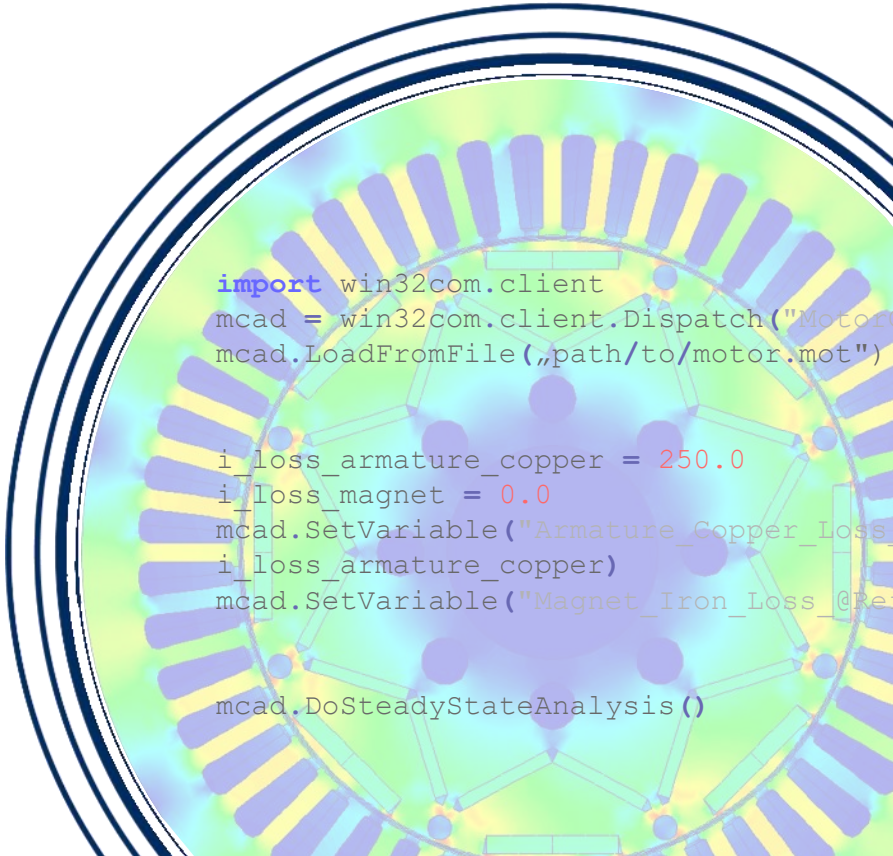
CADFEM Technologietage 2021
Elektromagnetik am Nachmittag

Elektromagnetik Best Practice

Faster Design Process through Scripting and Parallelization in Motor-CAD



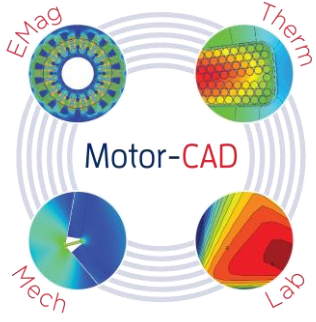
Philipp Siehr, CADFEM GmbH
psiehr@cadfem.de

CADFEM Ansys CERTIFIED ELITE CHANNEL PARTNER

Industry Standard Electric Machine Design Platform

CADFEM

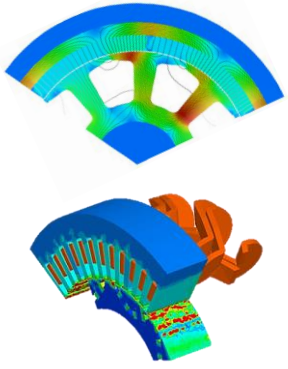
Design
Efficient Motor Design Toolkit



Motor Design Limited

Ansys

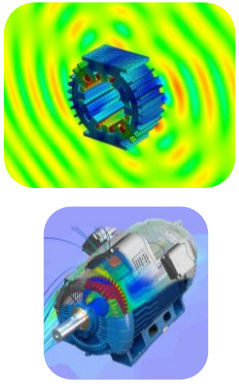
2D & 3D Analysis
Advanced Magnetics Modelling



Maxwell 2D & 3D

Ansys

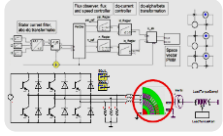
Coupled Analysis
NVH, Cooling, ...



