

MotorXP-AFM

Design and Analysis of Axial Flux Machines with Permanent Magnets

Version 1.2

User Manual

Veeco Technologies Inc

April 2023

CONTENTS

PART I. User Interface

1. INTRODUCTION	7
2. BRIEF THEORETICAL BACKGROUND.....	8
2.1. General description.....	8
2.2. Finite element mesh.....	10
2.3. Calculation of electromagnetic torque and force.	11
2.4. A quasi-3D finite element modelling of an axial flux machine.	14
2.5. D-Q model of a permanent magnet machine.....	17
2.6. Iron loss calculation.....	19
2.7. Demagnetization of permanent magnets.	19
2.8. Calculation of inverter power losses.	21
2.8.1. Dynamic D-Q analysis inverter power losses calculation (method 1).....	21
2.8.2. Steady-state D-Q analysis inverter power losses calculation (method 2).....	24
3. GETTING STARTED	27
3.1. Using MotorXP-AFM MATLAB version.....	27
3.2. Using MotorXP-AFM Standalone version.....	27
3.3. MotorXP-AFM basics.	28
3.4. Working with project files.....	30
3.5. Licensing options.	31
3.6. Activating the license.	33
4. CREATING DESIGN PROTOTYPES	37
4.1. Geometry Editor.	38
4.1.1. Stator editing mode of Geometry Editor.....	44
4.1.2. Rotor editing mode of Geometry Editor.	47
4.2. Assigning materials.	50
4.2.1. Adding new materials.	52
4.3. Winding Editor.....	56

4.4.	Mesh Editor.....	65
4.5.	Drive Settings.....	68
5.	MAGNETOSTATIC FINITE ELEMENT ANALYSIS.....	87
5.1.	Running Magnetostatic FE Analysis.....	87
5.2.	Viewing Magnetostatic FE Analysis results.	92
5.2.1.	Time plots.	92
5.2.2.	Air gap distribution plots.	96
5.2.3.	Cross-section distribution plots.....	99
5.2.4.	Animation.....	102
6.	STEADY STATE D-Q ANALYSIS.....	103
6.1.	Building a D-Q model.	103
6.2.	Efficiency map and other performance maps.....	110
6.3.	Extracting Id and Iq current components tables from the efficiency map data.....	113
6.4.	Extracting D-Q model parameters.....	113
7.	DYNAMIC D-Q ANALYSIS.....	116
7.1.	Running Dynamic D-Q Analysis.	116
7.2.	Viewing Dynamic D-Q Analysis results.....	118
8.	DYNAMIC FINITE ELEMENT ANALYSIS	123
8.1.	Running Dynamic FE Analysis.....	123
8.2.	Viewing Dynamic FE Analysis results.	128
8.2.1.	Time-averaged quantities.....	128
8.2.2.	Plot wizard.	128
8.2.3.	Time plots.	128
8.2.4.	Air gap distribution plots.	133
8.2.5.	Cross-section distribution plots.....	136
9.	DYNAMIC FE ANALYSIS SIMULATION SCRIPT FUNCTIONS	140
9.1.	General information.	140
9.2.	Writing a simulation script function.....	140
9.3.	Main data structures.	142

10.	USING ELECTRICAL CIRCUITS	150
10.1.	Writing an electrical circuit function.....	150
10.1.1.	Electrical circuit branches.	151
10.1.2.	Adding circuit components.	152
10.2.	Controlling power sources and electronic switches using simulation script functions.	155
10.3.	Default three-phase inverter circuit function.....	157
11.	MOTORXP-AFM SETTINGS	159
	APPENDIX.....	161
	Appendix A. Power balance, discretization error and accuracy of the results.....	161
	Appendix B. Magnetostatic Analysis results.	162
	Appendix C. Out of memory errors handling.	163

PART II. MotorXP-AFM MATLAB scripting API

	Overview.....	165
1.	Scripting API initialization.....	166
2.	Functions for working with mxa-files.....	166
2.1.	<i>openMXA</i> function.....	166
2.2.	<i>saveDesign2MXAfile</i> function.....	166
3.	Functions for setting and getting parameters.....	167
3.1.	<i>setParamafm</i> function.....	167
3.2.	<i>motorProps</i> structure.....	167
3.3.	<i>settingsMagnetostatic</i> structure.....	176
3.4.	<i>getParamafm</i> function.....	179
4.	Serial processing functions.....	179
4.1.	Assembling AFM designs in series using the <i>assembleMXA</i> function.....	179
4.2.	Running magnetostatic FE simulations in series using <i>runMStimesteppingafm</i> function....	180
4.3.	Parametric sweep script example.....	186
5.	Parallel processing functions.....	188
5.1.	Assembling AFM designs in parallel.....	188
5.2.	Running magnetostatic FE simulations in parallel.....	189
5.3.	Parametric sweep script example with parallel processing	190
6.	Automatic optimization workflows.....	193
6.1.	Overview.....	193

6.2.	Functions for automatic optimization workflow development.....	193
6.3.	Example of the automatic optimization workflow implementation.....	197
7.	MATLAB code debugging tips.....	199

PART I

User Interface

1. INTRODUCTION

Thank you for your interest in MotorXP-AFM.

MotorXP-AFM is software for design and analysis of permanent magnet axial flux machines including brushless DC and permanent magnet synchronous machines. MotorXP-AFM is based on automated finite element analysis (FEA) simulations combined with analytical methods and establishes a complete set of tools for design and analysis of permanent magnet axial flux machines.

We believe that our work will help to save our environment by supporting green technologies like electric vehicles or wind turbine energy generation as well as enhancing the efficiency and effectiveness of electric machines.

2. BRIEF THEORETICAL BACKGROUND

The purpose of this chapter is to give the user a brief description of the problem examined. This chapter contains the formulation of the problem, short description of the nonlinear solver organization, rotation of the finite element mesh, torque and force calculation and etc. This information is critical for proper application adjustment and getting desired results.

2.1. General description.

The magnetic field problems for a conventional radial flux permanent magnet machine can be solved using a two-dimensional approximation, which is based on the assumption that the magnetic field does not depend on z -coordinate (z -axis being parallel to the axis of the rotor shaft). Thus, the magnetic field is solved in the plane of the machine's cross-section (x - y plane). The current density and magnetic vector potential in two-dimensional problems only have z -components and can be expressed as follows:

$$\nabla \times \left(\frac{1}{\mu_r} \cdot \nabla \times A \right) = J \quad (2.1)$$

$$J = \sigma \cdot \frac{U}{l} - \sigma \cdot \frac{\partial A}{\partial t}, \quad (2.2)$$

where A – magnetic vector potential ($B = \nabla \times A$, B – magnetic flux density), μ_r – relative magnetic permeability, J – current density, σ – conductivity, U – voltage applied to the FE area, l – length in z direction.

There are two methods used in the MotorXP software to solve the problem defined by Exp. (2.1) and (2.2). First method is called a *magnetostatic finite element method* (FEM) which is used in the Magnetostatic Analysis. For the magnetostatic FEM the current density J is assumed to be predefined so only Exp. (2.1) is used to define the magnetostatic problem. Note that the magnetostatic FEM can only be used for analysis of steady-state regimes.

Another method used in the MotorXP software is called a *transient finite element method* which is used in the Dynamic FE Analysis. According to Exp. (2.2) the current density consists of two components, the first one is caused by external voltage and the second one is induced by the varying magnetic field. The idea behind the transient FEM is a representation of the time derivative of the magnetic vector potential as follows:

$$\frac{\partial A}{\partial t} = \frac{A_n - A_{n-1}}{\Delta t},$$

where Δt – time step, A_n and A_{n-1} – magnetic vector potentials on n -th and $(n-1)$ -th time steps.

Determining the initial magnetic vector potentials with magnetostatic FEM and assuming zero initial stator currents the iterative procedure can be used to find the magnetic vector potentials for each time step. This method allows to take into account induced eddy currents, nonlinearity of iron and the rotation of the rotor and can be used for analysis of both transient and steady-state regimes.

In both methods the discretization of the problem over the plane of the machine's cross-section using FEM results in the system of linear equations written in the matrix form as follows:

$$K \cdot X = F, \quad (2.3)$$

where K – stiffness matrix, X – vector of unknowns, F – right side vector of the problem. For the magnetostatic FEM the unknowns are the magnetic vector potential values in mesh nodes. For the transient FEM the vector of unknowns also includes values of stator currents and voltages.

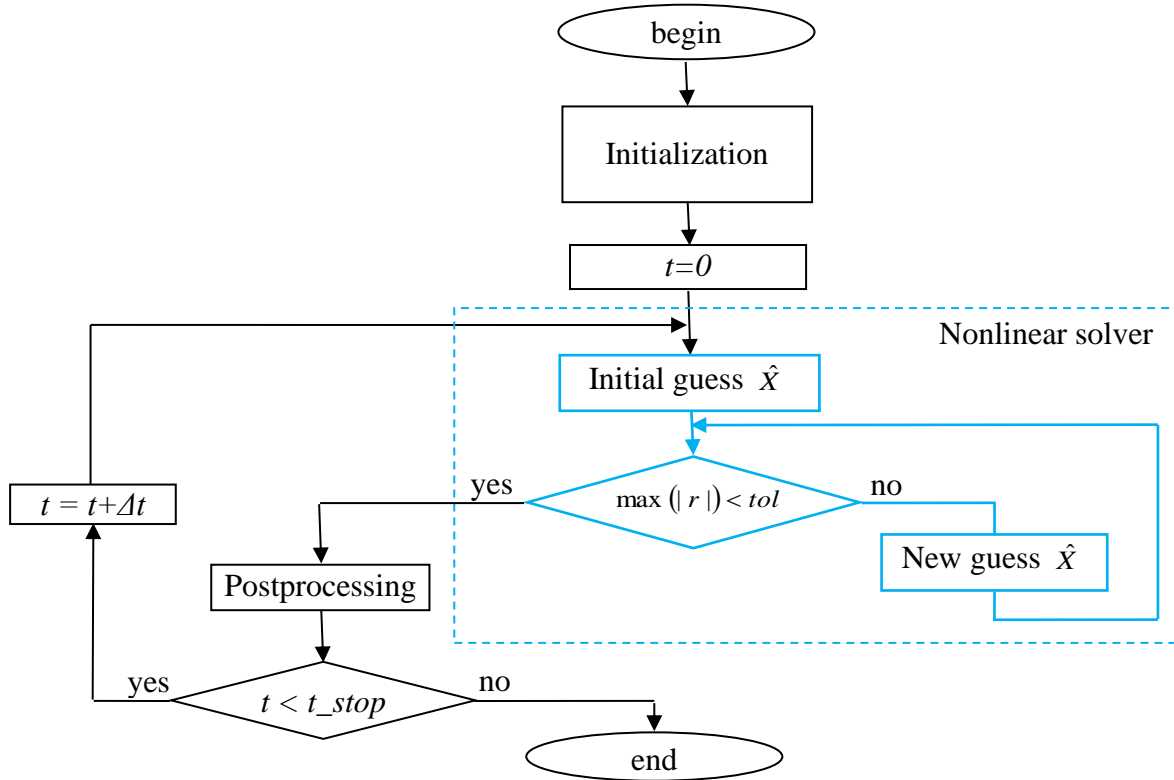


Figure 2.1. Generalized algorithm of solving a nonlinear problem with time-stepping FEM.

To solve the problem defined by Exp. (2.1) and (2.2) with a nonlinear B-H relationship the Gauss-Newton method is used. Assuming that we have a guess \hat{X} of the solution and stiffness matrix \hat{K} corresponding to the guess, then the residual vector of the guess \hat{X} will be defined as:

$$r = \hat{K} \cdot \hat{X} - F \quad (2.4)$$

Solving system (2.3) using Gauss-Newton iteration tends to be the minimization of the residual. The problem is said to be solved if the certain condition is satisfied. In MotorXP-AFM this condition is defined by the maximum absolute value of the residual vector:

$$\max(|r|) < tol \quad (2.5)$$

where tol – convergence tolerance defining the desired accuracy of the solution.

Both methods described above use an iterative time-stepping procedure to solve a time-varying magnetic field problem. Generalized algorithm used in the MotorXP software for a nonlinear case of the time stepping method is shown in Figure 2.1. An outer loop of the algorithm represents integration over time with step Δt . The magnetic field problem is solved on every integration step in the nonlinear solver loop (marked with blue color). Gauss-Newton iteration continues until the inequality (2.5) is satisfied.

2.2. Finite element mesh.

In MotorXP-AFM the first-order triangular finite element mesh is used.

Every time while the time-stepping FE simulation is in progress when the rotor position is changed the mesh should be also changed. It is implemented by rotating the rotor mesh connected to the fixed stator mesh via *sliding layer* located in the air gap. In MotorXP-AFM the air gap always has odd number of mesh layers (3, 5, 7 or 9) and the sliding layer is always located at the center of the air gap (see Figure 2.2).

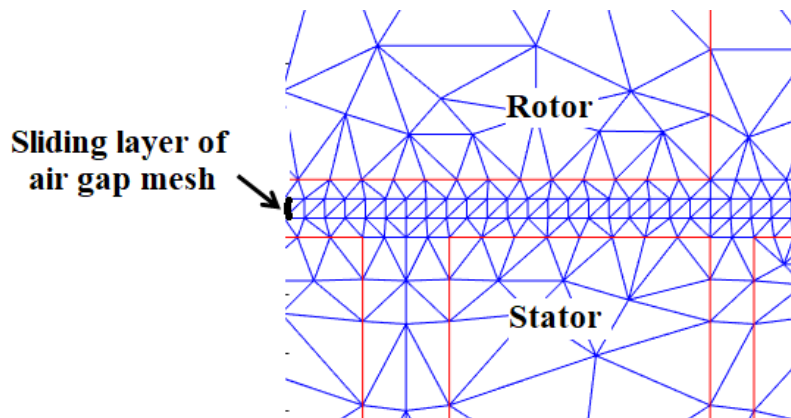


Figure 2.2. Three mesh layers in the air gap and the sliding layer at the center of the air gap.

To reduce number of finite elements and thus to reduce the computation time, periodic/antiperiodic boundary conditions can be applied. Using this type of boundary conditions is based on periodicity of the magnetic field of the machine.

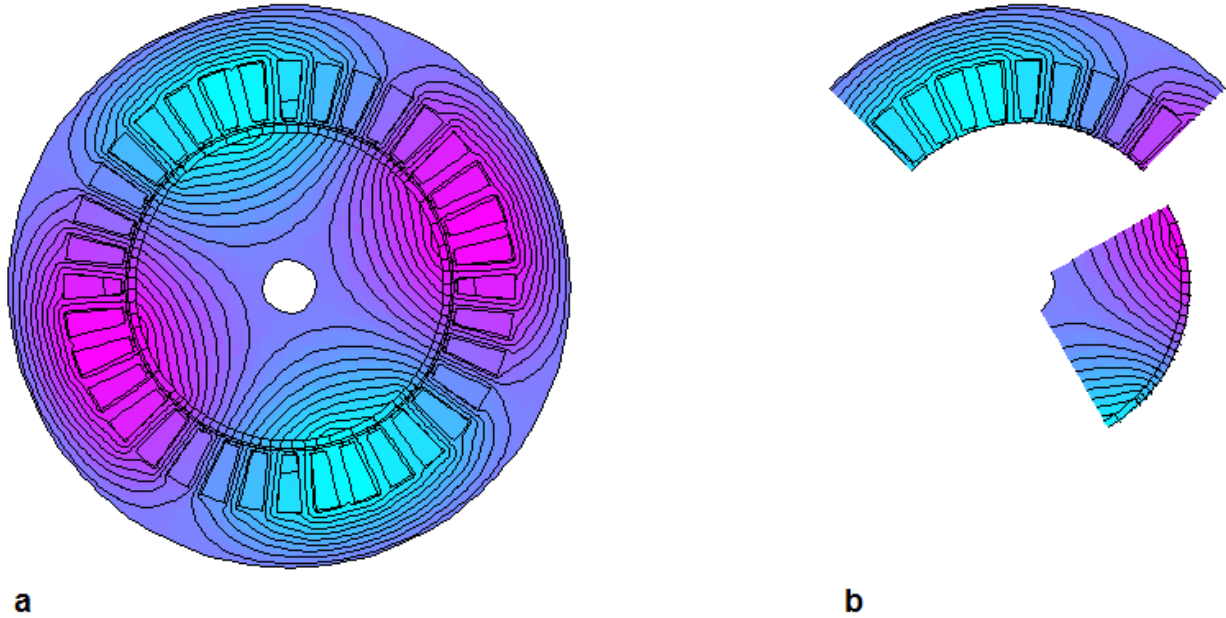


Figure 2.3. Using boundary conditions (on the example of a radial flux machine).

As it is seen from the example of four pole surface-mounted permanent magnet motor shown in Figure 2.3(a), the magnetic vector potentials repeat two times along the circle. Moreover, absolute values of the magnetic potential values repeat four times. Using this property of the field it is possible compute magnetic potentials for one quarter and then find the magnetic field for the whole cross-section as shown in Figure 2.3(b). In this case we use *antiperiodic* boundary conditions. If the magnetic potentials would be computed for the half, when *periodic* boundary conditions were applied. Note that using periodic or antiperiodic boundary conditions are only possible if geometry of the cross-section repeats together with the field. In practice it means that to apply periodic boundary conditions the number of stator slots should be divided by number of pole pairs, while for antiperiodic boundary conditions the number of stator slots should be divided by number of poles.

2.3. Calculation of electromagnetic torque and force.

There are two methods used in the MotorXP software for calculation of the torque. These are the Maxwell stress tensor method and virtual work method. For calculation of the axial force acting between the stator and rotor the virtual work method is used.

According to the Maxwell stress tensor method the electromagnetic torque for the inner rotor of a conventional radial flux machine can be expressed as the following:

$$T = - \frac{l \cdot (D_r + l_\delta)^2}{4\mu_0} \cdot \int_0^{2\pi} B_n B_t d\phi \quad (2.6)$$

where T – electromagnetic torque, l – lamination length in z direction, D_r – rotor diameter, l_δ – air gap length, μ_0 – permeability of the free space, B_n и B_t – normal and tangential components of the magnetic flux density in the air gap, as shown in Figure 2.4. This expression is used in the MotorXP software for calculation of the torque with the Maxwell stress tensor method.

The integration contour used in Exp. (2.6) always lies inside the sliding layer of the air gap mesh, as shown in Figure 2.4.

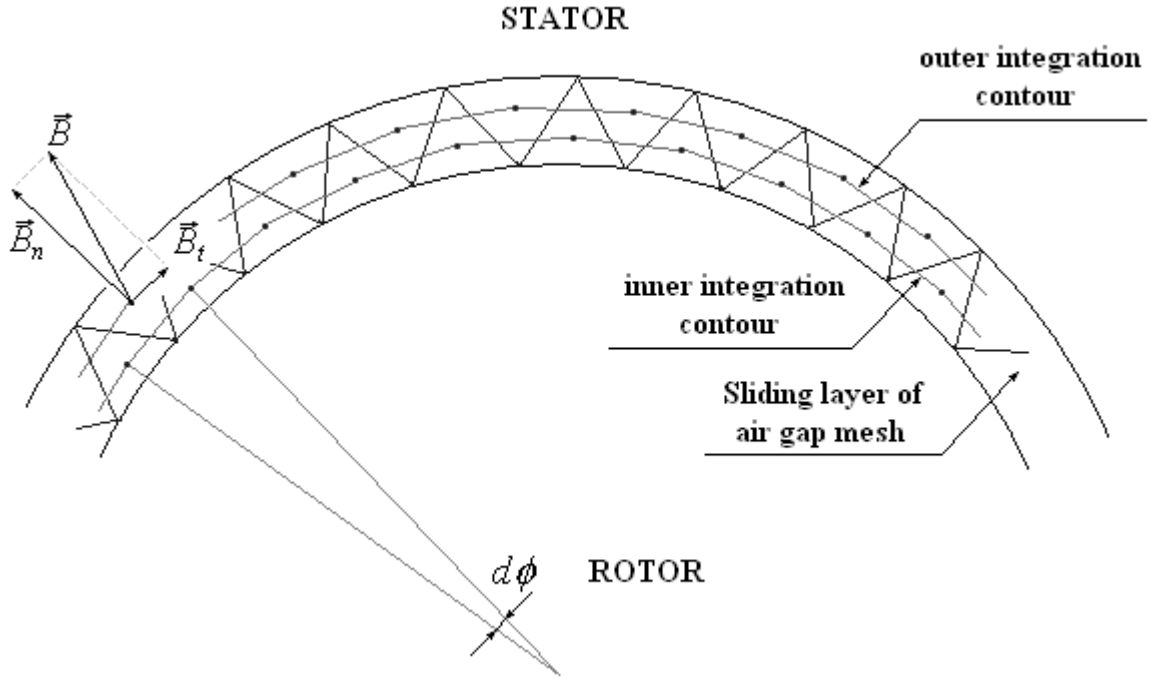


Figure 2.4. Calculation of the electromagnetic torque with the Maxwell stress tensor method (on the example of a radial flux machine).

Only the sliding layer of the air gap mesh is shown. Two integration contours are used - the inner contour passing through midpoints of the rotor-oriented triangles and the outer contour passing through midpoints of the stator-oriented triangles. The actual value of the torque is resulted as the average of two values calculated over the inner and outer integrated contours.

According to the virtual work principle, the electromagnetic force is given by the derivation of the magnetic energy with respect to the virtual displacement with constant flux linkage:

$$F_u = - \frac{\partial W}{\partial e} \bigg|_{\psi = \text{const}},$$

where ∂e - virtual displacement, ∂W - change of the magnetic energy between initial and final positions of the rotor.

Similarly, the expression for the torque calculation is defined as:

$$M = - \frac{\partial W}{\partial \phi} \bigg|_{\psi = \text{const}},$$

where $\partial \phi$ - virtual angular displacement.

When the virtual work principal is applied for the torque and force calculation, the accuracy significantly depends on the choice of the virtual displacement value. On the one hand the displacement should be small enough not to distort the finite element mesh. On the other hand, the round-off error arises when the displacement value is too small. The optimal value of the virtual displacement is not provided automatically and should be defined by the user in menu **File -> Settings**.

It is considered that the virtual work method is more accurate comparing with the Maxwell stress tensor one since the first one implies the integration over the whole machine's cross-section while using the Maxwell stress tensor method involves only finite elements of the air gap. Though in most cases the difference between two methods is not significant and the Maxwell stress tensor method is usually used. When the time-stepping FEM is applied the calculated torque value is used to determine the current rotor position. The rotor rotation is defined as follows:

$$T = T_{load} + J \frac{d\omega}{dt}$$

$$\omega = \frac{d\varphi}{dt},$$

where T_{load} – load torque on the motor shaft, J – combined rotor and load moment of inertia, ω - rotor angular speed, φ – rotor angular position.

2.4. A quasi-3D finite element modelling of an axial flux machine.

3D FEA is the most used technique for simulation of axial flux machines (AFM) due to their intrinsic 3D magnetic structure. But these simulations are extremely time consuming especially at initial design stages. There is also a well-know 2D FEA technique when the AFM is represented by one or several linear machines with equivalent topologies by slicing the 3D geometry of the AFM with cylindrical planes of different radii (2D Linear Machine Modeling Approach – 2D-LMMA*) as shown in Figure 2.5. Then the magnetic field problem is solved in each slice all at the same time with 2D FEM. The electromagnetic torque is calculated for every slice and the actual torque is obtained as a summation of all slice values.

The main drawback of 2D-LMMA is limited accuracy due to strong influence of 3D effects on the AFM performance. To include 3D effects into 2D-LMMA, MotorXP-AFM also computes the magnetic field in the radial cross-section of the AFM and uses geometry correction coefficients to take into account the exact shape of different parts of the machine. Example of the radial cross-section mesh of the AFM is shown in Figure 2.6.

The magnetic field of the AFM can be divided into circumferential and radial components. The circumferential component of the magnetic field is the one observed in the cylindrical cross-sections as shown in Figure 2.5 and can be accurately calculated by using 2D-LMMA. The radial component of the magnetic field is the one observed in the radial cross-sections as shown in Figure 2.6. The radial component is mostly determined by the end effects, such as the leakage flux along the radial cross-section plane and a variation of the inner and outer diameters of different parts of the AFM. The radial component of the magnetic field and the associated 3D effects are not directly included into 2D-LMMA but can be taken into account by also considering the magnetic field in the radial cross-section of the AFM. For example, Figure 2.6 shows AFM having different inner and outer diameters of stator teeth, rotor magnets and tooth tips. Magnetic field solution in the radial cross-section allows to consider the effect of these geometry characteristics on the AFM performance.

In order to reduce the number of machine cross-sections without sacrificing the accuracy while taking into account most of the 3D geometry details, the geometry correction coefficients are used. Figure 2.7 shows some examples of geometry details which are taken into account through the geometry correction coefficients. The geometry correction coefficients are automatically calculated for different parts of the AFM (rotor and stator back iron, tooth, magnet, and tooth tip) and incorporated into the AFM model.

The simulation time and accuracy are proportional to the number of cylindrical slices. The minimum number of cylindrical slices is 1 which can be used for getting fast simulation results at initial design stages. The recommended number of cylindrical slices is at least 3 providing sufficient accuracy in most cases.

* Gulec, M.; Aydin, M. Implementation of different 2-D finite element modelling approaches in axial flux permanent magnet disc machines. IET Electr. Power Appl. 2018, 12, 195–202.

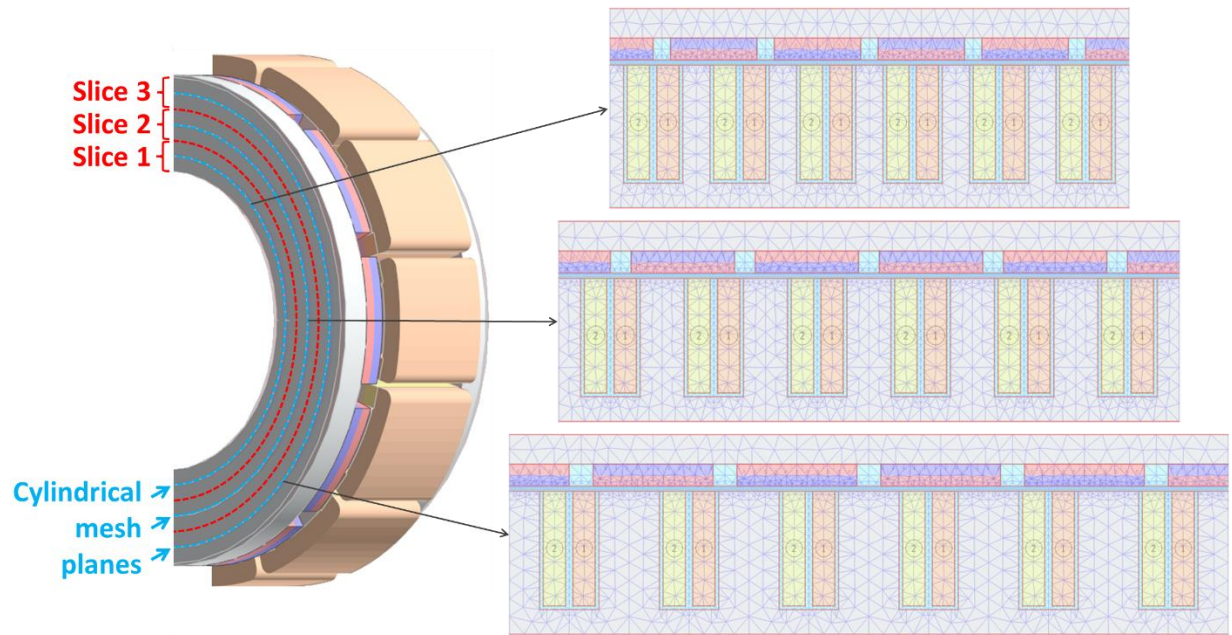


Figure 2.5. Axial flux machine 2D Linear Machine Modeling Approach (2D-LMMA) with three cylindrical mesh slices.

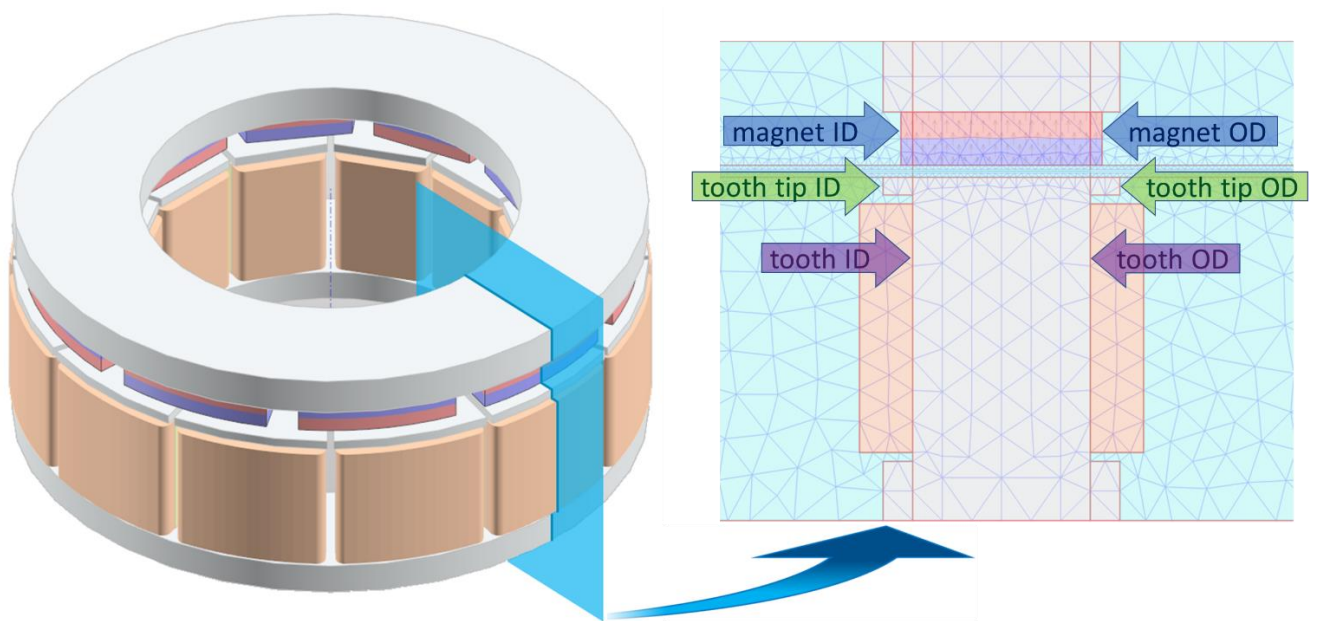
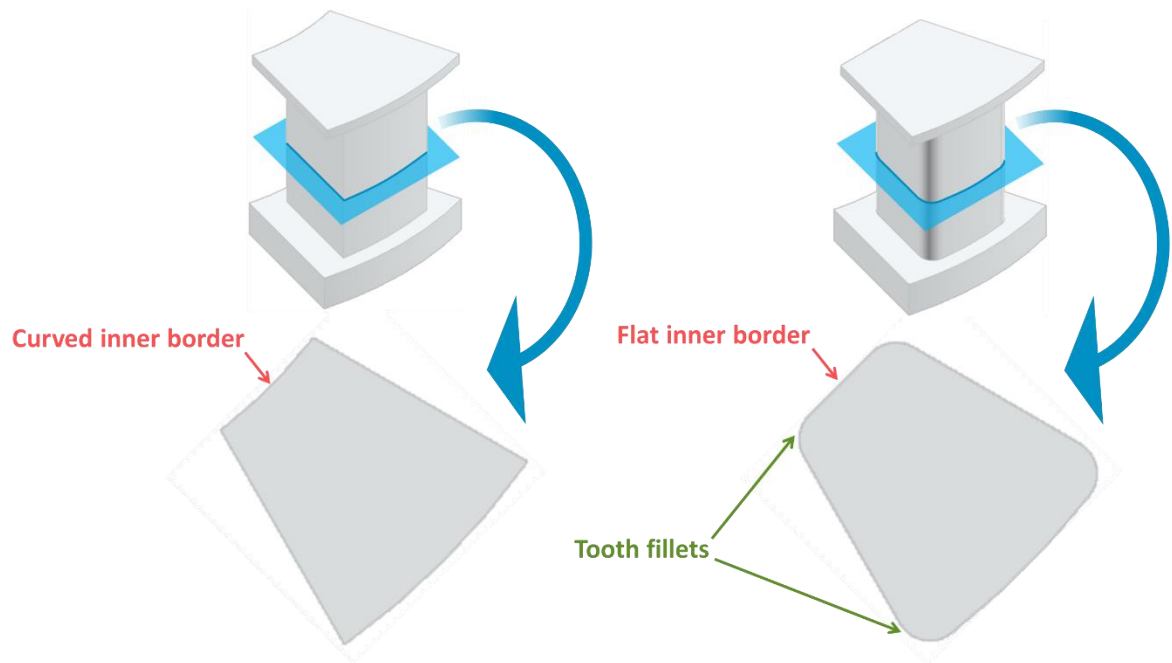
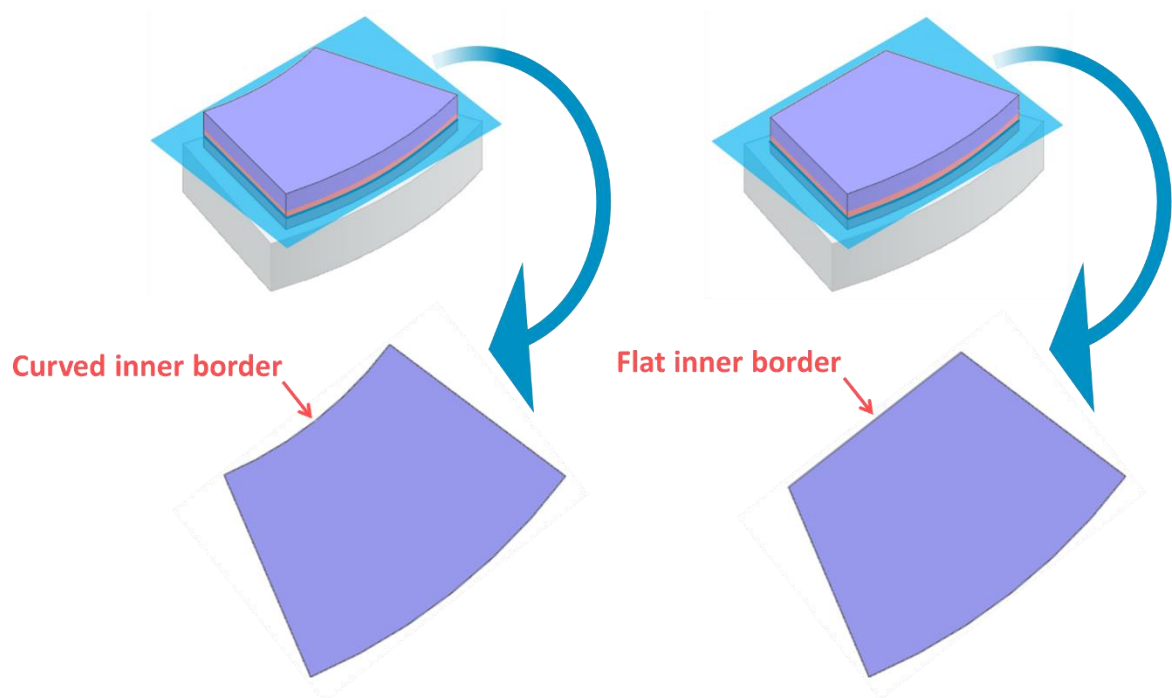


Figure 2.6. Axial flux machine radial cross-section mesh.



Examples of two stator tooth shapes which are taken into account using geometry correction coefficients



Examples of two rotor magnet shapes which are taken into account using geometry correction coefficients

Figure 2.7. Examples of use of the stator and rotor geometry correction coefficients.

2.5. D-Q model of a permanent magnet machine.

The dq-axes reference frame is a convenient way to represent sinusoidal quantities as constants. The dq-axes reference frame is fixed to the rotor and turning with the same speed as the rotor. As shown in Figure 2.8 on the example of a two pole surface-mounted permanent magnet rotor, the d-axis is always lying in the direction of the magnetic field of the permanent magnets. The q-axis is turned by 90 electrical degrees, i.e. it always lies between two magnetic poles of the machine.

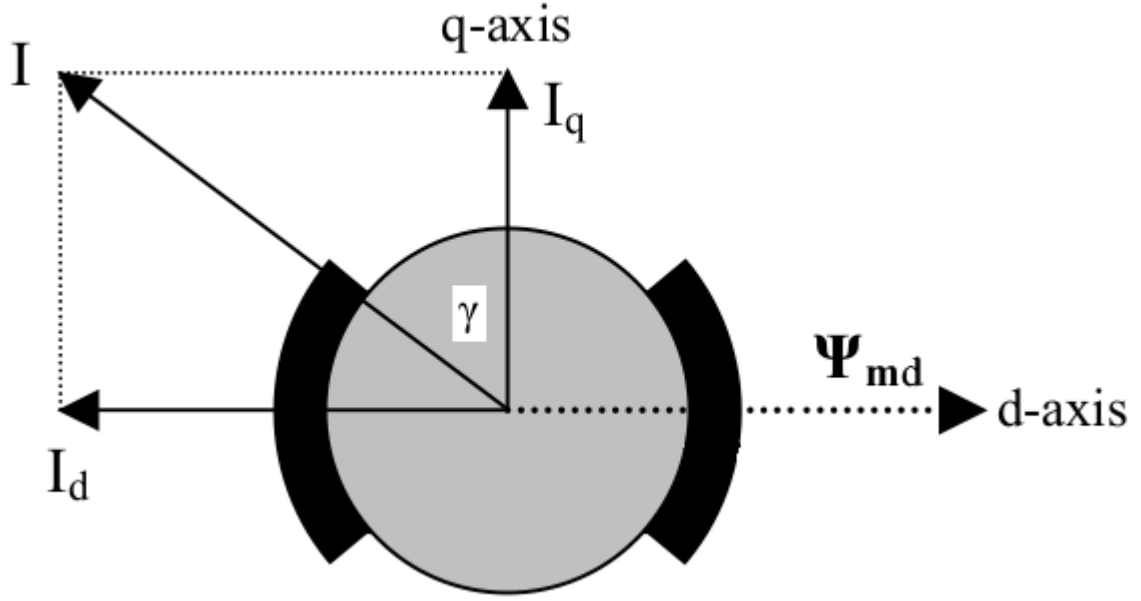


Figure 2.8. Definition of dq-axes reference frame in permanent magnet machines. ¹

As shown in Figure 2.8, the q-axis current I_q creates a magnetic field in the q-axis that interacts with the magnetic field of the permanent magnets Ψ_{md} lying in the d-axis to generate torque. The d-axis current I_d creates a magnetic field in the d-axis, which can be used to weaken the magnetic field of the permanent magnets when chosen negative (field weakening operation above base speed).

As shown in Figure 2.8, the current vector is defined by its peak value I and the advance angle γ towards the q-axis. The relationship between the amplitude and advance angle of the current vector and the dq-axes components is given by the following equations:

$$\begin{aligned}
 I_q &= I \cos \gamma \\
 I_d &= -I \sin \gamma \\
 I &= \sqrt{I_d^2 + I_q^2} \\
 \gamma &= \arctan \left(-\frac{I_d}{I_q} \right)
 \end{aligned} \tag{2.7}$$

¹ <https://www.emotor.com/blog/post/blac-machine-simulations/>

To take into account the effect of cross-saturation the cross-saturation inductance L_{dq} and cross saturation magnet flux linkage Ψ_{mqd} lying in the q-axis are included. With the cross saturation terms, the d-axis and q-axis flux linkages are given as follows:

$$\begin{aligned}\Psi_d &= \Psi_{md} + L_d I_d + L_{dq} I_q \\ \Psi_q &= \Psi_{mqd} + L_q I_q + L_{dq} I_d\end{aligned}\tag{2.8}$$

Where L_d , L_q , L_{dq} , Ψ_{md} and Ψ_{mqd} values depend on I_d and I_q to take into account saturation.

The steady state equations after resolving voltages into d and q components can be written in the general form:

$$\begin{aligned}V_d &= R_s I_d - \omega \Psi_q - \omega L_{sew} I_q \\ V_q &= R_s I_q + \omega \Psi_d + \omega L_{sew} I_d \\ V &= \sqrt{V_d^2 + V_q^2}\end{aligned}\tag{2.9}$$

The electromagnetic torque is calculated using the well-known equation:

$$T = \frac{3}{2} p (\Psi_d I_q - \Psi_q I_d)\tag{2.10}$$

Where R_s – stator winding phase resistance, L_{sew} – stator end winding inductance, ω – electrical operating speed, p – number of pole pairs.

The calculation of D-Q model parameters L_d , L_q , L_{dq} , Ψ_{md} and Ψ_{mqd} in MotorXP-AFM is based on finite element method with «permeance freezing» which allows using superposition to extract individual parameters of the machine under load and preserve information about saturation while taking into account the effect of cross-saturation as well.²

And finally, the dynamic equations written for the voltage components are given by the following expressions:

$$\begin{aligned}V_d &= \frac{d\Psi_d}{dt} + L_{sew} \frac{dI_d}{dt} + R_s I_d - \omega \Psi_q - \omega L_{sew} I_q \\ V_q &= \frac{d\Psi_q}{dt} + L_{sew} \frac{dI_q}{dt} + R_s I_q + \omega \Psi_d + \omega L_{sew} I_d\end{aligned}\tag{2.11}$$

² Žarko, Damir, Ban, Drago, Klarić, Ratko. (2006). Finite Element Approach to Calculation of Parameters of an Interior Permanent Magnet Motor. AUTOMATIKA: Journal for Control, Measurement, Electronics, Computing and Communications; Vol.46, No.3-4.

2.6. Iron loss calculation

Iron core losses in MotorXP-AFM are calculated during the post processing so it assumes that the iron losses do not influence the magnetic field distribution. There is a well-known expression for the iron loss estimation based on the Steinmetz equation:

$$P = K_h f^\alpha B_m^\beta + K_e (f \cdot B_m)^2 \quad (2.12)$$

First term in Exp. 2.12 represents hysteresis losses, the second one – eddy current loss term. Coefficients K_h , α , β and K_e are determined by fitting equation 2.12 to the measured iron loss data from the manufacturer at different frequencies.

Exp. 2.12 allows to estimate the iron losses at a single frequency, which is usually a supply frequency (fundamental harmonic iron losses). Estimation of the iron losses for other flux density harmonics with Exp. 2.12 would require FFT procedure, which is associated with long simulation times to provide the sufficient frequency resolution. Instead, MotorXP-AFM uses more advanced dynamic iron loss model³. This model uses the same coefficients as in Exp. 2.12 and takes into consideration the non-sinusoidal shape of the flux density waveform without using FFT. It allows one to include into analysis the iron loss contribution from higher harmonics of the flux density such as slot harmonics and PWM produced harmonics for both stator and rotor cores as well as minor hysteresis loops.

The same iron loss model is used for both magnetostatic FEA and transient FEA (Dynamic FE Analysis). Note that D-Q analysis in MotorXP-AFM takes into account only the fundamental harmonic iron losses.

2.7. Demagnetization of permanent magnets.

Figure 2.9 shows an example of demagnetization curves for permanent magnet material N52. Irreversible demagnetization occurs if the working point in any part of the permanent magnet exceeds the linear range, i.e. falls below the "knee" of the demagnetization curve, as shown in Figure 2.9. It means that the magnet irreversibly changes its properties (remanence flux density is irreversibly reduced) and the new demagnetization curve shown by dotted line emerges.

Important parameter which characterizes the ability of permanent magnet to resist the irreversible demagnetization is intrinsic coercivity H_{cj} . Intrinsic coercivity is the applied demagnetizing field required to fully demagnetize the permanent magnet material so when the demagnetizing field is removed the material does not act like a magnet anymore. The higher intrinsic coercivity the higher demagnetizing field can be sustained by the permanent magnet without risk of demagnetization.

It is a good practice to always check your simulations for the possible risk of demagnetization at worst operating conditions (maximum current, advance angle and temperature of permanent magnet) including

³ D. Lin, P. Zhou, W. N. Fu, Z. Badics and Z. J. Cendes, "A dynamic core loss model for soft ferromagnetic and power ferrite materials in transient finite element analysis," in IEEE Transactions on Magnetics, vol. 40, no. 2, pp. 1318-1321, March 2004.

possible fault conditions. In MotorXP-AFM the risk of demagnetization can be estimated from the maximum demagnetizing field intensity experienced by the permanent magnet during simulation. The maximum demagnetizing field intensity is available measured in A/m or in percentage of intrinsic coercivity H_{cj} . If the demagnetization curve for the given permanent magnet material is available, make sure that the maximum demagnetizing field value is to the right of the "knee" of the demagnetization curve. For neodymium magnets there is usually no risk of demagnetization if the maximum demagnetizing field is less than 70% of intrinsic coercivity.

As seen from Figure 2.9 the demagnetization curve changes significantly with the temperature and the risk of demagnetization increases at higher temperatures of the permanent magnet. MotorXP-AFM varies properties of the permanent magnet depending on the temperature so to better estimate the risk of demagnetization it is recommended to check simulation results with the maximum expected temperature of the permanent magnet. You should also keep in mind that the material data and demagnetization curves represent typical properties that may vary due to magnet shape and size with tolerances of up to $\pm 10\%$.

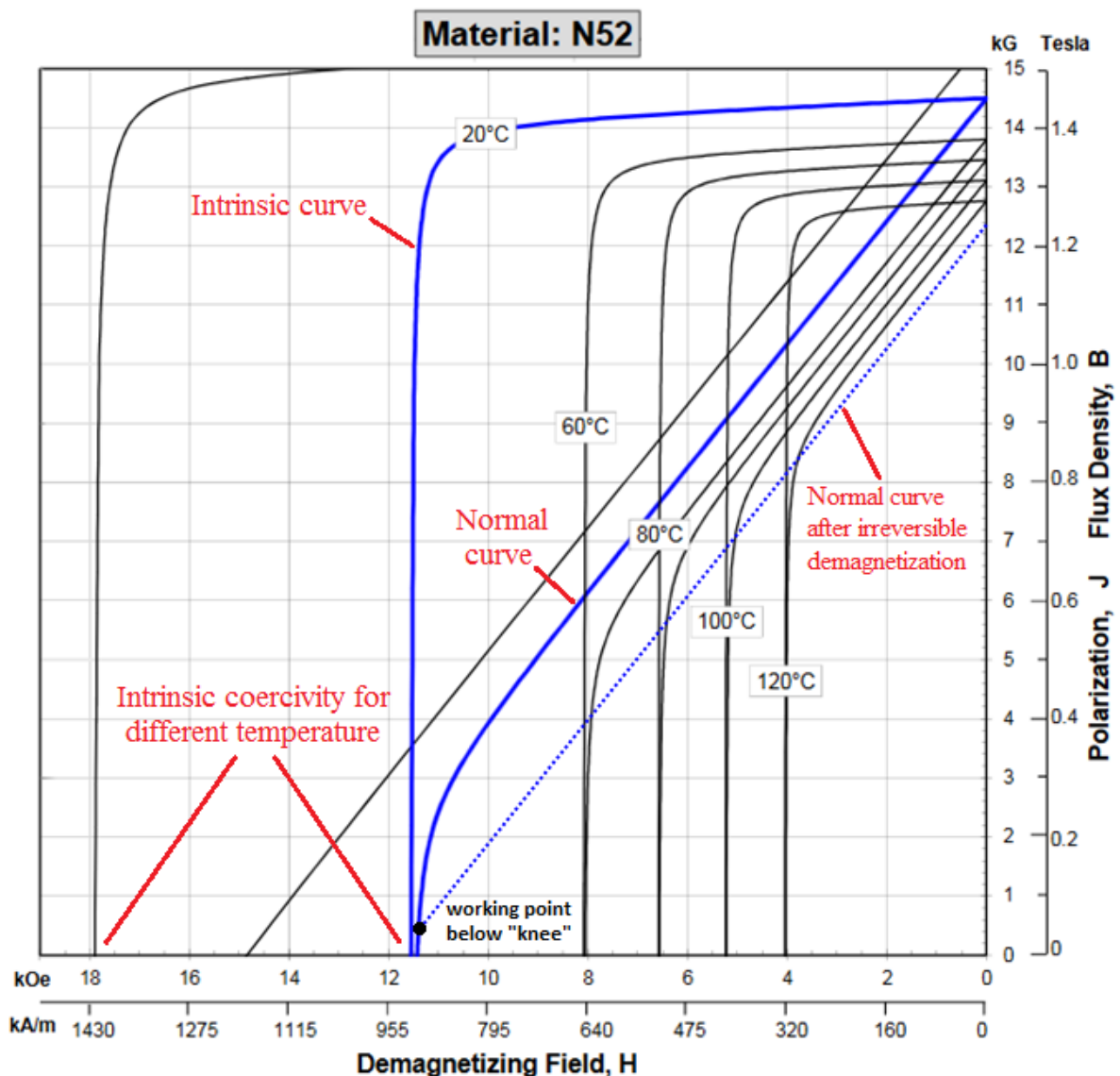


Figure 2.9. Demagnetization curves for permanent magnet material N52.

2.8. Calculation of inverter power losses.

Power losses in any semiconductor component, such as IGBT and MOSFET transistor or diode, operating in the switch-mode can be divided in conduction losses and switching losses. Conduction losses occur during on-state operation because of finite value of the on-state resistance. Switching losses occur when the semiconductor component changes its state (from on-state to off-state or from off-state to on-state) because of finite switching time.

There are two methods in MotorXP-AFM used for the inverter power losses calculation⁴. First method is used with **Dynamic D-Q Analysis** (see chapter 7) simulation and consists of calculating the instantaneous conduction and switching losses for every transistor and diode of the inverter for every time step and every switching event. This method is more accurate, but the computational speed is limited by the speed of the dynamic D-Q simulation. Second method is used with **Steady State D-Q Analysis** (see chapter 6) only for the space-vector PWM supply and based on the analytical expressions with the use of the peak inverter output current value. This method is very fast, but the accuracy is less, and the applicability is limited only by the space-vector PWM power supply.

Refer to section 4.5.1 to learn which parameters of the transistor are required for the inverter power losses calculation and how they can be extracted from the datasheet data.

2.8.1. *Dynamic D-Q analysis inverter power losses calculation (method 1).*

This method is used with **Dynamic D-Q Analysis** and suitable for all power supply types. The inverter power loss calculation is considered for two cases when transistors of IGBT and MOSFET type are used.

2.8.1.1. Transistor of MOSFET type.

Instantaneous conduction power losses of the MOSFET:

$$P_c^{MOSFET} = R_{DSon}(i_D) \cdot i_D^2,$$

where R_{DSon} – drain-to-source on-state resistance, i_D – instantaneous drain current.

Instantaneous conduction power losses of the freewheeling diode:

$$P_c^{diode} = v_f \cdot i_f$$

$$v_f = V_{f0} + R_f(i_f) \cdot i_f,$$

where v_f – instantaneous voltage across the diode, i_f – instantaneous diode current, R_f – diode on-state resistance, V_{f0} – diode on-state zero-current voltage.

⁴ D. Graovac, M. Pürschel, “IGBT Power Losses Calculation Using the Data-Sheet Parameters”, Application Note – Infineon, Jan. 2009.

D. Graovac, M. Pürschel, A. Kiep, “MOSFET Power Losses Calculation Using the Data-Sheet Parameters”, Application Note – Infineon, July 2006.

$$E_{on}^{MOSFET} = V_{DC} \cdot i_{Don} \cdot \frac{t_{ri} + t_{fu}}{2} + Q_{rr} \cdot V_{DC}$$

$$E_{on}^{diode} = \frac{1}{4} \cdot Q_{rr} \cdot V_{DC},$$

where V_{DC} – DC supply voltage, i_{Don} – drain current after the switching event, t_{ri} – drain current rise time from the datasheet, t_{fu} – voltage fall time, Q_{rr} – diode reverse recovery charge from the datasheet.

The voltage fall time is defined as follows:

$$t_{fu} = \frac{t_{fu1} + t_{fu2}}{2}$$

$$t_{fu1} = (V_{DC} - R_{DSon}(i_D) \cdot i_{Don}) \cdot R_{Dr} \cdot \frac{C_{GD1}}{V_{Dr} - V_{plateau}}$$

$$t_{fu2} = (V_{DC} - R_{DSon}(i_D) \cdot i_{Don}) \cdot R_{Dr} \cdot \frac{C_{GD2}}{V_{Dr} - V_{plateau}}$$

$$C_{GD1} = C_{rss}(V_{DS} = V_{DC})$$

$$C_{GD2} = C_{rss}(V_{DS} = R_{DSon}(i_D) \cdot i_{Don}),$$

where R_{Dr} – gate driver circuit resistance, V_{Dr} – gate driver circuit output voltage, $V_{plateau}$ – gate Muller plateau voltage from the datasheet, C_{GD1} and C_{GD2} – reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage $C_{rss} = f(V_{DS})$ at different V_{DS} levels.

Instantaneous on-state to off-state switching energy loss:

$$E_{off}^{MOSFET} = V_{DC} \cdot i_{Doff} \cdot \frac{t_{ru} + t_{fi}}{2}$$

where i_{Doff} – drain current before the switching event, t_{fi} – drain current fall time from the datasheet, t_{ru} – voltage rise time. The switch-off losses in the diode are neglected ($E_{off}^{diode} = 0$).

The voltage rise time is defined as follows:

$$t_{ru} = \frac{t_{ru1} + t_{ru2}}{2}$$

$$t_{ru1} = (V_{DC} - R_{DSon}(i_D) \cdot i_{Doff}) \cdot R_{Dr} \cdot \frac{C_{GD1}}{V_{plateau}}$$

$$t_{ru2} = (V_{DC} - R_{DSon}(i_D) \cdot i_{Doff}) \cdot R_{Dr} \cdot \frac{C_{GD2}}{V_{plateau}}$$

$$C_{GD1} = C_{rss}(V_{DS} = V_{DC})$$

$$C_{GD2} = C_{rss}(V_{DS} = R_{DSon}(i_D) \cdot i_{Doff}),$$

where C_{GD1} and C_{GD2} – reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage $C_{rss} = f(V_{DS})$ at different V_{DS} levels.

The instantaneous switching power losses are calculated as a sum of the switching energy losses over all switching events divided by the considered time interval t :

$$P_{sw}^{MOSFET} = \frac{\Sigma(E_{on}^{MOSFET}) + \Sigma(E_{off}^{MOSFET})}{t}$$

$$P_{sw}^{diode} = \frac{\Sigma(E_{on}^{diode})}{t}$$

2.8.1.2. Transistor of IGBT type.

Instantaneous conduction power losses of the IGBT:

$$P_c^{IGBT} = v_{CE}(i_C) \cdot i_C$$

$$v_{CE} = v_{CE0}(i_C) + R_C(i_C) \cdot i_C,$$

where v_{CE} – instantaneous collector-emitter on-state voltage, i_C – instantaneous collector current, v_{CE0} – instantaneous zero-current collector-emitter on-state voltage, R_C – collector-emitter on-state resistance.

Instantaneous conduction power losses of the freewheeling diode:

$$P_c^{diode} = v_f(i_f) \cdot i_f$$

$$v_f = v_{f0}(i_f) + R_f(i_f) \cdot i_f,$$

where v_f – instantaneous voltage across the diode, i_f – instantaneous diode current, R_f – diode on-state resistance, v_{f0} – diode on-state zero-current voltage.

Instantaneous turn-on and turn-off IGBT switching energy losses E_{on}^{IGBT} and E_{off}^{IGBT} are calculated from the corresponding turn-on and turn-off energy vs. collector current diagrams from the datasheet (see section 4.5.1).

Instantaneous diode turn-on switching energy losses consist mostly of the reverse-recovery energy:

$$E_{on}^{diode} = \frac{1}{4} \cdot Q_{rr} \cdot V_{DC},$$

where V_{DC} – DC supply voltage, Q_{rr} – diode reverse recovery charge from the datasheet.

The turn-off losses in the diode are neglected ($E_{off}^{diode} = 0$).

The instantaneous switching power losses are calculated as a sum of the switching energy losses over all switching events divided by the considered time interval t :

$$P_{sw}^{IGBT} = \frac{\Sigma(E_{on}^{IGBT}) + \Sigma(E_{off}^{IGBT})}{t}$$

$$P_{sw}^{diode} = \frac{\Sigma(E_{on}^{diode})}{t}$$

2.8.2. Steady-state D-Q analysis inverter power losses calculation (method 2).

This method is used with **Steady State D-Q Analysis** and suitable only for the space-vector PWM power supply. The inverter power losses calculation is considered for two cases when transistors of IGBT and MOSFET type are used.

2.8.2.1. Transistor of MOSFET type.

It is assumed that the freewheeling diodes do not open with the space-vector PWM since MOSFET conducts current in both directions and the on-state voltage across the MOSFET is very low (much less than the diode on-state zero-current voltage), so only MOSFET losses are taken into account.

Forward direction conduction power losses of the MOSFET:

$$P_{CF}^{MOSFET} = R_{DSon} \cdot I_{Om}^2 \cdot \left(\frac{1}{8} + \frac{m_a \cdot PF}{3 \cdot \pi} \right)$$

Reverse direction conduction power losses of the MOSFET:

$$P_{CR}^{MOSFET} = R_{DSon} \cdot I_{Om}^2 \cdot \left(\frac{1}{8} - \frac{m_a \cdot PF}{3 \cdot \pi} \right),$$

where R_{DSon} – drain-to-source on-state resistance, I_{Om} – peak inverter output current, m_a – inverter amplitude modulation index (0 - 0.8660 for space-vector PWM), PF – motor power factor.

Off-state to on-state switching energy losses:

$$E_{On}^{MOSFET} = V_{DC} \cdot \left(\frac{I_{Om}}{\pi} \right) \cdot \frac{t_{ri} + t_{fu}}{2} + Q_{rr} \cdot V_{DC},$$

where V_{DC} – DC supply voltage, t_{ri} – drain current rise time from the datasheet, t_{fu} – voltage fall time, Q_{rr} – diode reverse recovery charge from the datasheet.

The voltage fall time is defined as follows:

$$\begin{aligned} t_{fu} &= \frac{t_{fu1} + t_{fu2}}{2} \\ t_{fu1} &= \left(V_{DC} - R_{DSon} \cdot \left(\frac{I_{Om}}{\pi} \right) \right) \cdot R_{Dr} \cdot \frac{C_{GD1}}{V_{Dr} - V_{plateau}} \\ t_{fu2} &= \left(V_{DC} - R_{DSon} \cdot \left(\frac{I_{Om}}{\pi} \right) \right) \cdot R_{Dr} \cdot \frac{C_{GD2}}{V_{Dr} - V_{plateau}} \\ C_{GD1} &= C_{rss}(V_{DS} = V_{DC}) \\ C_{GD2} &= C_{rss} \left(V_{DS} = R_{DSon} \cdot \left(\frac{I_{Om}}{\pi} \right) \right), \end{aligned}$$

where R_{Dr} – gate driver circuit resistance, V_{Dr} – gate driver circuit output voltage, $V_{plateau}$ – gate Muller plateau voltage from the datasheet, C_{GD1} and C_{GD2} - reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage $C_{rss} = f(V_{DS})$ at different V_{DS} levels.

On-state to off-state switching energy losses:

$$E_{off}^{MOSFET} = V_{DC} \cdot \left(\frac{I_{Om}}{\pi} \right) \cdot \frac{t_{ru} + t_{fi}}{2}$$

where t_{fi} – drain current fall time from the datasheet, t_{ru} – voltage rise time.

The voltage rise time is defined as follows:

$$\begin{aligned} t_{ru} &= \frac{t_{ru1} + t_{ru2}}{2} \\ t_{ru1} &= \left(V_{DC} - R_{DSon} \cdot \left(\frac{I_{Om}}{\pi} \right) \right) \cdot R_{Dr} \cdot \frac{C_{GD1}}{V_{plateau}} \\ t_{ru2} &= \left(V_{DC} - R_{DSon} \cdot \left(\frac{I_{Om}}{\pi} \right) \right) \cdot R_{Dr} \cdot \frac{C_{GD2}}{V_{plateau}} \\ C_{GD1} &= C_{rss}(V_{DS} = V_{DC}) \\ C_{GD2} &= C_{rss} \left(V_{DS} = R_{DSon} \cdot \left(\frac{I_{Om}}{\pi} \right) \right), \end{aligned}$$

where C_{GD1} and C_{GD2} - reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage $C_{rss} = f(V_{DS})$ at different V_{DS} levels.

The switching losses are the product of switching energies and the switching frequency (f_s):

$$P_{sw}^{MOSFET} = 2 \cdot (E_{on}^{MOSFET} + E_{off}^{MOSFET}) \cdot f_s$$

2.8.2.2. Transistor of IGBT type.

Conduction power losses of the IGBT:

$$P_C^{IGBT} = V_{CE0} \cdot I_{Om} \cdot \left(\frac{1}{2 \cdot \pi} + \frac{m_a \cdot PF}{8} \right) + R_C \cdot I_{Om}^2 \cdot \left(\frac{1}{8} + \frac{m_a \cdot PF}{3 \cdot \pi} \right)$$

Conduction power losses of the freewheeling diode:

$$P_C^{diode} = V_{f0} \cdot I_{Om} \cdot \left(\frac{1}{2 \cdot \pi} - \frac{m_a \cdot PF}{8} \right) + R_f \cdot I_{Om}^2 \cdot \left(\frac{1}{8} - \frac{m_a \cdot PF}{3 \cdot \pi} \right),$$

where V_{CE0} – zero-current collector-emitter on-state voltage, R_C – collector-emitter on-state resistance, V_{f0} – diode on-state zero-current voltage, R_f – diode on-state resistance, I_{Om} – peak inverter output current, m_a – inverter amplitude modulation index (0 - 0.8660 for space-vector PWM), PF – motor power factor. Turn-on and turn-off IGBT switching energy losses E_{on}^{IGBT} and E_{off}^{IGBT} are calculated from the corresponding turn-on and turn-off energy vs. collector current diagrams from the datasheet (see section 4.5.1).

Diode turn-on switching energy losses consist mostly of the reverse-recovery energy:

$$E_{on}^{diode} = \frac{1}{4} \cdot Q_{rr} \cdot V_{DC},$$

where V_{DC} – DC supply voltage, Q_{rr} – diode reverse recovery charge from the datasheet.

The turn-off losses in the diode are neglected ($E_{off}^{diode} = 0$).

The switching losses are the product of switching energies and the switching frequency (f_s):

$$P_{sw}^{IGBT} = (E_{on}^{IGBT} + E_{off}^{IGBT} + E_{on}^{diode}) \cdot f_s$$

3. GETTING STARTED

3.1. Using MotorXP-AFM MATLAB version.

If you do not have MATLAB® refer to the next section to learn how to get started with MotorXP-AFM Standalone version.

MotorXP-AFM MATLAB version is a MATLAB-application and to use it you need to have MATLAB® installed on your computer. MotorXP-AFM was currently tested with MATLAB versions starting from MATLAB R2017a and later.

You do not need to install the MATLAB version of MotorXP-AFM – just copy all the file on your computer. To start MotorXP-AFM enter `motorxp` in the MATLAB Command Window. Note that the MATLAB Current Folder should be changed to the folder with the application files. MotorXP-AFM main window shown in Figure 3.1 will appear.

3.2. Using MotorXP-AFM Standalone version.

MotorXP-AFM Standalone version works like any Windows program and does not require MATLAB®. Use installation file *MotorXPv1.1Installer.exe* to install MotorXP-AFM Standalone version. Internet connection is required during installation since the installation package does not include MATLAB Compiler Runtime components.

If you receive a message stating "Error finding installer class" when trying to install MotorXP-AFM Standalone version, most likely, illegal characters are present in the path to the installation folder, or in the path to the Windows TEMP folder, or in the path to the folder containing the MotorXP-AFM installer. Visit <https://www.mathworks.com/matlabcentral/answers/92499-why-do-i-receive-a-message-stating-error-finding-installer-class-when-trying-to-install-matlab-on> for more details.

Important notice: *MotorXP-AFM Standalone version has several limitations* comparing to MotorXP-AFM MATLAB version. These limitations are related to the fact that the MCR application is not able to execute m-files which were not included while the application was compiled. Note that several m-files are provided together with the program. These m-files were wrapped into `motorxp.exe` file while the application was compiled. If you change any of these m-files or add your own m-file it will not have any effect.

