

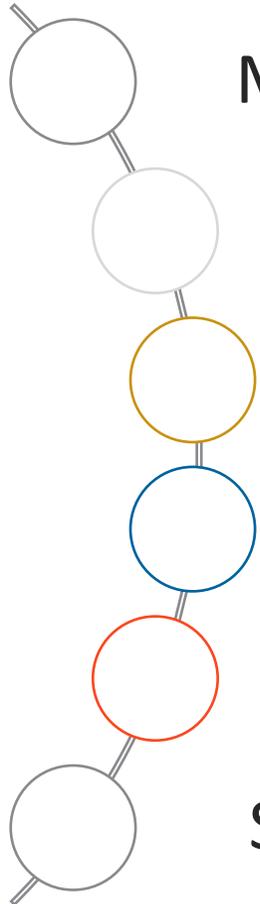


Powering Innovation That Drives Human Advancement

Adaptive Geometry Templates in Ansys Motor-CAD

Jonathan Godbehere, PhD

Agenda



Motor-CAD Templates and geometry customization

Introduction to Ansys Motor-CAD Adaptive Templates

Adaptive Template Creation

Workflows into other Ansys products

Getting started

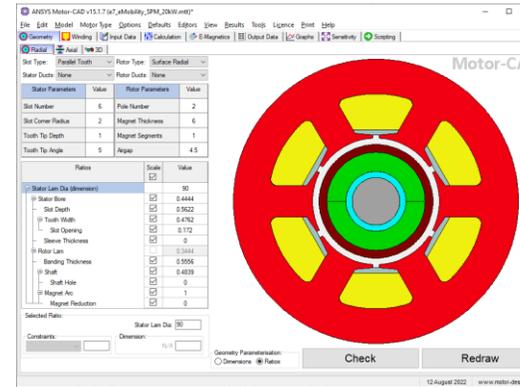
Summary



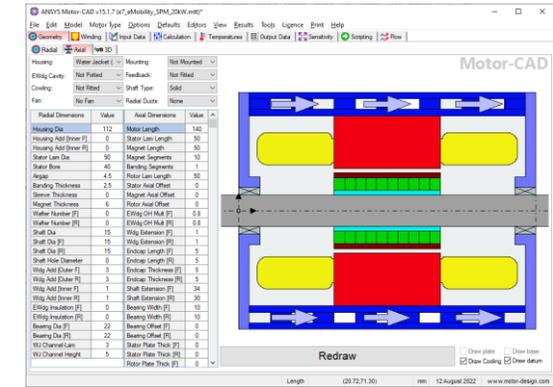
Ansys Motor-CAD Geometry Templates and Customizations

Geometry Templates – Advantages

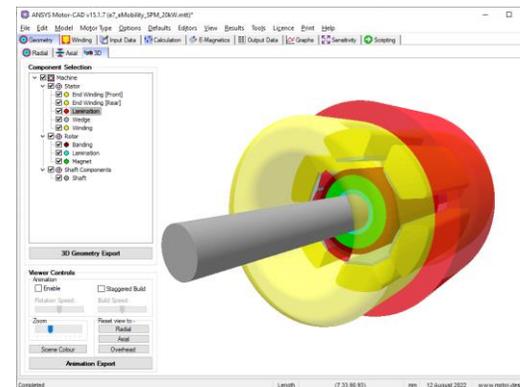
- Large range of parameterized templates: motor/rotor/slot type, rotor/stator ducts...
- Parameterization by dimensions and or by ratio for efficient design space exploration.
- Comprehensive model setup:
- Automated export into Ansys optiSLang for optimization



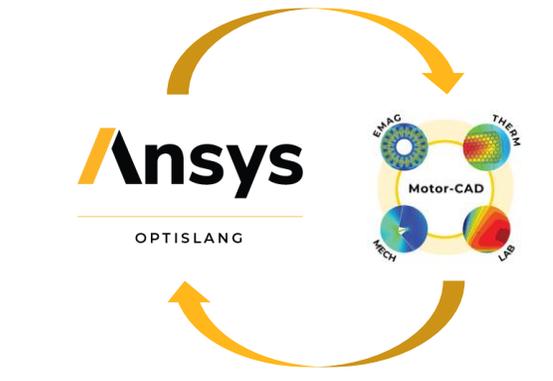
Radial cross section



Cooling Geometry



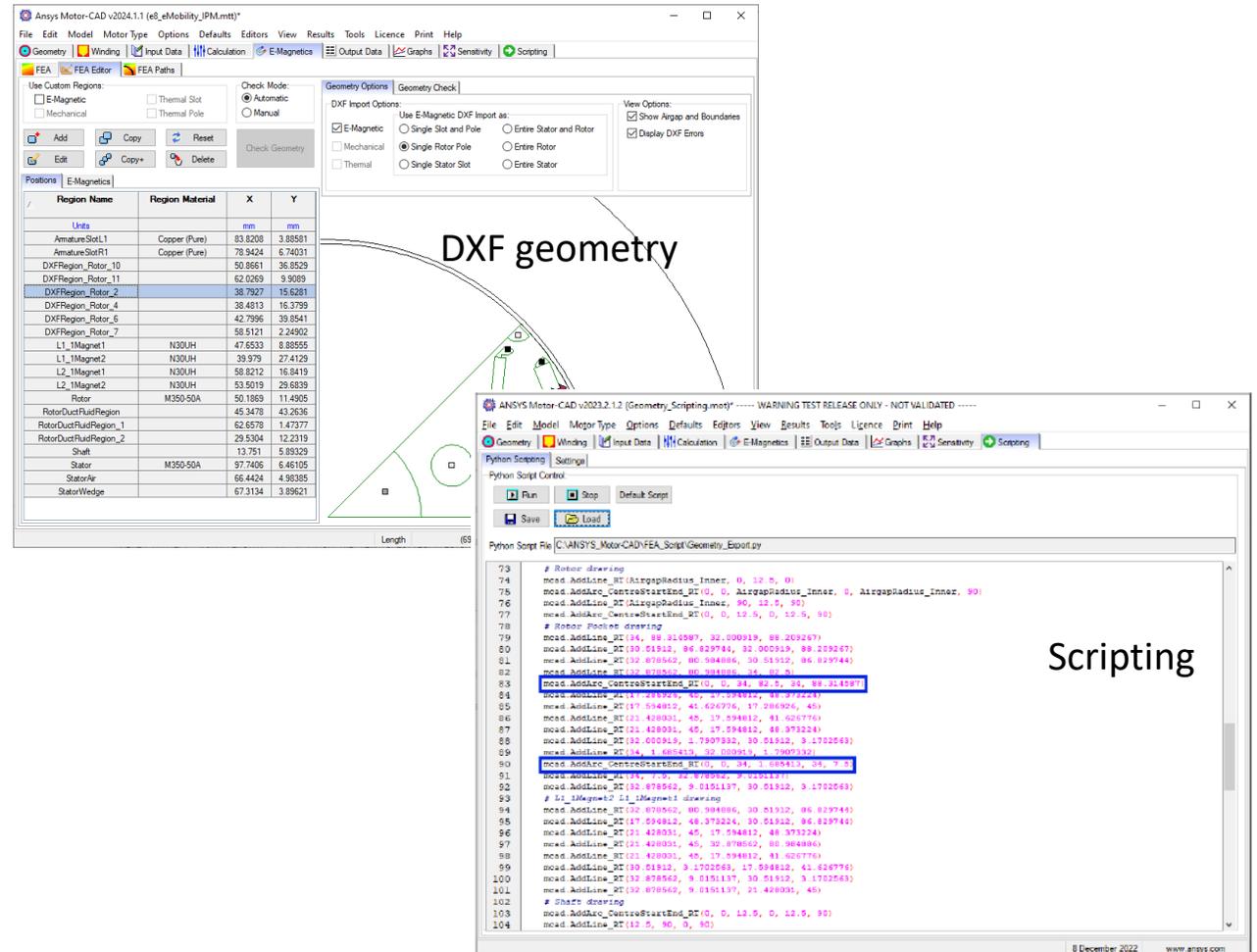
3D visualization



Optimization

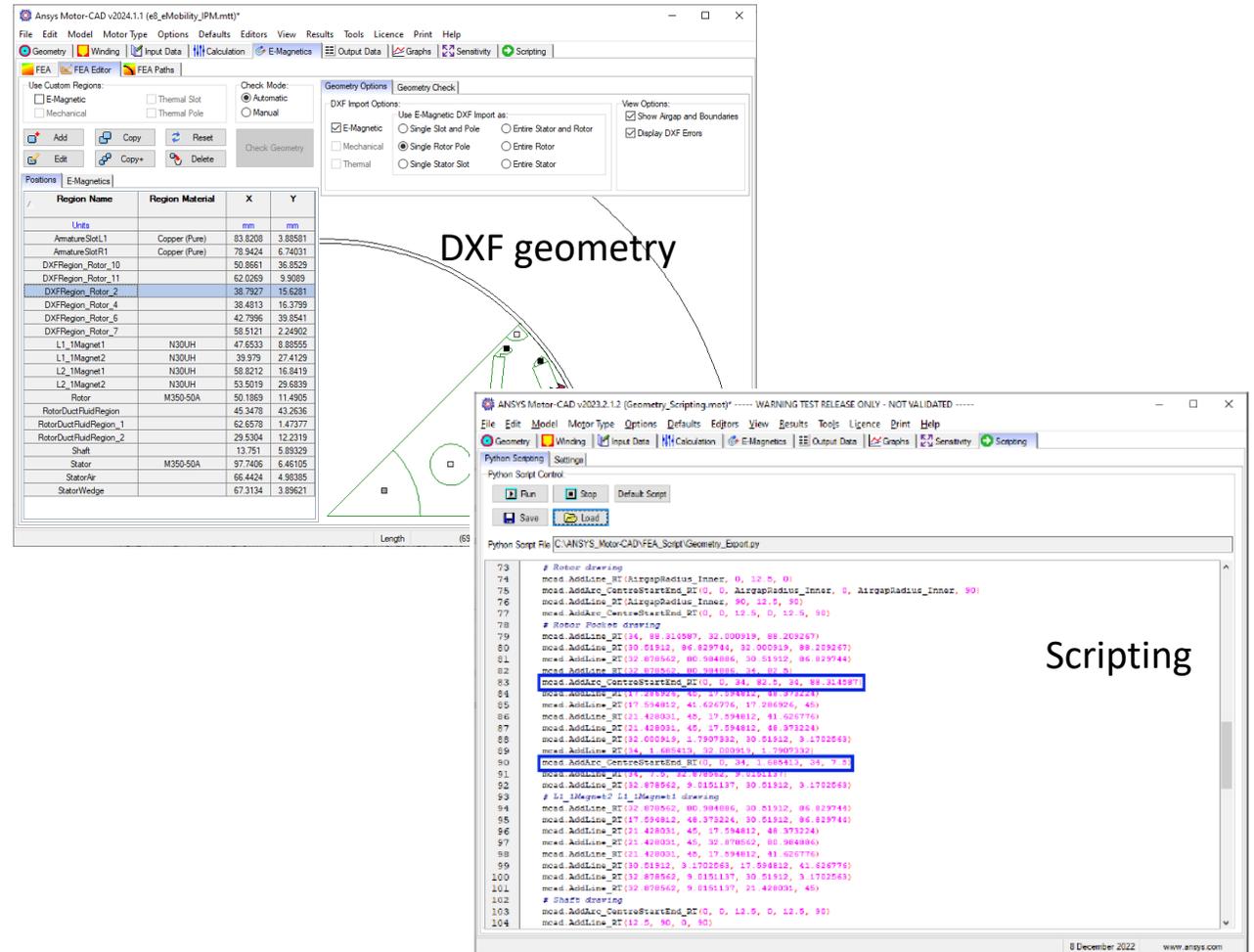
Options for Customization in Ansys Motor-CAD 2023 R2 – Part 1

