

FLUX: MAGNETIC ANALYSIS FOR ELECTRIC MOTORS

ALTAIR MULTIDISCIPLINARY DESIGN OPTIMIZATION PLATFORM
FOR ELECTRIC MOTORS

October 2021, Altair Flux / FluxMotor Valorization and Support Team

OBJECTIVES

Creating Flux magnetic analysis projects

- Base speed point simulation
- Specific speed point simulation

Motor performance analysis at base speed point

- Analyzing the motor performance (iron losses, magnet losses, efficiency, etc.)
- Creating HyperStudy connector for the MDO application

Motor performance analysis at specific speed point

- Analyzing the motor performance (iron losses, magnet losses, efficiency, etc.)
- Creating HyperStudy connector for the MDO application

OUTLINE

Creation of magnetic simulation scenarios

Magnetic analysis: base speed point

Magnetic analysis: specific operating point

Input file

- **Python script for model generation in Flux**
EMOTOR_Flux_Model_Generation.py

Software

- Altair Flux 2021.2.1 (or later versions)

Output files

- **Flux 2D projects**
 - EMOTOR_MAG_BASE_POINT.FLU
 - EMOTOR_MAG_SPECIFIC_POINT.FLU
- **Flux / HyperStudy connectors**
 - Connector_Fx_Hst_BasePoint.F2HST
 - Connector_Fx_Hst_OperatingPoint.F2HST

Automation script file

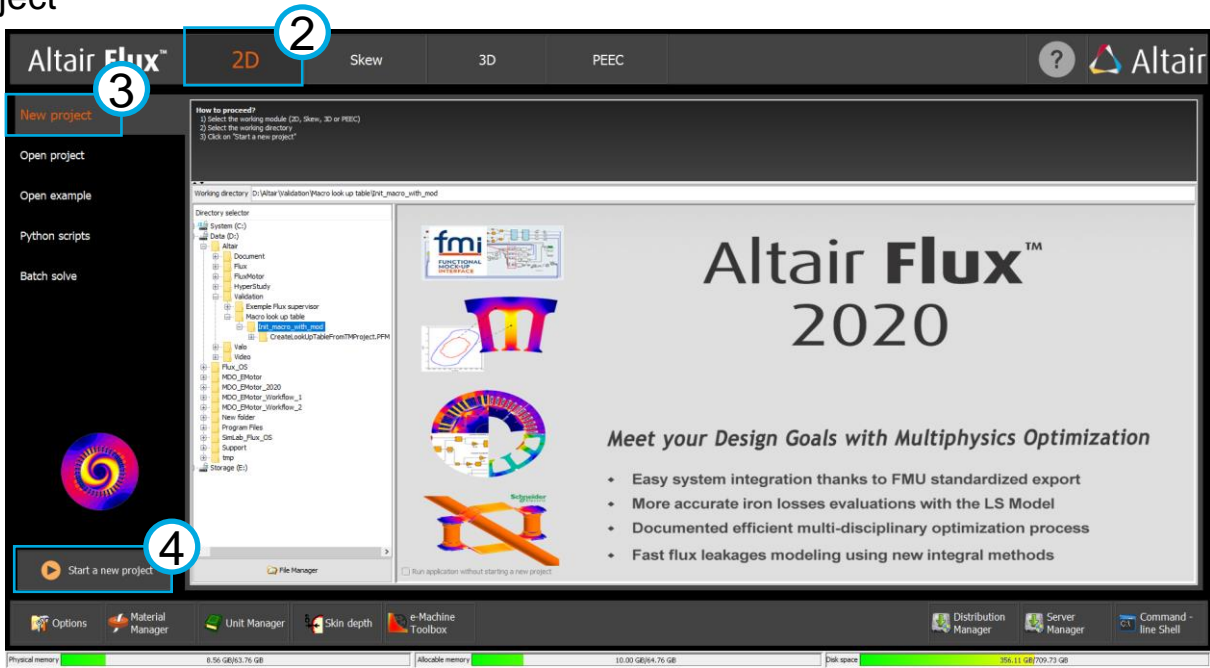
- EMOTOR_Magnetic_Analysis_Project.py

I. CREATION OF MAGNETIC SIMULATION SCENARIOS

CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project initiation
 - Start a new Flux 2D project


Step	Action
1	Open Flux supervisor
2	Select the [2D] simulation context
3	Click on [New project]
4	Click on [Start a new project]

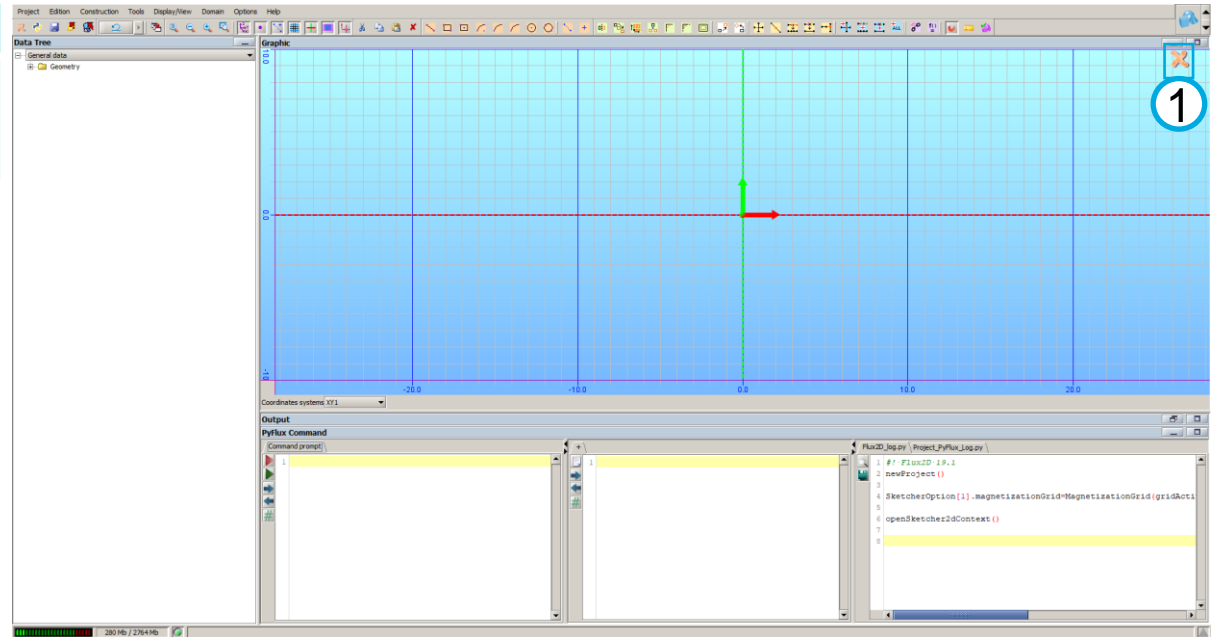


Working path ~\MDO_EMotor\Flux\FluxProjects\MAG_BASE_POINT

CREATION OF MAGNETIC SIMULATION SCENARIOS


- Flux 2D project initiation
 - Close Flux Sketcher 2D Context

Step	Action
1	Click on the icon  to close Flux Sketcher 2D Context



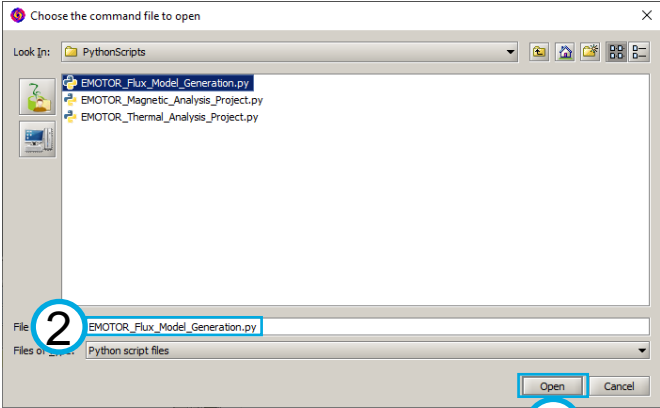
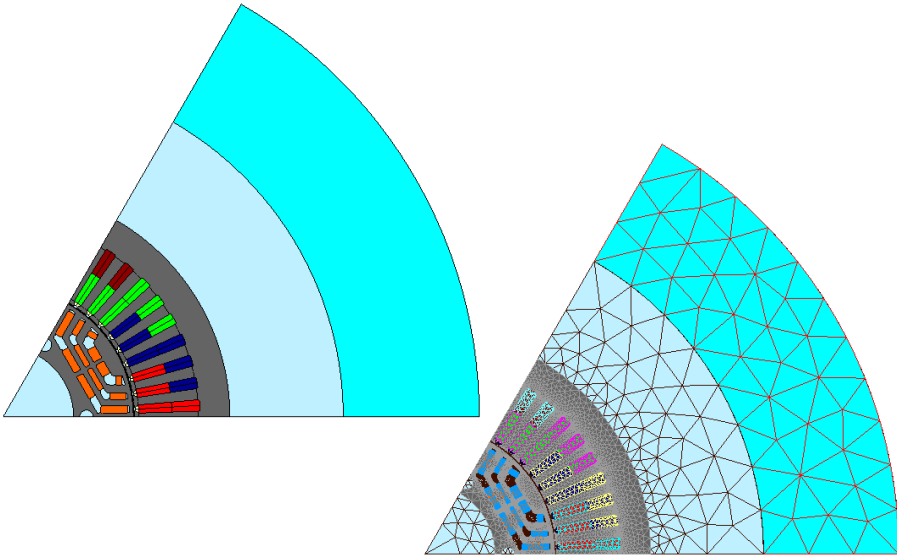
CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project initiation
 - Import the motor model (by executing the Python script generated by FluxMotor)

Step	Action
1	Click on the icon 
2	Select the file "EMOTOR_Flux_Model_Generation.py"
3	Click on [Open]



1



2

3

CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project initiation
 - Create solving scenario “Base Point”

Step	Action
1	Create a Solving scenario by clicking on [Solving] – [Solving scenario] – [New]
2	Define the scenario name as “36_STEPS”
3	Select “Control by position of mechanical set – Rotor”
4	In parameter control tab, define the simulation step as “Step number (lin)” <ul style="list-style-type: none">- Lower limit: 0.0- Higher limit: 126.666666- Step number: 39 Click on [>>]
5	Verify the step values in the [List of resulting values] tap
6	Click on [OK]

The image shows a sequence of screenshots from the Altair Flux 2D software interface, illustrating the steps to create a solving scenario. The screenshots are annotated with numbered callouts (1-6) corresponding to the actions in the table.

- 1:** The 'Solving scenario' menu is open, and the 'New' option is selected.
- 2:** The 'Name of the solving scenario' field in the 'New Solving Scenario' dialog is set to '36_STEPS'.
- 3:** The 'Control type of transient solving process' is set to 'Control by position of mechanical set', and 'ROTOR' is selected as the mechanical set.
- 4:** The 'Parameter control' tab is active, showing the 'Interval definition' section. The 'Lower limit' is 0, the 'Higher limit' is 126.666666, the 'Variation method' is 'Step number (lin)', and the 'Step number' is 39. The '>>' button is highlighted.
- 5:** The 'List of resulting values' table is visible, showing a list of values from 3.333333 to 50.0.
- 6:** The 'OK' button is highlighted.

CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project specification for “Base Speed Point” application
 - Update I/O parameter values

Step	Action
1	Select the parameters “CTRL_ANGLE”, “I_RMS” and “SPEED” from the Data Tree [Parameter/Quantity] – [Parameter I/O]
2	Right click, and click on [Edit array] to modify the values as shown in the following table, click on [OK]

Parameter	Value	Unit
CTRL_ANGLE	49.906	degree
I_RMS	280.0	A
SPEED	7200.0	RPM

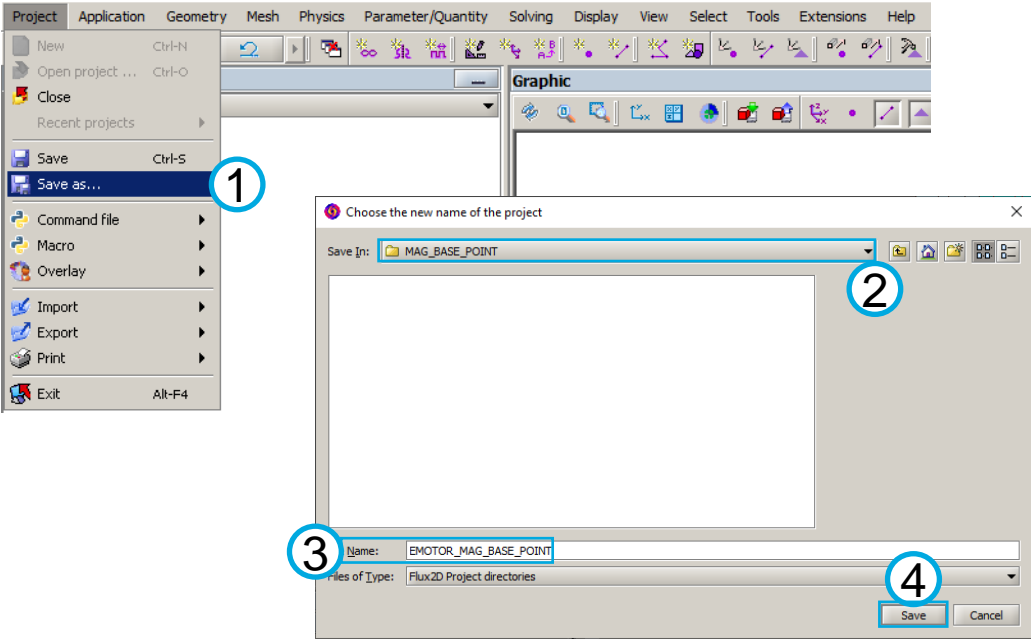
The screenshot shows the Altair Flux 2D software interface. The Data Tree on the left lists various parameters under 'Parameter I/O', including CTRL_ANGLE, I_RMS, and SPEED. A context menu is open over the CTRL_ANGLE parameter, with 'Edit array' selected. The 'Edit Physical parameter[CTRL_ANGLE,I_RMS,SPEED]' dialog box is open, showing a table of parameter values and units.

Entities	Modify all	CTRL_ANGLE	I_RMS	SPEED
Physical parameter				
NAME *		CTRL_ANGLE	I_RMS	SPEED
Comment	Initial values	Control angle	Line current, r...	Speed
Sub types	Initial values	Parameter con...	Parameter con...	Parameter con...
Parameter controlled via a scenario		Parameter con...	Parameter con...	Parameter con...
REFERENCE_VALUE *	Initial values	49.906	280.0	7200.0

CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project specification for “Base Speed Point” application
 - Save the Flux 2D project

Step	Action
1	Click on [Project] – [Save as]
2	Select the saving folder
3	Define the project name
4	Click on [Save]



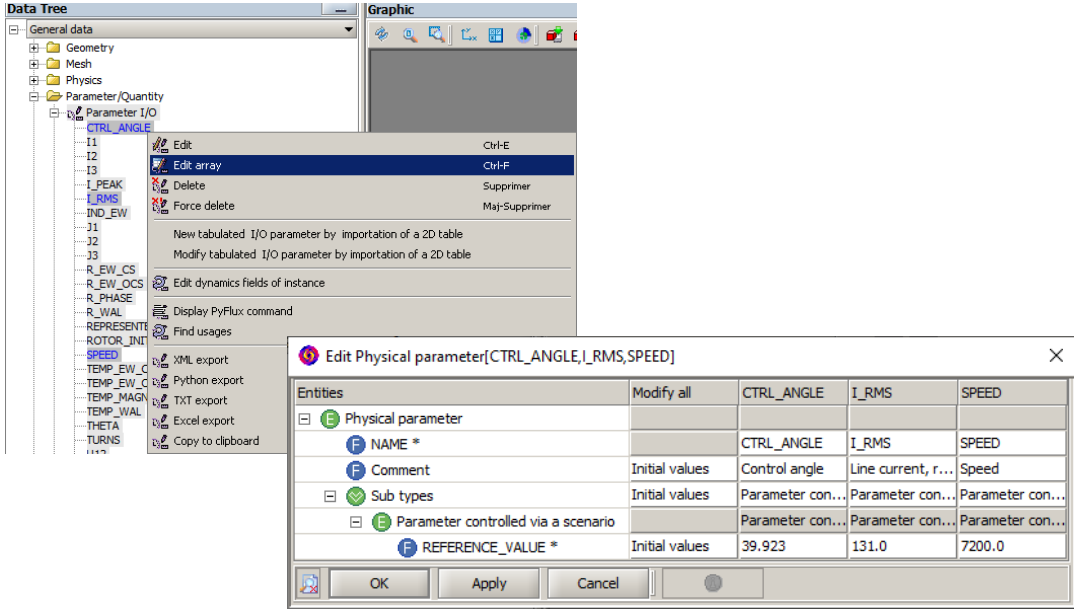
Project name	EMOTOR_MAG_BASE_POINT.FLU
Saving folder	~/MDO_EMOTOR/Flux/FluxProjects/ MAG_BASE_POINT

CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project specification for “Specific Operating Point” application
 - Update I/O parameter values

Step	Action
1	Select the parameters “CTRL_ANGLE”, “I_RMS” and “SPEED” from the Data Tree [Parameter/Quantity] – [Parameter I/O]
2	Right click, and click on [Edit array] to modify the values as shown in the following table, click on [OK]

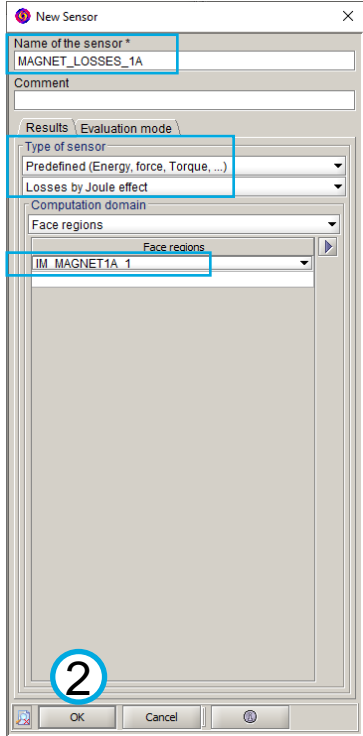
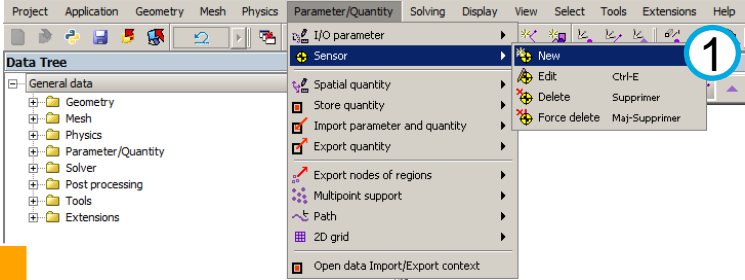
Parameter	Value	Unit
CTRL_ANGLE	39.923	degree
I_RMS	131.0	A
SPEED	7200.0	RPM



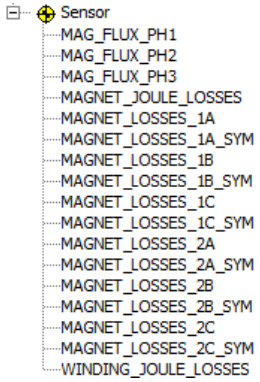
CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project specification for “Specific Operating Point” application
 - Create sensors for each magnet to measure magnet losses

Step	Action
1	Click on [Parameter / Quantity] – [Sensor] – [New]
2	Create the 12 sensors in the following table



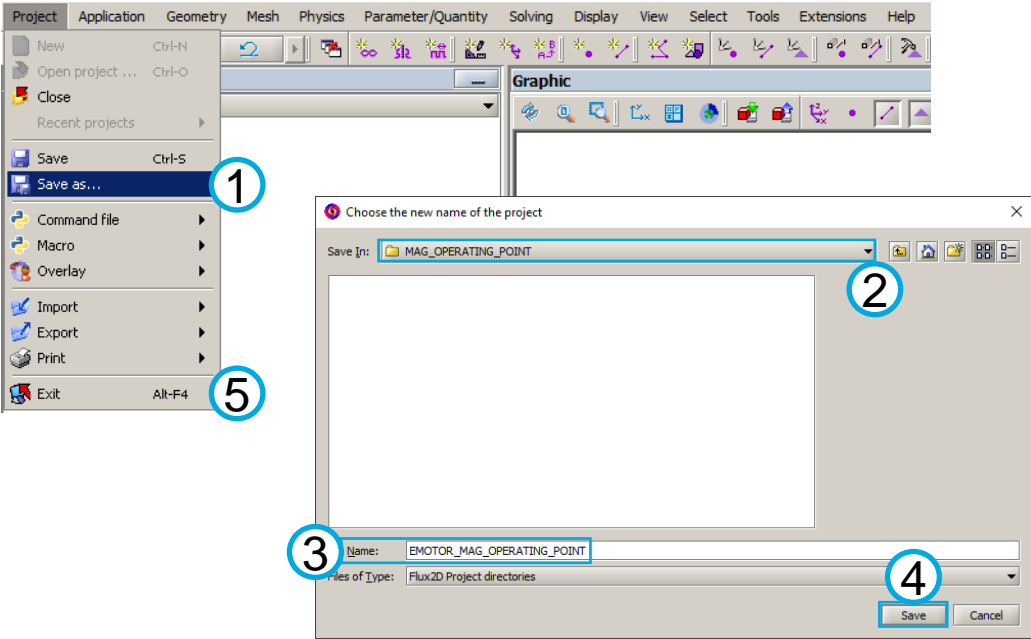
Sensor names	Corresponding face region names
MAGNET_LOSSES_1A	IM_MAGNET1A_1
MAGNET_LOSSES_1A_SYM	IM_MAGNET1A_SYM_1
MAGNET_LOSSES_1B	IM_MAGNET1B_1
MAGNET_LOSSES_1B_SYM	IM_MAGNET1B_SYM_1
MAGNET_LOSSES_1C	IM_MAGNET1C_1
MAGNET_LOSSES_1C_SYM	IM_MAGNET1C_SYM_1
MAGNET_LOSSES_2A	IM_MAGNET2A_1
MAGNET_LOSSES_2A_SYM	IM_MAGNET2A_SYM_1
MAGNET_LOSSES_2B	IM_MAGNET2B_1
MAGNET_LOSSES_2B_SYM	IM_MAGNET2B_SYM_1
MAGNET_LOSSES_2C	IM_MAGNET2C_1
MAGNET_LOSSES_2C_SYM	IM_MAGNET2C_SYM_1



