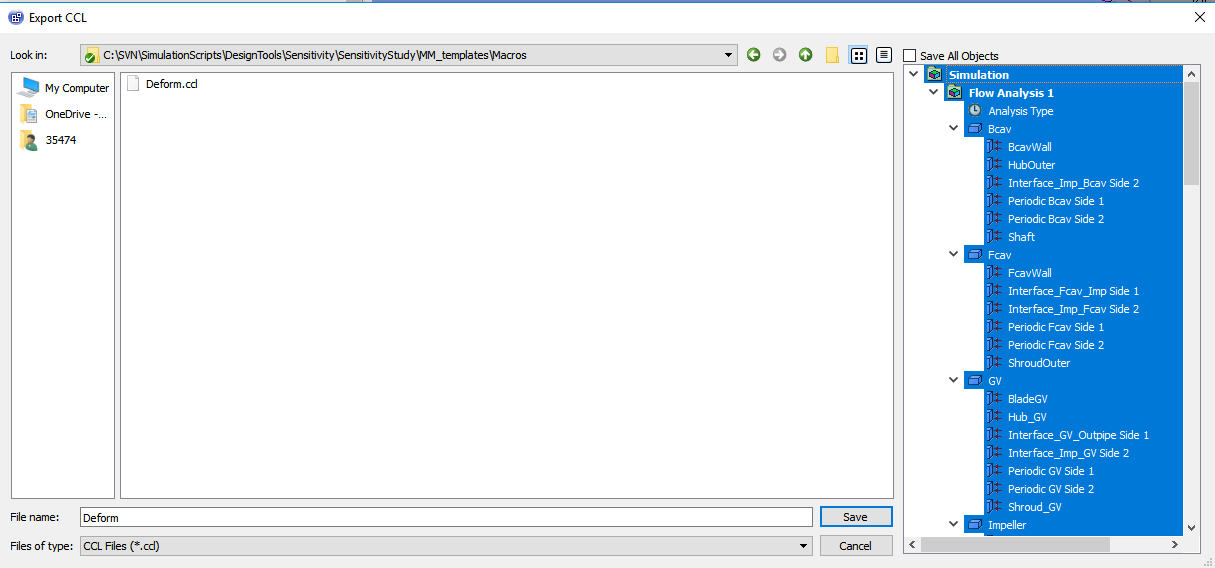


Figure 21: CCL export in CFX.

Afterwards the ccl needs to be modified so the deformation files are defined with relative paths, see Figure 22.