- Engineers wish to customize the geometry as a design progresses:
 - To maximize multiphysics performance
 - To capture manufacturing requirements
 - To optimize the design
- Two options were possible in previous versions of Motor-CAD:
 - DXF drawing import
 - Limited internal/external scripting.



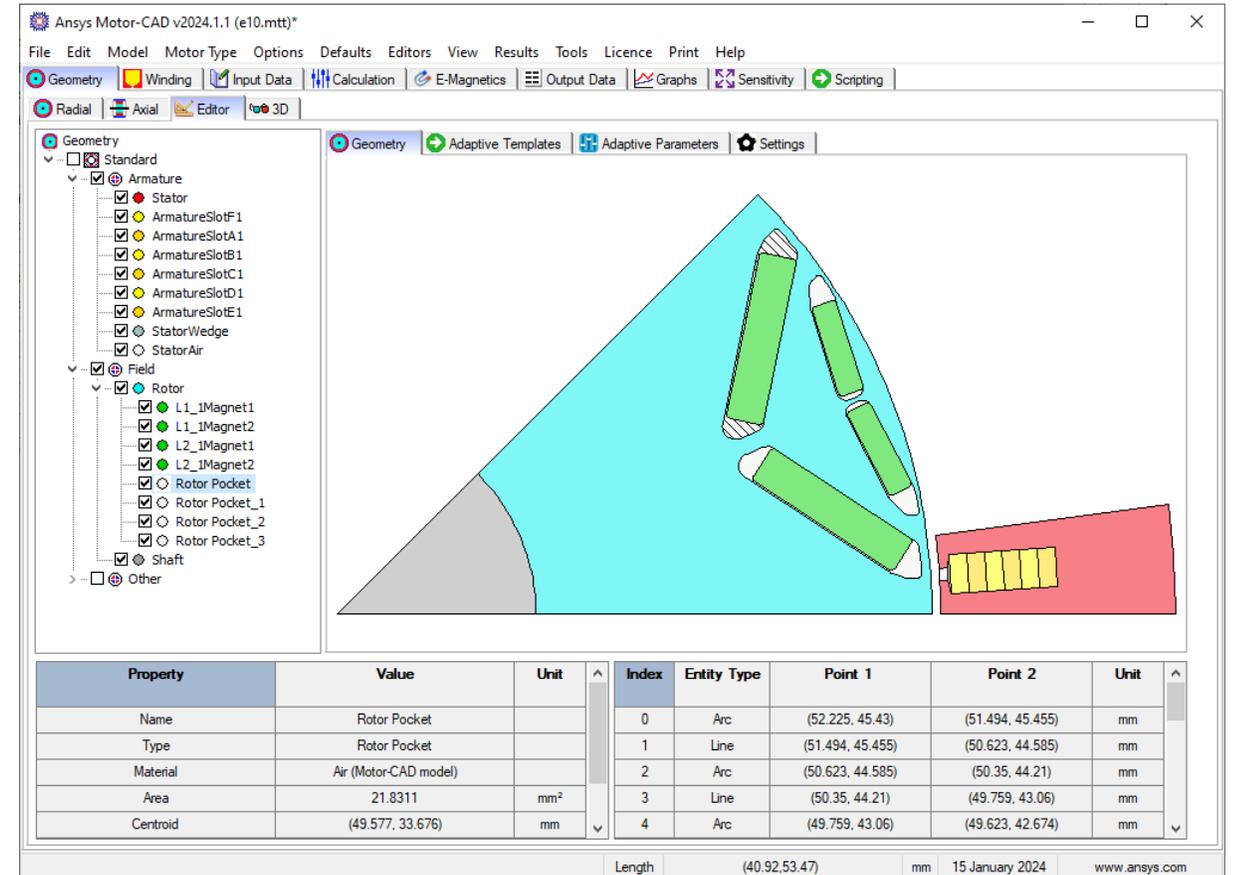
Options for Customization in Ansys Motor-CAD 2023 R2 – Part 2

- Some engineers have created automated workflows in the past:
 - Create a .dxf drawing in a 3rd party tool
 - Automating the creation of a drawing file was often very complex
 - New geometry templates would need to be created from first principles
 - Old internal scripting was very limited
- Therefore, this kind of workflow was rare with many E-machine designers instead:
 - Relying on drawing software to create their customizations
 - No automation means no scalability!



Enter: Ansys Motor-CAD 2024 R1 – Adaptive Templates

- The aim:
 - Provide engineers more freedom with geometry creation
 - Provide tools to enable this customization
 - Maintain the same ease-of-use seen with standard templates
- The benefits:
 - Automation and scalability
 - Faster motor design
 - New opportunities for optimization
 - Better Motor performance

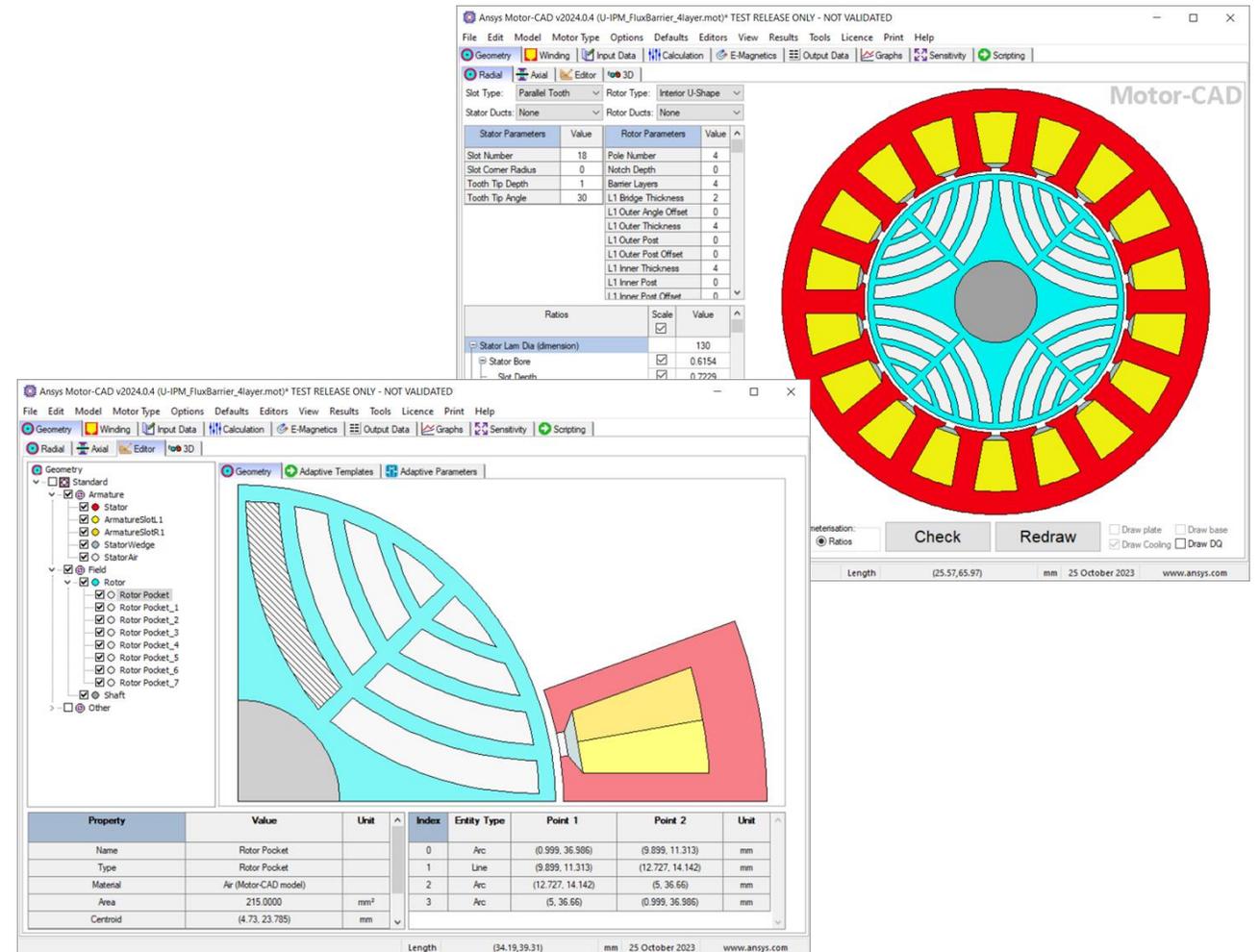




Overview of Ansys Motor-CAD 2024 R1 Adaptive Templates

Ansys Motor-CAD 2024 R1 – Adaptive Templates Overview

- Released February 6th, 2024
- Embed Python commands to reparametrize and customize the in-built template geometry
- Add custom geometry parameters
- Flexibility to innovate with the speed and ease of the templated geometry
- Enables IP library to be built up