CREATION OF MAGNETIC SIMULATION SCENARIOS

- Flux 2D project specification for “Specific Operating Point” application
 - Save the Flux 2D project

Step	Action
1	Click on [Project] – [Save as]
2	Define the project name
3	Select the saving folder
4	Click on [Save]
5	Click on [Project] – [Exit]



Project name	EMOTOR_MAG_OPERATING_POINT.FLU
Saving folder	~/MDO_EMOTOR/Flux/FluxProjects/ MAG_OPERATING_POINT

MAGNETIC ANALYSIS: BASE SPEED POINT

OUTLINE

Solving and
postprocessing:
base speed point

Generating
HyperStudy connector:
base speed point

Input file

- **Flux 2D project**
- EMOTOR_MAG_BASE_POINT.FLU

Software

- Altair Flux 2019.1 (or later versions)

Output files

- **Flux / HyperStudy connector**
Connector_Fx_Hst_BasePoint.F2HST
- **Flux project associated with the connector**
Connector_Fx_Hst_BasePoint.F2HST.FLU
- **Python script for the postprocessing in HyperStudy**
Connector_Fx_Hst_BasePoint.py

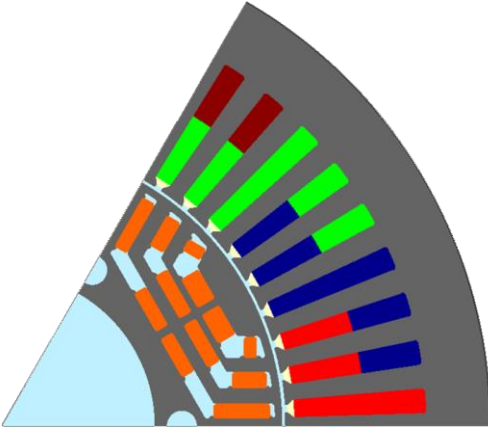
SOLVING AND POSTPROCESSING: BASE SPEED POINT

SOLVING AND POSTPROCESSING: BASE SPEED POINT

Solving the magnetic problem at base speed point

- Open Flux 2D project

Step	Action
1	Open the project "EMOTOR_MAG_BASE_POINT.FLU"



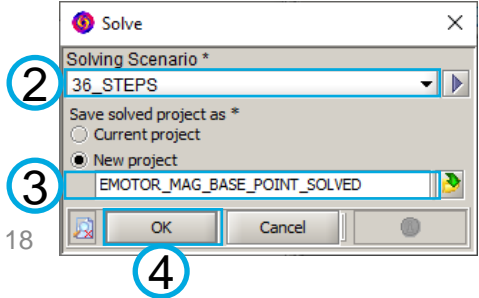
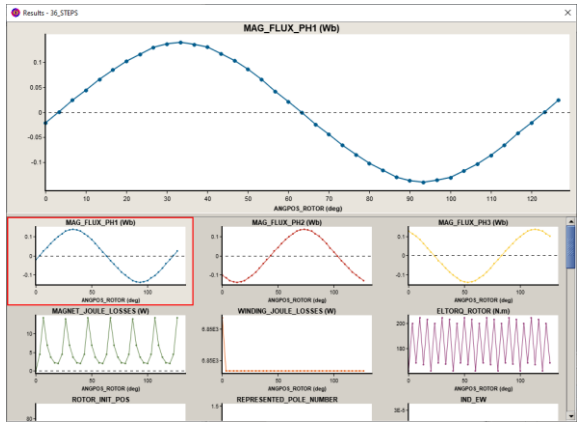
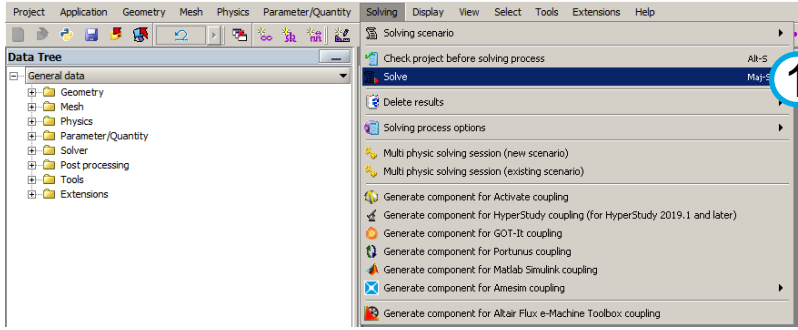
Project name	EMOTOR_MAG_BASE_POINT.FLU
Project folder	~/MDO_EMOTOR/Flux/FluxProjects/ MAG_BASE_POINT

SOLVING AND POSTPROCESSING: BASE SPEED POINT

Solving the magnetic problem at base speed point

- Solve the project

Step	Action
1	Click on [Solving] – [Solve]
2	Select the solving scenario “36_STEPS”
3	Save the solved project as a new project “EMOTOR_MAG_BASE_POINT_SOLVED.FLU”
4	Click on [OK]

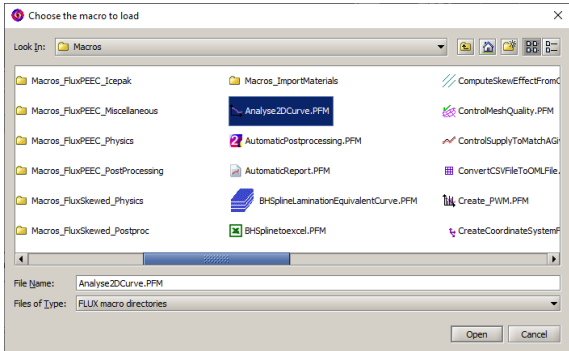
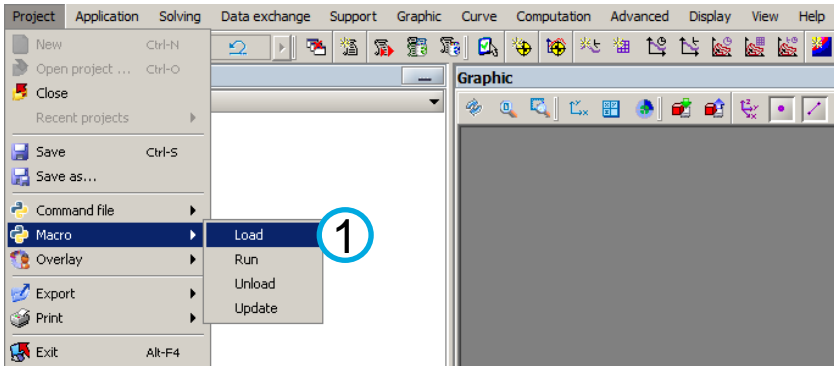


SOLVING AND POSTPROCESSING: BASE SPEED POINT

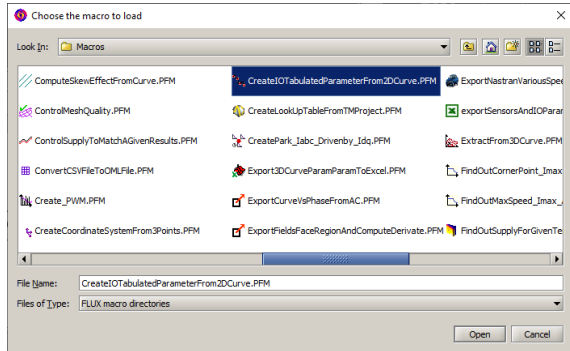
Magnetic analysis: postprocessing initialization

- Load Flux macros

Step	Action
1	Click on [Project] – [Macro] – [Load]
2	Load the following two macros: - Analyse2DCurve - CreateIOTabulatedParameterFrom2DCurve



This macro extracts different values on a 2D curve, and create corresponding I/O parameters



This macro intends to create a new tabulated I/O parameter from a 2D curve.

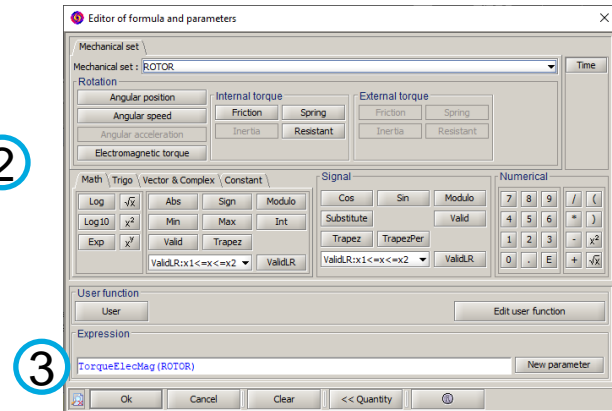
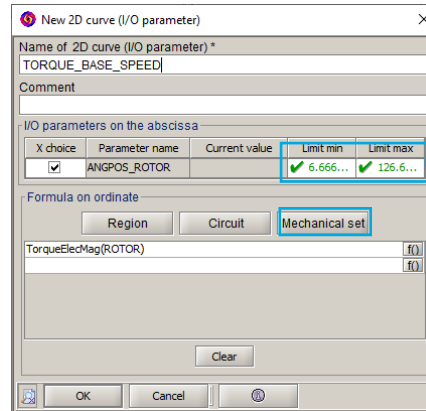
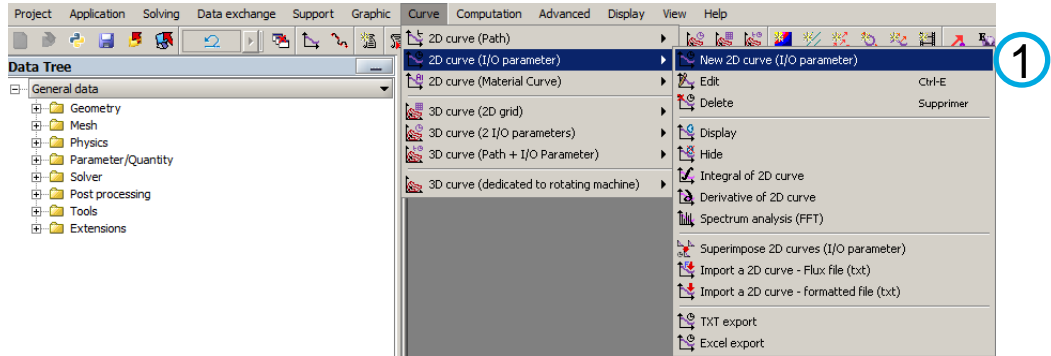
SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: magnetic torque at base speed point

- Plot the magnetic torque curve

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create a new 2D curve TORQUE_BASE_SPEED Limit min: 6.666666 Limit max: 126.666666
3	Click on [Mechanical set], select “Rotor”, and click on [Electromagnetic torque], click on [OK]
4	Click on [OK]

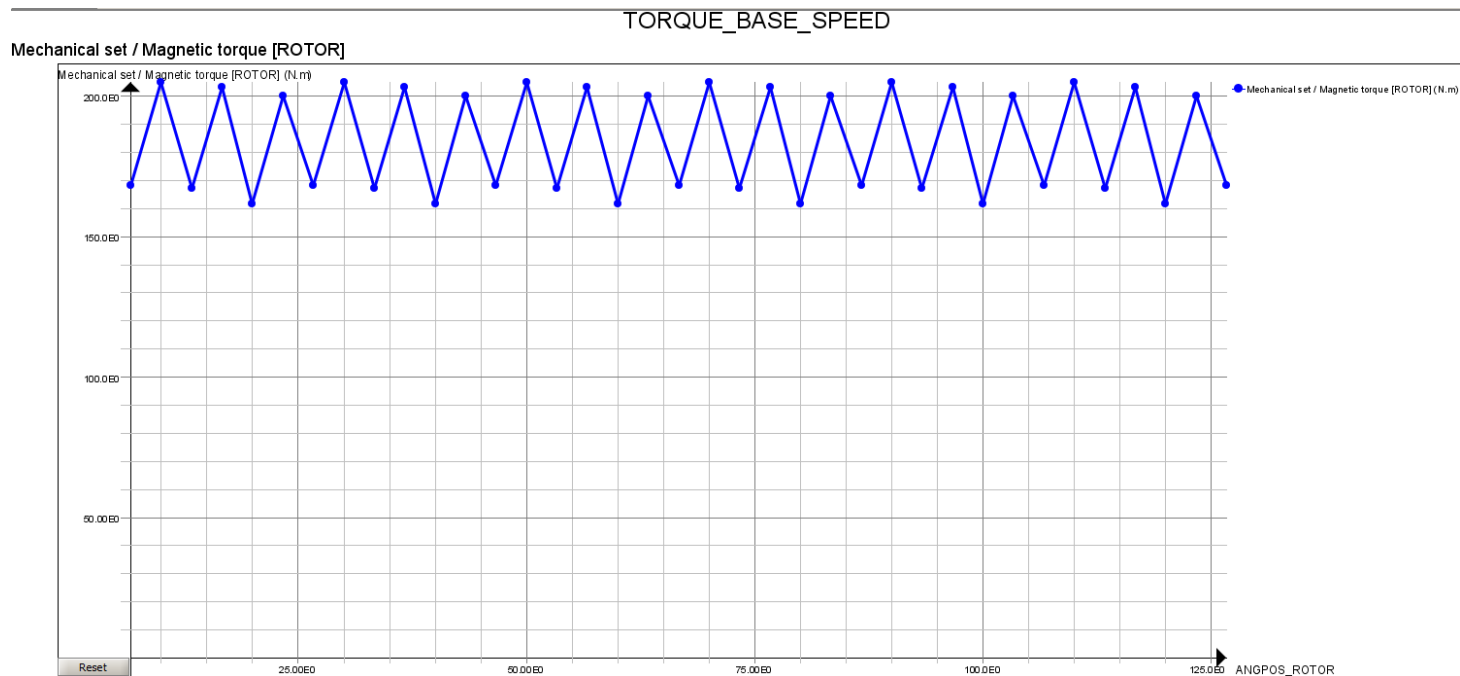
Curve name	TORQUE_BASE_SPEED
Formula	TorqueElecMag(ROTOR)



SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: magnetic torque at base speed point

- Plot the magnetic torque curve

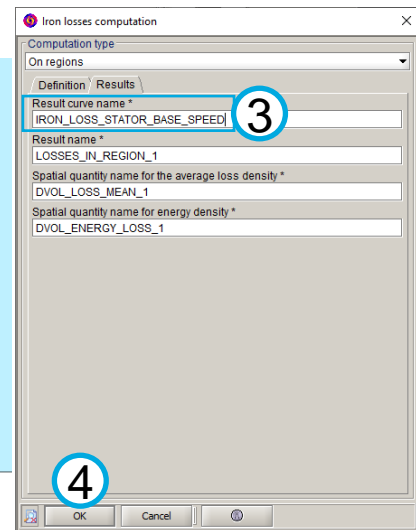
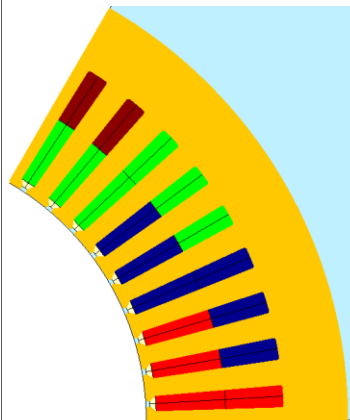
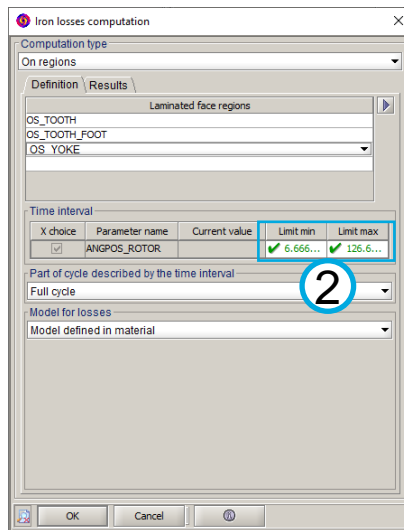
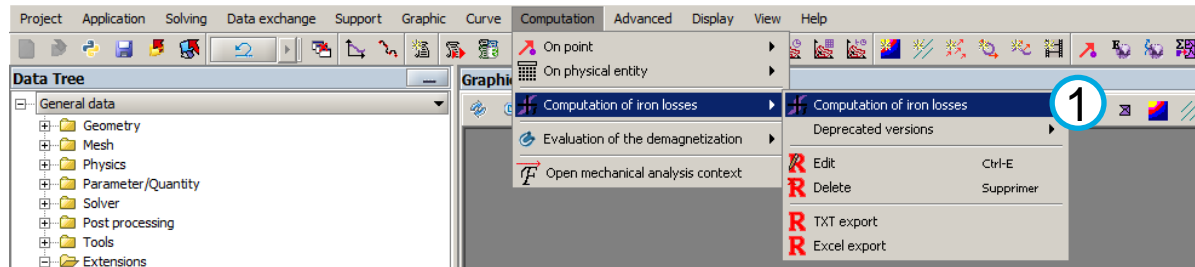


SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: stator iron losses at base speed point

- Plot the stator iron loss curve

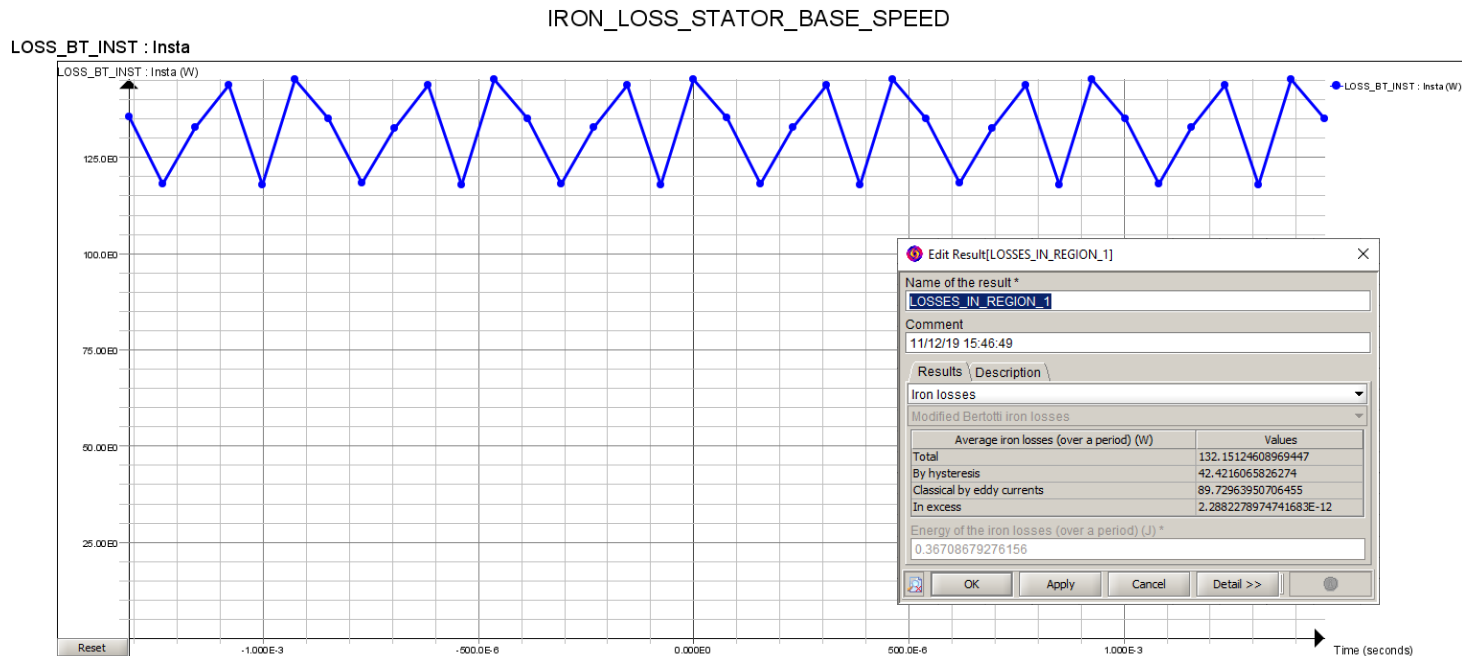
Step	Action
1	Click on [Computation] – [Computation of iron losses] – [Computation of iron losses]
2	Define the computation configuration in tab [Definition]: Face region: - OS_TOOTH - OS_TOOTH_FOOT - OS_YOKE Interval: [6.6666, 126.6666]
3	Define the computation configuration in tab [Results]: Result curve name: IRON_LOSS_STATOR_BASE_SPEED
4	Click on [OK]



SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: stator iron losses at base speed point