Limitations of MotorXP-AFM Standalone version:

1. Dynamic FE Analysis simulation script functions (see chapter 8 and 9) cannot be used except those provided together with the program (*simscript_hystpwm.m*, *simscript_spacevecpwm.m*, *simscript_sixstep.m*);

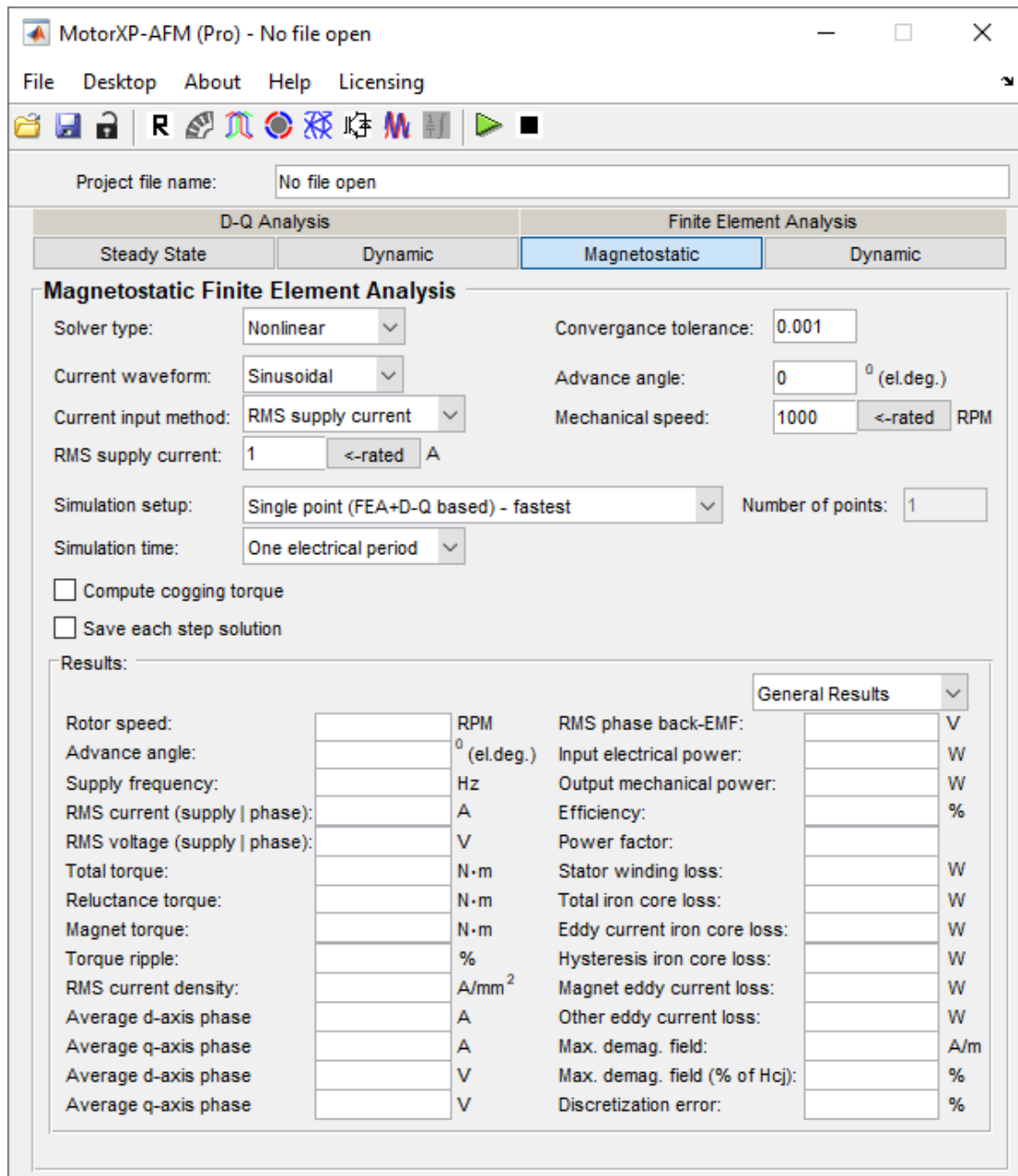


Figure 3.1. MotorXP-AFM main window.

2. Electrical circuit functions (see chapter 10) cannot be used except those provided together with the program (*StarConnection.m*, *DeltaConnection.m*, *InverterCircuit.m*);

If you use MotorXP-AFM Standalone version, keep in mind these limitations while reading this manual since some features described in the manual are available only in MotorXP-AFM MATLAB version.

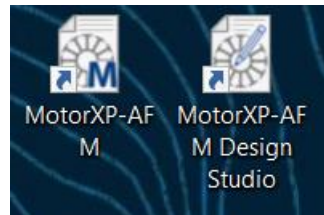
3.3. MotorXP-AFM basics.

When you start MotorXP-AFM, the main window appears as shown in Figure 3.1.

Using toolbar buttons, you can get access to all MotorXP-AFM tools to create and analyze the design prototypes such as MotorXP-AFM Design Studio (consisting of **Geometry Editor** / **Materials**, **Winding Editor** and **Mesh Editor** tabs), **Drive Settings**, **Rated Data** and **Plot Wizard**. These are also available from the **Desktop** menu. Next chapters provide the detailed information on using these tools.

There are four analysis types available in MotorXP-AFM: **Magnetostatic Finite Element Analysis**, **Steady State D-Q Analysis**, **Dynamic D-Q Analysis** and **Dynamic Finite Element Analysis**. To choose the analysis type, use the corresponding button under the toolbar. Table 3.1 presents a summary of MotorXP-AFM analysis types with a short description of each analysis type. Refer to the corresponding chapter of this manual to learn more about each analysis type.

After installation of MotorXP-AFM Standalone version, two shortcuts appear on the desktop for MotorXP-AFM and for MotorXP-AFM Design Studio:



For MotorXP-AFM MATLAB version the MotorXP-AFM Design Studio shortcut can be found in the folder with the application files.

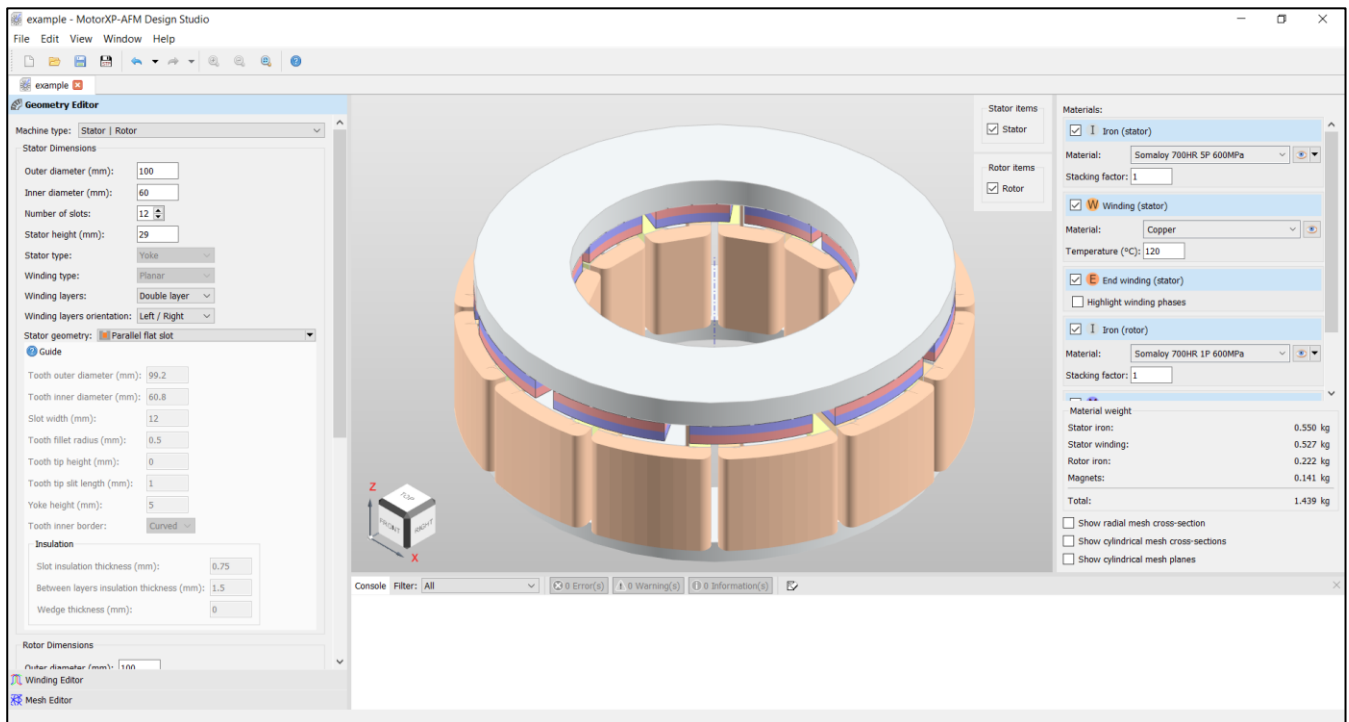



Figure 3.2. MotorXP-AFM Design Studio.

MotorXP-AFM Design Studio is a tool used to input geometry, material, winding and mesh parameters (see chapter 4). There are two versions of MotorXP-AFM Design Studio: as an integral part of MotorXP-AFM (accessible from the MotorXP-AFM toolbar and the **Desktop** menu), and as a standalone application. MotorXP-AFM Design Studio when used as a standalone application is shown in Figure 3.2. Figures 4.1, 4.13 and 4.15 demonstrate MotorXP-AFM Design Studio when used as a part of MotorXP-AFM. For example, it is more convenient to use the standalone version of MotorXP-AFM Design Studio when working with geometry scripts. You can also use the standalone version of MotorXP-AFM Design Studio to create a prototype of the machine and then open it in MotorXP-AFM for analysis. MotorXP-AFM Design Studio standalone and MotorXP-AFM use the same format of project files and fully compatible with each other – files created using a standalone version of MotorXP-AFM Design Studio can be open in MotorXP-AFM and vice versa.

3.4. Working with project files.


All machine parameters, simulation settings, finite element mesh and simulation results are stored in a *project file*. The project file has .mxa extension.

Standard file commands of creating, opening and saving project files are available from the **File** menu or using corresponding toolbar buttons.

Machine parameters, materials and finite element mesh presented in the **Geometry Editor**, **Winding Editor** and **Mesh Editor** tabs of MotorXP-AFM Design Studio should be defined before any analysis is started and cannot be changed afterwards in order not to confuse simulation results for different design prototypes. Once you started any analysis the project file gets locked (**Geometry Editor**, **Winding Editor** and **Mesh Editor** tabs of MotorXP-AFM Design Studio become not available for editing). If you want to change any parameter in **Geometry Editor**, **Winding Editor** and **Mesh Editor** tabs of MotorXP-AFM Design Studio after the project file got locked, use the **Unlock** toolbar button  (see Figure 3.1) to unlock the project file. Be aware that all simulation data of all analysis types will be deleted.

Open As New (Blank) Project item of the **File** menu allows you to create a new project based on another project. All machine parameters, materials, finite element mesh and analysis setting will be copied in the new project file while the analysis results will not be included. This is a convenient way to create several design prototypes (each design prototype corresponding to a separate project file) changing one or several parameters to find an optimal design.

Export results to MAT-file item of the **File** menu allows you to create a MATLAB data file with all the machine parameters and simulation results for further processing using MATLAB.

There is also an option to export the 3D motor geometry into STEP file using the button  of the MotorXP-AFM Design Studio toolbar (see Figure 3.2). Note that the geometry is exported exactly as it is shown on the screen, i.e., if some part of the geometry is chosen not to be shown it will not be included into the resulting STEP file.

3.4.1. Protected project files.

In some cases, there may be a need to hide the design details of the project, i.e. geometry, materials and winding configuration while **keeping the possibility to run all the analyses and view all the results**. It can be a situation when you are presenting the project to your customer for evaluation but do not want to disclose all the details about the design at this stage. In this case the **Save As Protected Project** option of the **File** menu can be used. The new project file with the extension .mxap will be created while all the information about the geometry, materials and winding configuration will be excluded from the file. When the protected file is open in MotorXP-AFM Design Studio, it will look like shown in Figure 3.3.

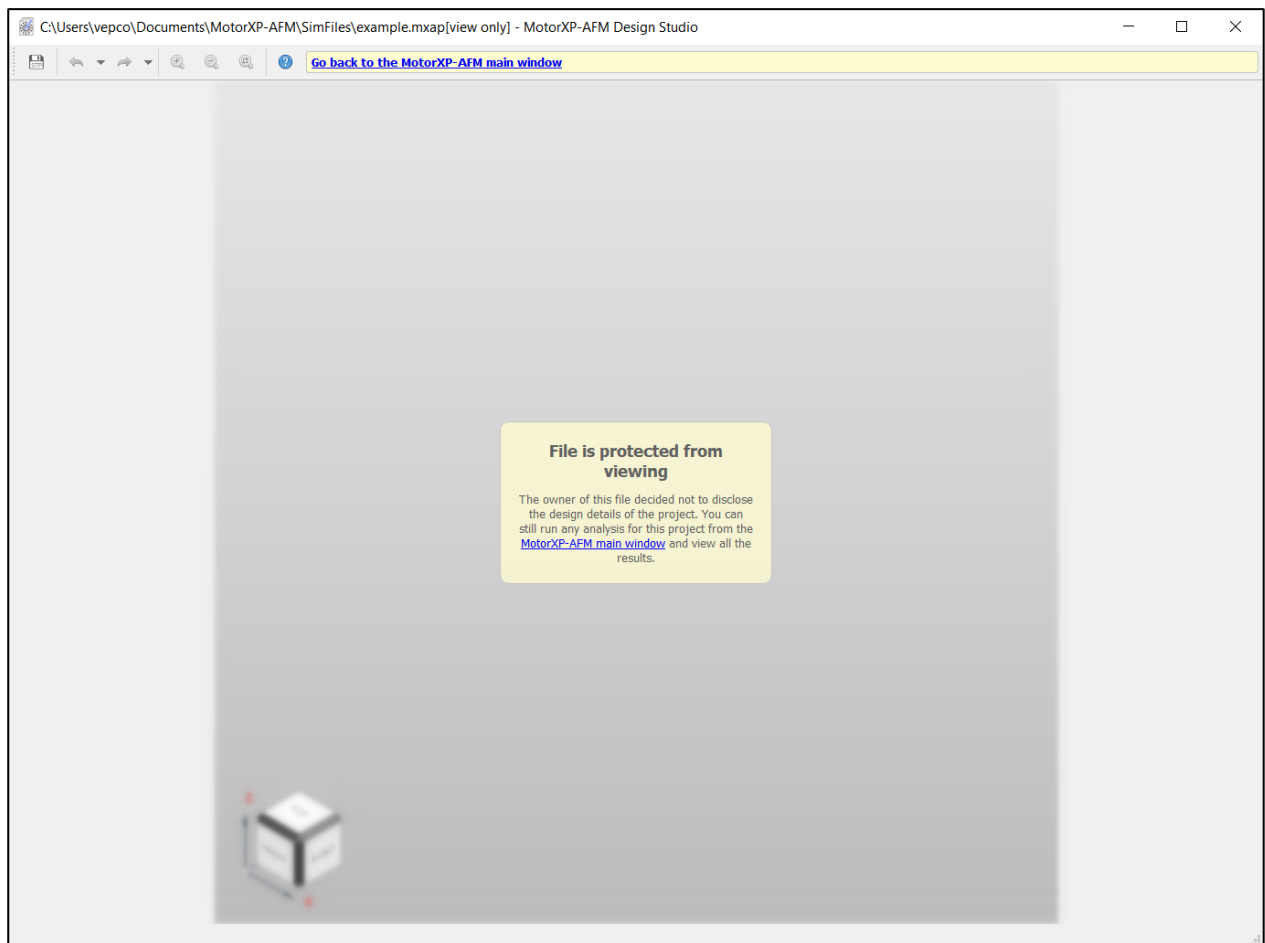


Figure 3.3. Protected project view in MotorXP-AFM Design Studio.

3.5. Licensing options.

There are several license types:

- FREE Edition
- Pro-Full
- Pro-FEAonly

By default, MotorXP-AFM is provided free of charge under the **FREE Edition** license – **MotorXP-AFM (FREE Edition)**. MotorXP-AFM (FREE Edition) can be upgraded to **MotorXP-AFM (Pro)** by purchasing a **License**. The **Pro-Full** license includes all four analysis types (Magnetostatic Finite Element Analysis, Dynamic Finite Element Analysis, Steady State D-Q Analysis and Dynamic D-Q Analysis) while the **Pro-FEAonly** license includes only Magnetostatic Finite Element Analysis and Dynamic Finite Element Analysis.

MotorXP-AFM (FREE Edition) has all the functionality of MotorXP-AFM (Pro) except that the FREE Edition of the program does not activate project files. It means that all the project files created or modified in MotorXP-AFM (FREE Edition) will not be available for any analysis and simulation using MotorXP-AFM (FREE Edition). After the project file has been activated in MotorXP-AFM (Pro), it also becomes available for analysis using MotorXP-AFM (FREE Edition). The project file can be activated in MotorXP-AFM (Pro) by running any analysis or by using the **Activate Project** option of the **File** menu. The sign to the right of the project file name (as shown in Figure 3.5) indicates whether the project has been activated or not, i.e. whether the project is available for analysis using MotorXP-AFM (FREE Edition) or not.

If you need to send your project file to someone else who is using MotorXP-AFM (FREE Edition) for analysis, make sure that there is the green “Activated” sign to the right of the project file name as shown in Figure 3.5. It enables the Magnetostatic and Dynamic FE analysis in MotorXP-AFM (FREE Edition) for this project file. To enable D-Q Analysis the D-Q model should be built (see section 6.1).

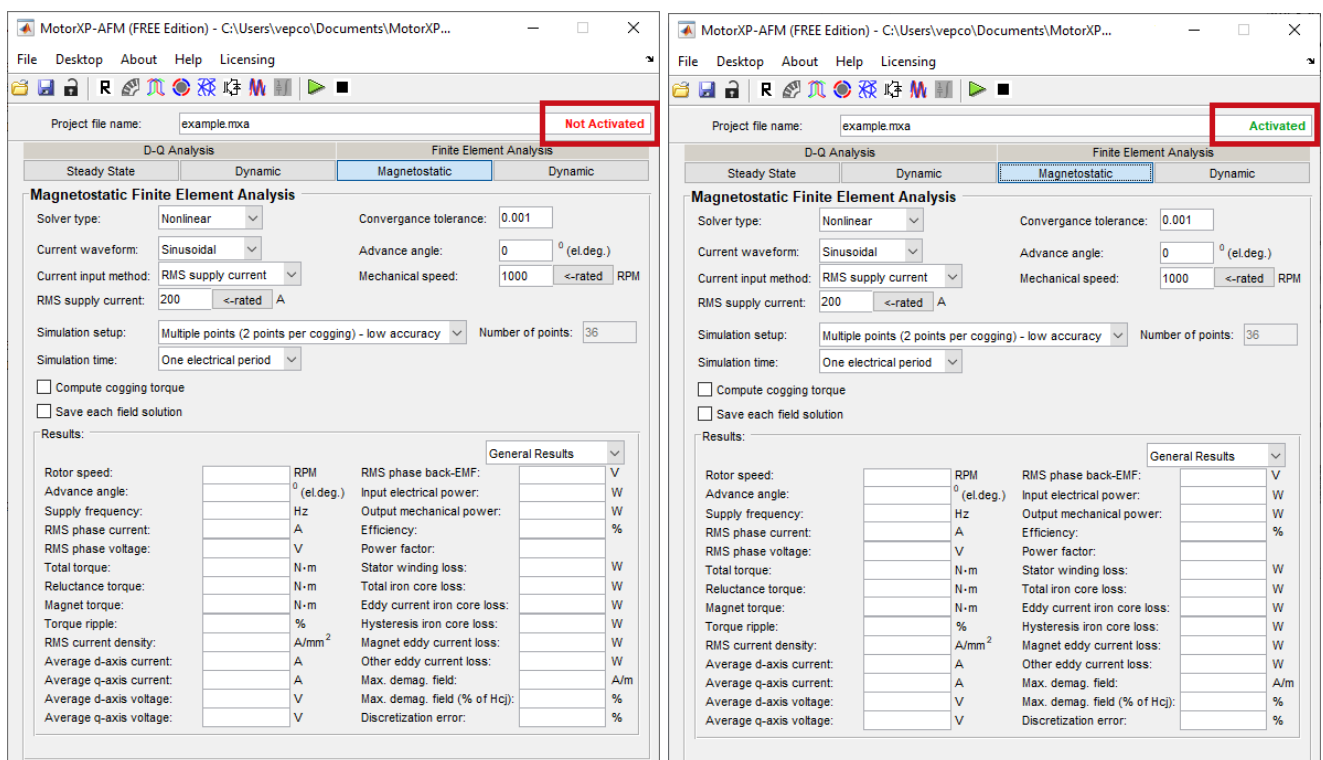
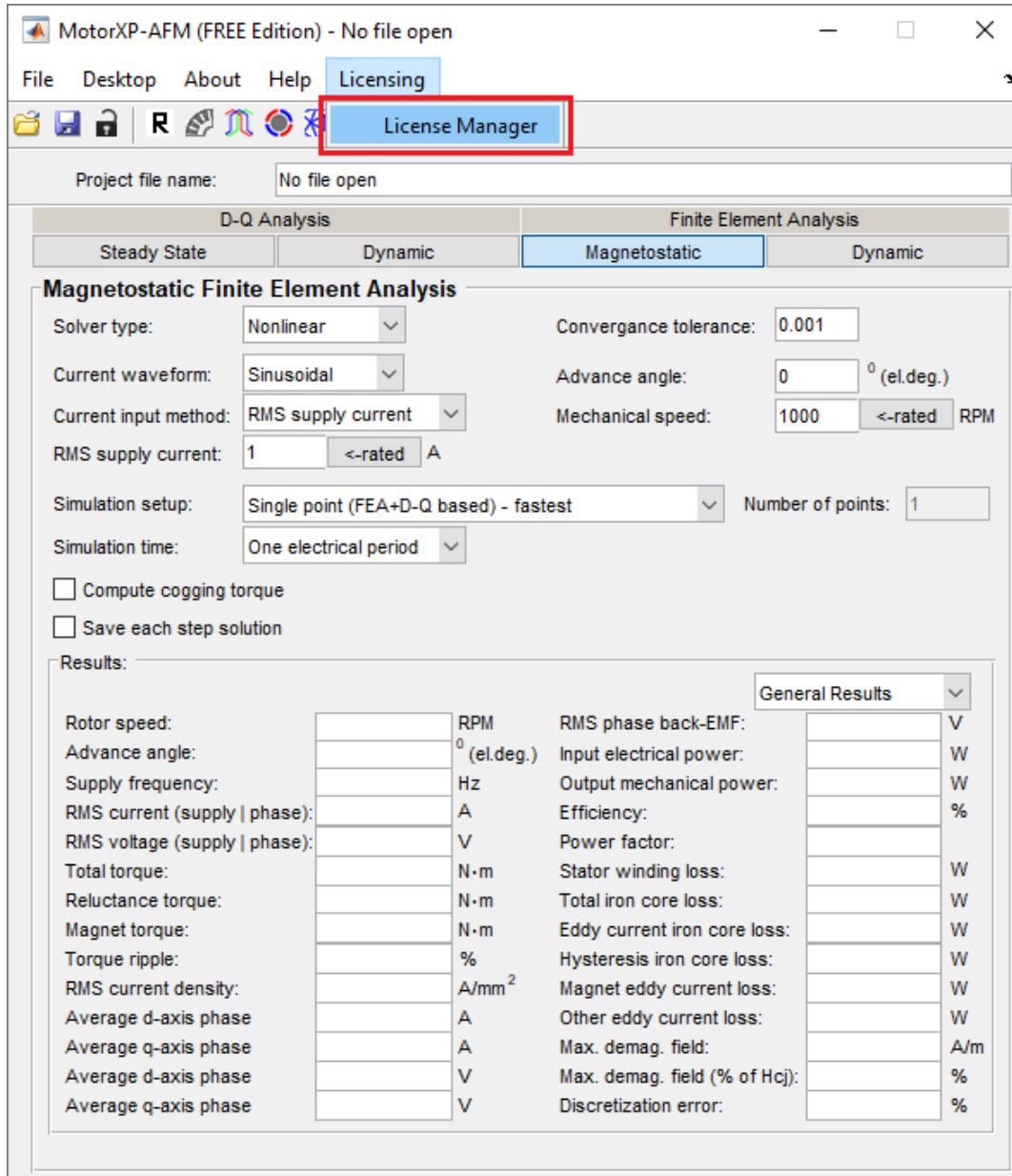


Figure 3.5. “Not Activated” and “Activated” project files open in MotorXP-AFM (FREE Edition).

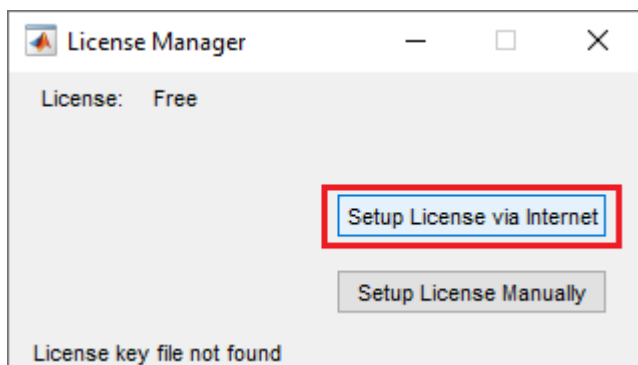
3.6. Activating the license.

In most cases the MotorXP-AFM license is linked to the MotorXP account and activated via Internet following the steps given below.

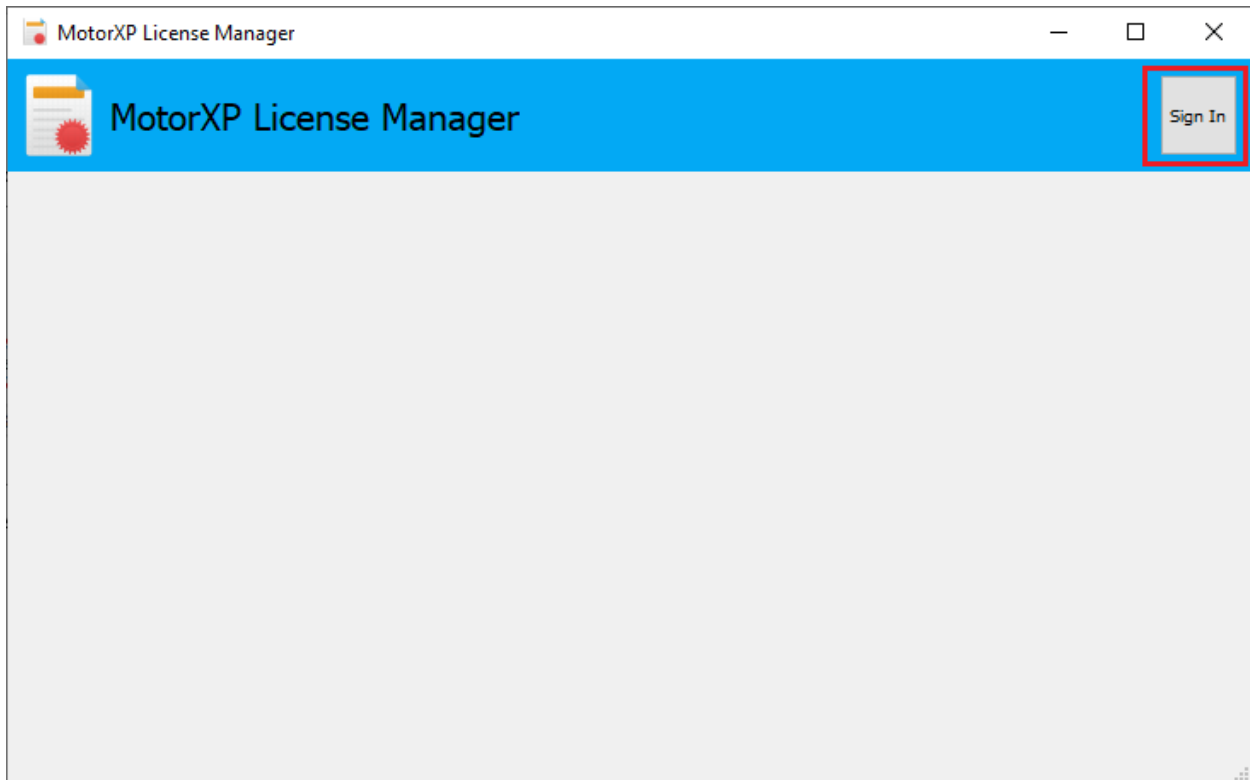
1. Open License Manager



2. In the License Manager window click the *Setup License via Internet* button



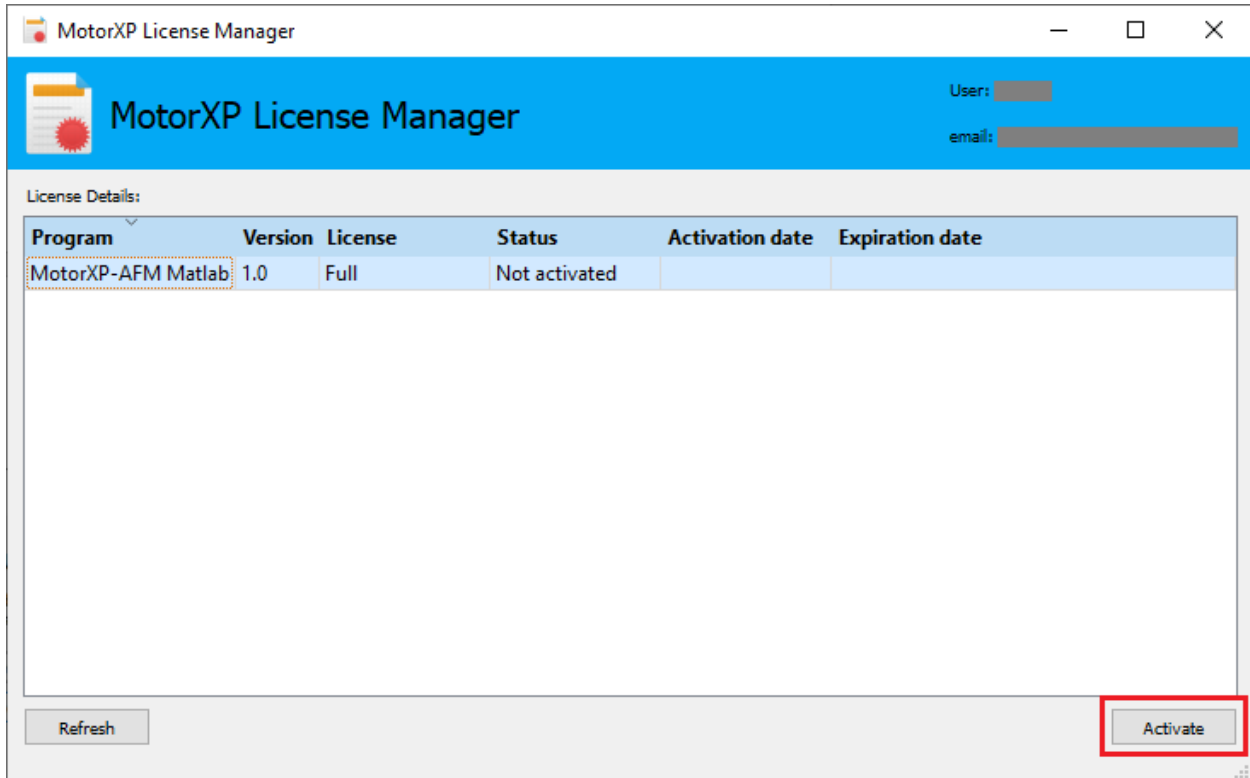
3. In the next window click the **Sign In** button:



4. Enter the MotorXP account user name and password then click the **OK** button:

A screenshot of a "Sing In" dialog box. It has a title bar with "Sing In" and a close button. The main content area has a question mark icon and the text "Please enter your Login from motorxp.com". Below this, it says "Please use the User name and Password from your MotorXP account. If you forgot the Login, please click on 'Forgot Password'." There are two input fields: "User name:" with a placeholder "<input user name>" and "Password:" with a placeholder "<input password>". Below the password field is a checkbox labeled "Remember me" which is checked. At the bottom, there are three buttons: "Forgot Password", "OK", and "Cancel".

5. Select the license you want to activate and click the **Activate** button



Close **License Manager**.

After successful activation the licensing status in the **License Manager** window will change from **Free** to either **Pro (full)** or **Pro (FEA only)**.

Table 3.1. Summary of MotorXP-AFM analysis types.

	Analysis method	Simulation models used for the analysis	What is included into analysis	Application	Results and visualization	Comments
Magnetostatic Analysis	Single point magnetostatic analysis	One step magnetostatic FE model with single rotor position and D-Q representation of the machine.	General electromagnetic performance; Demagnetization.	Simulation of steady state performance for single operating point with ideal sinusoidal current waveform.	Time-averaged parameters; Air gap distribution plots; Cross-section field plots.	Determines motor performance from single run of magnetostatic FE simulation for one rotor position and D-Q representation of the machine. Medium accuracy since rotor rotation is not considered, medium computation speed.
	Multiple points (time-stepping) magnetostatic analysis with sinusoidal and trapezoidal current waveform	Time-stepping magnetostatic FE model with multiple rotor positions.	General electromagnetic performance; Cogging torque and torque ripple; Back-EMF harmonics; Iron losses (including higher harmonics and minor hysteresis loops); Eddy current magnet and sleeve losses; Demagnetization.	Simulation of steady state performance for single operating point with ideal sinusoidal or trapezoidal current waveform.	Time-averaged parameters; Time plots; Air gap distribution plots; Cross-section field plots; Animation.	Current driven simulation (current waveform is predefined in advance). Rotor rotation is considered by running simulations for several rotor positions. High accuracy and low computation speed.
Steady State D-Q Analysis	Steady state D-Q analysis with sinusoidal supply	Steady state D-Q model. Simplified inverter losses model (see section 2.8.2).	General electromagnetic performance; Sinusoidal back-EMF assumed; Fundamental frequency iron losses; Field weakening strategy; Mechanical losses; Inverter losses (only space-vector PWM).	Simulation of steady state performance for the range of operating conditions with ideal sinusoidal current waveform.	Motor characteristic curves as a function of speed, current or advance angle; Efficiency map and other performance maps.	Requires preliminary parameterization of D-Q model *. Medium accuracy and high computation speed.
	Steady state D-Q analysis with “FEA based” model type	One step magnetostatic FE model with single rotor position and D-Q representation of the machine.	General electromagnetic performance; Field weakening strategy.		Motor characteristic curves as a function of speed, current or advance angle.	Does not requires preliminary parameterization of D-Q model *. Medium accuracy without iron losses taken into account and medium computation speed.
	Steady state D-Q analysis with PWM supply	Dynamic D-Q model.	General electromagnetic performance; Sinusoidal back-EMF assumed; Fundamental frequency iron losses; Field weakening strategy; PWM switching; Inverter voltage drop.	Simulation of steady state performance for the range of operating conditions with PWM sine wave voltage.	Motor characteristic curves as a function of speed, current or advance angle.	Requires preliminary parameterization of D-Q model *. Medium accuracy with influence of PWM switching taken into account. Medium computation speed.
Dynamic D-Q Analysis	Dynamic D-Q analysis with linear, linearized and nonlinear solvers	Dynamic D-Q model. Inverter losses model (see section 2.8.1)	General electromagnetic performance; Sinusoidal back-EMF assumed; Fundamental frequency iron losses; PWM switching; Inverter voltage drop; Inverter losses.	Simulation of transient and steady state performance for single operating point with PWM sine wave or six-step voltage.	Time-averaged parameters; Time plots	Requires preliminary parameterization of D-Q model *. Voltage driven simulation – current is determined by applied DC voltage and switching sequence. Medium accuracy and high computational speed.
Dynamic FE Analysis	Dynamic FE analysis	Transient finite element model coupled with electrical circuit; Dynamic D-Q model to calculate initial conditions and speed up simulation.	General electromagnetic performance; Induced eddy currents; Cogging torque and torque ripple; Back-EMF harmonics; Iron losses (including higher harmonics and minor hysteresis loops); Demagnetization; PWM switching; User defined electrical circuits; User defined control algorithms.	Simulation of transient and steady state performance for single operating point with PWM sine wave or six-step voltage. Simulation of specific operating conditions which cannot be simulated with other analysis methods.	Time-averaged parameters; Time plots; Air gap distribution plots; Cross-section field plots; Animation.	Very high accuracy but very low computational speed since the simulation should go through the initial transient until the steady state is reached.

* Preliminary parameterization of D-Q model is a process of extraction of D-Q model parameters using the finite element model of the machine. It should be done only once and takes from several minutes to several hours. Refer to section 6.1 for more information.

4. CREATING DESIGN PROTOTYPES

Design prototypes of an electric machine are created using a tool called MotorXP-AFM Design Studio which is a part of MotorXP-AFM. MotorXP-AFM Design Studio is accessible from the MotorXP-AFM main window toolbar shown in Figure 3.1. MotorXP-AFM Design Studio is also available as a standalone application as described in section 3.3. MotorXP-AFM Design Studio is divided into three tabs to input machine's geometry and materials (**Geometry Editor**), winding parameters (**Winding Editor**) and mesh parameters (**Mesh Editor**). Four toolbar buttons of the MotorXP-AFM main window can be used to open the corresponding tab of MotorXP-AFM Design Studio or the required tab can be accessed from the navigation panel. The view of the MotorXP-AFM Design Studio window when the **Geometry Editor** tab is open is shown in Figure 4.1.

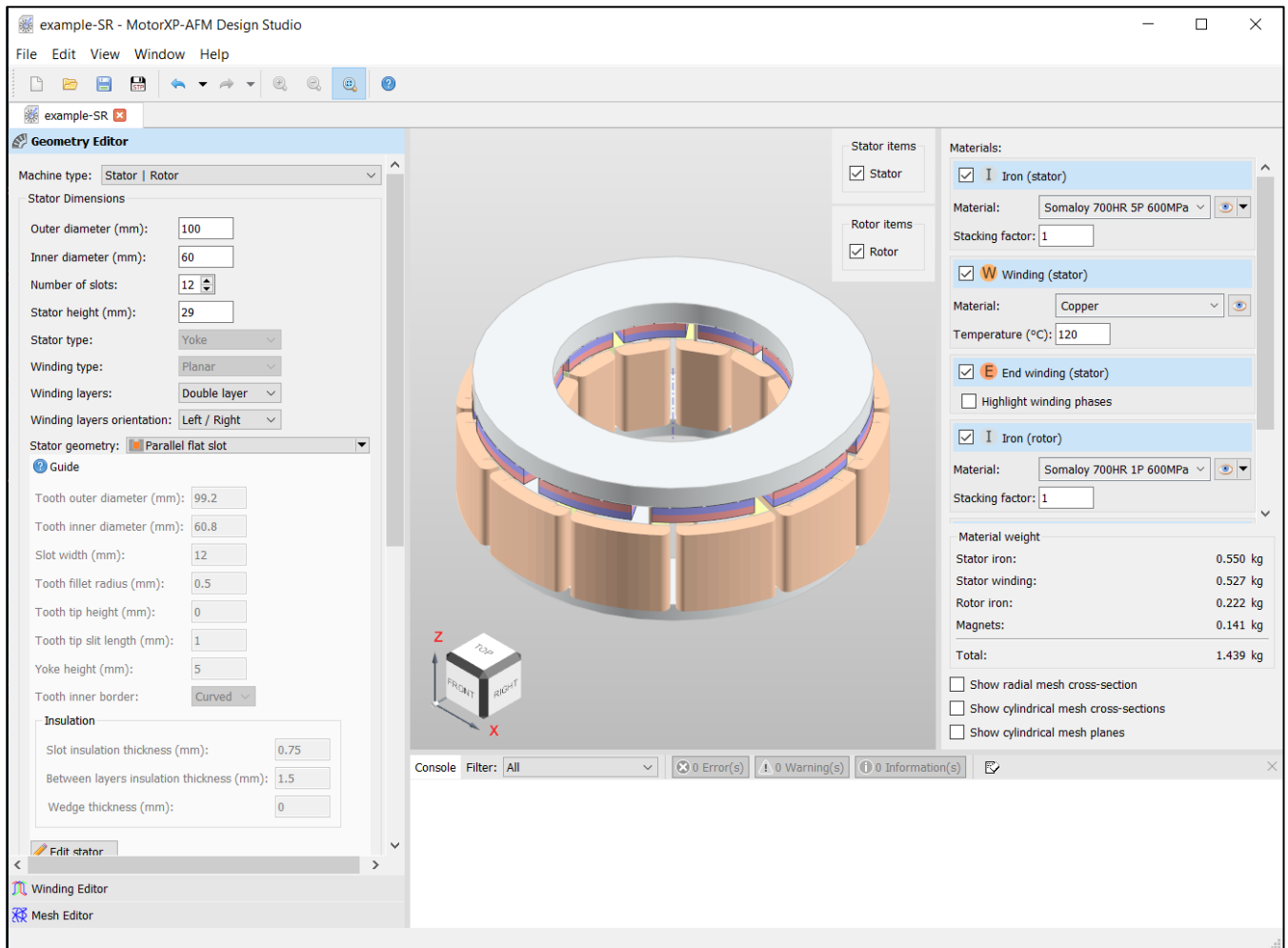


Figure 4.1. Geometry Editor of MotorXP-AFM Design Studio.

4.1. Geometry Editor.

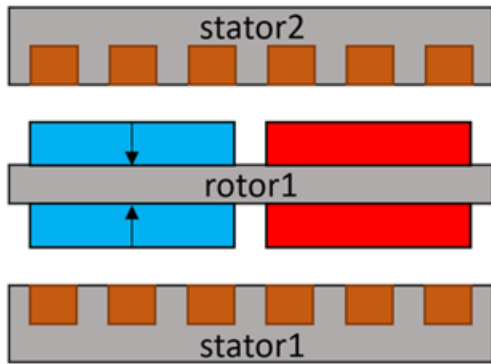
Geometry Editor tab of MotorXP-AFM Design Studio allows you to set up dimensions of the machine and specify materials and their characteristics. **Geometry Editor** tab of MotorXP-AFM Design Studio is shown in Figure 4.1.

Machine type specifies the axial machine geometry with different stator and rotor arrangements. There are three different arrangements available from the **Machine type** pop-up menu:

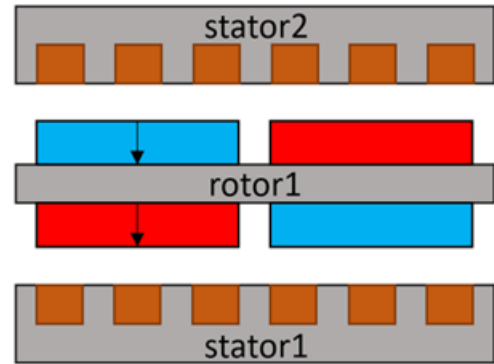
- Stator | Rotor
- Stator | Rotor | Stator
- Rotor | Stator | Rotor

There are designated panels for entering different stator and rotor parameters such as outer and inner diameters, number of stator slots and machine pole pairs, rotor(s) and stator(s) heights, stator winding type and number of winding layers. Depending on the machine type, there are yoke/yokeless options for stator/rotor and different magnet pole arrangements.

Pole arrangement specifies the orientation of the magnets of the same pole relative to air gaps in case if there are two air gaps. There are two possible pole arrangements for two air gaps: *N-N* and *N-S*, where “N” indicates that the north pole of the corresponding magnet is facing the air gap and “S” indicates that the south pole of the corresponding magnet is facing the air gap as shown below:



Pole arrangement: N-N



Pole arrangement: N-S

Stator angular displacement allows to rotate the position of one stator relative to another by specified angle in order to reduce the torque ripple.

Rotor angular displacement allows to rotate the position of one rotor relative to another by specified angle in order to reduce the torque ripple.

Stator | Rotor

This is the most basic type of an axial flux machine since it contains only one stator and one rotor.

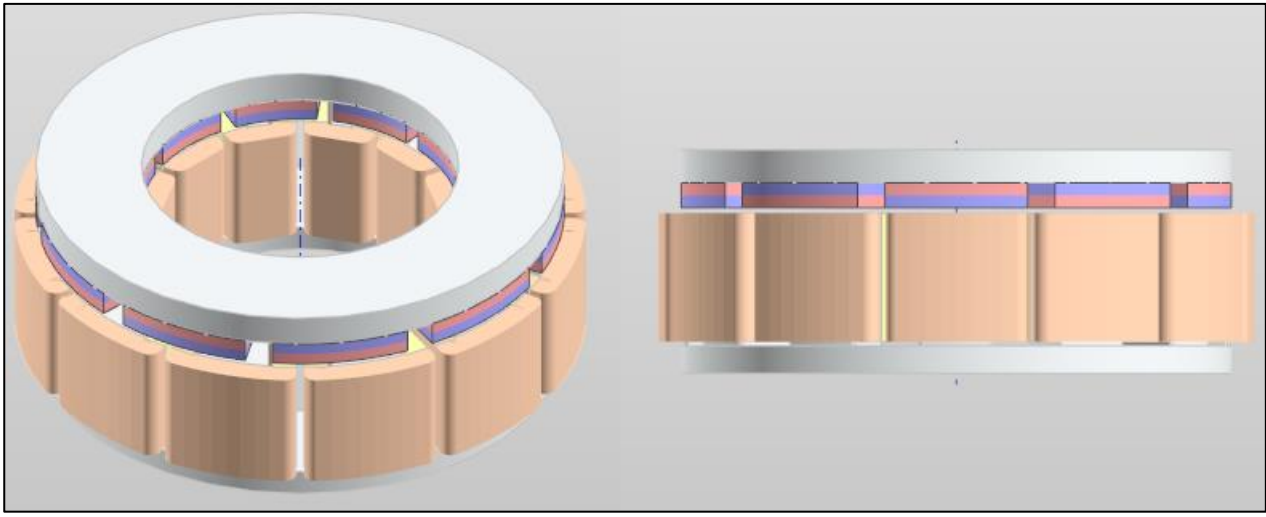
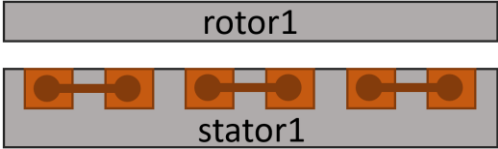
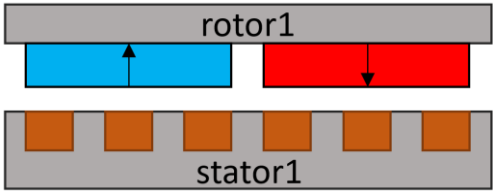


Figure 4.2. An example of the Stator | Rotor arrangement in Geometry Editor of MotorXP-AFM Design Studio.

Stator option:	Description:
	<i>Yoke stator</i>
Rotor option:	Description:
	<i>Yoke rotor</i>

Stator | Rotor | Stator

This machine type contains a rotor between two stators. Rotor of the machine can either be a *Yoke* type or *Yokeless*.

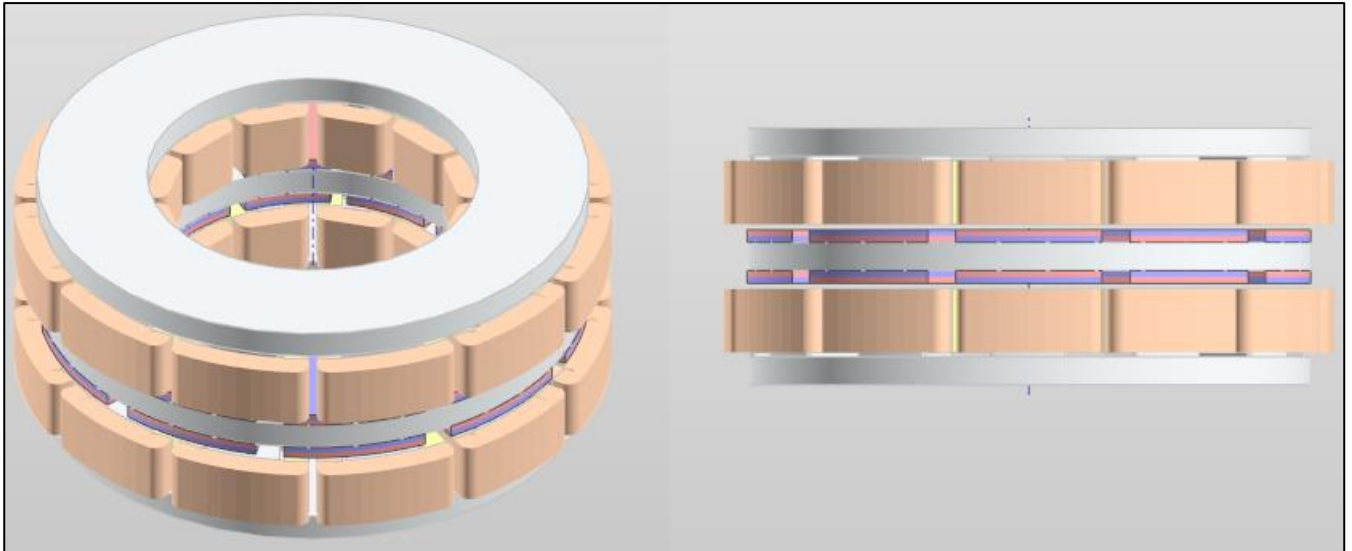


Figure 4.3. An example of Stator | Rotor | Stator (Yoke rotor) arrangement in Geometry Editor of MotorXP-AFM Design Studio.

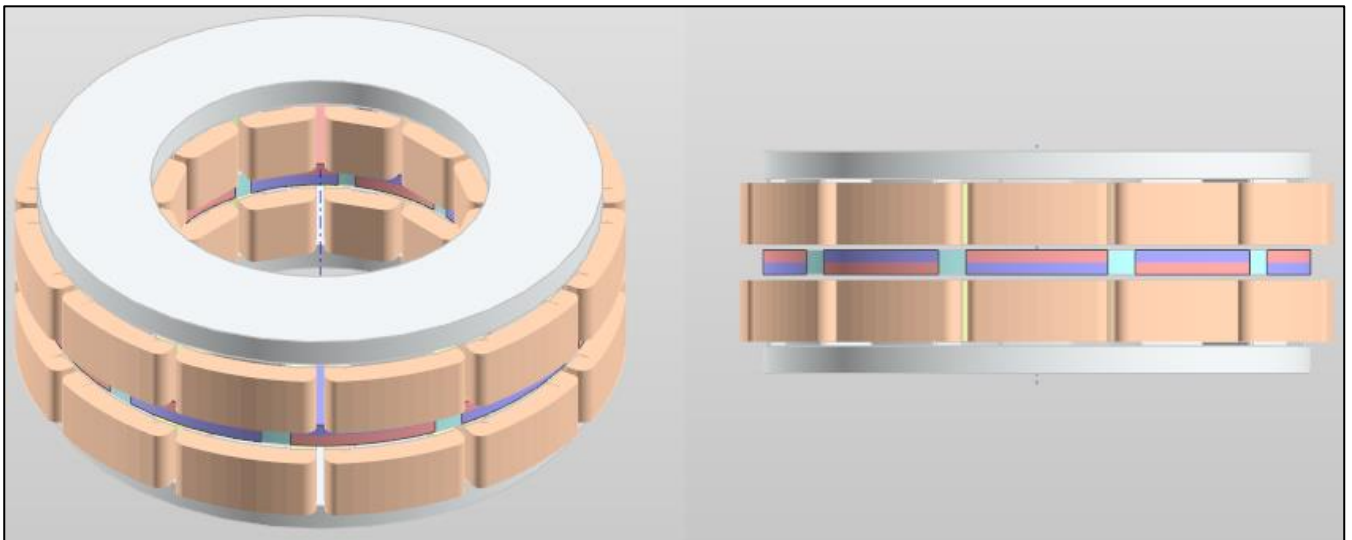
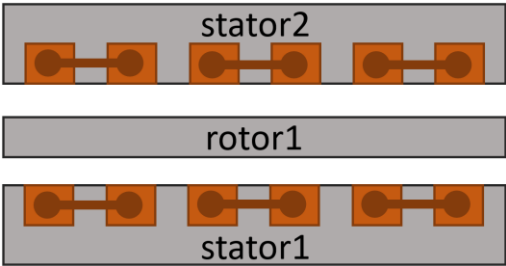
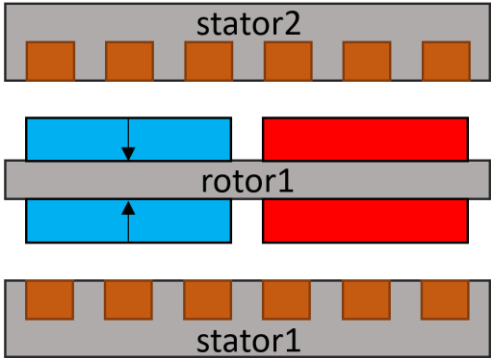
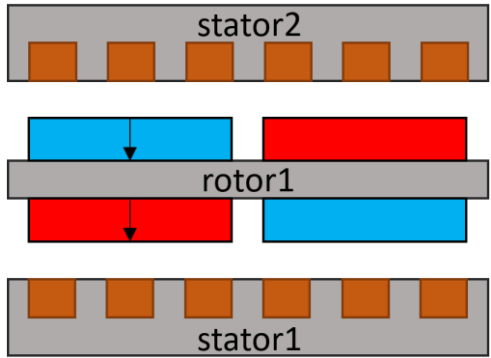
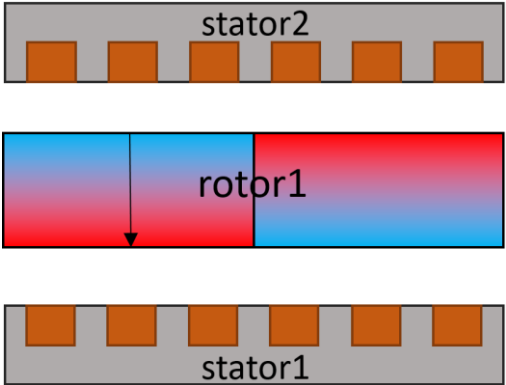


Figure 4.4. An example of Stator | Rotor | Stator (Yokeless rotor) arrangement in Geometry Editor of MotorXP-AFM Design Studio.

Stator option:	Description:
	<p>Yoke stator Pole arrangements:</p> <ul style="list-style-type: none">• N-N (only for Yoke rotor)• N-S

Rotor options:	Description:
	<p>Yoke rotor Pole arrangement:</p> <ul style="list-style-type: none">• N-N
	<p>Yoke rotor Pole arrangement:</p> <ul style="list-style-type: none">• N-S
	<p>Yokeless rotor Pole arrangement:</p> <ul style="list-style-type: none">• N-S

Rotor | Stator | Rotor

This machine type contains a stator between two rotors. Stator of the machine can either be a *Yoke* type or *Yokeless*. The winding for this machine type can either be *Planar* or *Toroidal*.

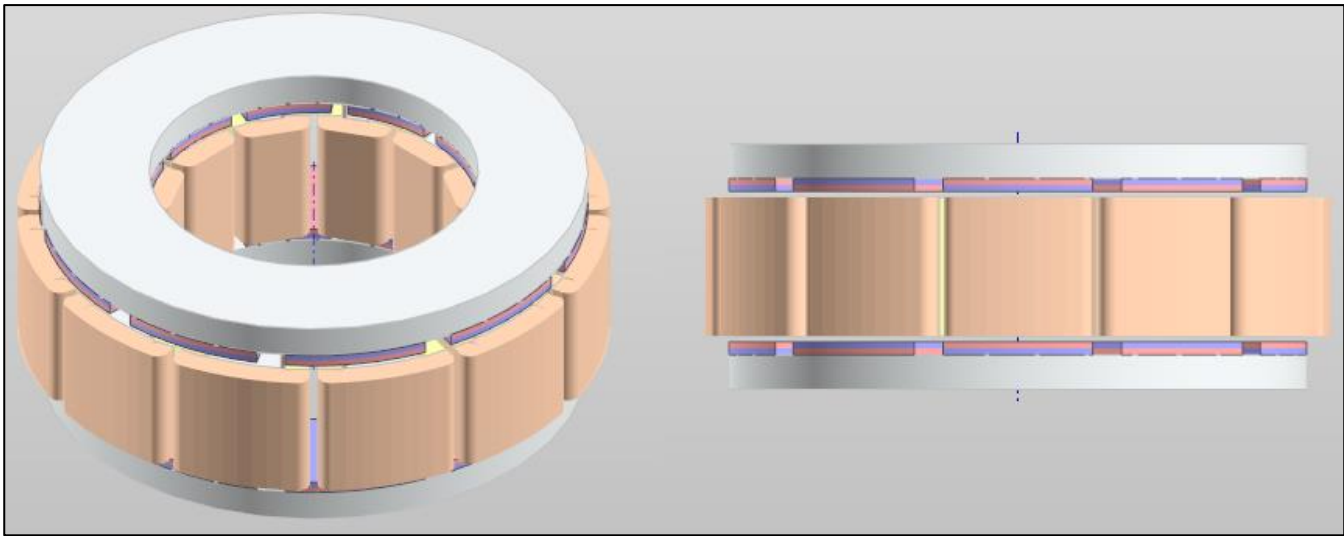
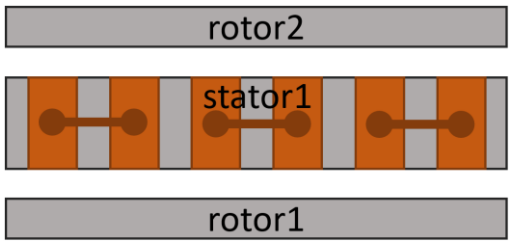
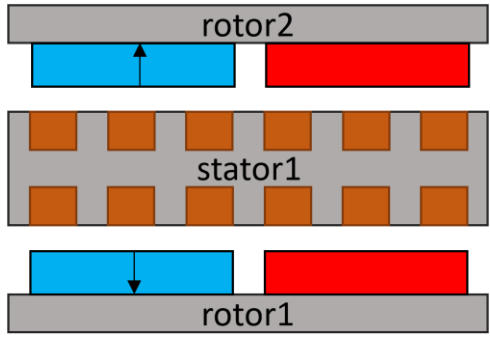
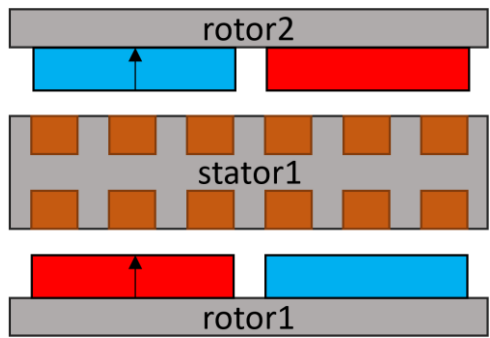


Figure 4.5. An example of Rotor | Stator | Rotor arrangement in Geometry Editor of MotorXP-AFM Design Studio.

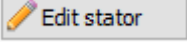
Stator options:	Description:
	Yoke stator Pole arrangements: <ul style="list-style-type: none">• N-N• N-S Planar winding (single or double layer)
	Yoke stator Pole arrangement: <ul style="list-style-type: none">• N-N Toroidal winding (single layer only)

	<p><i>Yokeless stator</i> Pole arrangement:</p> <ul style="list-style-type: none"> • N-S
Rotor options:	Description:
	<p><i>Yoke rotor</i> Pole arrangement:</p> <ul style="list-style-type: none"> • N-N
	<p><i>Yoke rotor</i> Pole arrangement:</p> <ul style="list-style-type: none"> • N-S

View angle of the machine can be adjusted by holding mouse left button and rotating on the geometry view. There is also a 3D coordinate view at the bottom-left corner. The view angle can also be adjusted by choosing the sides, edges and corners of the navigation cube. Selected stator/rotor items can be shown/hidden by clicking corresponding checkboxes at the top-right section of the machine geometry view.

Materials used in different parts of the machine can be seen at the right side of the **Geometry Editor** window. Certain material types can be shown/hidden by clicking the corresponding checkboxes of the materials panel. For more information and details about the materials, refer the section 4.2. of this manual. There are also additional options for viewing the geometry right under the materials panel. Radial mesh cross-section, cylindrical mesh cross-sections and cylindrical mesh planes can be viewed by selecting the corresponding checkboxes.

4.1.1. Stator editing mode of Geometry Editor.

Clicking the button  opens the stator geometry in editing mode as shown in Figure 4.6. **Outer diameter, Inner diameter, Number of slots, Stator height, Stator type (Yoke or Yokeless), Winding type (Planar or Toroidal), Winding layers (Single layer or Double layer), Winding layers orientation (Upper / Lower or Left / Right)** and stator angular displacement parameters are the same as shown on the default **Geometry Editor** view (Figure 4.1). Note that some options may be unavailable for certain types of machines.

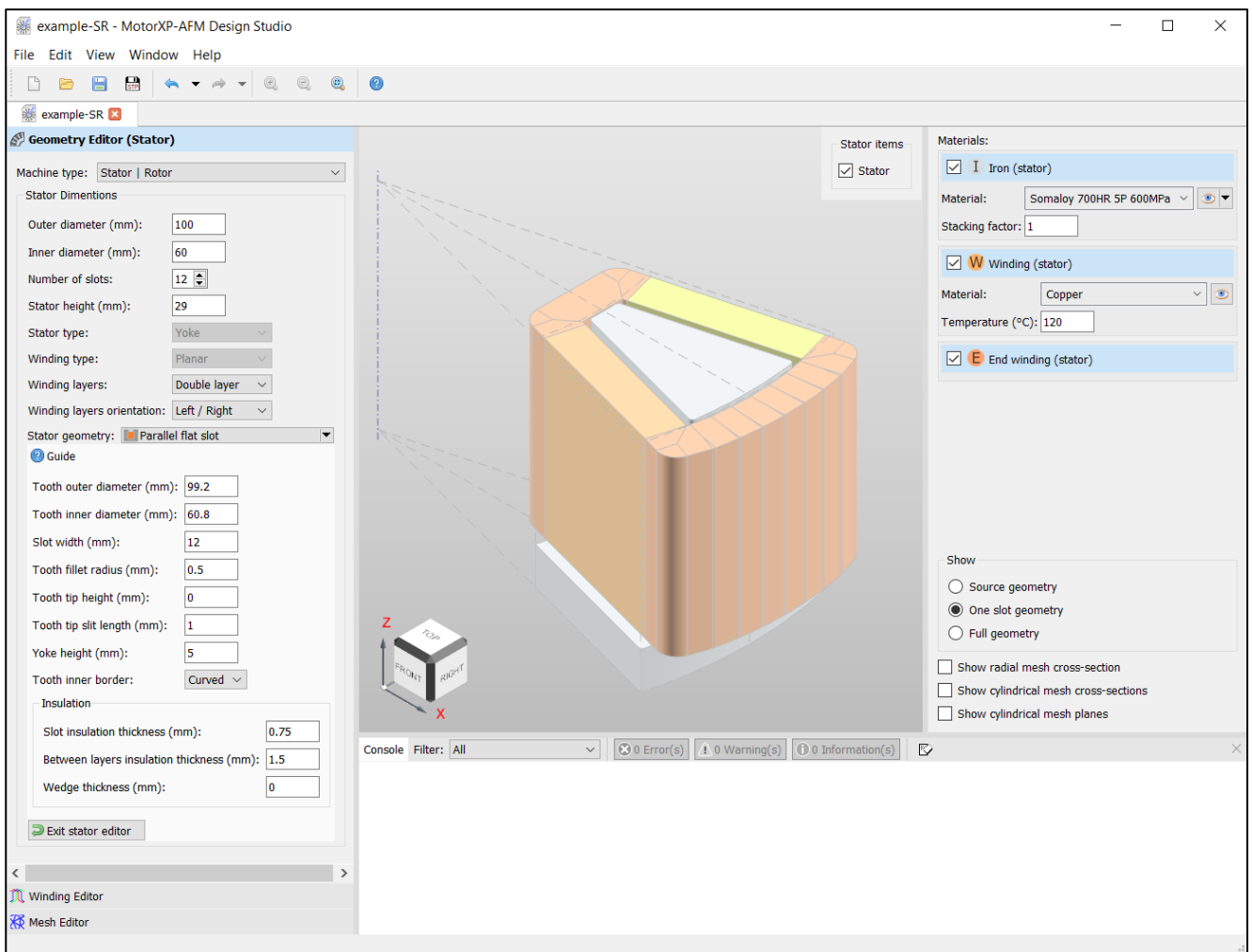


Figure 4.6. Geometry Editor of MotorXP-AFM Design Studio, stator geometry editing mode.

Winding layers pop-up menu allows you to choose either *Single layer* or *Double layer* stator winding. If the *Double layer* stator winding is chosen, the **Winding layers orientation** field appears.

Winding layers orientation field (*Upper / Lower* or *Left / Right*) specifies either the stator slot will be divided horizontally or vertically into two parts with equal areas. *Left / Right* orientation is appropriate for concentrated windings, while *Upper / Lower* orientation is appropriate for distributed windings.

There are two standard stator geometry templates available which can be chosen from the **Stator geometry** pop-up menu: *Parallel round slot* and *Parallel flat slot*. Depending on the stator geometry selection, parameters that define stator slot shape change according to the slot type (Figure 4.7.)

Stator geometry: Parallel flat slot

? Guide

Tooth outer diameter (mm): 95

Tooth inner diameter (mm): 65

Slot width (mm): 12

Tooth fillet radius (mm): 2

Tooth tip height (mm): 3

Tooth tip slit length (mm): 3

Yoke height (mm): 5

Tooth inner border: Curved

Insulation

Slot insulation thickness (mm): 0.75

Between layers insulation thickness (mm): 1.5

Wedge thickness (mm): 0.75

Exit stator editor

Stator geometry: Parallel round slot

? Guide

Slot width (mm): 12

Slot opening depth (mm): 1

Slot opening width (mm): 2

Tooth tip angle (°): 15

Slot bottom corner type: General

Slot bottom corner radius (mm): 3

Slot top corner radius (mm): 3

Yoke height (mm): 5

Insulation

Slot insulation thickness (mm): 0.5

Between layers insulation thickness (mm): 1

Exit stator editor

Figure 4.7. Parameters for different stator geometries in editing mode

Use button **Guide** right under the **Stator geometry** pop-up menu to see the detailed explanation for all the dimensions corresponding to the selected geometry template.

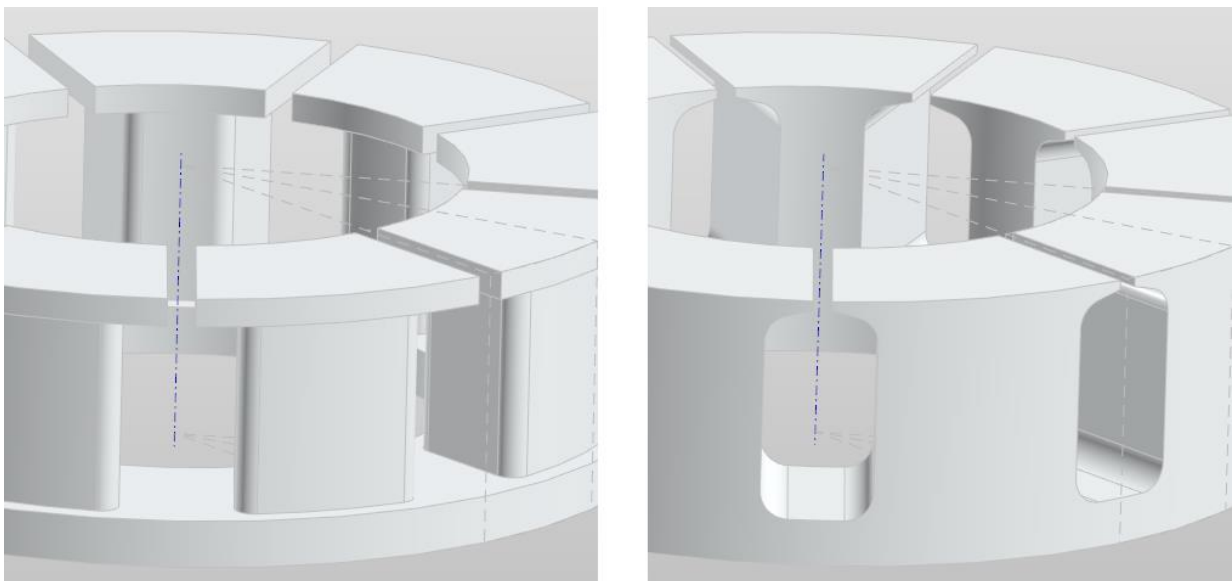
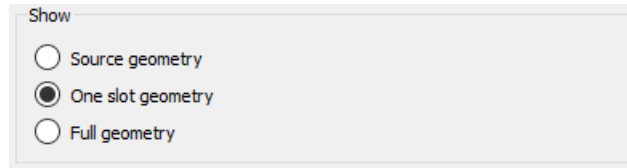


Figure 4.8. Examples of stator cores for *Parallel flat slot* (left) and *Parallel round slot* (right) stator geometries.