Mechanical &CFD

Ansys

System Validation
Control logic, software



Motor Design Limited

Concept Design System Validation

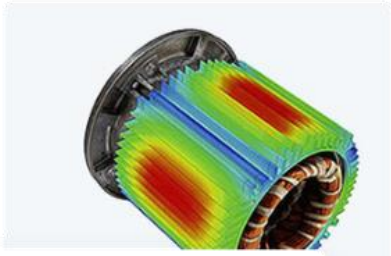
3D Physical Validation

Twin Builder

Ansys

dynardo optiSLang®

CADFEM Technologietage 2021



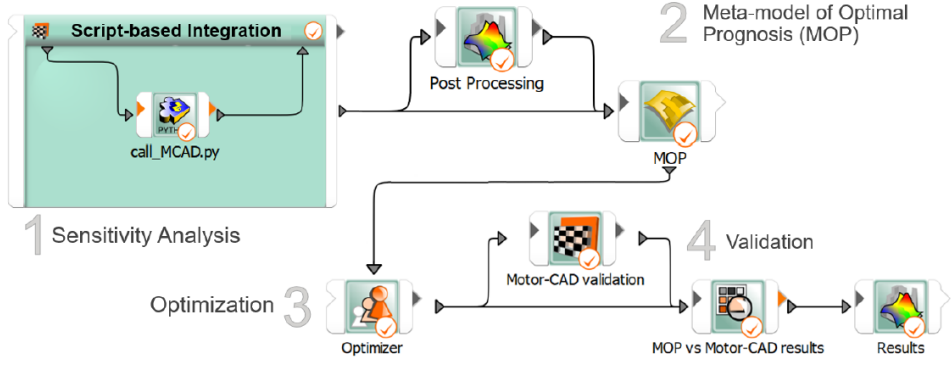
GEKOPPELTE SIMULATION

18. Mai 2021

Slides can be found here:
https://bit.ly/3vxrhgG

Electric machine optimisation for system level design of automotive traction units

Optimisation Workflow



A Meta-model of Optimal Prognosis (MOP) of the E-machine is built through a sensitivity analysis, using Motor-CAD.

The MOP model is then used in optimisation stage to create pareto fronts of 'best designs'

'Best designs' are validated in Motor-CAD

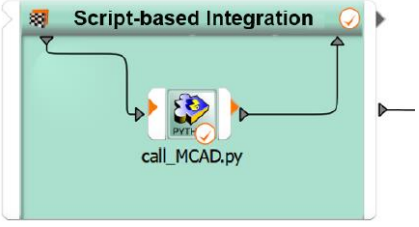
Jonathan Godbehere (Motor Design Ltd.)