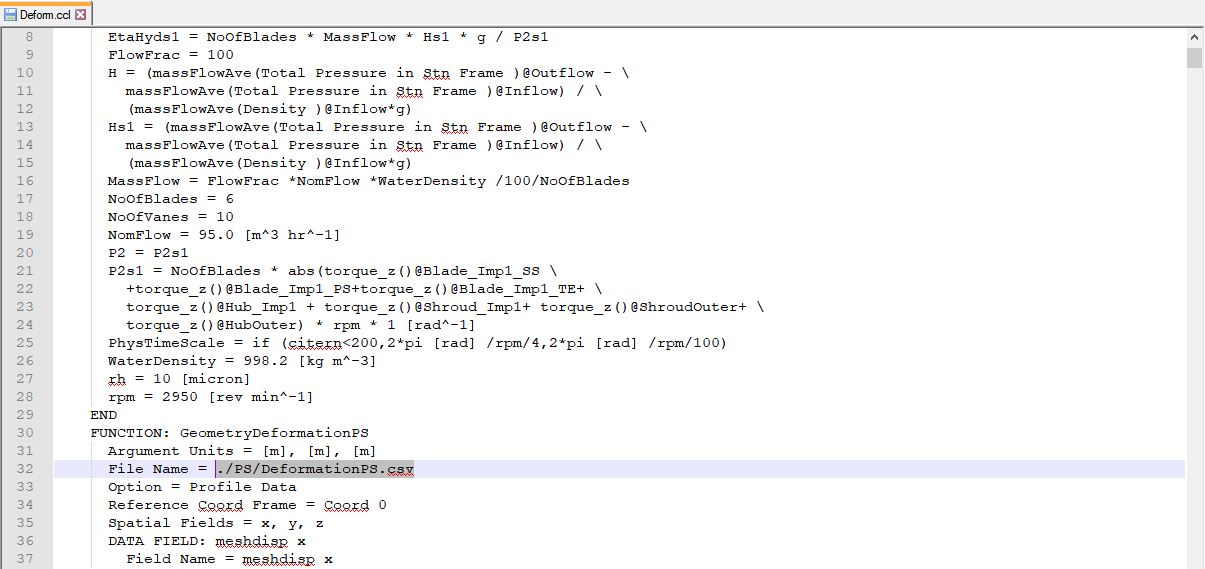
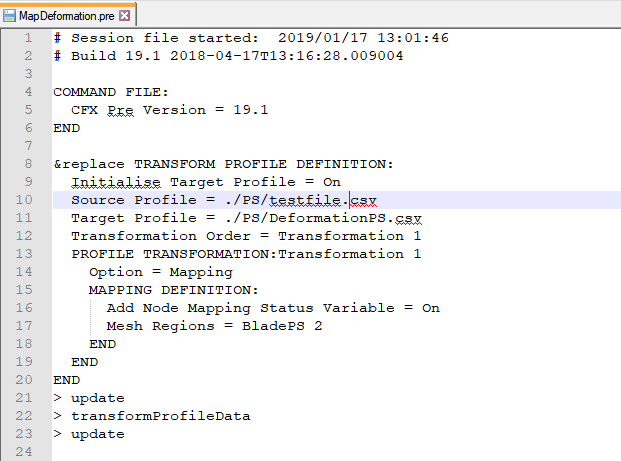
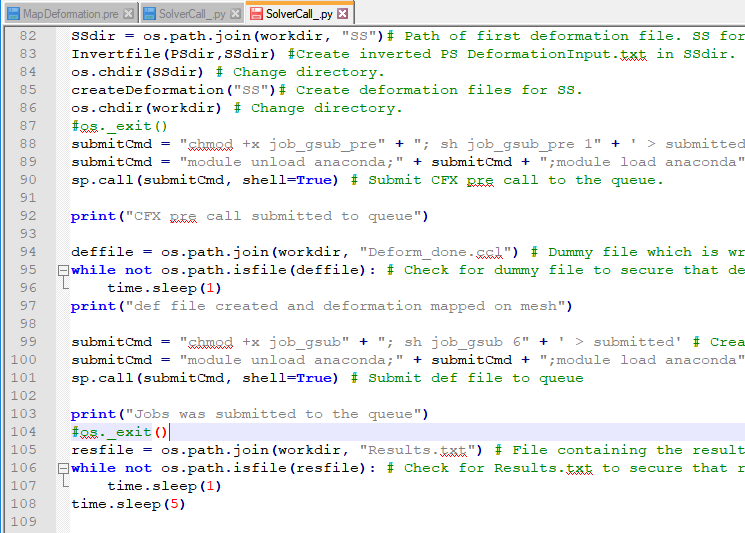


Figure 22: Relative paths in ccl.

Now the MapDeformation.pre can be setup. This file can just be directly edited. Change the mesh regions so it matches the new case.