- Plot the stator iron loss curve

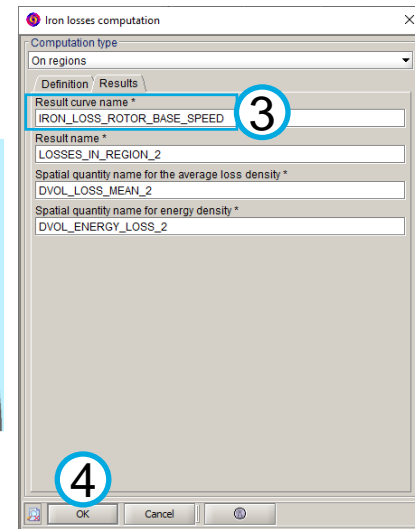
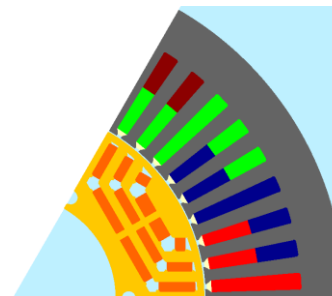
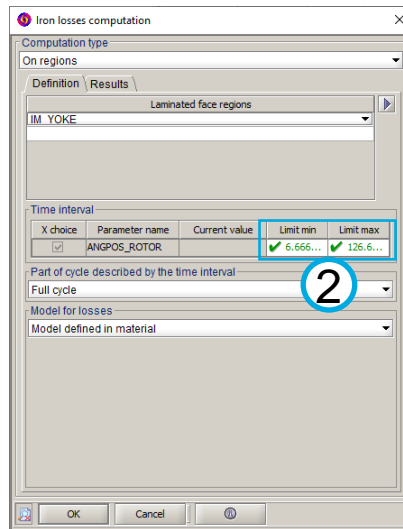
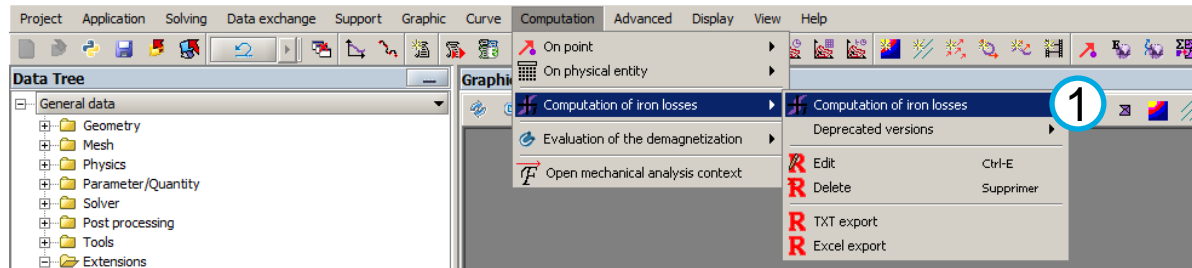


SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: rotor iron losses at base speed point

- Plot the rotor iron loss curve

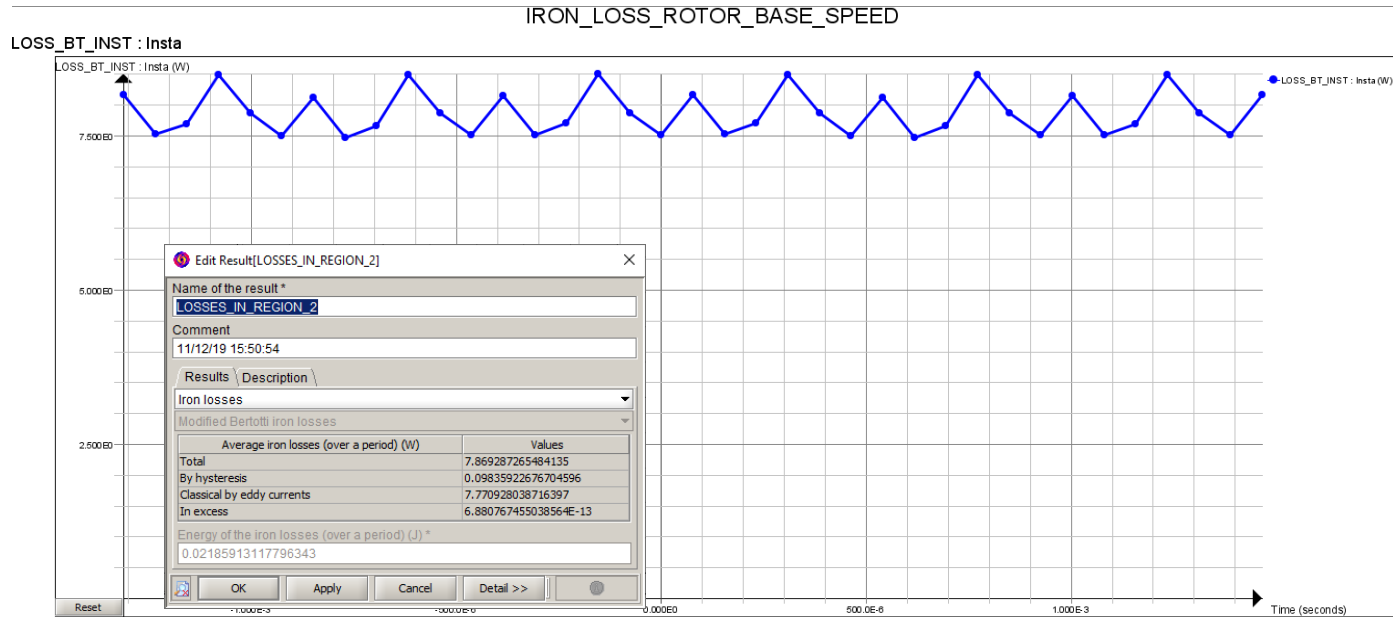
Step	Action
1	Click on [Computation] – [Computation of iron losses] – [Computation of iron losses]
2	Define the computation configuration in tab [Definition]: Face region: - IM_YOKE Interval: [6.6666, 126.6666]
3	Define the computation configuration in tab [Results]: Result curve name: IRON_LOSS_ROTOR_BASE_SPEED
4	Click on [OK]



SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: rotor iron losses at base speed point

- Plot the rotor iron loss curve



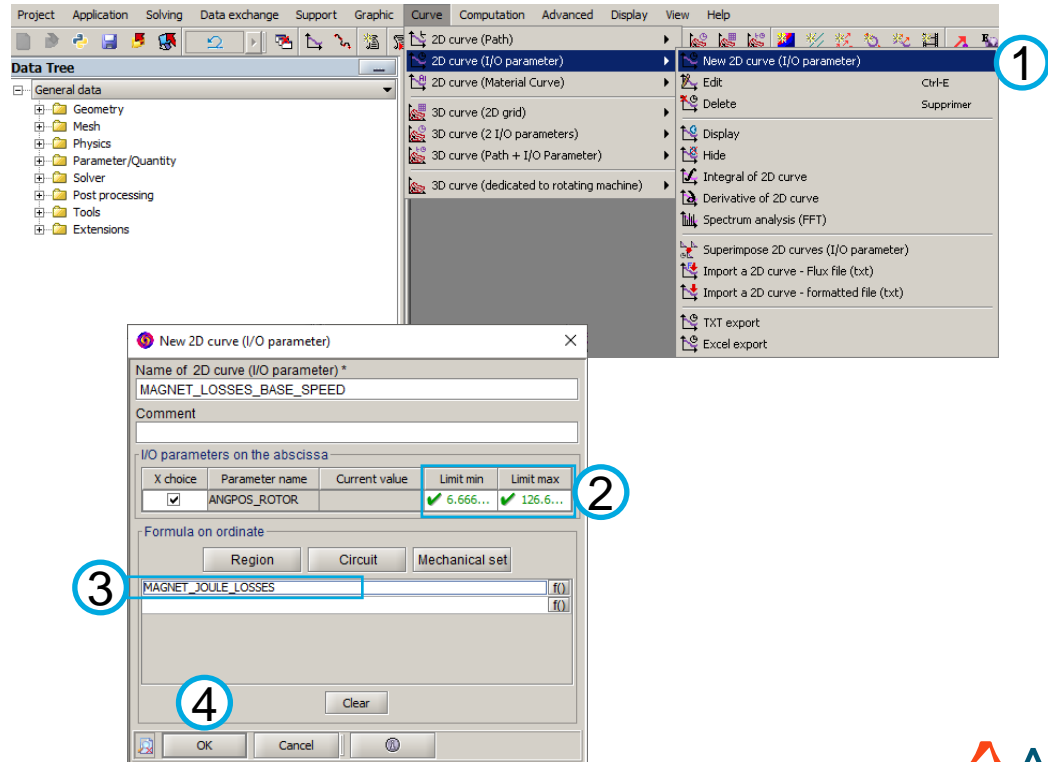
SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: magnet losses at base speed point

- Plot the magnet loss curve

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create a new 2D curve MAGNET_LOSSES_BASE_SPEED Limit min: 6.666666 Limit max: 126.666666
3	Define the formula as the predefined sensor MAGNET_JOULE_LOSSES
4	Click on [OK]

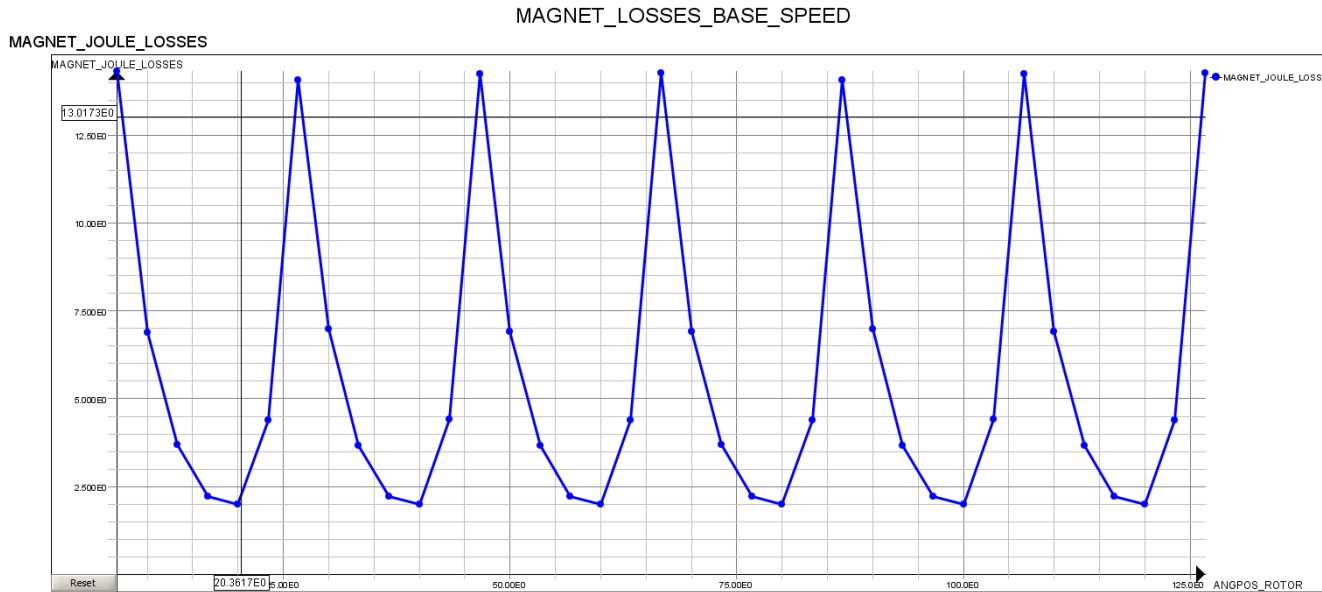
Curve name	MAGNET_LOSSES_BASE_SPEED
Formula	MAGNET_JOULE_LOSSES



SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: magnet losses at base speed point

- Plot the magnet loss curve



SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: Joule losses at base speed point

- Plot the Joule loss curve

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create a new 2D curve JOULE_LOSSES_BASE_SPEED Limit min: 6.666666 Limit max: 126.666666
3	Define the plotting formula as $P_{\text{jouleCC}}(\text{PHASE_1}) + P_{\text{jouleCC}}(\text{PHASE_2}) + P_{\text{jouleCC}}(\text{PHASE_3})$
4	Click on [OK]

Curve name	JOULE_LOSSES_BASE_SPEED
Formula	$P_{\text{jouleCC}}(\text{PHASE_1}) + P_{\text{jouleCC}}(\text{PHASE_2}) + P_{\text{jouleCC}}(\text{PHASE_3})$

The screenshot shows the Altair software interface with the following elements:

- Curve Menu:** A dropdown menu is open, showing options like '2D curve (I/O parameter)', '2D curve (Material Curve)', '3D curve (2D grid)', etc. A red circle with the number '1' highlights the 'New 2D curve (I/O parameter)' option.
- New 2D curve (I/O parameter) Dialog:** A dialog box is open with the following fields:
 - Name of 2D curve (I/O parameter) *: JOULE_LOSSES_BASE_SPEED
 - Comment: (empty)
 - I/O parameters on the abscissa: A table with columns 'X choice', 'Parameter name', 'Current value', 'Limit min', and 'Limit max'. The 'X choice' for 'ANGPOS_ROTOR' is checked, and the 'Limit min' and 'Limit max' are set to 6.666... and 126.6... respectively.
 - Formula on ordinate: $P_{\text{jouleCC}}(\text{PHASE_1}) + P_{\text{jouleCC}}(\text{PHASE_2}) + P_{\text{jouleCC}}(\text{PHASE_3})$
 - Buttons: Region, Circuit, Mechanic, OK, Cancel.
 A red circle with the number '2' highlights the 'Limit min' and 'Limit max' fields, and a red circle with the number '3' highlights the formula field.
- Computation on circuit Dialog:** A dialog box is open with the following fields:
 - Type of electrical component: Stranded coil conductor
 - Formula: $P_{\text{jouleCC}}(\text{PHASE_1})$
 - Buttons: Add, Add all, Delete, Delete all, OK, Cancel.
 A red circle with the number '4' highlights the 'OK' button.

SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: Joule losses at base speed point

- Plot the Joule loss curve



SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: useful power at base speed point

- Plot the useful power curve

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create a new 2D curve USEFUL_POWER_BASE_SPEED Limit min: 6.666666 Limit max: 126.666666
3	Define the plotting formula as TorqueElecMag(ROTOR)*AngSpeed(ROTOR)*PI()/180
4	Click on [OK]

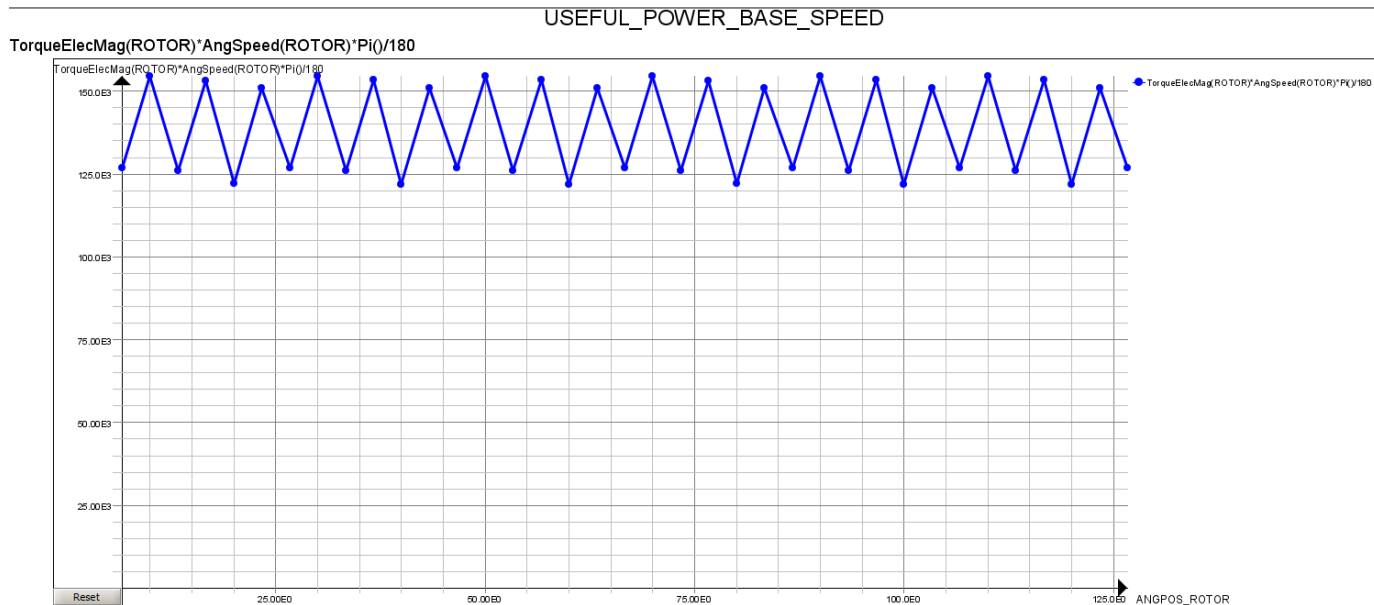
Curve name	USEFUL_POWER_BASE_SPEED
Formula	$\text{TorqueElecMag}(\text{ROTOR}) * \text{AngSpeed}(\text{ROTOR}) * \text{PI}() / 180$

The screenshot shows the software interface for creating a 2D curve. The 'Curve' menu is open, and 'New 2D curve (I/O parameter)' is selected (1). The 'New 2D curve (I/O parameter)' dialog box is shown with 'Name of 2D curve (I/O parameter)' set to 'USEFUL_POWER_BASE_SPEED' and 'Limit min' and 'Limit max' set to 6.666... and 126.6... respectively (2). The 'Editor of formula and parameters' dialog box is shown with the formula 'TorqueElecMag(ROTOR)*AngSpeed(ROTOR)*PI()/180' entered in the 'Expression' field (3). The 'OK' button is highlighted (4).

SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: useful power at base speed point

- Plot the useful power curve





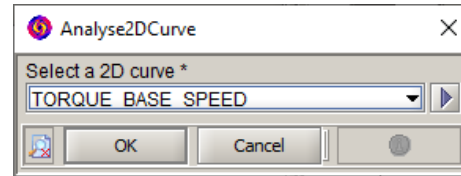
SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis

- Analyze the curves with macros

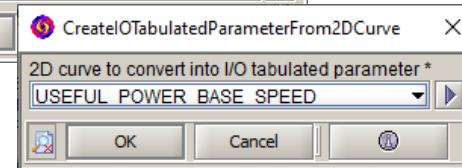
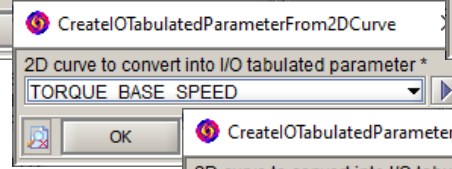
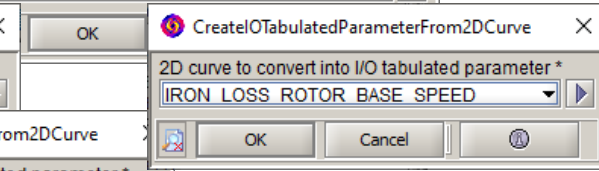
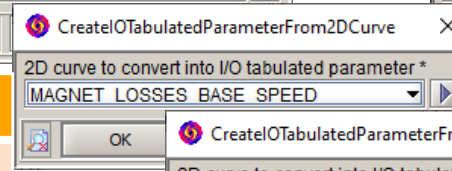
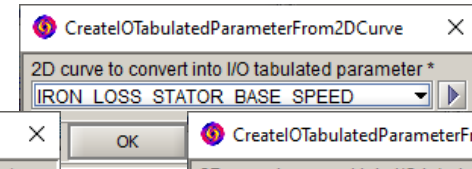
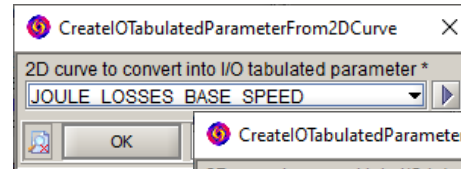


Step	Action
1	Click on the icon  to run the macro "Analyse2D Curve"
2	Click on the icon  to run the macro "Create I/O Tabulated Parameter"



Tips: the curve analysis by predefined macros in Flux will generate parameters to calculate motor efficiency

Analyse2D Curve	Create I/O Tabulated Parameter From 2D Curve
TORQUE_BASE_SPEED	IRON_LOSS_ROTOR_BASE_SPEED
	IRON_LOSS_STATOR_BASE_SPEED
	JOULE_LOSSES_BASE_SPEED
	MAGNET_LOSSES_BASE_SPEED
	TORQUE_BASE_SPEED
	USEFUL_POWER_BASE_SPEED