Three Steps to Success

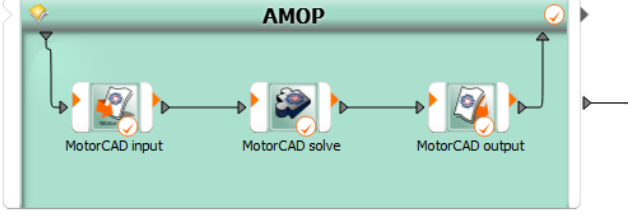
1) **Set Input Parameters**

2) **Run Computation**

3) **Extract Results**



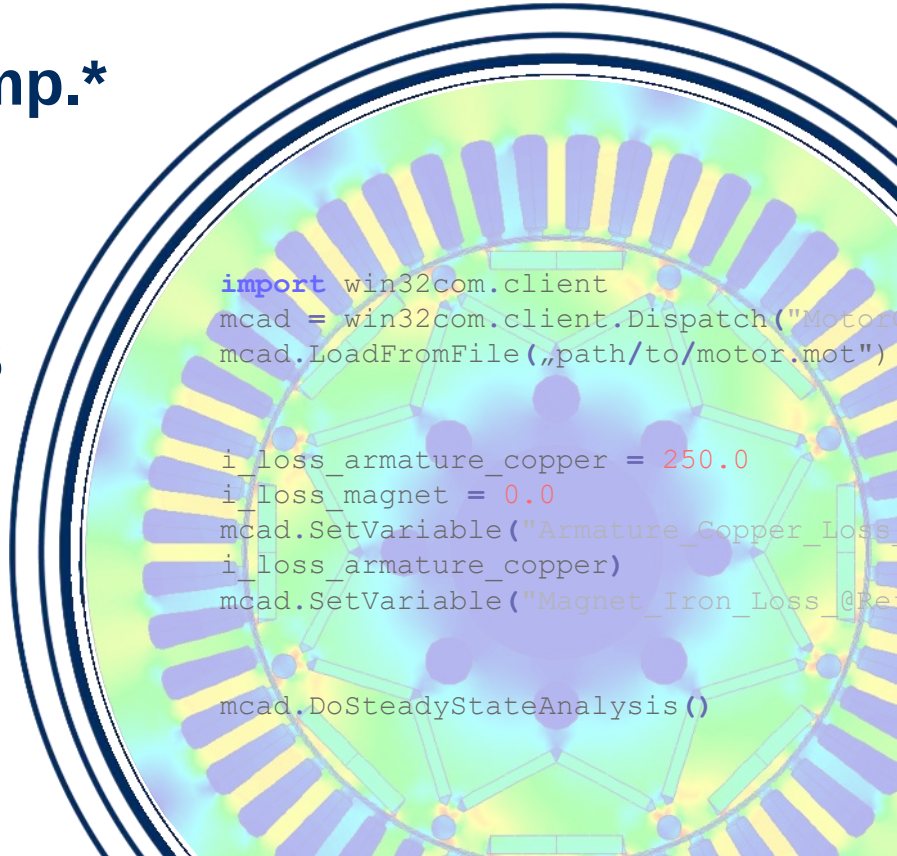
optiSLang connection via Python Script



optiSLang connection via Wizard

Agenda

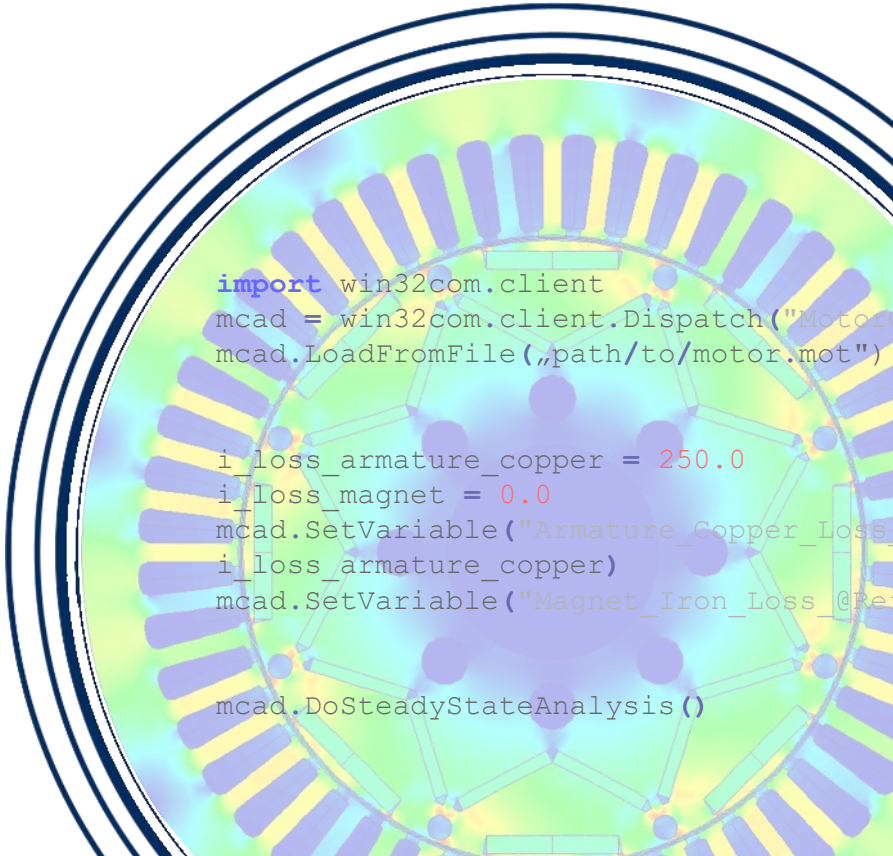
1. **Basic Example: Temperature Comp.***
2. **Extraction of Signals (Torque)**
3. **Parallelization with Blackbox***
4. **Comparison Python, Matlab, VBS**