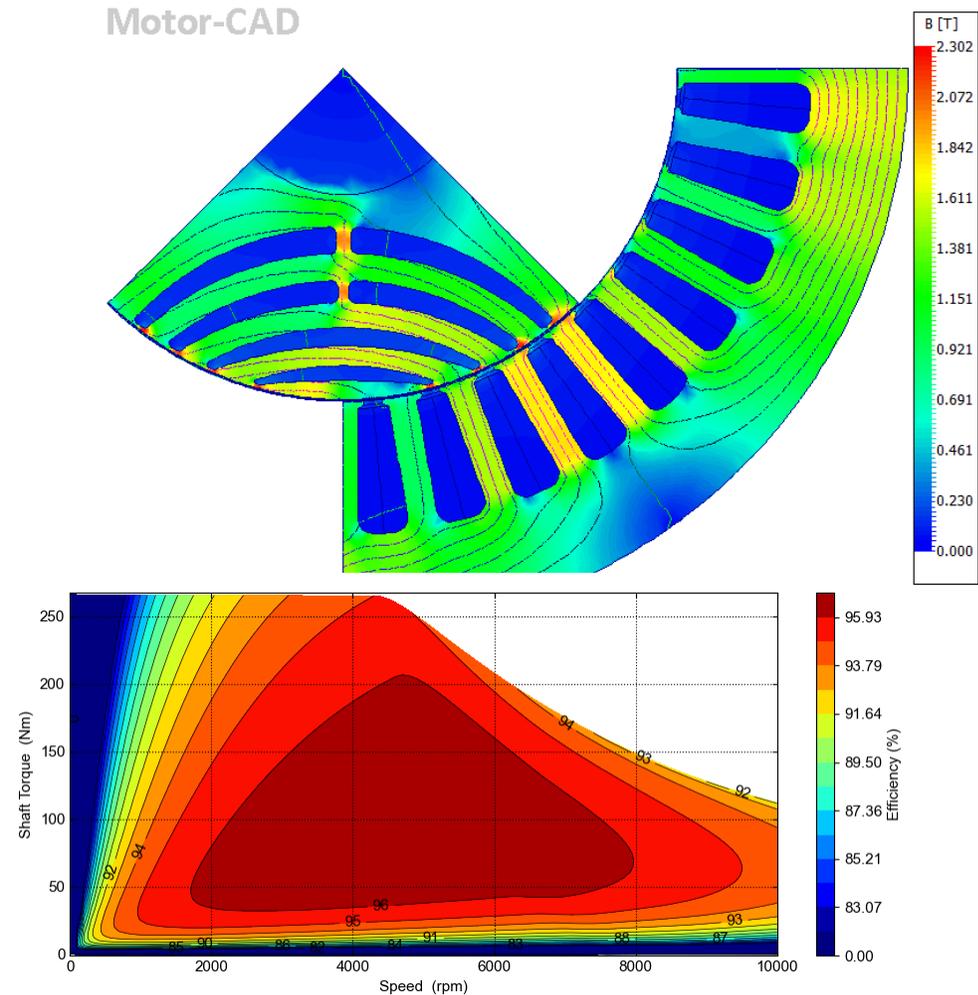




The Adaptive Templates and the Physics Modules of Ansys Motor-CAD

Adaptive Templates & Electromagnetics

- The electromagnetic module is the starting point
- Available to all motor types
- 2D FEA will solve the adapted geometry directly
- Automation of the model is maintained:
 - Meshing, materials, calculation, winding setup, etc.
- Make use of adaptive templates anywhere the FEA is used:
 - Lab module (efficiency mapping and duty cycles)
 - Saturation map export
 - Noise Vibration Harshness (NVH)



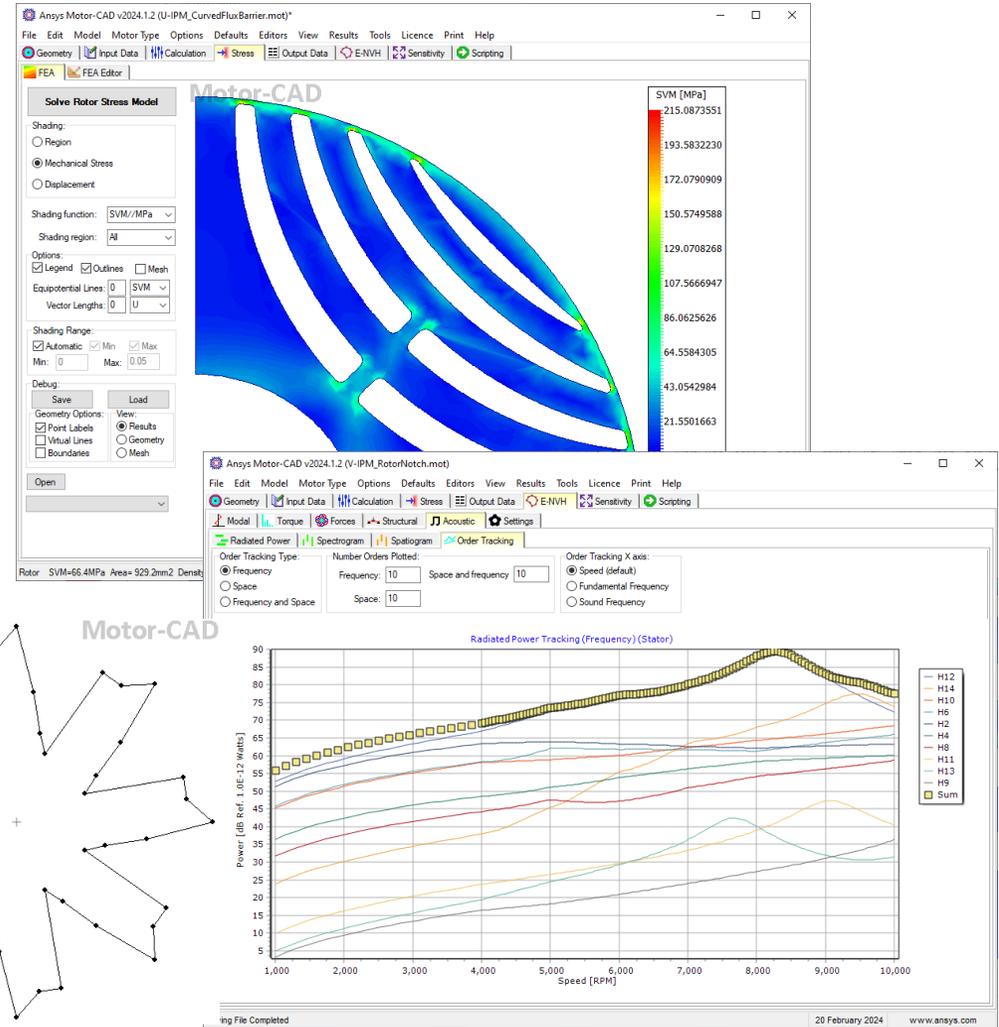
Adaptive Templates & Mechanics

- Rotor stress analysis:

- 2D FEA will solve the adapted geometry directly
- No additional setup necessary

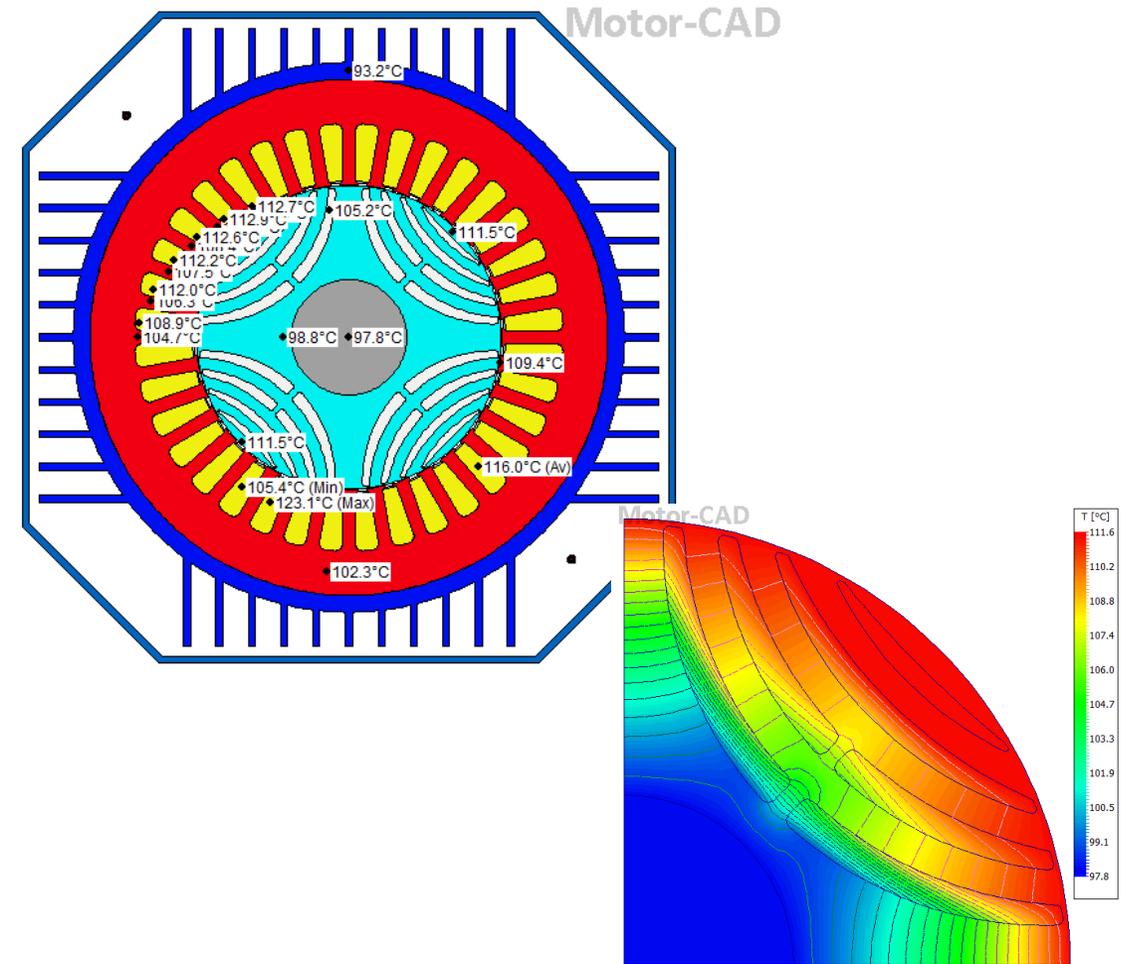
- Noise & Vibration Analysis:

- Force calculation uses adaptive template (E-mag)
- Operating point calculation uses adaptive template (Lab)



Adaptive Templates & Thermals

- Adaptive templates can only be applied to:
 - Active components (stator, rotor, magnets, banding/sleeve and shaft)
 - Radial cross-section
- Thermal model setup primarily follows the standard cooling templates
- Major customizations on the cooling are best served via changes to the thermal circuit
- However, cross-sectional area and the mass of components updates with the adaptive template:
 - Thermal masses & capacitances
 - 2D thermal FEA for calibration

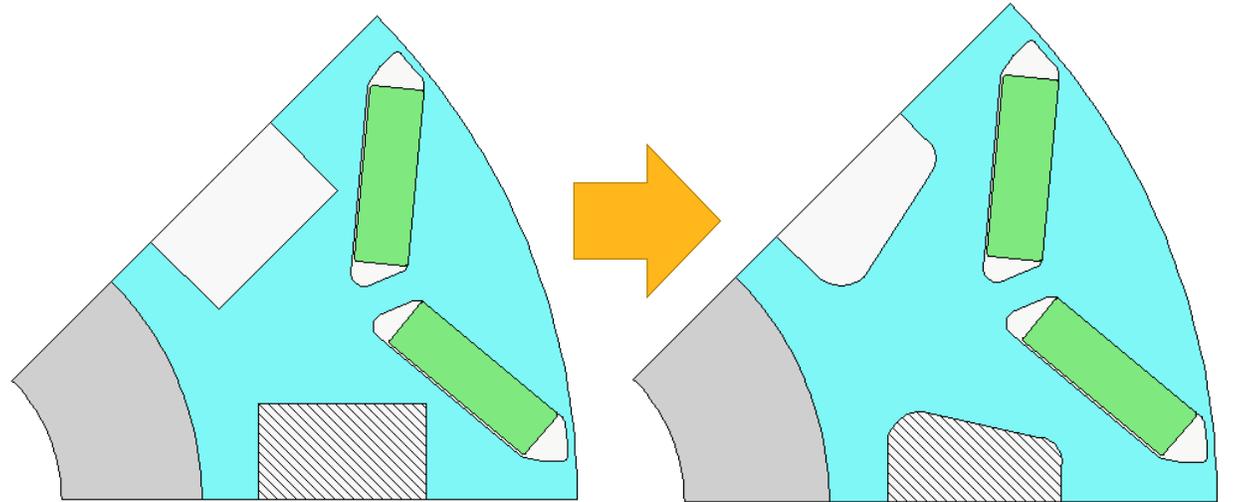




Adaptive Template Creation

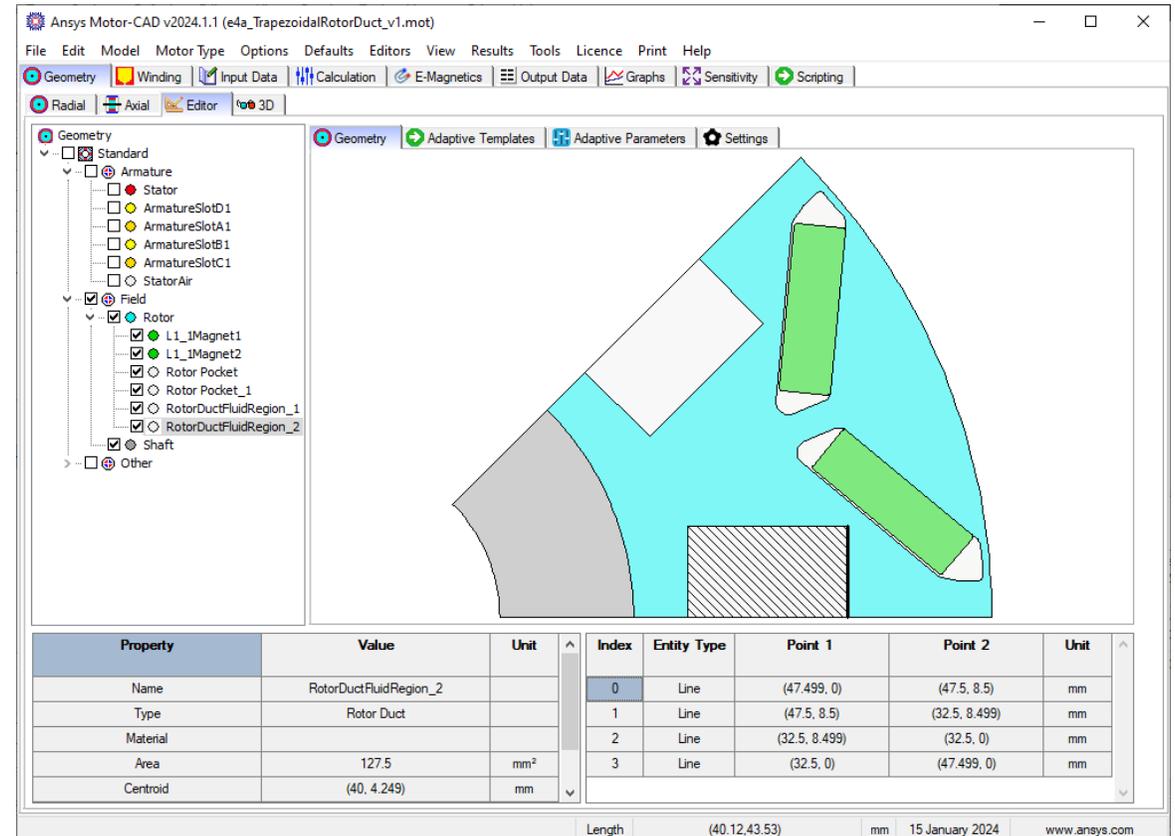
Adaptive Template Motor Examples

- Example 1: Creating a trapezoid rotor duct
 - Convert a square “standard” rotor duct into a trapezoid
 - We will introduce new parameters to define it
 - We’ll show the typical workflow for creating a new adaptive template



Trapezoid Rotor Duct, the Editor Window

- Dedicated **Geometry** -> **Editor** window:
 - Geometry defined into groups
 - Sub-groups have **regions**
 - **Regions** have **entities** (lines, arcs), **points** (co-ordinates in polar or cartesian) and material definitions
- The Graphical User Interface helps with the customization process



Trapezoid Rotor Duct, Python Scripting

- We can create new regions or make modifications to existing ones.
- However, the main principle of the scripting is to make changes on top of the standard geometry:
 - Significant reduction in the effort required
 - Most customized geometries share many base dimensions with a template: position of magnets, slot, conductors, inner & outer diameter
 - Customizations will adapt to large changes in the motor geometry – perfect for optimizations
 - Automatic setup of the thermal model, conductors, etc.

The screenshot displays the Ansys Motor-CAD v2024.1.1 interface. The left-hand tree view shows the project structure, with 'Rotor' expanded to show 'RotorDuctFluidRegion_2'. The central script editor shows a Python script for creating a trapezoidal rotor duct. The script defines a function to create a line segment based on rotor radius and pole count. The bottom table provides details for the 'RotorDuctFluidRegion_2' entity.

```
35 from ansys.motorcad.core.geometry_drawing import draw_regions
36
37 # $$
38 # Connect to Motor-CAD
39 mc = pymotorcad.MotorCAD(open_new_instance=False)
40
41 def get_rotor_d_axis_mirror_line():
42     rotor_radius = mc.get_variable("RotorDiameter")
43     number_poles = mc.get_variable("Pole_Number")
44     airgap_centre_x, airgap_centre_y = rt_to_xy(rotor_radius, (360 / number_poles) / 2)
45
46     return Line(Coordinate(0, 0), Coordinate(airgap_centre_x, airgap_centre_y))
47
48 # $$
49 # Get required parameters and objects
50 # -----
51 # Get the Adaptive parameters specified in Motor-CAD
52
53 trapezoid_top_width_ratio = mc.get_adaptive_parameter_value("trapezoid_top_width_ratio")
54
```

Property	Value	Unit	Index	Entity Type	Point 1	Point 2	Unit
Name	RotorDuctFluidRegion_2		0	Line	(47.5, 0)	(47.5, 4.25)	mm
Type	Rotor Duct		1	Line	(47.5, 4.25)	(32.5, 8.499)	mm
Material			2	Line	(32.5, 8.499)	(32.5, 0)	mm
Area	95.625	mm ²	3	Line	(32.5, 0)	(47.5, 0)	mm
Centroid	(40, 4.249)	mm					

Trapezoid Rotor Duct, Example Scripting Workflow – Part 1

- Start with a rectangular rotor duct from the standard geometry templates:
 - Minimize the scripting required
 - Keep the thermal model setup
- From the Motor-CAD editor window find the relevant region name and the Index number.
- Via Python script, grab the entity and a huge amount of information is automatically calculated:
 - Start/end co-ordinates (xy or rt)
 - Length, midpoint, gradient, angle from x/y-axes
 - These can be used for geometry manipulation

The screenshot displays a Python script in the ANSYS Motor-CAD console. The script uses the `mc.get_region()` method to retrieve a region object and then accesses its `entities` list to get a specific line entity. The console output shows the object's properties, including its start and end coordinates, length, midpoint, and angle.

```
60 # get pocket from Motor-CAD using unique name
61 duct_bottom_half = mc.get_region("RotorDuctFluidRegion_2") duct_bottom_half: <ansys.motorcad.core.geometry.Region object at 0x0000028E52B95330>
62
63 #draw_regions(duct_bottom_half)
64
65 entity0 = duct_bottom_half.entities[0] entity0: <ansys.motorcad.core.geometry.Line object at 0x0000028E52B95330>
66
67 entity0 = [Line] <ansys.motorcad.core.geometry.Line object at 0x0000028E52B95330>
68   angle = (float) 90.0
69   end = (Coordinate) [47.5, 8.5]
70   gradient = (float) inf
71   is_horizontal = (bool) False
72   is_vertical = (bool) True
73   length = (float) 8.5000000000000005
74   midpoint = (Coordinate) [47.5, 4.2499999999999997]
75   start = (Coordinate) [47.5, -5.89062235219235e-15]
76   y_intercept = (str) 'Traceback (most recent call last):\n  File "C:\Program Files\JetBrains\PyCharm Community Edition 2023.3.2\plugins\python-ce\helpers\py... View
77   Protected Attributes
```

The CAD interface below shows a 3D model of a rotor duct. The region `RotorDuctFluidRegion_2` is highlighted in the tree view. The properties table at the bottom provides detailed data for this region.