Slot shape can be optimized in MotorXP-AFM by changing the stator parameters shown in Figure 4.8. Tooth shape optimization can lead to reduction of torque ripple and reduction of noise of the machine. **Slot width** parameter should be carefully selected as it highly influences the magnetic saturation at the stator teeth.

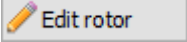
Insulation section contains the **Slot insulation thickness**; and if the **Winding layers** pop-up menu is set up to **Double layer**, it also contains the **Between layers insulation thickness** parameter.

The stator geometry view in the editing mode can be changed on the right side of **Geometry Editor**:



The **Source geometry** option displays the stator geometry as it is returned by the stator geometry script; it can be useful while debugging the geometry script. The **One slot geometry** option displays the geometry only for one slot pitch and the **Full geometry** option displays the whole stator of the machine.

4.1.2. Rotor editing mode of Geometry Editor.

Clicking the button  opens the rotor geometry in editing mode as shown in Figure 4.9. **Outer diameter, Inner diameter, Number of pole pairs, Rotor height, Rotor type (Yoke or Yokeless), Pole arrangement** and rotor angular displacement parameters are the same as shown on the default **Geometry Editor** view (Figure 4.1). Note that some options may be unavailable for certain types of machines.

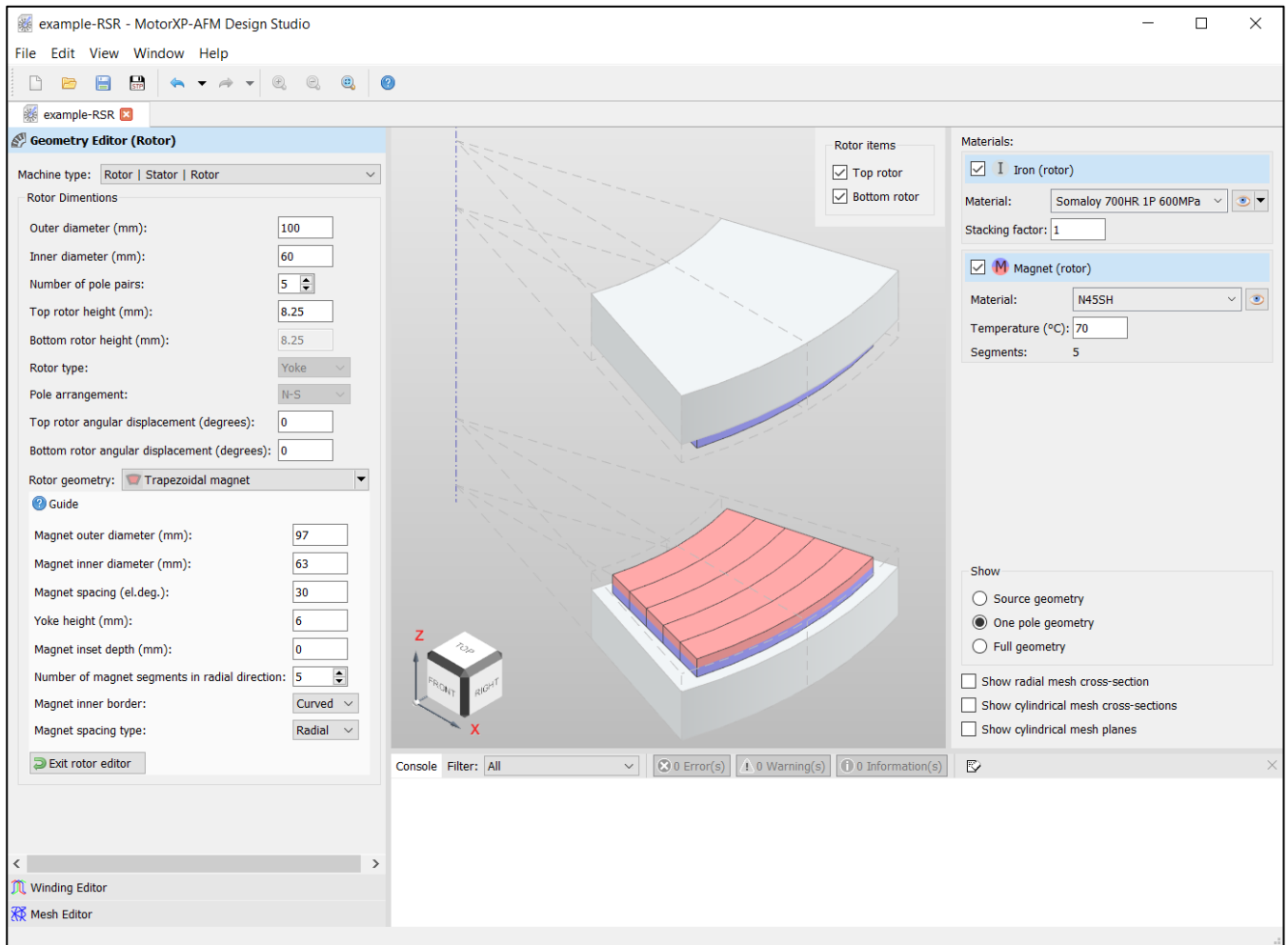


Figure 4.9. Geometry Editor of MotorXP-AFM Design Studio, rotor geometry editing mode.

Pole arrangement specifies the orientation of the magnets of the same pole relative to air gaps as described in section 4.1.

There are three standard rotor geometry templates available which can be chosen from the **Rotor geometry** pop-up menu: *Trapezoidal magnet*, *Rectangular magnet* and *Halbach array rotor*. Depending on the rotor geometry selection, parameters that define the rotor core and magnet dimensions change according to the rotor type (Figure 4.10.)

Figure 4.10. Parameters for different rotor geometries in editing mode

Use button **Guide** right under the **Rotor geometry** pop-up menu to see the detailed explanation for all the dimensions corresponding to the selected geometry template.

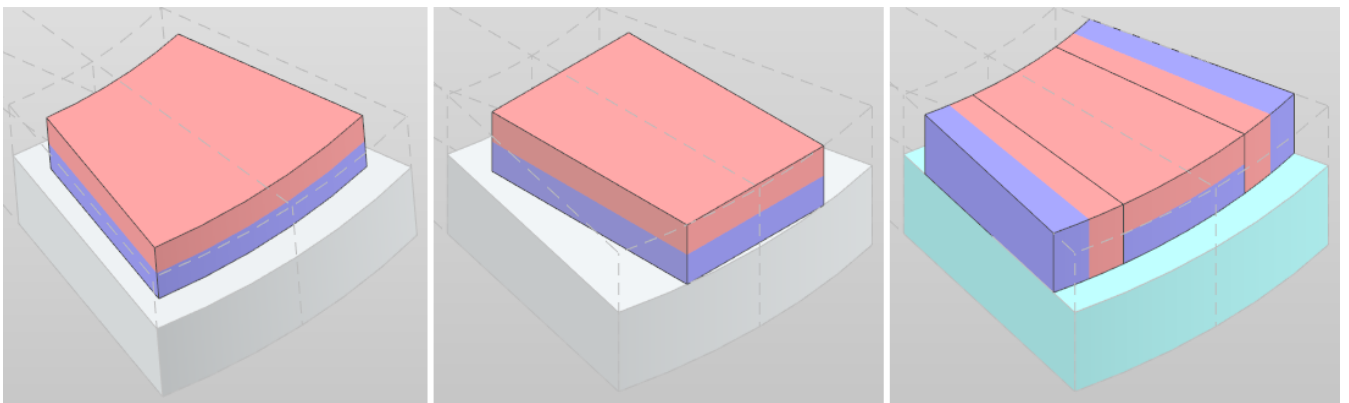
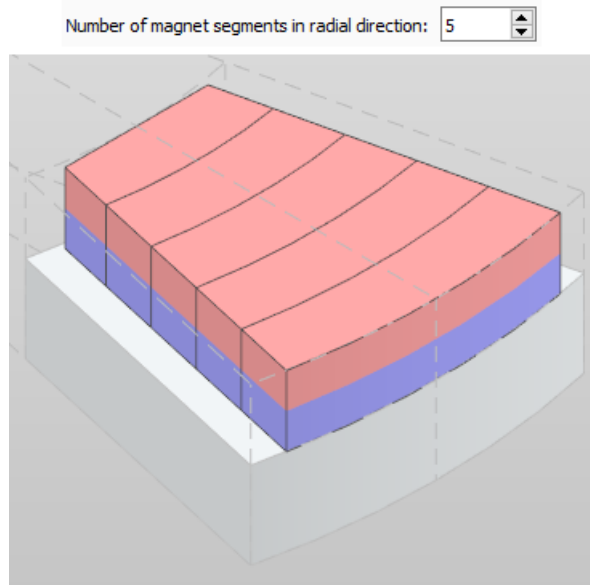


Figure 4.11. Examples of rotor views for *Trapezoidal magnet* (left), *Rectangular magnet* (middle) and *Halbach array rotor* (right) geometries.

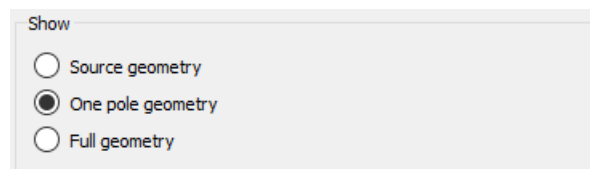
Magnet shape can be optimized in MotorXP-AFM by changing the rotor parameters shown in Figure 4.10. Magnet shape optimization can lead to better back-EMF waveform and reduction of torque ripple.

Yoke height and **Magnet inset depth** should be selected as there is no high saturation at the yoke of the rotor.

All standard rotor geometries also offer an option of segmenting the magnets in radial direction defined by the **Number of magnet segments in radial direction** parameter, i.e., dividing the magnet into several electrically isolated pieces forming concentric arcs to reduce eddy current losses in the magnet, as shown below for 5 magnet segments:



The rotor geometry view in the editing mode can be changed on the right side of **Geometry Editor**:



The ***Source geometry*** option displays the rotor geometry as it is returned by the rotor geometry script; it can be useful while debugging the geometry script. The ***One pole geometry*** option displays the geometry only for one pole pitch and the ***Full geometry*** option displays the whole rotor of the machine.

4.2. Assigning materials.

The panel for assigning materials is shown on the right when the **Geometry Editor** tab of MotorXP-AFM Design Studio is open (see Figure 4.12).

Materials:

☒ **I** Iron (stator)

Material: Somaloy 700HR 5P 600MPa

Stacking factor: 1

☒ **W** Winding (stator)

Material: Copper

Temperature (°C): 120

☒ **E** Endturn (stator)

☒ Highlight winding phases

☒ **I** Iron (rotor)

Material: Somaloy 700HR 1P 600MPa

Stacking factor: 1

☒ **M** Magnet (rotor)


Material: N45SH

Temperature (°C): 100

Segments: 1

Material weight	
Stator iron:	0.550 kg
Stator winding:	0.532 kg
Rotor iron:	0.222 kg
Magnets:	0.141 kg
Total:	1.445 kg

Figure 4.12. The panel for assigning materials in MotorXP-AFM Design Studio.

Different parts of the axial flux machine are named as **Iron**, **Winding** and **End winding** for stator(s); **Iron**, **Magnet** and **Conductor** for the rotor(s). There is also the **General** material type which corresponds to nonconductive and nonmagnetic material such as air or insulation. Clicking button  opens the file with material properties.

Stacking factor field specifies the ratio of the volume filled by electrical steel to the total volume of the iron core. The total volume of the iron core consists of the volume of lamination steel sheets and the volume of the coating between the sheets. Stacking factor reduces the flux carried by the iron core which is taken into account by MotorXP-AFM through the stacking factor value.

Temperature of stator winding specifies the temperature of the winding conductors. Winding phase resistance is automatically adjusted for the given temperature according to the winding material temperature coefficient of resistance. The expression for temperature effect on resistance is as follows:

$$R=R_{20^{\circ}\text{C}}*[1+\alpha*(T-20^{\circ}\text{C})],$$

where R – resistance at temperature T ; $R_{20^{\circ}\text{C}}$ – resistance at 20°C ; α – temperature coefficient of resistance; T – winding temperature.

Temperature of magnet has its effect on the permanent magnet properties such as remanence flux density and intrinsic coercivity. The remanence flux density used by MotorXP-AFM is defined as follows:

$$Br=Br_{20^{\circ}\text{C}}*[1+\alpha_{Br}*(T_{pm}-20^{\circ}\text{C})/100],$$

where Br – remanence flux density at temperature T_{pm} ; $Br_{20^{\circ}\text{C}}$ – remanence flux density at 20°C ; α_{Br} – temperature coefficient of remanence flux density in $\% / ^{\circ}\text{C}$; T_{pm} – permanent magnet temperature. Similarly, the intrinsic coercivity is defined as follows:

$$Hcj=Hcj_{20^{\circ}\text{C}}*[1+\alpha_{Hcj}*(T_{pm}-20^{\circ}\text{C})/100],$$

where Hcj – intrinsic coercivity at temperature T_{pm} ; $Hcj_{20^{\circ}\text{C}}$ – intrinsic coercivity at 20°C ; α_{Hcj} – temperature coefficient of intrinsic coercivity in $\% / ^{\circ}\text{C}$; T_{pm} – permanent magnet temperature.

Temperature of the **Conductor** material affects the resistance of the conductor according to its temperature coefficient of resistance:

$$R=R_{20^{\circ}\text{C}}*[1+\alpha*(T-20^{\circ}\text{C})],$$

where R – resistance of the conductor at temperature T ; $R_{20^{\circ}\text{C}}$ – resistance at 20°C ; α – temperature coefficient of resistance; T – conductor temperature.

4.2.1. Adding new materials.

You can add your own materials to the Materials Library or modify properties of the existing materials. All material files are stored in *[User data directory]\Materials*. Each group of materials is stored in the separate folder (*Conductor*, *Iron* and *Magnet*) inside the *Materials* folder. The materials can also be sorted by subcategories, for example, ferrite magnets are stored in *[User data directory]\Materials\Magnet\Ferrite Magnets*, while neodymium magnets are stored in *[User data directory]\Materials\Magnet\Neodymium Magnets*. Each material has the following properties:

Conductor:

- Electric resistivity at 20⁰C
- Temperature coefficient of resistance
- Mass Density

Iron:

- B-H curve
- Iron loss
- Electric resistivity at 20⁰C (only for soft magnetic composites)
- Mass Density

Magnet:

- Electric resistivity
- Relative permeability
- Coercivity
- Remanence flux density
- Intrinsic coercivity
- Temperature coefficient of remanence flux density
- Temperature coefficient of intrinsic coercivity
- Mass Density

File with material properties is a txt-file of a special format. The example of the file with material properties is shown below (only part of the file is shown, full text can be found in *[User data directory]\Materials\Iron\Non-oriented Silicon Steels\Sura NO20.txt*):

<Description>

Sura NO20

Non-oriented silicon steel

<B-H curve> at 20°C

% H(A/m)	B(T)
0	0
10	0.0190
20	0.0530


```

30          0.1100
37          0.2000
...
...
...
30000       1.9730
50000       2.0590
100000      2.1830
130000      2.2310

```

<Iron loss>

```

% frequency(Hz)  flux density(T)  iron loss(w/kg)
50              0.1000           0.0200
50              0.2000           0.0700
50              0.3000           0.1400
50              0.4000           0.2300
50              0.5000           0.3200
50              0.6000           0.4200
50              0.7000           0.5400
50              0.8000           0.6600
50              0.9000           0.8000
50              1.0000           0.9500
50              1.1000           1.1400
50              1.2000           1.3600
50              1.3000           1.6500
50              1.4000           2.0000
50              1.5000           2.4000
50              1.6000           2.7500
50              1.7000           3.0600
50              1.8000           3.3200
400             0.1000           0.1700
400             0.2000           0.7200
...
...
...
10000           0.1000           27.0000
10000           0.2000           95.6000
10000           0.3000           191.0000
10000           0.4000           315.0000

```

<Mass Density> (kg/m³)
7650

Each property is designated by a *property tag*, the name of the tag is enclosed inside the angle brackets: <B-H curve>, <Iron loss> and etc. The tag is followed by the *property content* up to the next property tag or the end of the file. If the property is not specified it means that the corresponding property tag is followed by empty line(s) up to the next property tag or the end of the file. The content of a line following after the percent sign “%” and after the property tag is ignored and treated as a comment.

Refer to the corresponding files in the Materials Library to check the structure and units of each property content while adding new material files. There are properties defined by a single value such as <Mass Density>, <Electric resistivity>, <Coercivity> and others are defined by arrays of values (<Iron loss>, <B-H curve>).

The name of the material is following right after the <Description> tag. Each property file must have a unique material name. The name of the property file may be different from the name of the material.

The magnetization properties of permanent magnets are defined by the following property tags: <Relative Permeability>, <Coercivity> and <Remanence flux density>. It is allowed to specify either all of them or any two of them and leave another one empty.

The iron loss of the iron can be defined either by iron loss data or by iron loss Steinmetz equation coefficients (see chapter 2.6). The iron loss defined by the iron loss data are shown in the example above.

The first and second columns are frequency and peak flux density values respectively in increasing order exactly as shown above and the third column is the corresponding iron loss values. There are the following requirements for the flux density values:

<Iron loss>		
% frequency(Hz)	flux density(T)	iron loss(W/kg)
50	0.1	0.018371768
50	0.2	0.06912301
50	0.3	0.138519028
50	0.4	0.228291933
50	0.5	0.337521845
50	0.6	0.447654955
50	0.7	0.593724411
50	0.8	0.721922481
50	0.9	0.858938885
50	1.0	1.00136478
100	0.1	0.04015814
100	0.2	0.154411359
100	0.3	0.323172176
100	0.4	0.532617086
100	0.5	0.787456215
100	0.6	1.044402552
100	0.7	1.355424938
100	0.8	1.721270226
100	0.9	2.092930066
100	1.0	2.490155197
200	0.1	0.102196015
200	0.2	0.384508248
200	0.3	0.804748873
200	0.4	1.326299205
200	0.5	1.960888187
200	0.6	2.600724444
200	0.7	3.449338764
200	0.8	4.380356263
200	0.9	5.443134384
200	1.0	6.33704502
400	0.1	0.277585148
400	0.2	1.05364468
400	0.3	2.185861503
400	0.4	3.602498209
400	0.5	5.326170862
400	0.6	7.219224808
400	0.7	9.574852126
400	0.8	12.15921843
1000	0.1	1.067337791
1000	0.2	3.929521078
1000	0.3	8.22421281

Example of the iron loss defined by the Steinmetz equation coefficients can be found in M-15 29 Ga.txt:

```
<Iron loss>
%Steinmetz coefficients
0.0227288148583325    % kh
1                      % alfa
1.77364043109264      % beta
2.11492029537421e-05  % ke
```

If the iron loss property is not specified (property tag <Iron loss> is followed by empty line(s) up to the next property tag or the end of the file) it is assumed that the iron losses are zero.

Soft magnetic composite materials also require the <Electric resistivity> property for calculation of the between-particle eddy current iron losses. If the electric resistivity is not specified for the soft magnetic composite material, the iron losses calculation result may be incorrect.

4.3. Winding Editor.

Winding Editor tab of MotorXP-AFM Design Studio allows you to set up parameters of the stator winding. **Winding Editor** is shown in Figure 4.13.

Parameters of **Winding Editor** such as **Number stator of slots**, **Number of pole pairs**, **Winding type** (*Planar* or *Toroidal*), **Winding layers** (*Single layer* or *Double layer*), **Winding layers orientation** (*Upper / Lower* or *Left / Right*) are the same as in **Geometry Editor** and described in section 4.1.

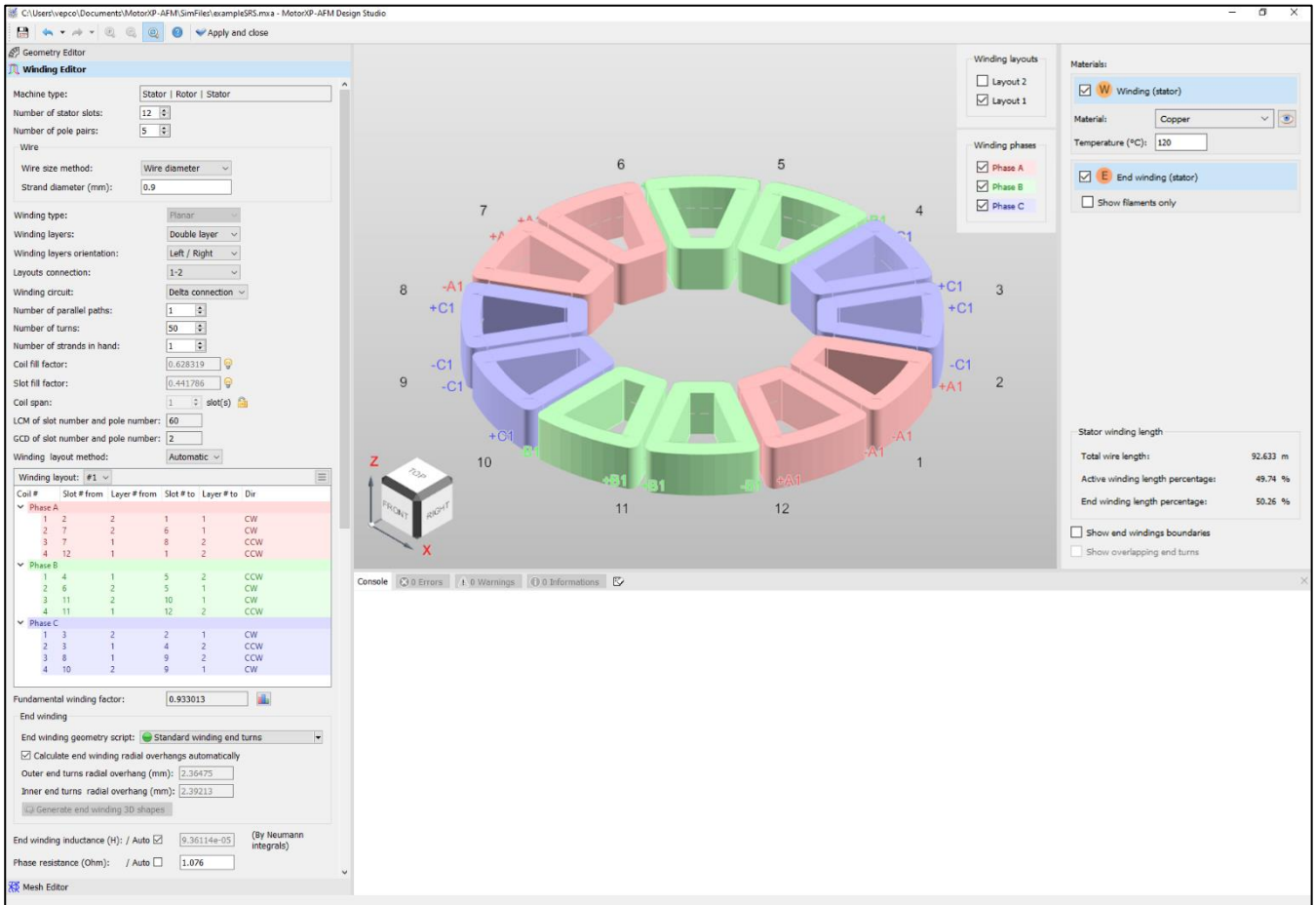


Figure 4.13. Winding Editor of MotorXP-AFM Design Studio.

The **Wire size method** pop-up menu specifies how the **Strand diameter** value is determined. **Strand diameter** specifies the diameter of one strand (wire) of the winding. There are four wire size methods used in MotorXP-AFM: **Wire diameter** (the wire diameter should be specified by the user), **AWG** (the wire diameter is specified by the AWG number according to the American wire gauge standard), **SWG** (the wire diameter is specified by the SWG number according to the Standard wire gauge), **Fill factor** (whether the **Slot fill factor** value or the **Coil fill factor** value should be specified by the user).

Layouts connection pop-up menu appears when there are whether two stators (each of them having its own winding) or one stator having two windings. The **Layouts connection** pop-up menu specifies the connection between these windings herein referred to as “layouts”. For example, **1-2** selection from the

Layouts connection pop-up menu specifies that the **Layout 1** and **Layout 2** are connected in series, whereas **1//2** selection specifies that the **Layout 1** and **Layout 2** are connected in parallel.

Winding circuit pop-up menu specifies whether the winding is star-connected or delta-connected.

Number of parallel paths specifies the number of groups of coils per phase for one layout connected in parallel.

Examples of stator winding configurations for star and delta connections and for one and three parallel paths are shown in Figures 4.14(a)-4.14(f).

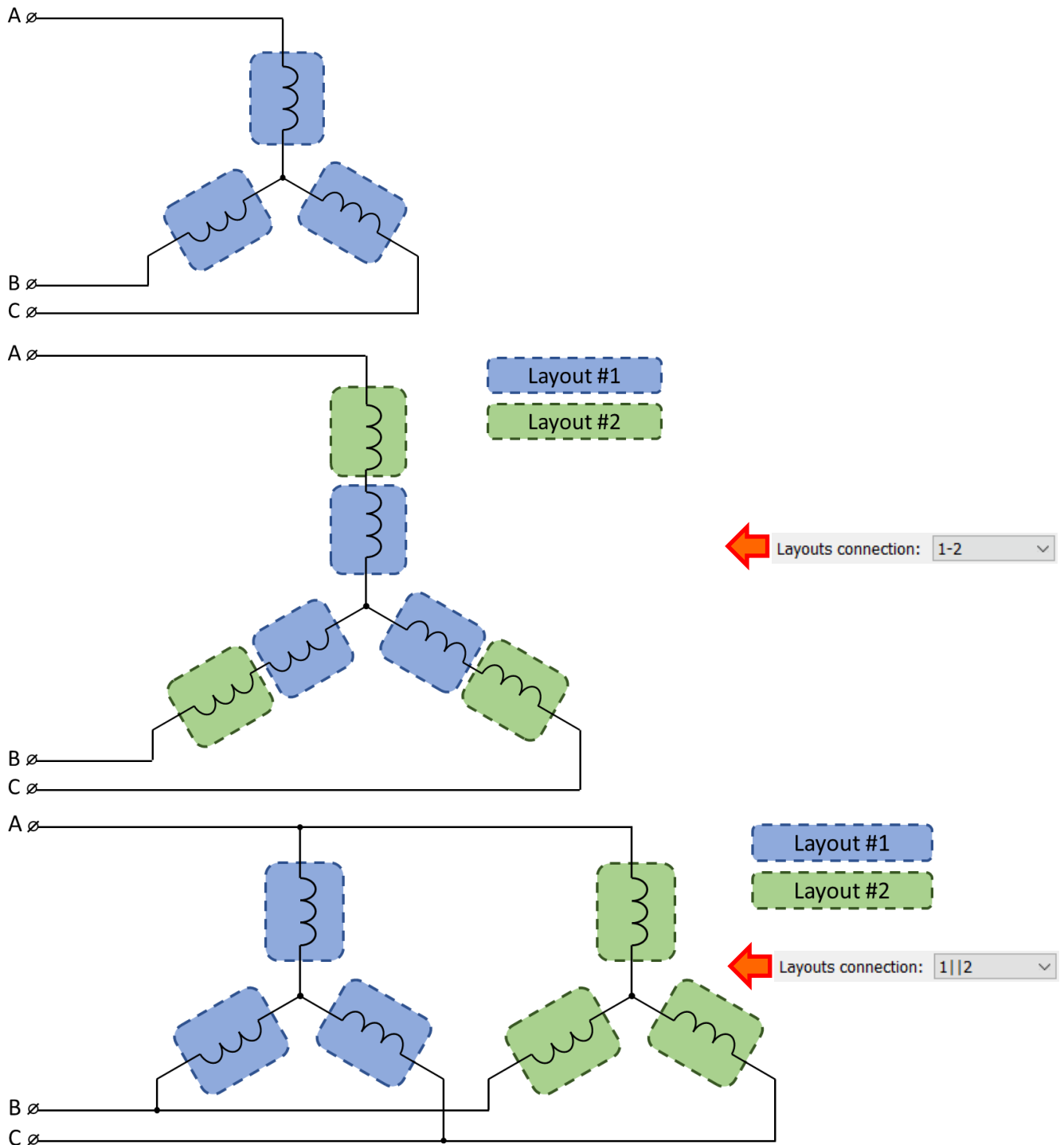


Figure 4.14(a). Examples of stator winding configurations for star connection, one parallel path and one layout (top), two layouts connected in series **1-2** (middle), two layouts connected in parallel **1||2** (bottom).

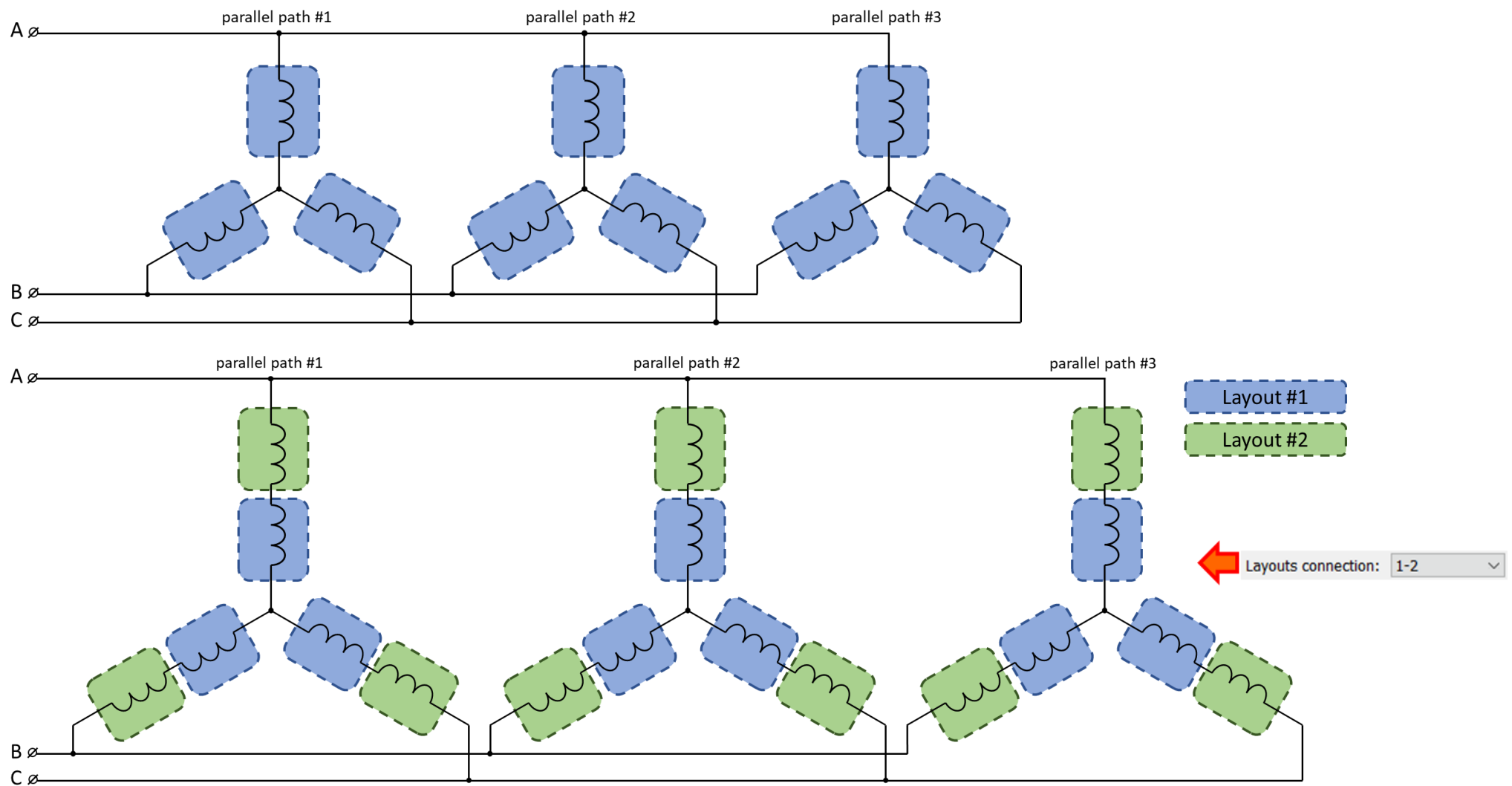


Figure 4.14(b). Examples of stator winding configurations for star connection, three parallel paths and one layout (top), two layouts connected in series **I-2** (bottom).

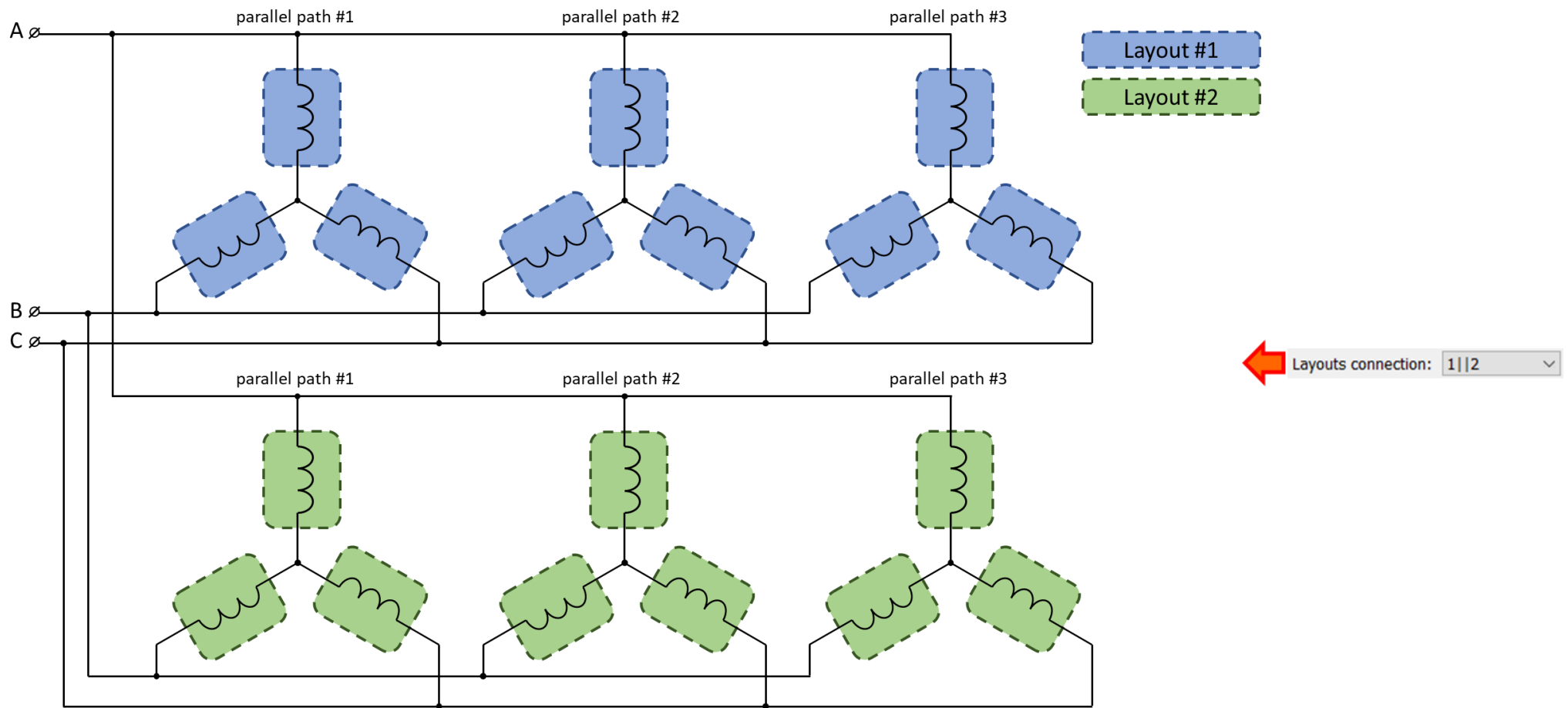


Figure 4.14(c). Example of stator winding configuration for star connection, three parallel paths and two layouts connected in parallel $1//2$.

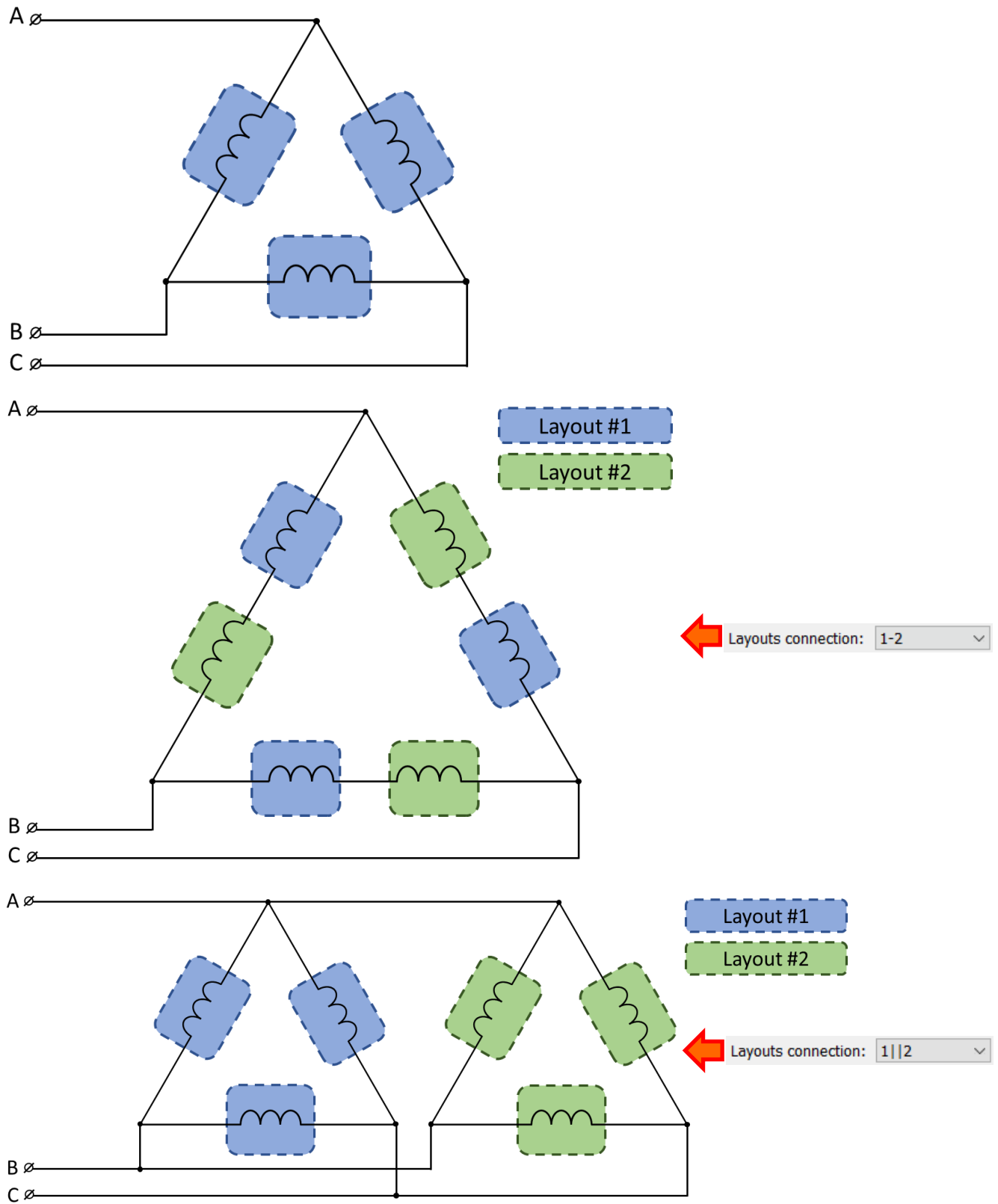


Figure 4.14(d). Examples of stator winding configurations for delta connection, one parallel path and one layout (top), two layouts connected in series *1-2* (middle), two layouts connected in parallel *1||2* (bottom).

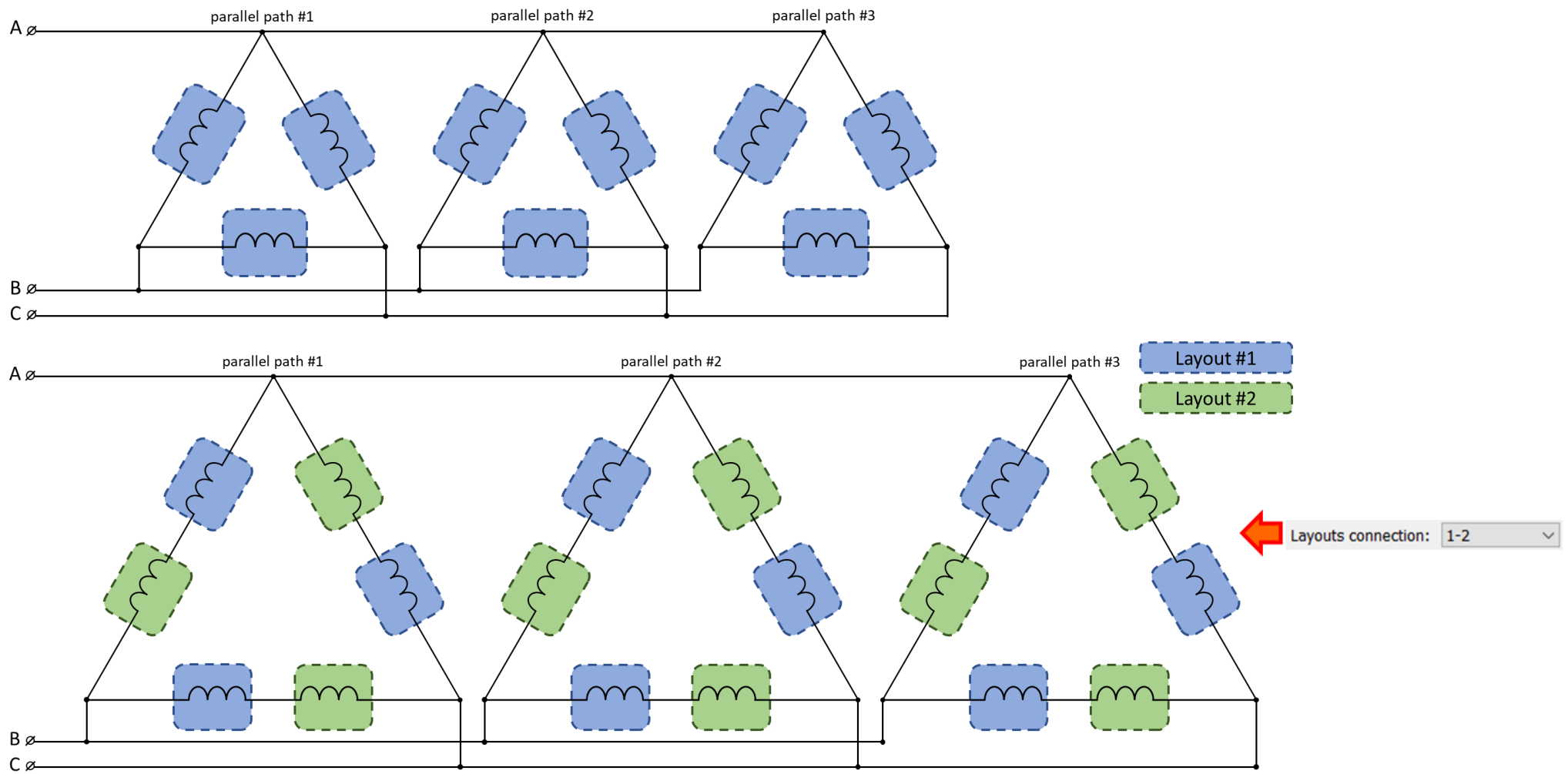


Figure 4.14(e). Examples of stator winding configurations for delta connection, three parallel paths and one layout (top), two layouts connected in series **I-2** (bottom).

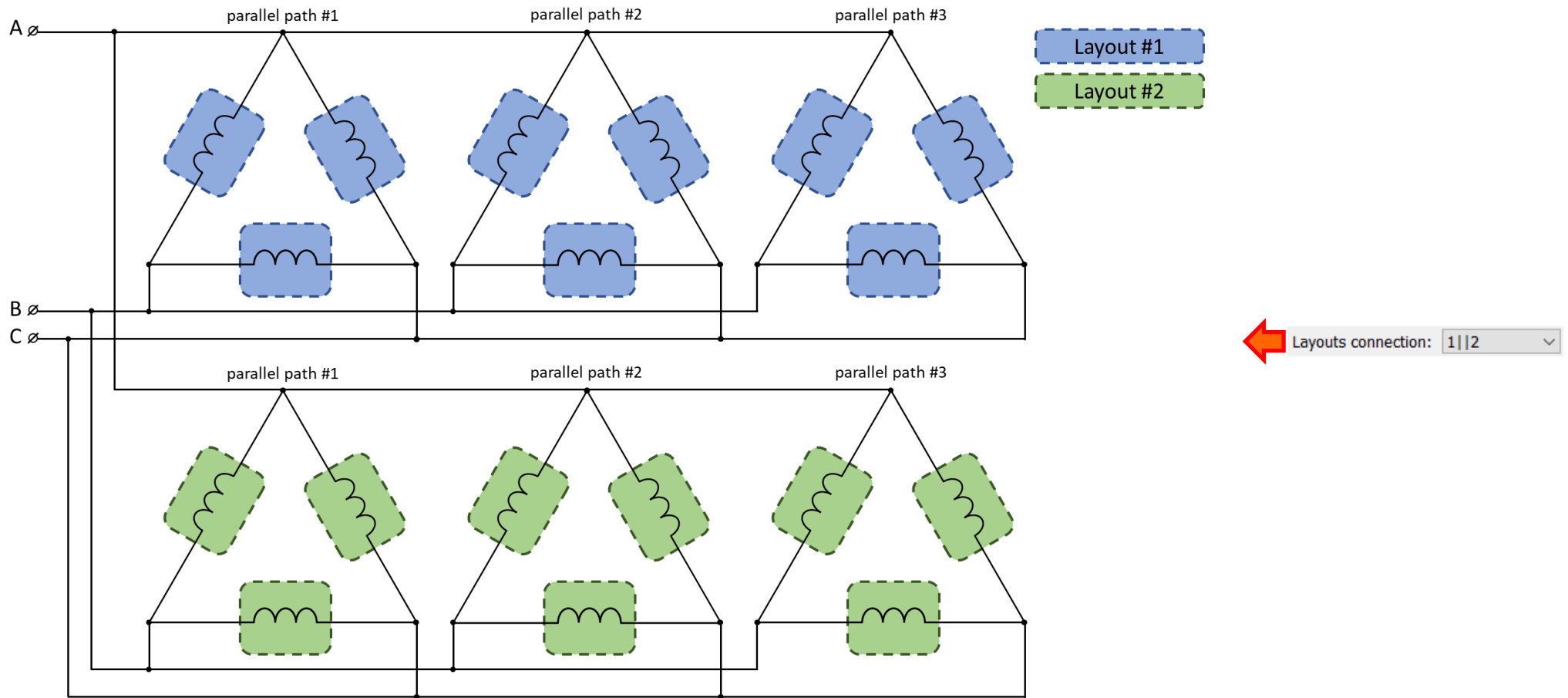




Figure 4.14(f). Example of stator winding configuration for delta connection, three parallel paths and two layouts connected in parallel $1/2$.

Number of turns specifies the number of turns per one coil, the same as the number of turns per one winding layer. **Number of strands in hand** specifies a number of smaller wires (strands) bundled together to form a larger conductor to reduce the skin-effect. If the conductor consists of one piece of metal wire the number of strands is one. **Slot fill factor** field specifies the ratio of the area of all conductors (i.e. pure copper or aluminum) of a slot to the total slot area (including slot insulation). **Coil fill factor** is the ratio of the area of all conductors (i.e. pure copper or aluminum) of a slot to the slot area occupied by the conductors NOT including slot insulation. Use button  on the right to highlight the area used for calculation of coil or slot fill factors.

Coil span is the distance between the two sides of a coil measured in number of slots. Coil span can be changed if the *Upper / Lower* winding layer orientation is selected. If the *Left / Right* winding layer orientation is selected the concentrated winding is assumed and the coil span is automatically set up to 1. **LCM of slot number and pole number** specifies the least common multiple (LCM) between the number of slots and number of poles. It is equal to the number of periods of cogging torque waveform per rotor revolution. LCM should be as high as possible to minimize cogging torque.


GCD of slot number and pole number specifies the greatest common divisor (GCD) between the number of slots and number of poles. It should be an even number and as high as possible to ensure better balance of axial forces with reduced noise and motor vibrations.

There are two methods to specify the stator winding layout: *Automatic* and *Manual*. When you choose the **Winding layout method** as *Automatic* you should also specify **Coil span** in number of slots. Winding layout is displayed as three tables corresponding to each phase. First column of each table is a coil number. Second column indicates the starting slot number, and third column indicates the starting layer number if the winding has more than one layer. Whereas fourth column indicates the ending slot number, and fifth column indicates the ending layer of each coil if the winding has more than one layer. Sixth column indicates the direction of each coil as clockwise (CW) or counter-clockwise (CCW). There is also an additional seventh column for the parallel path number appearing if the number of parallel paths is more than 1. The same parallel path number corresponds to coils of the same phase of the same layout connected in series. The **Winding layout #** pop-up indicates which winding layout is currently displayed in the table below. The winding layout can be copied or saved by clicking the  button at the top-right corner of the winding layout table.

Colored representation of the winding layout is displayed in the center part of **Winding Editor**, where the sign "+/-" specifies the forward and return direction of the conductors within a slot, the letter following the sign specifies the phase and the phase is followed by the parallel path number. Certain layouts and certain winding phases can be shown/hidden by using the corresponding checkboxes at the top-right corner of the winding view.

Fundamental winding factor is a ratio of flux linked by the winding compared to flux linked by a single-layer full-pitch non-skewed integer-slot winding with the same number of turns and one single slot per

pole per phase. Fundamental winding factor should be as high as possible (close to 1) to maximize torque.

Use button  on the right to see harmonic winding factors.

End winding geometry script pop-up menu specifies the JavaScript file generating the end winding 3D shapes. The default end winding geometry script is *Standard winding end turns*. Other end winding geometry scripts can be added by the user to define end turns for special winding arrangements.

Inner and outer end winding overhang distances can either be automatically created or manually specified.

☐ Calculate end winding radial overhangs automatically checkbox can be selected in order to find overhangs automatically, so the end turns take minimum possible volume and do not overlap. **Outer end turns radial overhang** specifies the radial distance between the teeth outer diameter and the farthest point of the winding outer end turns. This parameter can be calculated automatically (if the corresponding checkbox is selected) or defined manually. **Inner end turns radial overhang** specifies the radial distance between the teeth inner diameter and the farthest point of the winding inner end turns. This parameter can also be calculated automatically (if the corresponding checkbox is selected) or defined manually.

If the *Single layer* winding is selected and the **Coil span** is higher than 1, the **Outer end turns height** and **Inner end turns height** parameters will appear. **Outer end turns height** specifies the distance between the lowest and highest points of the winding outer end turns of the same layout. **Inner end turns height** specifies the distance between the lowest and highest points of the winding inner end turns of the same layout.

Generate end winding 3D shapes button is used to create the 3D shapes of the end windings. By default, after changing the geometry or winding parameters, only the contour lines (filaments) of the end windings are shown and the 3D shapes are created only by clicking the button. The corresponding error/warning message informs about a need to generate the end winding 3D shapes.

End winding inductance specifies the leakage inductance of the stator winding end turns per phase for one layout. End winding inductance can whether be calculated automatically (if / **Auto** is chosen) based on the winding configuration and machine dimensions or defined manually.

Phase resistance specifies the active DC resistance of the stator winding per phase for one layout for the winding temperature specified in the **Materials** panel. Winding phase resistance is automatically adjusted for the given temperature according to the conductor material temperature coefficient of resistance (see section 4.2). Phase resistance can whether be calculated automatically (if / **Auto** is chosen) based on the winding configuration and machine dimensions or defined manually.

Show end winding filaments only checkbox of the **Material** panel on the right side allows to choose between 3D end turns view and the view when only center lines (filaments) of the end turns are shown.

4.4. Mesh Editor.

Mesh Editor tab of MotorXP-AFM Design Studio is shown in Figure 4.15. It is recommended to examine the finite element mesh before starting the analysis. Quality of the mesh influences the simulation result accuracy. To enhance the accuracy of the simulation results the triangles of the mesh should be close to an equilateral triangle as much as possible.

Number of cylindrical slices field specifies the number of cylindrical slices the machine is divided in and used by the multi-slice FEA, i.e. the number of the machine's cylindrical cross-sections in which the magnetic field is simultaneously calculated. The simulation time is proportional to the number of cylindrical slices. The minimum number of cylindrical slices is 1 which can be used for getting fast simulation results at initial design stages. The recommended number of cylindrical slices is at least 3 providing sufficient accuracy in most cases. For more details on how the axial flux machine is modelled in MotorXP-AFM refer to section 2.4.

You can apply periodic or antiperiodic boundary conditions choosing corresponding item from the **Boundary conditions** pop-up menu. If *None* is chosen from the **Boundary conditions** pop-up menu, the boundary conditions are not used. If / **Auto** is chosen, the best possible boundary conditions are automatically assigned. For more details on boundary conditions refer to section 2.2 of this manual.

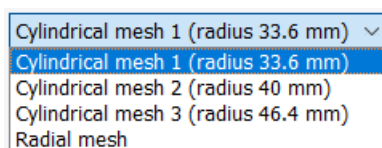
Maximum triangle side limits the length of the longest triangle side to the specified value. If / **Auto** is chosen, the **Maximum triangle side** value is set up to 10% of the height of the machine.

Number of layers in air gap specifies the total number of finite element layers placed in the air gap. Only odd numbers of finite element layers from 1 to 9 are allowed (refer to section 2.2 of this manual for more details).

Air gap mesh quality specifies the quality of the mesh in the air gap region (*Low*, *Medium* or *High*). The mesh of the highest quality would consist of equilateral triangles which is practically not possible for complex geometry. The lower air gap mesh quality usually leads to the smaller number of mesh triangles and faster computational speed sacrificing the accuracy.

The mesh view can be changed between *Flat view* and *3D view* at the top-left corner of the mesh view window. Both *Flat view* and *3D view* of the mesh are shown in Figure 4.15. As can be seen, *3D view* allows you to see how each cylindrical and radial mesh planes are oriented in space.

Each cylindrical/radial mesh planes can be viewed by selecting the corresponding item at the top-right corner of the mesh view window:



For the *Flat view* of the mesh, there is also the **Show subdomains with material types** checkbox at the bottom-right corner that allows to view material types at different subdomains as shown in Figure 4.16.

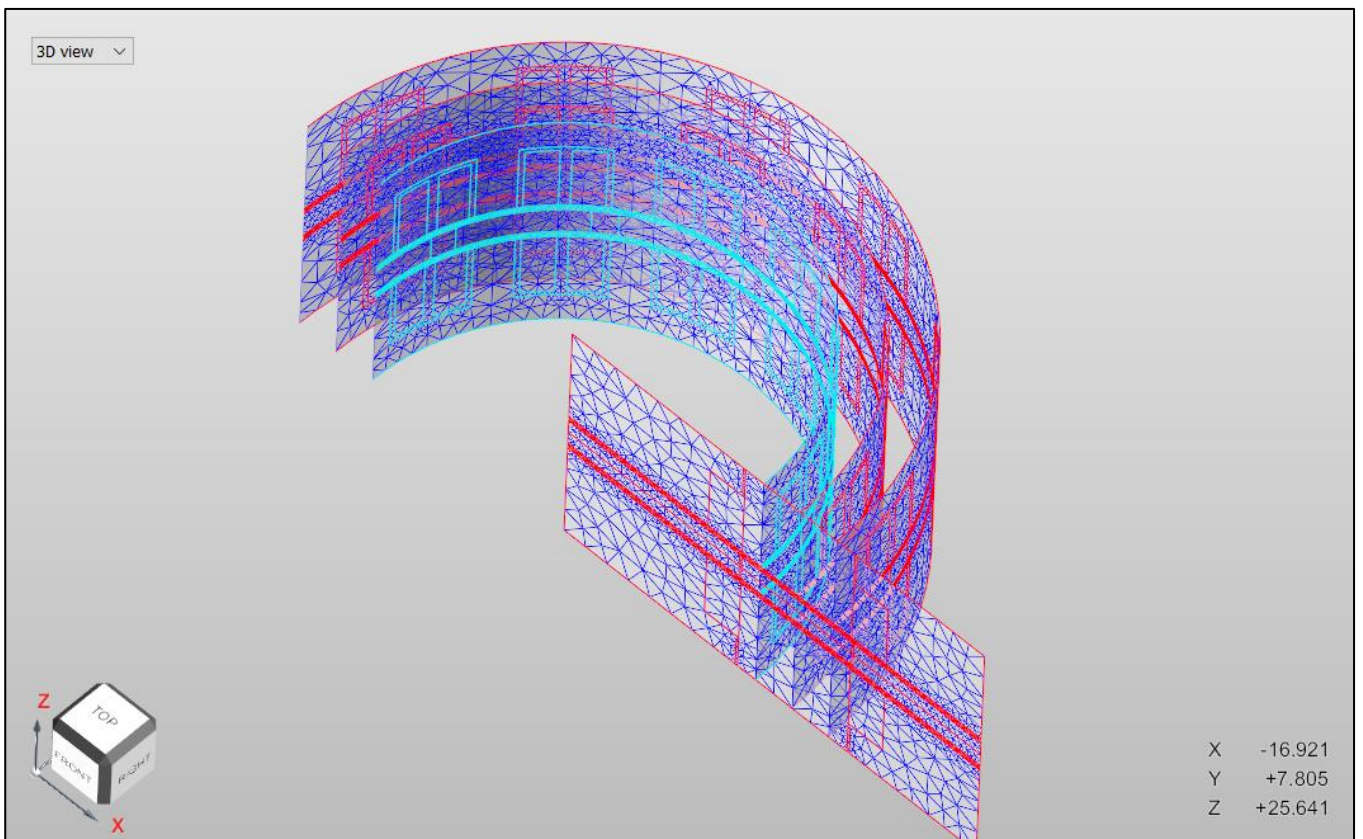
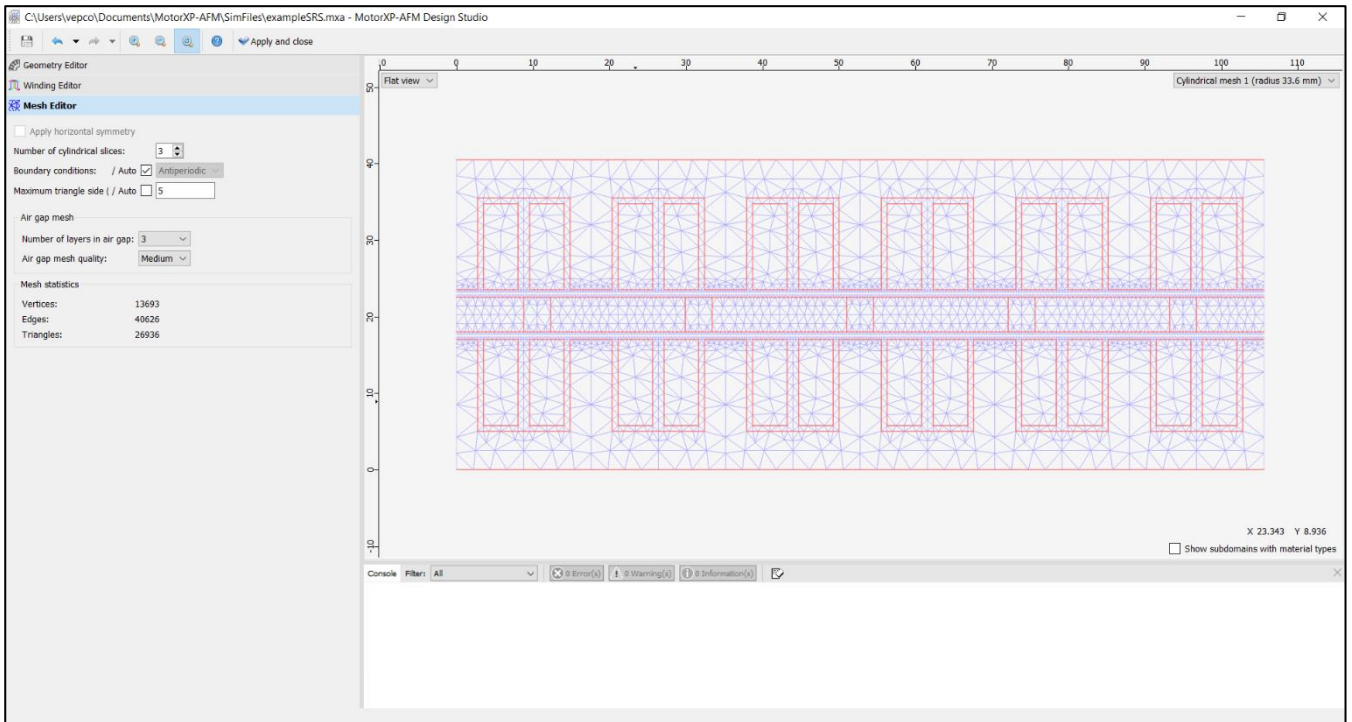


Figure 4.15. Mesh Editor of MotorXP-AFM Design Studio (top) and the view of the 2D mesh planes in Mesh Editor when **3D view** is selected (bottom).

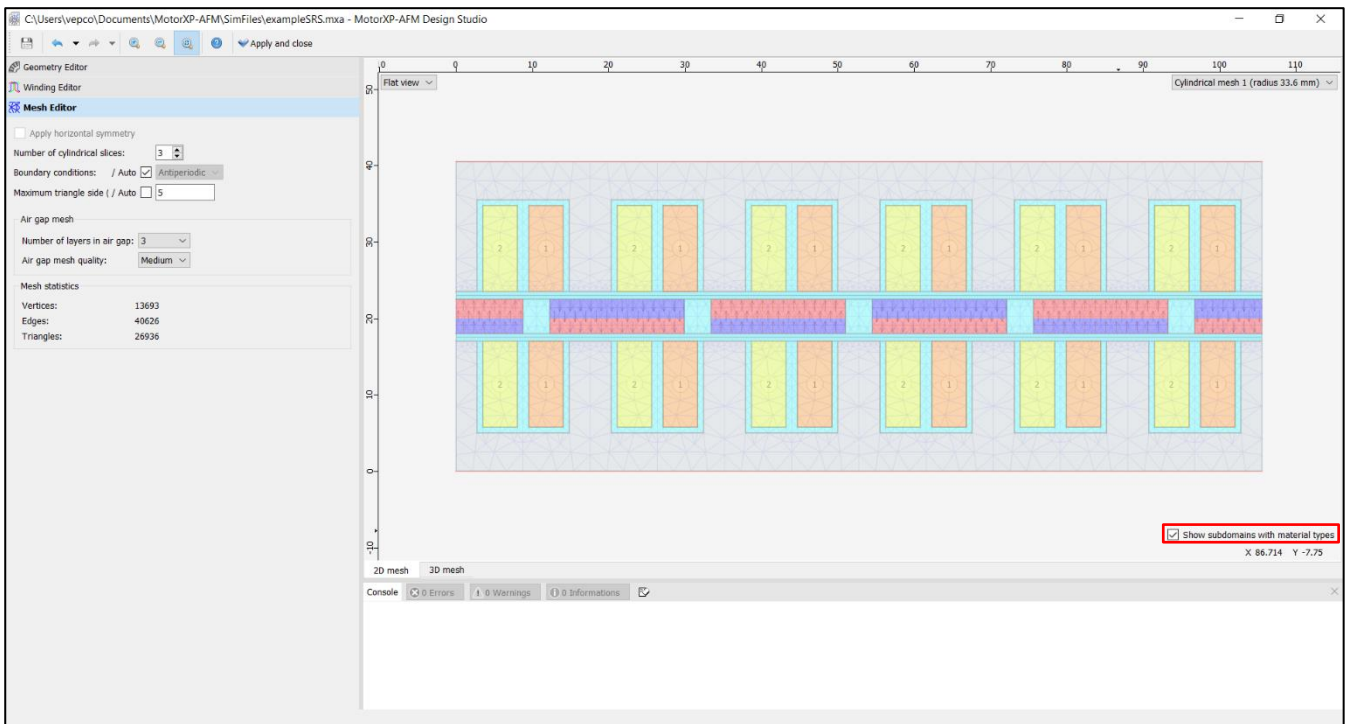


Figure 4.16. Mesh Editor with subdomains shown.