***Quick Demo**

CADFEM® Ansys CERTIFIED ELITE CHANNEL PARTNER

First Example:
Computation of temperatures with varying magnet and copper loss

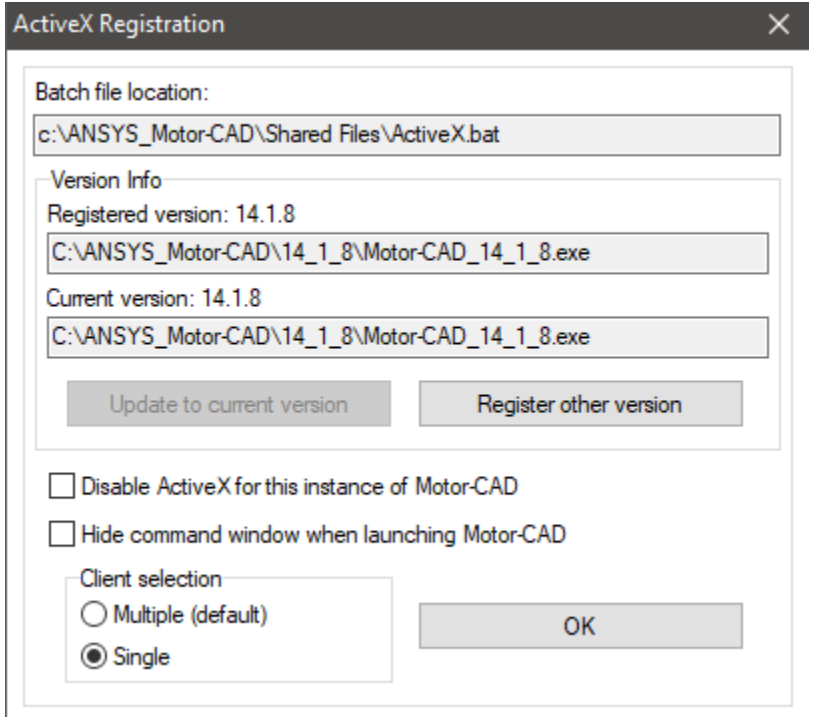


CADFEM Ansys CERTIFIED ELITE CHANNEL PARTNER

ActiveX Registration

Defaults > ActiveX Registration

- **Only one version of Motor-CAD can be enabled**
- **Multiple Clients**
  - **One Motor-CAD Instance for multiple sources**
  - **Necessary for Internal Scripting Environment**
  - **Parallel Working with GUI and Script**
- **Single Client**
  - **Parameter Studies**



Three Steps to Success

CADFEM

0) Create Motor-CAD Object

```
import win32com.client
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.LoadFromFile(„path/to/motor.mot")
```

1) Set Input Parameters

```
i_loss_armature_copper = 250.0
i_loss_magnet = 0.0
mcad.SetVariable("Armature_Copper_Loss_@Ref_Speed",
i_loss_armature_copper)
mcad.SetVariable("Magnet_Iron_Loss_@Ref_Speed",i_loss_magnet)
```

2) Run Computation

```
mcad.DoSteadyStateAnalysis()
```

3) Extract Results

```
# Motor-CAD returns a tuple [success, value], hence [1]
o_temp_winding_avg = mcad.GetVariable("T_[Winding_Average]")[1]
o_temp_winding_max = mcad.GetVariable("T_[Winding_Max]")[1]
o_temp_magnet = mcad.GetNodeTemperature(13)[1]
```

The full code can be found on the slides in the appendix.

Three Steps to Success

CADFEM

0) Create Motor-CAD Object

```
import win32com.client
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.LoadFromFile(„path/to/motor.mot")
```

1) Set Input Parameters

```
i_loss_armature_copper = 250.0
i_loss_magnet = 0.0
mcad.SetVariable("Armature_Copper_Loss_@Ref_Speed",
i_loss_armature_copper)
mcad.SetVariable("Magnet_Iron_Loss_@Ref_Speed",i_loss_magnet)
```