Property	Value	Unit	Index	Entity Type	Point 1	Point 2	Unit
Name	RotorDuctFluidRegion_2		0	Line	(47.499, 0)	(47.5, 8.5)	mm
Type	Rotor Duct		1	Line	(47.5, 8.5)	(32.5, 8.499)	mm
Material			2	Line	(32.5, 8.499)	(32.5, 0)	mm
Area	127.5	mm ²	3	Line	(32.5, 0)	(47.499, 0)	mm
Centroid	(40, 4.249)	mm					

Summary statistics: Length: (40.12, 43.53) mm, 15 January 2024, www.ansys.com

Trapezoid Rotor Duct, Example Scripting Workflow – Part 2

- A PyMotorCAD function can plot regions to help with scripting and debugging:

```
86 draw_regions(duct_bottom_half)
```

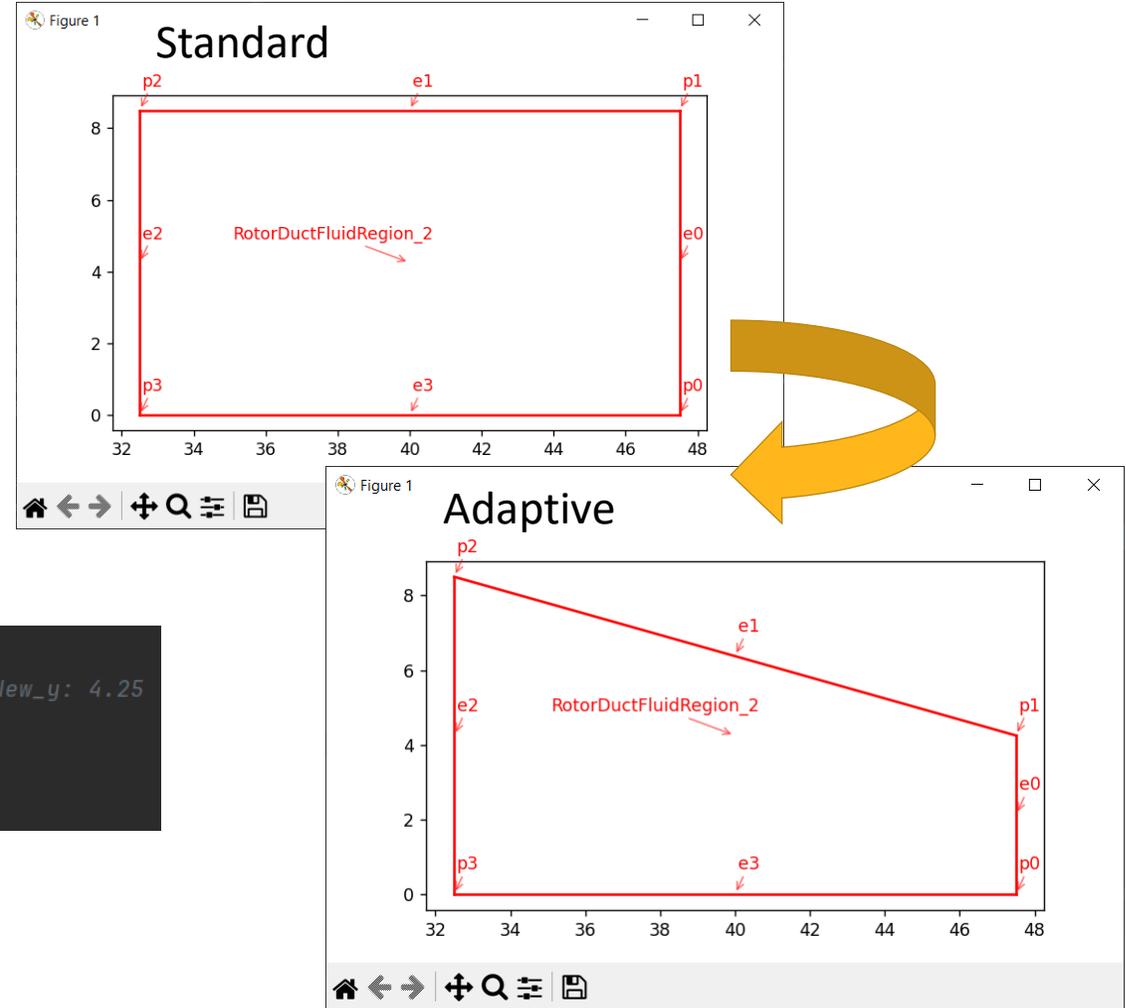
- Define a new adaptive template parameter to create a trapezoid:

Name	Value	Description
trapezoid_top_width_ratio	0.5	The proportion of the top of the trapezoid compared to the bottom

- Calculate a new Y value and edit p1

```
72 # Calculate the new y value
73 New_y = (trapezoid_duct_width / 2.0) * trapezoid_top_width_ratio New_y: 4.25
74 Coord1_New = Coordinate(Coord1.x, New_y) Coord1_New: [47.5, 4.25]
75 # Method 1: edit the existing point
76 duct_bottom_half.edit_point(Coord1, Coord1_New)
```

- Update the region and the lines have automatically shifted

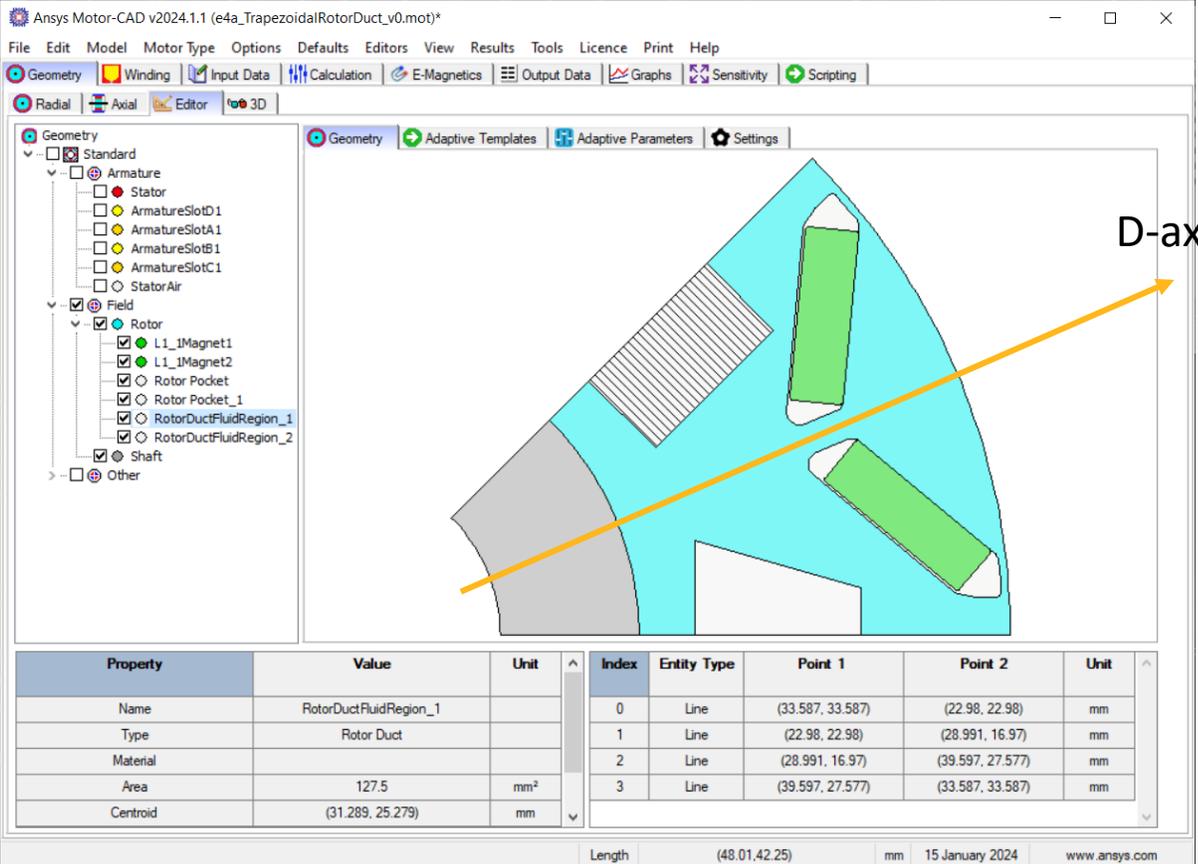


Trapezoid Rotor Duct, Example Scripting Workflow – Part 3

- Half of the region has been reshaped from a square to a trapezoid
- Use the PyMotorCAD mirror command to easily generate the co-ordinate on the opposite side of the D-axis:

```
88 d_axis_mirror_line = get_rotor_d_axis_mirror_line()
89
90 MirroredCoord = Coordinate.mirror(Coord1_New, d_axis_mirror_line)
```

- Update the region and the adaptive template is complete!