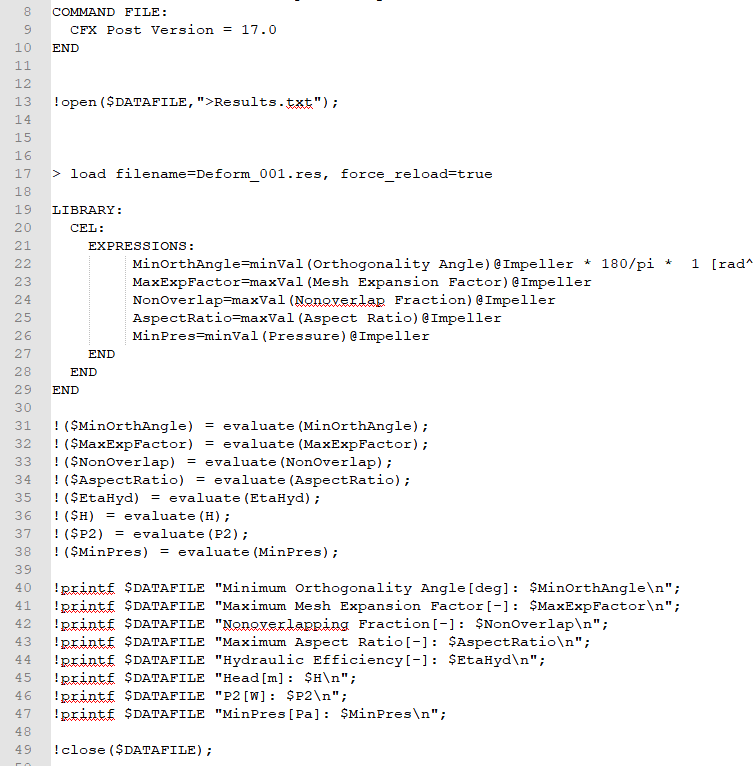


Now the case can be solved for both deformation and fluid part. This is done by in SolverCall\_.py adding # in line 87 in front of os.exit() and removing the # in line 104. This will stop the solver call after the job is submitted to the cluster and make it possible to create the postprocessing.

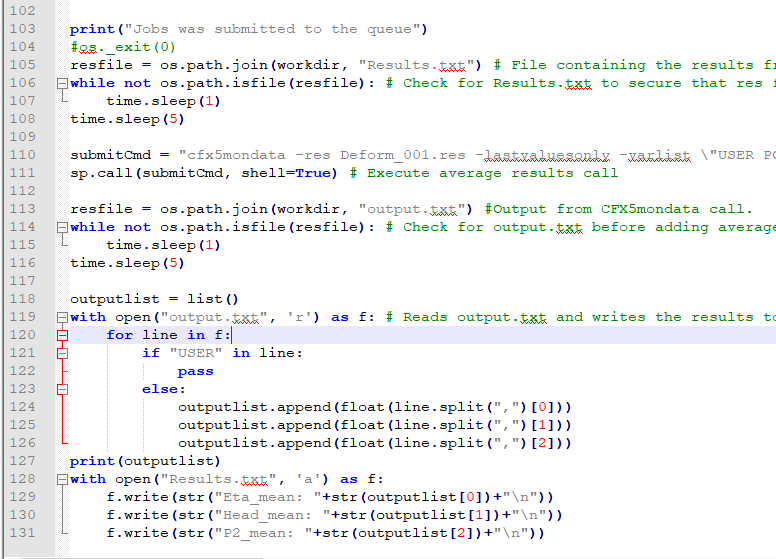


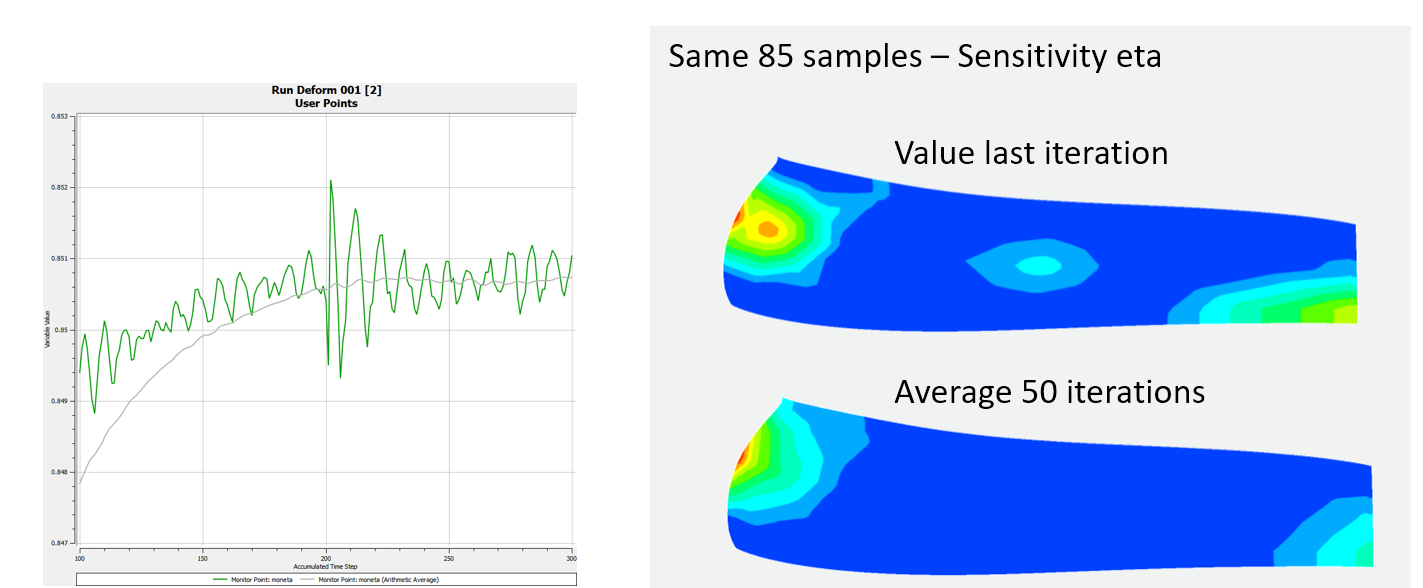
Between line 99 and 101 the job file is submitted to the cluster queue. This job file is both containing the submission of the def file and the post call for results extraction from the CFD.