2) Run Computation

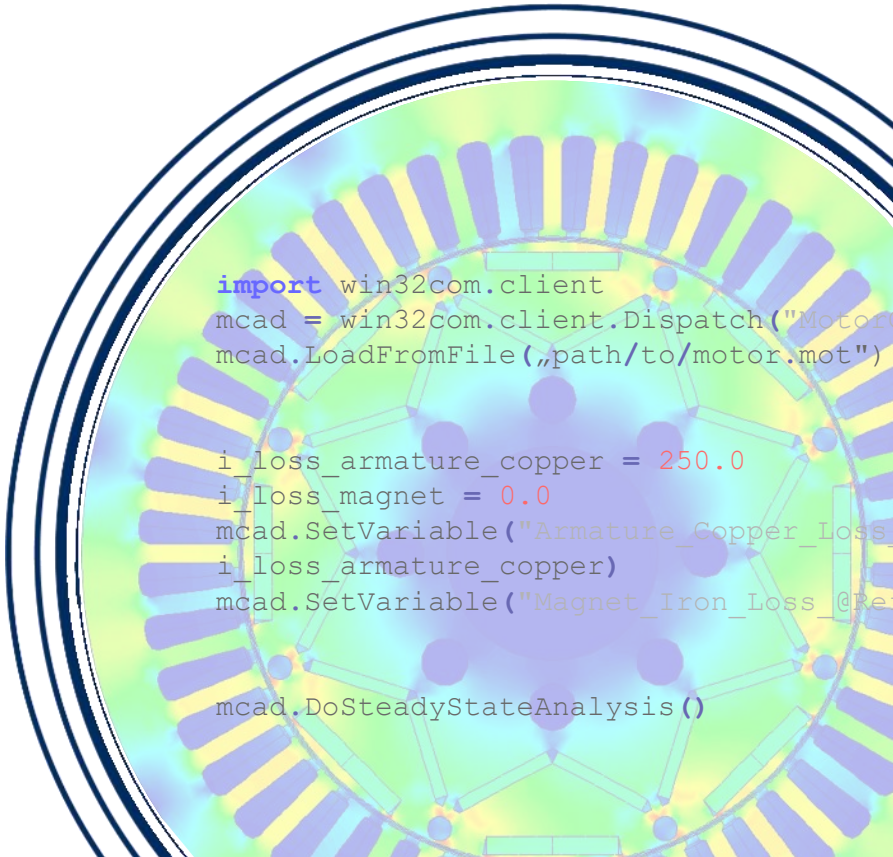
```
mcad.DoSteadyStateAnalysis()
```

3) Extract Results

```
# Motor-CAD returns a tuple [success, value], hence [1]
o_temp_winding_avg = mcad.GetVariable("T_[Winding_Average]")[1]
o_temp_winding_max = mcad.GetVariable("T_[Winding_Max]")[1]
o_temp_magnet = mcad.GetNodeTemperature(13)[1]
```

The full code can be found on the slides in the appendix.

Demo Thermal Simulation



CADFEM Ansys CERTIFIED ELITE CHANNEL PARTNER

Extraction of Signals

Motor-CAD allows to access any data stored in the Graph Viewer (Help>Graph Viewer)

To get the results, one needs to loop over the results and extract them point by point.

- **0 = successful**
- **-2 = graph does not exist**
- **-3 = no points in graph**
- **-4 = exceeded range of graph**

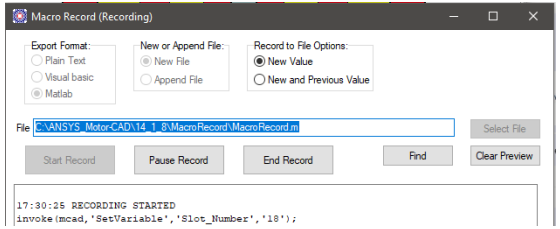
[Demo at the end, if there is still time]

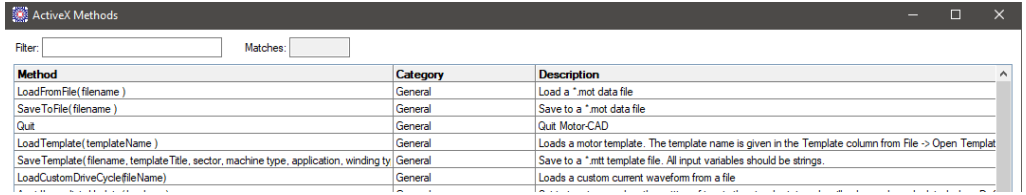
```
# Create the Motor-CAD Object and load model
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.LoadFromFile(„path/to/model.mot")

PointsPerCycle = int(mcad.GetVariable('TorquePointsPerCycle')[1])
NumberCycles = int(mcad.GetVariable('TorqueNumberCycles')[1])

# Running the computation
mcad.DoMagneticCalculation()

# Output
N = (PointsPerCycle*NumberCycles)+1
position_list = []
torque_list =[]
for it in range(0,N-1):
        success, x, y = mcad.GetMagneticGraphPoint('TorqueVW`,
                    it)
        position_list.append(x)
        torque_list.append(y)

average_torque = sum(torque_list)/len(torque_list)
```