The screenshot displays the Ansys Motor-CAD v2024.1.1 interface. The main window shows a 3D model of a rotor duct with a trapezoidal cross-section. A yellow arrow labeled "D-axis" points from the center of the duct towards the right. The software interface includes a menu bar (File, Edit, Model, Motor Type, Options, Defaults, Editors, View, Results, Tools, Licence, Print, Help), a toolbar (Geometry, Winding, Input Data, Calculation, E-Magnetics, Output Data, Graphs, Sensitivity, Scripting), and a tree view on the left. The tree view shows the following structure:

- Standard
 - Armature
 - Stator
 - ArmatureSlotD1
 - ArmatureSlotA1
 - ArmatureSlotB1
 - ArmatureSlotC1
 - StatorAir
 - Field
 - Rotor
 - L1_IMagnet1
 - L1_IMagnet2
 - Rotor Pocket
 - Rotor Pocket_1
 - RotorDuctFluidRegion_1
 - RotorDuctFluidRegion_2
 - Shaft
- Other

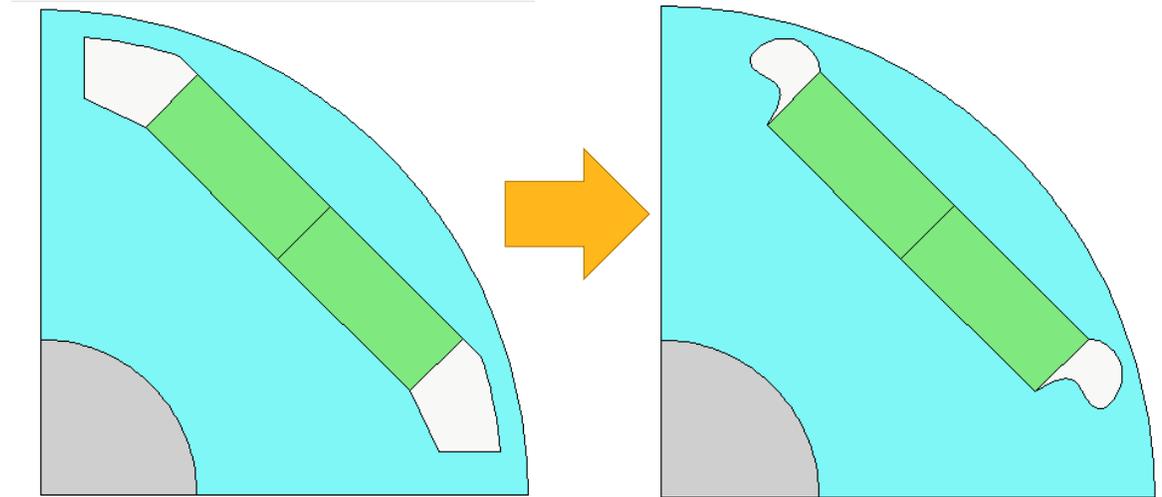
The property table at the bottom shows the following data:

Property	Value	Unit	Index	Entity Type	Point 1	Point 2	Unit
Name	RotorDuctFluidRegion_1		0	Line	(33.587, 33.587)	(22.98, 22.98)	mm
Type	Rotor Duct		1	Line	(22.98, 22.98)	(28.991, 16.97)	mm
Material			2	Line	(28.991, 16.97)	(39.597, 27.577)	mm
Area	127.5	mm ²	3	Line	(39.597, 27.577)	(33.587, 33.587)	mm
Centroid	(31.289, 25.279)	mm					

The status bar at the bottom shows: Length (48.01,42.25) mm, 15 January 2024, www.ansys.com

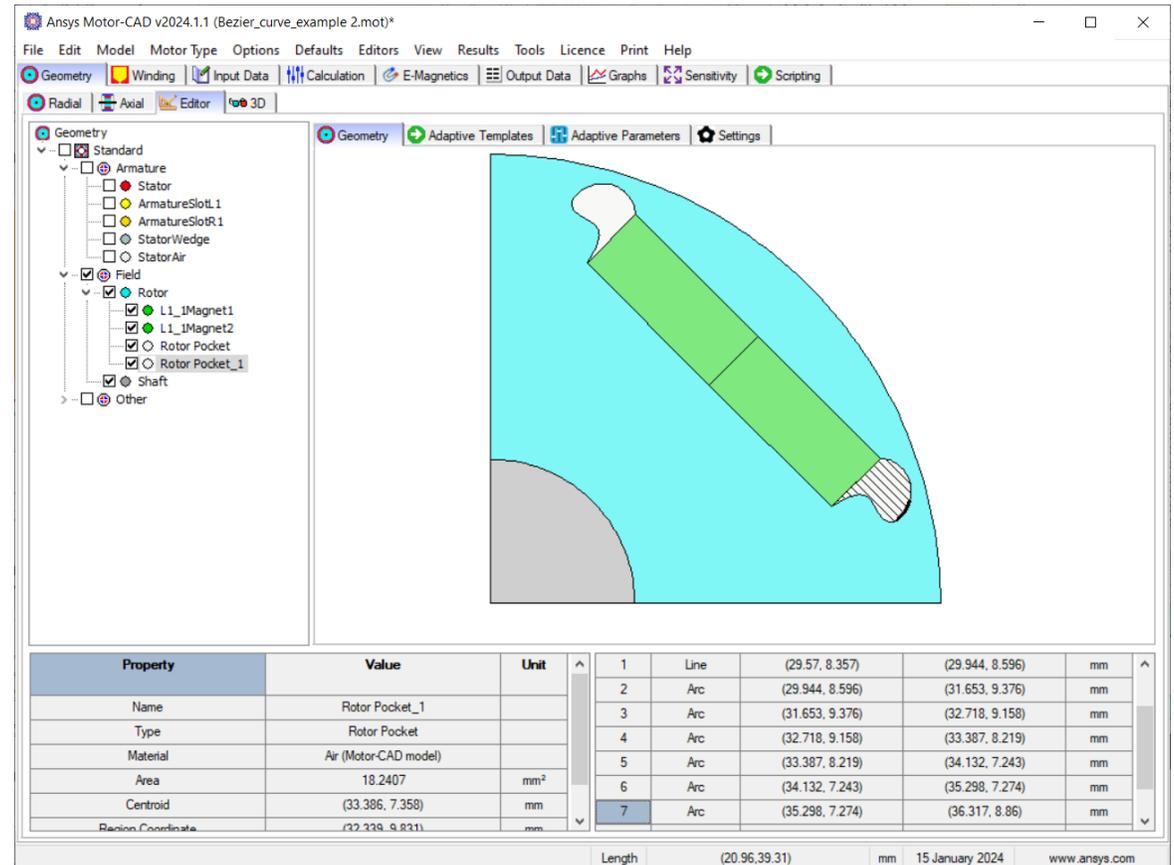
Adaptive Template Motor Examples

- Example 2: Advanced rotor pocket shaping with Bezier curves
 - More advanced geometry definition to show what is possible
 - Introduction to alternative methods of defining complex shapes



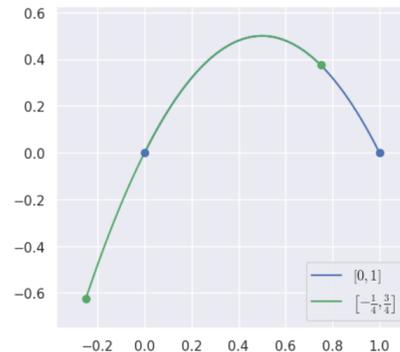
Rotor Pocket Shaping with Bezier Curves – Part 1

- The previous examples were all created using two types of geometry: line and arc
- More freeform shapes may be made up of many arcs
 - 4 parameters needed for every arc
 - Start, end, centre of arc and radius
- Instead, we can create shapes from several co-ordinates + a function for a curve
- Bezier curves are a common mathematical function used for this purpose (polyline is another).



Rotor Pocket Shaping with Bezier Curves – Part 2

- Starting with a standard template, the start and end-coordinate for the rocket pocket is found.
- Several adaptive parameters are created and combined with a python package for Bezier curves.
- Aim: gather a set of co-ordinates along the geometry



Name	Value	Description
bez_curve_projection	13	How far from the magnet the rotor pocket extends.
upperconvex	0.2	Magnitude and direction of the curvature on the top arc, closest to airgap
lowerconcave	-0.2	Magnitude and direction of the curvature on the bottom arc, closer to the shaft

Property	Value	Unit	Index	Entity Type	Point 1	Point 2	Unit
Name	L1_Magnet1		0	Line	(29.57, 8.357)	(33.813, 12.6)	mm
Type	Magnet		1	Line	(33.813, 12.6)	(23.206, 23.206)	mm
Material	N30UH		2	Line	(23.206, 23.206)	(18.964, 18.964)	mm
Area	90	mm ²	3	Line	(18.964, 18.964)	(29.57, 8.357)	mm
Centroid	(26.388, 15.782)	mm					

Rotor Pocket Shaping with Bezier Curves – Part 3

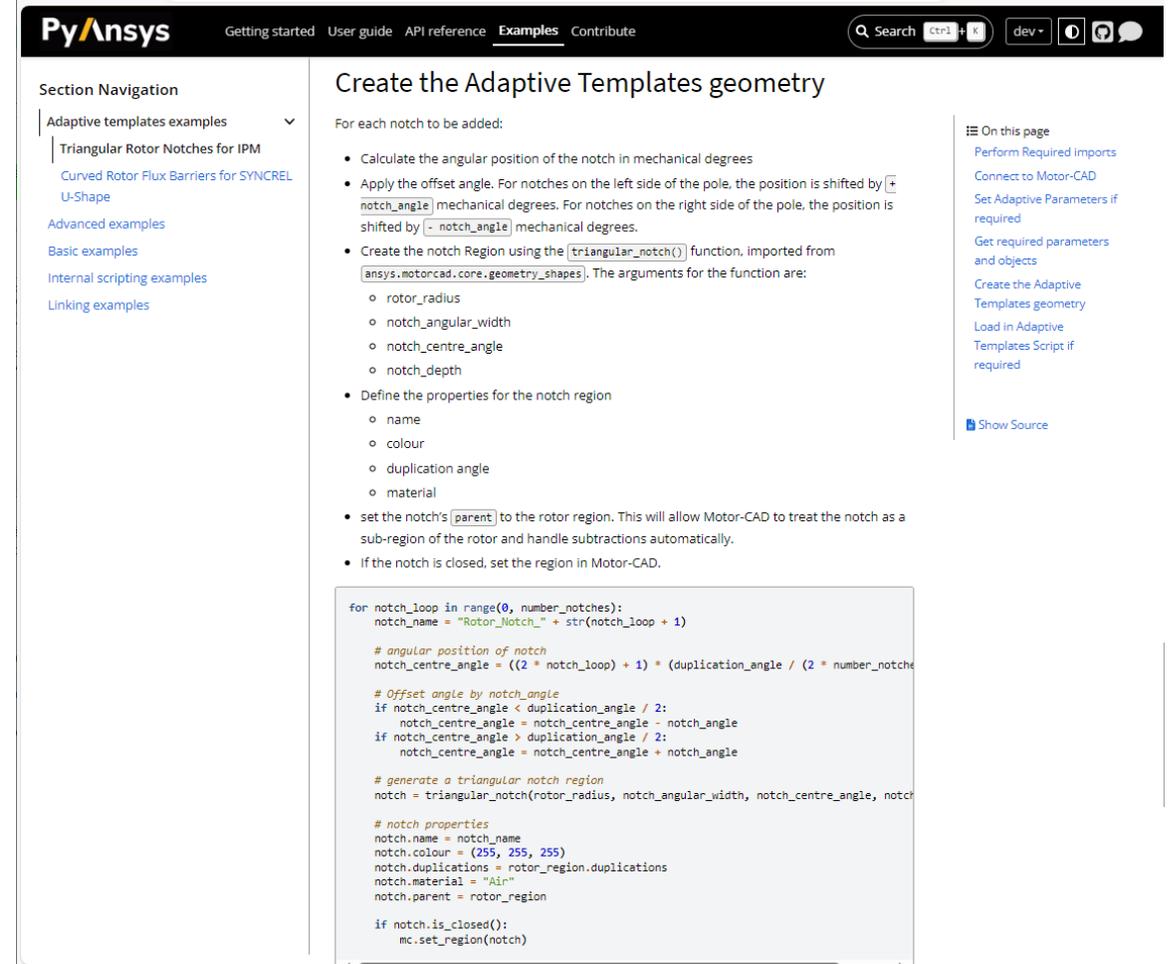
- A PyMotorCAD command can convert a set of co-ordinates automatically into a list of lines and arcs.
- The functions fits lines and arcs to the provided co-ordinates, with a given tolerance.
- The output is a minimized set of arcs and lines, which can be used to create a new adaptive template region.
- **LIVE DEMO**

```
66 def return_entity_list(coordinates, line_tolerance, arc_tolerance):
67     """Get list of entities from a list of coordinates.
68
69     Parameters
70     -----
71     coordinates : List of ansys.motorcad.core.geometry.Coordinate
72         coordinates from which to generate the geometry
73     line_tolerance : float
74         maximum allowed distance of point away from generated line
75     arc_tolerance : float
76         maximum allowed distance of point away from generated arc
77
78     Returns
79     -----
80     ansys.motorcad.core.geometry.EntityList
81
82     """
83     p = _PointFitting()
84     return p.return_entity_list(coordinates, line_tolerance, arc_tolerance)
```

```
63 bez_curve_entities = return_entity_list(xylist, linetolerance, arctolerance)
```