The postcall can be found in MM\_templates/Macros/post.cse



From line 13 to line 49 results are extracted to Results.txt. This should be modified so it fits the new case. From line 51 different pictures are saved in the folder for visual inspection and troubleshooting. These are done by loading the .res file in to cfx post and creating desired plot. Then the state needs to be saved(.cst). This can then be loaded in later when the plots are to be saved in batch.

The last part of the solver call(from 105 to 131) calculates the average values of the integral values. This is very useful if the integral values are fluctuating. This will remove the noise going in to OptiSlang. If these are needed for the new case the monitor point in the test case can be used as inspiration.



If there is no or minimal fluctuation in the integral values this can be ignored/commented out.

Modify /SolverCall/SolverCall.py so the changes matches SolverCall\_.py.

Now the solver call should be able to run with no intermediate stops and is ready to be implemented in OptiSlang.

## Setting up sensitivity study in OptiSlang

**Step 1:**Create a project directory with the following three subdirectories:

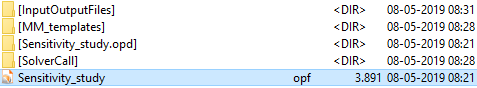
* InputOutputFiles
* MM\_Templates
* SolverCall

Each directory should contain the necessary files described in the previous steps of this guide.

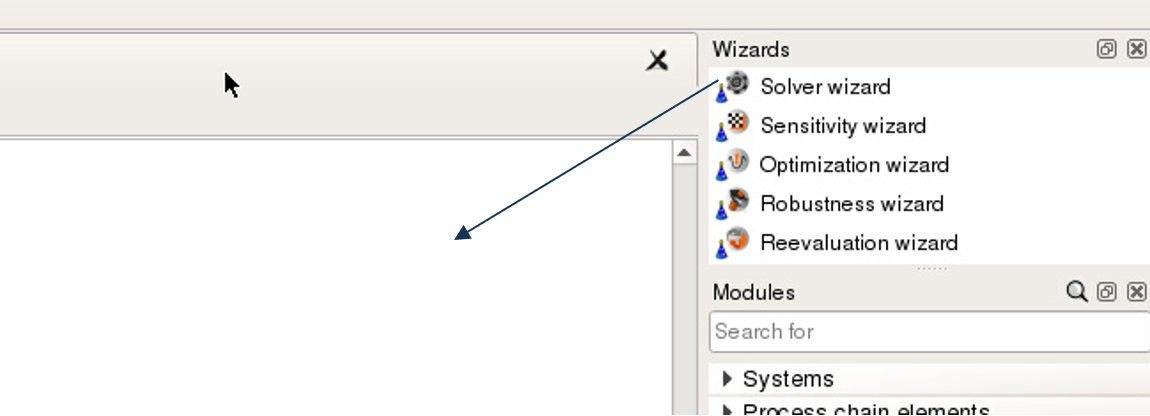
**Step 2:**Open a terminal on Linux and cd to the project directory. Open OptiSLang using the following command:



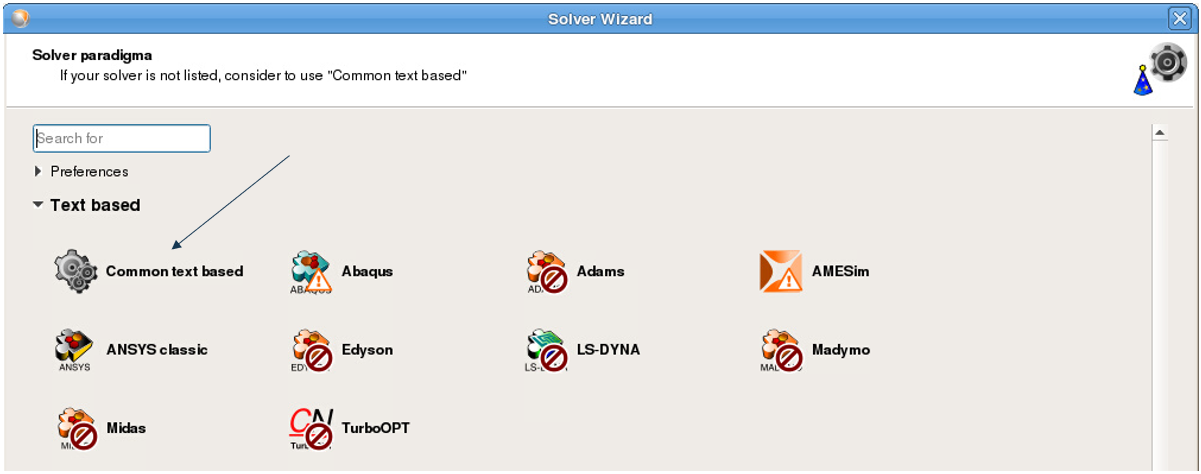
Then, create a new project in OptiSLang and save it in the project directory creating an OptiSLang project file and corresponding subdirectory. You should now have the following in the project directory:



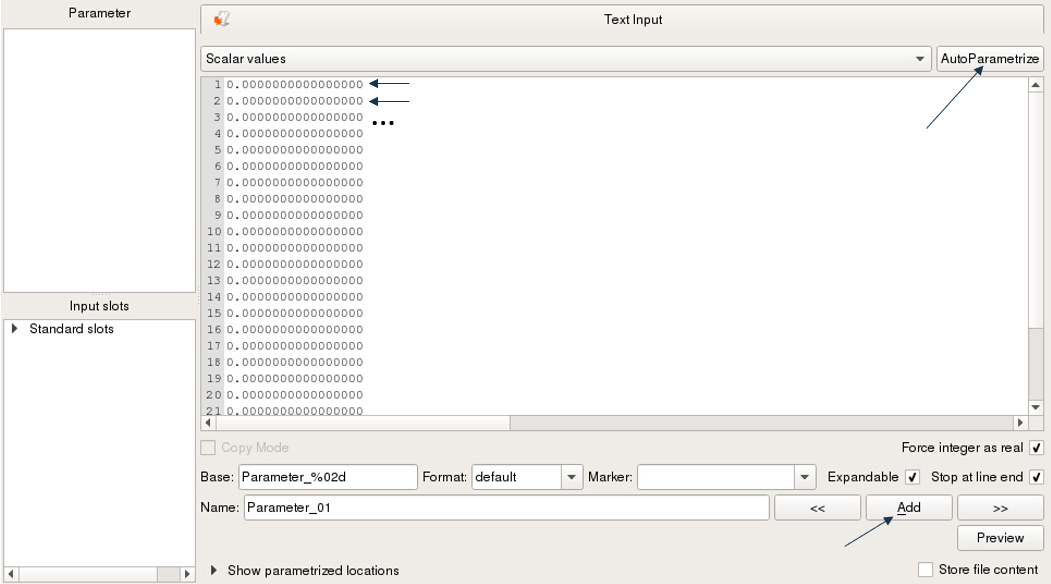
**Step 3:**The next step is to set up the solver call from OptiSlang. Do so by first dragging the “Solver wizard” into the work environment.



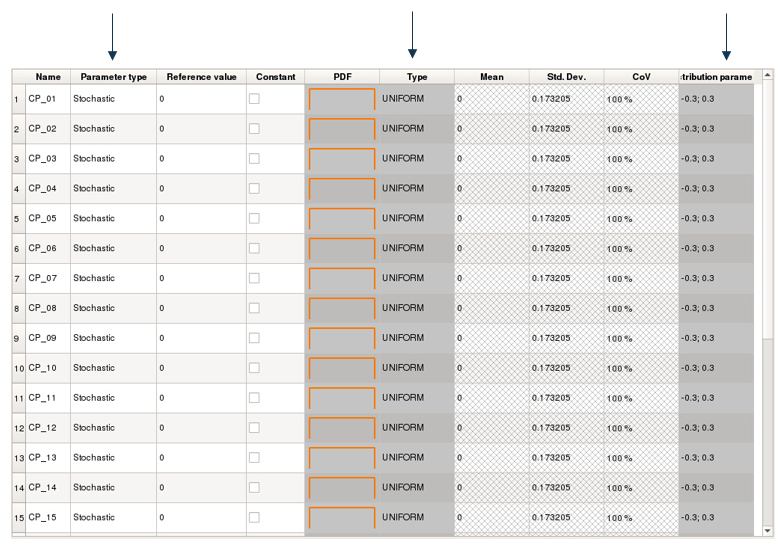
A prompt will appear asking you to select a solver type. Select the text-based solver:



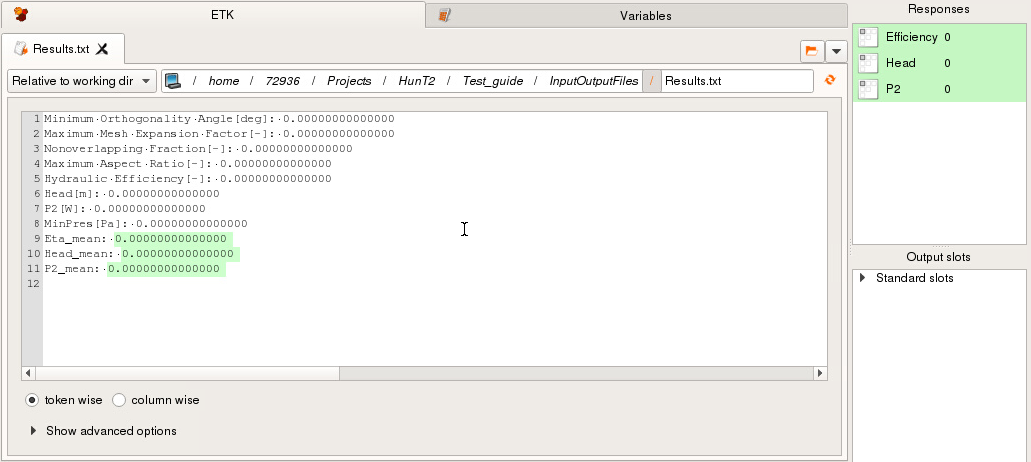
In the next window, navigate to the input file located in InputOutputFiles named “DeformationInput.txt”. Then, add each line as a new parameter by marking the value and clicking “add”. You can use the “AutoParametrize” button to ease this process.



Once finished, click next. The next window asks you to specify the properties of the input parameters. Here, specify the parameter type as stochastic, the type as uniform and the distribution range as desired. Other distribution types can be used in cases where the actual distribution of the input parameter is known.



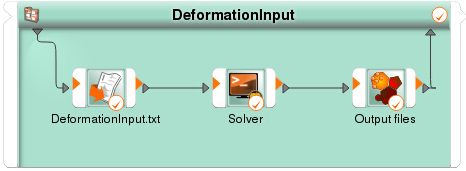
In the next window, you are asked to navigate to the output file. Similarly as before, you will also be asked to identify the responses. Do so in the same way is with the input files.



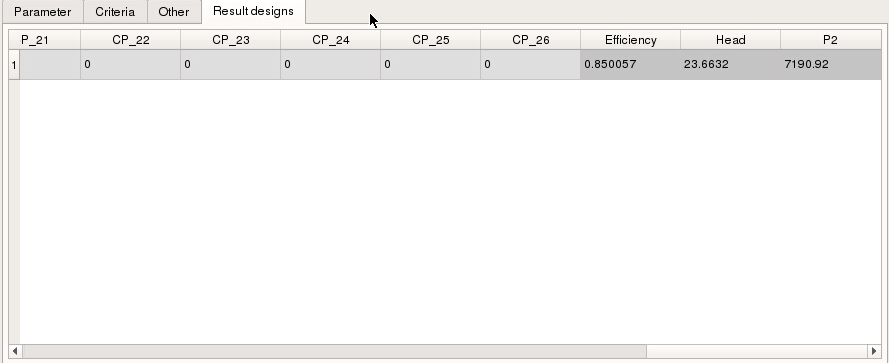
Clicking next will lead to the criteria window. Here, nothing needs to be specified as it is mainly relevant for optimizations. Click next and finish.