4.5. Drive Settings.

There are three drive types available in MotorXP-AFM (*Current hysteresis PWM*, *Space vector PWM* and *Six-step*) which can be chosen in the **Drive Settings** window as shown in Figures 4.18 – 4.22. It is assumed that the machine is supplied from the three-phase inverter shown in Figure 4.17 (for **Dynamic FE Analysis** the inverter circuit also includes diodes as shown in Figure 10.1):

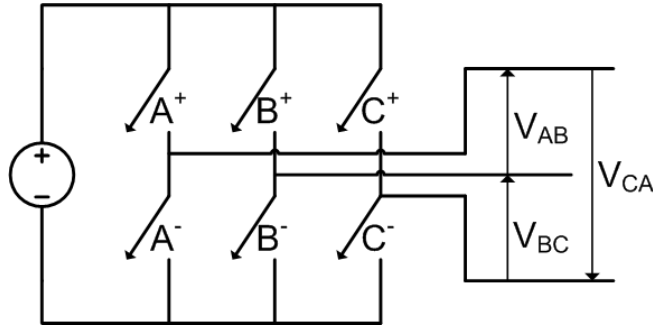


Figure 4.17. Three-phase inverter.

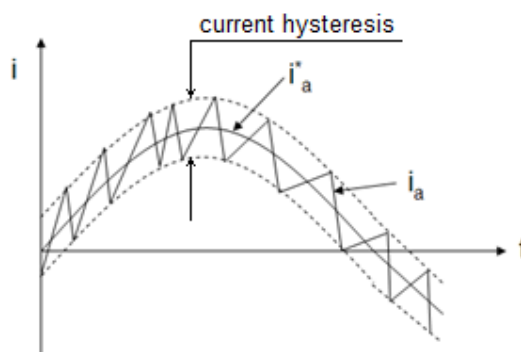
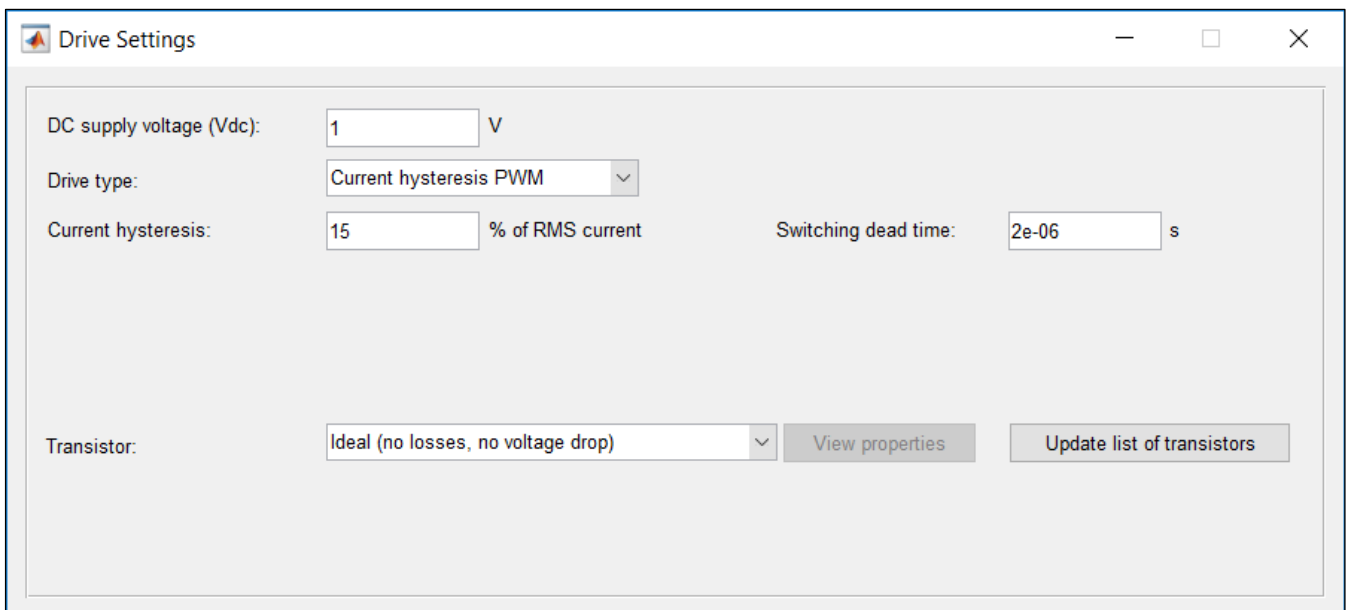


Figure 4.18. Current hysteresis PWM with the ideal transistor chosen.

The *Current hysteresis PWM* drive type is shown in Figure 4.18. The phase current is kept within the upper and lower bounds defined by the **Current hysteresis** field value.

The **Switching dead time** value normally varies from 0 to 5 microsecond. The switching dead time defines the small amount of time between switching edges for top and bottom switches of the same phase so during this time both switches are off. In the real inverter the switching dead time helps to avoid “short through” current between top and bottom switches because of the switch-off lag of the transistor. In MotorXP-AFM the switching dead time is taken into account in the dynamic D-Q simulations (see chapter 7 for more information on **Dynamic D-Q Analysis**); no dead time compensation methods are applied. The switching dead time is not included into standard simulation scripts of **Dynamic FE Analysis** (*simscrip_hystpwm.m*, *simscrip_spacevecpwm.m*, *simscrip_sixstep.m*), though the dead time can be implemented in the user defined simulation script (see chapters 8 and 9 for more information on **Dynamic FE Analysis** and simulation scripts).

The **Drive Settings** window when the *Space vector PWM* drive type is chosen is shown in Figure 4.19.

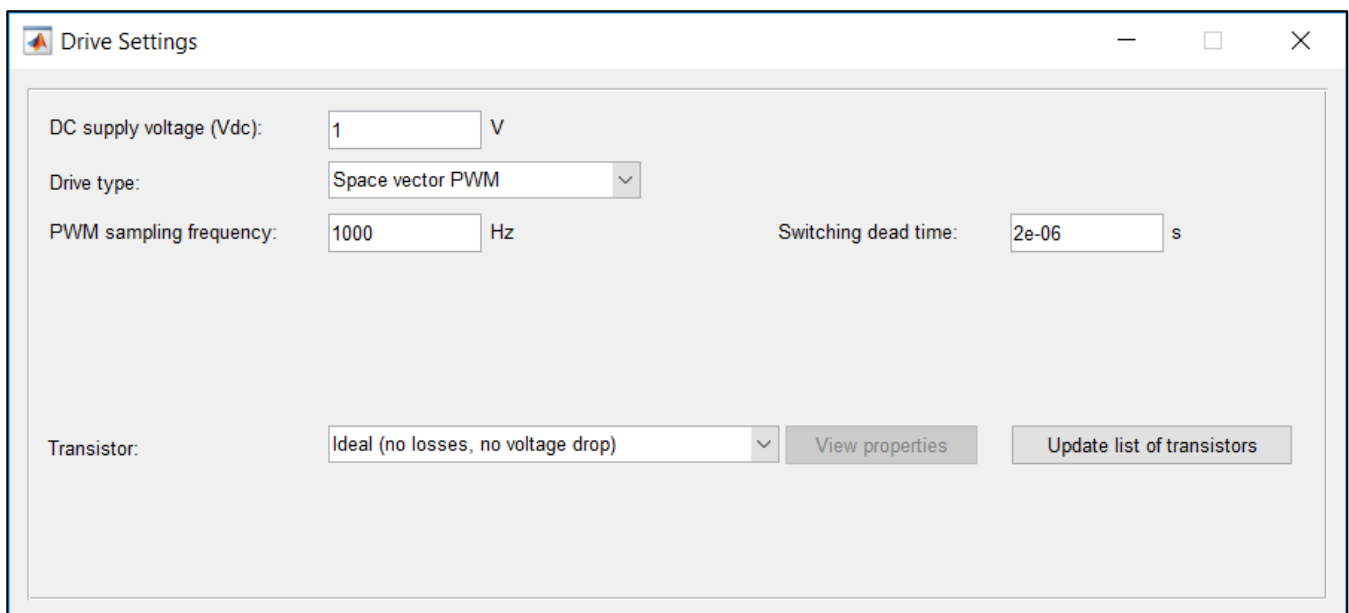
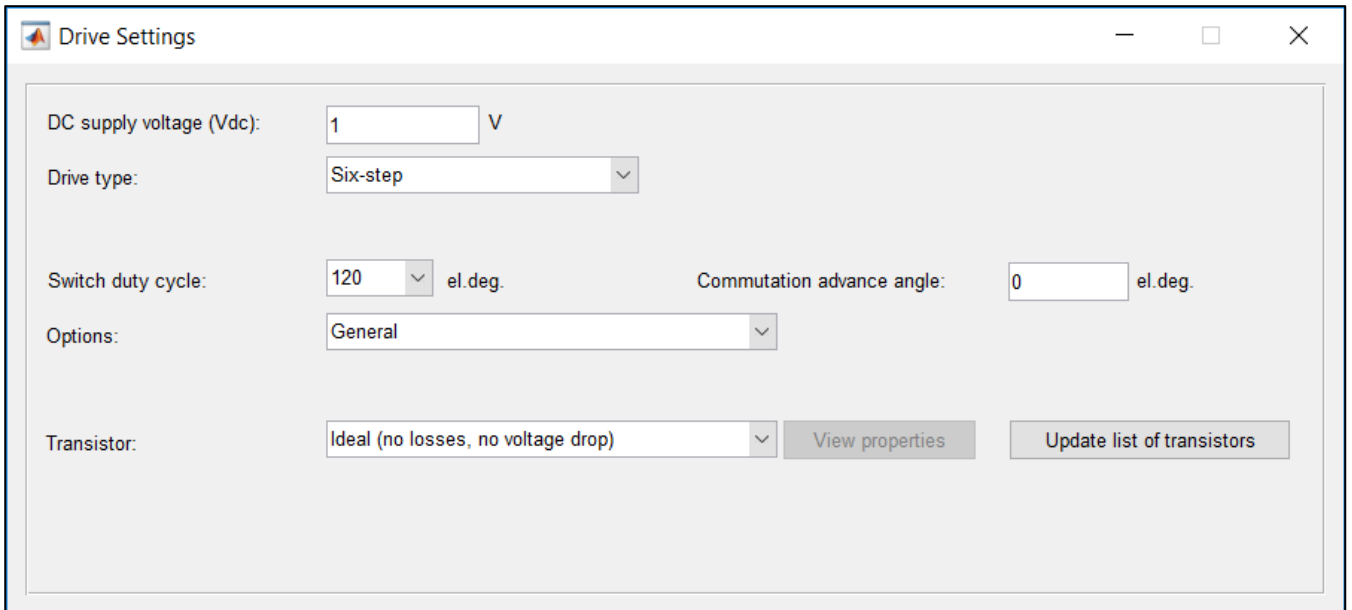


Figure 4.19. Space vector PWM with the ideal transistor chosen.

The **Drive Settings** window when the *Six step* drive type is chosen is shown in Figures 4.20 – 4.22.

Switch duty cycle specifies the time in electrical degrees when the switch is turned on. **120** degrees switch duty cycle corresponds to the inverter operation with two switches turned on at the same time; with **180** degrees switch duty cycle there will be three switches turned on at the same time.

In practice, the phase inductances and the non-ideal current commutations will cause a phase lag of the current with respect to the voltage. The lag of the current can be compensated by imposing the **Commutation advance angle** shifting forward the commutation timing.



Drive Settings

DC supply voltage (Vdc): 1 V

Drive type: Six-step

Switch duty cycle: 120 el.deg. Commutation advance angle: 0 el.deg.

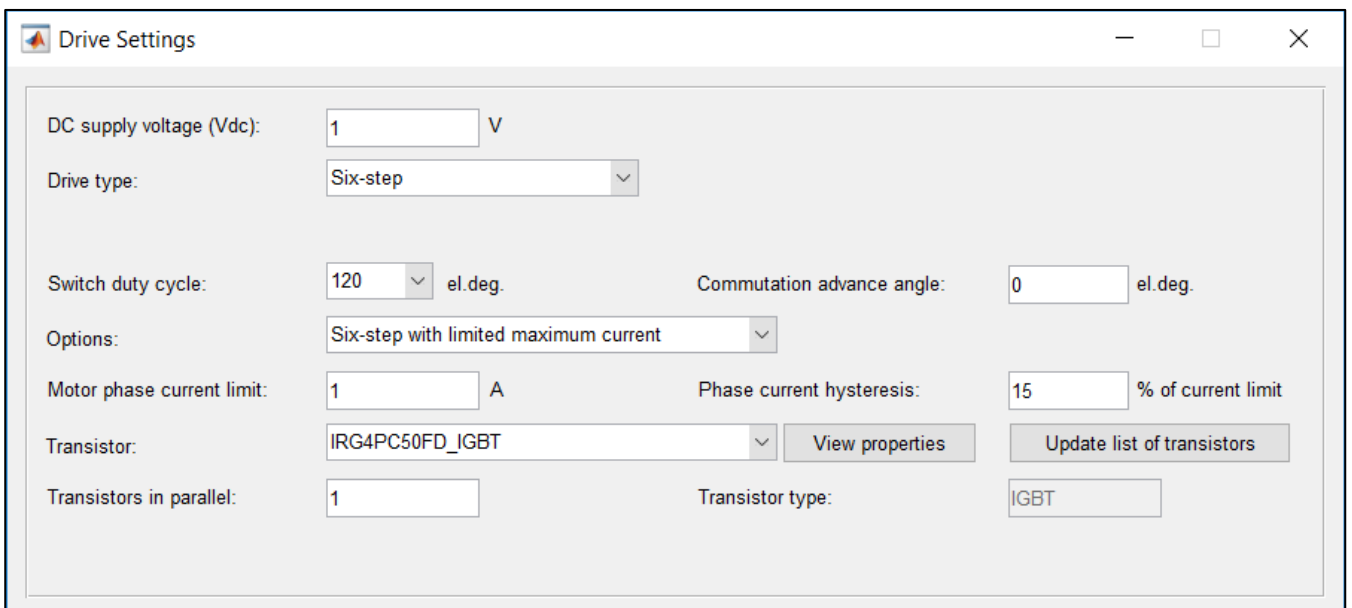
Options: General

Transistor: Ideal (no losses, no voltage drop) View properties Update list of transistors

Figure 4.20. Drive Settings window with six-step drive type and ideal transistor chosen.

There are two additional options for the six-step drive: *Six-step with limited maximum current* and *Six-step with variable DC voltage*.

When the six-step with limited maximum current is used (Figure 4.21), all switches are turned off once the current of any phase reaches the **Motor phase current limit**. The corresponding switches are turned on again when the maximum phase current becomes lower than the current limit minus the value specified by the **Phase current hysteresis** field.



Drive Settings

DC supply voltage (Vdc): 1 V

Drive type: Six-step

Switch duty cycle: 120 el.deg. Commutation advance angle: 0 el.deg.

Options: Six-step with limited maximum current

Motor phase current limit: 1 A Phase current hysteresis: 15 % of current limit

Transistor: IRG4PC50FD_IGBT View properties Update list of transistors

Transistors in parallel: 1 Transistor type: IGBT

Figure 4.21. Six-step drive with limited maximum current and IGBT transistor chosen.

When the six-step with variable DC voltage is used (Figure 4.22), the input DC voltage of the inverter is modulated so only the part of the battery voltage is used specified by the **Vdc usage percentage** field value. Make sure that the **PWM sampling frequency** is high enough not to interfere with the six-step commutation frequency.

The screenshot shows the 'Drive Settings' window with the following parameters:

- DC supply voltage (Vdc): 1 V
- Drive type: Six-step
- PWM sampling frequency: 1000 Hz
- Switch duty cycle: 120 el.deg.
- Commutation advance angle: 0 el.deg.
- Options: Six-step with variable DC voltage
- Vdc usage percentage: 100 % of Vdc
- Transistor: IAUT300N08S5N012_MOSFET (with a 'View properties' button next to it)
- Transistors in parallel: 1
- MOSFET driver resistor: 3.5 Ohm
- Buttons: 'Update list of transistors' and 'MOSFET' (in a separate box)

Figure 4.22. Six-step drive with variable DC voltage and MOSFET transistor chosen.

When the *ideal (no losses, no voltage drop)* item is chosen from the **Transistor** pop-up menu (as shown in Figures 4.18 – 4.20), no inverter losses are calculated, and no transistor on-state voltage drop is included into simulation. There are two transistor types in MotorXP-AFM supported at the moment for the inverter losses calculation: MOSFET and IGBT. Figure 4.21 shows the **Drive Settings** window when the transistor of IGBT type is chosen from the **Transistor** pop-up menu. The **Transistors in parallel** field, when specified greater than 1, allows to distribute the load current across the several transistors connected in parallel so the current of each individual transistor does not exceed the transistor limit, which also taken into account while the transistor losses are calculated. The **Drive Settings** window when the transistor of MOSFET type is chosen is shown in Figure 4.22. The **MOSFET driver resistor** field specifies the gate drive circuit external resistor value which influences the MOSFET switching losses. The inverter losses are only calculated with **Steady State D-Q Analysis** (see chapter 6) and **Dynamic D-Q Analysis** (see chapter 7). General information about the inverter losses calculation can be found in section 2.8.

4.5.1. Adding user defined transistors.

The properties of each transistor are specified in the separate txt-file of a special format stored in *[User data directory]\Materials\Transistor*. You can add your own transistor or modify properties of the

existing one. The format of the transistor property file is the same as the format of the material property file described in section 4.3.1.

Each property is designated by a *property tag*, the name of the tag is enclosed inside the angle brackets: <Drain-source resistance>, <Diode characteristic> and etc. The full list of properties of the transistor is given in Table 4.1 for MOSFET and in Table 4.2 for the transistor of IGBT type. The tag is followed by the *property content* up to the next property tag or the end of the file. If the property is not specified it means that the corresponding property tag is followed by empty line(s) up to the next property tag or the end of the file. The content of a line following after the percent sign “%” and after the property tag is ignored and treated as a comment.

4.5.1.1. Adding transistor of MOSFET type.

The parameters of the MOSFET specified in the property file are listed in Table 4.1.

Table 4.1. MOSFET parameters.

MOSFET parameter		Symbol [units]	Property file tag
Drain-to-source on-state resistance vs. drain current		R_{DSon} [mOhm] vs. I_D [A]	<Drain-source resistance>
Gate drive circuit output voltage		V_{Dr} [V]	<Driver output voltage>
Option 1	Gate Muller plateau voltage	$V_{plateau}$ [V]	<Plateau voltage>
	Reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage	C_{rss} [pF] vs. V_{DS} [V]	<Gate-drain capacitance>
	MOSFET drain current rise time	t_{ri} [ns]	<Rise time>
	MOSFET drain current fall time	t_{fi} [ns]	<Fall time>
Option 2	Turn-on MOSFET switching losses vs. on-state drain current	E_{on} [mJ] vs. I_D [V]	<Switch-on loss>
	Turn-off MOSFET switching losses vs. on-state drain current	E_{off} [mJ] vs. I_D [V]	<Switch-off loss>
Freewheeling diode reverse recovery charge		Q_{rr} [nC]	<Diode reverse recovery charge>
Freewheeling diode current vs. diode voltage		I_f [A] vs. V_f [V]	<Diode characteristic>

The process of creating the MOSFET property file is demonstrated on the examples of the IAUT300N08S5N012 transistor and SiC MOSFET transistor FF11MR12W1M1_B11, the corresponding property files can be found in [User data directory]\Materials\Transistor\ (files IAUT300N08S5N012_MOSFET.txt and FF11MR12W1M1_B11_MOSFET.txt) and the corresponding datasheet files can be found in [User data directory]\Materials\Transistor\datasheet\ (files IAUT300N08S5N012.pdf and Infineon-FF11MR12W1M1_B11-DS-v02_02-EN.pdf).

Drain-to-source on-state resistance vs. drain current characteristic $R_{DSon} = f(I_D)$ is read from the datasheet diagram as shown in Figure 4.23. At least two points are required. Web-based tool <https://automeris.io/WebPlotDigitizer> can be used to extract data from the diagram. As can be seen from Figure 4.23 the $R_{DSon} = f(I_D)$ characteristic used for the property file corresponds to the gate-to-source

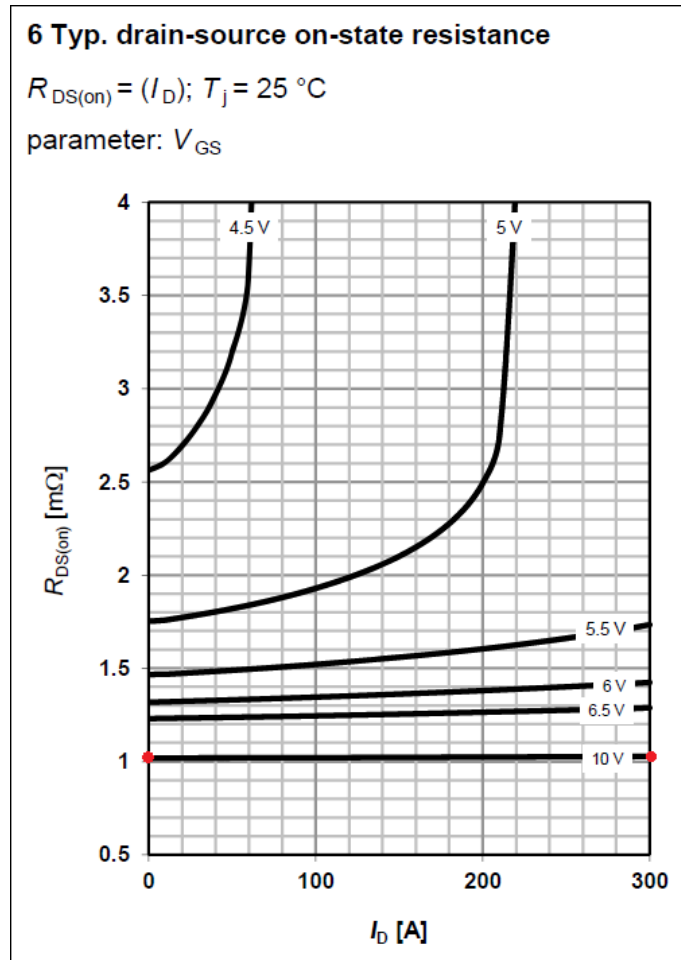
voltage $V_{GS} = 10V$ equal to the MOSFET gate driver output voltage. The R_{DSon} diagram shown in Figure 4.23 refers to the junction temperature $T_j = 25^{\circ}C$. While this should be sufficient for the majority of applications, most datasheets also provide the R_{DSon} variation diagram versus temperature so the R_{DSon} values for the expected junction temperature can be calculated.

There are two options for adding MOSFET transistor switching losses to the property file depending on the data provided by the manufacturer in the datasheet (see Table 4.1). Option 1 (on the example of IAUT300N08S5N012 transistor): using Gate Muller plateau voltage, reverse transfer capacitance vs. drain-to-source voltage diagram and MOSFET drain current rise and fall time; Option 2 (on the example of FF11MR12W1M1_B11 transistor): using turn-on and turn-off MOSFET switching losses vs. on-state drain current diagrams (see Table 4.1). Either of these two groups of parameters can be presented in the MOSFET transistor property file.

The gate Muller plateau voltage $V_{plateau}$ can be read from the MOSFET datasheet table or extracted from the typical gate charge vs. gate-to-source voltage diagram (whichever of them is available; see Figure 4.24). The plateau voltage $V_{plateau} = 4.5V$ appears in the MOSFET property file as shown in Figure 4.25. Reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage characteristic $C_{rss} = f(V_{DS})$ is read from the datasheet diagram as shown in Figure 4.25. At least two points are required. Figure 4.26 and Figure 4.27 show reading the current rise time and fall time and the diode reverse recovery charge from the datasheet.

Turn-on and turn-off MOSFET switching energy losses vs. on-state drain current can be read from the datasheet diagrams as shown in Figure 4.28 for the FF11MR12W1M1_B11 transistor.

Reading the freewheeling diode current vs. diode voltage characteristic $I_f = f(V_f)$ is shown in Figure 4.29.



```

<Drain-source resistance> Drain-to-source on-state resistance vs. drain current
% Id[A]      Rds_on[mOhm]          @ Tj=25*C, Vgs=10V
0            1
300          1

<Driver output voltage>
10  % [V]

```

Figure 4.23. Drain-to-source on-state resistance vs. drain current diagram from the datasheet (IAUT300N08S5N012) and the corresponding part of the MOSFET property file.

Gate Charge Characteristics²⁾

Gate to source charge	Q_{gs}	$V_{DD}=40\text{ V}, I_D=100\text{ A},$ $V_{GS}=0\text{ to }10\text{ V}$	-	56	73	nC
Gate to drain charge	Q_{gd}		-	37	56	
Gate charge total	Q_g		-	178	231	
Gate plateau voltage	$V_{plateau}$		-	4.5	-	V

15 Typ. gate charge
 $V_{GS} = f(Q_{gate}); I_D = 300\text{ A pulsed}$

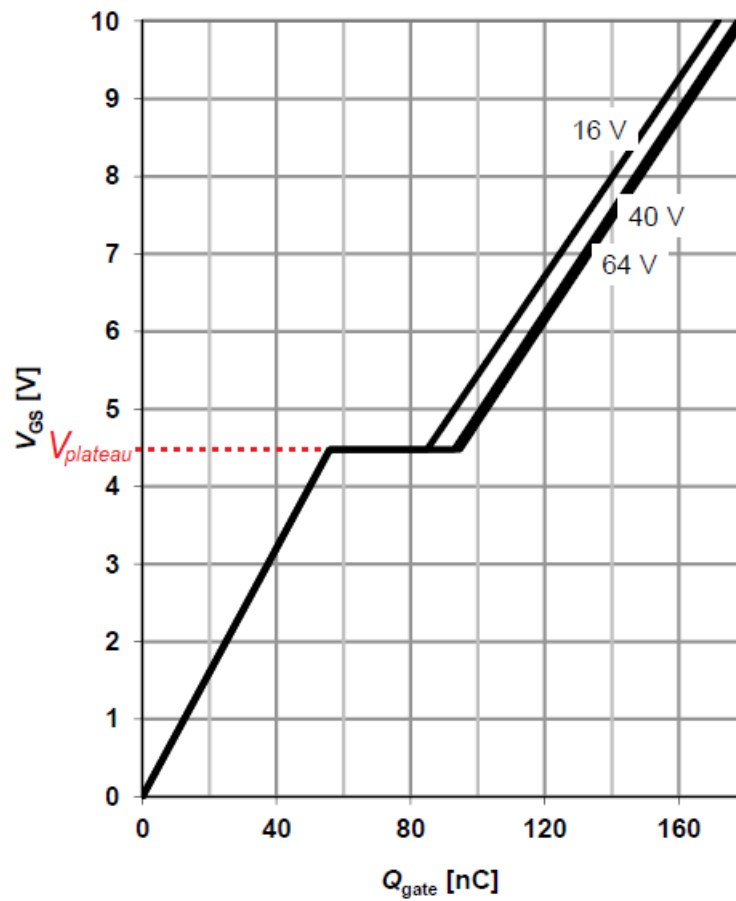
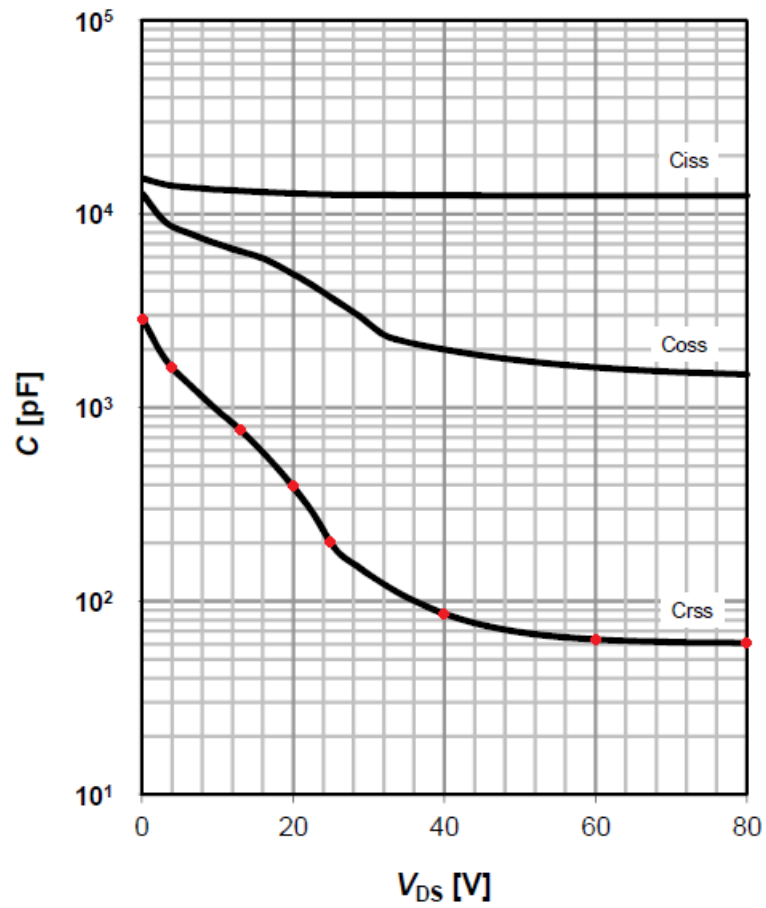
 parameter: V_{DD}


Figure 4.24. Two ways to determine the gate Muller plateau voltage $V_{plateau}$ from the MOSFET datasheet (IAUT300N08S5N012).

10 Typ. capacitances

$C = f(V_{DS}); V_{GS} = 0 \text{ V}; f = 1 \text{ MHz}$



<Plateau voltage>

4.5 % [V]

<Gate-drain capacitance>

% Reverse transfer capacitance vs. drain-source voltage

% vds[V] Crss[pF]

0 2832

4 1610

13 765

20 398

25 202

40 88

60 64

80 62

Figure 4.25. Reverse transfer (gate-to-drain) capacitance vs. drain-to-source voltage diagram from the datasheet (IAUT300N08S5N012) and the corresponding part of the MOSFET property file.

Dynamic characteristics ²⁾						
Input capacitance	C_{iss}	$V_{GS}=0\text{ V}, V_{DS}=40\text{ V},$ $f=1\text{ MHz}$	-	12500	16250	pF
Output capacitance	C_{oss}		-	2000	2600	
Reverse transfer capacitance	C_{rss}		-	86	130	
Turn-on delay time	$t_{d(on)}$	$V_{DD}=40\text{ V}, V_{GS}=10\text{ V},$ $I_D=100\text{ A}, R_G=3.5\ \Omega$	-	31	-	ns
Rise time	t_r		-	19	-	
Turn-off delay time	$t_{d(off)}$		-	69	-	
Fall time	t_f		-	55	-	

```

<Rise time>
19      % [ns]

<Fall time>
55      % [ns]

```

Figure 4.26. Reading the current rise time and fall time from the datasheet (IAUT300N08S5N012) and the corresponding part of the MOSFET property file.

Reverse Diode						
Diode continuous forward current ²⁾	I_S	$T_C=25\text{ °C}$	-	-	300	A
Diode pulse current ²⁾	$I_{S,pulse}$		-	-	1200	
Diode forward voltage	V_{SD}	$V_{GS}=0\text{ V}, I_F=100\text{ A},$ $T_J=25\text{ °C}$	-	0.9	1.2	V
Reverse recovery time ²⁾	t_{rr}	$V_R=40\text{ V}, I_F=50\text{ A},$ $di_F/dt=100\text{ A}/\mu\text{s}$	-	86	-	ns
Reverse recovery charge ²⁾	Q_{rr}		-	177	-	nC

```

<Diode reverse recovery charge>
177      % [nC]

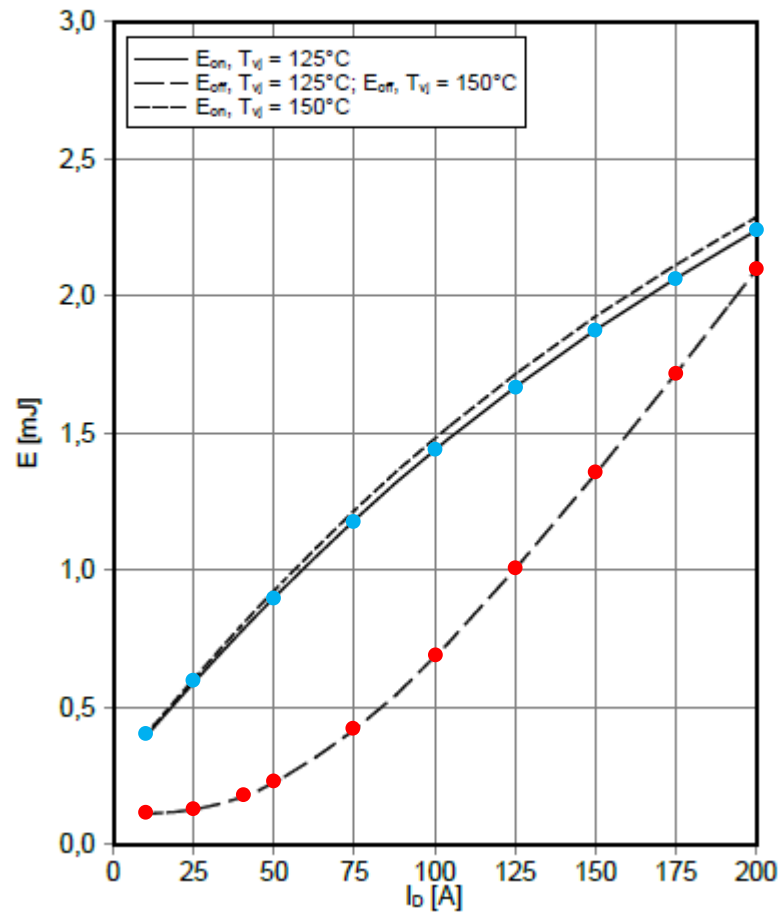
```

Figure 4.27. Reading the diode reverse recovery charge from the datasheet (IAUT300N08S5N012) and the corresponding part of the MOSFET property file.

Schaltverluste MOSFET (typisch)
switching losses MOSFET (typical)

$$E_{on} = f(I_D), E_{off} = f(I_D)$$

$$V_{GS} = -5 \text{ V} / 15 \text{ V}, R_{Gon} = 3,9 \, \Omega, R_{Goff} = 3,9 \, \Omega, V_{DS} = 600 \text{ V}$$



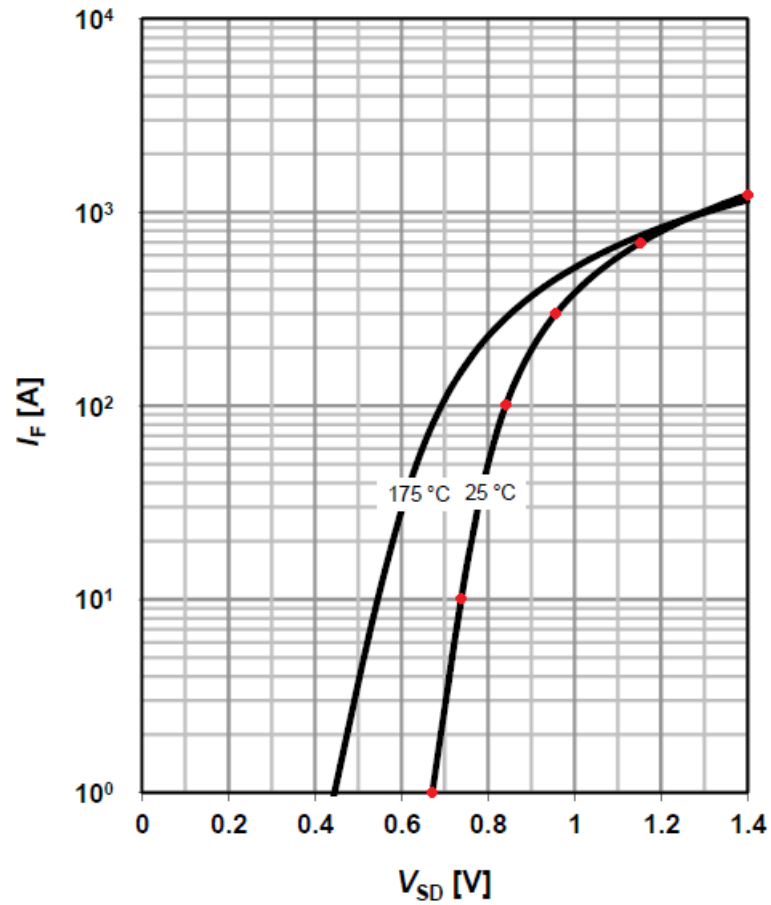
<Switch-on loss> @ Tj=125°C		<Switch-off loss> @ Tj=125°C	
% Id[A]	Eon[mJ]	% Id[A]	Eoff[mJ]
0	0	0	0
10.174	0.39770	10.174	0.10614
25.000	0.59207	25.000	0.12660
49.855	0.89898	39.390	0.16752
74.709	1.1803	49.855	0.22379
100.00	1.4361	75.145	0.41816
124.85	1.6662	100.00	0.68414
150.15	1.8760	124.85	1.0013
175.00	2.0652	149.71	1.3491
199.85	2.2391	175.00	1.7174
		199.85	2.0959

Figure 4.28. Turn-on and turn-off MOSFET switching energy losses vs. drain current diagrams from the datasheet (FF11MR12W1M1_B11) and the corresponding part of the MOSFET property file.

11 Typical forward diode characteristics

$$I_F = f(V_{SD})$$

parameter: T_j



<Diode characteristic>		@ $T_j=25^{\circ}\text{C}$
% Vf[V]	If[A]	
0.67	1	
0.74	10	
0.84	100	
0.96	298	
1.16	698	
1.4	1241	

Figure 4.29. Freewheeling diode current vs. diode voltage diagram from the datasheet (IAUT300N08S5N012) and the corresponding part of the MOSFET property file.

4.5.1.2. Adding transistor of IGBT type.

The parameters of the IGBT specified in the property file are listed in Table 4.2.

Table 4.2. IGBT parameters.

IGBT parameter	Symbol [units]	Property file tag
On-state collector current vs. collector-to-emitter voltage	$I_C [A]$ vs. $V_{CE} [V]$	<Output characteristic>
Freewheeling diode current vs. diode voltage	$I_f [A]$ vs. $V_f [V]$	<Diode characteristic>
Turn-on IGBT switching energy losses vs. on-state collector current	$E_{on} [mJ]$ vs. $I_C [V]$	<Switch-on loss>
Turn-off IGBT switching energy losses vs. on-state collector current	$E_{off} [mJ]$ vs. $I_C [V]$	<Switch-off loss>
Freewheeling diode switching energy losses vs. diode current *	$E_{rec} [mJ]$ vs. $I_f [V]$	<Diode switch-on loss>
Freewheeling diode reverse recovery charge *	$Q_{rr} [nC]$	<Diode reverse recovery charge>

* These parameters are not necessarily required for the property file. Only one of these parameters <Diode switch-on loss> or <Diode reverse recovery charge> can be specified in the property file to calculate the freewheeling diode switching losses. If none of these parameters specified in the property file, the freewheeling diode switching losses will not be included into the inverter losses calculation (diode switching losses are usually much less comparing with the transistor switching losses).

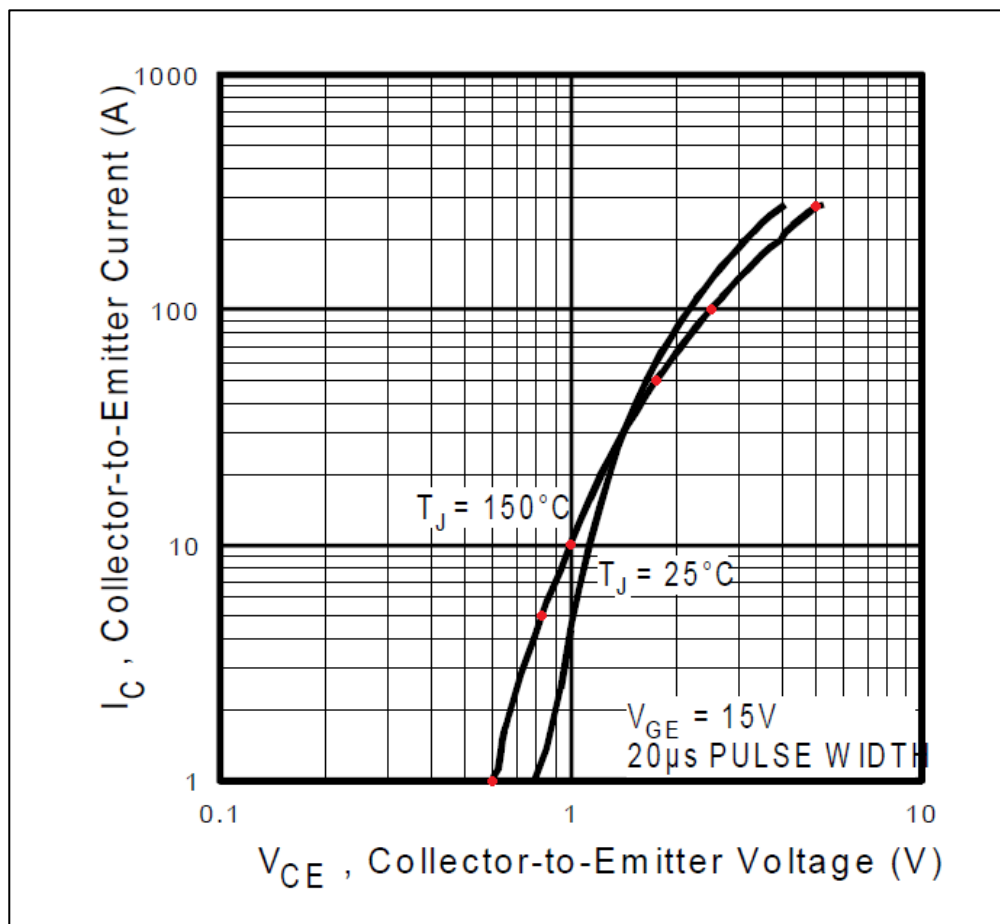
The process of creating the IGBT property file is demonstrated on the example of the IRG4PC50FD and IXGH24N60AU1 transistors, the corresponding property files can be found in [User data directory]\Materials\Transistor\ and the corresponding datasheets can be found in [User data directory]\Materials\Transistor\datasheet\.

On-state collector current vs. collector-to-emitter voltage characteristic $I_C = f(V_{CE})$ and the freewheeling diode current vs. diode voltage characteristic $I_f = f(V_f)$ are read from the datasheet diagrams as shown in Figure 4.30 and Figure 4.31, respectively. At least two points are required. Web-based tool <https://automeris.io/WebPlotDigitizer> can be used to extract data from the diagram.

Turn-on and turn-off IGBT switching energy losses can also be read from the datasheet diagrams as shown in Figure 4.32 for the FF450R12KT4 transistor. Some manufacturers do not provide turn-on and turn-off losses diagrams in the datasheet though the required information can still be extracted from the datasheet data. One example is shown in Figure 4.33 for the IRG4PC50FD transistor where only the total IGBT switching losses vs. collector current diagram is provided in the datasheet. Assuming that the correlation between the turn-on and turn-off energy losses (E_{on}/E_{off} ratio) does not depend on the collector current and the junction temperature and using the specific turn-on and turn-off switching losses values from the datasheet (see Figure 4.33) the turn-on and turn-off IGBT losses vs. collector current characteristics have been restored and recorded in the property file. Another example is shown in Figure 4.34 for the IXGH24N60AU1 transistor where only the turn-off losses E_{off} diagram is available

from the datasheet. The turn-on IGBT losses E_{on} vs. collector current characteristic has been restored using the $E_{on}/E_{off} = 0.8/2.3$ ratio from the datasheet data (see Figure 4.34).

If the freewheeling diode switching energy losses vs. diode current diagram is provided in the datasheet (see Figure 4.35 for the FF450R12KT4 transistor) when this characteristic can be directly specified in the IGBT property file for the diode switching calculation. Otherwise, the diode reverse recovery charge from the datasheet can be used instead as shown in Figure 4.36 for the IRG4PC50FD transistor. If none of these parameters specified in the property file, the freewheeling diode switching losses will not be included into the inverter losses calculation (diode switching losses are usually much less comparing with the transistor switching losses).



```
<Output characteristic> @ Tj=150*C, Vge=15V
% Vce[V]    Ic[A]
0.60        1
0.83        5
0.99        10
1.75        50
2.50        103
5.00        277
```

Figure 4.30. On-state collector current vs. collector-to-emitter voltage diagram from the datasheet (IRG4PC50FD) and the corresponding part of the IGBT property file.

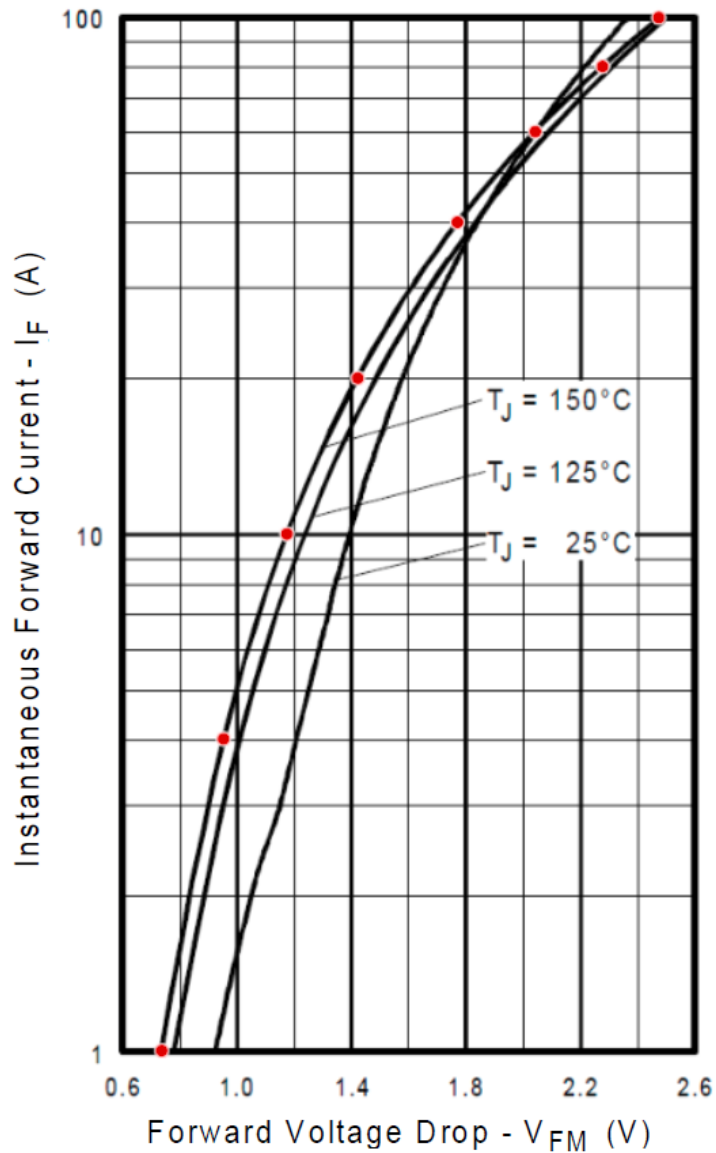


Fig. 13 - Maximum Forward Voltage Drop vs. Instantaneous Forward Current

<Diode characteristic>		@ $T_J=150^\circ\text{C}$
% Vf[V]	If[A]	
0.73	1	
0.95	4	
1.17	10	
1.42	20	
1.77	40	
2.04	60	
2.28	80	
2.47	100	

Figure 4.31. Freewheeling diode current vs. diode voltage diagram from the datasheet (IRG4PC50FD) and the corresponding part of the IGBT property file.

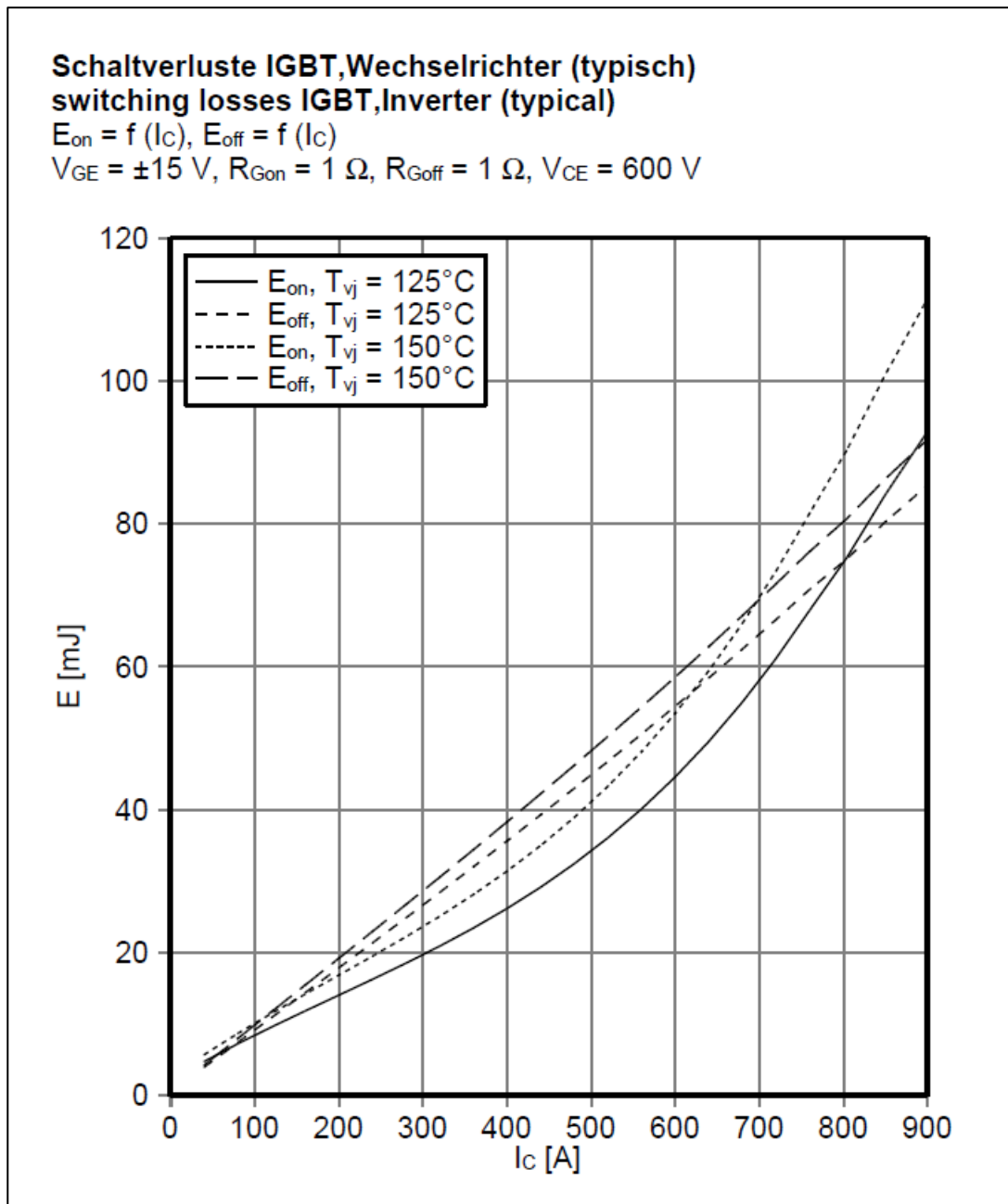
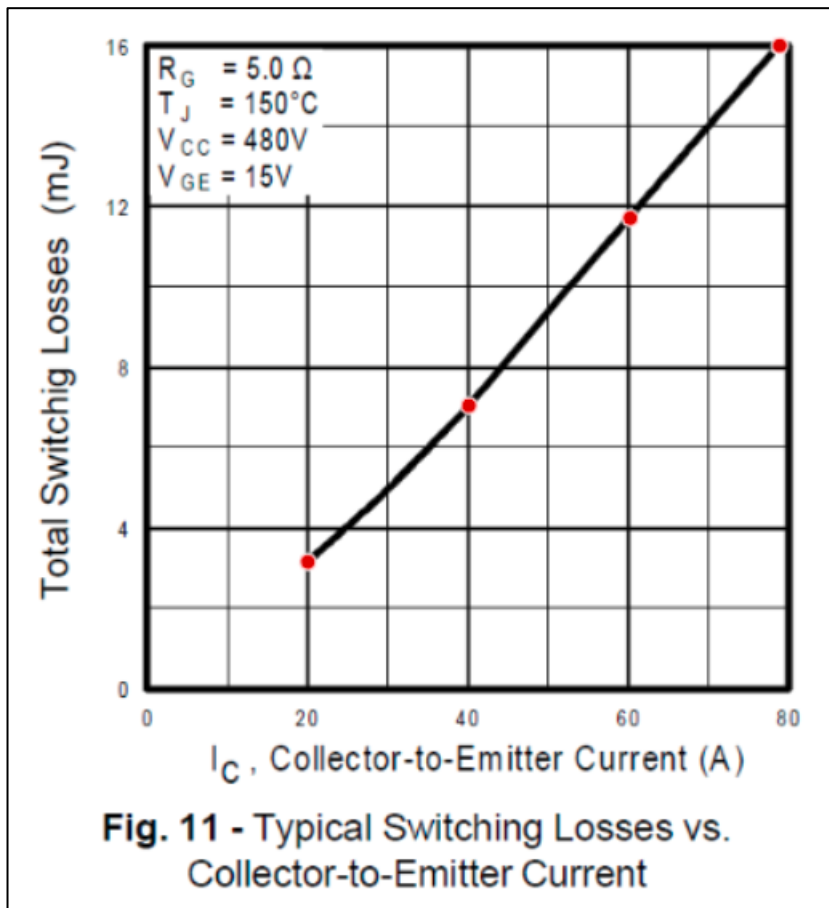


Figure 4.32. Turn-on and turn-off IGBT switching energy losses vs. collector current diagrams from the datasheet (FF450R12KT4).


Switching Characteristics @ $T_J = 25^\circ\text{C}$ (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
Q_g	Total Gate Charge (turn-on)	----	190	290	nC	$I_C = 39\text{A}$ $V_{CC} = 400\text{V}$ $V_{GE} = 15\text{V}$ See Fig. 8
Q_{ge}	Gate - Emitter Charge (turn-on)	----	28	42		
Q_{gc}	Gate - Collector Charge (turn-on)	----	65	97		
$t_{d(on)}$	Turn-On Delay Time	----	55	----	ns	$T_J = 25^\circ\text{C}$ $I_C = 39\text{A}$, $V_{CC} = 480\text{V}$ $V_{GE} = 15\text{V}$, $R_G = 5.0\Omega$ Energy losses include "tail" and diode reverse recovery. See Fig. 9, 10, 11, 18
t_r	Rise Time	----	25	----		
$t_{d(off)}$	Turn-Off Delay Time	----	240	360		
t_f	Fall Time	----	140	210		
E_{on}	Turn-On Switching Loss	----	1.5	----	mJ	
E_{off}	Turn-Off Switching Loss	----	2.4	----		
E_{ts}	Total Switching Loss	----	3.9	5.0		

<Switch-on loss> @ $T_J = 150^\circ\text{C}$

% I_C [A] E_{on} [mJ]

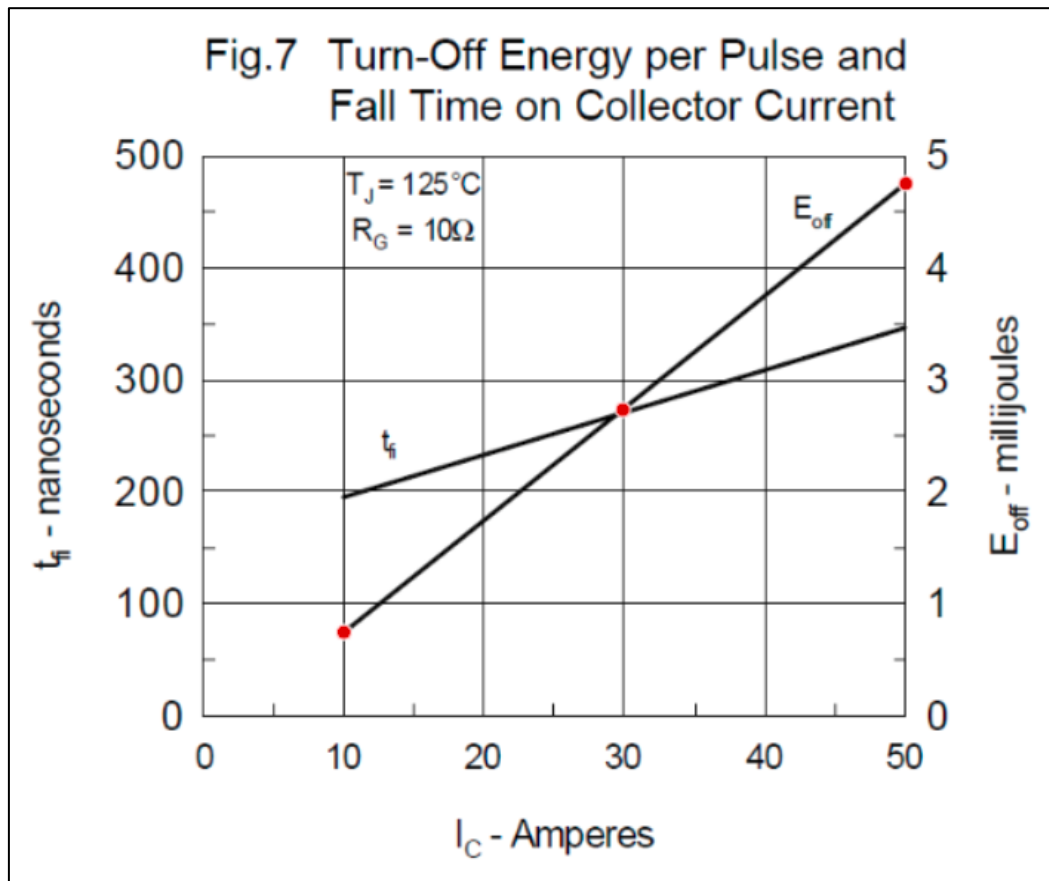
0	0
19.9	1.2
40.0	2.7
60.0	4.5
78.7	6.2

<Switch-off loss> @ $T_J = 150^\circ\text{C}$

% I_C [A] E_{off} [mJ]

0	0
19.9	1.9
40.0	4.3
60.0	7.2
78.7	9.8

Figure 4.33. Example (1) of extracting the turn-on and turn-off IGBT switching energy losses vs. collector current from the datasheet (IRG4PC50FD) and corresponding part of the IGBT property file.



$t_{d(on)}$ t_{ri} E_{on} $t_{d(off)}$ t_{fi} E_{off}	Inductive load, $T_J = 125^\circ\text{C}$		25	ns
	$I_C = I_{C90}, V_{GE} = 15\text{ V}, L = 100\text{ }\mu\text{H}$		15	ns
	$V_{CE} = 0.8 V_{CES}, R_G = R_{off} = 10\text{ }\Omega$		0.8	mJ
	Remarks: Switching times may increase for V_{CE} (Clamp) $> 0.8 \cdot V_{CES}$, higher T_J or increased R_G		250	ns
			400	ns
			2.3	mJ

<Switch-on loss> @ $T_J=150^\circ\text{C}$	
% I_C [A]	E_{on} [mJ]
10	0.26
30	0.95
50	1.66
<Switch-off loss> @ $T_J=150^\circ\text{C}$	
% I_C [A]	E_{off} [mJ]
10	0.75
30	2.74
50	4.76

Figure 4.34. Example (2) of extracting the turn-on and turn-off IGBT switching energy losses vs. collector current from the datasheet (IXGH24N60AU1) and corresponding part of the IGBT property file.

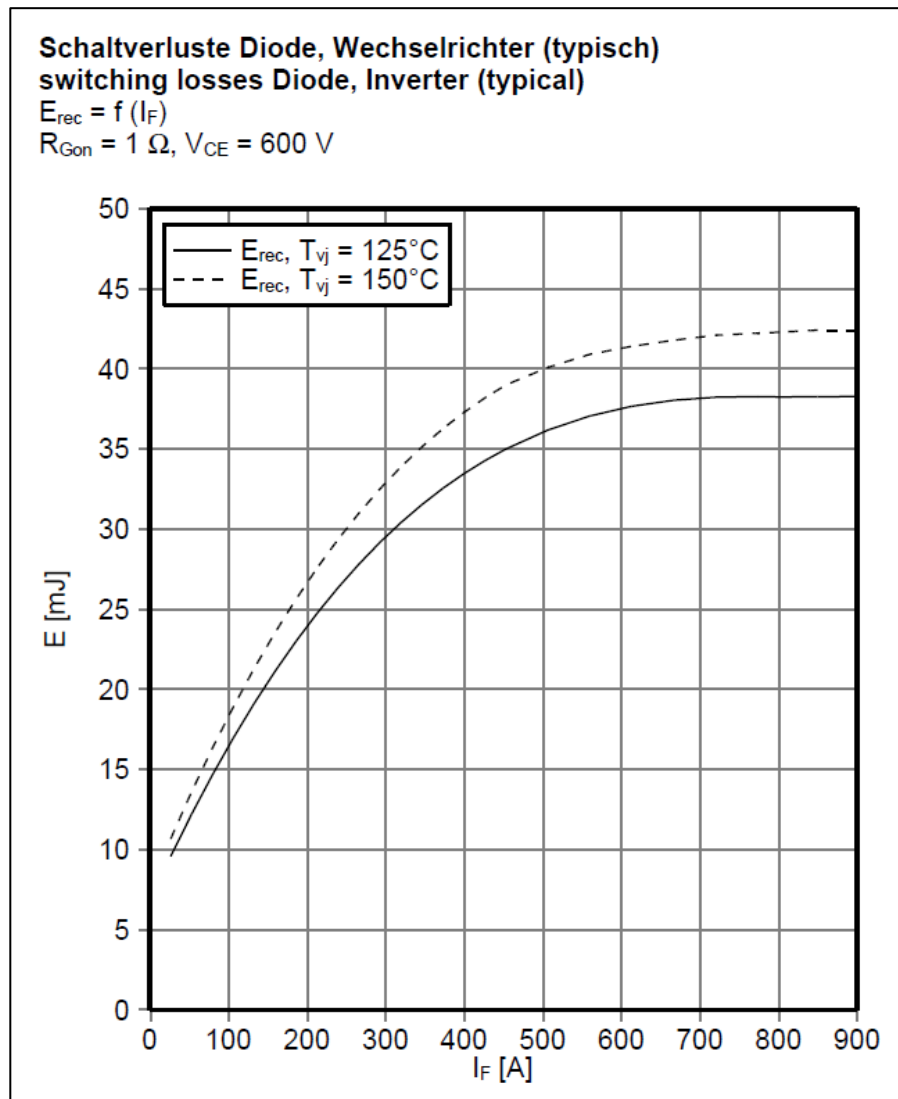


Figure 4.35. Freewheeling diode switching energy losses vs. diode current diagrams from the datasheet (FF450R12KT4).

t_{rr}	Diode Reverse Recovery Time	----	50	75	ns	$T_J = 25^\circ C$	See Fig.	$I_F = 25A$ $V_R = 200V$ $di/dt \ 200A/\mu s$
		----	105	160		$T_J = 125^\circ C$	14	
I_{rr}	Diode Peak Reverse Recovery Current	----	4.5	10	A	$T_J = 25^\circ C$	See Fig.	
		----	8.0	15		$T_J = 125^\circ C$	15	
Q_{rr}	Diode Reverse Recovery Charge	----	112	375	nC	$T_J = 25^\circ C$	See Fig.	
		----	420	1200		$T_J = 125^\circ C$	16	
$di_{(rec)M}/dt$	Diode Peak Rate of Fall of Recovery During t_b	----	250	----	A/ μs	$T_J = 25^\circ C$	See Fig.	
		----	160	----		$T_J = 125^\circ C$	17	

<Diode reverse recovery charge>
1200 % [nC]

Figure 4.36. Reading the diode reverse recovery charge from the datasheet (IRG4PC50FD) and the corresponding part of the IGBT property file. Maximum value of the reverse recovery charge is used to represent the worst case of the diode switching losses.

5. MAGNETOSTATIC FINITE ELEMENT ANALYSIS

Magnetostatic FE Analysis allows the user to estimate motor parameters like voltage, current, back-EMF, power, torque, power factor, efficiency, losses and their waveforms as well as air gap and cross-section distribution plots of various quantities such as flux density, permeability, current density and etc. assuming an ideal sinusoidal or trapezoidal current waveforms.

5.1. Running Magnetostatic FE Analysis.

The view of the main window when **Magnetostatic FE Analysis** is chosen is shown in Figure 5.1. **Magnetostatic FE Analysis** consists of a series of finite element simulations (series of time steps) over one electrical period (or specified time interval) with predefined currents and rotor positions defined by the rotor speed. The current waveform is defined by the **Current waveform** field and can be either sinusoidal or trapezoidal. There are several current input methods defined by the **Current input method** field which also depend on the current waveform. For the sinusoidal current waveform, the following current input methods are available: **RMS supply current**, **Peak supply current** and **RMS current density**. When the **RMS supply current** input method is chosen the phase current in each winding will depend on the **RMS supply current** field value I_s , on the stator winding connection (star or delta) and on the connection between winding layouts if there are two winding layouts (two stators, each of them having its own winding, or one stator having two windings), as specified by the **Layouts connection** pop-up menu of **Winding Editor** (**I-2** selection from the **Layouts connection** pop-up menu specifies that two winding layouts are connected in series, whereas **I//2** selection specifies that two winding layouts are connected in parallel). The phase current calculation in each winding for all possible cases is given in Table 5.1. Examples of stator winding configurations for star and delta connections and for different numbers of winding layouts are shown in Figure 4.14.

Table 5.1. Phase current per one winding layout depending on the winding connection and connection between layouts.

	One winding	Two winding layouts connected in series (I-2)	Two winding layouts connected in parallel (I//2)
Star connection	I_s	I_s	$I_s/2$
Delta connection	$I_s/\sqrt{3}$	$I_s/\sqrt{3}$	$I_s/\sqrt{3}/2$

I_s - RMS supply current

Since the supply current corresponds to a line current, for the star connection the phase current is equal to the supply current divided by the number of winding layouts connected in parallel, for delta connection the phase current is equal to the supply current divided by the number of winding layouts connected in parallel and divided by $\sqrt{3}$ (see table 5.1).

Peak supply current input method is similar to the previous one, but peak value of supply current is used instead. When the **RMS current density** input method is chosen the RMS current density in stator slots can be directly defined.