Python Scripting, PyMotorCAD and GitHub

- PyMotorCAD in 2024 R1 now contains:
 - Many new and powerful functions to aid custom geometry creation
 - Expanded guides to help users get started
 - Regularly updated throughout the year!
- GitHub is an open-source community
 - Download, use and alter existing examples
 - Ansys created as well as user submitted
- Engineers can build up their own internal libraries of motor geometries, easily scalable to any Motor design



The screenshot shows the PyAnsys website with a navigation menu on the left and a main content area. The main content area is titled "Create the Adaptive Templates geometry" and contains a list of steps for creating a notch. Below the steps is a code block with Python code for creating a notch region.

Section Navigation

- Adaptive templates examples
 - Triangular Rotor Notches for IPM
 - Curved Rotor Flux Barriers for SYNCREL
 - U-Shape
- Advanced examples
- Basic examples
- Internal scripting examples
- Linking examples

Create the Adaptive Templates geometry

For each notch to be added:

- Calculate the angular position of the notch in mechanical degrees
- Apply the offset angle. For notches on the left side of the pole, the position is shifted by `[notch_angle]` mechanical degrees. For notches on the right side of the pole, the position is shifted by `[- notch_angle]` mechanical degrees.
- Create the notch Region using the `triangular_notch()` function, imported from `[ansys.motorcad.core.geometry.shapes]`. The arguments for the function are:
 - o `rotor_radius`
 - o `notch_angular_width`
 - o `notch_centre_angle`
 - o `notch_depth`
- Define the properties for the notch region
 - o `name`
 - o `colour`
 - o `duplication angle`
 - o `material`
- set the notch's `parent` to the rotor region. This will allow Motor-CAD to treat the notch as a sub-region of the rotor and handle subtractions automatically.
- If the notch is closed, set the region in Motor-CAD.

```
for notch_loop in range(0, number_notches):
    notch_name = "Rotor_Notch_" + str(notch_loop + 1)

    # angular position of notch
    notch_centre_angle = ((2 * notch_loop) + 1) * (duplication_angle / (2 * number_notches))

    # Offset angle by notch_angle
    if notch_centre_angle < duplication_angle / 2:
        notch_centre_angle = notch_centre_angle - notch_angle
    if notch_centre_angle > duplication_angle / 2:
        notch_centre_angle = notch_centre_angle + notch_angle

    # generate a triangular notch region
    notch = triangular_notch(rotor_radius, notch_angular_width, notch_centre_angle, notch_depth)

    # notch properties
    notch.name = notch_name
    notch.colour = (255, 255, 255)
    notch.duplications = rotor_region.duplications
    notch.material = "Air"
    notch.parent = rotor_region

    if notch.is_closed():
        mc.set_region(notch)
```



Workflows in Other Ansys Products

Ansys Tools and the Adaptive Templates of Ansys Motor-CAD



MAXWELL



OPTISLANG



DISCOVERY

Ansys Motor-CAD & Ansys optiSLang – Motor Optimization

- Any adaptive template parameters created by a user, can be found within the Motor-CAD to optiSLang export window.
- The adaptive template script will move automatically with the export into optiSLang.
- A PyMotorCAD function can be used to check if geometry is valid (overlapping)

collides

Region.collides(*regions*)

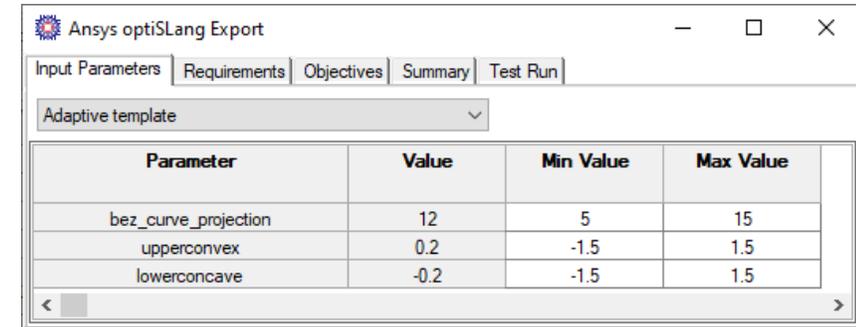
Check whether any of the specified regions collide with self.

Parameters:

regions : `ansys.motorcad.core.geometry.Region` or List of

`ansys.motorcad.core.geometry.Region`

Motor-CAD region object/list of objects

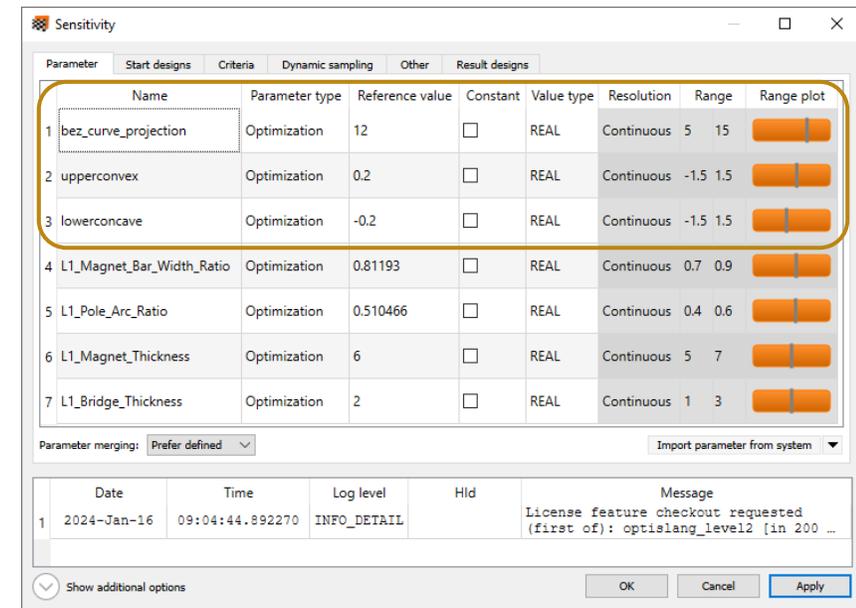


Ansys optiSLang Export

Input Parameters | Requirements | Objectives | Summary | Test Run

Adaptive template

Parameter	Value	Min Value	Max Value
bez_curve_projection	12	5	15
upperconvex	0.2	-1.5	1.5
lowerconcave	-0.2	-1.5	1.5



Sensitivity

Parameter | Start designs | Criteria | Dynamic sampling | Other | Result designs

Name	Parameter type	Reference value	Constant	Value type	Resolution	Range	Range plot
1 bez_curve_projection	Optimization	12	<input type="checkbox"/>	REAL	Continuous	5 15	
2 upperconvex	Optimization	0.2	<input type="checkbox"/>	REAL	Continuous	-1.5 1.5	
3 lowerconcave	Optimization	-0.2	<input type="checkbox"/>	REAL	Continuous	-1.5 1.5	
4 L1_Magnet_Bar_Width_Ratio	Optimization	0.81193	<input type="checkbox"/>	REAL	Continuous	0.7 0.9	
5 L1_Pole_Arc_Ratio	Optimization	0.510466	<input type="checkbox"/>	REAL	Continuous	0.4 0.6	
6 L1_Magnet_Thickness	Optimization	6	<input type="checkbox"/>	REAL	Continuous	5 7	
7 L1_Bridge_Thickness	Optimization	2	<input type="checkbox"/>	REAL	Continuous	1 3	