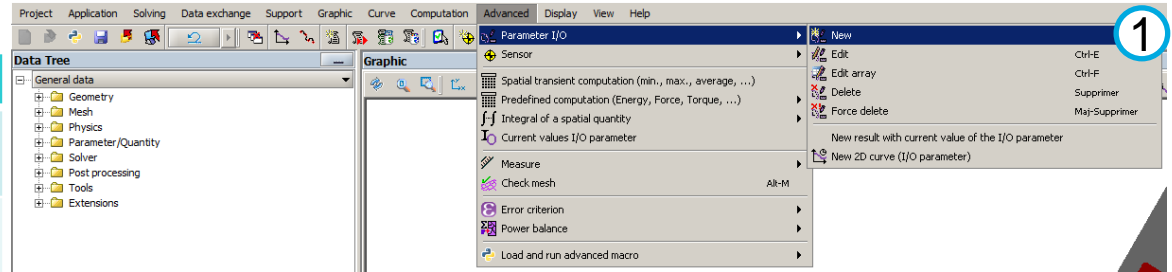


SOLVING AND POSTPROCESSING: BASE SPEED POINT

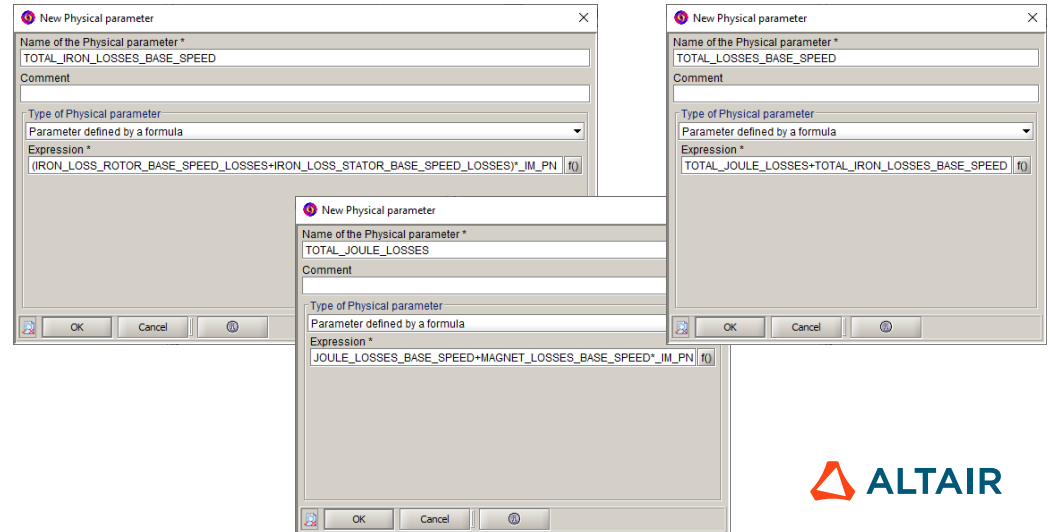
Magnetic analysis: motor efficiency at base speed point

- Define variation parameters

Step	Action
1	Click on [Advanced] – [Parameter I/O] – [New]
2	Create three new variation parameters from previous curves - TOTAL_IRON_LOSSES_BASE_SPEED - TOTAL_JOULE_LOSSES - TOTAL_LOSSES_BASE_SPEED



Parameter name	Formula expression
TOTAL_IRON_LOSSES_BASE_SPEED	$(\text{IRON_LOSS_ROTOR_base_speed} + \text{IRON_LOSS_STATOR_base_speed}) * \text{IM_PN}$
TOTAL_JOULE_LOSSES	$\text{JOULE_LOSSES_base_speed} + \text{MAGNET_LOSSES_base_speed} * \text{IM_PN}$
TOTAL_LOSSES_BASE_SPEED	$\text{TOTAL_Joule_losses} + \text{Total_Iron_losses_base_speed}$

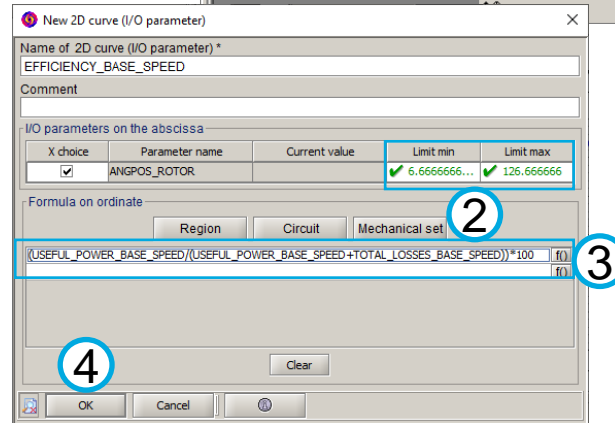
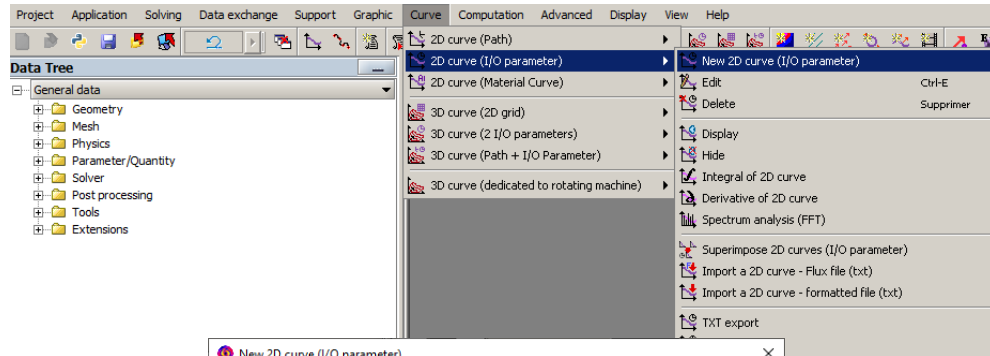


SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: motor efficiency at base speed point

- Plot the motor efficiency curve

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create a new 2D curve EFFICIENCY_BASE_SPEED Limit min: 6.666666 Limit max: 126.666666
3	Define the plotting formula as (USEFUL_POWER_BASE_SPE ED/(USEFUL_POWER_BASE_S PEED+TOTAL_LOSSES_BASE_S PEED))*100
4	Click on [OK]



Curve name	EFFICIENCY_BASE_SPEED
Formula	$(\text{USEFUL_POWER_BASE_SPEED} / (\text{USEFUL_POWER_BASE_SPEED} + \text{TOTAL_LOSSES_BASE_SPEED})) * 100$

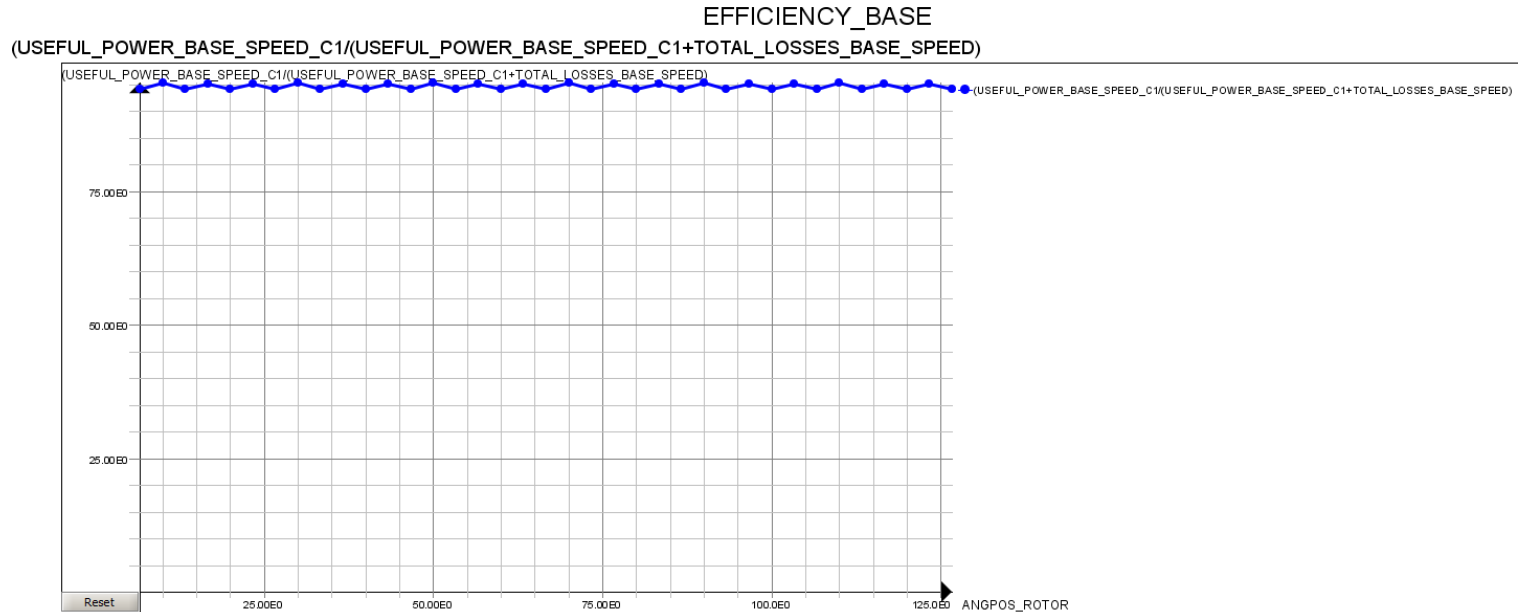
34

Tips: these three parameters are to calculate motor efficiency

SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: motor efficiency at base speed point


- Plot the motor efficiency curve

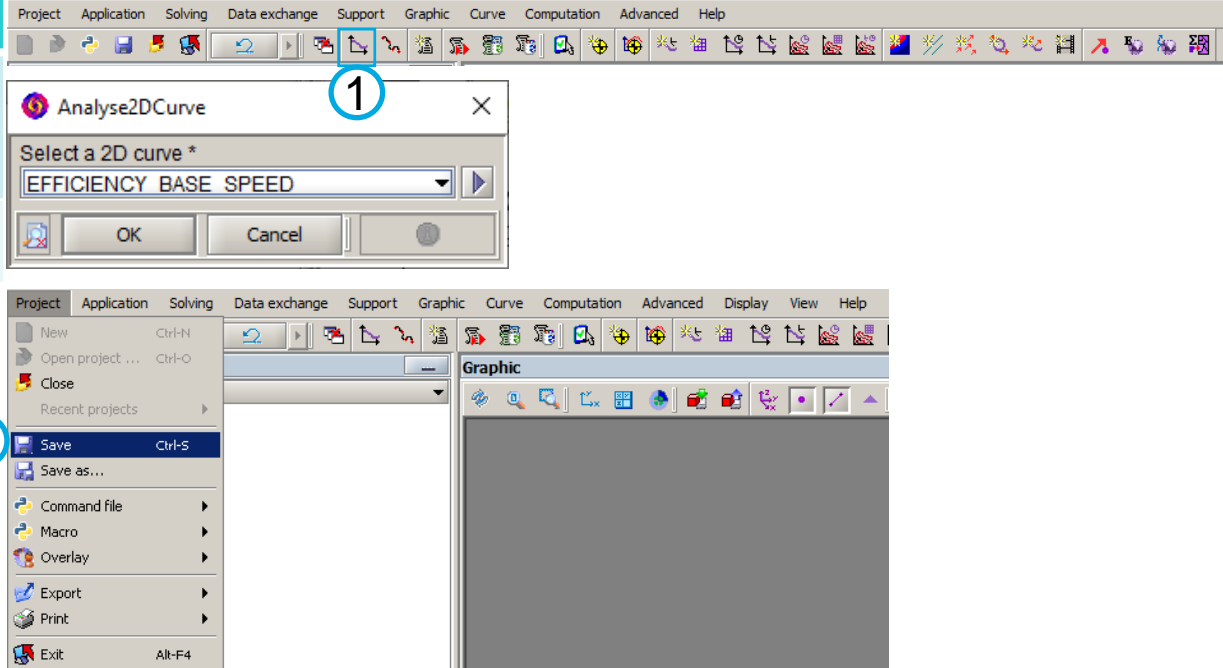


SOLVING AND POSTPROCESSING: BASE SPEED POINT

Magnetic analysis: motor efficiency at base speed point

- Analyze the efficiency curve with macros

Step	Action
1	Click on the icon  to run the macro "Analyse2DCurve", select "EFFICIENCY_BASE_SPEED"
2	Click on [Project] – [Save] to save the project

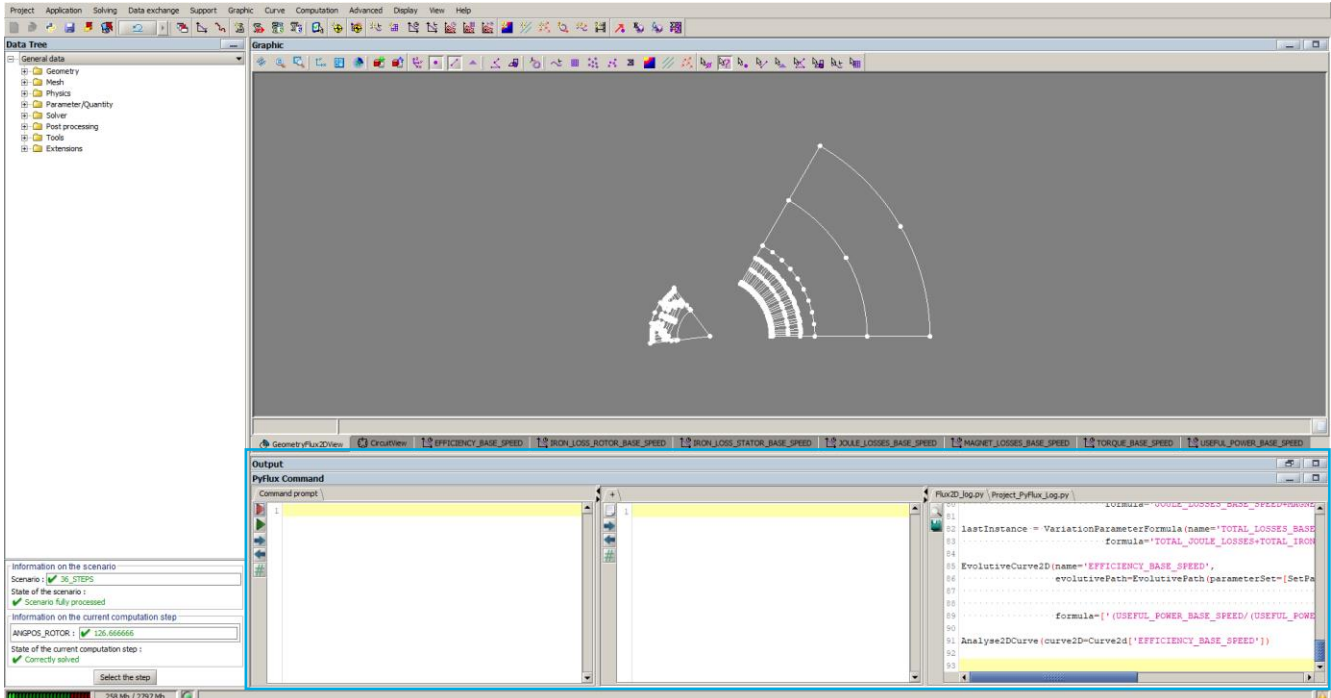


The screenshot shows the Altair software interface. The top menu bar includes Project, Application, Solving, Data exchange, Support, Graphic, Curve, Computation, Advanced, and Help. The 'Analyse2DCurve' dialog box is open, with a blue circle '1' around the 'Analyse2DCurve' icon in the toolbar. The dialog box has a 'Select a 2D curve *' dropdown menu with 'EFFICIENCY_BASE_SPEED' selected. Below the dialog box, the 'Project' menu is open, with a blue circle '2' around the 'Save' option. The 'Save' option is highlighted in blue. The 'Graphic' window is visible on the right side of the interface.

GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT


Preparing post-processing Python script for HyperStudy connector

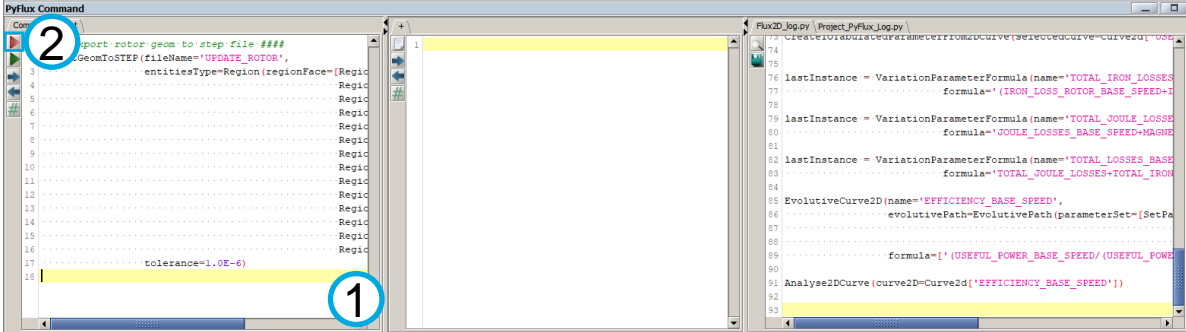


GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Preparing post-processing Python script for HyperStudy connector

- Run scripts to export the rotor geometry file

Step	Action
1	Copy the following scripts and paste in the [Command prompt] windows
2	Click on the icon  to execute the scripts




```
selectCurrentStep(activeScenario=Scenario['36_STEPS'],  
parameterValue=['ANGPOS_ROTOR=59.999996842105'])
```

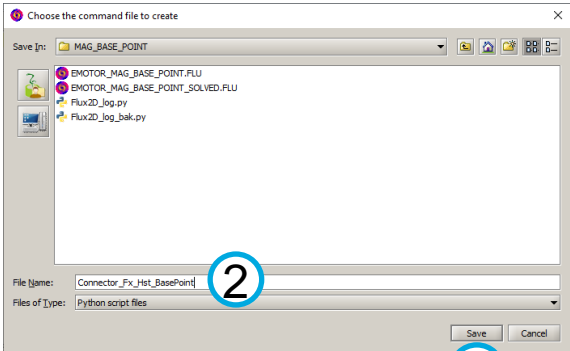
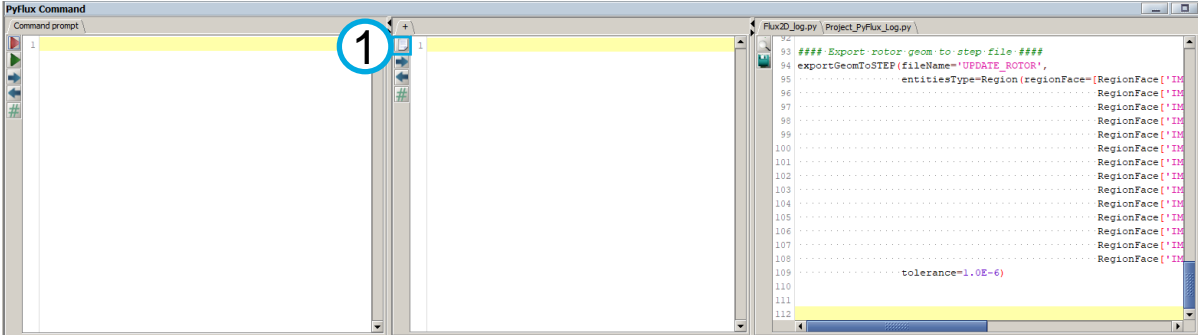
```
#### Export rotor geom to step file ####  
exportGeomToSTEP(fileName='UPDATE_ROTOR',  
entitiesType=Region(regionFace=[RegionFace['IM_YOKE'],  
RegionFace['IM_EDGE'],  
RegionFace['IM_MAGNET2A_1'],  
RegionFace['IM_MAGNET2B_1'],  
RegionFace['IM_MAGNET2C_1'],  
RegionFace['IM_MAGNET1A_1'],  
RegionFace['IM_MAGNET1B_1'],  
RegionFace['IM_MAGNET1C_1'],  
RegionFace['IM_MAGNET1C_SYM_1'],  
RegionFace['IM_MAGNET1B_SYM_1'],  
RegionFace['IM_MAGNET1A_SYM_1'],  
RegionFace['IM_MAGNET2C_SYM_1'],  
RegionFace['IM_MAGNET2B_SYM_1'],  
RegionFace['IM_MAGNET2A_SYM_1']],  
tolerance=1.0E-6)
```

GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Preparing post-processing Python script for HyperStudy connector

- Create a new Python script

Step	Action
1	Click on the icon  to create a new Python script file
2	Define the script name and the saving folder
3	Click on [Save]




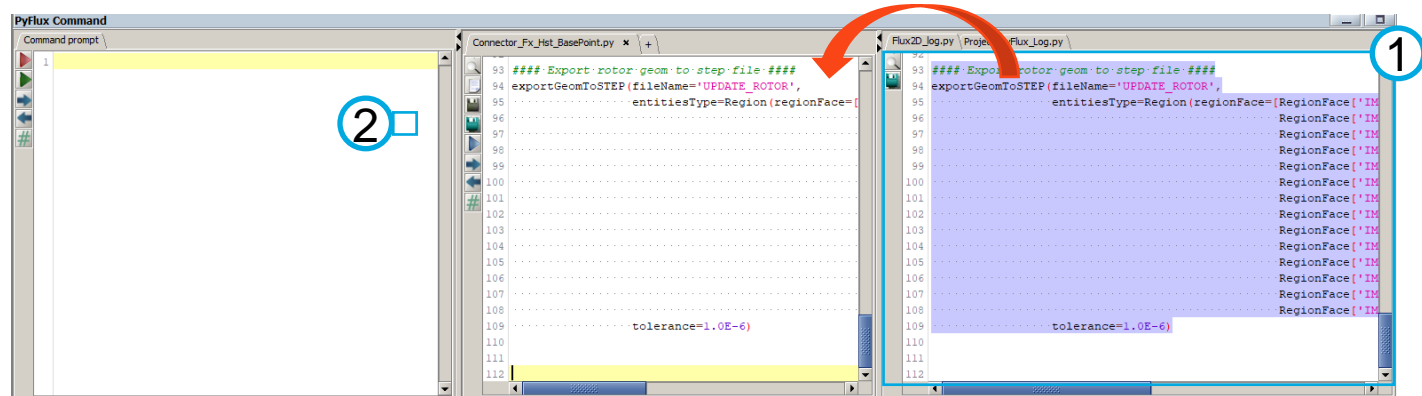
Script name	Connector_Fx_Hst_BasePoint.py
Saving folder	~/MDO_EMOTOR/Flux/FluxProjects/ MAG_BASE_POINT

GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Preparing post-processing Python script for HyperStudy connector

- Create backup of the log script

Step	Action
1	Copy all the script from the Flux2D_log.py file into the created Python script
2	Click on the icon  to save the script
3	Click on [Project] – [Exit]



GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Preparing post-processing Python script for HyperStudy connector

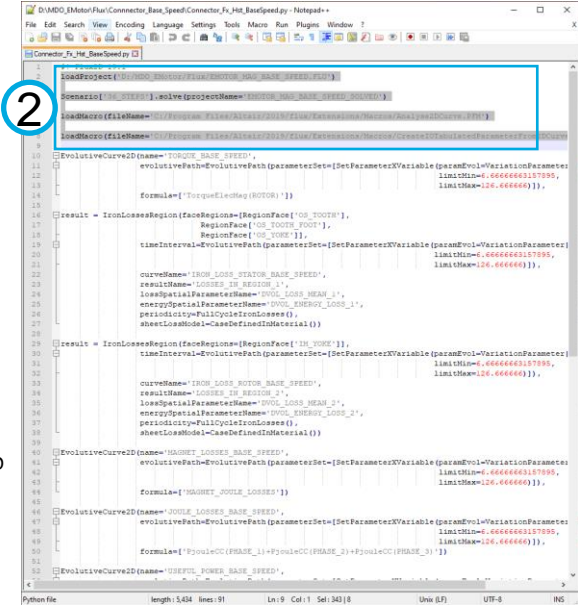
- Replace Flux macro import commands

Step	Action
1	Open the Python script by Notepad editor
2	Delete the scripts before post-processing
3	Insert the following scripts before post-processing

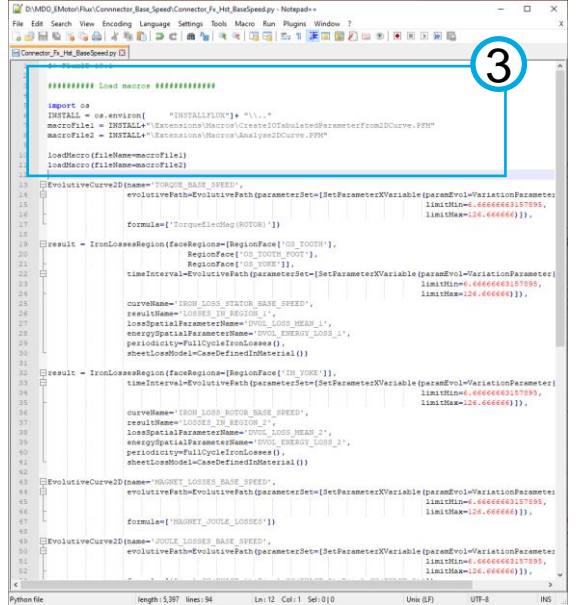
```
##### Load macros #####

import os
INSTALL = os.environ[ "INSTALLFLUX"]+ "\\."
macroFile1 =
INSTALL+"Extensions\Macros\CreateOTabulatedParameterFrom2D
Curve.PFM"
macroFile2 = INSTALL+"Extensions\Macros\Analyse2DCurve.PFM"

loadMacro(fileName=macroFile1)
loadMacro(fileName=macroFile2)
```



```
loadProject("D:\MDO\Motor\FuelConnector_BaseSpeedConnector_Fx_Ht_BaseSpeed.py - Notepad+
...
Scenario["HT_STATOR"].evolve(pjobjectName=...)
loadMacro(fileName=...)
loadMacro(fileName=...)
...
EvolutiveCurve2D(name="TORQUE_BASE_SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
formula=["TorqueElecMag(ROTOR)"]
...
result = IronLossesRegion(faceRegions={RegionFace["OS_TOOTH"],
RegionFace["OS_TOOTH_FOOT"],
RegionFace["OS_TOOTH_TIP"]},
timeInterval=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
curveName="IRON_LOSS_STATOR_BASE_SPEED",
resultName="LOSSES_STATOR_REGION_1",
loadSpatialParameterName="DVOL_ENERGY_NEAR_1",
energySpatialParameterName="DVOL_ENERGY_LOSS_1",
periodicityType=FullCycleIronLosses(),
sheetLoadModel=CasDefinedMaterial())
...
result = IronLossesRegion(faceRegions={RegionFace["IN YOKE"]},
timeInterval=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
curveName="IRON_LOSS_ROTOR_BASE_SPEED",
resultName="LOSSES_STATOR_REGION_2",
loadSpatialParameterName="DVOL_ENERGY_NEAR_2",
energySpatialParameterName="DVOL_ENERGY_LOSS_2",
periodicityType=FullCycleIronLosses(),
sheetLoadModel=CasDefinedMaterial())
...
EvolutiveCurve2D(name="MAGNET LOSSES BASE SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
formula=["MAGNET_JOULE_LOSSES"]
...
EvolutiveCurve2D(name="JOULE LOSSES BASE SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
formula=["PjoulECC(STRASE_1)+PjoulECC(STRASE_2)+PjoulECC(STRASE_3)"]
...
EvolutiveCurve2D(name="USER_POWER_BASE_SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
```



```
##### Load macros #####

import os
INSTALL = os.environ[ "INSTALLFLUX"]+ "\\."
macroFile1 = INSTALL+"Extensions\Macros\CreateOTabulatedParameterFrom2DCurve.PFM"
macroFile2 = INSTALL+"Extensions\Macros\Analyse2DCurve.PFM"

loadMacro(fileName=macroFile1)
loadMacro(fileName=macroFile2)
...
EvolutiveCurve2D(name="TORQUE_BASE_SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
formula=["TorqueElecMag(ROTOR)"]
...
result = IronLossesRegion(faceRegions={RegionFace["OS_TOOTH"],
RegionFace["OS_TOOTH_FOOT"],
RegionFace["OS_TOOTH_TIP"]},
timeInterval=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
curveName="IRON_LOSS_STATOR_BASE_SPEED",
resultName="LOSSES_STATOR_REGION_1",
loadSpatialParameterName="DVOL_ENERGY_NEAR_1",
energySpatialParameterName="DVOL_ENERGY_LOSS_1",
periodicityType=FullCycleIronLosses(),
sheetLoadModel=CasDefinedMaterial())
...
result = IronLossesRegion(faceRegions={RegionFace["IN YOKE"]},
timeInterval=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
curveName="IRON_LOSS_ROTOR_BASE_SPEED",
resultName="LOSSES_STATOR_REGION_2",
loadSpatialParameterName="DVOL_ENERGY_NEAR_2",
energySpatialParameterName="DVOL_ENERGY_LOSS_2",
periodicityType=FullCycleIronLosses(),
sheetLoadModel=CasDefinedMaterial())
...
EvolutiveCurve2D(name="MAGNET LOSSES BASE SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
formula=["MAGNET_JOULE_LOSSES"]
...
EvolutiveCurve2D(name="JOULE LOSSES BASE SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
...
formula=["PjoulECC(STRASE_1)+PjoulECC(STRASE_2)+PjoulECC(STRASE_3)"]
...
EvolutiveCurve2D(name="USER_POWER_BASE_SPEED",
evolutivePath=EvolutionaryPath(parameterSet={SetParameterXVariable(paramVol=VariationParameter
limitMin=6.666666157895,
limitMax=126.666666)})
```

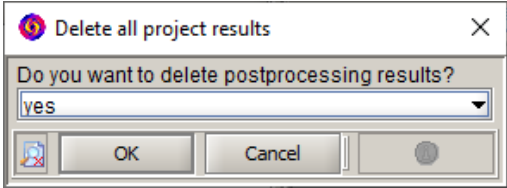
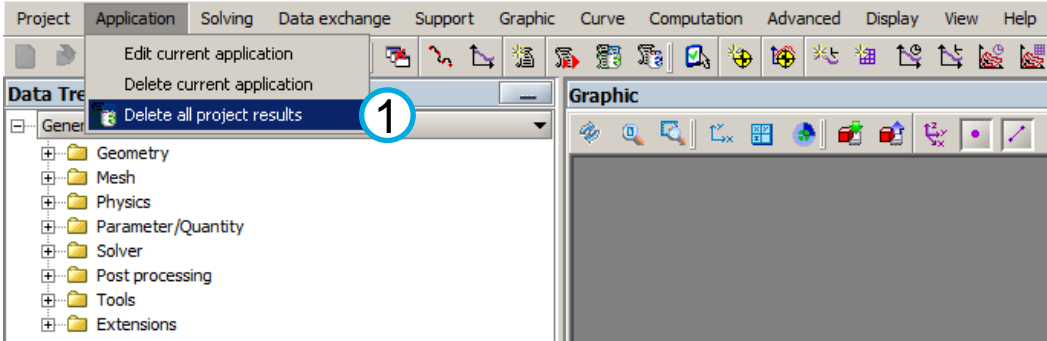
Attention: 1) the project will be solved automatically by HyperStudy;
 2) the Flux macro location is depended on the Flux installation path. Therefore, these scripts should be replaced by a generic way to adapted to all the user.

GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Generating HyperStudy connector

- Delete all project results

Step	Action
1	Click on [Application] – [Delete all project results]
2	Select [Yes], and click on [OK]

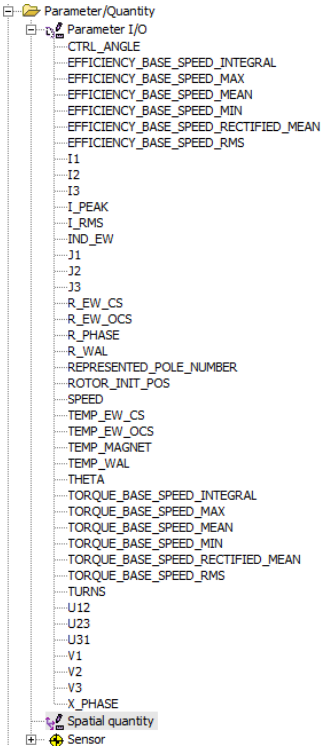
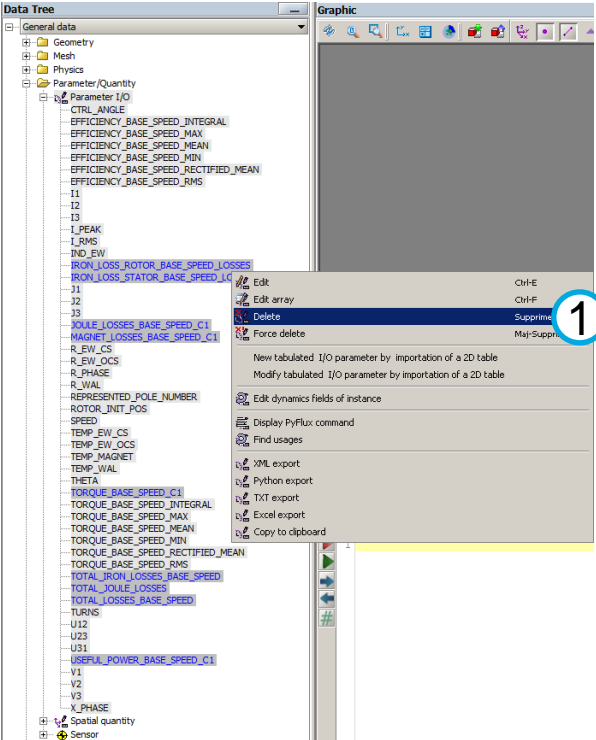
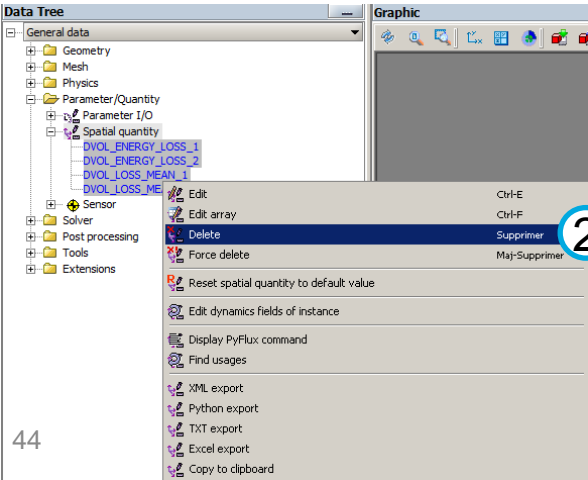


GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Generating HyperStudy connector

- Delete parameters and spatial quantities not needed in connector

Step	Action
1	Select the following 9 I/O parameters, right click and click on [Delete]
2	Select the 4 spatial quantities, right click and click on [Delete]

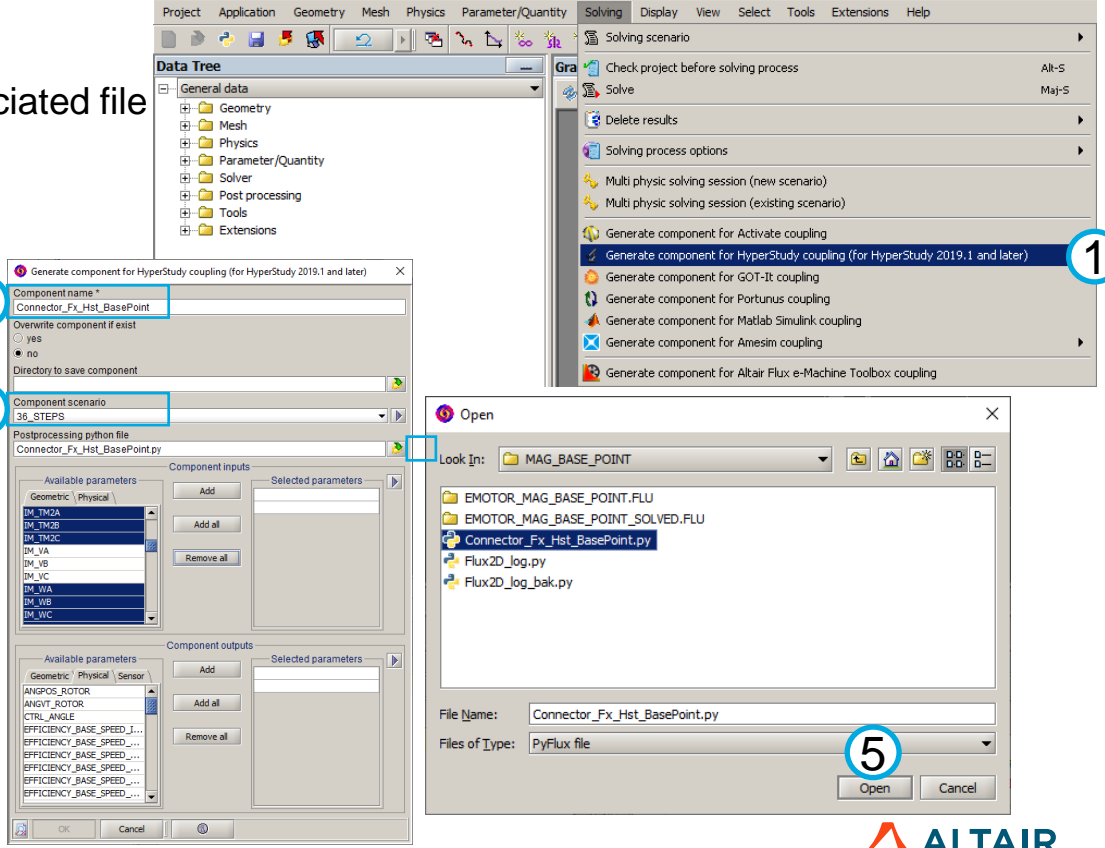


GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Generating HyperStudy connector

- Define connector name and associated file

Step	Action
1	Click on [Solving] – [Generate component for HyperStudy coupling (for HyperStudy 2019.1 and later)]
2	Define the component name as "Connector_Fx_Hst_BasePoint"
3	Directory by default
4	Define the component scenario "36_STEPS"
5	Select the "Connector_Fx_Hst_BasePoint.py" as the postprocessing Python file



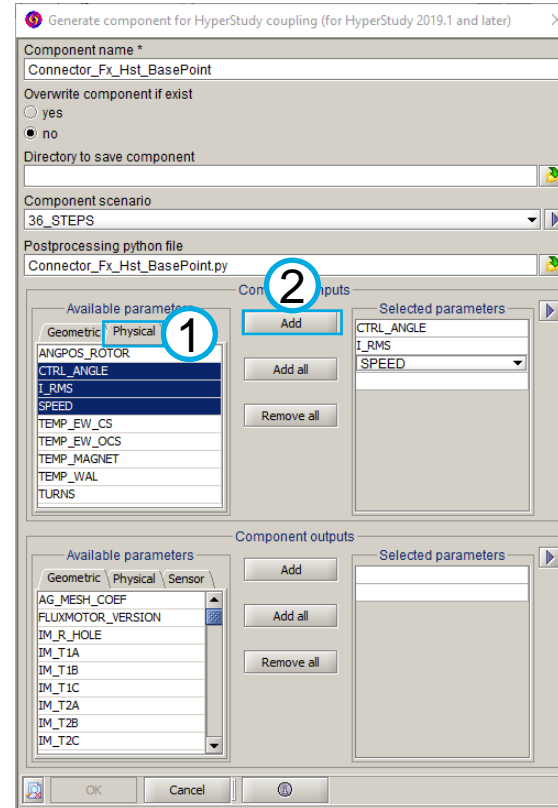
Connector name	Connector_Fx_Hst_BasePoint
Postprocessing Python file	Connector_Fx_Hst_BasePoint.py

GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Generating HyperStudy connector

- Define component inputs (physical)

Step	Action
1	Click on [Physical] tab, select the following three parameters: - CTRL_ANGLE - I_RMS - SPEED
2	Click on [Add]



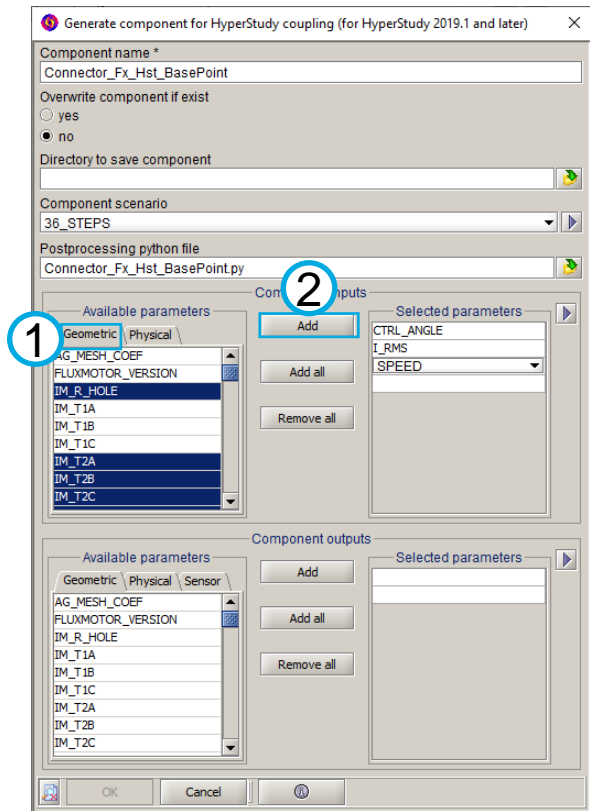
GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Generating HyperStudy connector

- Define component inputs (geometric)

Step	Action
1	Click on [Geometric] tab, select the following 22 parameters in the table
2	Click on [Add]

Input geometric variables	
IM_R_HOLE	IM_TM2B
IM_T2A	IM_TM2C
IM_T2B	IM_WA
IM_T2C	IM_WB
IM_T3A	IM_WC
IM_T3B	IM_WM1A
IM_T3C	IM_WM1B
IM_TM1A	IM_WM1C
IM_TM1B	IM_WM2A
IM_TM1C	IM_WM2B
IM_TM2A	IM_WM2C



GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

Generating HyperStudy connector

- Define component outputs

Step	Action
1	Click on [Physical] tab in the output option
2	Select the following parameters and click on [Add]