**Step 4:**Execute the project by clicking the play button.

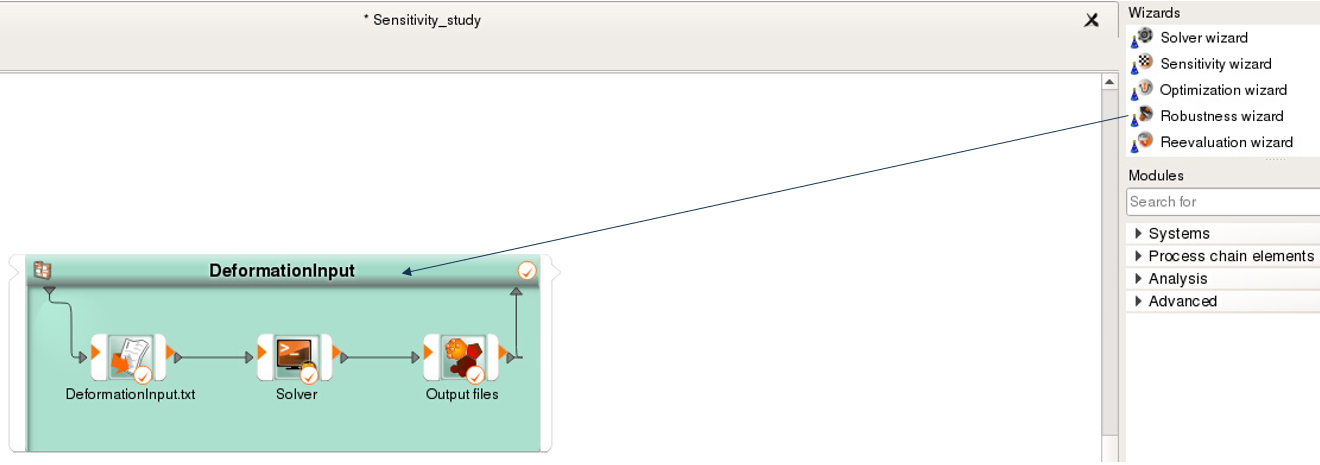
**Step 5:**If your project has completed successfully, it should be green as shown below.



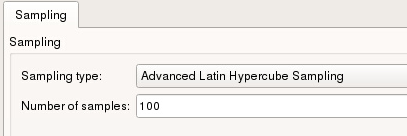
Checking that a result is actually obtained can be verified by double clicking anywhere in the green area except on any of the three small boxes. Scrolling to the end of the shown window should reveal the responses.



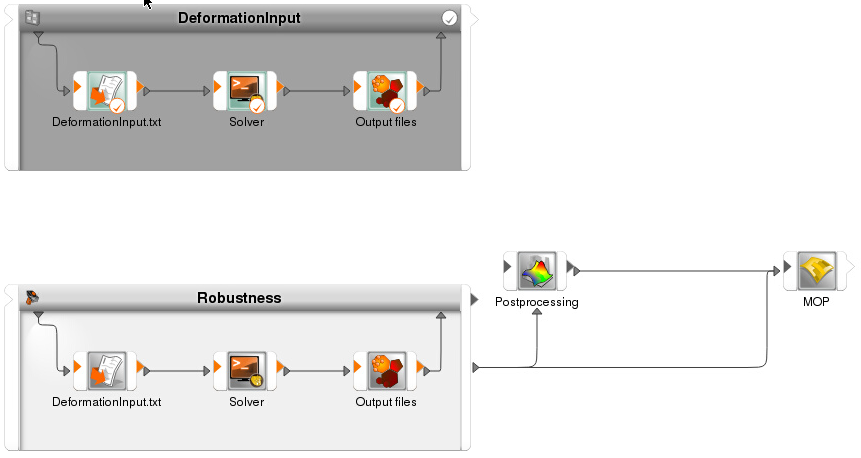
**Step 6:**Now drag the robustness wizard onto the system as shown below.



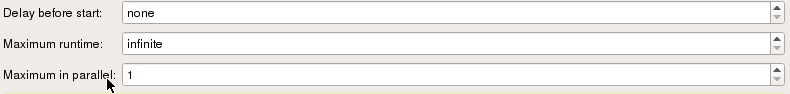
This will automatically transfer all settings you need, so just click next until you reach the window with the sampling settings. By default, the number of samples is 100, but it can be changed based on the number of inputs. It is recommended to have a sampling number of at least 2 times the number of input parameters.



Click next/finish until the window closes and a new system appears in the working environment.