For the trapezoidal current waveform, the following current input methods are available: **RMS phase current**, **DC supply current** and **RMS current density**. When the **RMS phase current** input method is chosen the phase current can be directly defined. When the **DC supply current** input method is chosen the phase current will depend on the **DC supply current** field value, stator winding connection, number of layouts connected in parallel and switch duty cycle. Switch duty cycle (120 or 180 electrical degrees) is defined in the **Drive Settings** window when six-step drive type is chosen (see section 4.4). Figure 5.2 shows an example of trapezoidal current waveforms for star and delta connection and 120 degrees switch duty cycle. Be aware that the real current waveform for the six-step drive can be significantly different from those used in **Magnetostatic FE Analysis**. When the **RMS current density** input method is chosen the RMS current density in stator slots can be directly defined.

The **Advance angle** field value defines the timing of the current commutation in relation to the rotor position assuming ideal current commutation and varies between 0 and 360 electrical degrees. The magnetostatic FEA simulation is performed for fixed rotor speed defined by the **Mechanical speed** field. **Number of points** defines the number of time steps per one electrical period or per the selected time interval (depending on the **Simulation time** pop-up menu item) the FEA simulation is run for. Several options for the number of points are available from the **Simulation setup** pop-up menu depending on the required accuracy. The number of points can be automatically determined based on the number of points (time steps) per one period of cogging torque (choose between **Multiple points (2 points per cogging) - low accuracy**, **Multiple points (4 points per cogging) - medium accuracy**, **Multiple points (8 points per cogging) - high accuracy**). To consider the effect of the cogging torque on the motor performance, at least two points per cogging torque period are required. Number of periods of cogging torque waveform per rotor revolution equals to the least common multiple (LCM) of number of slots (N_s) and number of poles (N_p) so the cogging torque period can be determined as $T_{cogging} = N_p / (2f_s \cdot LCM(N_s, N_p))$, where f_s is supply frequency.



Besides the multiple points simulation setup, for the sinusoidal current waveform the **Single point (FEA + D-Q based) – fastest** item from the **Simulation setup** pop-up menu is also available which means the machine parameters are determined from the magnetostatic FEA simulation for one rotor position with the help of the D-Q representation of the machine (see section 2.5). In this case the iron losses and magnet losses are not calculated, and time plots are not available. Finally, the number of points can be directly specified if the corresponding item is chosen from the **Simulation setup** pop-up menu.

By default, the magnetostatic FEA simulation is run for one electrical period, however the specific time interval can be specified if **Define time interval** is chosen from the **Simulation time** pop-up menu.

Button '**<-rated**' to the right of some fields allows you to set up the corresponding parameter to its rated value specified in the **Rated Data** window.

Solver type specifies either linear or nonlinear FEA is used. Using linear solver is recommended only for testing purposes. **Convergence tolerance** specifies the accuracy of FEA solution as defined by Exp. 2.5. The cogging torque is computed with zero stator current and if **RMS supply current (DC supply current)** field value is not zero when it will require one additional FEA simulation for each time step to compute cogging torque. Be aware of this fact when checking **Compute cogging torque**.

Save each field solution checkbox enables (if checked) to store additional data of each FEA solution such as magnetic vector potential distribution and permeability distribution values. These data are not stored by default because of large amount of hard drive space required. Data for each time step are saved in a separate file so the number of files saved is equal to the **Number of points** field value.

To start the analysis, click the **Start magnetostatic simulation** button  on the MotorXP-AFM main window toolbar. To stop the analysis, click the **Stop simulation** button  which is to the right of the **Start magnetostatic simulation** button. If the analysis is stopped all the simulation data of the running analysis will be lost.

Once the simulation is complete the results are displayed at the bottom of the MotorXP-AFM main window. The results are divided into four groups: **General Results**, **Machine Constants**, **Flux Density Levels** and **Inductances**, and available from the corresponding pup-up menu as shown in Figure 5.1.

MotorXP-AFM (Pro) - C:\Users\vepco\Documents\MotorXP-AFM\SimFiles\ex... — □ ×

File Desktop About Help Licensing

Project file name: exampleSRS.mxa Activated

D-Q Analysis **Finite Element Analysis**

Steady State Dynamic **Magnetostatic** Dynamic

Magnetostatic Finite Element Analysis

Solver type: Nonlinear Convergence tolerance: 0.001

Current waveform: Sinusoidal Advance angle: 0 (el.deg.)

Current input method: RMS supply current Mechanical speed: 6000 <-rated RPM

RMS supply current: 5.5 <-rated A

Simulation setup: Multiple points (8 points per cogging) - high accuracy Number of points: 96

Simulation time: One electrical period

☒ Compute cogging torque

☒ Save each step solution in folder: C:\Users\vepco\Documents\MotorXP-AFM\SimFiles\exampl...

Results:

General Results

Rotor speed:	6000	RPM	RMS phase back-EMF:	190.503	V
Advance angle:	0	(el.deg.)	Input electrical power:	1886.03	W
Supply frequency:	500	Hz	Output mechanical power:	1784.36	W
RMS current (supply phase):	5.5 3.17543	A	Efficiency:	90.7325	%
RMS voltage (supply phase):	114.888 198.957	V	Power factor:	0.975799	
Total torque:	2.8399	N·m	Stator winding loss:	65.098	W
Reluctance torque:	-0.0002871	N·m	Total iron core loss:	57.7044	W
Magnet torque:	2.84018	N·m	Eddy current iron core loss:	25.219	W
Torque ripple:	9.17246	%	Hysteresis iron core loss:	32.4855	W
RMS current density:	4.9738	A/mm ²	Magnet eddy current loss:	59.4546	W
Average d-axis phase	-6.32568e-17	A	Other eddy current loss:	0	W
Average q-axis phase	4.49073	A	Max. demag. field:	671461	A/m
Average d-axis phase	-27.8091	V	Max. demag. field (% of Hcj):	73.7363	%
Average q-axis phase	279.989	V	Discretization error:	1.9393	%

Results:

Machine Constants

Rotor speed:	6000	RPM
Advance angle:	0	(el.deg.)
Supply frequency:	500	Hz
RMS current (supply phase):	5.5 3.17543	A
Velocity constant (Kv) in [RPM/V]:	20.9419	RPM/V
Velocity constant (Kv) in [rad/(V·s)]:	2.19303	rad/(V·s)
Back-EMF constant (Ke) in [V/RPM]:	0.0477511	V/RPM
Back-EMF constant (Ke) in [V·s/rad]:	0.455989	V·s/rad
Torque constant (Kt):	0.894335	N·m/A
Motor constant (Km):	0.35198	N·m/sqrt(W)

Results:

Flux Density Levels

Rotor speed:	6000	RPM
Advance angle:	0	(el.deg.)
Supply frequency:	500	Hz
RMS current (supply phase):	5.5 3.17543	A
Average airgap flux density:	0.610876	T
Maximum airgap flux density:	1.23192	T
Peak of stator tooth average flux density:	1.75181	T
Peak of stator back iron average flux density:	1.40734	T
Peak of rotor back iron average flux density:	0.649914	T

Results:

Inductances

Rotor speed:	6000	RPM
Advance angle:	0	(el.deg.)
Supply frequency:	500	Hz
RMS current (supply phase):	5.5 3.17543	A
D-axis inductance (Ld):	0.0757647	mH
Q-axis inductance (Lq):	0.0772449	mH
Phase self inductance (Lself):	0.0699321	mH
Phase mutual inductance (Lmutual):	-0.00657265	mH

Figure 5.1. MotorXP-AFM main window for Magnetostatic FE Analysis with different results shown.

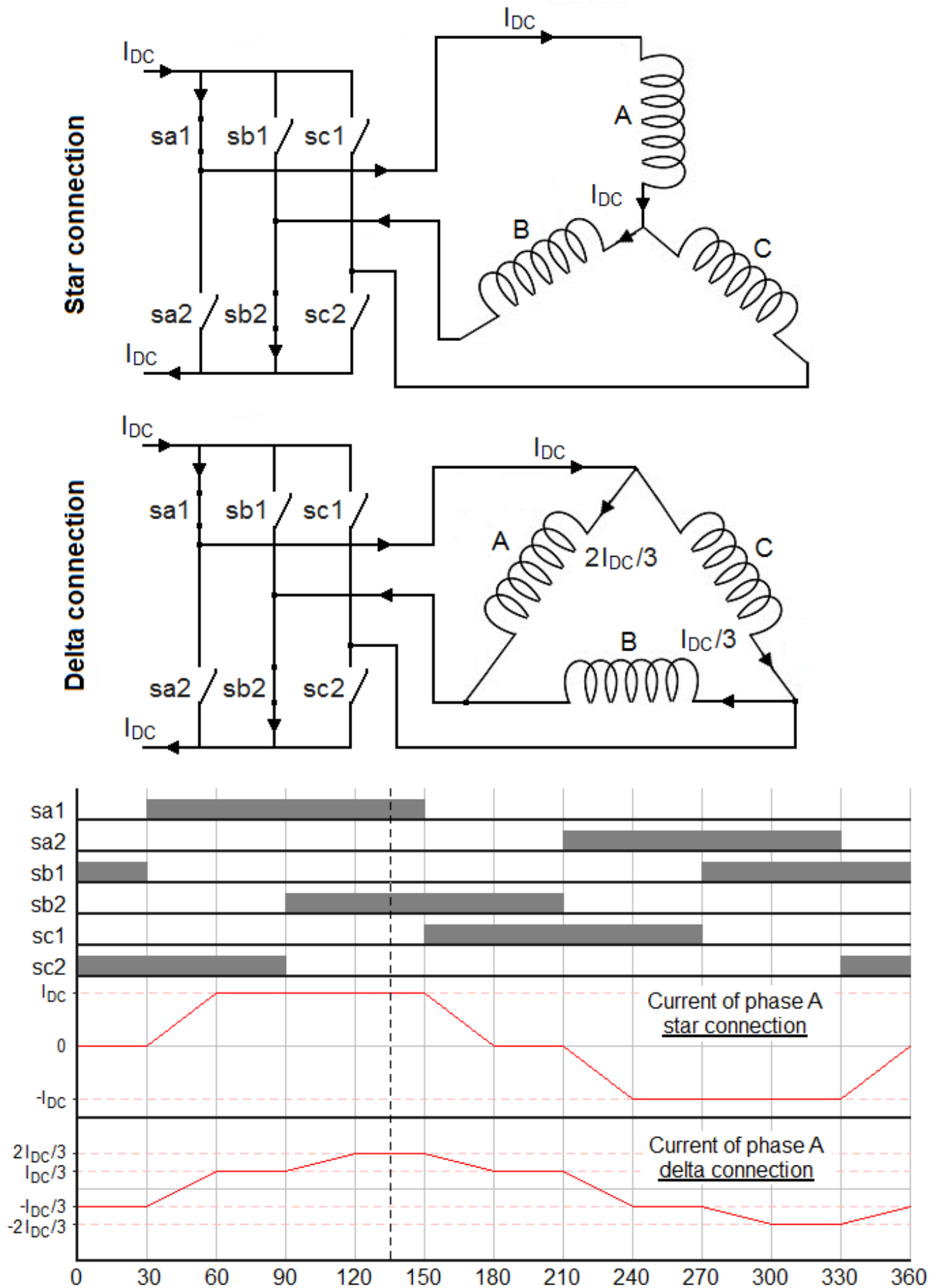


Figure 5.2. Example of trapezoidal current waveforms and distribution for star and delta connection (for one winding layout).

5.2. Viewing Magnetostatic FE Analysis results.

View of the **Plot Wizard** window when **Magnetostatic FE Analysis** is chosen is shown in Figure 5.3.

Several plot types are available for **Magnetostatic FE Analysis**:

- time-series data plot and frequency spectrum of the time-series data;
- air gap distribution plot and frequency spectrum of the air gap distribution;
- cross-section distribution plot;
- animation.

Plot Wizard also allows saving waveforms to a CSV-file (text file with comma-separated values) or to a Microsoft® Excel® spreadsheet file (Figure 5.3).

5.2.1. Time plots.

Time-series data plots are available from the **Time plot** panel of the **Plot Wizard** window. It is organized as a standard MATLAB plotting construction consisting of a set of `subplot` and `plot` functions. **Subplot** checkboxes allow you to control the number of axes or rectangular panes displayed within a current figure window. The corresponding axes are activated or deactivated by a mouse click within a cell of the **Subplot** column. When the subplot is activated, the **change** button allows you to choose quantities to be plotted into the selected axes. Quantities can be chosen from the dialog shown in Figure 5.4. Use Ctrl key to select several quantities. The list of available quantities is given below:

- Stator phase current (phase A, B, C);
- Phase current (d-axis, q-axis);
- Phase voltage (phase A, B, C);
- Phase voltage (d-axis, q-axis);
- Effective advance angle;
- Back-EMF (phase A, B, C);
- Back-EMF (d-axis, q-axis);
- Input electrical power;
- Mechanical power on the rotor shaft;
- Stator winding losses;
- Eddy current magnet losses;
- Electromagnetic torque by Maxwell stress tensor;
- Electromagnetic torque by virtual work method;
- Electromagnetic torque by flux linkage and current;
- Magnet torque (by Maxwell stress tensor);
- Reluctance torque (by Maxwell stress tensor);
- Cogging torque (by Maxwell stress tensor);

- Torque in each air gap (by Maxwell stress tensor);
- Flux linkage (phase A, B, C);
- Flux linkage, (d-axis, q-axis);
- Electromagnetic normal force on each rotor (by Maxwell stress tensor);
- Electromagnetic normal force on each stator (by Maxwell stress tensor);

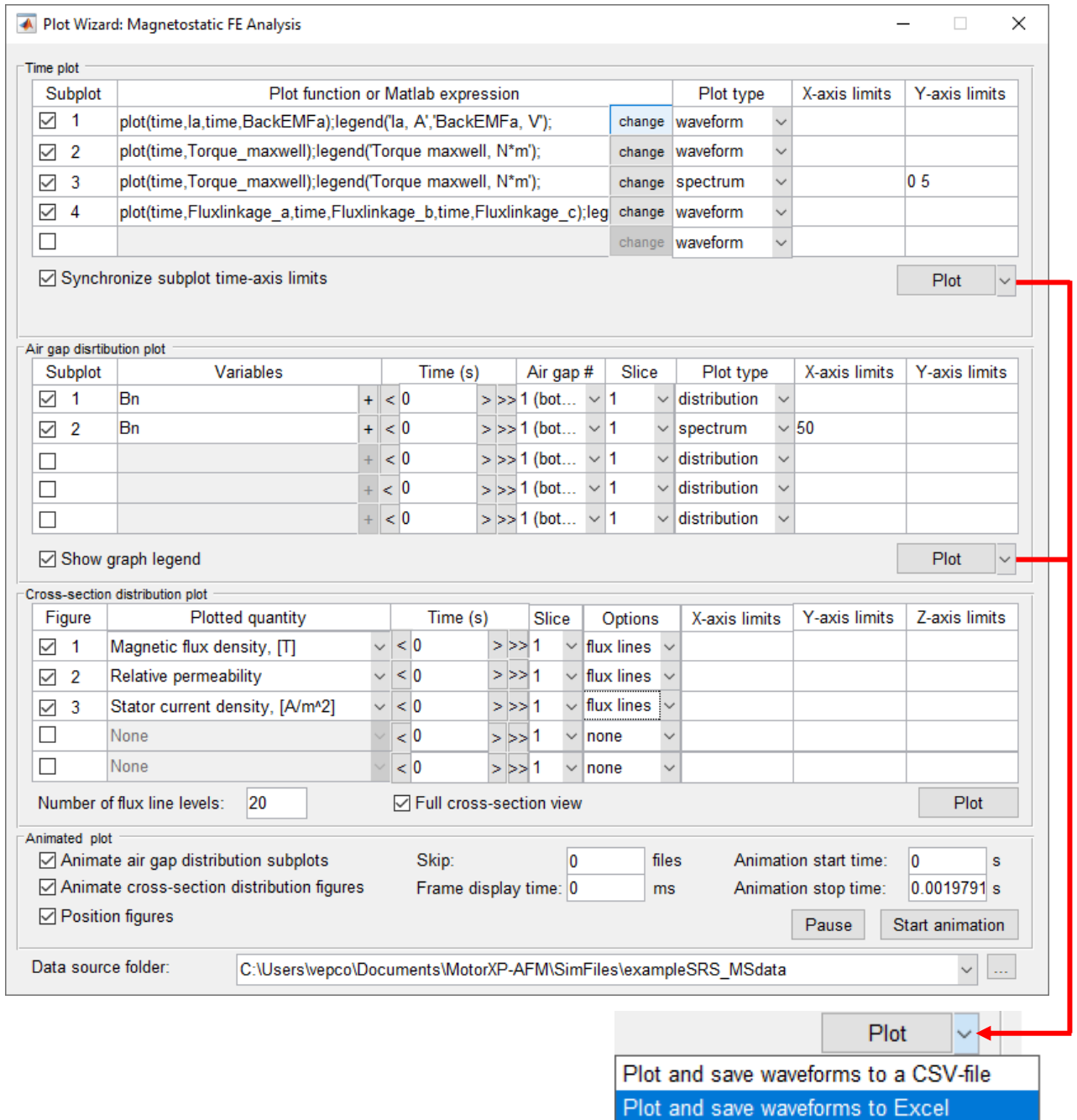


Figure 5.3. Magnetostatic FE Analysis Plot Wizard window and export data to a file options.

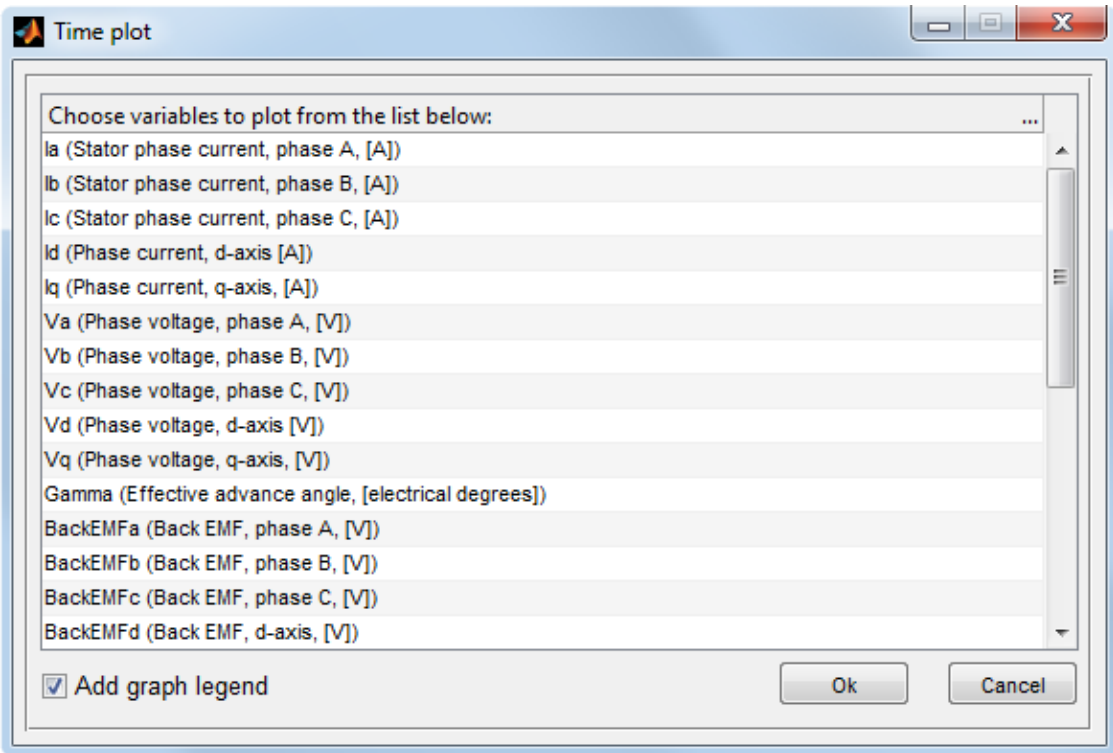


Figure 5.4. Choosing quantities to be plotted.

By clicking the **OK** button of the dialog (Figure 5.4), the MATLAB-expression is constructed to plot the selected quantities appearing in the corresponding line as shown in Figure 5.5 for plotting phase A stator current and back-EMF waveforms (first line), torque waveform (second line) and torque spectrum (third line). The fourth line is not active and, therefore, will not be plotted. When the **Plot** button is clicked, the figure window with plots of the selected quantities will appear (see Figure 5.6).

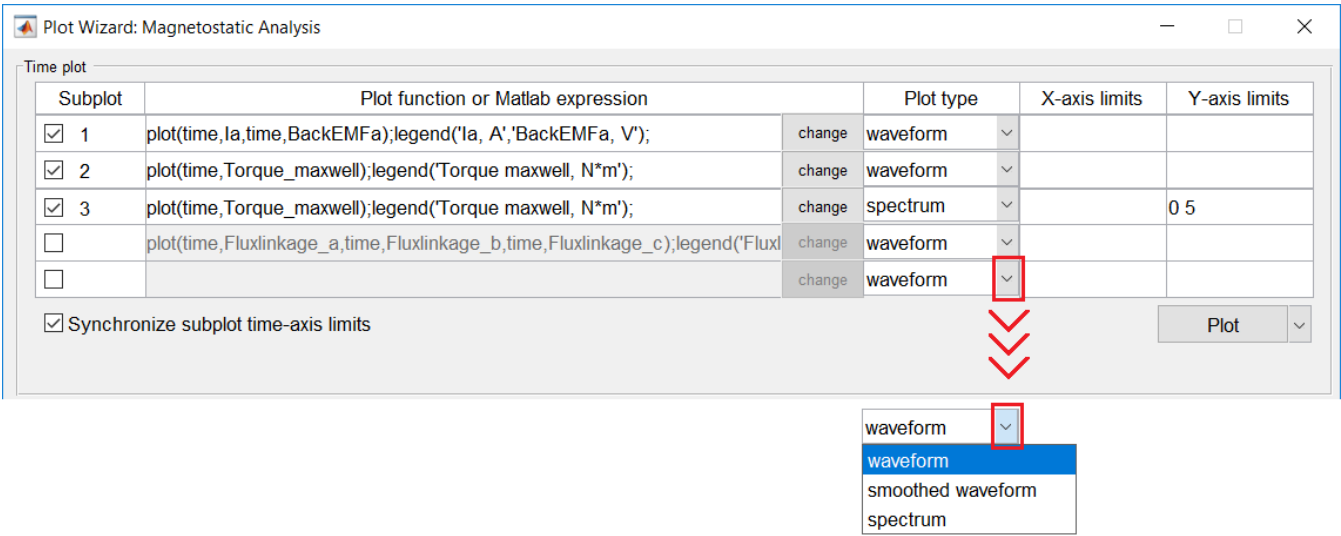


Figure 5.5. Example of using **Plot Wizard** for creating time plots.

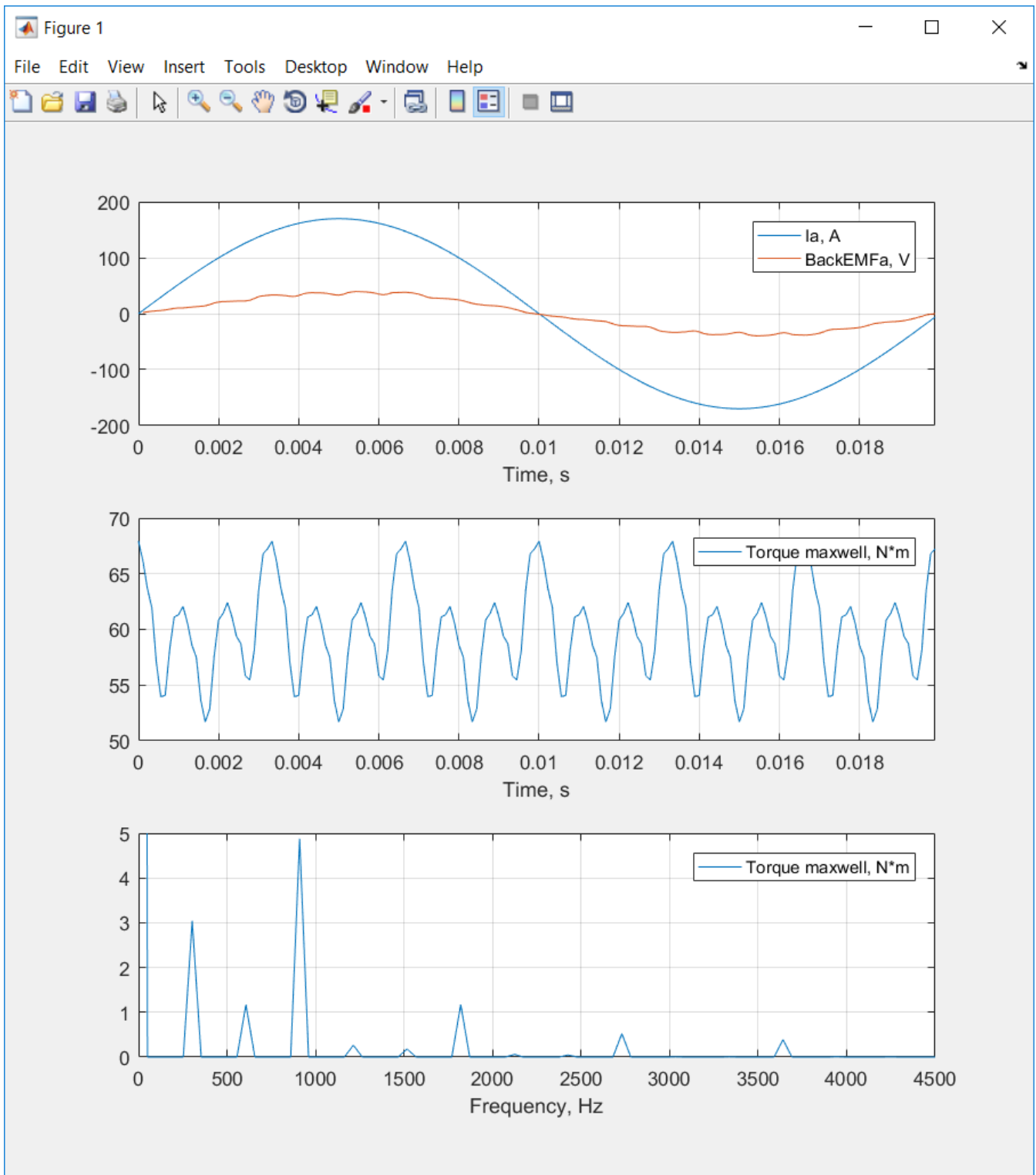


Figure 5.6. Figure window with plots of the selected quantities.

As it is seen, the plotting expression consists of the `plot` function with selected variables used as input arguments. If the **Add graph legend** checkbox of the dialog is checked, the `legend` function is added to the plotting expression so the graph legend of the corresponding axes will be shown. All plotting expressions are editable, so you can use any MATLAB plotting options and functions to change the way the plots appear on the screen. You can also change `time` variable to any other variable you would like

to use as an input argument of the `plot` function. There is also a list of additional variables which can be used within a plotting expression (see section 9.3).

There are several plotting options available from the **Plot type** pop-up menu: *waveform*, *smoothed waveform* and *spectrum* (see Figure 5.5). Frequency spectrum is calculated over one electrical period.

X-axis limits and **Y-axis limits** fields of the **Time plot** panel allow you to set the x-axis and y-axis limits, respectively, to the specified values. Two limit values within a cell are separated by a space, comma ‘,’ or semicolon ‘;’. If the cell is empty, the limits of the corresponding axis will be chosen automatically. If the **Synchronize subplot time-axis limits** checkbox is checked, all subplots of the figure will have identical limits along the time-axis when you zoom or pan one of the subplots of the figure.

5.2.2. Air gap distribution plots.

Air gap distribution plots allow you to view the distribution of the particular quantity over the machine’s air gap in the selected cylindrical cross-section (slice) as well as its frequency spectrum. It is available from the **Air gap distribution plot** panel. **Subplot** checkboxes allow you to control the number of axes or rectangular panes displayed within a current figure window. The corresponding axes are activated or deactivated by a mouse click within a cell of the **Subplot** column. When the subplot is activated, the “+” button allows you to choose quantities to be plotted from the dialog shown in Figure 5.7. Use Ctrl key to select several quantities.

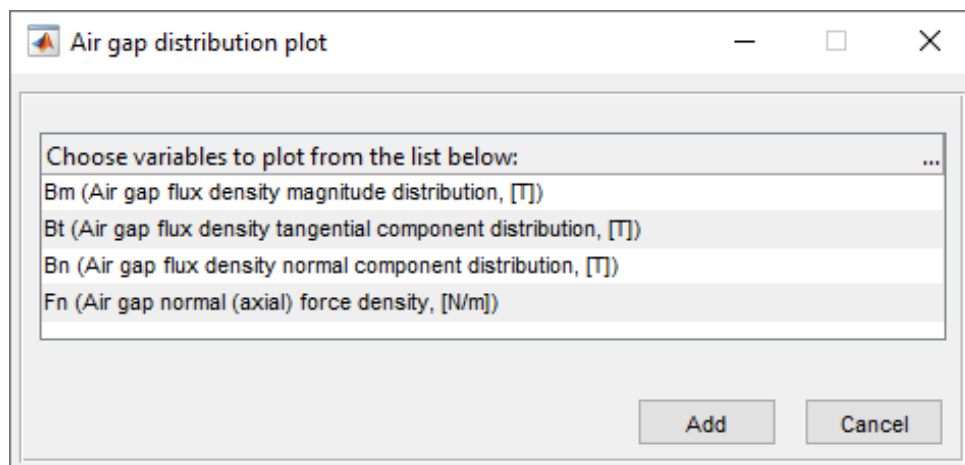


Figure 5.7. Choosing quantities to be plotted.

The following quantities are listed in the dialog of Figure 5.7:

Quantity	Comments
Bm (Air gap flux density magnitude distribution, [T])	Defined as $B_m = \sqrt{B_t^2 + B_n^2}$ where B_n and B_t – normal and tangential components of the magnetic flux density in the center of the air gap, as shown in Figure 2.4.
Bt (Air gap flux density tangential component distribution, [T])	B_n and B_t – normal and tangential components of the magnetic flux density in the center of the air gap, as shown in Figure 2.4.
Bn (Air gap flux density normal component distribution, [T])	
Fn (Air gap normal (radial) force distribution, [N])	According to Maxwell stress tensor method the normal force in each point of the round path can be expressed as the following: $F_n = -\frac{B_n^2 - B_t^2}{2\mu_0} d \cdot l$ where l – lamination length in z direction, d – length of the path in the center of the air gap between two consecutive points, μ_0 – permeability of the free space, B_n and B_t – normal and tangential components of the magnetic flux density in the center of the air gap, as shown in Figure 2.4.

By clicking the **Add** button of the dialog (Figure 5.7), the selected quantities are added to the corresponding line separated by commas as shown in Figure 5.8. Each line of the **Variables** column is editable, so you can use MATLAB arithmetic operations to obtain desired plots. For example, the second and third lines in Figure 5.8 produce plots of the tangential air gap force density distribution and the radial air gap force density distribution, respectively, where μ_0 – variable corresponding to the permeability of free space. According to Exp. 2.6 the tangential air gap force produces the electromagnetic torque so the plot of the electromagnetic torque distribution over an air gap can be obtained.

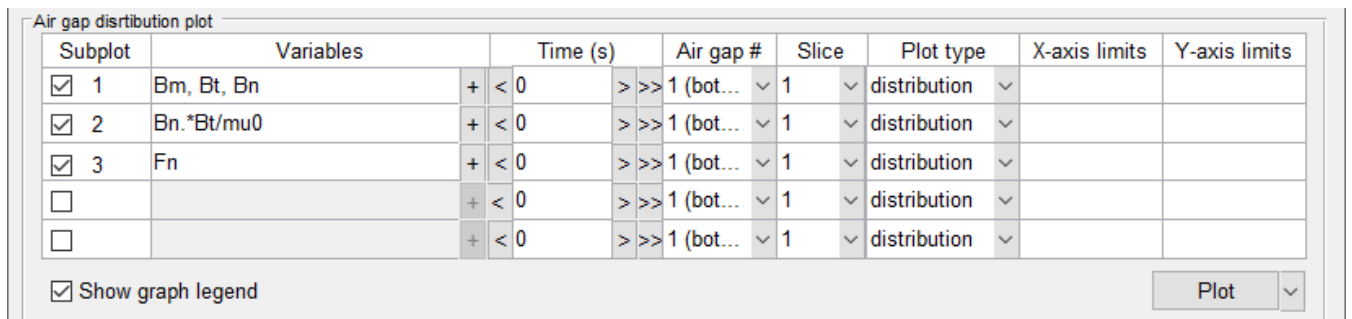


Figure 5.8. Example of using **Plot Wizard** for creating air gap plots.

Time fields of the **Air gap distribution plot** panel allow you to specify the time point which the selected quantities are plotted for. By default, zero time point is set up. Plotting for the time points other than zero is only possible, if the file containing data for the desired time point was previously saved. These data-files are being saved when **Save each field solution** is checked in the MotorXP-AFM main window. So, if you are interested in viewing the air gap distribution plots for different time points make sure that the corresponding data-files are being saved. The folder used as a source of data-files is specified in the **Data source folder** field located at the bottom of the **Plot Wizard** window (Figure 5.3). You can change it by clicking the button to the right, if needed.

X-axis limits and **Y-axis limits** fields allow you to set the x-axis and y-axis limits, respectively, to the specified values in the same way as for the **Time plot** panel. **Slice** fields allow you to choose the machine's cylindrical cross-section which the selected quantities are plotted for, if several slices are used (see section 2.4 for more details on multi-slice simulations). If the machine has several air gaps the **Air gap #** field selects the air gap number starting from the bottom of the machine which the selected quantities are plotted for.

Besides the air gap distribution, it is also possible to calculate the air gap harmonic components of the selected quantities. To obtain the frequency spectrum, select the *spectrum* item in the corresponding **Plot type** pop-up menu. An example of plotting the air gap distribution of the normal component of the magnetic flux density and its spectrum is shown in Figure 5.9. Only first 50 harmonics are shown, since the value specified in the corresponding **X-axis limits** field is 50 (if the minimum limit equals to 0, it can be omitted).

Show graph legend field allows you to choose whether the graph legend will be displayed.

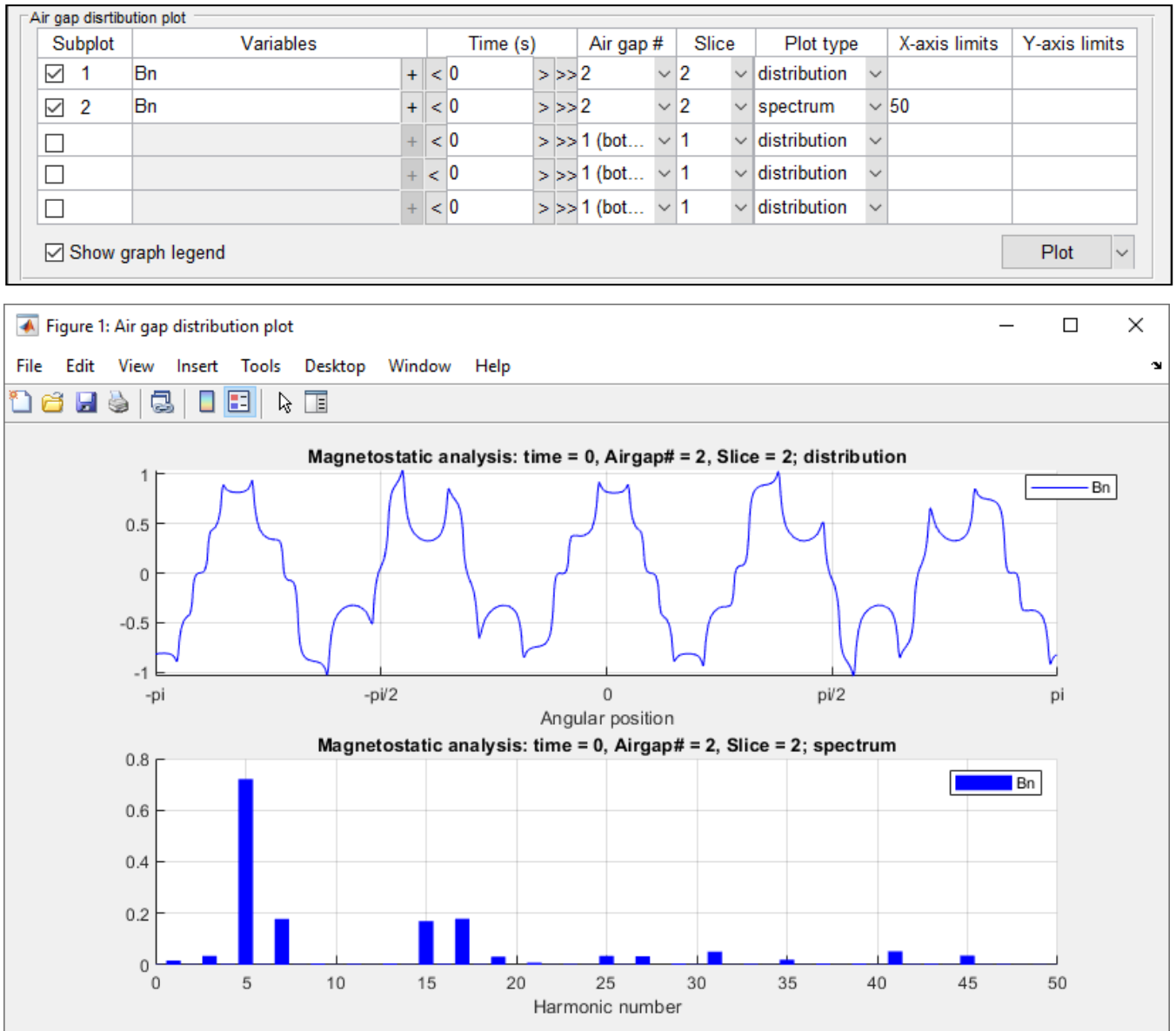


Figure 5.9. Plotting the air gap distribution of the normal flux density component and its spectrum in second air gap in second mesh slice.

5.2.3. Cross-section distribution plots.

Cross-section distribution plots are available from the **Cross-section distribution plot** panel of the **Plot Wizard** window and allow you to view the distribution of the particular quantity in the selected cylindrical cross-section (slice) of the machine. As opposed to two previous plot types, the cross-section distribution for each quantity is plotted in a separated window which contains only one axes. **Figure** checkboxes allow you to control the number of windows appearing when the **Plot** button is clicked. The corresponding figure is activated or deactivated by a mouse click within a cell of the **Figure** column. When the figure is activated, the corresponding **Plotted quantity** pop-up menu allows you to choose the quantity to be plotted. If **None** is chosen, the machine's cross-section geometry will be plotted.

The following items are available from the **Plotted quantity** pop-up menu:

- Magnetic vector potential, [T*m];
- Magnetic flux density, [T];
- Magnetic field intensity, [A/m];
- Relative permeability;
- Stator current density, [A/m²];
- Squared flux density, [T];
- Magnetic field energy density, [J/m³];
- Stator loss density, [W/m³];
- Total loss density (stator+iron), [W/m³];
- Iron loss density, [W/m³];
- Demagnetizing field, [A/m];
- Demagnetizing field, [% of H_{cj}];
- Finite element mesh.

Time fields of the **Cross-section distribution plot** panel allow you to specify the time point which the selected quantity is plotted for. By default, zero time point is set up. Plotting for the time point other than zero is only possible, if the file containing data for the desired time point was previously saved. These data-files are being saved when **Save each field solution** is checked in the MotorXP-AFM main window. So, if you are interested in viewing the cross-section distribution plots for different time points make sure that the corresponding data-files are being saved. The folder used as a source of data-files is specified in the **Data source folder** field located at the bottom of the **Plot Wizard** window (Figure 5.3). You can change it by clicking the button to the right, if needed.

Slice fields allow you to choose the machine's cylindrical cross-section which the selected quantity is plotted for, if several slices are used (see section 2.4 for more details on multi-slice simulations).

Options field allows you to show the magnetic flux lines (if *flux lines* is chosen) or magnetic flux arrows (if *flux arrows* is chosen) on the corresponding plot. Flux arrows are plotted such that the direction of the arrow indicates the direction of the flux and the size of the arrow indicates the magnitude of the flux density. **Number of flux line levels** field allows you to alter the flux lines density. If periodic/antiperiodic boundary conditions are used, by default, only part of the cross-section (as it appears in **Mesh Editor**) will be plotted. Check the **Full cross-section view** checkbox if you want to plot the whole cross-section.

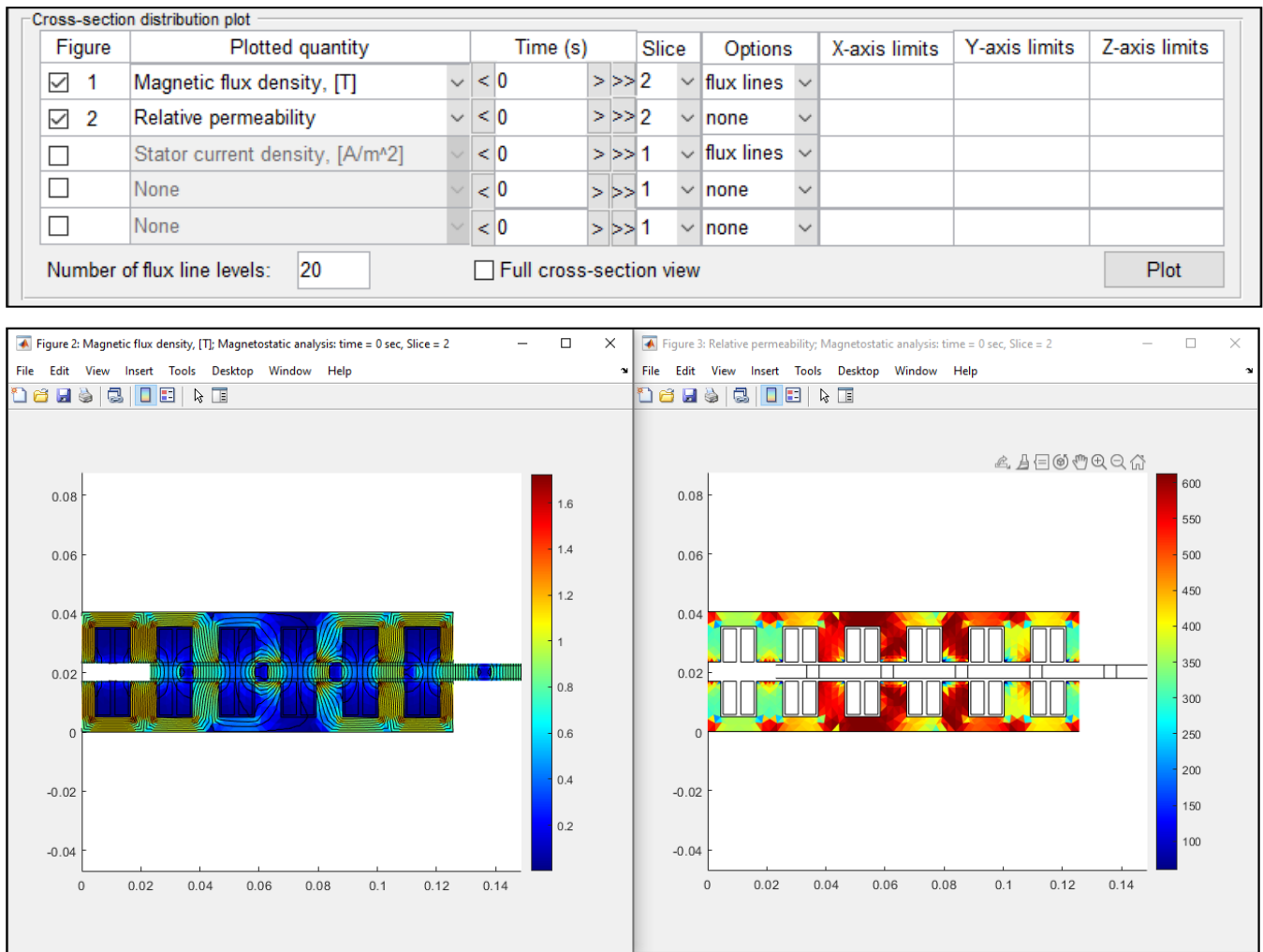


Figure 5.10. Example of using **Plot Wizard** for creating cross-section distribution plots.

X-axis limits and **Y-axis limits** fields allow you to set the x-axis and y-axis limits, respectively. If x-axis and y-axis limits are not specified, their values will be determined by the size of the machine's cross-section. **Z-axis limits** field sets the color scale limits to specified minimum and maximum values separated by a space, comma ',', or semicolon ';'. Values between z-axis limits are linearly mapped to the used color scale (colormap). Data values smaller or greater than specified z-axis limits are mapped to the minimum limit or to the maximum limit, respectively.

An example of plotting the cross-section distribution of the magnetic flux density and relative permeability is shown in Figure 5.10. When the **Plot** button is clicked, two windows will appear. As it is seen, the **flux lines** option is chosen for the magnetic flux density, so the flux lines are additionally shown. Since the third figure is not active, the Magnetic field intensity is not plotted. Note that only part of the machine's cylindrical cross-section is shown, since **Full cross-section view** is not checked.

5.2.4. Animation.

Animated plot panel is used to create animated sequences of the air gap distribution plots and/or the cross-section distribution plots (Figure 5.3). **Animate air gap distribution subplots** and **Animate cross-section distribution figures** checkboxes allow you to choose plot types to be animated. If **Animate air gap distribution subplots** is checked, all selected subplots of the **Air gap distribution plot** panel will be involved in the animation. Similarly, if **Animate cross-section distribution figures** is checked, all selected figures of the **Cross-section distribution plot** panel will be involved in the animation. If **Position figures** control is checked, the size and location on the screen for all figure windows will be determined automatically so that the windows fit best the screen size.

Skip and **Frame display time** fields allow you to specify the way how frames change each other while the animation is in progress. **Skip** field specifies the number of data-files or frames to be skipped. So, if **Skip** field is set to 0, data for each time step will be shown on the screen, if **Skip** field is set to 1, data for each second time step will be shown, if **Skip** field is set to 2 – for each third time step and so on. **Frame display time** field specifies the time each frame is displayed on the screen. Note that the least time the frame is displayed is limited by the animation function runtime. So, if the **Frame display time** field is 0, the actual time the frame is displayed on the screen will be the least possible in this case.

Animation start time and **Animation stop time** fields set up the time interval for the animation. **Data source folder** field specifies the folder with data-files to be animated. The same folder is used for animations and for air gap distribution and cross-section distribution plots. Using animation is only possible, if the files containing data for the time interval to be animated were previously saved.

To start the animation, click the **Start animation** button. The current time point, animated quantity and other information is displayed at the top of each window involved in the animation. The animation continues until the time specified in the **Animation stop time** field is reached or until all windows involved in the animation are closed. You can also pause the animation using the **Pause** button. While the animation is in progress or paused, you can use all MATLAB figure tools, for example, changing the scale and position of the plots.

6. STEADY STATE D-Q ANALYSIS

As it was already pointed out, the D-Q analysis group consists of steady state and dynamic analysis types. **Steady State D-Q Analysis** allows to estimate steady-state machine parameters and characteristics such as torque – speed curve, torque – advance angle curve and etc. as well as to compute the efficiency map and other performance maps. Since the variation of D-Q model parameters with rotor positions is not taken into account and the back-EMF waveform is assumed to be sinusoidal the accuracy of **Steady State D-Q Analysis** can be lower comparing with the finite element analysis.

6.1. Building a D-Q model.

The view of the main window when **Steady State D-Q Analysis** is chosen is shown in Figure 6.1. In order to use **Steady State D-Q Analysis** the D-Q model of the machine should be previously built, i.e. parameterized using FEA simulations. Use the **Build D-Q model** button to compute the D-Q model parameters of the machine. The short description of the d-q representation of a permanent magnet machine is presented in section 2.5. While the D-Q model is being built the model parameters L_d , L_q , L_{dq} , Ψ_{md} and Ψ_{mqd} are derived from the FEA solution for each pair of supply current and advance angle values to include saturation and cross-saturation effects into account. The supply current values are defined by the **Maximum RMS current** and **Current step** field values. The relationship between the supply current and the phase currents in each winding is defined in the same way as shown in Table 5.1. The advance angle values are in the range between 0^0 and 360^0 electrical degrees with the step defined by the **Advance angle step** field value. **Number of rotor positions** field specifies the number of rotor positions for which the D-Q model parameters are averaged. Rotor positions are spaced at intervals corresponding to half of cogging torque period. At least 2 rotor positions are recommended to compensate variation of the D-Q model parameters with the rotor rotation due to cogging torque. Iron losses can be included into the D-Q model using the **Include iron losses** checkbox. If iron losses are included, the D-Q model parameterization takes significantly longer time. Once the D-Q model is built, different machine characteristics can be obtained.

6.2. Viewing Steady State D-Q Analysis results.

The view of the **Plot Wizard** window when **Steady State D-Q Analysis** is chosen is shown in Figure 6.2. Use the **Quantities to plot** button to see the list of all available parameters. The plotted quantities can be displayed as a function of *Speed*, *Supply current* or *Advance angle* depending on the chosen item in the **X-axis quantity** pop-up menu. Speed, current and advance angle values should be entered in the corresponding fields. If a list of values is used, the values can be separated by space, comma or semicolon, alternatively the statement ‘start:step:stop’ can be used to enter linearly spaced values between ‘start’ and ‘stop’.

There are three **Model types** to choose for **Steady State D-Q Analysis**: *D-Q with sinusoidal supply*, *D-Q with PWM supply* and *FEA based. D-Q with sinusoidal supply* assumes ideal sinusoidal current

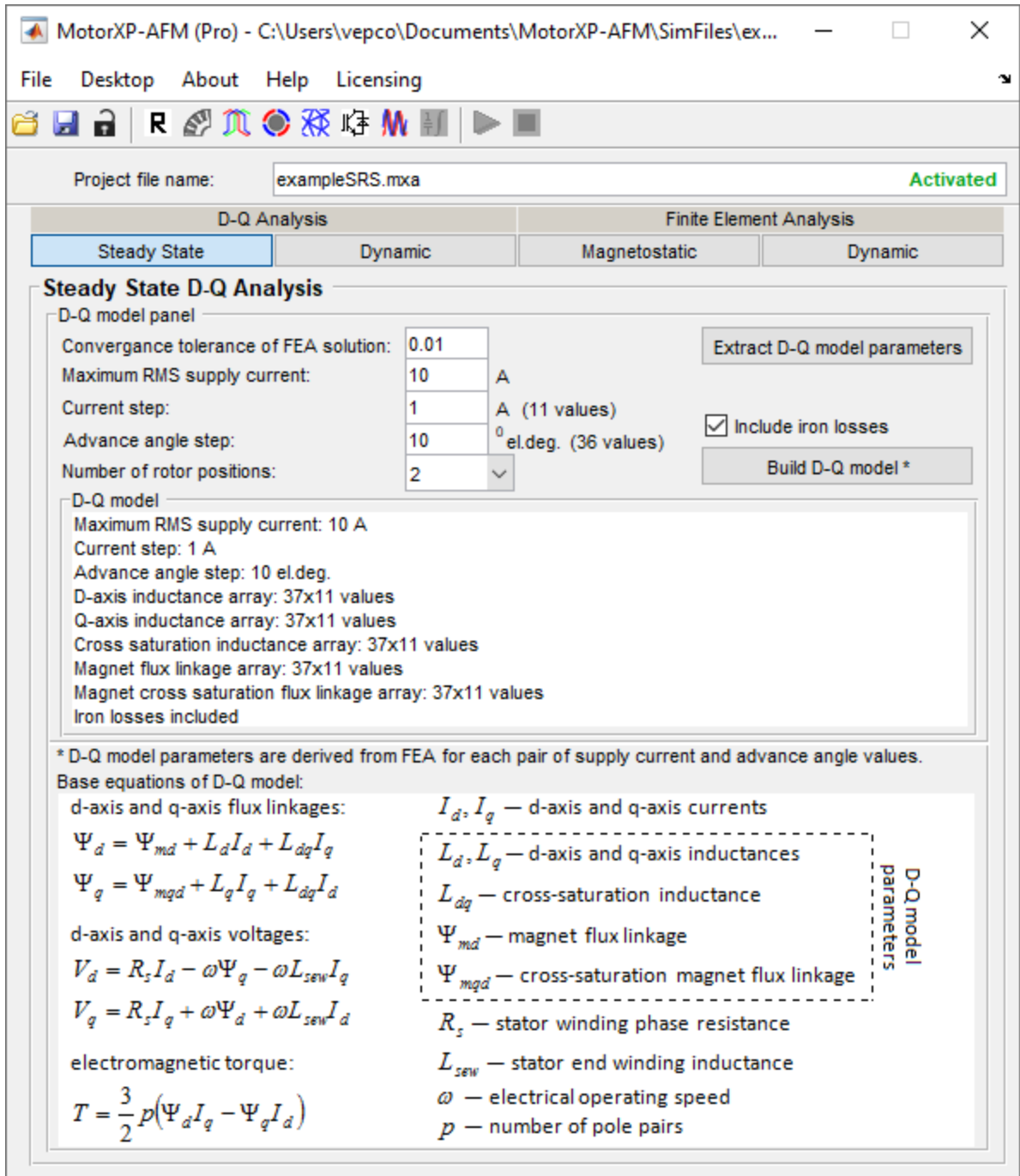


Figure 6.1. MotorXP-AFM main window for Steady State D-Q Analysis.

and voltage waveforms, ***D-Q with PWM supply*** uses the dynamic D-Q simulation (refer to chapter 7) and corresponding drive type (***Current hysteresis PWM*** or ***Space vector PWM*** specified in the **Drive Settings** window as described in section 4.5) to obtain the result. ***FEA based*** is the same as ***D-Q with sinusoidal supply*** but the result is determined directly from FEA solution (only for one rotor position) so the analysis can be done without the need to build the D-Q model.

Max. phase voltage selection defines either the ideal current source with no voltage limit is used (if ***unlimited voltage*** is chosen) or the limited supply voltage is used and how the **Max. RMS phase voltage** available from the inverter is determined. If ***by Vdc and PWM method*** is chosen, the **Max. RMS phase voltage** is determined based on the DC supply voltage V_{dc} and drive type

Plot Wizard: Steady State D-Q Analysis

D-Q analysis plot

Quantities to plot <<

X-axis quantity: Speed

Rotor speed value(s): 0:50:7000 RPM

RMS supply current value(s): 230 A

Advance angle value(s): 50 el.deg.

Max. phase voltage selection: by Vdc and PWM method

Max. RMS phase voltage: 204.124 ? V

Field-weakening control: max-torque-limited-current

Model setup

Model type: D-Q with sinusoidal supply

D-Q parameters interpolation: Nonlinear

☒ Multiple y-axes

Plot

Quantities to plot

- ☒ RMS phase current
- ☒ RMS phase voltage
- ☐ Average d-axis current
- ☐ Average q-axis current
- ☐ Average d-axis voltage
- ☐ Average q-axis voltage
- ☒ Advance angle
- ☒ Total torque
- ☐ Magnet torque
- ☐ Reluctance torque
- ☒ Input electrical power
- ☒ Output mechanical power
- ☐ Reactive power
- ☒ Efficiency
- ☒ Power factor
- ☐ Stator winding loss
- ☐ Iron loss
- ☐ RMS phase back-EMF
- ☐ D-axis inductance Ld
- ☐ Q-axis inductance Lq
- ☐ Mutual inductance Ldq
- ☐ D-axis magnet flux linkage
- ☐ Q-axis magnet flux linkage
- ☐ Lq/Ld ratio

Performance maps plot

Maps to plot >>

Control method: Max. efficiency

Advance angle values: 0:0.1:90 el.deg.

Maximum RMS phase voltage: 204.124 <-rated V

Maximum RMS phase current: 200 <-rated A

Maximum input power: <-rated W

Maximum speed: 7000 <-rated RPM

Mechanical loss coefficients: 2e-5 3.5e-3 0

Map resolution:

Torque step: 5 N*m

Speed step: 50 RPM

Model setup

Model type: D-Q with sinusoidal supply

D-Q parameters interpolation: Linearized

Plot Maps

Create Maps

Additional plotting options

Plot

- Plot and save to a CSV-file
- Plot and save to Excel
- Plot and save to a MAT-file
- Plot from a MAT-file

Figure 6.2. Steady State D-Q Analysis Plot Wizard window.

(*Current hysteresis PWM* or *Space vector PWM*) specified in the **Drive Setting** window as well as stator winding connection (star or delta). For the space vector PWM the maximum inverter phase peak voltage is $\frac{V_{DC}}{\sqrt{3}}$. For the current hysteresis PWM in order to keep the voltage and current sinusoidal, the inverter

phase peak voltage should not exceed $\frac{V_{DC}}{2}$. Note that for delta-connected winding the phase voltage of the machine is the line-to-line voltage of the inverter.

If the motor operates with the limited voltage the field-weakening operation above the base speed can be analyzed. There are two field-weakening strategies available: *max-torque-limited-current* and *const-output-power*. If the field-weakening is applied the current and advance angle are adjusted to meet the corresponding requirements of the field-weakening control algorithm. If *const-advance-angle* is chosen, the current and advance angle are no longer controlled - current starts to naturally decrease above the base speed.

D-Q parameters interpolation specifies how D-Q model parameters are interpolated. *Linearized* interpolation is sufficient in most cases provided that the D-Q model has been built with enough number of supply current and advance angle pairs.

When **Model type** is set to *D-Q with PWM supply* the simulation speed/accuracy setup can be additionally adjusted by clicking the **Simulation speed/accuracy** button as shown in Figure 6.3.

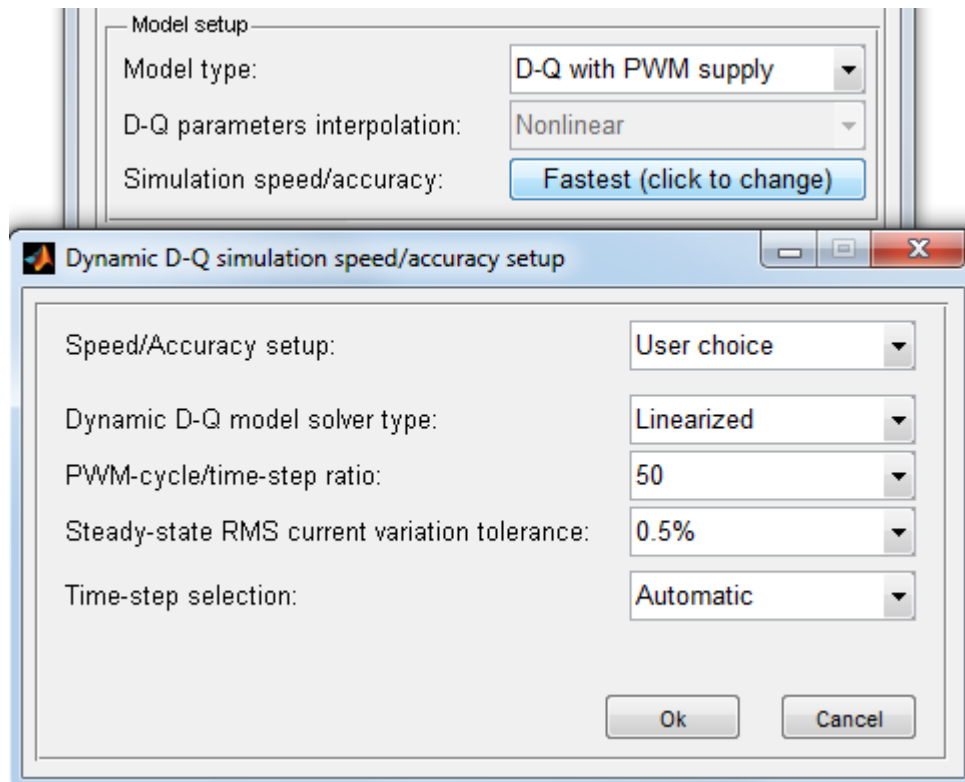


Figure 6.3. Dynamic D-Q simulation speed/accuracy setup.

You can choose one of the predefined setups (*Fastest*, *Balanced* or *Accurate*) from the **Speed/Accuracy setup** pop-up menu or manually adjust the simulation speed/accuracy setup choosing the *User choice* item.

Dynamic D-Q model solver type set to *Linearized* is sufficient in most cases. Be careful when using *Linear* solver type since in this case the fixed D-Q model parameters are used – linear solver can only be used if the current waveform is very close to sinusoidal.

PWM-cycle/time-step ratio determines the choice of the time step when **Time-step selection** is set to *Automatic*. Good practice is to manually specify the time-step since in some cases the optimal time-step cannot be determined by the automatic selection.

Steady-state RMS current variation tolerance defines the acceptable variation of the RMS current from one period to the next to determine whether the steady-state is reached. The simulation stops when the steady state is reached i.e. when the variation of the RMS current from one period to the next does not exceed the specified variation tolerance.

Several examples of **Steady State D-Q Analysis** plots are shown in Figures 6.4-6.5. Figure 6.4 shows the plot corresponding to the **Steady State D-Q Analysis** setup shown in Figure 6.2 with ideal sinusoidal supply and “max-torque-limited-current” field-weakening control above the base speed. Figure 6.5 compares plots obtained with ideal sinusoidal supply with no voltage limit (*D-Q with sinusoidal supply*) and with space vector PWM inverter supply (*D-Q with PWM supply*) for 1500RMP rotor speed. Supply current is varied in the range between 0 and 300A in both cases. As shown for the ideal sinusoidal supply (top) the phase current changes from 0 to 173A (delta-connected winding), the phase voltage is freely increasing, and the advance angle is strictly 0 electrical degrees. For the PWM supply (bottom) after some point the inverter is no longer able to maintain the desired current since the phase voltage reaches its limit. Note that the effective advance angle is shown for PWM supply.

There are additional plotting options as shown in Figure 6.2. These allow saving the plotted quantities to a CSV-file (text file with comma-separated values), to a Microsoft® Excel® spreadsheet file or to a MAT-file or plot the selected quantities from the MAT-file previously saved.

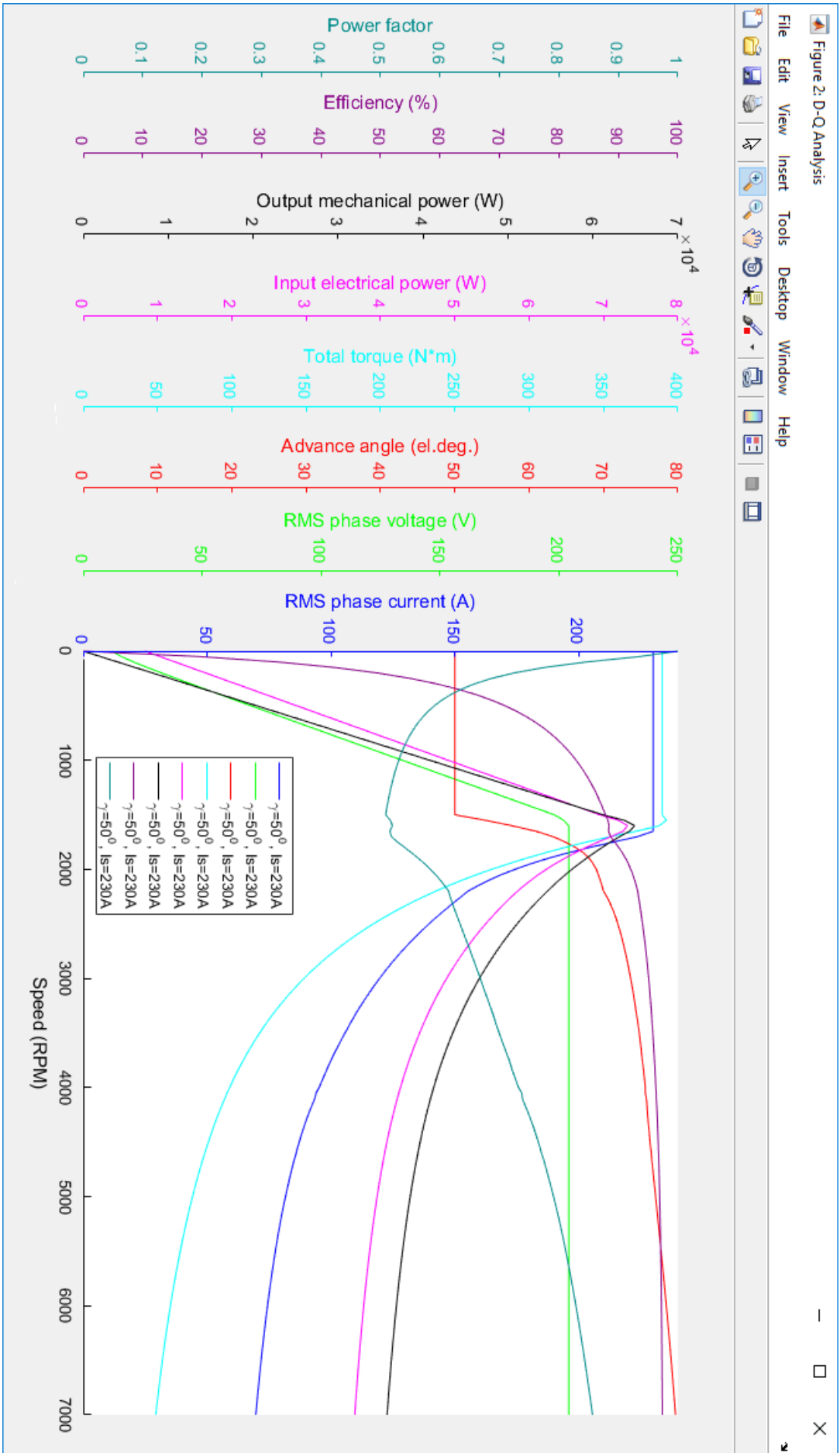


Figure 6.4. D-Q Analysis plot with ideal sinusoidal supply and *max-torque-limited-current* field-weakening control above the base speed.