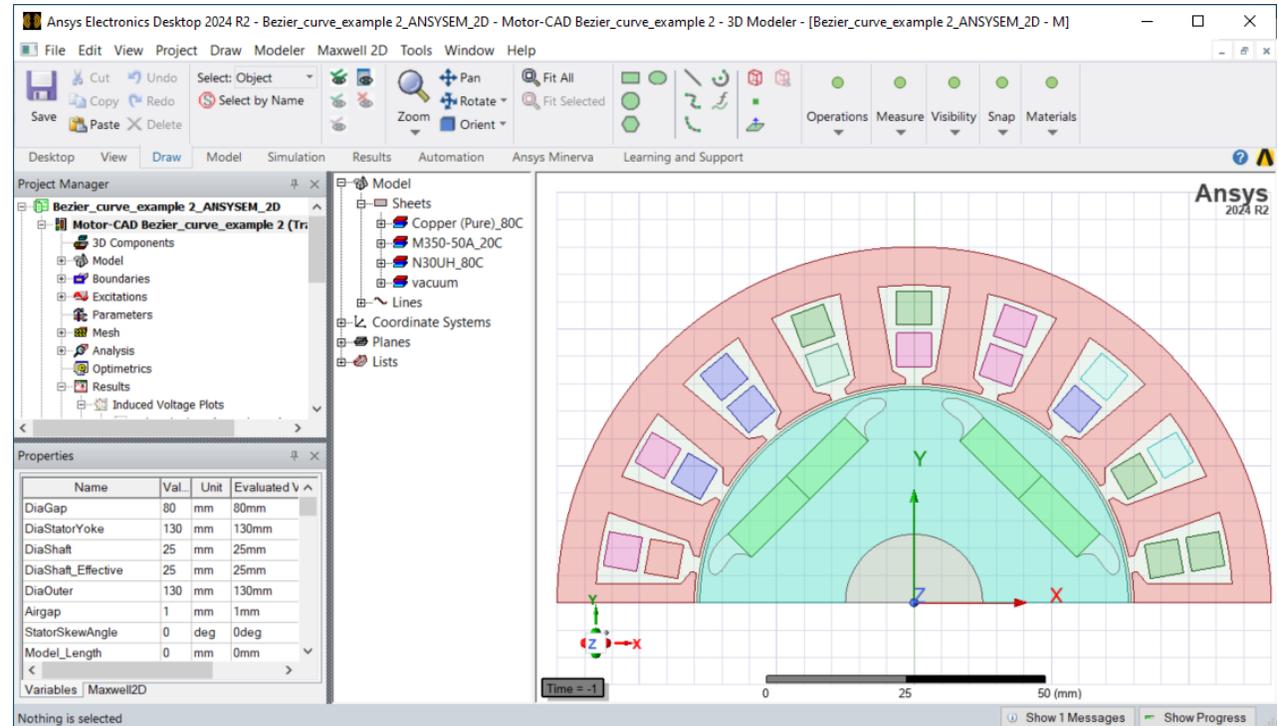
Parameter merging: Prefer defined | Import parameter from system

Date	Time	Log level	HId	Message
1 2024-Jan-16	09:04:44.892270	INFO_DETAIL		License feature checkout requested (first of): optislang_level12 [in 200 ...

Show additional options | OK | Cancel | Apply

Ansys Motor-CAD & Ansys Maxwell – Advanced Electromagnetics

- Exporting into Maxwell will carry across the adaptive template geometry!
- However, currently the geometry is fixed and will not carry across the parameterization.
- Bringing the parameterization across is on the roadmap.
- The Motor-CAD to Discovery export follows a similar process.

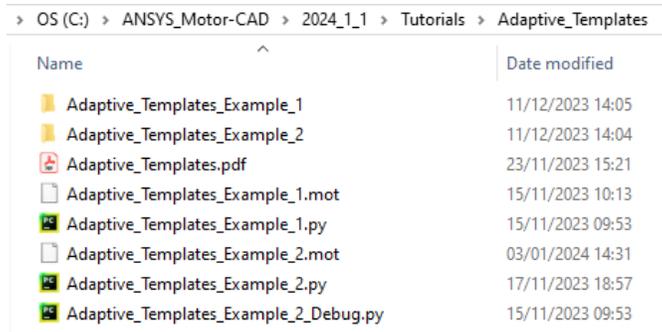




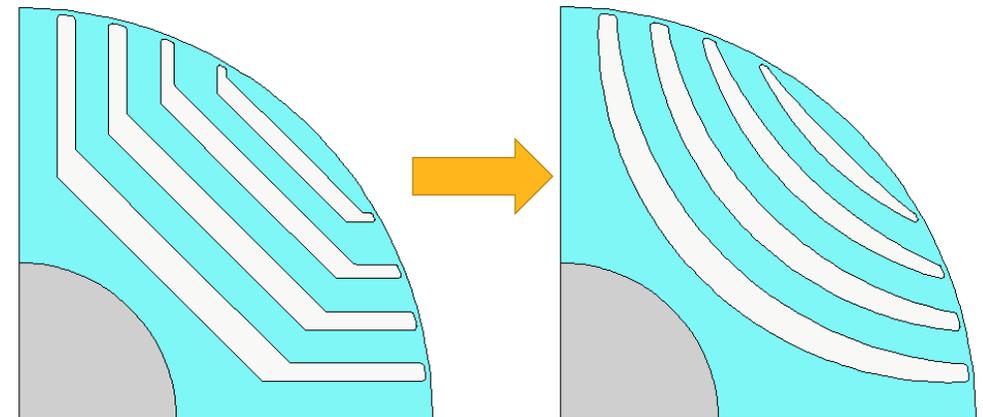
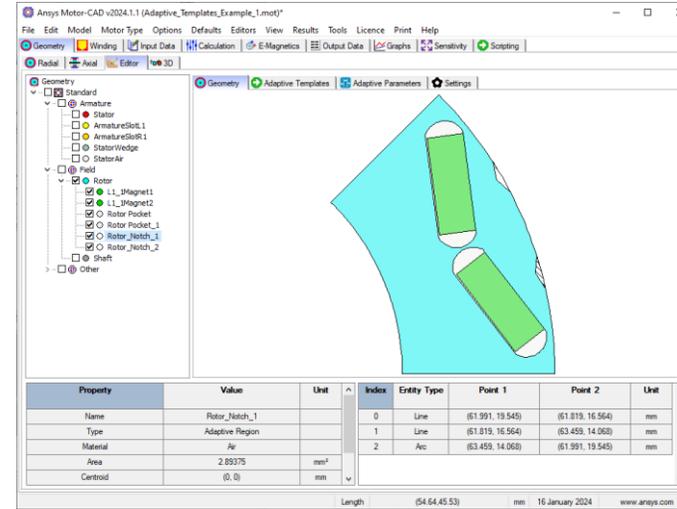
Getting Started

Ansys Motor-CAD 2024 R1 – Adaptive Templates Tutorial

- A tutorial is installed locally with 2024 R1:

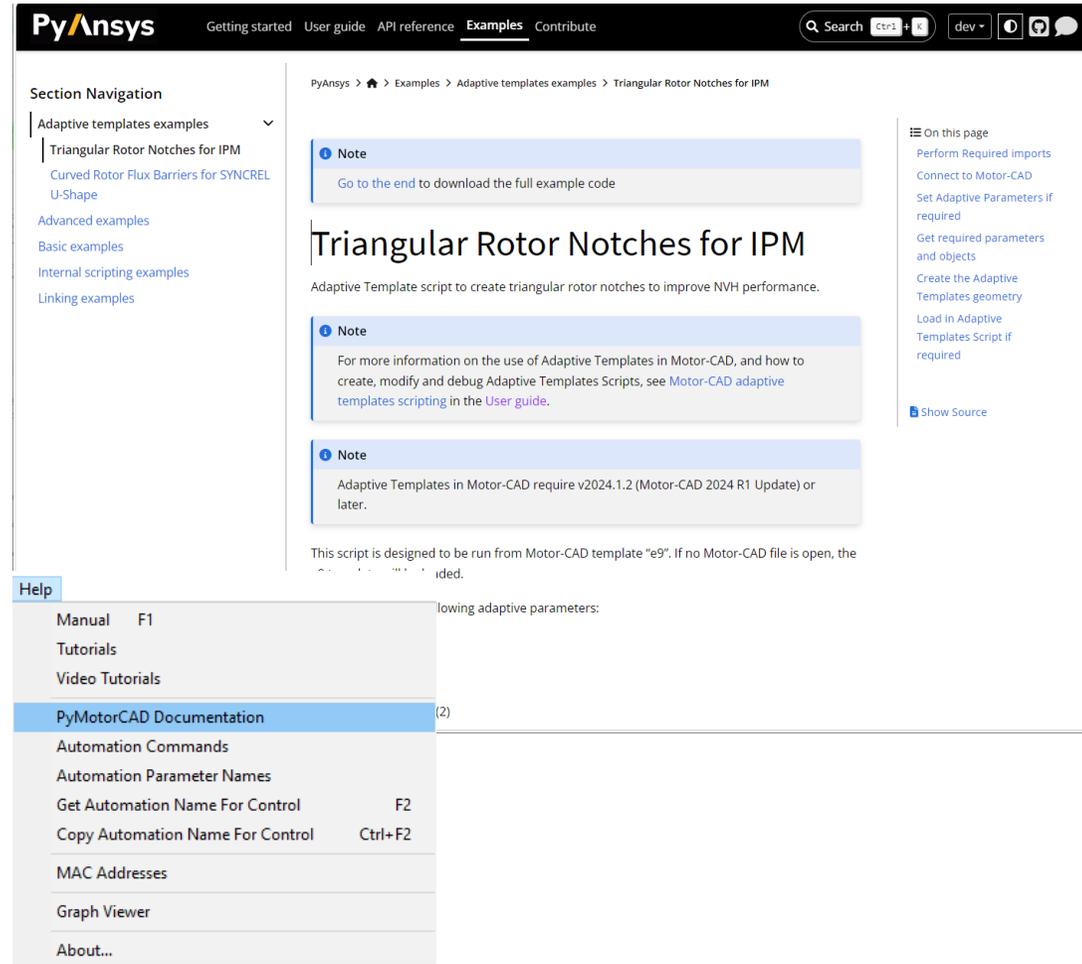


- The tutorial goes through the fundamentals of the feature: the Motor-CAD GUI, scripting etc.
- The tutorial comes with two examples:
 - Adding a V-shape rotor notch
 - Converting the Synchronous reluctance U-IPM template into a curved barrier



Ansys Motor-CAD 2024 R1 – PyMotorCAD & GitHub

- Check in on PyMotorCAD for regular updates
 - More detailed guides to help with python scripting
 - New adaptive template examples
- Find the link from within Ansys Motor-CAD, Help -> PyMotorCAD documentation





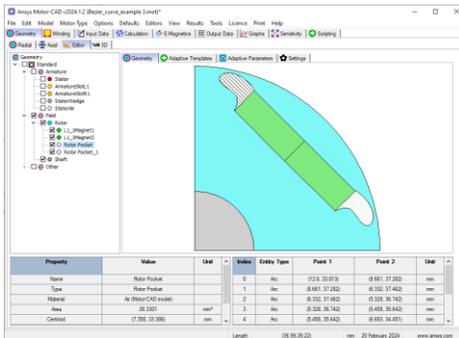
Summary

Ansys Motor-CAD 2024 R1 – Adaptive Templates

1

Enable Engineers to:

- Easily create customized shapes
- Maximize multiphysics motor performance
- Build up IP library



2

Major Benefits:

- Flexibility to innovate with ease
- Faster motor design
- Increased automation & scalability
- New opportunities for optimization
- Better motor performance



The Ansys logo consists of a yellow slanted bar followed by the word "Ansys" in a bold, black, sans-serif font.