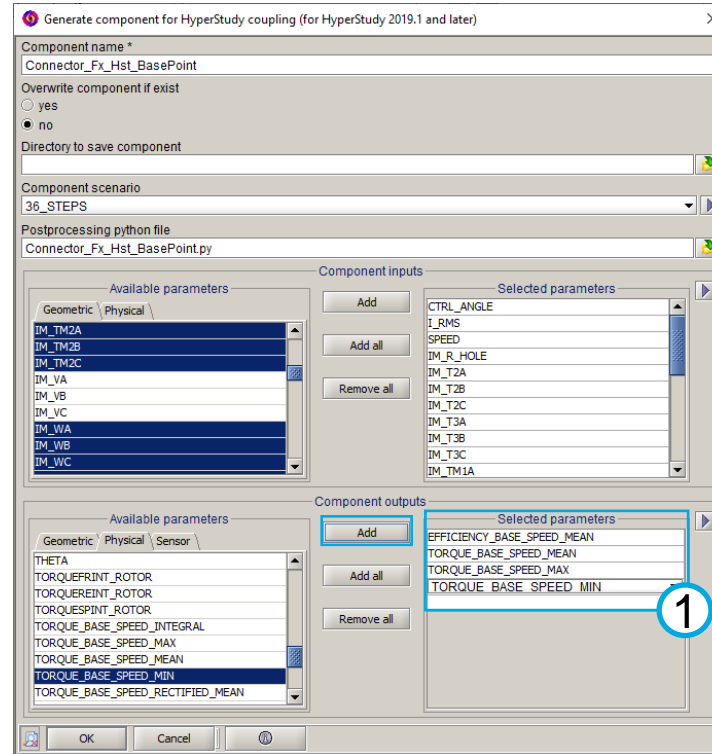
Output variables

EFFICIENCY_BASE_MEAN

TORQUE_BASE_SPEED_MEAN

TORQUE_BASE_SPEED_MAX

TORQUE_BASE_SPEED_MIN



GENERATING HYPERSTUDY CONNECTOR: BASE SPEED POINT

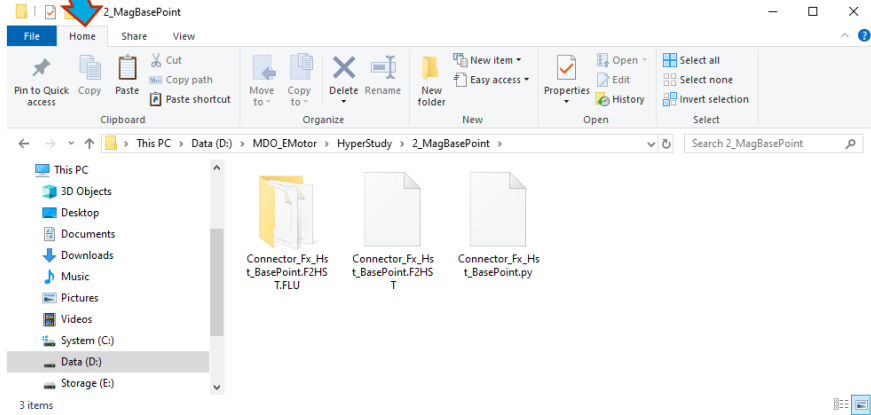
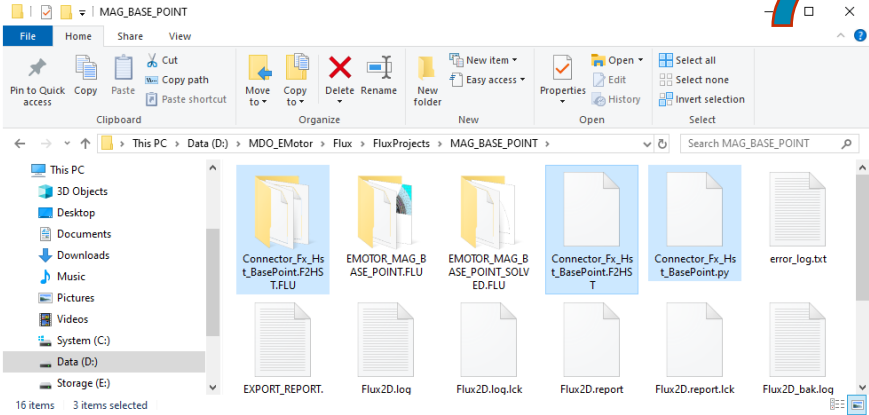
Generating HyperStudy connector

- Exchange files for the Flux / HyperStudy connector

Step	Action
1	Copy and paste the connector files: - Connector_Fx_Hst_BasePoint.F2HST - Connector_Fx_Hst_BasePoint.F2HST.FLU - Connector_Fx_Hst_BasePoint.py to the HyperStudy project folder

Path for saving the Flux magnetic analysis connector 1

~\MDO_EMotor\HyperStudy\2_BasePoint



MAGNETIC ANALYSIS: SPECIFIC OPERATING POINT

OUTLINE

Solving and
postprocessing:
specific operating point

Generating
HyperStudy connector:
specific operating point

Input file

- **Flux 2D project:**
- EMOTOR_MAG_OPERATING_POINT.FLU

Software

- Altair Flux 2019.1 (or later version)

Output documents

- **Flux / HyperStudy connector:**
Connector_Fx_Hst_OperatingPoint.F2HST
- **Flux project associated with the connector**
Connector_Fx_Hst_OperatingPoint.F2HST.FLU
- **Python script for the postprocessing in HyperStudy**
Connector_Fx_Hst_OperatingPoint.py

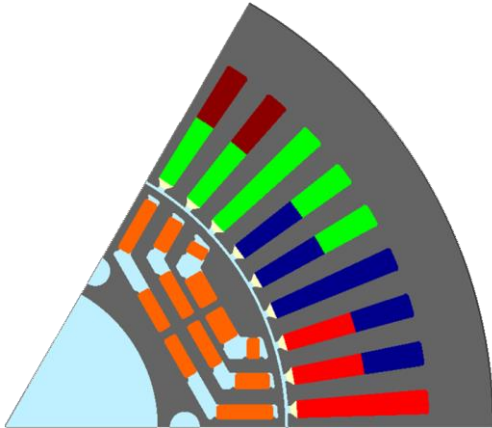
SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Solving the magnetic problem at specific operating point

- Open Flux 2D project

Step	Action
1	Open the project "EMOTOR_MAG_OPERATING_POINT.FLU"



Altair Flux™ 2D Skew 3D PEEC

How to proceed?

- Select the working module (2D, Skew, 3D or PEEC)
- Select the working directory
- Select the Flux project in the current projects
- Click on "Open the selected project" (or double-click directly on the selected project)

Working directory: D:\MDO_Emotor\Flux\FluxProjects\MAG_OPERATING_POINT

Current projects			
Name	Size	Date	Type
EMOTOR_MAG_OPERATING_POINT.FLU	5.75 MB	2018/11/23 5:22 PM	Flux 2D project

Recent projects			
Name	Size	Date	Type
EMOTOR_MAG_OPERATING_POINT.FLU	5.75 MB	2018/11/23 5:21 PM	Flux 2D project
EMOTOR_MAG_BASE_POINT.FLU	5.757 MB	2018/11/23 5:17 PM	Flux 2D project
EMOTOR_MAG_BASE_POINT_SOLVED.FLU	6.305 MB	2018/11/22 2:11 PM	Flux 2D project
Thermal_HISTORY.FLU	20.746 MB	2018/11/18 3:31 PM	Flux 2D project

EMOTOR_MAG_OPERATING_POINT.FLU

Application: Transient Magnetic 2D
 Status: Not solved
 Version: 15.1.0
 Comment:
 Enter your comment

Physical memory: 7.56 GB(93.76 GB) Available memory: 9.71 GB(64.76 GB) Disk space: 108.16 GB(709.73 GB)

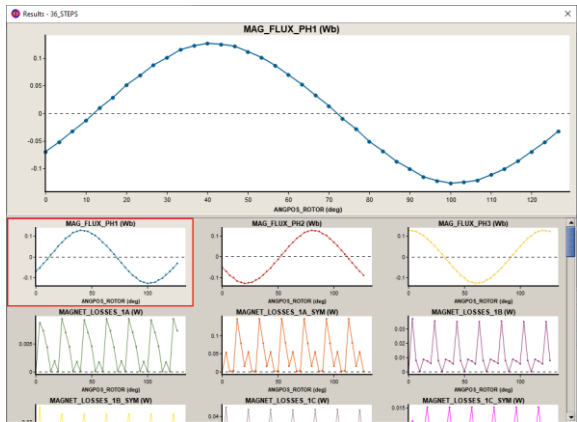
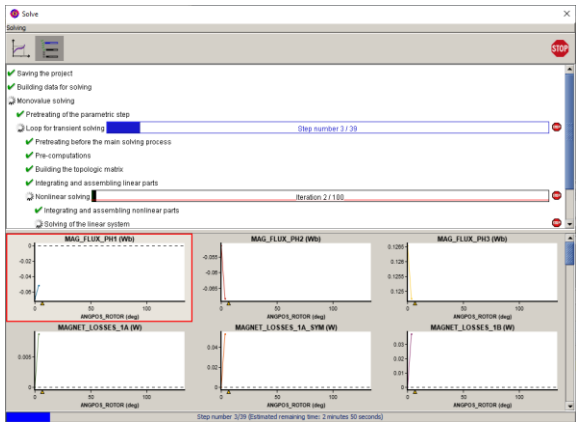
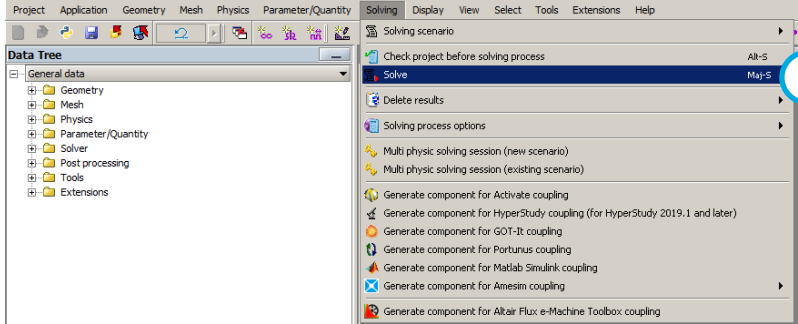
Project name	EMOTOR_MAG_OPERATING_POINT.FLU
Project folder	~/MDO_EMOTOR/Flux/FluxProjects/ MAG_OPERATING_POINT

SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Solving the magnetic problem at specific operating point

- Solve the project

Step	Action
1	Click on [Solving] – [Solve]
2	Select the solving scenario “36_STEPS”
3	Save the solved project as a new project “EMOTOR_MAG_OPERATING_POINT_SOLVED.FLU”
4	Click on [OK]



2

3

54

4

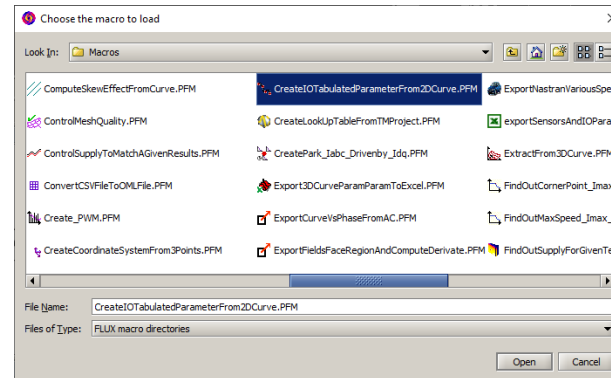
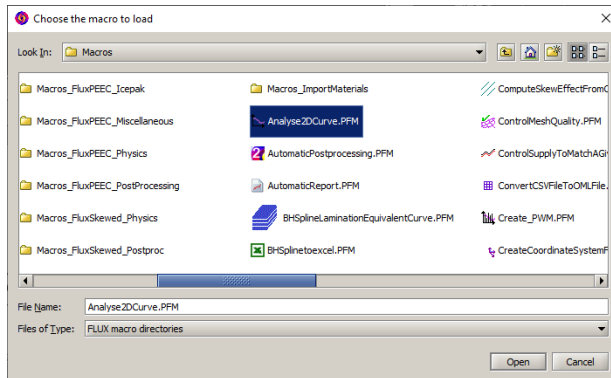
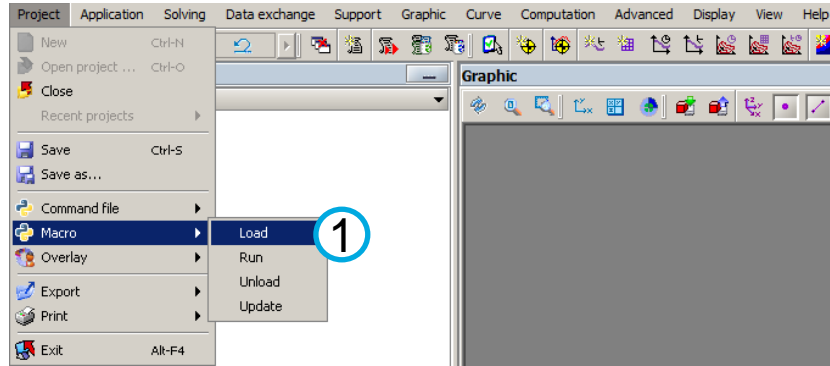
R

SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: postprocessing initialization

- Load Flux macros

Step	Action
1	Click on [Project] – [Macro] – [Load]
2	Load the following two macros: - Analyse2DCurve - CreateIOTabulatedParameterFrom2DCurve



SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: Joule losses at the specific operating point



- Plot the Joule loss curve

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create a new 2D curve JOULE_LOSSES_OPERATING_POINT Limit min: 6.666666 Limit max: 126.666666
3	Define the plotting formula as $P_{\text{jouleCC}}(\text{PHASE}_1) + P_{\text{jouleCC}}(\text{PHASE}_2) + P_{\text{jouleCC}}(\text{PHASE}_3)$
4	Click on [OK]

Curve name	JOULE_LOSSES_OPERATING_POINT
Formula	$P_{\text{jouleCC}}(\text{PHASE}_1) + P_{\text{jouleCC}}(\text{PHASE}_2) + P_{\text{jouleCC}}(\text{PHASE}_3)$

1

2

3

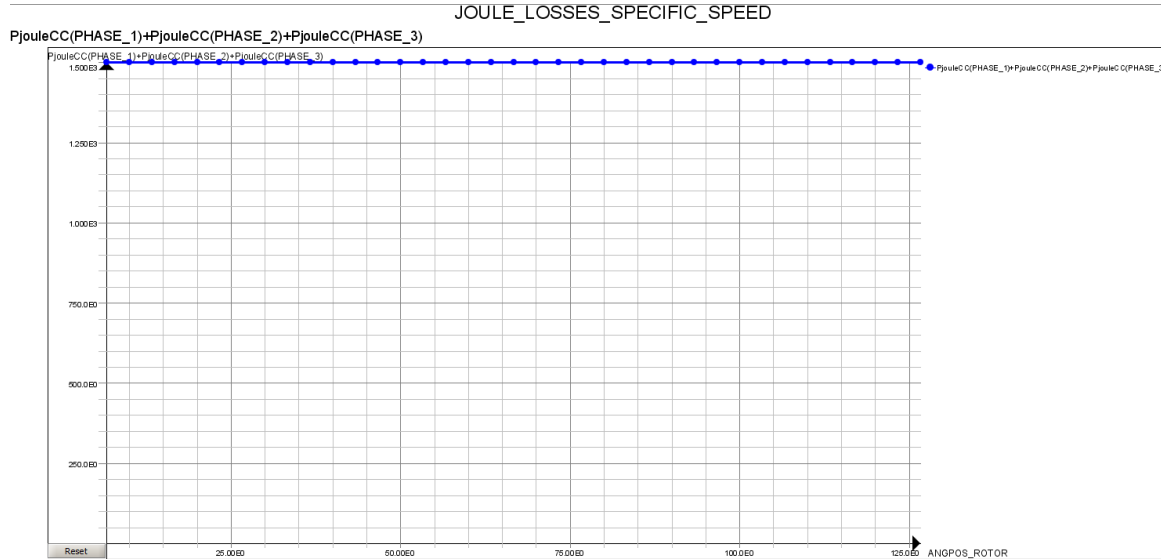
4

SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: Joule losses at the specific operating point



- Plot the Joule loss curve



SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: stator iron losses at the specific operating point



- Plot the stator iron loss curve

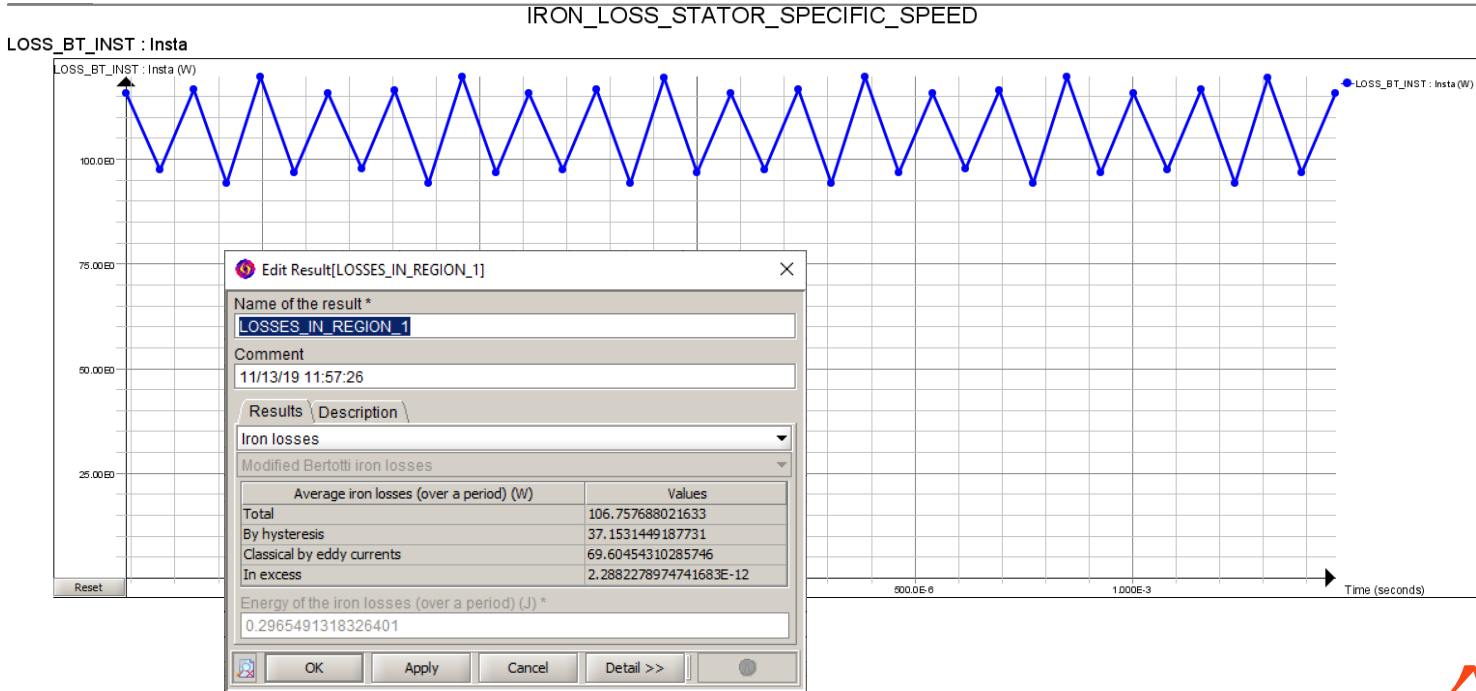
Step	Action
1	Click on [Computation] – [Computation of iron losses] – [Computation of iron losses]
2	Define the computation configuration in tab [Definition]: Face region: - OS_TOOTH - OS_TOOTH_FOOT - OS_YOKE Interval: [6.6666, 126.6666]
3	Define the computation configuration in tab [Results]: Result curve name: IRON_LOSS_STATOR_OPERATING_POINT
4	Click on [OK]

SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: stator iron losses at the specific operating point



- Plot the stator iron loss curve



SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: rotor iron losses at the specific operating point



- Plot the rotor iron loss curve

Step	Action
1	Click on [Computation] – [Computation of iron losses] – [Computation of iron losses]
2	Define the computation configuration in tab [Definition]: Face region: - IM_YOKE Interval: [6.6666, 126.6666]
3	Define the computation configuration in tab [Results]: Result curve name: IRON_LOSS_ROTOR_OPERATING_POINT
4	Click on [OK]

SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: rotor iron losses at the specific operating point



- Plot the rotor iron loss curve

Edit Result[LOSSES_IN_REGION_2]

Name of the result *
LOSSES_IN_REGION_2

Comment
11/13/19 11:59:43

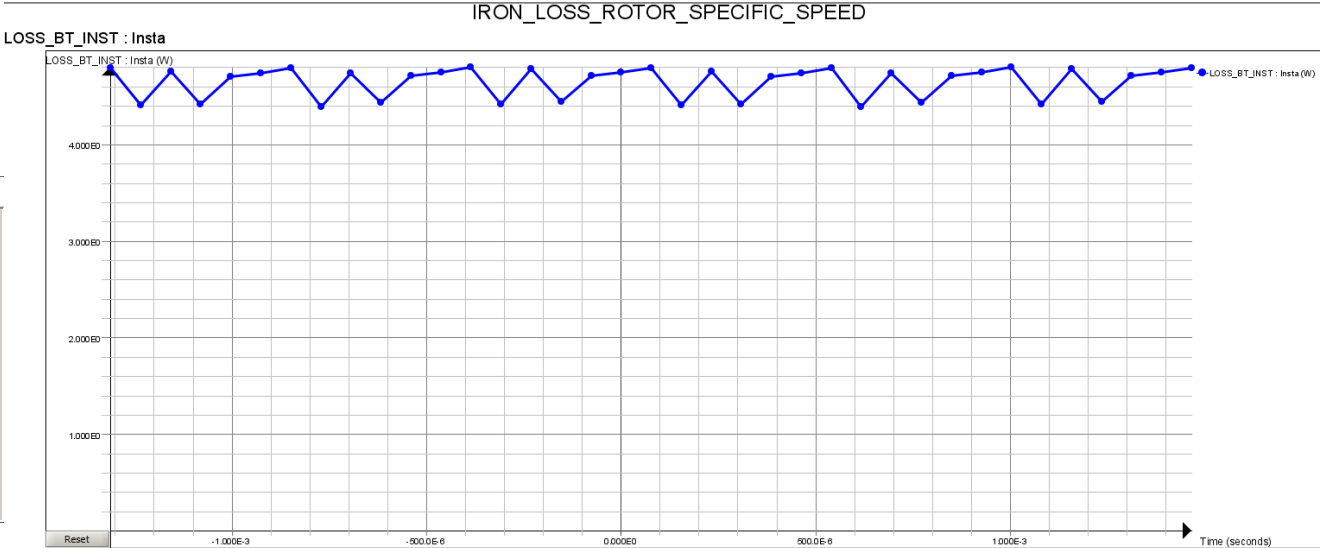
Results \ Description
Iron losses

Modified Bertotti iron losses

Average iron losses (over a period) (W)	Values
Total	4.644410852419666
By hysteresis	0.06003508164872695
Classical by eddy currents	4.584375770770241
In excess	6.880767455038564E-13