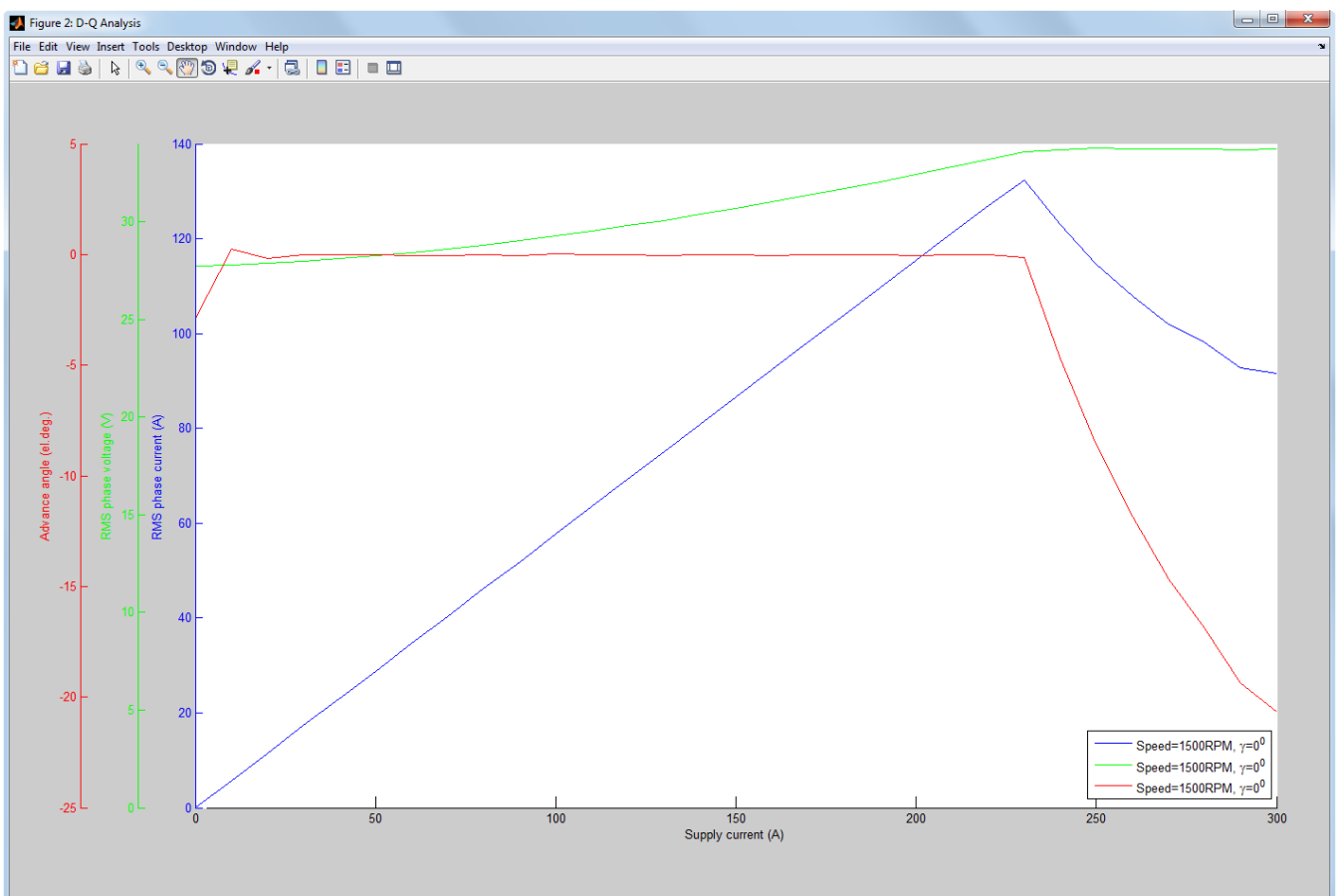
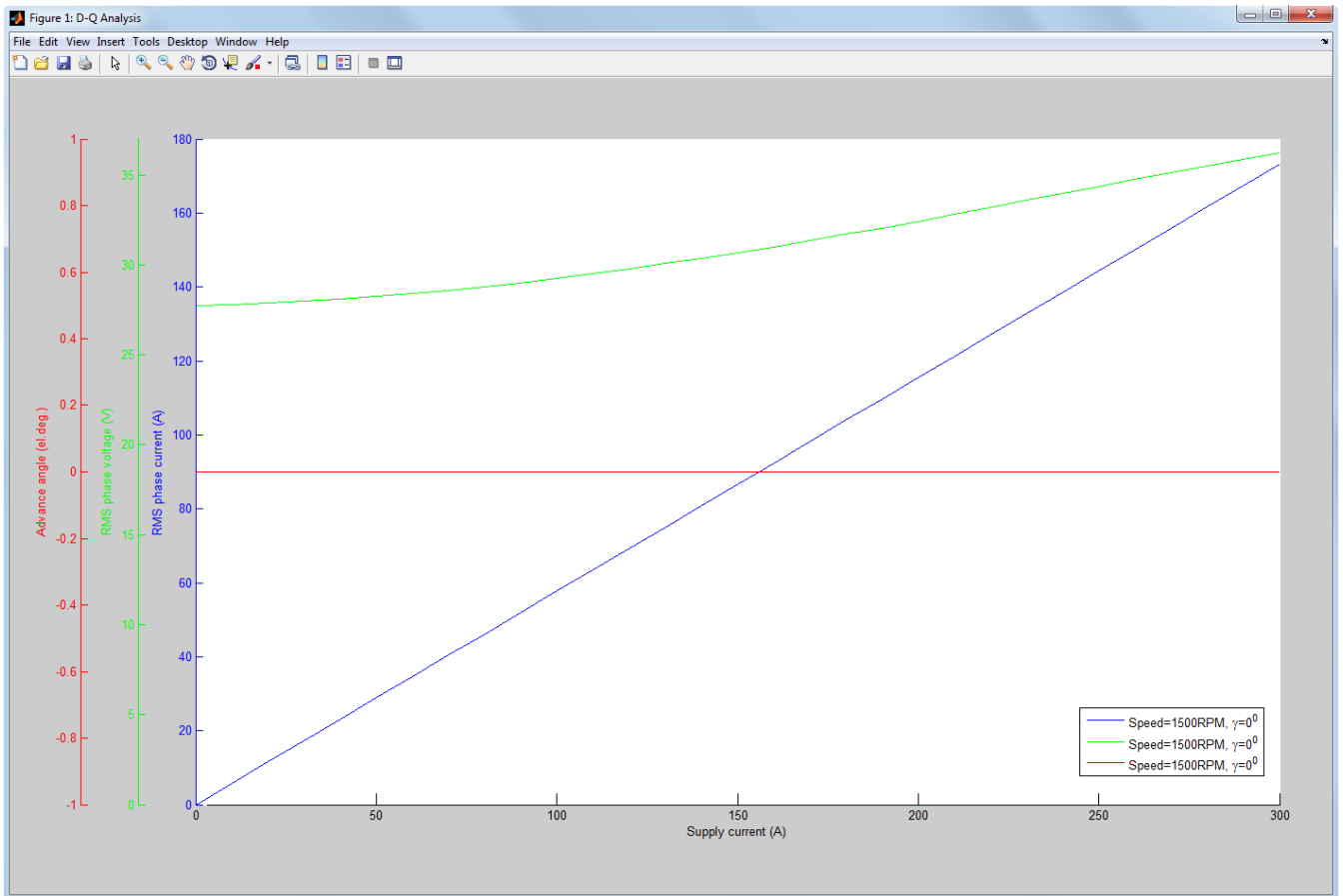


Figure 6.5. Comparison between *D-Q with sinusoidal supply* and *D-Q with PWM supply* model types.

6.2. Efficiency map and other performance maps.

Another option of **Steady State D-Q Analysis** is the generation of the efficiency map and other performance maps such as current map, voltage map, power map, loss map and etc. Use the **Maps to plot** button to see the full list of available performance maps as shown in Figure 6.6. There are two control strategies available to generate the performance maps of the machine specified by the **Control method** pop-up menu: *Max. efficiency* and *MTPA/MTPV*. If *Max. efficiency* is chosen, the calculation of the performance maps is based on the machine operation with maximum possible efficiency, i.e. the iterative procedure is applied to search for the RMS current and advance angle values which provide the maximum efficiency for the specific values of speed and load torque taking into account the current and voltage limits. If *MTPA/MTPV* is chosen, the calculation of the performance maps is based on the maximum torque per ampere (MTPA) and maximum torque per volt (MTPV) operation with the current and voltage limits taken into account. The example of the efficiency maps for motor and inverter and some other performance maps are shown in Figure 6.7. Note that the part of the efficiency map below 80% is not shown as the **Hide efficiency less than** field is set to 80% in the **Plot Wizard** window.

Advance angle values field specifies the range of advance angles the performance maps are computed for. Use the statement ‘start:step:stop’ to enter linearly spaced values between ‘start’ and ‘stop’, where ‘step’ determines the accuracy of the calculated performance maps in terms of the advance angle tolerance. Advance angle values in the range between 0° and 90° electrical degrees correspond to the motor operation mode, the range of advance angle values between 90° and 180° degrees correspond to the generator mode. **Maximum RMS phase voltage** and **Maximum RMS phase current** define the supply voltage and current limits applied while the performance maps are computed. **Maximum input power** does influence the calculation of performance maps and only affects the way how the maps are displayed: the part of the map beyond the input power limit specified by the **Maximum input power** field will not be shown. If the **Maximum input power** field is left empty when the input power is not limited (limited only by maximum voltage and current) and no part of the map will be hidden.

Button ‘<-rated’ to the right of some fields allows you to set up the corresponding parameter to its rated value specified in the **Rated Data** window.

Use the **Mechanical loss coefficients** field to apply the mechanical losses while the efficiency map of the machine is plotted. Mechanical loss coefficients do not influence the calculation of performance maps and they are applied after the performance maps are calculated. Mechanical loss coefficients are entered as a vector whose elements are the coefficients in descending powers of the mechanical loss equation. For the mechanical loss coefficients [2e-5 3.5e-3 0] shown in Figure 6.6 the mechanical loss equation is as follows:

$$W_{mech.loss} = 0.00002(rpm)^2 + 0.0035(rpm) + 0,$$

where *rpm* – rotor speed in RPM.

Performance maps plot

Maps to plot <<

Control method: Max. efficiency

Advance angle values: 0:0.1:90 el.deg.

Maximum RMS phase voltage: 204.124 <-rated V

Maximum RMS phase current: 200 <-rated A

Maximum input power: <-rated W

Maximum speed: 7000 <-rated RPM

Mechanical loss coefficients: 2e-5 3.5e-3 0

Map resolution:

Torque step: 5 N*m

Speed step: 50 RPM

Model setup

Model type: D-Q with sinusoidal supply

D-Q parameters interpolation: Linearized

Plot Maps

Create Maps

Maps to plot

☒ Efficiency (motor only)

☒ Efficiency (inverter only)

☒ Efficiency (motor+inverter)

☒ Phase current

☒ Phase voltage

☐ Input electrical power

☐ Reactive power

☐ Power factor

☒ Advance angle

☐ Stator winding loss

☐ Iron loss

☐ Inverter loss

☐ Input power envelope

Hide efficiency less than 80 %

Plot Maps

Save selected maps in mat-file

Save selected maps in csv-files

Display efficiency table (motor only)

Display efficiency table (inverter only)

Display efficiency table (motor+inverter)

Display phase current table

Display phase voltage table

Display input power table

Display reactive power table

Display power factor table

Display advance angle table

Display stator winding loss table

Display iron loss table

Display inverter loss table

Display efficiency map details

Extract Id, Iq tables

Extract Id, Iq tables dialog box

Extract Id, Iq tables

Id and Iq current components tables are extracted from the previously computed efficiency map data*. These tables can be uploaded into controller to guarantee the best motor performance.

Id, Iq current tables dimensions (rows x columns)

Number of torque values (rows): 20

Number of speed values (columns): 20

Output file format: MAT-file

*Efficiency map data:

Advance angles range: 0:0.1:90 el.deg.

Maximum RMS phase voltage: 204.124 V

Maximum RMS phase current: 200 A

Maximum speed: 7000RPM

Number of torque values: 72 (torque step 5 N*m)

Number of speed values: 141 (speed step 50 RPM)

Control method: Max. efficiency

Extract Id, Iq

Close

Figure 6.6. Performance maps plot panel of Plot Wizard with the Extract Id, Iq tables dialog box.

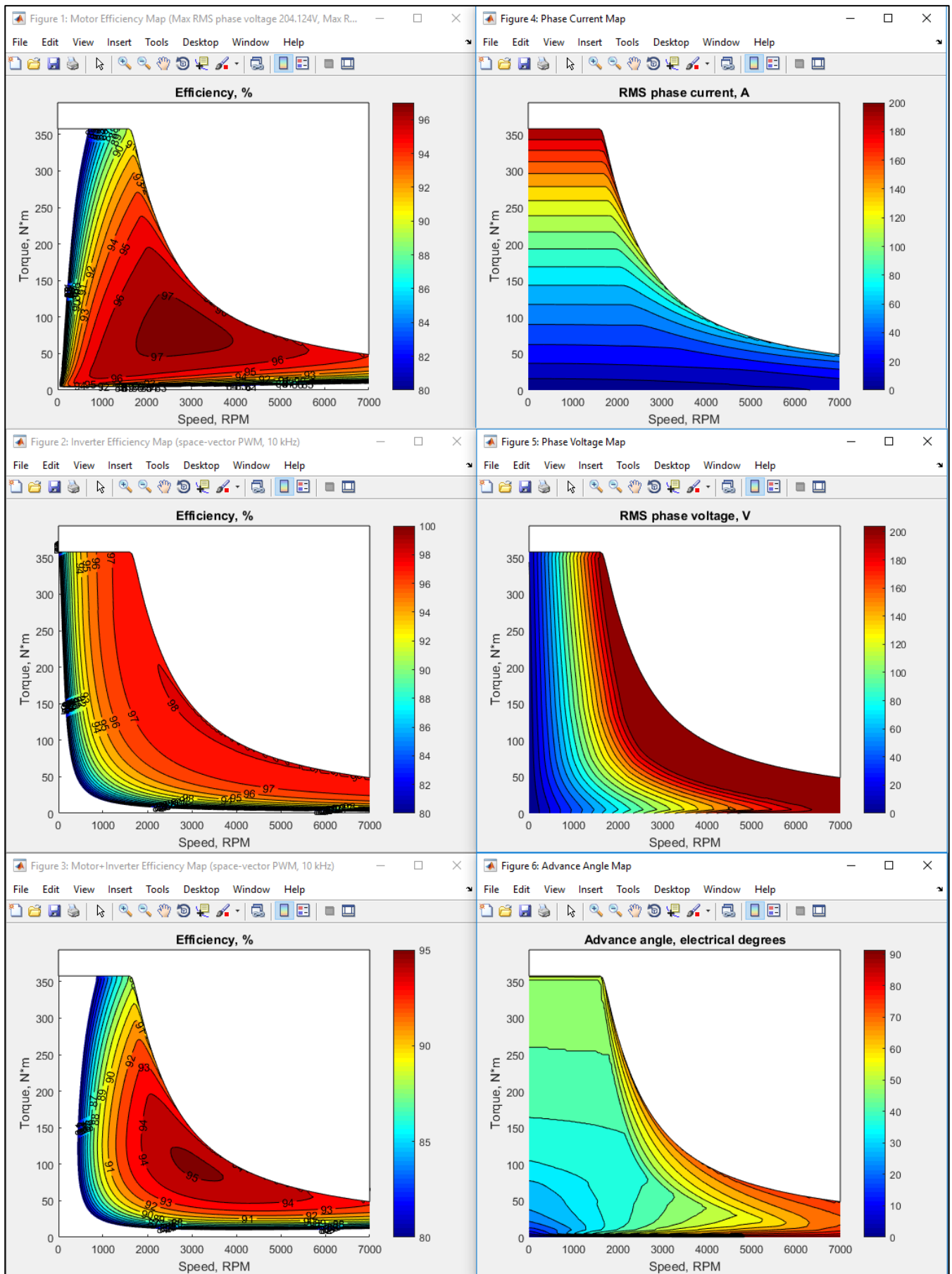


Figure 6.7. Some performance maps generated by Steady State D-Q Analysis Plot Wizard.

Number of points (speed-torque pairs) at which the performance maps are calculated is defined by the **Map resolution** panel.

D-Q parameters interpolation specifies how D-Q model parameters are interpolated. *Linearized* interpolation is sufficient in most cases provided that the D-Q model has been built with enough number of supply current and advance angle pairs. Performance maps are calculated assuming ideal sinusoidal current and voltage waveforms.

To generate the performance maps, click the **Create Maps** button. All the performance maps will be calculated no matter which ones are selected in the Plot Wizard window; however only selected maps will be displayed. Once the performance maps are computed for the first time, they will be saved within the project mxa-file. Every time when you click the **Create Maps** button once again you will be asked whether you want to replace previously computed maps with the new ones. Use the **Plot Maps** button to plot the performance map previously saved. There are also some additional options available to the right of the **Plot Maps** button as shown in Figure 6.6, such as displaying the performance map as a table choosing the corresponding *Display ... table* item or saving selected maps in MAT-file or CSV-files.

6.3. Extracting Id and Iq current components tables from the efficiency map data.

Calculation of the performance maps corresponds to determining the optimum motor operating conditions with maximum motor efficiency or MTPA/MTPV operation depending on the selected **Control method** pop-up menu item. This is a valuable information to design the control system. Id and Iq current components can be extracted from the efficiency map data and then uploaded into controller. Figure 6.6 shows the **Extract Id, Iq tables** dialog box. The output data produced by the **Extract Id, Iq tables** dialog box are 2-D lookup table representations of the Id and Iq current components with rows corresponding to the torque values and columns corresponding to the speed values. Number of torque and speed value can be different from those used for the calculation of the performance maps (defined by the **Torque step** and **Speed step** fields of the **Map resolution** panel). The output tables can be saved as a MAT-file (all tables in one file) or CSV-files (each table in separate CSV-file) depending on the selected **Output file format** pop-up menu item.

6.4. Extracting D-Q model parameters.

Parameters of the D-Q model such as the d-axis and q-axis inductances L_d and L_q , magnet flux linkage Ψ_{md} , cross-saturation inductance L_{dq} and cross saturation magnet flux linkage Ψ_{mqd} , can be extracted and saved in a separate file (files) to use the generated D-Q model in another application. The **Extract D-Q model parameters** window shown in Figure 6.8 is available by clicking the **Extract D-Q model parameters** button of the **Steady State D-Q Analysis** main window (Figure 6.1). The output data are available in two formats: as a *function of phase RMS current and advance angle* (see Figure 6.8(a)) and as a *function of D-axis and Q-axis currents Id and Iq* (see Figure 6.8(b)), depending on the selected **Output data format** pop-up menu item.

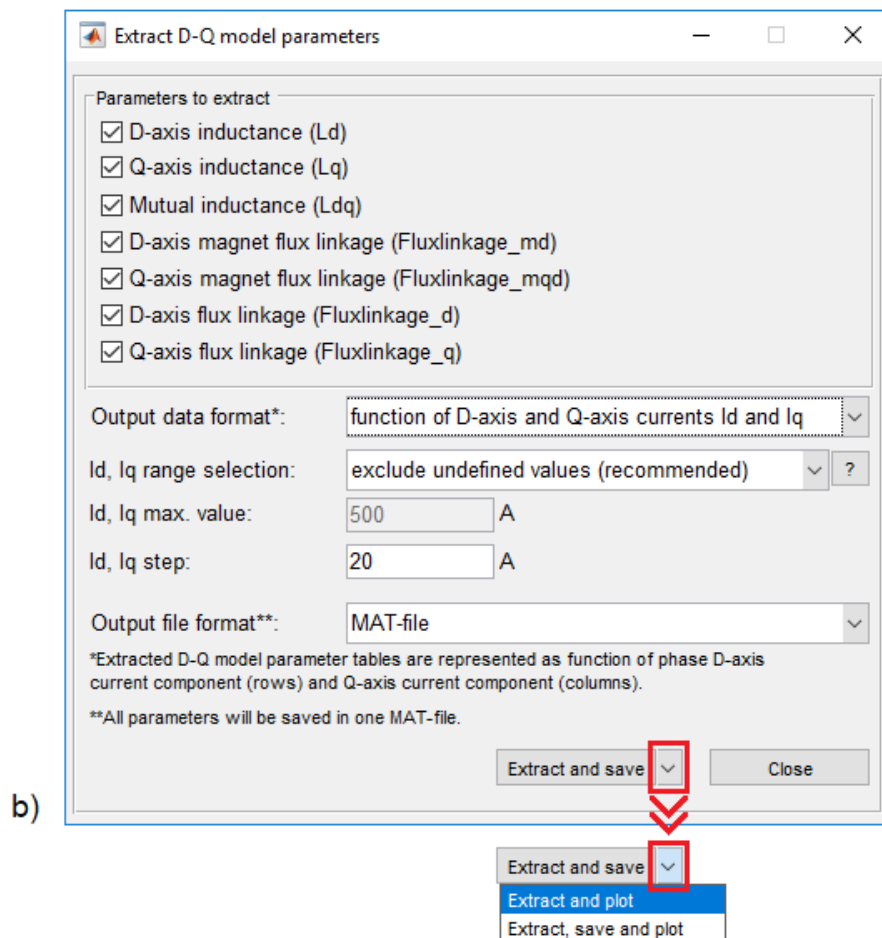
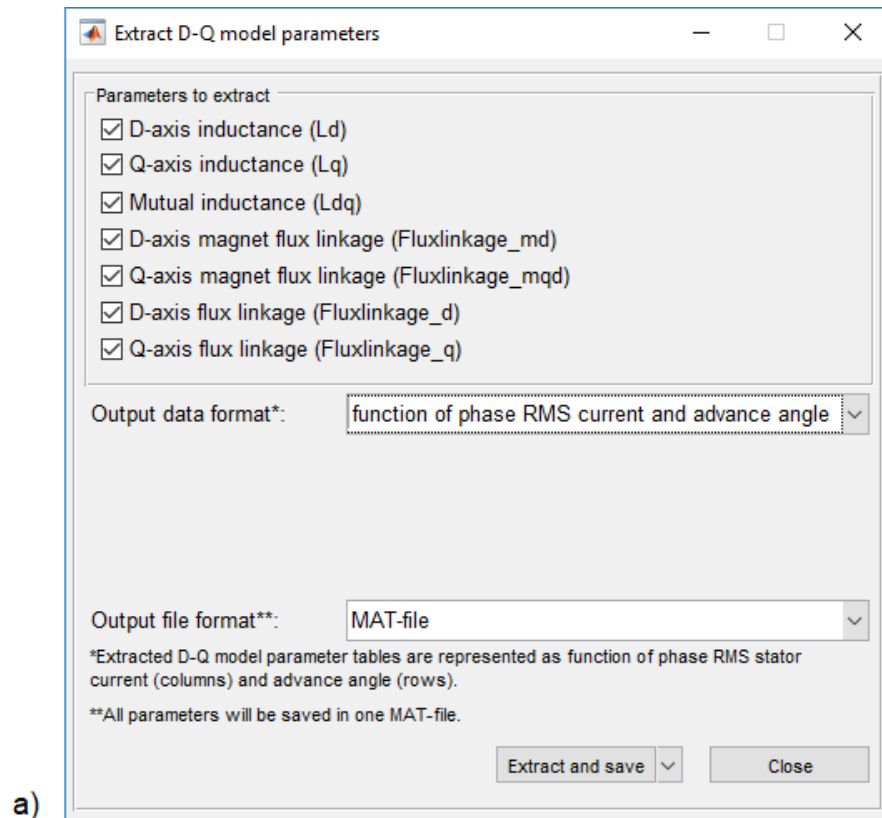


Figure 6.8. **Extract D-Q model parameters** window for different output data formats: (a) *function of phase RMS current and advance angle*, (b) *function of D-axis and Q-axis currents I_d and I_q* .

When the *function of D-axis and Q-axis currents I_d and I_q* output data format is used the **Id, Iq range selection** pop-up menu determines the maximum value of I_d and I_q (I_d and I_q has the same maximum value which is displayed in the **Id, Iq max. value** field). Figure 6.9 explains the I_d , I_q range selection principle. Since initially the D-Q model parameters are determined as a function of phase RMS current and advance angle, when the *maximum possible range* item is selected from the **Id, Iq range selection** pop-up menu, there will be some I_d and I_q pairs which the D-Q model parameters are not defined for (“undefined values” in Figure 6.9). Undefined values are replaced with “NaN” (not a number) in the output tables. I_m in Figure 6.9 corresponds to the maximum phase current amplitude for which the D-Q model parameters are determined ($I_m/\sqrt{2}$ is the RMS phase current corresponding to the **Maximum RMS supply current** value of the **Steady State D-Q Analysis** main window (Figure 6.1)). To exclude undefined values of the D-Q model parameters from the output tables the *exclude undefined values* item should be selected from the **Id, Iq range selection** pop-up menu. **Id, Iq step** determines the size of the output tables (number of rows is equal to the number of columns). The output tables can be saved as a MAT-file (all tables in one file) or CSV-files (each table in separate CSV-file) depending on the selected **Output file format** pop-up menu item.

Additional options to the right of the **Extract and save** button (see Figure 6.8) allow also to plot the selected parameters versus I_d and I_q or versus RMS phase current and advance angle depending on the selected **Output data format** pop-up menu item.

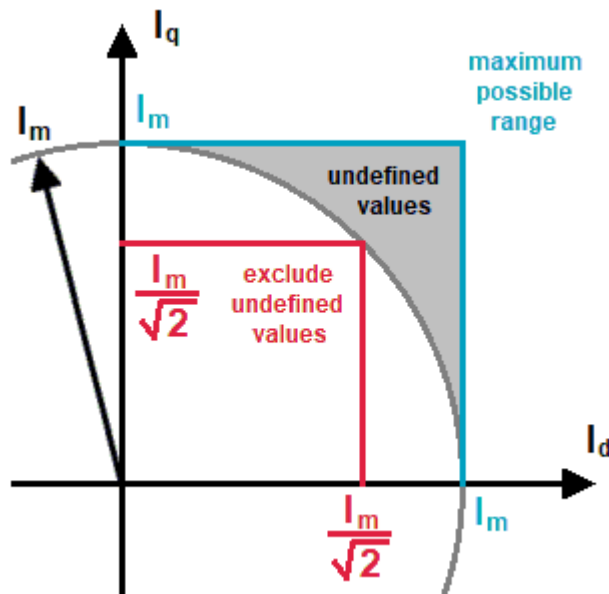




Figure 6.9. Explanation on the I_d and I_q current components range selection.

7. DYNAMIC D-Q ANALYSIS

Dynamic D-Q Analysis is based on Exp. 2.11 and allows the user to simulate the dynamic behavior of the machine taking into account non-sinusoidal voltage and current waveforms imposed by inverter switching. In order to avoid misunderstanding of the results you should be aware of the following limitations of **Dynamic D-Q Analysis**:

- Back-EMF waveform is assumed to be sinusoidal. To examine the Back-EMF waveform, use **Magnetostatic FE Analysis** or **Dynamic FE Analysis** instead.
- Variation of the D-Q model parameters with rotor position is not taken into account and the torque is defined by Exp. 2.10, therefore the torque ripple is not properly determined. To determine the torque ripple, use **Magnetostatic FE Analysis** or **Dynamic FE Analysis** instead.

7.1. Running Dynamic D-Q Analysis.

The view of the main window when **Dynamic D-Q Analysis** is chosen is shown in Figure 7.1. In order to use **Dynamic D-Q Analysis** the D-Q model of the machine should be built as described in the previous chapter. Use toolbar buttons  and  to start and stop the simulation. If the simulation is stopped all the data of the running simulation will be lost.

Solver type specifies how D-Q model parameters are interpolated. *Linearized* solver type is sufficient in most cases provided that the D-Q model has been built with enough number of supply current and advance angle pairs. Be careful when using *Linear* solver since in this case fixed D-Q model parameters are used – linear solver can only be used if the current waveform is very close to sinusoidal, i.e. the switching ripple in the current is considerably small comparing with the current amplitude.

Be aware that **Time step** should be small enough comparing with the PWM sampling period to get accurate results. Control the *Discretization error* in the **Time Average Quantities** window shown in Figure 7.3 to adjust the time step. It is recommended that the discretization error does not exceed 1%.

There are two ways to stop the dynamic D-Q simulation in MotorXP-AFM: by specifying the **Simulation stop time value** or by reaching the steady state, i.e. the simulation will be automatically stopped when the steady state is reached. **Steady-state RMS current variation tolerance** defines the acceptable variation of RMS current from one period to the next to determine whether the steady-state is reached.

The phase current of the machine depends on the **Target RMS supply current** field value and on the stator winding connection. Since the supply current is a line current, for star connection the phase current is equal to the supply current, for delta connection the phase current is equal to the supply current divided by $\sqrt{3}$. **Target advance angle** determines an angle between the current phasor and q-axis of the rotor as shown in Figure 2.6. For the current hysteresis and space vector PWM drive types the current control algorithm is applied to adjust the inverter voltage in order to maintain d-axis and q-axis current components defined by the **Target RMS supply current** and **Target advance angle** field

values. For the space vector PWM the field oriented current control is used, for the current hysteresis PWM the hysteresis (bang-bang) current control is used. For the six-step drive the current is not controlled – the current is determined by the DC voltage, rotor speed, switch duty cycle (120 or 180 electrical degrees) and commutation advance angle.

Rotor speed dependency – specifies whether the rotor speed is fixed (when *Fixed speed simulation* is chosen) or varies depending on the load and electromagnetic torque of the motor (when *Variable speed simulation* is chosen). If *Fixed speed simulation* is chosen, you should also specify the speed in the field appearing to the right. If *Variable speed simulation* is chosen, the initial speed should be specified.

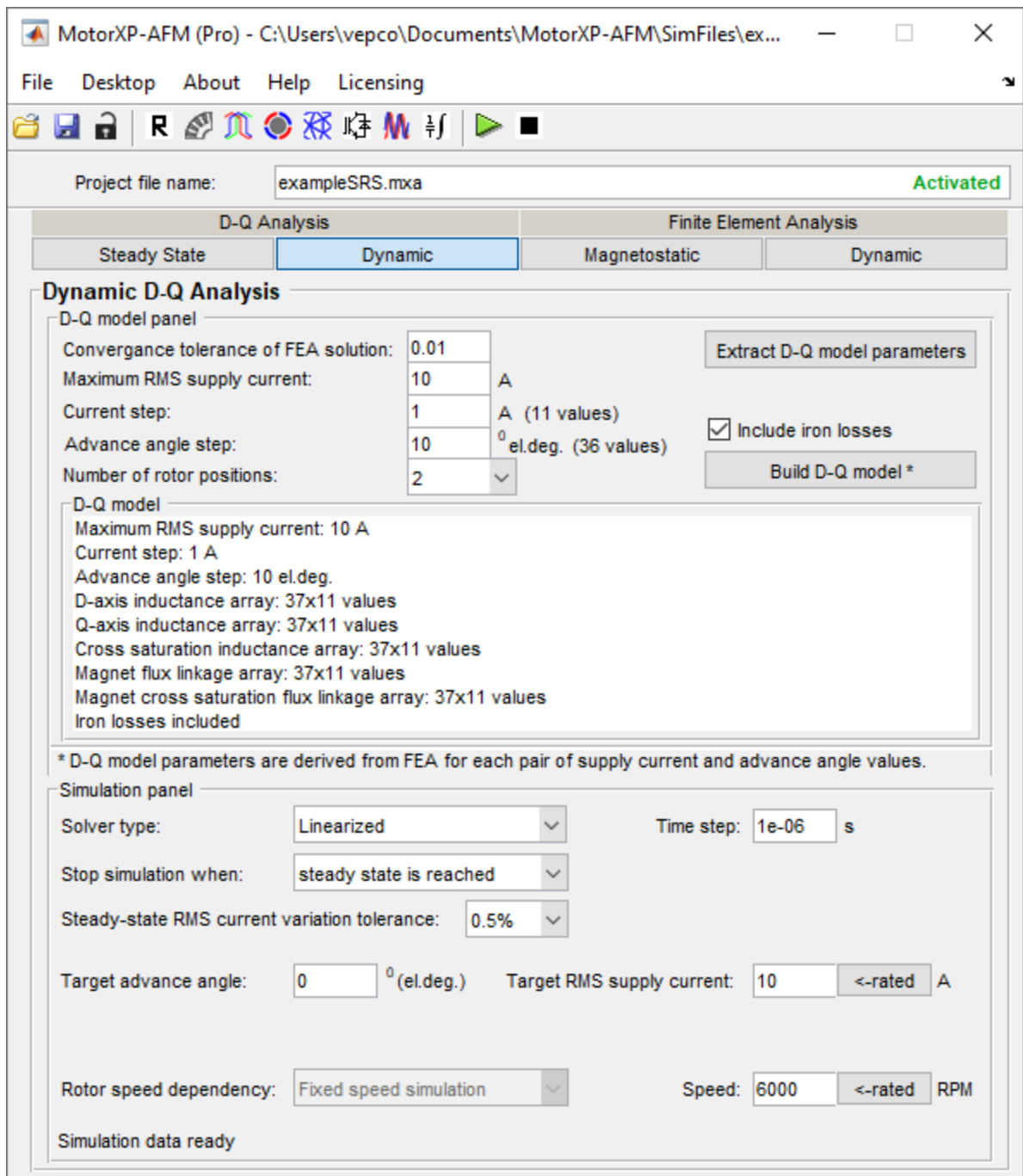


Figure 7.1. MotorXP-AFM main window for Dynamic D-Q Analysis.

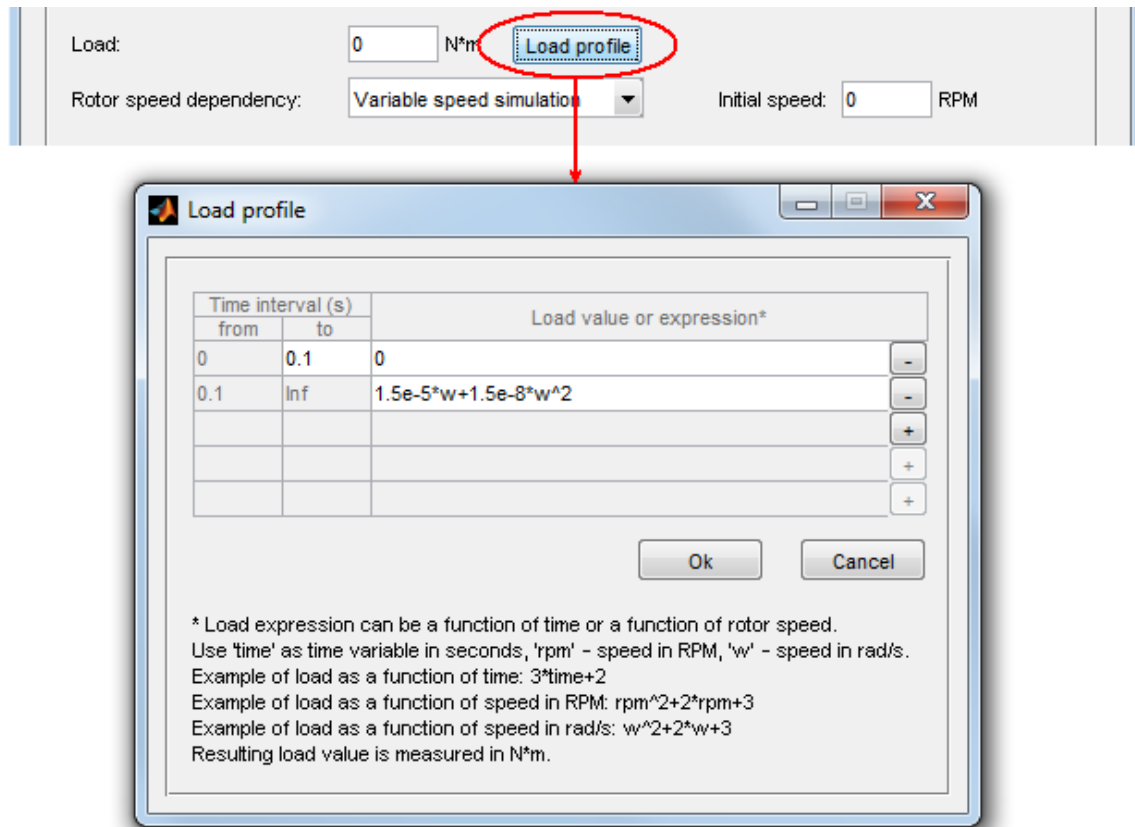


Figure 7.2. Specifying the load for the dynamic D-Q simulation.

Button '<-rated' to the right of some fields allows you to set up the corresponding parameter to its rated value specified in the **Rated Data** window.

If simulation with variable speed is used the load should be specified. The load can be specified either by a single value entered in the **Load** field or different load values can be specified for each time interval using the **Load profile** window as shown in Figure 7.2. Load can be a function of time or speed. Speed can be measured in rad/s or in RPM. Use +/- buttons to add or delete the time interval.

7.2. Viewing Dynamic D-Q Analysis results.

It is possible to view the **Dynamic D-Q Analysis** results as time-averaged quantities or waveform and spectrum plots. Use $\frac{1}{T}\int$ toolbar button to view time-averaged quantities as shown in Figure 7.3. The displayed quantities can be averaged over one, two or three electrical periods or the averaging time can be defined directly choosing *User choice* from the **Averaging time selection** pop-up menu.

The view of the **Plot Wizard** window when **Dynamic D-Q Analysis** is chosen is shown in Figure 7.4. It is organized as a standard MATLAB plotting construction consisting of a set of `subplot` and `plot` functions. **Subplot** checkboxes allow you to control the number of axes or rectangular panes displayed within a current figure window. The corresponding axes are activated or deactivated by a mouse click within a cell of the **Subplot** column. When the subplot is activated, the **change** button allows you to choose quantities to be plotted into the selected axes.

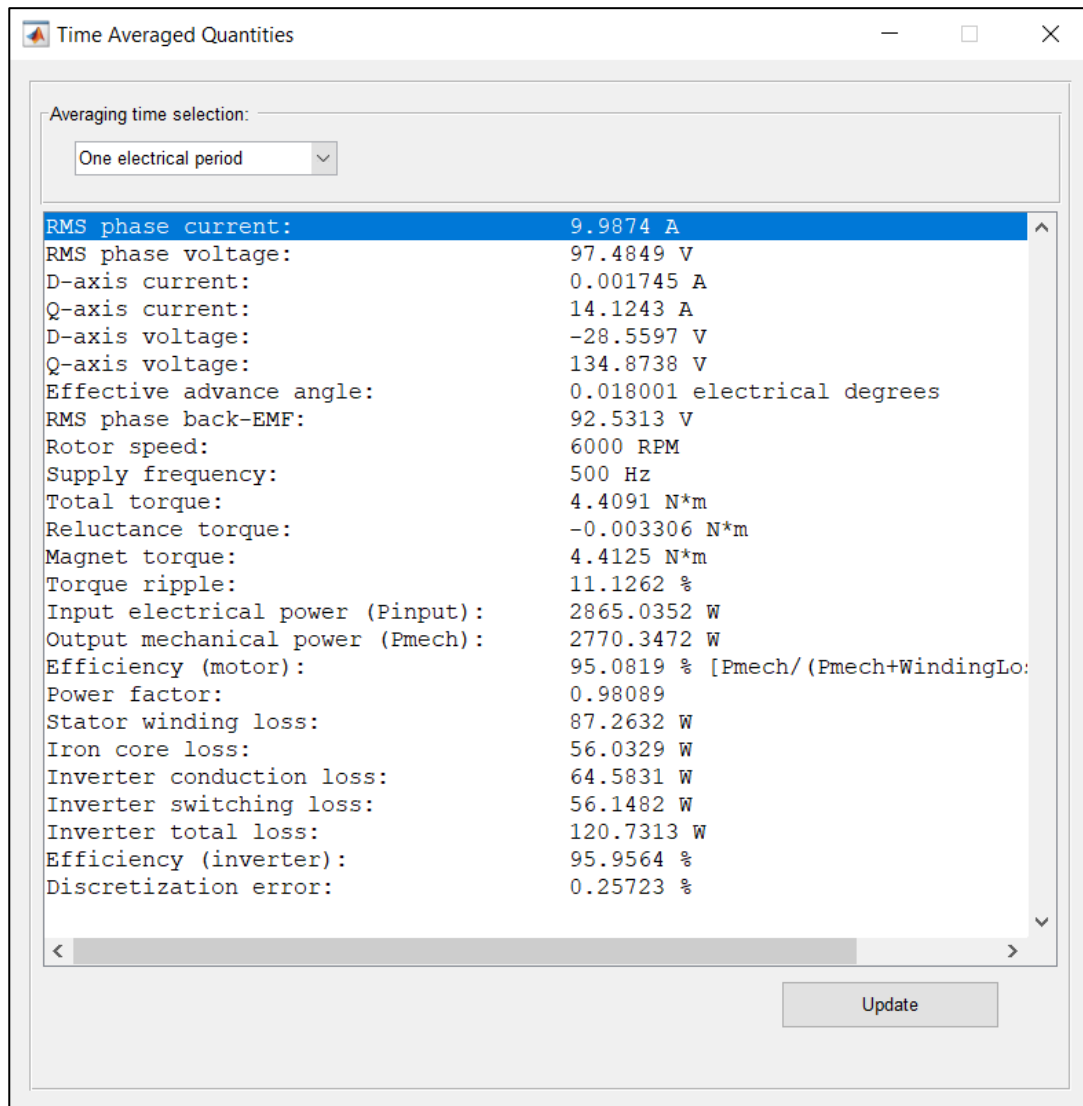


Figure 7.3. Time-averaged quantities.

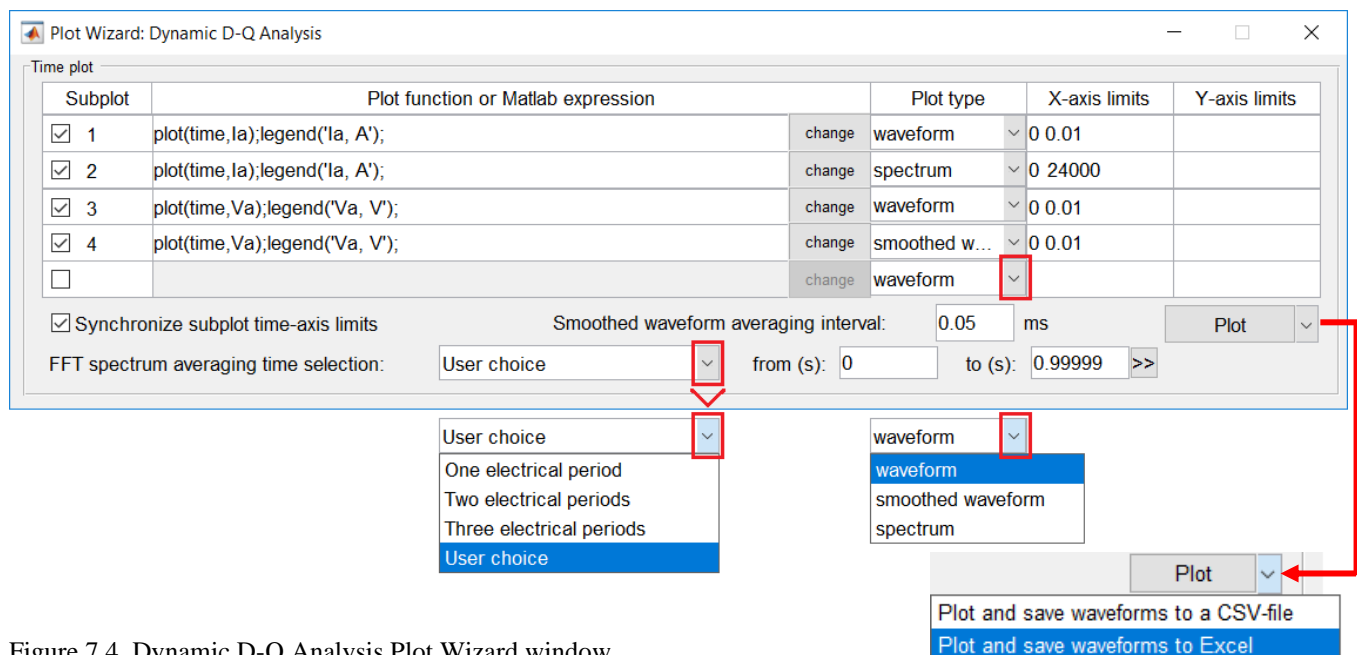


Figure 7.4. Dynamic D-Q Analysis Plot Wizard window.

Quantities to plot can be chosen from the dialog shown in Figure 7.5. Use Ctrl key to select several quantities. The list of available quantities is given below:

- Stator phase current (phase A, B, C);
- Phase current, d-axis;
- Phase current, q-axis;
- Phase voltage (phase A, B, C);
- Phase voltage, d-axis;
- Phase voltage, q-axis;
- Effective advance angle;
- Back-EMF (phase A, B, C);
- Back-EMF, d-axis;
- Back-EMF, q-axis;
- Electromagnetic torque (by flux linkage and current);
- Load torque on the motor shaft;
- Rotor speed;
- Rotor angular position;
- Input active electrical power;

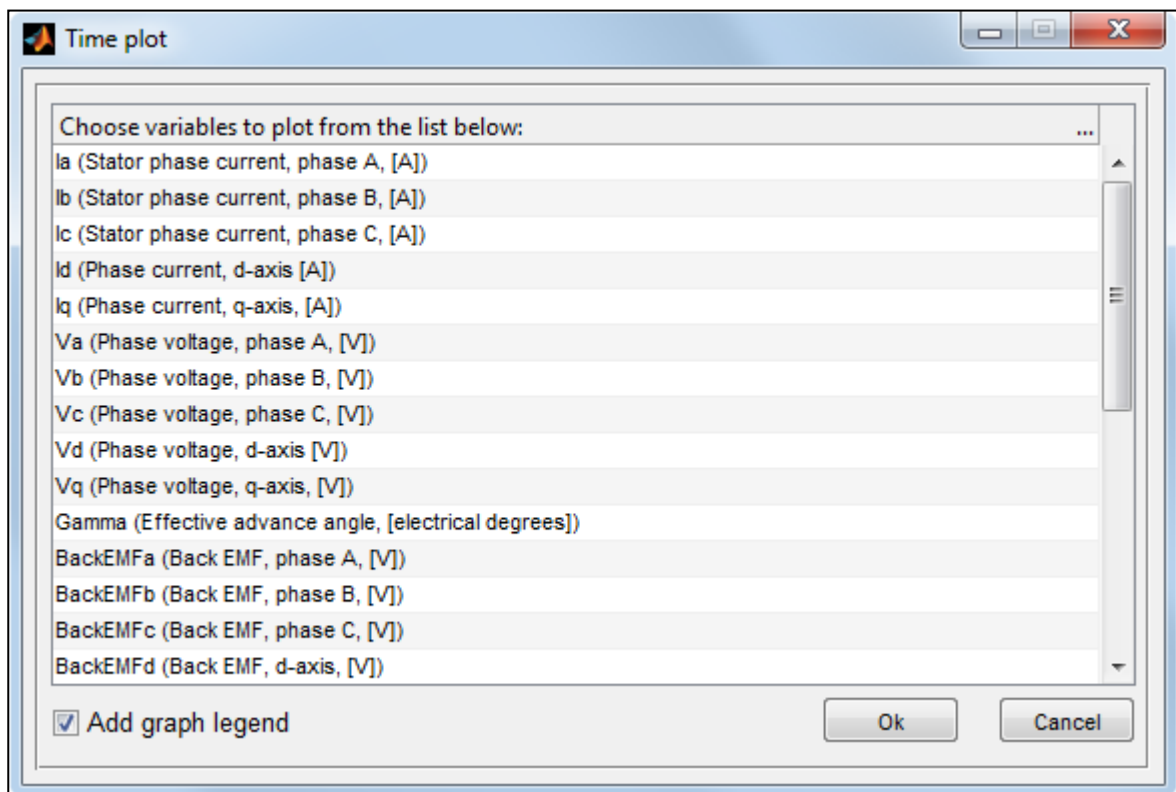


Figure 7.5. Choosing quantities to be plotted.

- Reactive electrical power;
- Mechanical power on the rotor shaft;
- Stator winding losses;
- Time derivative of the magnetic field energy;
- Flux linkage (phase A, B, C);
- Flux linkage, d-axis;
- Flux linkage, q-axis;
- Magnet torque (by flux linkage and current);
- Reluctance torque (by flux linkage and current);
- D-axis inductance;
- Q-axis inductance;
- Cross-saturation inductance;
- D-axis magnet flux linkage;
- Q-axis cross-saturation magnet flux linkage;

By clicking the **OK** button of the dialog (Figure 7.5), the MATLAB-expression is constructed to plot the selected quantities appearing in the corresponding line as shown in Figure 7.4 for plotting the phase A stator current waveform (first line), phase A stator current spectrum (second line), phase A stator voltage waveform (third line) and phase A stator voltage waveform after smoothing (fourth line). When the **Plot** button is clicked, the MATLAB figure window with corresponding plots of the selected quantities will appear (see Figure 7.6).

As it is seen, the plotting expression consists of the `plot` function with selected variables used as input arguments. If the **Add graph legend** checkbox of the dialog is checked, the `legend` function is added to the plotting expression so the graph legend of the corresponding axes will be shown. All plotting expressions are editable, so you can use any MATLAB plotting options and functions to change the way the plots appear on the screen. You can also change `time` variable to any other variable you would like to use as an input argument of the `plot` function. There is also a list of additional variables which can be used within a plotting expression (see section 9.3).

X-axis limits and **Y-axis limits** fields allow you to set the x-axis and y-axis limits, respectively, to the specified values. Two limit values within a cell are separated by a space, comma ‘,’ or semicolon ‘;’. If the cell is empty, the limits of the corresponding axis will be chosen automatically. If the **Synchronize subplot time-axis limits** checkbox is checked, all subplots of the figure will have identical limits along the time-axis when you zoom or pan one of the subplots of the figure.

There are several plotting options available from the **Plot type** pop-up menu: *waveform*, *smoothed waveform* and *spectrum* (see Figure 7.4). Smoothed waveform plot type can be useful when plotting the

pulse width modulated waveforms to filter out the PWM carrier frequency and obtain the smooth waveform of the signal (compare subplots 3 and 4 of Figure 7.6). The **Smoothed waveform averaging interval** field specifies the sliding window width of the smoothing algorithm. The resolution of the frequency spectrum, plotted when *spectrum* is chosen from the **Plot type** pop-up menu, is controlled by the **FFT spectrum averaging time selection** pop-up menu. The spectrum can be calculated over one, two or three electrical periods or the time can be defined directly choosing *User choice* from the **FFT spectrum averaging time selection** pop-up menu (see Figure 7.4).

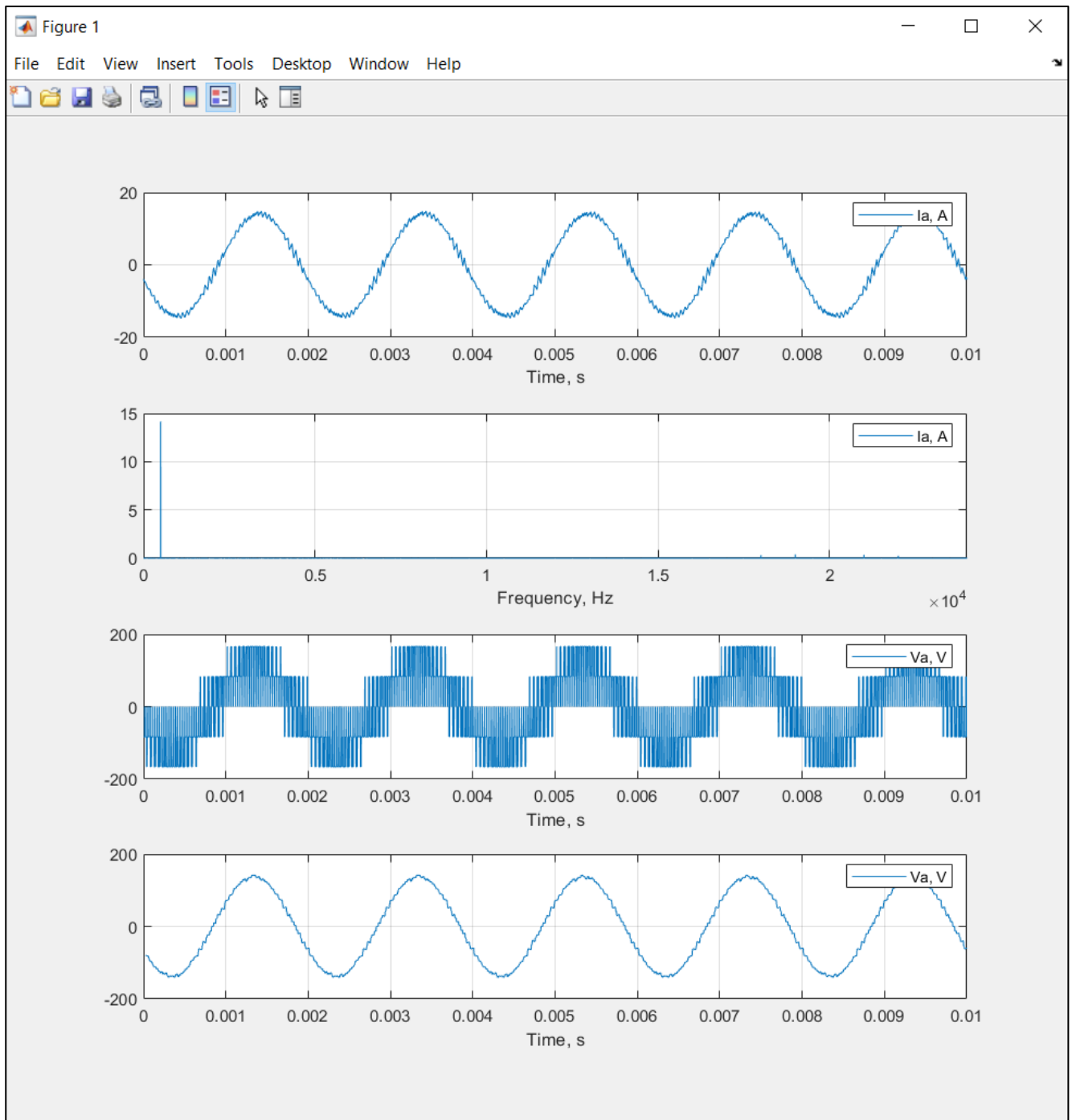




Figure 7.6. MATLAB figure window with plots of the selected quantities.

8. DYNAMIC FINITE ELEMENT ANALYSIS

Dynamic FE Analysis is the most powerful and most accurate of all the analysis methods taking into account rotation of the rotor, eddy currents, higher harmonics, PWM switching and etc. It simulates the machine connected to a power supply electronic circuit with the time-stepping transient finite element method. By default, the electrical circuit is a three-phase inverter as shown in Figure 10.1, but any power supply circuit can be used (see chapter 10). However, this type of analysis takes considerably more time comparing with other analysis types.

8.1. Running Dynamic FE Analysis.

The view of the main window when **Dynamic FE Analysis** is chosen is shown in Figure 8.1. To start the simulation, click the **Start dynamic FE simulation** button  on the MotorXP-AFM main window toolbar. The first run of the simulation begins at zero time. Every subsequent run of the simulation resumes from the time where it was previously paused. The simulation continues, until you pause the simulation, until the simulation reaches **Simulation stop time** or until an error occurs. The first run of the simulation starts with the initialization which may take a few minutes. To pause the simulation, click the **Pause simulation** button  which is to the right of the **Start dynamic FE simulation** button. The simulation will be paused when the calculation of the current time step is completed. It is not recommended to interrupt the simulation using Ctrl+C shortcut otherwise the project file may be corrupted, and all data will be lost.

Solver type specifies whether the linear or nonlinear FEA is used. Using linear solver type is recommended only for testing purposes. **Convergence tolerance** specifies the accuracy of FEA solution as defined by Exp. 2.5.

Simulation settings field specifies either the default **Simulation script file** and the default **Stator electrical circuit file** are used (if *General* is chosen) or the simulation script and the stator circuit are defined by the user (if *Advanced* is chosen). **Simulation script file** is a MATLAB-function which is called on each simulation time step and allows you to change all general simulation settings, compute and store user defined variables, control power supply sources and electronic switches, implement user defined motor control algorithms. Refer to chapter 9 for more details on using simulation script files. **Stator electrical circuit file** is a MATLAB-function defining the coupled electrical circuit which is solved simultaneously with the magnetic field. Refer to chapter 10 for more details on using stator electrical circuits.

Be aware that the **Time step** value should be small enough comparing with the PWM sampling period to get the accurate results. Control the **Discretization error** in the **Time Average Quantities** window shown in Figure 8.3 to adjust the time step. It is recommended that the discretization error does not exceed 1%.

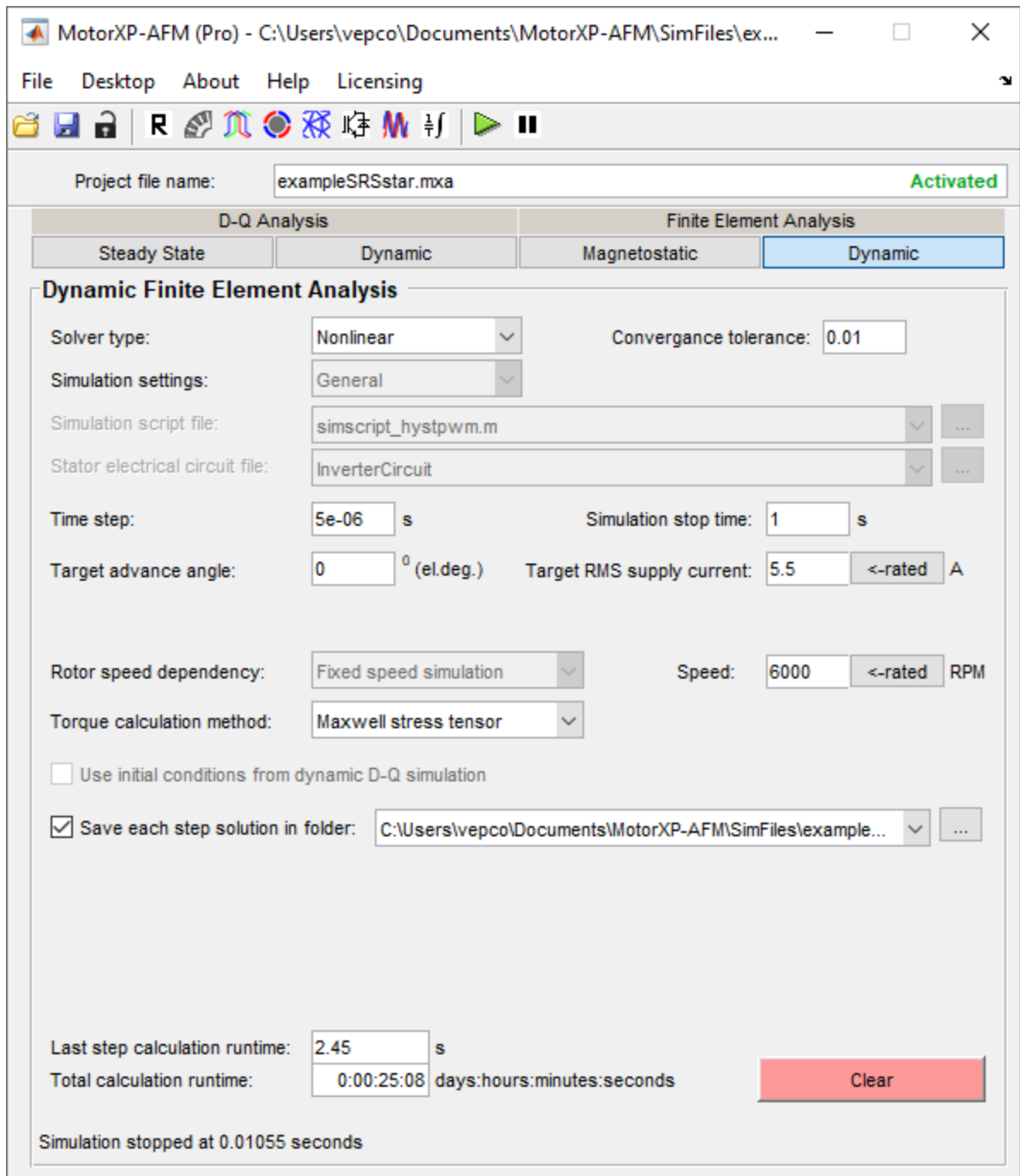


Figure 8.1. MotorXP-AFM main window for Dynamic FE Analysis.

The **Target RMS supply current** field value defines the desired supply current of the machine drawn from the inverter. The phase current in each winding will depend on the supply current I_s , on the stator winding connection (star or delta) and on the connection between winding layouts if there are two winding layouts (two stators, each of them having its own winding, or one stator having two windings), as specified by the **Layouts connection** pop-up menu of **Winding Editor** (*1-2* selection from the **Layouts connection** pop-up menu specifies that two winding layouts are connected in series, whereas *1//2* selection specifies that two winding layouts are connected in parallel). The example of phase current calculation in each

winding for all possible cases is given in Table 8.1. Examples of stator winding configurations for star and delta connections and for different numbers of winding layouts are shown in Figure 4.14.

Table 8.1. Phase current per one winding depending on the winding connection and connection between windings.

	One winding	Two winding layouts connected in series (<i>I-2</i>)	Two winding layouts connected in parallel (<i>I//2</i>)
Star connection	I_s	I_s	$I_s/2$
Delta connection	$I_s/\sqrt{3}$	$I_s/\sqrt{3}$	$I_s/\sqrt{3}/2$

I_s - RMS supply current

Since the supply current corresponds to a line current, for the star connection the phase current is equal to the supply current divided by the number of winding layouts connected in parallel, for delta connection the phase current is equal to the supply current divided by the number of winding layouts connected in parallel and divided by $\sqrt{3}$ (see table 8.1).

Target advance angle determines the desired angle between the current phasor and q-axis of the rotor as shown in Figure 2.6. For the current hysteresis and space vector PWM drive types the current control algorithm is applied to adjust the inverter voltage in order to maintain d-axis and q-axis current components defined by the **Target RMS supply current** and **Target advance angle** field values. For the space vector PWM the field oriented current control is used, for the current hysteresis PWM the hysteresis (bang-bang) current control is used. For the six-step drive the current is not controlled – the current is determined by the DC voltage, rotor speed, switch duty cycle (120 or 180 electrical degrees) and commutation advance angle.

Rotor speed dependency – specifies whether the rotor speed is fixed (when *Fixed speed simulation* is chosen) or varies depending on the load and electromagnetic torque of the motor (when *Variable speed simulation* is chosen). If *Fixed speed simulation* is chosen you should also specify the speed in the field appearing to the right. If *Variable speed simulation* is chosen, the initial speed should be specified.

If simulation with variable speed is used the load should be specified. The load can be specified either by a single value entered in the **Load** field or different load values can be specified for each time interval using the **Load profile** window as shown in Figure 8.2. Load can be a function of time or speed. Speed can be measured in rad/s or in RPM. Use +/- buttons to add or delete the time interval.

Button ‘<-rated’ to the right of some fields allows you to set up the corresponding parameter to its rated value specified in the **Rated Data** window.

Torque calculation method – two torque calculation methods are available: *Maxwell stress tensor* and *Virtual work*. Maxwell stress tensor method is usually used. Refer to section 2.3 for more details.

If **Use initial conditions from dynamic D-Q simulation** is checked the FE simulation is started with the initial state (initial values of magnetic field, currents and voltages) computed using the dynamic D-Q model so the dynamic FE simulation reaches the steady-state in less number of time steps.

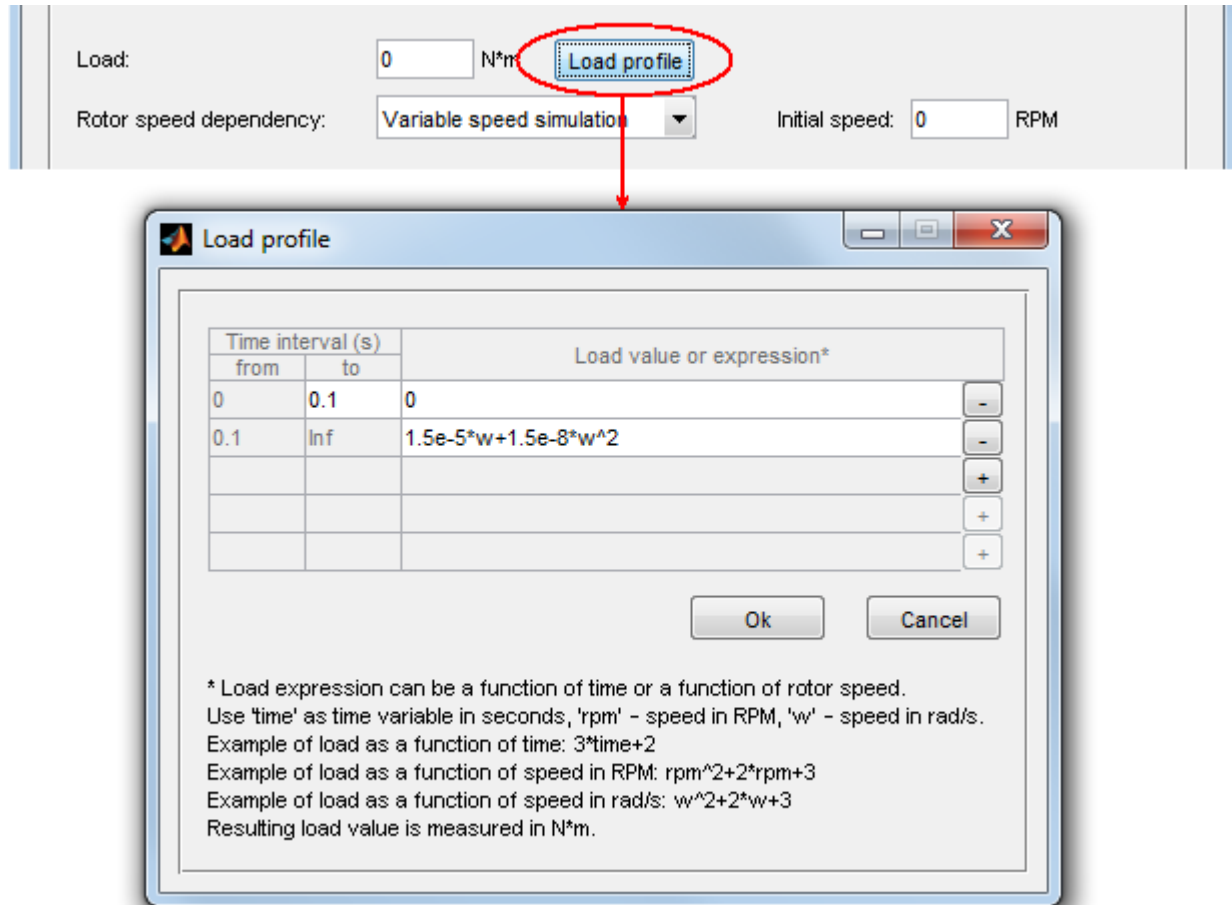


Figure 8.2. Specifying the load for the dynamic FE simulation.