**Step 7:**Double click the solver block and set the desired number of parallel runs.

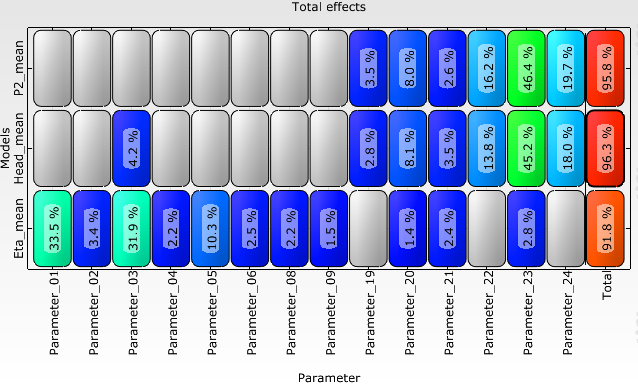


**Step 8:**

Execute the project.

## Post-processing of OptiSlang results

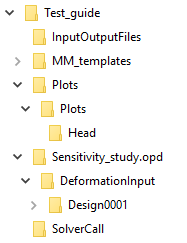
Once the robustness evaluation is done, the post-processing window will appear and create a meta model. Part of this window is the CoP matrix, which contains the information relevant to this study. An example of a CoP matrix is shown below.



The CoP shows which input parameters (control points) have the greatest impact on the difference responses. The horizontal axis contains the input parameters while the vertical axis shows the different responses. The “Total” value in the end shows how well the observed variation is explained by the sensitivities displayed in the matrix. For instance, for P2 it can be seen that the parameters 19 to 24 accounts for 95.8 % of the total observed variation.It can also be seen that parameter 23 has the greatest impact on the variation in P2. The following steps will show how to visualize this result by plotting the sensitivity on the surface of the investigated geometry.

**Step 1:**To make this process simpler, this guide will simply illustrate the sensitivities by mesh morphing the blade by values corresponding to the sensitivities. However, instead of using OptiSLang, this call will be made manually. The first step is therefore to create a directory with subdirectories corresponding to what occurs in OptiSLang.

To see how OptiSLang structures the project, go to the directory that contains the OptiSLang project, template files and so on. By unfolding the .opd directory, the underlying file structure can be seen. This should be replicated in a new directory such as the “Plots” directory shown below.

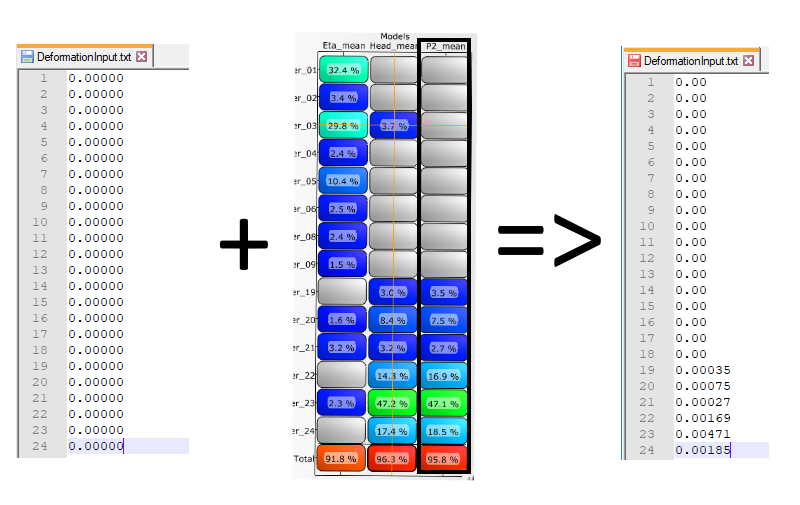


The “Head” directory in this case corresponds to a design directory in OptiSLang. At this level it is also possible to have other directories such as “P2”, “eta” and so on. They can be named anything.

Inside the “Head” directory, two files should be placed: The solver bash script and the input file as shown below.



**Step 2:**Open “DeformationInput.txt” in your favorite text editor. Then, edit the values so they correspond to the sensitivities shown in the CoP matrix. It is recommended that the sensitivities are divided by for example 10 or 100 to avoid aggressively deforming the blade mesh.



**Step 3:**Now open a terminal on linux and cd to the “Head” directory. Here, you can execute the solver call using the bash script.

**Step 4:**The sensitivity can now be shown by plotting the deformation of the mesh on the surface of the blade. Using scripting/sessions, pictures of this can be automatically generated with the solver call.