Energy of the iron losses (over a period) (J) *
0.012901141188820553

OK Apply Cancel Detail >>



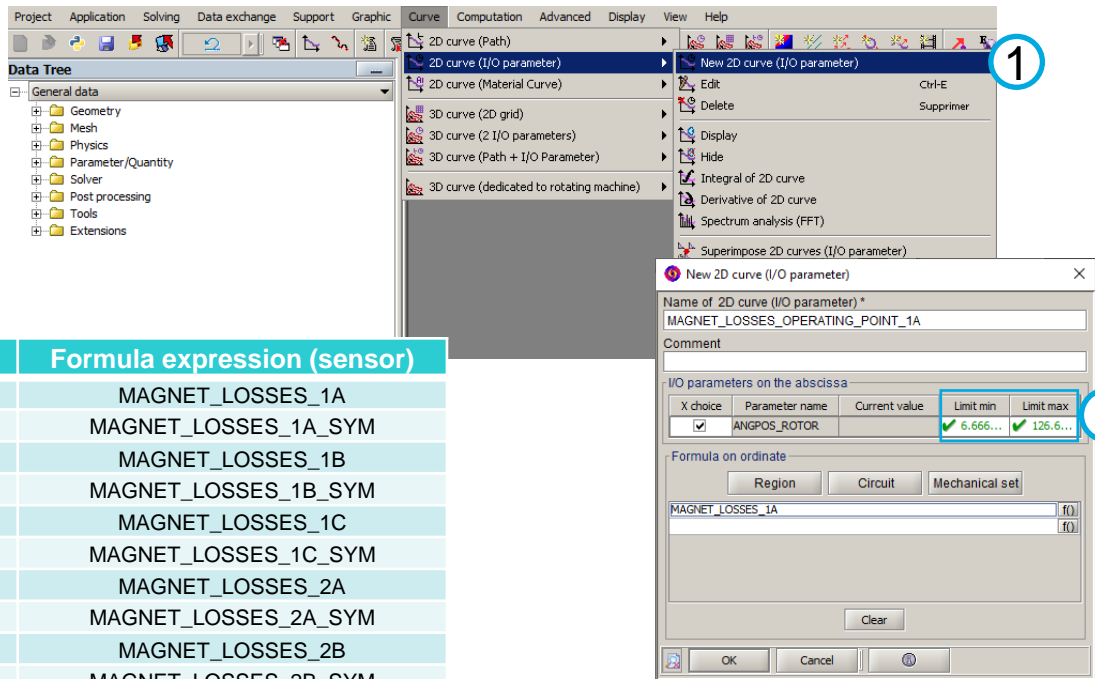
SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: magnet losses at the specific operating point



- Plot magnet loss curves for all predefined magnet regions

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [New 2D curve (I/O parameter)]
2	Create new 2D curves with the following interval - Limit min: 6.666666 - Limit max: 126.666666



Curve name	Formula expression (sensor)
MAGNET_LOSSES_OPERATING_POINT_1A	MAGNET_LOSSES_1A
MAGNET_LOSSES_OPERATING_POINT_1A_SYM	MAGNET_LOSSES_1A_SYM
MAGNET_LOSSES_OPERATING_POINT_1B	MAGNET_LOSSES_1B
MAGNET_LOSSES_OPERATING_POINT_1B_SYM	MAGNET_LOSSES_1B_SYM
MAGNET_LOSSES_OPERATING_POINT_1C	MAGNET_LOSSES_1C
MAGNET_LOSSES_OPERATING_POINT_1C_SYM	MAGNET_LOSSES_1C_SYM
MAGNET_LOSSES_OPERATING_POINT_2A	MAGNET_LOSSES_2A
MAGNET_LOSSES_OPERATING_POINT_2A_SYM	MAGNET_LOSSES_2A_SYM
MAGNET_LOSSES_OPERATING_POINT_2B	MAGNET_LOSSES_2B
MAGNET_LOSSES_OPERATING_POINT_2B_SYM	MAGNET_LOSSES_2B_SYM
MAGNET_LOSSES_OPERATING_POINT_2C	MAGNET_LOSSES_2C
MAGNET_LOSSES_OPERATING_POINT_2C_SYM	MAGNET_LOSSES_2C_SYM

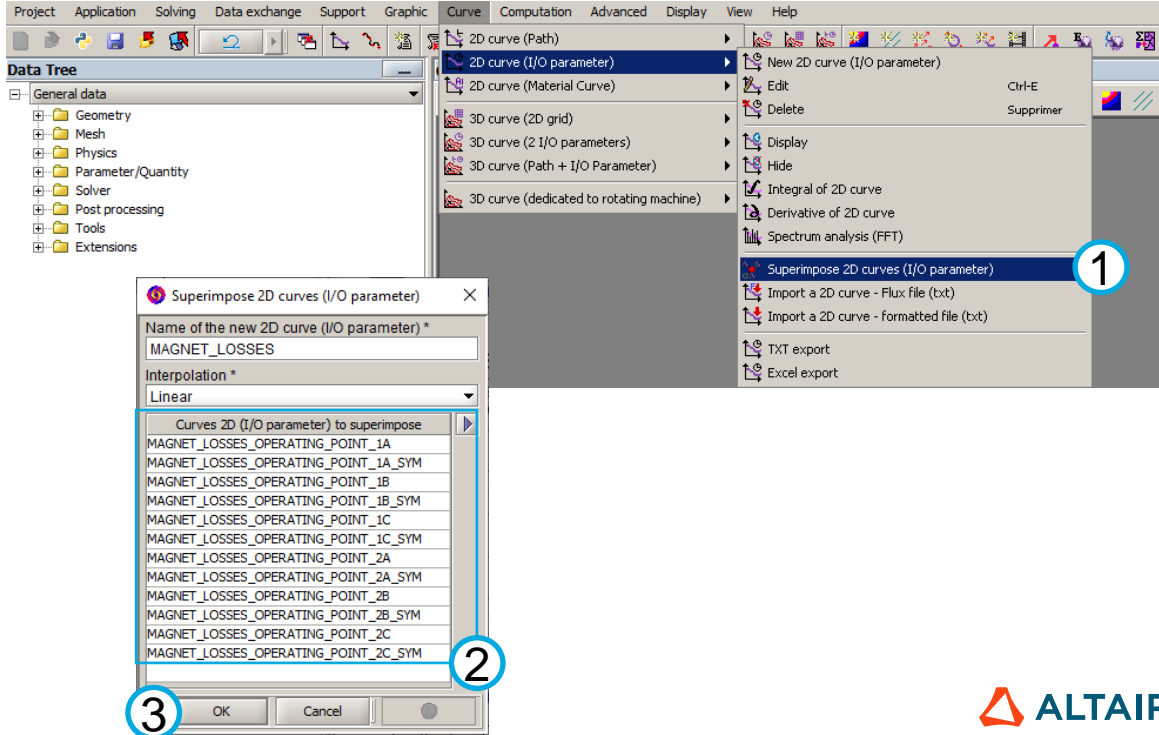
SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: magnet losses at the specific operating point



- Plot magnet loss curves for all predefined magnet regions

Step	Action
1	Click on [Curve] – [2D curve (I/O parameter)] – [Superimpose 2D curves ((I/O parameter)]
2	Select all the magnet loss curves
3	Click on [OK]



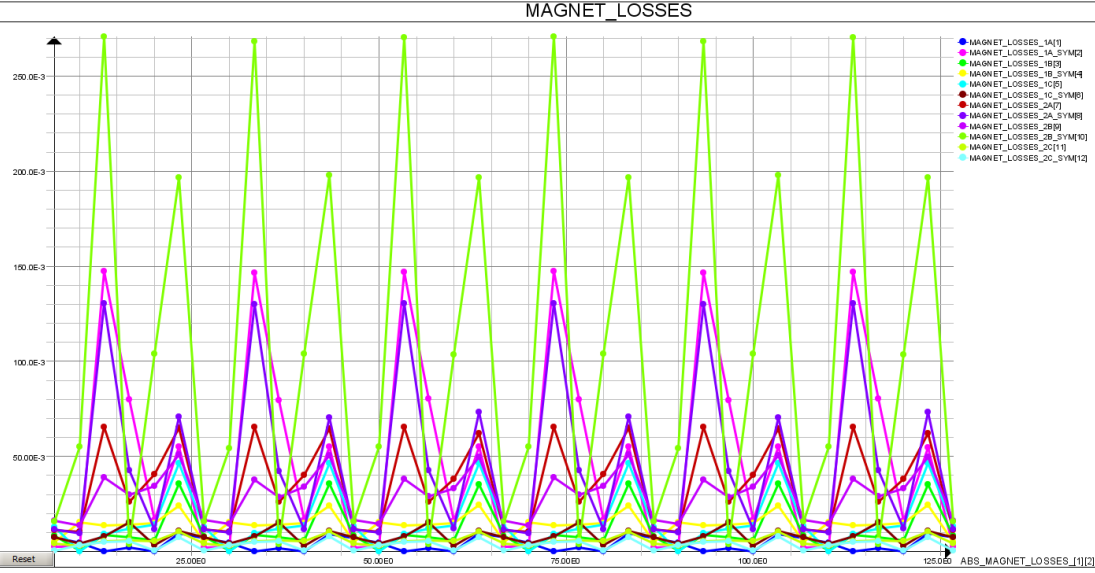
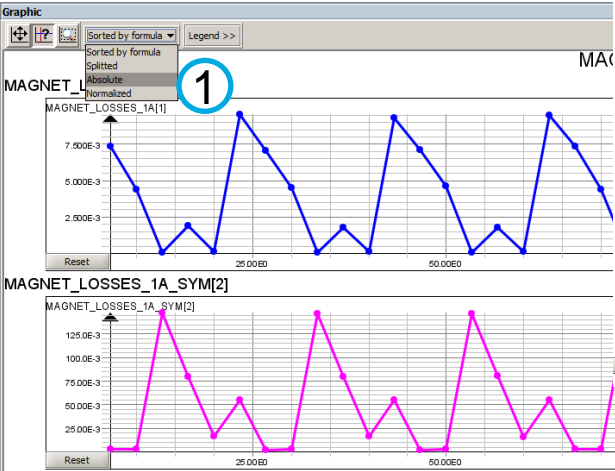
SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

Magnetic analysis: magnet losses at the specific operating point



- Plot magnet loss curves for all predefined magnet regions

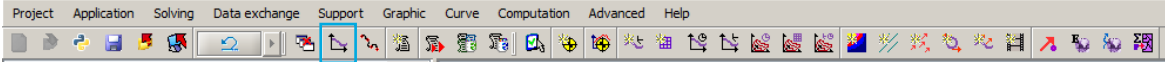
Step	Action
1	Select the plot configuration as "Absolute"




SOLVING AND POSTPROCESSING: SPECIFIC OPERATING POINT

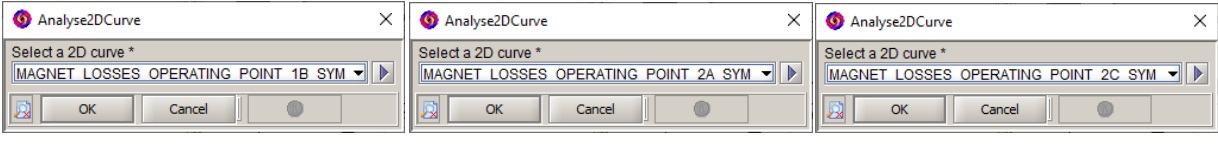
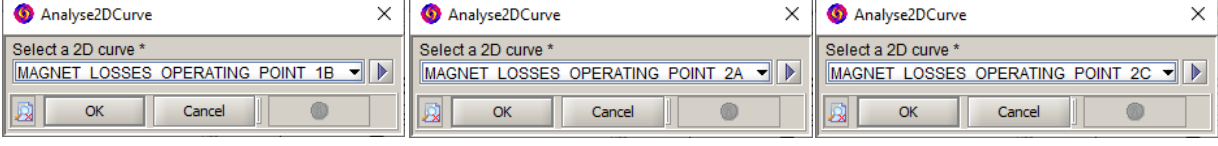
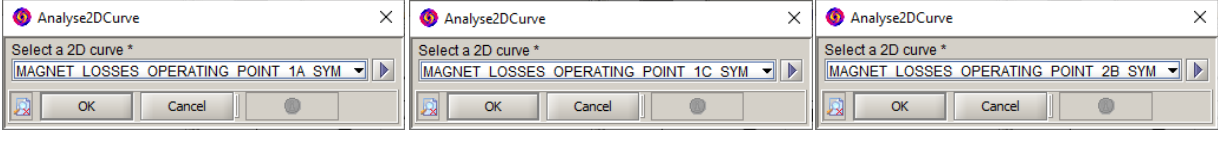
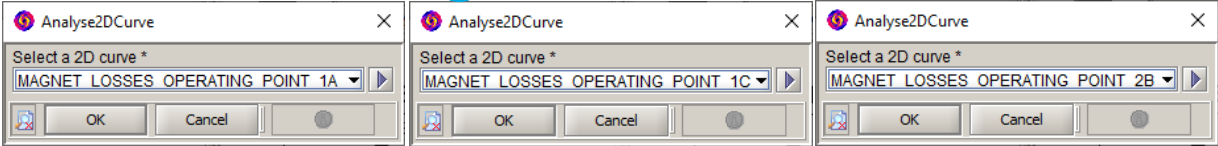
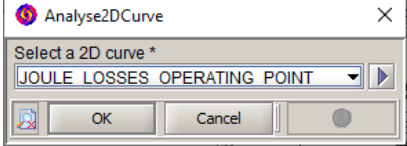
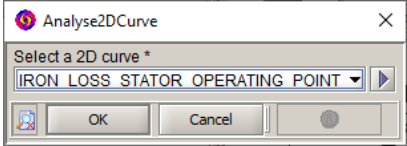
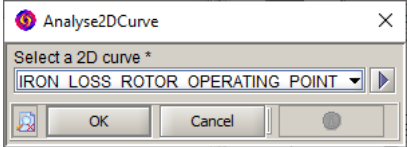
Magnetic analysis

- Analyze the curves with macros



1

Step	Action
1	Click on the icon  to run the macro "Analyse2DCurve", and analyze respectively ALL the created curves (3+12=15)

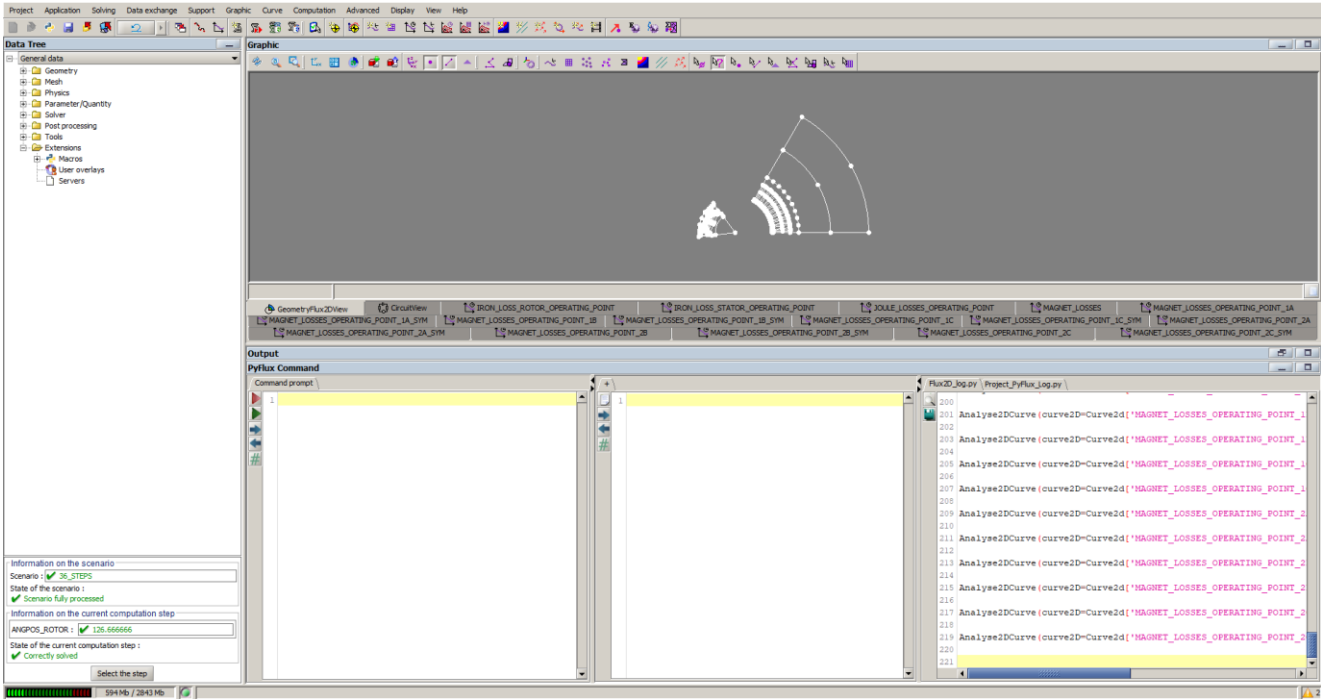


Attention: Do NOT select the superimpose curves "MAGNET_LOSSES"

GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT


Preparing post-processing Python script for HyperStudy connector

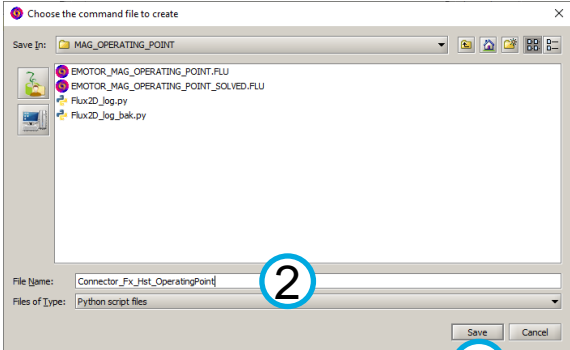
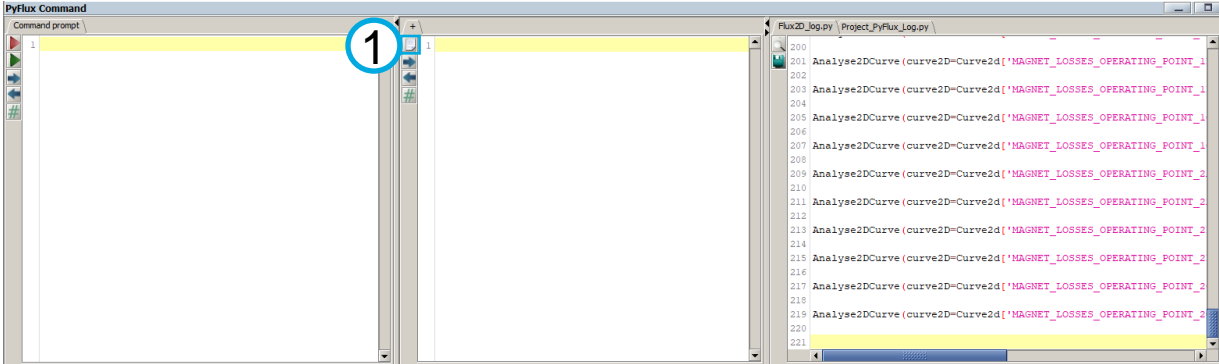


GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Preparing post-processing Python script for HyperStudy connector

- Create a new Python script

Step	Action
1	Click on the icon  to create a new Python script file
2	Define the script name and the saving folder
3	Click on [Save]




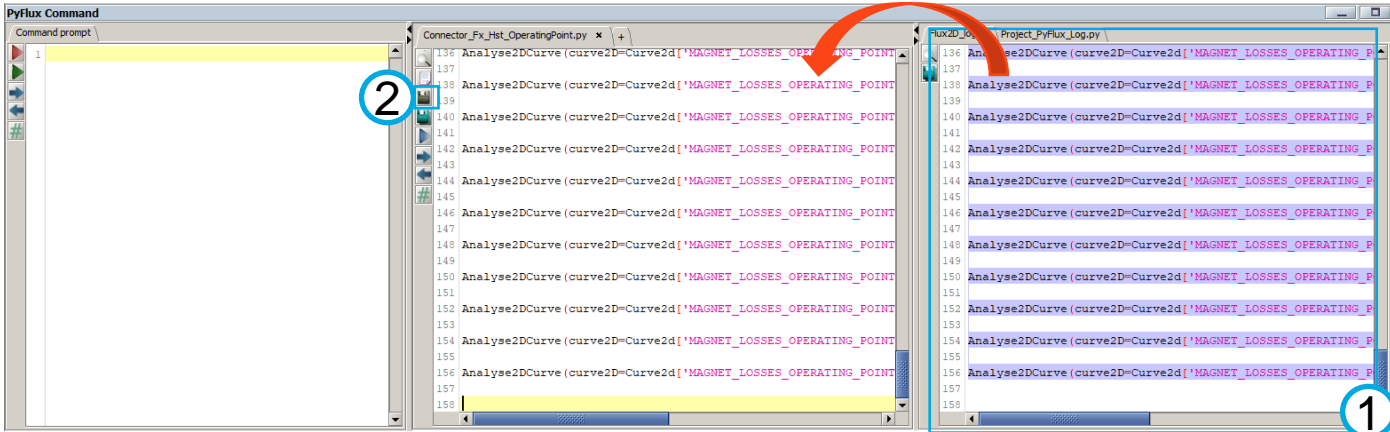
Script name	Connector_Fx_Hst_OperatingPoint
Saving folder	~/MDO_EMOTOR/Flux/FluxProjects/ MAG_OPERATING_POINT

GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Preparing post-processing Python script for HyperStudy connector

- Create backup of the log script

Step	Action
1	Copy all commandes from the Flux2D_log.py file into the created Python script
2	Click on the icon  to save the script
3	Click on [Project] – [Exit]



GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Preparing post-processing Python script for HyperStudy connector

- Replace Flux macro import commands

Step	Action
1	Open the Python script by Notepad editor
2	Delete the scripts before post-processing
3	Insert the following scripts before post-processing

Load macros

```
import os
INSTALL = os.environ["INSTALLFLUX"] + "\\."
macroFile1 =
INSTALL+"Extensions\Macros\CreateOTabulatedParameterFrom2D
Curve.PFM"
macroFile2 = INSTALL+"Extensions\Macros\Analyse2DCurve.PFM"

loadMacro(fileName=macroFile1)
loadMacro(fileName=macroFile2)
```

```
##### Load macros #####
import os
INSTALL = os.environ["INSTALLFLUX"] + "\\."
macroFile1 =
INSTALL+"Extensions\Macros\CreateOTabulatedParameterFrom2DCurve.PFM"
macroFile2 = INSTALL+"Extensions\Macros\Analyse2DCurve.PFM"
loadMacro(fileName=macroFile1)
loadMacro(fileName=macroFile2)

EvolutiveCurve2D(name="VOUL_LOSSES_OPERATING_POINT",
evolutionPath=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.6666663157895,
limitMax=126.6666664)},
formula=["PjouleCC(PHASE_1)+PjouleCC(PHASE_2)+PjouleCC(PHASE_3)"])
result = IronLossesRegion(faceRegions={RegionFace["OS_TOOTH"],
RegionFace["OS_TOOTH_ROT"],
RegionFace["OS_TOOTH_1"]},
timeInterval=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.6666663157895,
limitMax=126.6666664)},
curveName="IRON_LOSS_STATOR_OPERATING_POINT",
resultName="LOSSES_IN_REGION_1",
lossSpatialParametersName="VOUL_LOSS_NEAR_1",
energySpatialParametersName="VOUL_ENERGY_LOSS_1",
periodicity=FullyCyclicIronLosses(),
sheetLossModel=CaseDefinedMaterial())
result = IronLossesRegion(faceRegions={RegionFace["IM_TOOTH"],
timeInterval=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.6666663157895,
limitMax=126.6666664)},
curveName="IRON_LOSS_ROTOR_OPERATING_POINT",
resultName="LOSSES_IN_REGION_2",
lossSpatialParametersName="VOUL_LOSS_NEAR_2",
energySpatialParametersName="VOUL_ENERGY_LOSS_2",
periodicity=FullyCyclicIronLosses(),
sheetLossModel=CaseDefinedMaterial())
EvolutiveCurve2D(name="MAGNET_LOSSES_OPERATING_POINT_1A",
evolutionPath=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.666666,
limitMax=133.3333}),
formula=["MAGNET_LOSSES_1A"])
EvolutiveCurve2D(name="MAGNET_LOSSES_OPERATING_POINT_1A_SYM",
evolutionPath=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
```

```
##### Load macros #####
import os
INSTALL = os.environ["INSTALLFLUX"] + "\\."
macroFile1 =
INSTALL+"Extensions\Macros\CreateOTabulatedParameterFrom2DCurve.PFM"
macroFile2 = INSTALL+"Extensions\Macros\Analyse2DCurve.PFM"
loadMacro(fileName=macroFile1)
loadMacro(fileName=macroFile2)

EvolutiveCurve2D(name="VOUL_LOSSES_OPERATING_POINT",
evolutionPath=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.6666663157895,
limitMax=126.6666664)},
formula=["PjouleCC(PHASE_1)+PjouleCC(PHASE_2)+PjouleCC(PHASE_3)"])
result = IronLossesRegion(faceRegions={RegionFace["OS_TOOTH"],
RegionFace["OS_TOOTH_ROT"],
RegionFace["OS_TOOTH_1"]},
timeInterval=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.6666663157895,
limitMax=126.6666664)},
curveName="IRON_LOSS_STATOR_OPERATING_POINT",
resultName="LOSSES_IN_REGION_1",
lossSpatialParametersName="VOUL_LOSS_NEAR_1",
energySpatialParametersName="VOUL_ENERGY_LOSS_1",
periodicity=FullyCyclicIronLosses(),
sheetLossModel=CaseDefinedMaterial())
result = IronLossesRegion(faceRegions={RegionFace["IM_TOOTH"],
timeInterval=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.6666663157895,
limitMax=126.6666664)},
curveName="IRON_LOSS_ROTOR_OPERATING_POINT",
resultName="LOSSES_IN_REGION_2",
lossSpatialParametersName="VOUL_LOSS_NEAR_2",
energySpatialParametersName="VOUL_ENERGY_LOSS_2",
periodicity=FullyCyclicIronLosses(),
sheetLossModel=CaseDefinedMaterial())
EvolutiveCurve2D(name="MAGNET_LOSSES_OPERATING_POINT_1A",
evolutionPath=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
VariationParameter["ANGPOS_ROTOR"],
limitMin=6.666666,
limitMax=133.3333}),
formula=["MAGNET_LOSSES_1A"])
EvolutiveCurve2D(name="MAGNET_LOSSES_OPERATING_POINT_1A_SYM",
evolutionPath=EvolutivePath(parametersSet={SetParameterXVariable(paramVol=
```

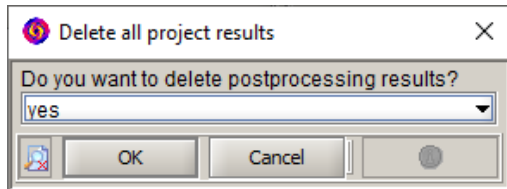
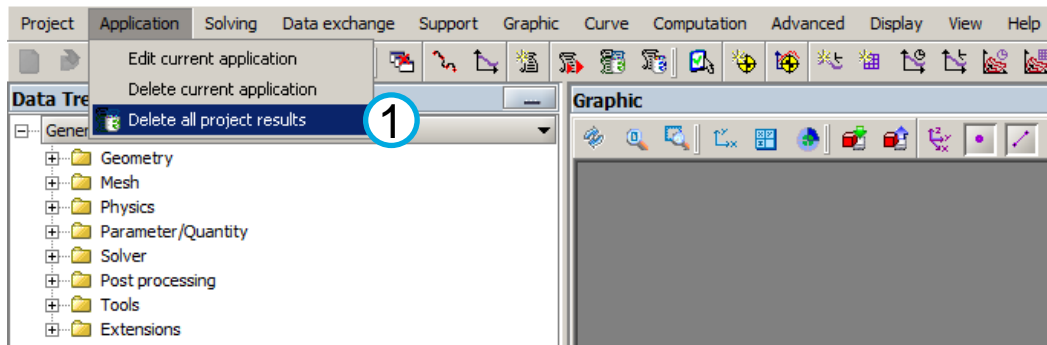
Attention: 1) the project will be solved automatically by HyperStudy;
 2) the Flux macro location is depended on the Flux installation path. Therefore, these scripts should be replaced by a generic way to adapted to all the user.

GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Generating HyperStudy connector

- Delete all project results

Step	Action
1	Click on [Application] – [Delete all project results]
2	Select [Yes], and click on [OK]

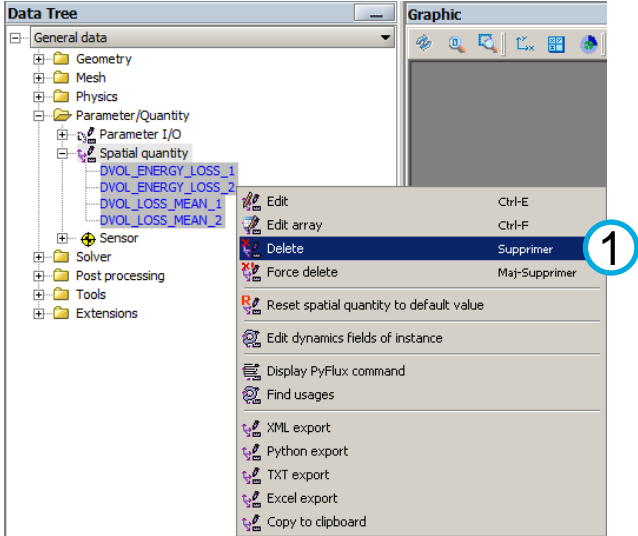


GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Generating HyperStudy connector

- Delete spatial quantities not needed in connector

Step	Action
1	Select the 4 spatial quantities, right click and click on [Delete]



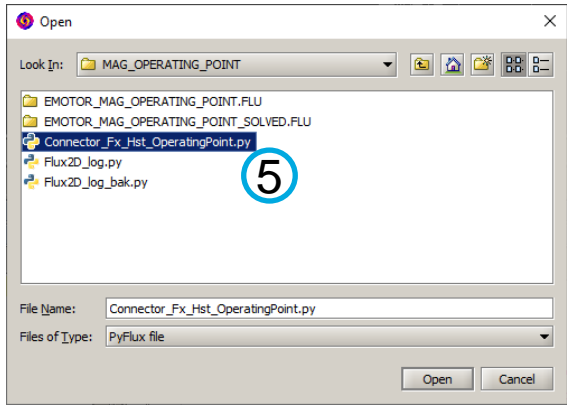
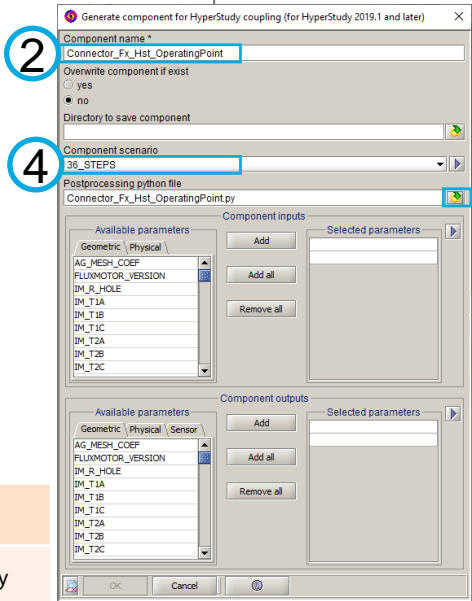
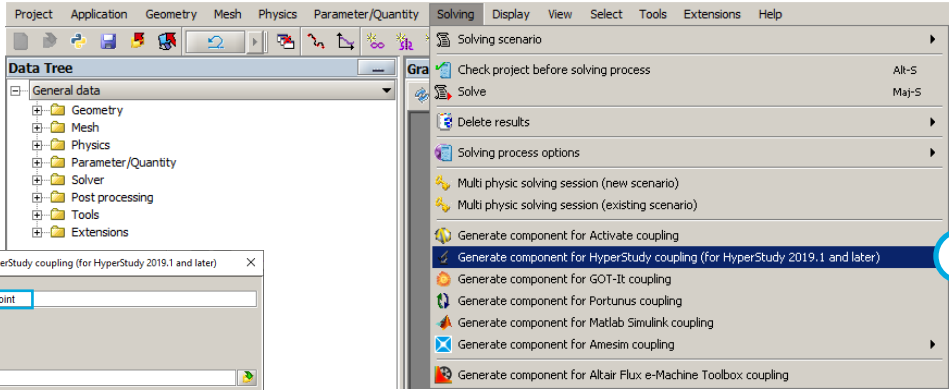
GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Generating HyperStudy connector

- Define connector name and associated file

Step	Action
1	Click on [Solving] – [Generate component for HyperStudy coupling (for HyperStudy 2019.1 and later)]
2	Define the component name as “Connector_Fx_Hst_OperatingPoint”
3	Directory by default
4	Define the component scenario “36_STEPS”
5	Define the Postprocessing python file as “Connector_Fx_Hst_OperatingPoint .py”

Connector name	Connector_Fx_Hst_OperatingPoint
Postprocessing Python file	Connector_Fx_Hst_OperatingPoint.py

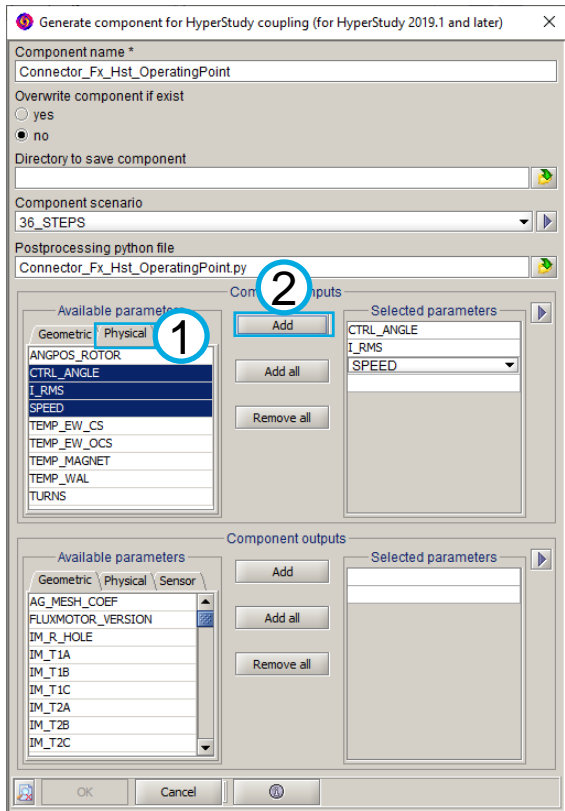


GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Generating HyperStudy connector

- Define component inputs (physical)

Step	Action
1	Click on [Physical] tab, select the following three parameters: - CTRL_ANGLE - I_RMS - SPEED
2	Click on [Add]



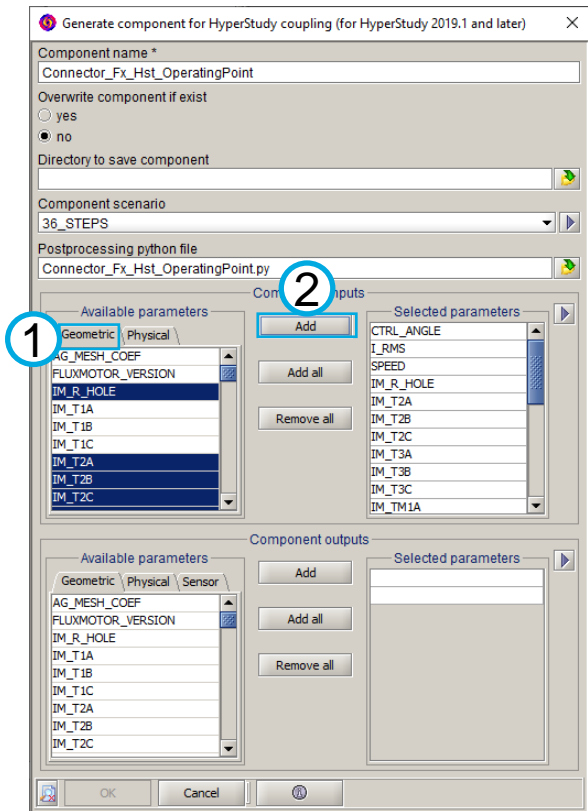
GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Generating HyperStudy connector

- Define component inputs (geometric)

Step	Action
1	Click on [Geometric] tab, select the following 22 parameters in the table
2	Click on [Add]

Input geometric variables	
IM_R_HOLE	IM_TM2B
IM_T2A	IM_TM2C
IM_T2B	IM_WA
IM_T2C	IM_WB
IM_T3A	IM_WC
IM_T3B	IM_WM1A
IM_T3C	IM_WM1B
IM_TM1A	IM_WM1C
IM_TM1B	IM_WM2A
IM_TM1C	IM_WM2B
IM_TM2A	IM_WM2C



GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

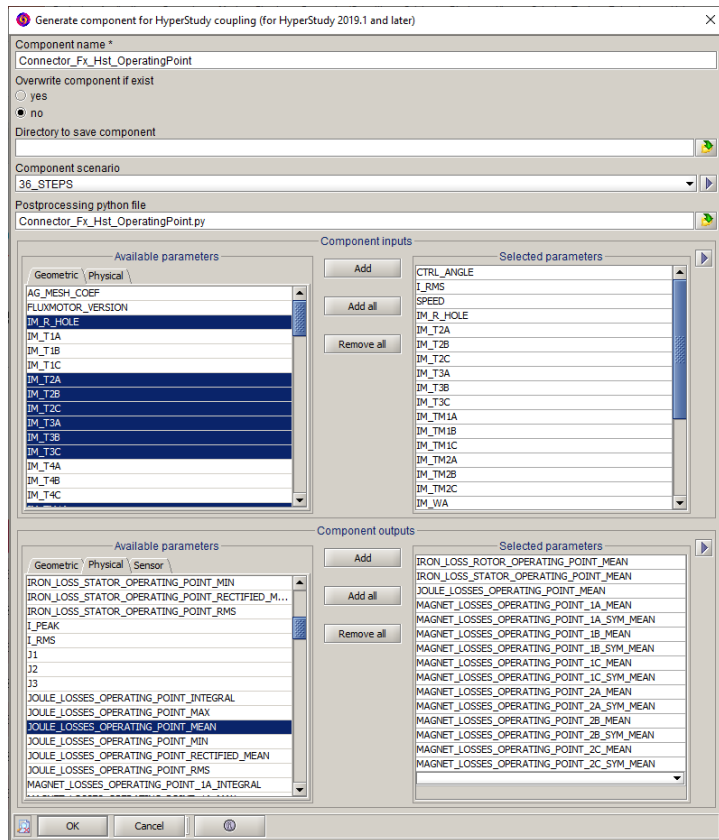
Generating HyperStudy connector

- Define component outputs

Step	Action
1	Select "Component outputs" 15 output variables: - 02 Iron losses - 01 Joule losses - 12 Magnet losses

Output variables

IRON_LOSSES_ROTOR_SPECIFIC_SPEED_MEAN	MAGNET_LOSSES_OPERATING_POINT_1C_SYM
IRON_LOSSES_STATOR_SPECIFIC_SPEED_MEAN	MAGNET_LOSSES_OPERATING_POINT_2A
JOULE_LOSSES_SPECIFIC_SPEED_MEAN	MAGNET_LOSSES_OPERATING_POINT_2A_SYM
MAGNET_LOSSES_OPERATING_POINT_1A	MAGNET_LOSSES_OPERATING_POINT_2B
MAGNET_LOSSES_OPERATING_POINT_1A_SYM	MAGNET_LOSSES_OPERATING_POINT_2B_SYM
MAGNET_LOSSES_OPERATING_POINT_1B	MAGNET_LOSSES_OPERATING_POINT_2C
MAGNET_LOSSES_OPERATING_POINT_1B_SYM	MAGNET_LOSSES_OPERATING_POINT_2C_SYM
MAGNET_LOSSES_OPERATING_POINT_1C	



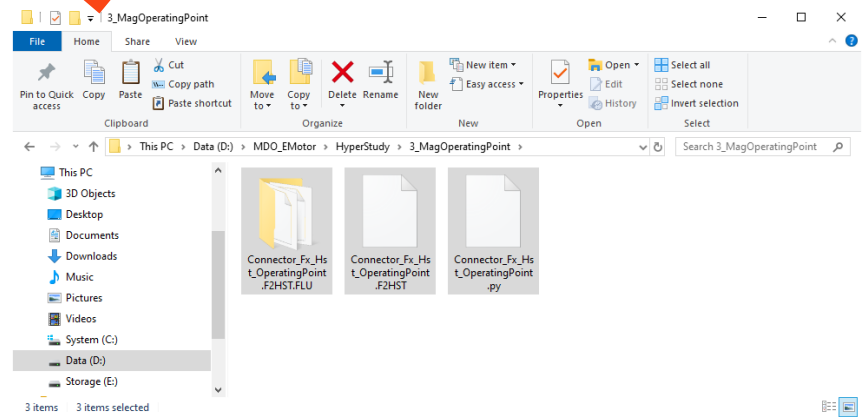
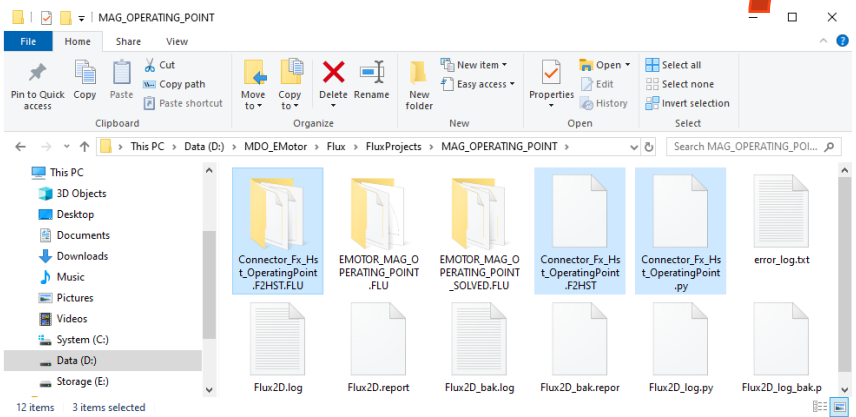
GENERATING HYPERSTUDY CONNECTOR: SPECIFIC OPERATING POINT

Generating HyperStudy connector

- Exchange files for the Flux / HyperStudy connector

Step	Action
1	Copy and paste the connector files: - Connector_Fx_Hst_OperatingPoint.F2HST - Connector_Fx_Hst_OperatingPoint.F2HST.FLU - Connector_Fx_Hst_OperatingPoint.py to the HyperStudy project folder

Path for saving the Flux magnetic analysis connector 2
~\MDO_EMotor\HyperStudy\2_OperatingPoint





THANK YOU

altair.com



#ONLYFORWARD