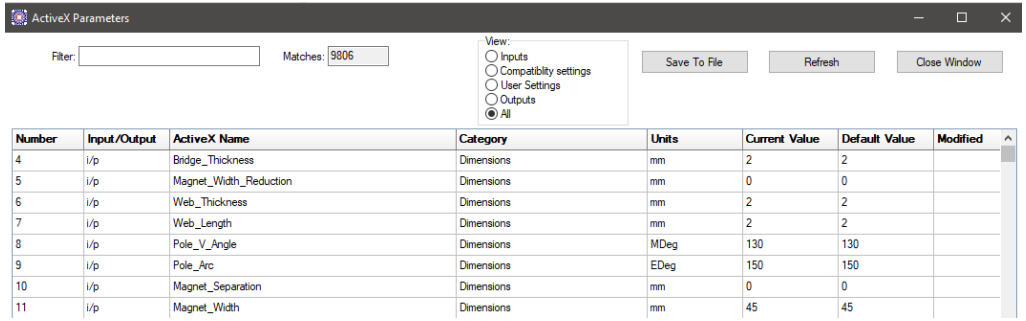
The full code can be found on the slides in the appendix.

How do I get the commands?

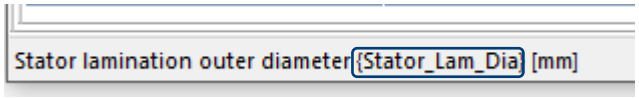
- **Tutorials → Summary Slide**
- **Tools > Macro Record**
- **Help > ActiveX Commands**
- **Help > ActiveX Parameter List**
- **Help > Get ActiveX […] "F2"**
- **Hovering over Parameter**







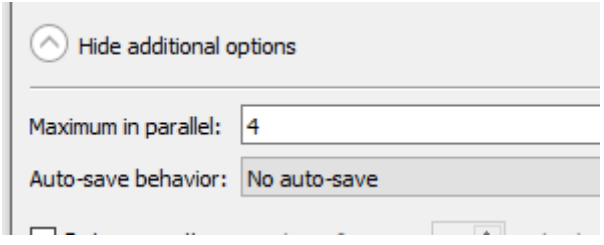
Number	Input/Output	ActiveX Name	Category	Units	Current Value	Default Value	Modified
4	i/p	Bridge_Thickness	Dimensions	mm	2	2	
5	i/p	Magnet_Width_Reduction	Dimensions	mm	0	0	
6	i/p	Web_Thickness	Dimensions	mm	2	2	
7	i/p	Web_Length	Dimensions	mm	2	2	
8	i/p	Pole_V_Angle	Dimensions	MDeg	130	130	
9	i/p	Pole_Arc	Dimensions	EDeg	150	150	
10	i/p	Magnet_Separation	Dimensions	mm	0	0	
11	i/p	Magnet_Width	Dimensions	mm	45	45	



* ~10.000 Parameter verfügbar

Blackbox Solver

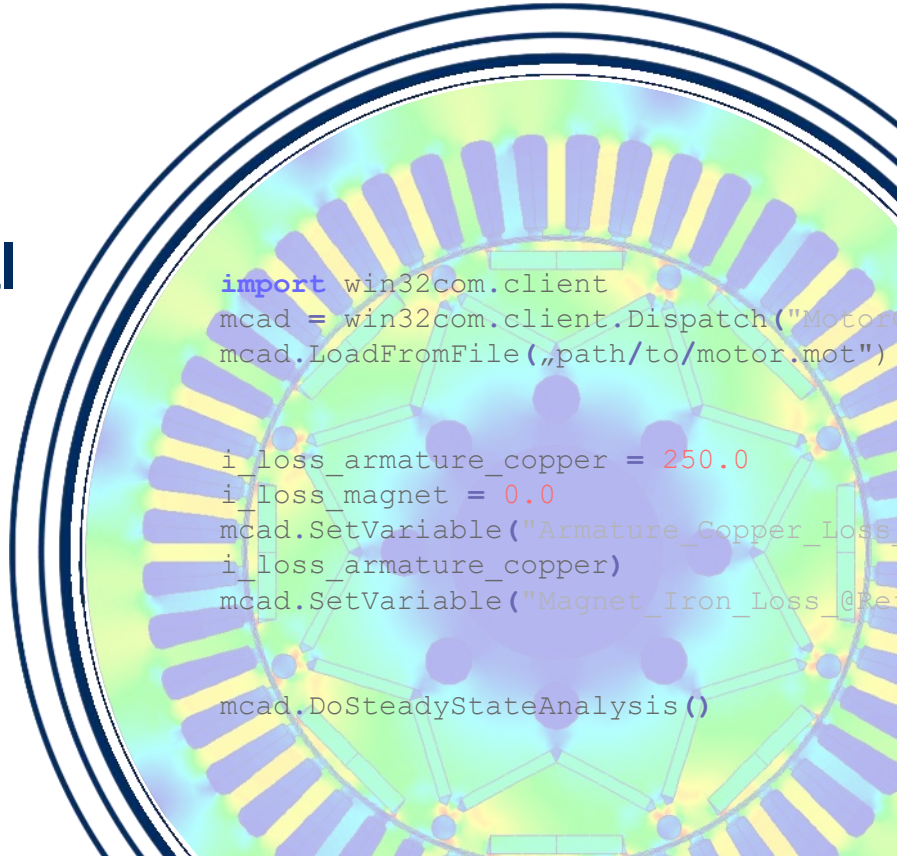
- **Blackbox Solvers are …**
  - **Non graphical**
  - **Ideal for parameter studies**
  - **Accessible via direct parallelization through scripting or with optiSLang**
  - **Enabled via the parameter Run_Type in the Defaults.INI (C:\ANSYS_Motor-CAD\14_1_8\Motor-CAD Data)**
    - **Run_Type = 0 → Regular Solver**
    - **Run_Type = 1 → Blackbox Solver**