Save each field solution checkbox enables (if checked) to store additional data of each FEA solution such as magnetic vector potential distribution and permeability distribution values. These data are not saved by default because of large amount of hard drive space required. Data for each time step are saved in a separate file so the number of files saved is equal to the number of computed time steps.

Use **Clear** button to delete all dynamic FEA simulation data and start the simulation from zero time.

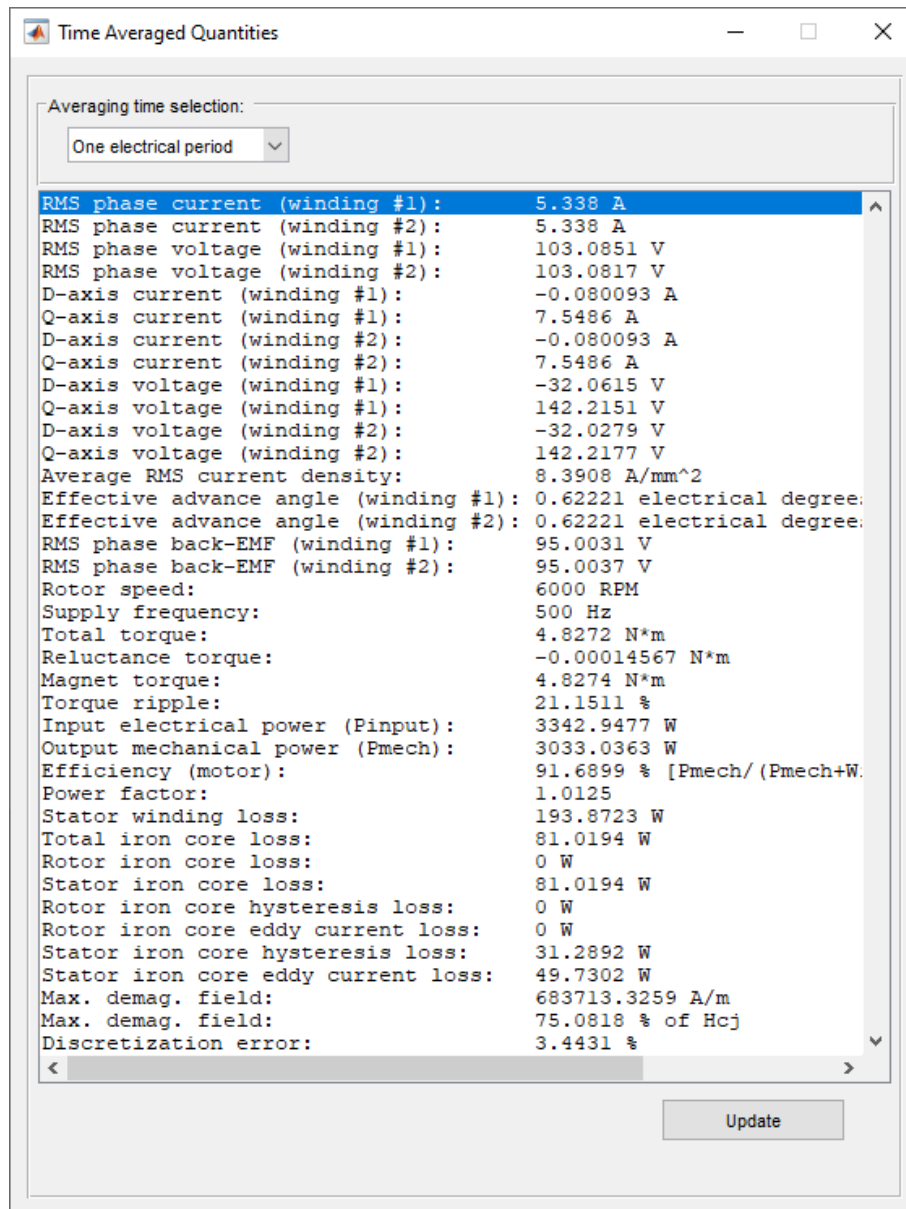



Figure 8.3. Time averaged quantities.

8.2. Viewing Dynamic FE Analysis results.

It is possible to view the **Dynamic FE Analysis** results as time-averaged quantities or plots.

8.2.1. Time-averaged quantities.

Use  toolbar button to view time-averaged quantities as shown in Figure 8.3. The displayed quantities can be averaged over one, two or three electrical periods or the averaging time can be defined directly choosing *User choice* from the **Averaging time selection** pop-up menu.

For more details on the interpretation of the maximum demagnetizing field results see section 2.7.

8.2.2. Plot wizard.

The view of the **Plot Wizard** window when **Dynamic FE Analysis** is chosen is shown in Figure 8.4. Several plot types are available for **Dynamic FE Analysis**:

- time-series data plot and frequency spectrum of the time-series data;
- air gap distribution plot and frequency spectrum of the air gap distribution;
- cross-section distribution plot;
- animation.

Plot Wizard also allows saving waveforms to a CSV-file (text file with comma-separated values) or to a Microsoft® Excel® spreadsheet file (Figure 8.4).

8.2.3. Time plots.

Time-series data plots are available from the **Time plot** panel of the **Plot Wizard** window. It is organized as a standard MATLAB plotting construction consisting of a set of `subplot` and `plot` functions. **Subplot** checkboxes allow you to control the number of axes or rectangular panes displayed within a current figure window. The corresponding axes are activated or deactivated by a mouse click within a cell of the **Subplot** column. When the subplot is activated, the **change** button allows you to choose quantities to be plotted into the selected axes. Quantities can be chosen from the dialog shown in Figure 8.5. Use Ctrl key to select several quantities. The list of available quantities is given below:

- Stator phase current (phase A, B, C) for each winding (if there are several windings) and for each parallel path (if there are several parallel paths);
- Stator phase current (d-axis, q-axis) for each winding (if there are several windings);
- Phase voltage (phase A, B, C) for each winding (if there are several windings);
- Phase voltage (d-axis, q-axis) for each winding (if there are several windings);
- Effective advance angle for each winding (if there are several windings);
- Back-EMF (phase A, B, C) for each winding (if there are several windings);
- Back-EMF (d-axis, q-axis) for each winding (if there are several windings);
- Input apparent electrical power;
- Consumed apparent power;

- Apparent power (real and reactive) consumed by a stator circuit;
- Mechanical power on the rotor shaft;
- Time derivative of the magnetic field energy;
- Electromagnetic torque;
- Load torque on the motor shaft;
- Rotor speed;
- Rotor angular position;
- Electromagnetic torque by Maxwell stress tensor;
- Electromagnetic torque by virtual work method;
- Electromagnetic torque by flux linkage and current;
- Magnet torque (by Maxwell stress tensor);
- Reluctance torque (by Maxwell stress tensor);
- Torque in each air gap (by Maxwell stress tensor) if there are several air gaps;
- Cogging torque (by Maxwell stress tensor);
- Flux linkage (phase A, B, C) for each winding (if there are several windings);
- Flux linkage (d-axis, q-axis) for each winding (if there are several windings);
- Electromagnetic normal force on each rotor;
- Electromagnetic normal force on each stator;

Plot Wizard: Dynamic FE Analysis

Time plot

Subplot	Plot function or Matlab expression	Plot type	X-axis limits	Y-axis limits
<input checked="" type="checkbox"/> 1	plot(time, Ia1); legend('Ia1, A'); change	waveform	0.01	
<input checked="" type="checkbox"/> 2	plot(time, Ia1); legend('Ia1, A'); change	spectrum	0 20000	0.2
<input checked="" type="checkbox"/> 3	plot(time, Va1); legend('Va1, V'); change	waveform	0.01	-230 230
<input checked="" type="checkbox"/> 4	plot(time, Va1); legend('Va1, V'); change	smoothed...	0.01	
<input type="checkbox"/>	change	waveform		

☒ Synchronize subplot time-axis limits Smoothed waveform averaging interval: 0.1 ms Plot

FFT spectrum averaging time selection: User choice from (s): 0 to (s): 1 >>

Air gap distribution plot

Subplot	Variables	Time (s)	Air gap #	Slice	Plot type	X-axis limits	Y-axis limits
<input checked="" type="checkbox"/> 1	Bn	< 0.010550 >>	2	2	distribution		
<input checked="" type="checkbox"/> 2	Bn	< 0.010550 >>	2	2	spectrum	50	
<input type="checkbox"/>		< 0.010550 >>	1 (bot...	1	distribution		
<input type="checkbox"/>		< 0.010550 >>	1 (bot...	1	distribution		
<input type="checkbox"/>		< 0.010550 >>	1 (bot...	1	distribution		

☒ Show graph legend Plot

Cross-section distribution plot

Figure	Plotted quantity	Time (s)	Slice	Options	X-axis limits	Y-axis limits	Z-axis limits
<input checked="" type="checkbox"/> 1	Magnetic flux density, [T]	< 0.010550 >>	2	flux lines			
<input checked="" type="checkbox"/> 2	Relative permeability	< 0.010550 >>	2	none			
<input checked="" type="checkbox"/> 3	Stator current density, [A/m^2]	< 0.010550 >>	2	flux lines			
<input type="checkbox"/>	None	< 0.010550 >>	1	none			
<input type="checkbox"/>	None	< 0.010550 >>	1	none			

Number of flux line levels: 20 ☐ Full cross-section view Plot

Animated plot

☒ Animate air gap distribution subplots Skip: 0 files Animation start time: 0.000005 s

☒ Animate cross-section distribution figures Frame display time: 0 ms Animation stop time: 0.010550 s

☒ Position figures Pause Start animation

Data source folder: C:\Users\wepco\Documents\MotorXP-AFM\SimFiles\exampleSRSstar_dynFEdata ...

Plot

Plot and save waveforms to a CSV-file
Plot and save waveforms to Excel

Figure 8.4. Dynamic FE Analysis Plot Wizard window and export data to a file options.

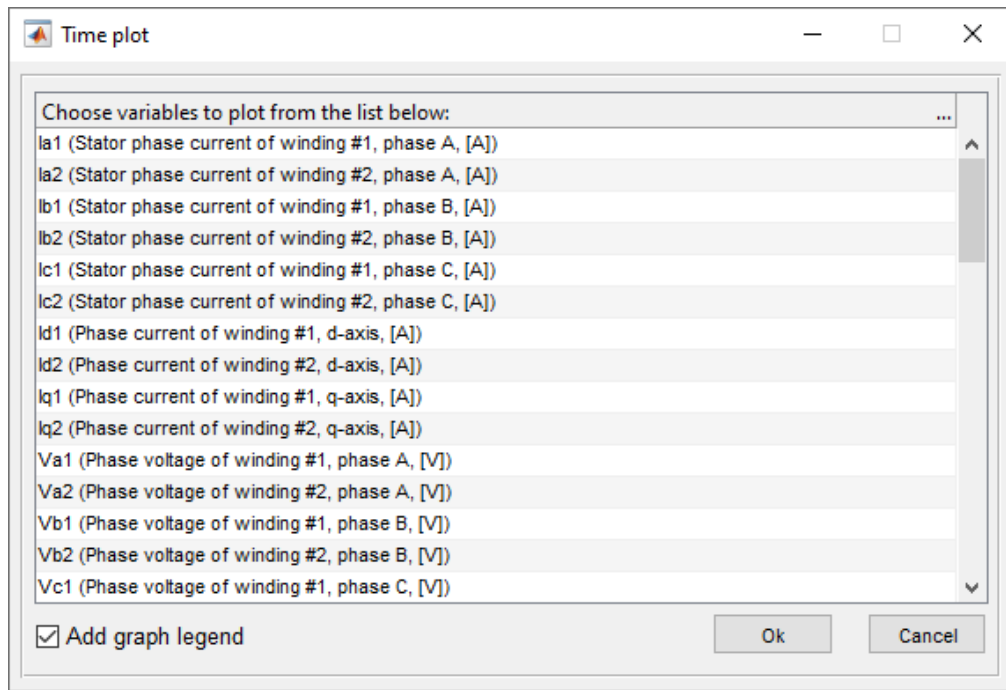


Figure 8.5. Choosing quantities to be plotted.

All user defined variables stored in the `Userdata` structure appear at the end of the list shown in Figure 8.5. For more details on saving your own variables and using the `Userdata` structure refer to section 9.4.

By clicking the **OK** button of the dialog (Figure 8.5), the MATLAB-expression is constructed to plot the selected quantities appearing in the corresponding line as shown in Figure 8.6 for plotting phase A stator current of winding #1 (first line), spectrum of phase A stator current of winding #1 (second line), phase A stator voltage of winding #1 (third line) and phase A stator voltage (winding #1) waveform after smoothing (fourth line).

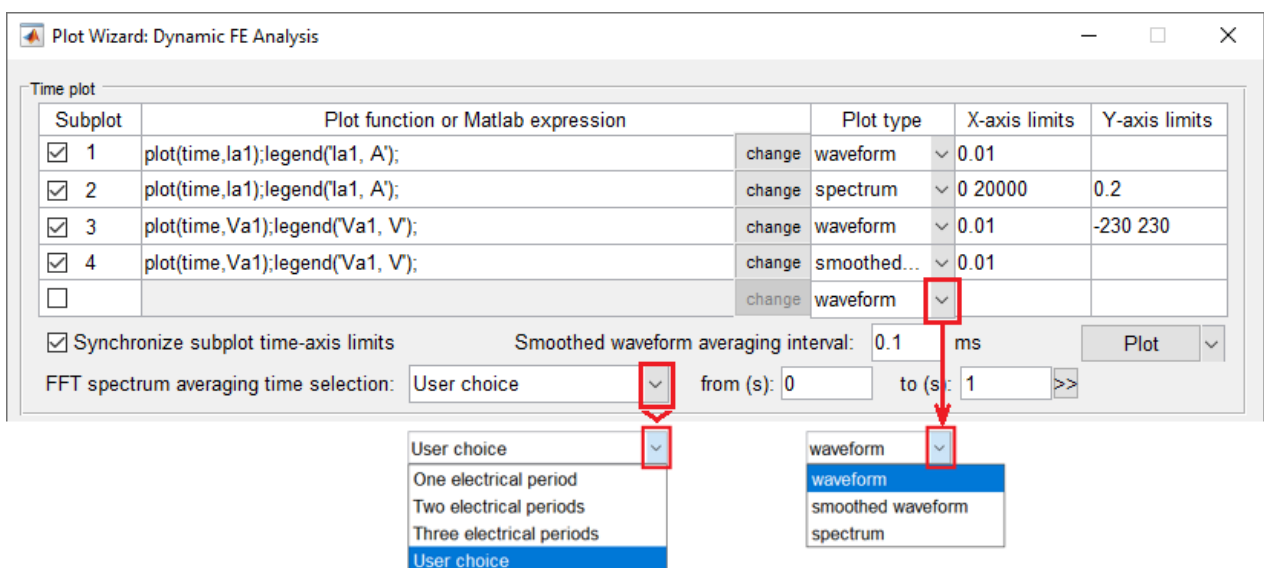


Figure 8.6. Example of using **Plot Wizard** for creating time plots.

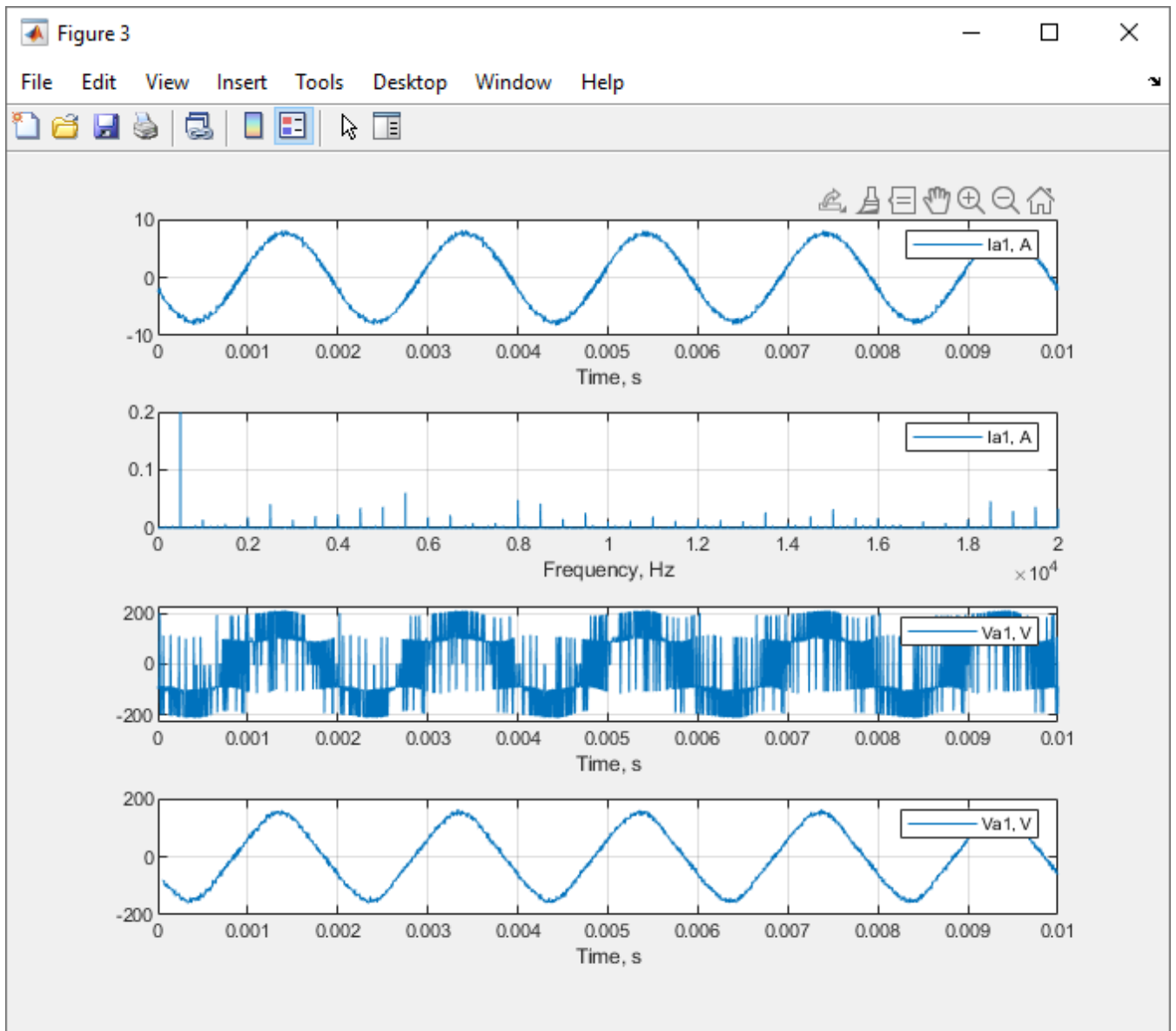


Figure 8.7. MATLAB figure window with time plots of the selected quantities.

When the **Plot** button is clicked, the MATLAB figure window with plots of the selected quantities will appear (see Figure 8.7).

As it is seen, the plotting expression consists of the `plot` function with selected variables used as input arguments. If the **Add graph legend** checkbox of the dialog is checked, the `legend` function is added to the plotting expression so the graph legend of the corresponding axes will be shown. All plotting expressions are editable, so you can use any MATLAB plotting options and functions to change the way the plots appear on the screen. You can also change `time` variable to any other variable you would like to use as an input argument of the `plot` function. There is also a list of additional variables which can be used within a plotting expression (see section 9.3).

X-axis limits and **Y-axis limits** fields of the **Time plot** panel allow you to set the x-axis and y-axis limits, respectively, to the specified values. Two limit values within a cell are separated by a space, comma ‘,’

or semicolon ‘;’. If a cell is empty, the limits of the corresponding axis will be chosen automatically. If the **Synchronize subplot time-axis limits** checkbox is checked, all subplots of the figure will have identical limits along the time-axis when you zoom or pan one of the subplots of the figure.

There are several plotting options available from the **Plot type** pop-up menu: *waveform*, *smoothed waveform* and *spectrum* (see Figure 8.6). Smoothed waveform plot type can be useful when plotting the pulse width modulated waveforms to filter out the PWM carrier frequency and obtain the original waveform of the signal (compare subplots 3 and 4 of Figure 8.7). The **Smoothed waveform averaging interval** field specifies the sliding window width of the smoothing algorithm. The resolution of the frequency spectrum, plotted when *spectrum* is chosen from the **Plot type** pop-up menu, is controlled by the **FFT spectrum averaging time selection** pop-up menu. The spectrum can be calculated over one, two or three electrical periods or the time can be defined directly choosing *User choice* from the **FFT spectrum averaging time selection** pop-up menu (see Figure 8.6).

8.2.4. Air gap distribution plots.

Air gap distribution plots allow you to view the distribution of the particular quantity over the machine’s air gap in the selected cylindrical cross-section (slice) as well as its frequency spectrum. It is available from the **Air gap distribution plot** panel. **Subplot** checkboxes allow you to control the number of axes or rectangular panes displayed within a current figure window. The corresponding axes are activated or deactivated by a mouse click within a cell of the **Subplot** column. When the subplot is activated, the “+” button allows you to choose quantities to be plotted from the dialog shown in Figure 8.8. Use Ctrl key to select several quantities.

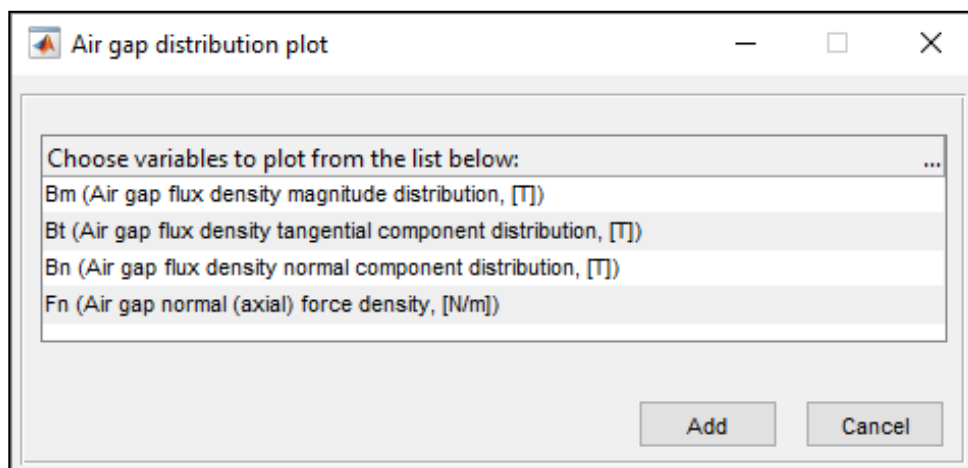


Figure 8.8. Choosing quantities to be plotted.

The following quantities are listed in the dialog of Figure 8.8:

Quantity	Comments
Bm (Air gap flux density magnitude distribution, [T])	Defined as $B_m = \sqrt{B_t^2 + B_n^2}$ where B_n and B_t – normal and tangential components of the magnetic flux density in the center of the air gap, as shown in Figure 2.4.
Bt (Air gap flux density tangential component distribution, [T])	B_n and B_t – normal and tangential components of the magnetic flux density in the center of the air gap, as shown in Figure 2.4.
Bn (Air gap flux density normal component distribution, [T])	
Fn (Air gap normal (radial) force distribution, [N])	According to Maxwell stress tensor method the normal force in each point of the round path can be expressed as the following: $F_n = -\frac{B_n^2 - B_t^2}{2\mu_0} d \cdot l$ where l – lamination length in z direction, d – length of the path in the center of the air gap between two consecutive points, μ_0 – permeability of the free space, B_n and B_t – normal and tangential components of the magnetic flux density in the center of the air gap, as shown in Figure 2.4.

By clicking the **Add** button of the dialog (Figure 8.8), the selected quantities are added to the corresponding line separated by commas as shown in Figure 8.9. Each line of the **Variables** column is editable, so you can use MATLAB arithmetic operations to obtain desired plots. For example, the second and third lines in Figure 8.9 produce plots of the tangential air gap force density distribution and the radial air gap force density distribution, respectively, where μ_0 – variable corresponding to the permeability of free space. According to Exp. 2.6 the tangential air gap force produces the electromagnetic torque so the plot of the electromagnetic torque distribution over an air gap can be obtained.

Air gap distribution plot									
Subplot	Variables		Time (s)		Air gap #	Slice	Plot type	X-axis limits	Y-axis limits
<input checked="" type="checkbox"/> 1	Bm, Bt, Bn	+ <	1	> >>	1 (bot...	1	distribution		
<input checked="" type="checkbox"/> 2	Bn.*Bt/mu0	+ <	1	> >>	1 (bot...	1	distribution		
<input checked="" type="checkbox"/> 3	Fn	+ <	1	> >>	1 (bot...	1	distribution		
<input type="checkbox"/>		+ <	1	> >>	1 (bot...	1	distribution		
<input type="checkbox"/>		+ <	1	> >>	1 (bot...	1	distribution		
<input checked="" type="checkbox"/> Show graph legend								Plot <input type="button" value="Plot"/>	

Figure 8.9. Example of using **Plot Wizard** for creating air gap plots.

Time fields of the **Air gap distribution plot** panel allow you to specify the time point which the selected quantities are plotted for. By default, last computed time point is set up. Plotting for the time points other than the last computed time point is only possible, if the file containing data for the desired time point was previously saved. These data-files are being saved when **Save each field solution** is checked in the MotorXP-AFM main window (Figure 8.1). So, if you are interested in viewing the air gap distribution plots for different time points make sure that the corresponding data-files are being saved. The folder used as a source of data-files is specified in the **Data source folder** field located at the bottom of the **Plot Wizard** window (Figure 8.4). You can change it by clicking the button to the right, if needed.

X-axis limits and **Y-axis limits** fields allow you to set the x-axis and y-axis limits, respectively, to the specified values in the same way as for the **Time plot** panel. **Slice** fields allow you to choose the machine's cylindrical cross-section which the selected quantities are plotted for, if several slices are used (see section 2.4 for more details on multi-slice simulations). If the machine has several air gaps the **Air gap #** field selects the air gap number starting from the bottom of the machine which the selected quantities are plotted for.

Besides the air gap distribution, it is also possible to calculate the air gap harmonic components of the selected quantities. To obtain the frequency spectrum, select the *spectrum* item in the corresponding **Plot type** pop-up menu. An example of plotting the air gap distribution of the normal component of the magnetic flux density and its spectrum is shown in Figure 8.10. Only first 50 harmonics are shown, since the value specified in the corresponding **X-axis limits** field is 50 (if the minimum limit equals to 0, it can be omitted).

Show graph legend field allows you to choose whether the graph legend will be displayed.

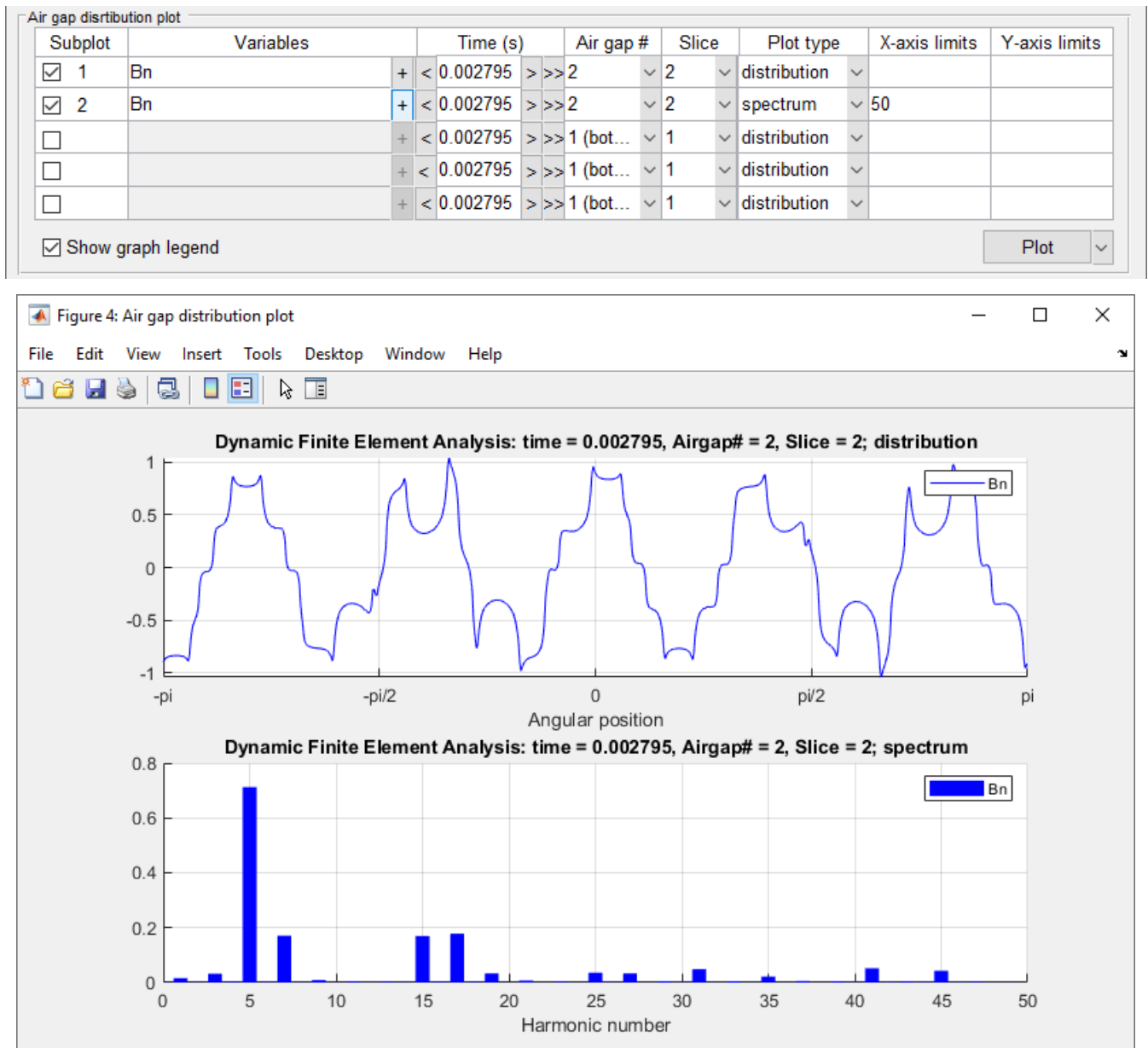


Figure 8.10. Plotting the air gap distribution of the normal flux density component and its spectrum in second air gap in second mesh slice.

8.2.5. Cross-section distribution plots.

Cross-section distribution plots are available from the **Cross-section distribution plot** panel of the **Plot Wizard** window (Figure 8.4) and allow you to view the distribution of the particular quantity in the selected cylindrical cross-section (slice) of the machine. As opposed to two previous plot types, the cross-section distribution for each quantity is plotted in a separated window which contains only one axes. **Figure** checkboxes allow you to control the number of windows appearing when the **Plot** button is clicked. The corresponding figure is activated or deactivated by a mouse click within a cell of the **Figure** column. When the figure is activated, the corresponding **Plotted quantity** pop-up menu allows you to choose the quantity to be plotted. If **None** is chosen, the machine's cross-section geometry will be plotted.

The following items are available from the **Plotted quantity** pop-up menu:

- Magnetic vector potential, [T*m];
- Magnetic flux density, [T];
- Magnetic field intensity, [A/m];
- Relative permeability;
- Stator current density, [A/m²];
- Squared flux density, [T];
- Magnetic field energy density, [J/m³];
- Stator loss density, [W/m³];
- Iron loss density, [W/m³];
- Total loss density (stator+iron), [W/m³];
- Demagnetizing field, [A/m];
- Demagnetizing field, [% of H_{cj}];
- Finite element mesh.

Time fields of the **Cross-section distribution plot** panel allow you to specify the time point which the selected quantity is plotted for. By default, last computed time point is set up. Plotting for the time points other than the last computed time point is only possible, if the file containing data for the desired time point was previously saved. These data-files are saved when **Save each field solution** is checked in the MotorXP-AFM main window (Figure 8.1). So, if you are interested in viewing the cross-section distribution plots for different time points make sure that the corresponding data-files are being saved. The folder used as a source of data-files is specified in the **Data source folder** field located at the bottom of the **Plot Wizard** window (Figure 8.4). You can change it by clicking the button to the right, if needed. **Slice** fields allow you to choose the machine's cylindrical cross-section which the selected quantity is plotted for, if several slices are used (see section 2.4 for more details on multi-slice simulations).

Options field allows you to show the magnetic flux lines (if *flux lines* is chosen) or magnetic flux arrows (if *flux arrows* is chosen) on the corresponding plot. Flux arrows are plotted such that the direction of the arrow indicates the direction of the flux and the size of the arrow indicates the magnitude of the flux density. **Number of flux line levels** field allows you to alter the flux lines density. If periodic/antiperiodic boundary conditions are used, by default, only part of the cross-section (as it appears in **Mesh Editor**) will be plotted. Check the **Full cross-section view** checkbox if you want to plot the whole cross-section.

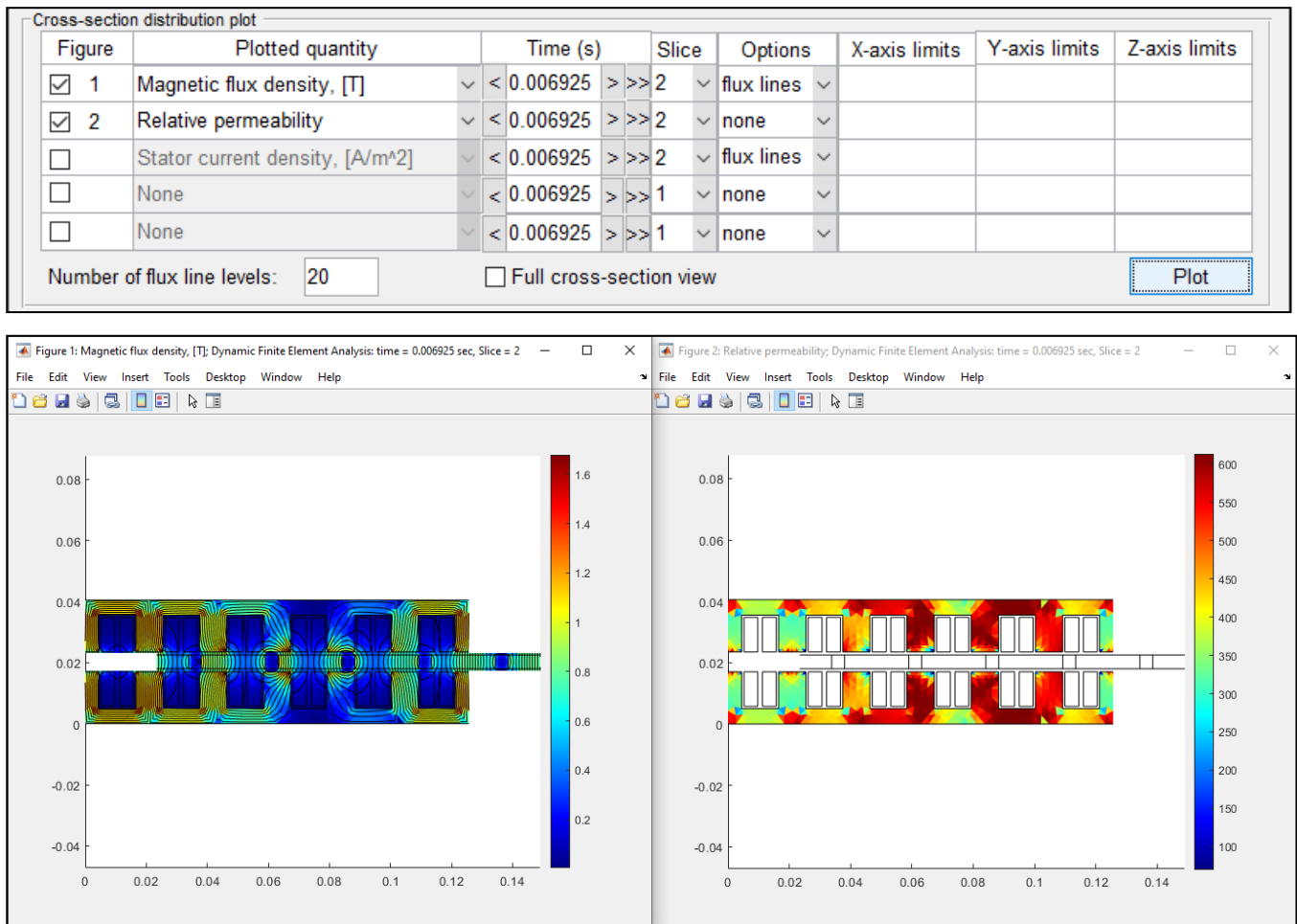


Figure 8.11. Example of using **Plot Wizard** for creating cross-section distribution plots.

X-axis limits and **Y-axis limits** fields allow you to set the x-axis and y-axis limits, respectively. If x-axis and y-axis limits are not specified, their values will be determined by the size of the machine's cross-section. **Z-axis limits** field sets the color scale limits to specified minimum and maximum values separated by a space, comma ',', or semicolon ';'. Values between z-axis limits are linearly mapped to the used color scale (colormap). Data values smaller or greater than specified z-axis limits are mapped to the minimum limit or to the maximum limit, respectively.

An example of plotting the cross-section distribution of the magnetic flux density and relative permeability is shown in Figure 8.11. When the **Plot** button is clicked, two windows appear. As it is seen, the **flux lines** option is chosen for the magnetic flux density, so the flux lines are additionally shown. Since the third figure is not active, the Magnetic field intensity is not plotted. Note that only part of the machine's cylindrical cross-section is shown, since **Full cross-section view** is not checked.

8.2.6. Animation.

Animated plot panel is used to create animated sequences of the air gap distribution plots and/or the cross-section distribution plots (Figure 8.4). **Animate air gap distribution subplots** and **Animate cross-section distribution figures** checkboxes allow you to choose plot types to be animated. If **Animate air**

gap distribution subplots is checked, all selected subplots of the **Air gap distribution plot** panel will be involved in the animation. Similarly, if **Animate cross-section distribution figures** is checked, all selected figures of the **Cross-section distribution plot** panel will be involved in the animation. If **Position figures** control is checked, the size and location on the screen for all figure windows will be determined automatically so that the windows fit best the screen size.

Skip and **Frame display time** fields allow you to specify the way how frames change each other while the animation is in progress. **Skip** field specifies the number of data-files or frames to be skipped. So, if **Skip** field is set to 0, data for each time step will be shown on the screen, if **Skip** field is set to 1, data for each second time step will be shown, if **Skip** field is set to 2 – for each third time step and so on. **Frame display time** field specifies the time each frame is displayed on the screen. Note that the least time the frame is displayed is limited by the animation function runtime. So, if the **Frame display time** field is 0, the actual time the frame is displayed on the screen will be the least possible in this case.

Animation start time and **Animation stop time** fields set up the time interval for the animation. **Data source folder** field specifies the folder with data-files to be animated. The same folder is used for animations and for air gap distribution and cross-section distribution plots. Using animation is only possible, if the files containing data for the time interval to be animated were previously saved.

To start the animation, click the **Start animation** button. The current time point, animated quantity and other information is displayed at the top of each window involved in the animation. The animation continues until the time specified in the **Animation stop time** field is reached or until all windows involved in the animation are closed. You can also pause the animation using the **Pause** button. While the animation is in progress or paused, you can use all MATLAB figure tools, for example, changing the scale and position of the plots.

9. DYNAMIC FE ANALYSIS SIMULATION SCRIPT FUNCTIONS

A simulation script is a file with .m extension containing MATLAB-function of a specific format. While the dynamic FEA simulation is running this function is called on each simulation time step and allows to automatically change all simulation settings, compute and store your own variables, control power supply sources and electronic switches, implement motor control algorithms and etc. For example, the simulation script function is used to control electronic switches (IGBT, etc.) to implement PWM switching sequence. To understand this chapter, some familiarity with MATLAB programming is required. Refer to MATLAB help documentation for more details on MATLAB programming.

9.1. General information.

MotorXP-AFM offers you a variety of options to properly set up your simulation and in most cases, there is no need to use simulation scripts. On the other hand, simulation scripts give you much more flexibility in using MotorXP-AFM. Note that simulation scripts are used only for **Dynamic FE Analysis** and available only in MotorXP-AFM **MATLAB** version. Refer to section 3.2 for more details on limitations of MotorXP-AFM Standalone version.

The chosen simulation script file is displayed in the **Simulation script file** field of the MotorXP-AFM main window when **Dynamic FE Analysis** is chosen (see Figure 8.1). There are three simulation scripts used by default: *simscript_hystpwm.m*, *simscript_spacevecpwm.m*, *simscript_sixstep.m* which can be found in *[MotorXP-AFM installation directory]\SimScripts*. If the **Simulation settings** field is set to **General** (see Figure 8.1), the corresponding simulation script is chosen automatically depending on the drive type. If the **Simulation settings** field is set to **Advanced**, the simulation script should be specified by the user. Click the button to the right of the **Simulation script file** field to choose your own simulation script.

If the simulation script file is used all simulation settings displayed in the MotorXP-AFM main window can be overwritten with those specified in the simulation script function. For example, you can specify any RMS supply current or advance angle values regardless of those specified in the **Target RMS supply current** and **Target Advance angle** fields of the MotorXP-AFM main window.

9.2. Writing a simulation script function.

The definition of the simulation script function `simscript` and a minimum amount of code appearing in file *simscript.m* is as follows:

```
function [ShaftPosition,CircuitControl,Settings,Enforce,Userdata,updateCoefs3D] ...
= simscript(Outputs,Settings,Userdata,Circuit, Drive,Geometry,Mesh,...
            Windings,A,cell_p,cell_t,cell_Nu,path)
updateCoefs3D = 0;
ShaftPosition = [];
CircuitControl = [];
```



```
Enforce = [];
```

The first line of the function always starts with the keyword `function`. The names of the M-file and of the function should be the same. Refer to MATLAB help documentation for more details on writing MATLAB-functions.

Be aware that the presented example of the simulation script function will not work with stator electrical circuit file *InverterCircuit.m* since states of switches are not defined.

The function's output arguments:

`ShaftPosition` – reserved for future releases. For now, `ShaftPosition = []`.

`CircuitControl` – structure containing current and voltage values which are supplied by current and voltage courses as well as states of electronic switches. See chapter 10 for more details on using current and voltage sources and electronic switches.

`Settings` – structure of simulation settings, the same as input argument `Settings`.

`Enforce` – structure to control rotor speed. If `Enforce=[]`, when rotor speed is computed depending on the load and electromagnetic torque (variable speed simulation). To set the rotor speed to some fixed value use the statement `Enforce.speed=fixedspeed`, where `fixedspeed` is the rotor speed in rad/s.

`Userdata` – structure to store user data, the same as input argument `Userdata`. By default, `Userdata` is empty. Refer to section 9.4 to figure out how to use the `Userdata` structure to store your own data using simulation script functions.

`updateCoefs3D` – variable determining whether to update the coefficients for taking into account 3D effects on the current time step.

The function's input arguments:

`Outputs` – structure containing the dynamic FEA simulation results.

`Settings` – structure containing the dynamic FEA simulation settings.

`Userdata` – structure to store user data.

`Circuit` – structure containing stator electrical circuit parameters.

`Drive` – structure containing drive parameters.

`Geometry` – structure containing the machine's dimensions data.

`Mesh` – structure containing mesh data.

`Windings` – structure containing the machine's windings data.

`A` – array of magnetic vector potential node values for the current time step for all slices (if several slices are used; see section 2.4).

`cell_p` – cell-array containing x and y coordinates of finite element mesh nodes for every slice, as it is described in MATLAB PDE Toolbox help documentation (see help on `initmesh` function for more details).

`cell_t` – cell-array containing finite elements data for every slice, as it is described in MATLAB PDE Toolbox help documentation (see help on `initmesh` function for more details).

`cell_Nu` – cell-array containing midpoint magnetic reluctivity values for every slice. Magnetic reluctivity is $\nu = \frac{1}{\mu_0 \mu_r}$, where μ_0 – permeability of the free space, μ_r – relative permeability.

`path` – directory with application files.

9.3. Main data structures.

Outputs structure fields:

Field	Size, type	Units	Description
<code>Outputs.CurrentTime</code>	1×1, double	second	Simulation current time point.
<code>Outputs.nPolePairs</code>	1×1, double		Number of pole pairs.
<code>Outputs.gamma0</code>	1×1, double	Electrical degree	Angle between a-phase axis and rotor q-axis for initial rotor position.
<code>Outputs.Va</code>	nLayouts×n, double	Volt	Instantaneous values of voltages measured at stator winding terminals for each winding layout; nLayouts – number of winding layouts (see section 4.3); n – number of computed time steps.
<code>Outputs.Vb</code>			
<code>Outputs.Vc</code>			
<code>Outputs.Vd</code>	nLayouts×n, double	Volt	Instantaneous values of d-axis and q-axis voltage components for each winding layout.
<code>Outputs.Vq</code>			
<code>Outputs.Ia</code>	(Npp*nLayouts)×n, double	Ampere	Current flowing in each parallel path of each phase of each winding layout; number of array rows corresponds to number of parallel paths multiplied by number of winding layouts; Npp – number of parallel paths.
<code>Outputs.Ib</code>			
<code>Outputs.Ic</code>			
<code>Outputs.Id</code>	nLayouts×n, double	Ampere	Instantaneous values of d-axis and q-axis current components for each winding layout.
<code>Outputs.Iq</code>			
<code>Outputs.time</code>	1×n, double	second	Time points.
<code>Outputs.Gamma</code>	nLayouts×n, double	Electrical degree	Instantaneous values of advance angle for each winding layout.
<code>Outputs.BackEMFa</code>	nLayouts×n, double	Volt	Instantaneous values of phase back-EMF for each winding layout.
<code>Outputs.BackEMFb</code>			
<code>Outputs.BackEMFc</code>			
<code>Outputs.BackEMFd</code>	nLayouts×n, double	Volt	Instantaneous values of d-axis and q-axis back-EMF components for each winding layout.
<code>Outputs.BackEMFq</code>			
<code>Outputs.Psi</code>	nCircuitNodes×n, double	Volt	Stator electrical circuit potentials. nCircuitNodes – total number of nodes of stator electrical circuits.
<code>Outputs.Torque</code>	1×n, double	N*m	Electromagnetic torque; equals to one of the variables <code>Torque_maxwell</code> or

			Torque_vwork depending on the selected torque calculation method.
Outputs.Load	1×n, double	N*m	Load torque on the motor shaft.
Outputs.Speed	1×n, double	rad/s	Rotor angular speed.
Outputs.Rotang	1×n, double	rad	Rotor angular position.
Outputs.Pinput	1×n, double	Watt	Input apparent power delivered by all current and voltage sources.
Outputs.Pcons	1×n, double	Watt	Consumed apparent power.
Outputs.Ps	1×n, double	Watt	Apparent power (real and reactive) consumed by the stator electrical circuit.
Outputs.Pmf	1×n, double	Watt	Magnetic energy time derivative.
Outputs.Pmech	1×n, double	Watt	Mechanical power on the shaft.
Outputs.Piron_hyst_stator	1×n, double	Watt	Stator hysteresis iron loss.
Outputs.Piron_eddy_stator	1×n, double	Watt	Stator eddy current iron loss.
Outputs.Piron_hyst_rotor	1×n, double	Watt	Rotor hysteresis iron loss.
Outputs.Piron_eddy_rotor	1×n, double	Watt	Rotor eddy current iron loss.
Outputs.Torque_maxwell	1×n, double	N*m	Electromagnetic torque calculated with the Maxwell stress tensor method.
Outputs.Torque_vwork	1×n, double	N*m	Electromagnetic torque calculated with the Virtual work method.
Outputs.Torque_fluxlinkage	1×n, double	N*m	Electromagnetic torque calculated by currents and flux linkages using Exp. 2.10.
Outputs.Torque_magnet	1×n, double	N*m	Magnet component of electromagnetic torque.
Outputs.Torque_reluctance	1×n, double	N*m	Reluctance component of electromagnetic torque.
Outputs.Fluxlinkage_a	nLayouts×n, double	Weber	Instantaneous values of flux linkages for each winding layout.
Outputs.Fluxlinkage_b			
Outputs.Fluxlinkage_c			
Outputs.Fluxlinkage_d	nLayouts×n, double	Weber	Instantaneous values of d-axis and q-axis flux linkage components for each winding layout.
Outputs.Fluxlinkage_q			
Outputs.Fn_rotor	nRotors×n, double	N	Normal (axial) electromagnetic force on each rotor; nRotors – number of rotors.
Outputs.Fn_stator	nStators×n, double	N	Normal (axial) electromagnetic force on each stator; nStators – number of stators.
Outputs.Torque_airgap	nAirgaps×n, double	N	Electromagnetic torque in each air gap; nAirgaps – number of air gaps.
Outputs.DemagH_max	1×1, double	A/m	Maximum value of the permanent magnet demagnetizing magnetic field registered up to the current time point.
Outputs.DemagHpercent_max	1×1, double	% of H _{cj}	

Settings structure fields:

Field	Value	Description
Settings.DF_script	Directory and/or file name	Corresponds to the Simulation script file field of the main window.
Settings.DF_solvertype	'Nonlinear'	Corresponds to the Solver type field of the main window.
	'Linear'	
Settings.DF_tol	Number > 0	Corresponds to the Convergence tolerance field of the main window.
Settings.DF_statorcircuit	Function name	Name of the function specifying the stator electrical circuit; corresponds to the Stator electrical circuit file field of the main window.
Settings.DF_timestep	Number > 0	Simulation time step; corresponds to the Time step field of the main window.
Settings.DF_stoptime	Number > 0	Corresponds to the Simulation stop time field of the main window.
Settings.DF_Isrms	Number >= 0	RMS supply current to be supplied by inverter; corresponds to the Target RMS supply current field of the main window.
Settings.DF_Gamma	Number >= 0	Corresponds to the Target advance angle field of the main window.
Settings.DF_motorload	Number >= 0	Load torque on the motor shaft, corresponds to the Load field of the main window.
Settings.DF_InitSpeed	Number	Initial speed of the rotor; corresponds to the Initial Speed field of the main window when variable speed simulation is chosen.
Settings.DF_saveeachsolution	0	Corresponds to the Save each filed solution checkbox value of the main window.
	1	
Settings.DF_saveeachsolutionfolder	Directory	Folder to store data-files when Save each filed solution is checked.
Settings.DF_torquemethod	'Maxwell stress tensor'	Electromagnetic torque calculation method; corresponds to the Torque calculation method field of the main window.
	'Virtual work'	

Geometry structure fields:

Field	Value	Description
Geometry.lag	Number > 0	Air gap length; corresponds to the Air gap field of Geometry Editor .
Geometry.motortype	'stator-rotor'	Corresponds to the Machine type field of Geometry Editor .
	'stator-rotor-stator'	
	'rotor-stator-stator'	

Geometry.slotlayertype	'Single layer'	Determined either single layer or double layer winding is used; corresponds to the Winding layers field of Geometry Editor .
	'Double layer'	
Geometry.Ns	Number > 0	Corresponds to the Number of slots field of Geometry Editor .
Geometry.layerpos	Upper/Lower'	Determined either winding layers are oriented horizontally or vertically inside the slot; corresponds to the Layers orientation field of Geometry Editor .
	'Left/Right'	
Geometry.Ls	1×nStators, double	Corresponds to the Stator height field of Geometry Editor .
Geometry.Lr	1×nRotors, double	Corresponds to the Rotor height field of Geometry Editor .

Windings structure fields:

Field	Value	Description
Windings.Npp	Number > 0	Number of parallel paths of the stator winding per phase per layout; corresponds to the Number of parallel paths field of Winding Editor .
Windings.Lsew	Number > = 0	Leakage inductance of the stator winding end-turns per phase per layout; corresponds to the End winding inductance field of Winding Editor .
Windings.Rs	Number > = 0	Active DC resistance of the stator winding per phase per layout depending on the winding temperature; corresponds to the Phase resistance field of Winding Editor .
Windings.layout1	Cell arrays	Stator winding layout for each phase; corresponds to the Winding layout tables of Winding Editor .
Windings.layout2		
Windings.layout3		
Windings.W	Number > 0	Total number of conductors in one stator slot (for double layer winding – number of conductors in both layers of the slot).
Windings.nstrands	Number > 0	Corresponds to the Number of strands in hand field of Winding Editor .
Windings.fillfactor	0 < Number <= 1	Stator coil fill factor; corresponds to the Coil fill factor field of Winding Editor .
Windings.ks	Number > 0	Stator winding material conductivity depending on the winding temperature.

Windings.statorcircuit	'StarConnection'	Determines the stator winding connection; corresponds to the Winding circuit field of Winding Editor .
	'DeltaConnection'	
Windings.Hsew_outer	Number > 0	Corresponds to the Outer end turns radial overhang field of Winding Editor .
Windings.Hsew_inner	Number > 0	Corresponds to the Inner end turns radial overhang field of Winding Editor .
Windings.layoutmethod	'Automatic'	Stator winding layout input method; corresponds to the Winding layout method field of Winding Editor .
	'Manual'	
Windings.nPolePairs	Number > 0	Corresponds to the Number of pole pairs field of Winding Editor .
Windings.nLayouts	Number > 0	Number of stator winding layouts (see section 4.3);
Windings.statorconnection	[]	Determines the connection between winding layouts (see section 4.3); corresponds to the Layouts connection field of Winding Editor .
	'1 2'	
	'1-2'	
Windings.isToroidal	'0'	Determines whether the winding is toroidal or planar
	'1'	

Drive structure fields:

Field	Value	Description
Drive.Vdc	Number >= 0	Corresponds to the DC supply voltage (Vdc) field of Drive Settings .
Drive.DriveType	'Current hysteresis PWM'	Corresponds to the Drive type field of Drive Settings .
	'Space vector PWM'	
	'Six-step'	
Drive.Is_hyst_perc	Number > 0	Current hysteresis in percentage of RMS current; corresponds to the Current hysteresis field of Drive Settings .
Drive.fspwm	Number > 0	Corresponds to the PWM sampling frequency field of Drive Settings .
Drive.SwitchDutyCycle_sixstep	'120'	Corresponds to the Switch duty cycle field of Drive Settings .
	'180'	

Drive. CommutationAdvanceAngle _sixstep	Number > 0	Corresponds to the Commutation advance angle field of Drive Settings .
Drive. SixStepOptions	'General'	Corresponds to the Options field of Drive Settings .
	'Six-step with limited maximum current'	
	'Six-step with variable DC voltage'	
Drive. Is_max_sixstep	Number > 0	Corresponds to the Motor phase current limit field of Drive Settings .
Drive. Is_hyst_perc_sixstep	Number > 0	Phase current hysteresis in percentage of current limit; corresponds to the Phase current hysteresis field of Drive Settings .
Drive. Vdc_perc_sixstep	$0 \leq \text{Number} \leq 100$	DC voltage usage percentage in % of Vdc; corresponds to the Vdc usage percentage field of Drive Settings .
Drive.Transistor	String	Corresponds to the Transistor field of Drive Settings .
Drive.TransistorType	'MOSFET'	Corresponds to the Transistor type field of Drive Settings .
	'IGBT'	
Drive.nTransistors	Integer number ≥ 1	Corresponds to the Transistors in parallel field of Drive Settings .
Drive.Rg	Number > 0	Corresponds to the MOSFET driver resistor field of Drive Settings .
Drive.deadtime	Number ≥ 0	Corresponds to the Switching dead time field of Drive Settings .

9.4. Simple example of simulation script function.

This example shows how to write a simple simulation script function which changes simulation time step and stores some user defined variables in the `Userdata` structure. The source code of the function presented below can be found in *[MotorXP-AFM installation directory]\SimScripts\simscript_example.m*.

```
function [ShaftPosition, CircuitControl, Settings, Enforce, Userdata,
updateCoefs3D] = ...

simscript_example(Outputs,Settings,Userdata,Circuit,Drive,Geometry,Mesh,Windings,A,
cell_p,cell_t,cell_Nu,path)
% Example of simulation script function (should be used with circuit function
InvertedCircuit.m)
if ~Settings.DF_DQinitcnd
    error('Checkbox ''Use initial conditions from dynamic D-Q simulation'' should
be checked.');
```

```

updateCoefs3D=0;
% call default simulation script function for space vector PWM
[ShaftPosition, CircuitControl, Settings, Enforce, Userdata] = ...

simscript_spacevecpwm(Outputs,Settings,Userdata,Circuit,Drive,Geometry,Mesh,Winding
s,A,cell_p,cell_t,cell_Nu,path);

% Current simulation time
if isempty(Outputs.time)
    CurrentTime = 0;
else
    CurrentTime = Outputs.time(end);
end
% Change time step if simulation time reaches 0.1 sec
if CurrentTime>0.01
    Settings.DF_timestep=10^-6;
else
    Settings.DF_timestep=10^-5;
end

```

At the beginning of the code the default simulation script function for the space vector PWM `simscript_spacevecpwm` is called so there is no need to worry about assigning `ShaftPosition`, `CircuitControl` and `Enforce` structures since they are assigned by `simscript_spacevecpwm`. Note that checkbox “Use initial conditions from dynamic D-Q simulation” of the main window should be checked, otherwise the error will occur.

In this example the simulation time step is changed when the simulation reaches 0.01 sec. You can define your own condition to change any simulation parameter in the `Settings` structure.

To store user defined variables (switching functions for this example) the following piece of code is used:

```

% Save switching function for each phase in Userdata structure
if ~isfield(Userdata,'Switch_a')
    % Create fields 'Switch_a', 'Switch_b', 'Switch_c' in Userdata when
    % simscript_example is called for the first time
    Userdata.Switch_a=[];
    Userdata.Switch_b=[];
    Userdata.Switch_c=[];
else
    % CircuitControl.switch_x1 - upper switch state
    if CircuitControl.switch_a1==1
        sA=1;
    else
        sA=-1;
    end
    if CircuitControl.switch_b1==1
        sB=1;
    else
        sB=-1;
    end
    if CircuitControl.switch_c1==1
        sC=1;
    else
        sC=-1;
    end
    % Collect switching function values for each time step
    Userdata.Switch_a=[Userdata.Switch_a sA];
    Userdata.Switch_b=[Userdata.Switch_b sB];

```



```
Userdata.Switch_c=[Userdata.Switch_c sC];  
end
```

When the function is called for the first time, fields `Switch_a`, `Switch_b` and `Switch_c` are created in the `Userdata` structure. The switching function values are stored in the corresponding fields of the `Userdata` structure on each subsequent call of the function. Variables saved in the `Userdata` structure can be plotted as a function of time using the **Time plot** panel of **Plot Wizard** as discussed in section 8.2.3. Be aware that in order to plot the variable, the length of the variable should be the same as number of time steps, i.e. the variable should be an array and its elements should be saved on each time step as shown in the example above.

Fields of the `CircuitControl` structure corresponding to states of electronic switches used in default simulation scripts are discussed in section 10.3.

10. USING ELECTRICAL CIRCUITS

MotorXP-AFM allows you to connect the stator winding to the electrical circuit consisting of the following components:

- resistors
- capacitors
- inductances
- ideal diodes
- electronic switches
- voltage sources
- current sources.

Electrical circuits are generated programmatically using a library of special functions. User defined electrical circuits can be used only for **Dynamic FE Analysis** and available only in MotorXP-AFM **MATLAB** version. Refer to section 3.2 for more details on limitations of MotorXP-AFM Standalone version.

The electrical circuit is specified through the function which retrieves the electrical circuit object. The name of the function appears in the **Stator electrical circuit file** pop-up menu of the main window when **Dynamic FE Analysis** is chosen. Clicking the button to the right allows you to specify your own electrical circuit choosing corresponding M-file from the list of files. Function used for the default electrical circuits of the three phase inverter (shown in Figure 10.1) is implemented in *[MotorXP-AFM installation directory]\Circuits\InverterCircuit.m*. You can use this file as an example to write your own electrical circuit functions.

To understand this chapter some familiarity with MATLAB programming is required. Refer to MATLAB help documentation for more details on MATLAB programming.

10.1. Writing an electrical circuit function.

The definition of the electrical circuit function `InverterCircuit` appears in file *InverterCircuit.m* as follows:

```
function Schematic=InverterCircuit(p,t,Subdomains,Circuit,l,nper,~,~,~)
```

The function retrieves one output argument `Schematic` which contains the electrical circuit description.

The input arguments of the function:

`p` – array of finite element mesh points;

`t` – array of finite element mesh triangles;

`Subdomains` – array of subdomains;

`Circuit` – structure containing stator electrical circuit parameters.

l – length of simulation domain defined as a difference between the outer radius of the last slice and the inner radius of the first slice (see section 2.4).

n_{per} – number of periodicities to define periodic/antiperiodic boundary conditions.

The following fields of the `Circuit` structure can be used:

`Circuit.Npp` – number of parallel paths of the stator winding per phase for one layout; this variable is taken from the **Number of parallel paths** field of **Winding Editor** (see section 4.3 for more details).

`Circuit.Rs` – active resistance of the stator winding per phase for one layout for the specified stator winding temperature; this variable is taken from the **Phase resistance** field of **Winding Editor** (see section 4.3 for more details).

`Circuit.Lsew` – leakage inductance of the stator winding end-turns per phase for one layout; this variable is taken from the **End winding inductance** field of **Winding Editor** (see section 4.3 for more details).

`Circuit.cell_layout` – array of stator winding layouts (see section 4.3 for more details).

`Circuit.statorconnection` – connection between stator winding layouts; empty if there is only one winding layout; is equal to `'1||2'`, if two layouts are connected in parallel; is equal to `'1-2'`, if two layouts are connected in series (see section 4.3 for more details).

`Circuit.statorcircuit` – connection between phases; is equal to either `'StarConnection'` or `'DeltaConnection'`.

Circuit construction procedure always begins with the following function:

```
Schematic = CircuitCreateSchematic();
```

This function does not have input arguments and retrieves the empty circuit object `Schematic`.

The procedure of circuit construction consists of building circuit branches and adding branches to the circuit object.

10.1.1. Electrical circuit branches.

The following function creates an electrical circuit branch:

```
Branch = CircuitCreateBranch(node1,node2);
```

All circuit nodes should be previously numbered beginning with zero node. The function receives start and end node numbers of the circuit branch, respectively, and retrieves an empty branch object `Branch`. When all circuit components are added to the branch object (see section 10.1.2), the branch should be added to the circuit object using the following function:

```
Schematic = CircuitAddBranch(Schematic,Branch);
```

The function receives the circuit and branch objects and retrieves the circuit objects with the new branch added to the circuit.

10.1.2. Adding circuit components.

10.1.2.1. To add a resistor to the branch, the following function is used:

```
Branch = CircuitAddR(Branch,name,value);
```

Branch – variable corresponding to the branch which the resistor is added to, name – unique circuit component name, value – resistance value in Ohm.

Example of adding resistor R1 of 1 kOhm to the branch defined by variable Branch:

```
Branch = CircuitAddR(Branch, 'R1', 1000);
```

10.1.2.2. To add an inductance to the branch, the following function is used:

```
Branch = CircuitAddL(Branch,name,value);
```

Branch – variable corresponding to the branch which the inductance is added to, name – unique circuit component name, value – inductance value in Henries.

Example of adding inductance L1 of 100 mH to the branch defined by variable Branch:

```
Branch = CircuitAddL(Branch, 'L1', 0.1);
```

10.1.2.3. To add a capacitor to the branch, one of the following sentences can be used:

```
Branch = CircuitAddC(Branch,name,value);
```

```
Branch = CircuitAddC(Branch,name,value,Vinit)
```

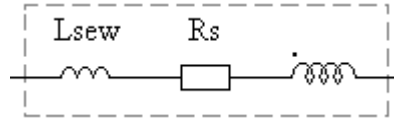
Branch – variable corresponding to the branch which the capacitor is added to, name – unique circuit component name, value – capacitance value in Farads, Vinit – initial voltage on the capacitor. In the first case the initial voltage is zero.

Example of adding capacitor C1 of 100 μ F to the branch defined by variable Branch:

```
Branch = CircuitAddC(Branch, 'C1', 100*10^-6);
```

Since in this case the input variable Vinit is not used the initial voltage is zero.

10.1.2.4. Each stator winding coil or a group of coils can be treated as an independent circuit component consisting of an active resistance, stator end winding inductance and conductors within stator slots designated as a spiral with a dot at the input terminal:



A coil or a group of coils is herein referred to as a coil object or a coil component. Each coil object combines all stator slots or stator slot layers associated with the same phase, the same parallel path, and the same winding layout.

To add a coil object to the branch, the following function is used:

```
Branch = CircuitAddCoilafm(Branch, name, phase, Subdomains, R, Lsew, ...
    npath, nlayout, Npp, coilorientation, p, t, L, nper);
```

Branch – variable corresponding to the branch which the coil object is added to, name – unique circuit component name, phase – phase accepts one of the following values: 'a', 'b', 'c'; Subdomains – array of subdomains; R and Lsew – active resistance and end winding inductance of the coil component, respectively, npath – parallel path number (npath = 1 if there are no parallel paths), nlayout – winding layout number (nlayout = 1 if there is only one winding layout), Npp – number of parallel paths of the stator winding per phase for one layout; coilorientation = 1 if the coil component is oriented concordantly with the branch which the coil component is added to, otherwise coilorientation = -1. The branch orientation is defined by node1 and node2 arguments of CircuitAddBranch function and the coil component orientation is defined by a dot at the input terminal. Input arguments p, t, L, nper are the same as p, t, l, nper in section 10.1.