Demo Blackbox with optiSLang

Copper and Magnet loss full factorial variation:

(0,0), (0,100), (100,0), (100,100)



CADFEM Ansys CERTIFIED ELITE CHANNEL PARTNER

Comparison Python, Matlab, VBS

- **Python**

```python
import win32com.client
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.SetVariable("Armature_Copper_Loss_@Ref_Speed",
i_loss_armature_copper)
mcad.DoSteadyStateAnalysis()
o_temp_winding_avg = mcad.GetVariable("T_[Winding_Average]")[1]
```

- **Matlab**

```matlab
mcad = actxserver('MotorCAD.AppAutomation`);
invoke(mcad, 'SetVariable', 'Armature_Copper_Loss_@Ref_Speed',
i_loss_armature_copper);
invoke(mcad, 'DoSteadyStateAnalysis`);
[success, o_temp_winding_avg] = invoke(mcad, 'GetVariable',
'T_[Winding_Average]')
```

- **VBS**

```vbs
dim mcad
set mcad = createobject("motorcad.appautomation")
call mcad.SetVariable("Armature_Copper_Loss_@Ref_Speed",
i_loss_armature_copper)
call mcad.DoSteadyStateAnalysis
call mcad.GetVariable("T_[Winding_Average]",
o_temp_winding_avg)
```

Summary & Outlook

A basic introduction into scripting with Motor-CAD has been shown.
There are three basic commands: **SetVariable, Do“AComputation“, GetVariable**.
With these three commands a parameter study with Motor-CAD can be run.

Blackbox Solvers can be used to parallelize computations for parameter studies / optimizations.

Further literature:

- General scripting tutorial: C:\ANSYS_Motor-CAD\14_1_8\Tutorials\ActiveX_Scripting.pdf
- FEA Geometry Scripting: C:\ANSYS_Motor-CAD\14_1_8\Tutorials\FEA_Geometry_Scripting
- OptiSLang Scripting: C:\ANSYS_Motor-CAD\14_1_8\Tutorials\Ansys_Optislang\Advance IPM

For the shown basic examples please contact me: psiehr@cadfem.de

Appendix – Full Minimal Example (1/2)

```python
# This Python Script is a minimal example for running Motor-CAD.
# It is meant to test the ActiveX connection to Motor-CAD.
# Only basic programming is used, especially no error handling is included.
# Input is loss data, output are temperatures.

import win32com.client

# Input Variables
i_loss_armature_copper = 250.0
i_loss_magnet = 0.0

# Output Variables
o_temp_magnet = 0.0
o_temp_winding_avg = 0.0
o_temp_winding_max = 0.0

# Create the Motor-CAD Object and load model
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.LoadFromFile("C:/Users/psiehr/Documents/motorcad/optiSLang/02_single_instance_motorcad/motor.mot")

# Disable Popups to avoid interuption
mcad.SetVariable('MessageDisplayState',2)
```

Appendix – Full Minimal Example (2/2)

```
# Setting the input variables mcad.SetVariable("Armature_Copper_Loss_@Ref_Speed", i_loss_armature_copper)
mcad.SetVariable("Magnet_Iron_Loss_@Ref_Speed",i_loss_magnet)

# Running the computation
mcad.DoSteadyStateAnalysis()

# Extracting temperatures
# Motor-CAD returns a tuple [success, value], hence [1] o_temp_winding_avg =
mcad.GetVariable("T_[Winding_Average]")[1] o_temp_winding_max = mcad.GetVariable("T_[Winding_Max]")[1]
o_temp_magnet = mcad.GetNodeTemperature(13)[1]

print("T_winding_max= ", o_temp_winding_max)
print("T_winding_avg= ", o_temp_winding_avg)
print("T_magnet= ", o_temp_magnet)

#Closing Motor-CAD
mcad.Quit()
```

Appendix – Torque Signal (1/2)

```python
# This Python Script is a minimal example for running Motor-CAD.
# It is meant to test the coupling of optiSLang and Motor-CAD.
# Only basic programming is used, especially no error handling is included.
# Input nothing, output is the torque signal

import win32com.client
# Create the Motor-CAD Object and load model
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.LoadFromFile("C:/Users/psiehr/Documents/konferenzen/2021_technologietage/Tag2/vortrag_psi/motor2.mot")

# Disable Popups to avoid interuption
mcad.SetVariable('MessageDisplayState',2)

PointsPerCycle = int(mcad.GetVariable('TorquePointsPerCycle')[1])
NumberCycles = int(mcad.GetVariable('TorqueNumberCycles')[1])
print("PointsPerCycle=", PointsPerCycle)
print("NumberCycle=", NumberCycles)

# Running the computation
mcad.DoMagneticCalculation()
```

Appendix – Torque Signal (2/2)

```
#Outputs
o_AvTorque = mcad.GetVariable('AvTorqueVW')[1]

N = (PointsPerCycle*NumberCycles)+1
position_list = []
torque_list =[]

for it in range(0,N-1):
          success, x, y = mcad.GetMagneticGraphPoint('TorqueVW', it)       position_list.append(x)
          torque_list.append(y)

average_torque = sum(torque_list)/len(torque_list) print('o_AvTorque=',o_AvTorque)
print('average_torque=',average_torque)

#Closing Motor-CAD
mcad.Quit()
```

Appendix – Minimal Example for optiSLang (1/2)

```
# This Python Script is a minimal example for running Motor-CAD.
# It is meant to test the coupling of optiSLang and Motor-CAD.
# Only basic programming is used, especially no error handling is included.
# Input is loss data, output are temperatures.

import win32com.client import time

# Check if Python is called outside optiSLang
if not 'OSL_REGULAR_EXECUTION' in locals():
             OSL_REGULAR_EXECUTION = False

# values to default if not existing yet
if not OSL_REGULAR_EXECUTION:
             # test run mode
             i_loss_armature_copper = 0.0
             i_loss_magnet = 0.0

# Output Vars
o_temp_magnet = 0.0
o_temp_winding_avg = 0.0
o_temp_winding_max = 0.0

# Create the Motor-CAD Object and load model
mcad = win32com.client.Dispatch("MotorCAD.AppAutomation")
mcad.LoadFromFile("C:/Users/psiehr/Documents/konferenzen/2021_technologietage/Tag2/vortrag_psi/motor.mot")
```

Appendix – Minimal Example for optiSLang (2/2)

```
# This pauses the execution for 20sec and is not necessary for real parameter studies.
# optiSLang will execute the parallel processes with a short delay (~5sec).
# Since this minimal example runs very fast, the parallel execution could not be observed without this pause.
time.sleep(20)

# Disable Popups to avoid interuption
mcad.SetVariable('MessageDisplayState',2)

# Setting the input variables
mcad.SetVariable("Armature_Copper_Loss_@Ref_Speed", i_loss_armature_copper)
mcad.SetVariable("Magnet_Iron_Loss_@Ref_Speed",i_loss_magnet)

# Running the computation
mcad.DoSteadyStateAnalysis()

# Extracting temperatures
# Motor-CAD returns a tuple [success, value], hence [1]
o_temp_winding_avg = mcad.GetVariable("T_[Winding_Average]")[1]
o_temp_winding_max = mcad.GetVariable("T_[Winding_Max]")[1]
o_temp_magnet = mcad.GetNodeTemperature(13)[1]

#Closing Motor-CAD
mcad.Quit()
```