Note that in order to prevent incorrect results **each branch should include only one coil component**, otherwise coil components must be divided by additional nodes. For example, if you have two coils connected in series you should put additional node between coils, so each coil has its own branch object.

Example of adding the coil component to the branch defined by variable Branch:

```
Branch = CircuitAddCoilafm(Branch, 'coil_c2', 'c', Subdomains, ...
    Rs, Lsew, 2, 1, Npp, 1, p, t, L, nper);
```

In this case the coil component corresponds to the second group (parallel path 2) of coils of phase “c” named as coil_c2, oriented concordantly with the branch, with the active resistance Rs and end winding inductance Lsew.

10.1.2.5. To add a diode to the branch, the following function is used:

```
Branch = CircuitAddDiode(Branch, name, Roff, Ron, direction);
```

Branch – variable corresponding to the branch which the diode is added to, name – unique circuit component name, Roff and Ron – backward and forward resistance of the diode in Ohm;

`direction = 1` if the diode forward direction is oriented from `node1` to `node2` of the branch (see description of `CircuitCreateBranch` in section 10.1.1), otherwise `direction = -1`.

Note that backward and forward resistances should be as follows:

$$0 < R_{off} < \infty$$

$$0 < R_{on} < \infty$$

Example of adding diode D1 to the branch defined by variable `Branch`:

```
Branch = CircuitAddDiode(Branch, 'D1', 10^8, 10^-6, 1);
```

10.1.2.6. To add a voltage source to the branch, the following function is used:

```
Branch = CircuitAddE(Branch, name, data);
```

`Branch` – variable corresponding to the branch which the voltage source is added to, `name` – unique circuit component name, `data` – field of the `CircuitControl` structure used by a simulation script function to control the voltage source (see section 10.2).

10.1.2.7. To add a current source to the branch, the following function is used:

```
Branch = CircuitAddJ(Branch, name, data);
```

`Branch` – variable corresponding to the branch which the current source is added to, `name` – unique circuit component name, `data` – field of the `CircuitControl` structure used by a simulation script function to control the current source (see section 10.2).

10.1.2.8. To add a switch to the branch, the following function is used:

```
Branch = CircuitAddSwitch(Branch, name, data, Roff, Ron);
```

`Branch` – variable corresponding to the branch which the switch is added to, `name` – unique circuit component name, `data` – field of the `CircuitControl` structure used by a simulation script function to control the switch state (see section 10.2), `Roff` and `Ron` – ‘off’ and ‘on’ resistance of the switch in Ohm.

Note that ‘off’ and ‘on’ resistances should be as follows:

$$0 < R_{off} < \infty$$

$$0 < R_{on} < \infty$$

10.2. Controlling power sources and electronic switches using simulation script functions.

While the dynamic FE simulation is running, the voltage and current values supplied by the voltage and current sources as well as the state of each electronic switch can be specified at each time step by the simulation script function. For this purpose, the `CircuitControl` structure, output argument of the simulation script function, is used. Each voltage or current source and switch is associated with its field of the `CircuitControl` structure through the input argument `data`, when the `CircuitAddE`, `CircuitAddJ` or `CircuitAddSwitch` function is called. For the power sources, the value assigned to the corresponding field of the `CircuitControl` structure at a specific time step is the voltage or current output value of the voltage or current source associated with this field. Similarly, the state of the switch is determined by the value assigned to the corresponding field of the `CircuitControl` structure at a specific time step: 1 for state 'on', 0 – for state 'off' (see section 10.3 for more details on using electronic switches).

The following example provides an explanation for the power source control principle. Assume that the voltage source named V1 was added to the stator circuit calling the function

```
Branch = CircuitAddE(Branch, 'V1', 'CircuitControl.v1');
```

In this case the voltage source V1 is associated with the field `v1` of the `CircuitControl` structure. The following piece of code can be used in the simulation script function to specify the voltage value supplied by the voltage source V1 at a given simulation time step:

```
CurrentTime = Outputs.CurrentTime;
CircuitControl.v1 = 220*sqrt(2)*sin(2*pi*50*CurrentTime);
```

In this case we have the source of sinusoidal voltage with frequency 50Hz and RMS value 220V. Note that power sources can have any desired voltage or current waveform.

Similarly, for the switch S1 created by calling the following function:

```
Branch = CircuitAddSwitch(Branch, 'S1', 'CircuitControl.s1', Roff, Ron);
```

Writing simulation script function, you can use the following command to set the switch S1 to 'on' state:

```
CircuitControl.s1 = 1;
```

The following command will set the switch S1 to 'off' state:

```
CircuitControl.s1 = 0;
```

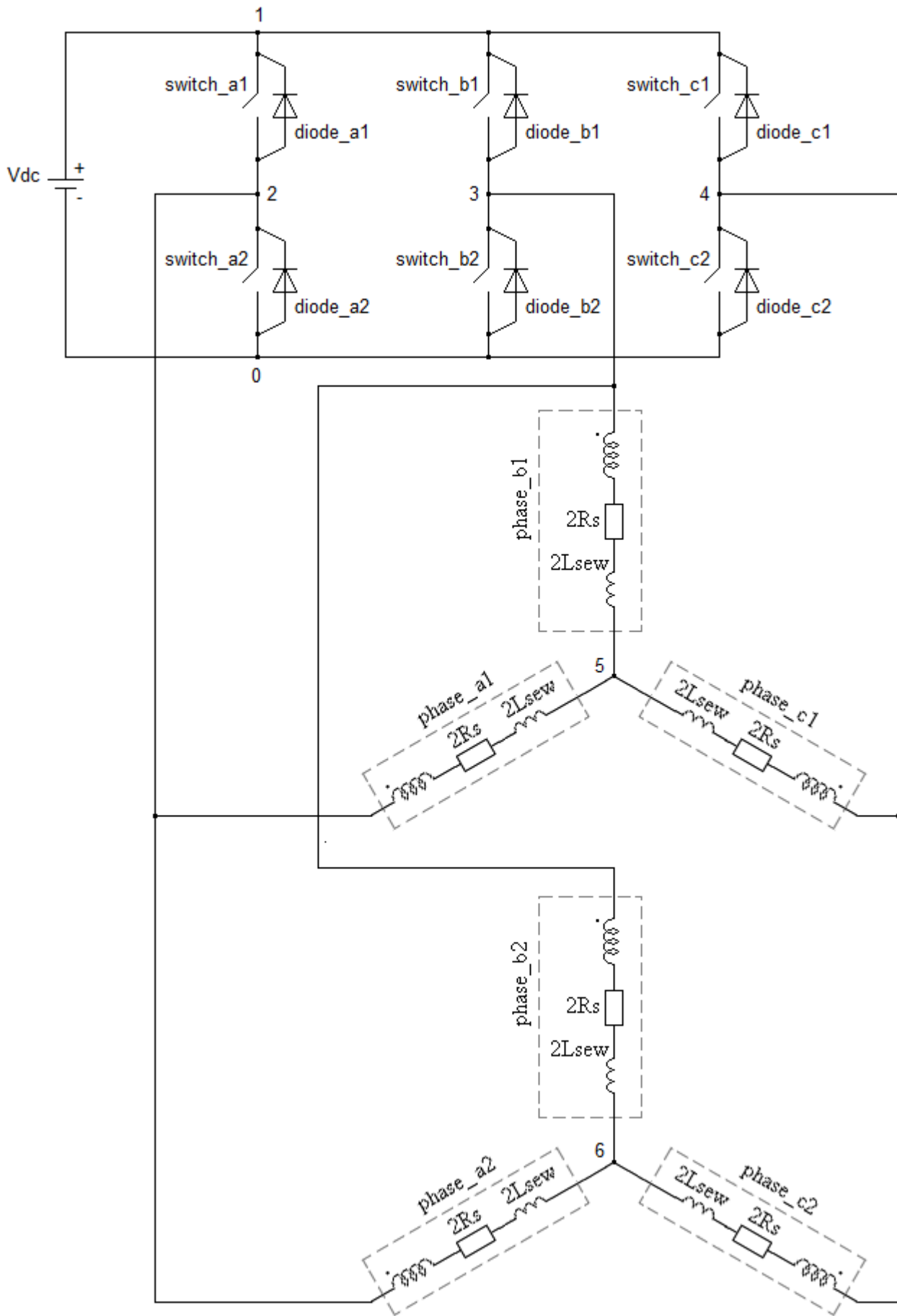


Figure 10.1. Three-phase inverter electrical circuit with star-connected winding (one layout) and two parallel paths.

10.3. Default three-phase inverter circuit function.

As mentioned before the three-phase inverter circuit function `InverterCircuit` is used by default for **Dynamic FE Analysis**. The source code of the function can be found in *[MotorXP-AFM installation directory]\Circuits\InverterCircuit.m*. Example of the electrical circuit with the star-connected stator winding, one winding layout and two parallel paths is shown in Figure 10.1. More examples of stator winding configurations for star and delta connections and for different numbers of winding layouts which can be defined using the circuit function `InverterCircuit` are shown in Figure 4.14. The electrical circuit is modified automatically depending on the winding connection, number of winding layouts and number of parallel paths. The following piece of code constructs the inverter circuit:

```
Schematic = CircuitCreateSchematic();
Branch = CircuitCreateBranch(0,1);
Branch = CircuitAddE(Branch, 'Vdc', 'CircuitControl.vdc');
Schematic = CircuitAddBranch(Schematic, Branch);

Branch = CircuitCreateBranch(2,1);
Branch = CircuitAddSwitch(Branch, 'switch_a1', 'CircuitControl.switch_a1', Roff, Ron);
Schematic = CircuitAddBranch(Schematic, Branch);
Branch = CircuitCreateBranch(2,1);
Branch = CircuitAddDiode(Branch, 'diode_a1', Roff, Ron, 1);
Schematic = CircuitAddBranch(Schematic, Branch);

Branch = CircuitCreateBranch(3,1);
Branch = CircuitAddSwitch(Branch, 'switch_b1', 'CircuitControl.switch_b1', Roff, Ron);
Schematic = CircuitAddBranch(Schematic, Branch);
Branch = CircuitCreateBranch(3,1);
Branch = CircuitAddDiode(Branch, 'diode_b1', Roff, Ron, 1);
Schematic = CircuitAddBranch(Schematic, Branch);

Branch = CircuitCreateBranch(4,1);
Branch = CircuitAddSwitch(Branch, 'switch_c1', 'CircuitControl.switch_c1', Roff, Ron);
Schematic = CircuitAddBranch(Schematic, Branch);
Branch = CircuitCreateBranch(4,1);
Branch = CircuitAddDiode(Branch, 'diode_c1', Roff, Ron, 1);
Schematic = CircuitAddBranch(Schematic, Branch);

Branch = CircuitCreateBranch(0,2);
Branch = CircuitAddSwitch(Branch, 'switch_a2', 'CircuitControl.switch_a2', Roff, Ron);
Schematic = CircuitAddBranch(Schematic, Branch);
Branch = CircuitCreateBranch(0,2);
Branch = CircuitAddDiode(Branch, 'diode_a2', Roff, Ron, 1);
Schematic = CircuitAddBranch(Schematic, Branch);

Branch = CircuitCreateBranch(0,3);
Branch = CircuitAddSwitch(Branch, 'switch_b2', 'CircuitControl.switch_b2', Roff, Ron);
Schematic = CircuitAddBranch(Schematic, Branch);
Branch = CircuitCreateBranch(0,3);
Branch = CircuitAddDiode(Branch, 'diode_b2', Roff, Ron, 1);
Schematic = CircuitAddBranch(Schematic, Branch);

Branch = CircuitCreateBranch(0,4);
Branch = CircuitAddSwitch(Branch, 'switch_c2', 'CircuitControl.switch_c2', Roff, Ron);
Schematic = CircuitAddBranch(Schematic, Branch);
Branch = CircuitCreateBranch(0,4);
Branch = CircuitAddDiode(Branch, 'diode_c2', Roff, Ron, 1);
Schematic = CircuitAddBranch(Schematic, Branch);
```

Part of the code connecting stator winding to the inverter is not shown. All possible options for the stator winding configuration can be found in section 4.3.

As described in the previous section the state of each switch is controlled by the corresponding field of the `CircuitControl` structure. There are three default simulation script functions used together with the `InverterCircuit` function: `simscript_hystpwm`, `simscript_spacevecpwm` and `simscript_sixstep` located in *[MotorXP-AFM installation directory]\SimScripts*. The following piece of code is used to control switch states in the `simscript_spacevecpwm` function:

```
[sA,sB,sC] = spacevecpwm(Vai_ref,Vbi_ref,Vci_ref,fspwm,CurrentTime,Vdc);
if sA==1
    CircuitControl.switch_a1=1;
    CircuitControl.switch_a2=0;
else % sA== -1
    CircuitControl.switch_a1=0;
    CircuitControl.switch_a2=1;
end
if sB==1
    CircuitControl.switch_b1=1;
    CircuitControl.switch_b2=0;
else % sB== -1
    CircuitControl.switch_b1=0;
    CircuitControl.switch_b2=1;
end
if sC==1
    CircuitControl.switch_c1=1;
    CircuitControl.switch_c2=0;
else % sC== -1
    CircuitControl.switch_c1=0;
    CircuitControl.switch_c2=1;
end
```

Use the source code of *InverterCircuit.m*, *simscript_hystpwm.m*, *simscript_spacevecpwm.m* and *simscript_sixstep.m* as an example to write your own electrical circuit functions and simulation scripts.

11. MOTORXP-AFM SETTINGS

MotorXP-AFM settings are available from menu **File -> Settings** and shown in Figure 12.1.

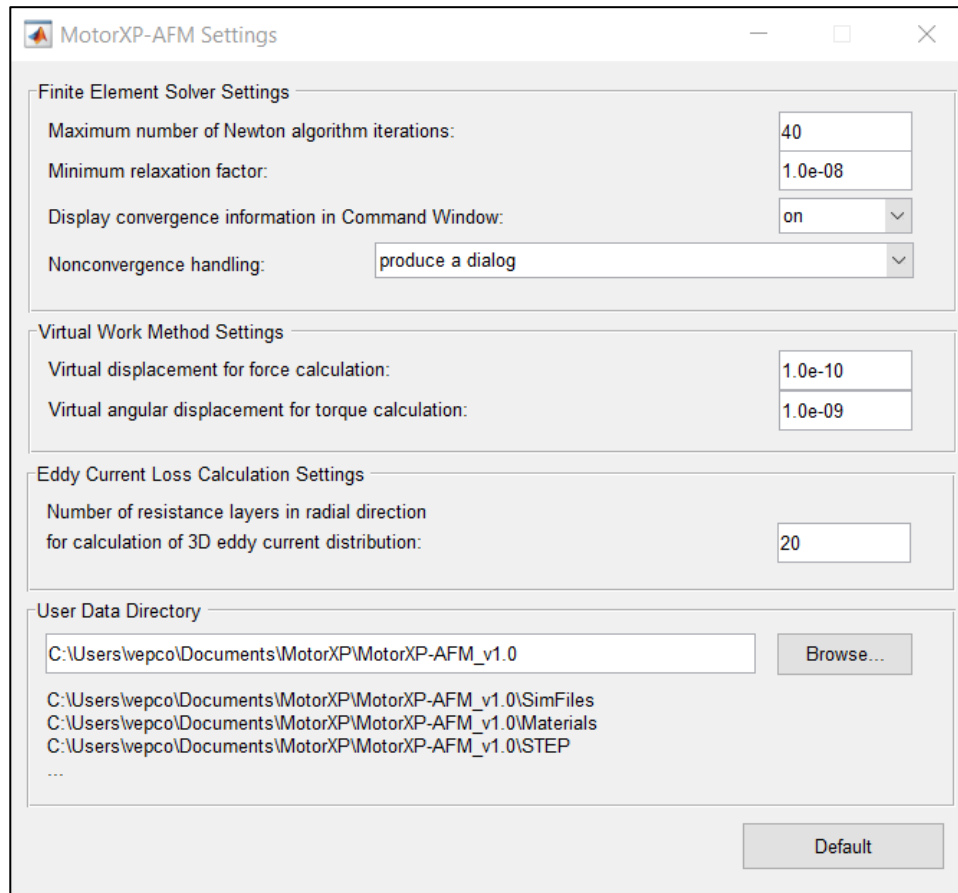


Figure 12.1. MotorXP-AFM settings.

Finite Element Solver Settings.

Maximum number of Newton algorithm iterations – if the number of Newton algorithm iterations exceeds the value specified by this field it is said that the solution does not converge. The subsequent action will depend on the **Nonconvergence handling** field value.

Minimum relaxation factor – if the solution does not converge on the current Newton algorithm iteration, the nonlinear solver damps down the search step with relaxation factor $0 < \alpha < 1$. The relaxation factor iteratively decreases until the convergence occurs. The smaller relaxation factor, the worse convergence. If the relaxation factor becomes less than **Minimum relaxation factor**, it is said that the solution does not converge. The subsequent action will depend on the **Nonconvergence handling** field value.

Display convergence information in Command Window – determines either the convergence information (i.e. Newton algorithm iteration number, the maximum and average residual and the relaxation factor) is displayed in the Command Window.

Nonconvergence handling – determines an action the program takes when the solution does not converge. If *ignore nonconvergence and continue simulation* is chosen, the simulation is continued with the accuracy reached regardless the value specified in the **Convergence tolerance** field of the main window; if *produce a dialog* is chosen, the dialog with the convergence information is provided to allow the user to decide either to continue or interrupt the simulation; if *interrupt simulation and report an error* is chosen, the simulation is interrupted with an error report.

Virtual Work Method Settings.

Fields of this panel define the virtual displacement for torque and force calculation with the virtual work method as discussed in section 2.3.

Eddy Current Loss Calculation Settings.

This panel influences the accuracy of the magnet loss calculation and eddy current loss calculation. By default, number of the resistance layers in radial direction is 20.

User Data Directory.

Specifies the directory where all the user files are stored such as projects and simulation data files (*SimFiles* folder), material library files (*Materials* folder) and winding layout files (*layoutfiles* folder).

The **Default** button resets all the MotorXP-AFM settings shown in Figure 12.1 to its default values except **User Data Directory**.

APPENDIX

Appendix A. Power balance, discretization error and accuracy of the results.

If the power balance is satisfied it means that the input apparent power delivered by all current and voltage sources equals to the consumed apparent power:

$$P_{input} = P_{cons}$$

The input power is calculated from voltages and currents of all power sources:

$$P_{input} = \sum_{n=1}^m i_n u_n$$

The consumed power is defined as follows:

$$P_{cons} = P_s + P_{mech} + \frac{\Delta W_{mf}}{\Delta t},$$

where P_s – apparent power (real and reactive) of the stator electrical circuit, P_{mech} – mechanical power on the shaft, $\Delta W_{mf}/\Delta t$ – magnetic field energy time derivative.

The discretization error of the simulation is a percentage difference between the input power and consumed power. The discretization error is mostly influenced by the simulation time step and can be used to assess the accuracy of the simulation results.

Appendix B. Magnetostatic Analysis results.

In the Magnetostatic Analysis the instantaneous values of the input power are calculated as the following:

$$p_{input} = i_a v_a + i_b v_b + i_c v_c \quad (13.4)$$

And the input power value displayed in the **Results** panel of the Magnetostatic Analysis is the average of the instantaneous input power defined by the Exp. (13.4). Note that the iron loss, magnet loss and other eddy current loss are calculated during the post processing and not included in the input power.

The efficiency is calculated as the following:

$$Efficiency = \frac{P_{mech}}{P_{mech} + P_s + P_{iron} + P_{magnet} + P_{other}} \quad (13.5)$$

where P_{mech} – mechanical power on the shaft, P_s – stator winding loss, P_{iron} – total iron core loss, P_{magnet} – magnet eddy current loss, P_{other} – other eddy current loss.

The power factor is calculated as the following:

$$PowerFactor = \frac{P_{mech} + P_s}{3 \cdot I_{RMS} \cdot V_{RMS}} \quad (13.6)$$

where I_{RMS} and V_{RMS} – RMS values of the phase current and phase voltage.

The reason why the input power is not used for the efficiency and power factor calculation is that the Exp. (13.5) and (13.6) are more stable to the discretization error as the simulation time step increases.

The discretization error shows how the input power matches with the mechanical power and stator winding loss and defined as follows:

$$Error = 100 \cdot abs \left[\frac{P_{input} - (P_{mech} + P_s)}{P_{input}} \right] \quad (13.7)$$

The discretization error decreases as the simulation time step decreases and the mismatch between the input power, mechanical power, losses, efficiency and power factor also decrease.

Appendix C. Out of memory errors handling.

“Out of memory” errors can occur when the size of the mesh is too large. According to the MATLAB help documentation, to avoid “out of memory” errors when using MotorXP-AFM, you can increase the size of the swap file or add more memory to the system. Restarting MATLAB when “out of memory” error occurs may also help in some cases.

PART II

MotorXP-AFM MATLAB scripting API

Overview

MotorXP-AFM MATLAB scripting API allows changing the geometry, winding and mesh parameters of the axial flux machine (AFM) design as well as simulation settings and running magnetostatic finite element (FE) simulations directly from the MATLAB scripts and functions without using the user interfaces. MotorXP-AFM MATLAB scripting API enables parallel processing, i.e. running several simulations at the same time fully utilizing the capabilities of the modern multi-core and multi-CPU computers.

MotorXP-AFM MATLAB scripting API does not require any additional MATLAB toolboxes.

Note that MotorXP-AFM scripting API requires MATLAB Runtime R2020b (9.9). Download and install it from MathWorks website <https://www.mathworks.com/products/compiler/matlab-runtime.html>.

1. Scripting API initialization

[initMXAscriptingAPI](#) – function initializing the MotorXP-AFM MATLAB scripting API environment, i.e., adding all the necessary paths and loading necessary libraries.

2. Functions for working with mxa-files

[openMXA](#) – function that opens an AFM design from a mxa-file creating the *motorProps* and *settingsMagnetostatic* structures.

[saveDesign2MXAfile](#) – function that saves an AFM design defined by the *motorProps* and *settingsMagnetostatic* structures to a mxa-file.

3. Functions for setting and getting parameters

[setParamafm](#) – function used to write geometry, winding and mesh data parameters to the *motorProps* structure as well as simulation settings to the *settingsMagnetostatic* structure.

[getParamafm](#) – function used to retrieve geometry, winding and mesh data parameters from the *motorProps* structure as well as simulation settings from the *settingsMagnetostatic* structure.

4. Serial processing functions

4.1. Assembling AFM designs in series

[assembleMXA](#) – function preparing all the necessary structures for a specific AFM design for running simulations.

4.2. Running magnetostatic FE simulations in series

[runMStimesteppingafm](#) – run magnetostatic FE simulation for a specific AFM design.

5. Parallel processing functions

5.1. Assembling AFM designs in parallel

[initAssembleMXA_par](#) – initializes assembling multiple AFM designs in parallel.

[runAssembleMXA_par](#) – starts assembling multiple AFM designs in parallel.

[getAssembleMXA_par](#) – retrieves structures for assembled AFM designs.

5.2. Running magnetostatic FE simulations in parallel

[initMStimesteppingafm_par](#) – initializes running multiple magnetostatic FE simulations in parallel.

[runMStimesteppingafm_par](#) – starts running multiple magnetostatic FE simulations in parallel.

[getMStimesteppingafm_par](#) – retrieves results for finished magnetostatic FE simulations.

6. Functions for automatic optimization workflow development

[runTSEMOptimizationafm](#) – function that executes the optimization algorithm.

[evalDesigns_emrax228](#) – example of the design function for evaluating generated AFM designs.

[plotPareto](#) – displays the optimization algorithm results as the Pareto plot.

[plotParetoFromFile](#) – displays the Pareto plot for the optimization algorithm results previously saved to a mat-file.

1. Scripting API initialization

1.1. *initMXAscriptingAPI*

The *initMXAscriptingAPI* function initializes the necessary path folders that will be used during the analysis and load necessary libraries. It is recommended to start the script calling this function.

1.2. Syntax

```
initMXAscriptingAPI
```

This function does not have inputs and outputs.

2. Functions for working with mxa-files

2.1. *openMXA* function

2.1.1. Syntax

```
[motorProps, settingsMagnetostatic] = openMXA(file)
```

2.1.2. Description

The *openMXA* function is used to create two fundamental structures: [*motorProps*](#) and [*settingsMagnetostatic*](#). To create these structures, initial AFM design data stored in a mxa-file are needed. Once *openMXA* is called, the script can access the model data and create the necessary structures for the simulation.

2.1.3. Input and outputs

To input the initial mxa-file for the analysis, the user should provide its path. When an AFM design is saved using the MotorXP-AFM user interface, a mxa-file is automatically created. To simplify the process of providing a path, the *getafmpath* function can be used to return the path of the mxa-file. This allows for relative paths to be used, as shown in the example:

```
file = getafmpath('SimFiles/EMRAX-228/EMRAX-228.mxa');
```

The outputs of *openMXA* are two structures: *motorProps* and *settingsMagnetostatic*, as described below.

2.1.3.1. *motorProps* structure

This is a structure that contains the parameters of the motor model, such as the geometry of the stator and rotor, winding, air gap, and mesh. All the parameters defining the AFM design are stored within this structure. The detailed description for all the fields of *motorProps* is provided in [section 3.2](#).

2.1.3.2. *settingsMagnetostatic* structure

This is a structure that contains the magnetostatic FE simulation settings such as solver type, convergence tolerance, mechanical speed, stator current, etc. The detailed description for all the fields of *settingsMagnetostatic* is provided in [section 3.3](#).

2.2. *saveDesign2MXAfile* function

2.2.1. Syntax

```
saveDesign2MXAfile(initdesignfile, motorProps, MXAfile)
```

2.2.2. Description

The *saveDesign2MXAfile* function is used to save the specific AFM design to a mxa-file so it can then be opened using MotorXP-AFM user interface for detailed analysis.

2.2.3. Input and outputs

initdesignfile – initial mxa-file storing the basic parameters of the AFM design.

motorProps – [motorProps](#) structure containing desired parameters of the AFM design which will overwrite the basic parameters from *initdesignfile*.

MXAfile – path and mxa-file name for the created new mxa-file.

3. Functions for setting and getting parameters

3.1. *setParamafm* function

3.1.1. Syntax

```
paramStruct = setParamafm(paramStruct,paramName,paramValue)
```

3.1.2. Description

The *setParamafm* function is used to modify values in structures [motorProps](#) and [settingsMagnetostatic](#).

3.1.3. Inputs

The input *paramStruct* is either [motorProps](#) or [settingsMagnetostatic](#) structure. If you try to use another structure, an error message will be displayed: “*Incorrect input structure, only motorProps and settingsMagnetostatics are valid*”.

The possible values for inputs *paramName* and *paramValue* are defined by json-files as described in sections [3.1.5](#) and [3.1.6](#) and in Tables 1-7.

3.1.4. Output

The output structure *paramStruct* is the same as the input structure *paramStruct* with the modified field value defined by *paramName*.

3.1.5. Dependencies on json-files

The *setParamafm* function depends on some json-files which define the possible *paramName* values for the [motorProps](#) and [settingsMagnetostatic](#) structures. There are several default json-files as listed below.

In *\bin\assets\scripts*:

- *motorProps.json* – general [motorProps](#) parameters.
- *settingsMagnetostatic.json* – general [settingsMagnetostatic](#) parameters.

In *\bin\assets\scripts\Rotor*:

- *Rectangular_magnet.json* – geometry parameters for the **Rectangular magnet** rotor geometry template.
- *Trapezoidal_magnet.json* – geometry parameters for the **Trapezoidal magnet** rotor geometry template.
- *Trapezoidal_magnet_halbach.json* – geometry parameters for the **Halbach array rotor** geometry template.

In *\bin\assets\scripts\Stator*:

- *Parallel_slot_flat.json* – geometry parameters for the **Parallel flat slot** stator geometry template.
- *Parallel_slot_round.json* – geometry parameters for the **Parallel round slot** stator geometry template.

3.1.6. *Stator and rotor geometry template json-files*

As can be noticed from the json-file names listed above, each stator and rotor geometry template js-file have the corresponding json-file with the same name. This is how the *setParamafm* and *getParamafm* functions recognize which geometry parameter defined by *paramName* corresponds to each field name in *motorProps.stator.scriptProps* (for stator) or in *motorProps.rotor.scriptProps* (for rotor). For example, in file *Parallel_slot_flat.json* there are the follow lines:

```
{
  .....
  "Stator tooth outer diameter": "stator.scriptProps.Doht",
  "Stator tooth inner diameter": "stator.scriptProps.Dith"
}
```

For example, to set the stator tooth outer diameter to 10, use the following

```
motorProps = setParamafm(motorProps, 'Stator tooth outer
                           diameter', 10);
```

3.2. *motorProps structure*

3.2.1. *Main paramName parameter description and paramValue values*

The possible *paramName* and their respective description and values are shown in Table 1.

Table 1. The *paramName* and *paramValue* variable values used for the *motorProps* structure.

Name (<i>paramName</i>)	Description	Values (<i>paramValue</i>)
Stator inner diameter	Corresponds to the Stator inner diameter (mm) field of Geometry Editor (Stator).	Number > 0
Stator outer diameter	Corresponds to the Stator outer diameter (mm) field of Geometry Editor (Stator).	Number > 0
Number of slots	Corresponds to the Number of slots field of Geometry Editor (Stator).	Number >= 6
Stator type	Corresponds to the Stator type field of Geometry Editor (Stator).	'Yoke' 'Yokeless'
Top stator angular displacement	Corresponds to the Top stator angular displacement (degrees) field of Geometry Editor (Stator).	Number >= 0
Bottom stator angular displacement	Corresponds to the Bottom stator angular displacement (degrees) field of Geometry Editor (Stator).	Number >= 0
Stator height	Corresponds to the Stator height (mm) field of Geometry Editor (Stator).	Number > 0
Rotor inner diameter	Corresponds to the Rotor inner diameter (mm) field of Geometry Editor (Rotor).	Number > 0

Number of pole pairs	Corresponds to the Number of pole pairs field of Geometry Editor (Rotor).	Number ≥ 1
Rotor outer diameter	Corresponds to the Rotor outer diameter (mm) field of Geometry Editor (Rotor).	Number > 0
Pole arrangement	Corresponds to the Pole arrangement field of Geometry Editor (Rotor).	'N-N' 'N-S'
Top rotor angular displacement	Corresponds to the Top rotor angular displacement (degrees) of Geometry Editor (Rotor).	Number ≥ 0
Bottom rotor angular displacement	Corresponds to the Bottom rotor angular displacement (degrees) field of Geometry Editor (Rotor).	Number ≥ 0
Rotor height	Corresponds to the Rotor height (mm) field of Geometry Editor (Rotor).	Number > 0
Rotor type	Corresponds to the Rotor type field of Geometry Editor (Rotor).	'Yoke' 'Yokeless'
End winding inductance	Leakage inductance of the stator winding end-turns per phase for one layout; corresponds to the End winding inductance field of Winding Editor . If set to 'Auto', the value will be calculated automatically based on the winding configuration and machine dimensions.	'Auto' or Number ≥ 0
Number of parallel paths	Number of parallel paths of the stator winding per phase for one layout; corresponds to the Number of parallel paths field of Winding Editor .	Number ≥ 1
Phase resistance	Active DC resistance of the stator winding per phase for one layout; corresponds to the Phase resistance field of Winding Editor . If set to 'Auto', the value will be calculated automatically based on the winding configuration, machine dimensions and winding temperature.	'Auto' or Number ≥ 0
Coil span	Corresponds to the Coil span field of Winding Editor . If set to 'Auto', the value will be calculated automatically.	'Auto' or $1 \leq \text{Number} \leq \text{max_value}$

Coil fill factor	Stator coil fill factor; corresponds to the Coil fill factor field of Winding Editor . This parameter is required only if 'Wire size method' is set to 'Fill factor'.	$0 < \text{Number} \leq 1$
Winding layers orientation	Determined either winding layers are oriented horizontally or vertically inside the slot; corresponds to the Winding layers orientation field of Winding Editor .	'Upper / Lower' 'Left / Right'
Number of strands in hand	Corresponds to the Number of strands in hand field of Winding Editor .	$\text{Number} \geq 1$
Winding layers	Determined either single layer or double layer winding is used; corresponds to the Winding layers field of Winding Editor .	'Single layer' 'Double layer'
Number of turns	Specifies the number of turns per one coil, the same as the number of turns per one winding layer; corresponds to the Number of turns field of Winding Editor .	$\text{Number} \geq 1$
Winding circuit	Determines the stator winding connection; corresponds to the Winding circuit field of Winding editor .	'StarConnection' 'DeltaConnection'
Layouts connection	Specifies the connection between the winding layouts; corresponds to the Layouts connection field of Winding editor .	' ' (empty string) '1-2' '1 2'
Strand diameter	Corresponds the Strand diameter field of Winding editor . Strand diameter, by default, is calculated based on coil fill factor, number of turns and number of strands in hand. This parameter is required only if 'Wire size method' is set to 'Wire diameter'.	$\text{Number} > 0$
Winding type	Corresponds the Winding type field of Winding editor .	'Planar' 'Toroidal'

Wire gauge number	Wire diameter according to the Standard wire gauge (SWG) or American wire gauge (AWG) standard; corresponds the Wire gauge number field of Winding Editor . This parameter is required only if 'Wire size method' is set to 'AWG' or 'SWG'.	AWG Table value, e.g., '21 AWG', or SWG Table value, e.g., '21 SWG',
Wire size method	Wire size method specifies how the Strand diameter value is determined; corresponds the Wire size method field of Winding editor .	'Wire diameter' 'AWG' 'SWG' 'Fill factor'
Air gap	Air gap length; corresponds to the Air gap field of Geometry Editor .	Number > 0
Air gap mesh quality	Air gap mesh quality modifies the quality of the mesh in the air gap region; corresponds the Air gap mesh quality field of Mesh Editor .	'Low' 'Medium' 'High'
Maximum triangle side	The upper bound of length of the longest mesh triangle side; corresponds the Maximum triangle side (mm) field of Mesh Editor . If set to 'Auto', the value will be calculated automatically.	'Auto' or Number > 0.1
Number of layers in air gap	Number of mesh layers in the air gap; corresponds to the Number of layers in air gap field of Mesh Editor .	3, 5, 7 or 9
Number of cylindrical slices	Number of cylindrical slices the machine is divided in and used by the multi-slice FEM; corresponds to the Number of cylindrical slices field of Mesh Editor .	Integer number from 1 to 20
Boundary conditions	Periodic/antiperiodic boundary conditions; corresponds to the Boundary conditions field of Mesh Editor . If set to 'Auto', best possible boundary conditions will be used.	'Auto' 'None' 'Periodic' 'Antiperiodic'

3.2.2.Examples.

Example to change the air gap length passing 'Air gap' as **paramName** and '1.2' as **paramValue**:

```
motorProps = setParamafm(motorProps, 'Air gap', 1.2);
```

Example to change the number of layers in air gap passing 'Number of layers in air gap' as **paramName** and '9' as **paramValue**:

```
motorProps = setParamafm(motorProps, 'Number of layers in air gap', 9);
```

Example to change the number of slots passing 'Number of slots' as **paramName** and 48 as **paramValue**:

```
motorProps = setParamafm(motorProps, 'Number of slots', 48);
```

Example to change phase resistance passing 'Phase resistance' as **paramName** and 0.346 as **paramValue**:

```
motorProps = setParamafm(motorProps, 'Phase resistance', 0.346);
```

3.2.3.Some limitations for **paramValue** imposed by the machine topology.

There are three possible machine topologies, whose are:

- *Stator / Rotor.*
- *Stator / Rotor / Stator.*
- *Rotor / Stator / Rotor.*

A more detailed description of the supported topologies can be found in [section 4.1](#) of Part I of this user manual.

Some rotor and stator configurations are not possible for some topologies with a few examples given below.

For the *Rotor / Stator* topology, trying to change the rotor type passing 'Rotor type' as **paramName** and 'Yokeless' as **paramValue**:

```
motorProps = setParamafm(motorProps, 'Rotor type', 'Yokeless');
```

An error message will be displayed: “*Only Yoke type is possible for this topology*”.

For the *Rotor / Stator* topology, trying:

```
motorProps = setParamafm(motorProps, 'Pole arrangement', 'N-N');
```

An error message will be shown: “*The parameter - Pole arrangement - cannot be assigned in this topology*”.

For the *Rotor / Stator* topology trying:

```
motorProps = setParamafm(motorProps, 'Layouts connection', '1||2');
```

An error message will be shown: “*The parameter - Layouts connection - cannot be assigned in this topology*”.

3.2.4.The **paramName** and **paramValue** variable values for default rotor and stator geometries.

There are two default stator geometries (*Parallel slot flat* and *Parallel slot round*) and three default rotor geometries (*Rectangular magnet*, *Trapezoidal magnet* and *Halbach array rotor*). The possible

paramName and *paramValue* variable values of the *setParamafm* function for the default rotor and stator geometries are listed in Tables 2-6. Refer to [section 4.1](#) of Part I of this user manual for more details.

Table 2. Stator geometry *Parallel slot flat*: *paramName* and *paramValue* variable values.

<i>paramName</i>	Values (<i>paramValue</i>)
Stator tooth outer diameter	Number > 0
Stator tooth inner diameter	Number > 0
Slot width	Number > 0
Tooth fillet radius	Number >= 0
Tooth tip height	Number >= 0
Tooth tip slit length	Number > 0
Stator yoke height	Number > 0
Tooth inner border	'Flat' 'Curved'
Slot insulation thickness	Number >= 0
Between layers insulation thickness	Number >= 0
Wedge thickness	Number >= 0

Table 3. Stator geometry *Parallel slot round*: *paramName* and *paramValue* variable values.

<i>paramName</i>	Values (<i>paramValue</i>)
Slot width	Number > 0
Stator yoke height	Number > 0
Slot opening depth	Number >= 0
Slot opening width	Number >= 0
Tooth tip angle	Number >= 0
Slot bottom corner type	'General ' 'Round '
Slot bottom corner radius	Number >= 0
Slot top corner radius	Number >= 0
Slot insulation thickness	Number >= 0
Between layers insulation thickness	Number >= 0

Table 4. Rotor geometry *Rectangular magnet*: *paramName* and *paramValue* variable values.

<i>paramName</i>	Values (<i>paramValue</i>)
Magnet inner diameter	Number > 0
Magnet length	Number > 0
Magnet width	Number > 0
Rotor yoke height	Number > 0
Magnet inner border	'Curved ' 'Flat '
Magnet outer border	'Curved ' 'Flat '
Magnet inset depth	Number >= 0
Number of magnet segments in radial direction	Number >= 1

Table 5. Rotor geometry *Trapezoidal magnet*: *paramName* and *paramValue* variable values.

<i>paramName</i>	Values (<i>paramValue</i>)
Magnet outer diameter	Number > 0
Magnet inner diameter	Number > 0
Magnet spacing	Number >= 0
Rotor yoke height	Number > 0
Magnet inner border	'Flat' 'Curved'
Magnet outer border	'Flat' 'Curved'
Magnet spacing type	'Parallel' 'Radial'
Magnet inset depth	Number >= 0
Number of magnet segments in radial direction	Number >= 1

Table 6. Rotor geometry *Halbach array rotor*: *paramName* and *paramValue* variable values.

<i>paramName</i>	Values (<i>paramValue</i>)
Magnet outer diameter	Number >0
Magnet inner diameter	Number > 0
Vertical magnet percentage	Number > 0
Rotor yoke height	Number > 0
Magnet inner border	'Curved' 'Flat'
Magnet inset depth	Number >= 0

Number of magnet segments in radial direction	Number ≥ 1
Outermost rotor core type	'Ironless' 'Iron'

3.3. *settingsMagnetostatic* structure

This structure contains settings which are used for running magnetostatic FE simulations. The settings are the same as displayed in the MotorXP-AFM main window for Magnetostatic FE Analysis, see Figure 1.

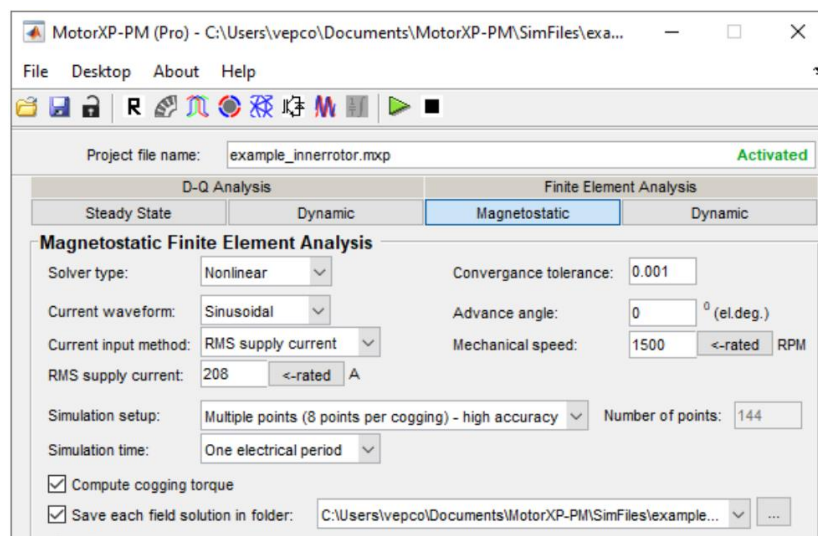


Figure 1. MotorXP-AFM main window with Magnetostatic FEA settings.

3.3.1. *Main paramName parameter description and paramValue values*

The possible *paramName* and their respective description and values for the *settingsMagnetostatic* structure are shown in Table 7. A more detailed description of the magnetostatic FE simulation settings can be found in [section 5.1](#) of Part I of this user manual.

Table 7: The *paramName* and *paramValue* variable values used for the *settingsMagnetostatic* structure.

<i>paramName</i>	Description	Values (<i>paramValue</i>)
Solver type	Corresponds to the Solver type pup-up menu of the main window, see Figure 1.	'Nonlinear' 'Linear'
Convergence tolerance	Corresponds to the Convergence tolerance field of the main window, see Figure 1.	Number > 0
Number of points	Defines the number of time steps (rotor positions) per one electrical period or per selected time interval.	Number ≥ 1

	This can be calculated with another external function <code>setNumberOfPoints('High accuracy', nPolePairs, Ns)</code> . See section 3.3.2 .	
Mechanical speed	The mechanical speed of the rotor in RPM; corresponds to the Mechanical Speed field of the main window, see Figure 1.	Number ≥ 0 Units: RPM
Advance angle	Corresponds to the Advance angle field of the main window, see Figure 1.	Number ≥ 0 Units: electrical degrees
Compute cogging torque	Enables calculation of the cogging torque, corresponds to the Compute cogging torque checkbox value of the main window, see Figure 1. Note that calculation of the cogging torque slows down the simulation.	0 1
Save each step solution	Corresponds to the Save each step solution checkbox value of the main window, see Figure 1.	0 1
Solution folder	Folder to store data-files when 'Save each field solution' is set 1.	Directory path e.g., 'C:\example'
Simulation time	Predefined simulation time intervals. Simulation can be run either for one electrical period starting from zero or for the time interval specified by 'Time from' and 'Time to' (see below).	'One electrical period' 'Define time interval'
Time from	Simulation start time when 'Simulation time' is set to 'Define time interval'.	Number > 0 Units: seconds
Time to	Simulation stop time when 'Simulation time' is set to 'Define time interval'.	Number > 0 Units: seconds
Current waveform	Defines the stator current waveform used for the simulation.	'Sinusoidal' 'Trapezoidal'
RMS supply current Peak supply current RMS current density RMS phase current DC supply current	The Stator current input method which also depends on the current waveform. For the sinusoidal current waveform, the following current input methods are available: <i>RMS supply current</i> , <i>Peak supply current</i> , and <i>RMS current density</i> . For the trapezoidal current waveform, the following current input methods are available: <i>RMS phase current</i> , <i>DC supply current</i> , and <i>RMS current density</i> .	Number ≥ 0 Units: for current: Amperes for current density: Amperes per square millimeter

3.3.2. Examples

Example to change the solver type passing 'Solver type' as *paramName* and 'Linear' as *paramValue*:

```
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'Solver
                                type', 'Linear');
```

Example to change the solver type passing 'Mechanical speed' as *paramName* and '1500' as *paramValue*:

```
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'Mechanical
                                speed', 1500);
```

There are some particularities to change the values of the parameters related to the current input method and current density. Two fields are assigned at the same time, which are shown in the example below, where 'RMS supply current' is a current input method and 200 is the RMS supply current value. The current waveform must be separately assigned.

```
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'Current
                                waveform', 'Sinusoidal');
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'RMS
                                supply current', 200);
```

Another case is shown below when a current input method is wrongly used:

```
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'Current
                                waveform', 'Trapezoidal');
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'RMS
                                supply current', 200);
```

In this case an error message will be displayed “*The parameter – RMS supply current – cannot be assigned with this current waveform.*” since the 'RMS supply current' option is not available for the trapezoidal current waveform.

Example to change the number of points passing 'Number of points' as *paramName* and the result of the *setNumberOfPoints* function as *paramValue*:

```
settingsMagnetostatic = setParamafm(settingsMagnetostatic, 'Number of
                                points', setNumberOfPoints('High accuracy', nPolePairs, Ns));
```

Note that the *setNumberOfPoints* function returns the number of points related to desired accuracy. The first input variable of *setNumberOfPoints* defines the number of points (time-steps) per one period of cogging torque which determines the accuracy of the simulation results. The following options are available:

- 'High accuracy' – 8 time-steps per one period of cogging torque
- 'Medium accuracy' – 4 time-steps per one period of cogging torque
- 'Low accuracy' – 2 time-steps per one period of cogging torque

The number of points is then calculated using the number of pole pairs (nPolePairs) and number of stator slots (Ns) input variables.

If 'Number of points' is set to 1, it will correspond to the *Single point (FEA + D-Q based)* – fastest item of the **Simulation setup** pop-up menu of the main window, see Figure 1. Refer to [section 5.1](#) of Part I of this user manual for more details.

3.4. *getParamafm* function

3.4.1. *Syntax*

```
paramValue = getParamafm(paramStruct,paramName)
```

3.4.2. *Description*

The *getParamafm* function is used to query the values from the structures [motorProps](#) and [settingsMagnetostatic](#).

3.4.3. *Inputs*

The input *paramStruct* is either [motorProps](#) or [settingsMagnetostatic](#) structure. If you try to use another structure instead of [motorProps](#) or [settingsMagnetostatic](#), an error message will be displayed: “*Incorrect input structure, only motorProps and settingsMagnetostatics are valid*”.

The possible values for input *paramName* are the same as for *setParamafm* and listed in Tables 1-7. Additionally, 'all' can be used to display all the values from [motorProps](#) or [settingsMagnetostatic](#), as shown below:

```
getParamafm(motorProps,'all')
getParamafm(settingsMagnetostatic,'all')
```

3.4.4. *Output*

The output *paramValue* is the corresponding value from [motorProps](#) or [settingsMagnetostatic](#) as listed in Tables 1-7.

3.4.5. *Dependencies on json-files*

The *getParamafm* function depends on the same json-files as *setParamafm*. Refer to [section 3](#) for more details.

3.4.6. *Examples*

Example to query the value of 'Air gap mesh quality' from the [motorProps](#) structure:

```
value = getParamafm(motorProps,'Air gap mesh quality')
```

Example to query the value of 'Rotor type' from the [motorProps](#) structure:

```
value = getParamafm(motorProps,'Rotor type')
```

Example to query the value of 'Convergence tolerance' from the [settingsMagnetostatic](#) structure:

```
value = getParamafm(settingsMagnetostatic,'Convergence tolerance')
```

4. Serial processing functions

4.1. Assembling AFM designs in series using the *assembleMXA* function

4.1.1. *Syntax*

```
[Geometry, Windings, Mesh, Materials, RadMesh, SubdomainProperty,
Error, motorProps_output, Weight] =
assembleMXA(initMXAfile,motorProps)
```

4.1.2. Description

The ***assembleMXA*** function is used to prepare AFM design structures necessary for running magnetostatic FE simulations. This process is called AFM design assembling. During assembling the program creates the 3D geometry of the machine, builds finite element mesh, assigns subdomain properties, etc.

4.1.3. Inputs

There are two inputs, ***motorProps*** structure and the file path of the initial mxa-file project.

4.1.4. Outputs

The outputs are listed below with a brief description:

Geometry – structure containing information about the AFM geometry and topology.

Windings – structure with windings information, such as the number of turns, stator connection, number of pole pairs, etc. There are the following field which can be useful:

- ***Winding.Rs*** – active DC resistance of the stator winding per phase for one layout considering the winding temperature specified in ***Materials.Tsw***.
- ***Winding.Lsew*** – the leakage inductance of the stator winding end turns per phase for one layout.

Mesh – structure with mesh data.

Materials – structure with materials data.

RadMesh – structure with the radial mesh data.

SubdomainProperty – structure containing information about subdomain properties.

Error – if the AFM design failed assembling, the error message will be stored in this structure.

motorProps_output – output ***motorProps*** structure. In some cases, there can be some inconsistencies and improperly assigned values in the input ***motorProps*** structure. So, the ***assembleMXA*** function corrects it and returns the corrected values in the ***motorProps_output*** structure. The user can compare the corresponding fields of the ***motorProps*** and ***motorProps_output*** structures to make sure the resulting AFM design looks as intended.

Weight - return the mass in kilograms for each motor active material. This parameter had 5 fields for each active part of the machine:

- ***Weight.statoriron*** – stator iron core mass.
- ***Weight.rotoriron*** – rotor iron core mass.
- ***Weight.statorwinding*** – stator winding mass.
- ***Weight.magnet*** – magnet mass.
- ***Weight.conductor*** – conductive parts.

4.2. Running magnetostatic FE simulations in series using the ***runMStimesteppingafm*** function

4.2.1. Syntax

```
[Results, Timesteppingdata, Coefs3D, field, ~, ~, Private] =
    runMStimesteppingafm(Private, Geometry, Mesh, Windings,
        settingsMagnetostatic, RadMesh, Materials, SubdomainProperty,
        SwitchDutyCycle, Coefs3D);
```

4.2.2. Description

This function runs the magnetostatic finite element simulation.

4.2.3. *Inputs*

Private – structure with internal data of the FE solver. If **Private** is empty, it will be calculated inside the **runMStimesteppingafm** function before the simulation is started. **Private** depends on the AFM design parameters (i.e. input arguments **Geometry**, **Windings**, **Mesh**, **Materials**, **RadMesh**, **SubdomainProperty**), but does not depend on the simulation settings (i.e. input arguments **settingsMagnetostatic** and **SwitchDutyCycle**). It means that **Private** should be recalculated (i.e. should be empty while calling **runMStimesteppingafm**) for each new AFM design. If only simulation settings are changed (current, advance angle, speed, number of rotor positions, convergence tolerance or any other parameter in **settingsMagnetostatic**), there is no need to recalculate **Private** – the one can use **Private** calculated with the previous call of **runMStimesteppingafm** – refer to output argument **Private** below.

Geometry, **Windings**, **Mesh**, **Materials**, **Radmesh**, **SubdomainProperty** – structures which define the AFM design.

settingsMagnetostatic – magnetostatic FE simulation settings. Refer to [section 3.3](#) for more details.

SwitchDutyCycle – switch duty cycle (120 or 180 degrees) for trapezoidal current waveform. This parameter can be empty for sinusoidal current waveform. Refer to [section 4.5](#) of Part I of this user manual for more details.

Coefs3D – structure with internal data of the FE solver with coefficients for considering 3D effects. **Coefs3D** is affected by saturation, so when the current or advance angle are changed there may be a need to recalculate **Coefs3D**. If **Coefs3D** is empty, it will be calculated inside the **runMStimesteppingafm** function before the simulation is started. **Coefs3D** depends on some simulation settings (i.e. input arguments **Current input method**, **Current waveform**, **Advance angle**, etc.), but does not depend on other simulation settings (i.e. input arguments **Rotor speed** and **Number of points**). It means that **Coefs3D** should be recalculated (i.e. should be empty while calling **runMStimesteppingafm**) for each change that can affect saturation in the machine. In some cases, there is no need to recalculate **Coefs3D** – the one can use **Coefs3D** calculated with the previous call of **runMStimesteppingafm** – refer to output argument **Coefs3D** below.

4.2.4. *Outputs*

Results –structure with average magnetostatic FE simulation results. Refer to Table 8 for more details on the **Results** structure fields.

Timesteppingdata –structure with time-stepping magnetostatic FE simulation results. Refer to Table 9 for more details on the **Timesteppingdata** structure fields.

Coefs3D – structure with coefficients for considering 3D effects. Note that function **runMStimesteppingafm** also has an input argument **Coefs3D** (see above). If the input argument **Coefs3D** is empty while calling **runMStimesteppingafm**, it will be calculated inside the **runMStimesteppingafm** function and returned as the output argument **Coefs3D**. Since calculation of the **Coefs3D** structure may be time consuming, the output argument **Coefs3D** can be used for subsequent calls of **runMStimesteppingafm** for the same AFM design. If the input argument **Coefs3D** is not empty while calling **runMStimesteppingafm**, the input and output **Coefs3D** structures will be equal.

Field – structure with field data results.

Private – structure with internal data of the FE solver. Note that function **runMStimesteppingafm** also has an input argument **Private** (see above). If the input argument **Private** is empty while calling **runMStimesteppingafm**, it will be calculated inside the **runMStimesteppingafm** function and returned as the output argument **Private**. Since calculation of the **Private** structure may be time consuming, the output argument **Private** can be used for subsequent calls of **runMStimesteppingafm** for the same AFM design. If the input argument **Private** is not empty while calling **runMStimesteppingafm**, the input and output **Private** structures will be equal.

Table 8. The **Results** structure fields.

Field	Description
SolutionConverged	SolutionConverged = 1 indicates that all points of the simulation converged. If SolutionConverged = 0, there was a problem with the solution convergence.
speed	Rotor speed (RPM).
f1	Supply frequency (Hz).
gamma	Advance angle °(el.deg).
torque	Total torque (N*m).
torque_reluctance	Reluctance torque (N*m).
torque_airgap	Array with torque values for each airgap (N*m).
torque_magnet	Magnet torque (N*m).
Is	Array with two average values: [SupplyCurrent PhaseCurrent] (A).
Id	Average d-axis current (A).
Iq	Average q-axis current (A).
Vs	Array with two average values: [SupplyVoltage PhaseVoltage] (V).
Vd	Average d-axis voltage (V).
Vq	Average q-axis voltage (V).
backEMF	RMS phase back-EMF (V).
Pinput	Input electrical power (W).
Pmech	Output mechanical power (W).
Ps	Stator winding loss (W).

Piron	Total iron core loss (W).
Piron_hyst	Hysteresis iron core loss (W).
Piron_eddy	Iron eddy current loss (W).
MagnetLoss	Magnet loss (W).
OtherECloss	Other eddy current loss (W).
magnetloss_rotor	Magnet losses for each rotor (W).
otherECloss_rotor	Other eddy current losses for each rotor (W).
Piron_tooltip	This field has the decomposition for each stator and rotor iron losses.
PowerFactor	Power factor.
Efficiency	Efficiency (%).
TorqueRipple	Torque ripple (%).
Error	Discretization error (%).
CurrentDensity	Current Density (A/mm ²)
DemagH_max	Max. demagnetization field (A/m).
DemagHpercent_max	Max. demagnetization field (% of H _{cj}).
Ke_rms	Back-EMF constant (V/RPM).
Ke_rad	Back-EMF constant (V*s/rad).
Kv_rms	Velocity constant (RPM/V).
Kv_rad	Velocity constant (rad/(V*s)).
Kt	Torque constant (N*m/A).
Km	Motor constant (N*m/sqrt(W)).

Bav_max_tooth	Peak of stator tooth average flux density (T).
Bav_max_backiron_stator	Peak of stator back iron average flux density (T).
Bav_max_backiron_rotor	Peak of rotor back iron average flux density (T).
Bav_airgap	Average airgap flux density (T).
Bmax_airgap	Maximum airgap flux density (T).
Lself	Phase self-inductance (mH).
Lmutual	Phase mutual inductance (mH).
Ld	D-axis inductance (mH).
Lq	Q-axis inductance (mH).

Table 9. The *Timesteppingdata* structure fields.

Field	Description
time	Time of each calculated point (s).
BackEMFa	Phase A RMS back-EMF (V).
BackEMFb	Phase B RMS back-EMF (V).
BackEFMc	Phase C RMS back-EMF (V).
BackEMFd	D-axis back-EMF (V).
BackEMFq	Q-axis back-EMF (V).
Gamma	Advance angle °(el.deg).
Torque_maxwell	Electromagnetic torque by Maxwell stress tensor (N*m).
Torque_work	Electromagnetic torque by virtual work method (N*m).
Fn_rotor	Electromagnetic normal force on rotor (by Maxwell stress tensor) (N).

Fn_stator	Electromagnetic normal force on stator (by Maxwell stress tensor) (N).
Torque_fluxlinkage	Electromagnetic torque by flux linkage and current (N*m).
Torque_magnet	Magnet torque (by Maxwell stress tensor) (N*m).
Torque_reluctance	Reluctance torque (by Maxwell stress tensor) (N*m).
Torque_airgap	Torque for each airgap (N*m).
Torque_cogging	Cogging torque (by Maxwell stress tensor) (N*m).
Ia	Stator phase A RMS current (A).
Ib	Stator phase B RMS current (A).
Ic	Stator phase C RMS current (A).
Id	D-axis stator current (A).
Iq	Q-axis stator current (A).
Va	Stator phase A RMS voltage (V).
Vb	Stator phase B RMS voltage (V).
Vc	Stator phase C RMS voltage (V).
Vd	D-axis stator phase voltage (V).
Vq	Q-axis stator phase voltage (V).
Fluxlinkage_a	Flux linkage phase A.
Fluxlinkage_b	Flux linkage phase B.
Fluxlinkage_c	Flux linkage phase C.
Fluxlinkage_d	Flux linkage d-axis.
Fluxlinkage_q	Flux linkage q-axis.

Pinput	Electrical input power (W).
Pmech	Output mechanical power (W).
Ps	Stator winding losses (W).
MagnetLoss	Magnet losses (W).
OtherECloss	Other eddy current losses (W).
Convergence.converged	Indicating whether the solution has converged (1) or not (0) for each time step for each cylindrical mesh slice.
Convergence.residual	Residual of the FEA solution for each time step for each cylindrical mesh slice.
Convergence.message	Message that may indicate the reason for non-convergence.

4.3. Parametric sweep script example

In this section, it is explained how to run a parametric sweep analysis using an example of a rotor-stator-rotor motor. The analysis involves running multiple simulations to examine how the motor's efficiency is impacted by variations of the *SlotWidth* and *StatorHeight* parameters. To provide an overview of the process, a code diagram is shown in Figure 2, and the complete code can be found in *CustomScripts/ExampleParametricSweep.m*.

To start the parametric sweep analysis, the script is initialized by calling [initMXAscriptingAPI](#), and the initial mxa-file project is opened. Two arrays are then created, each containing six possible values for the parameters being swept: *SlotWidth* and *StatorHeight*. This generates a total of 36 possible combinations of parameter variations, with each combination resulting in a simulation run.

To set the initial values for the magnetostatic FE simulations and define the model geometry, the function [setParamafm](#) is used. With the updated structures, they are then used to assemble the MXA design using the [assembleMXA](#) function.

To vary the *SlotWidth* and *StatorHeight* parameters, two for-loops are created. To ensure that all the designs are compared under the same conditions, another loop is used to adjust the input stator current until the torque is approximately equal to the assigned target torque. It's worth noting that the initial magnetostatic simulations are performed using only one rotor position. During this stage, two additional structures are calculated and stored to improve the total simulation time: *Private* and *Coefs3D*. The *Private* structure only needs to be redefined when the geometry changes, while the *Coefs3D* is recalculated every time as the current changes since these coefficients are affected by the motor's saturation.

After the first stage, the simulation with multiple points is run using [runMStimesteppingafm](#). If all time steps converge, the results are saved, and the process is repeated with for the new combination of *SlotWidth* and *StatorHeight* values until all possible combinations have been processed.

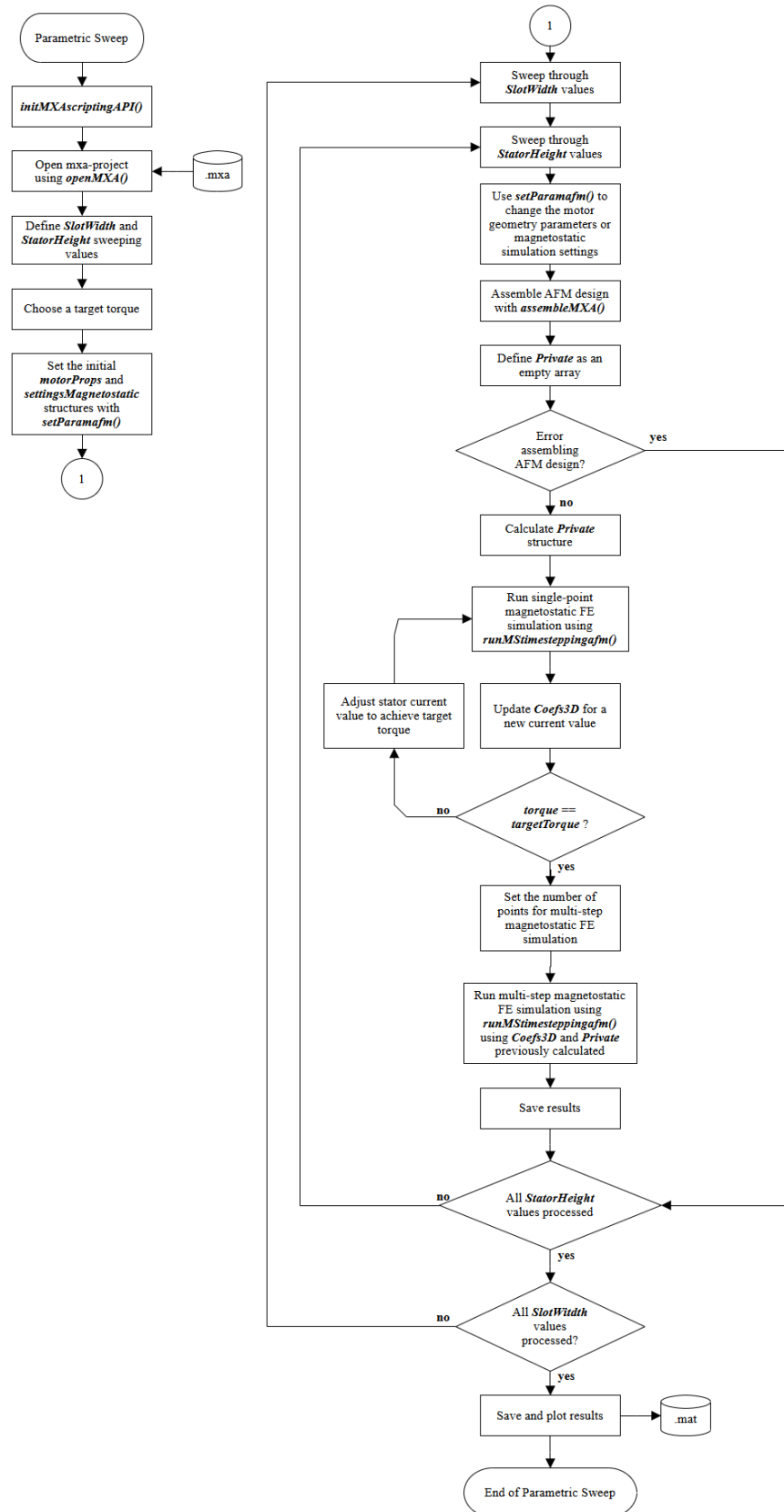


Figure 2. Parametric sweep code diagram.

5. Parallel processing functions

5.1. Assembling AFM designs in parallel

5.1.1. *initAssembleMXA_par* function

The *initAssembleMXA_par* function initializes assembling AFM designs in parallel.

5.1.1.1. Syntax

```
initAssembleMXA_par()
```

5.1.2. *runAssembleMXA_par* function

5.1.2.1. Syntax

```
runAssembleMXA_par(initMXAfile, motorProps)
```

5.1.2.2. Description

Function *runAssembleMXA_par* is used to assemble several AFM designs in parallel.

The *runAssembleMXA_par* function is similar to the [assembleMXA](#) function and has the same input arguments. Each time *runAssembleMXA_par* is called, it initiates a new MotorXP-AFM process for assembling one AFM design which enables parallel processing, i.e. assembling several AFM designs at the same time.

5.1.2.3. Inputs

There are two inputs: [motorProps](#) structure and the file path of the initial mxa-file project.

5.1.2.4. Outputs

There are no outputs; *getAssembleMXA_par* needs to be called to get the resulting cell arrays containing data for each MXA design.

5.1.3. *getAssembleMXA_par* function

5.1.3.1. Syntax

```
[cell_Geometry, cell_Windings, cell_Mesh,
 cell_Materials, cell_RadMesh, cell_SubdomainProperty,
 cell_Error, cell_motorProps_output, cell_Weight, numberOfProcesses] =
    getAssembleMXA_par()
```

5.1.3.2. Description

The *getAssembleMXA_par* function should be called after calling *runAssembleMXA_par* to get the cell arrays with assembled AFM designs. The number of cells in each output cell array is equal to the number of the *runAssembleMXA_par* function calls.

5.1.3.3. Outputs

The output arguments of *getAssembleMXA_par* are similar to the output arguments of the [assembleMXA](#) function. The difference is that each output of *getAssembleMXA_par* is a cell array where each cell contains data equivalent to outputs of *assembleMXA*.

cell_Geometry, *cell_Windings*, *cell_Mesh*, *cell_Materials*, *cell_RadMesh*, *cell_SubdomainProperty* - cell arrays of the structures containing each assembled AFM design data. Refer to [section 4.1.4](#) for more details.

cell_Error - cell array of the error messages if the corresponding AFM design failed assembling. Refer to description of **Error** in [section 4.1.4](#) for more details.

cell_motorProps_output - cell array of the output [motorProps](#) structures for each AFM design. Refer to description of **motorProps_output** in [section 4.1.4](#) for more details.

cell_Weight - cell array of the active material weights for each AFM design. Refer to description of **Weight** in [section 4.1.4](#) for more details.

numberOfProcesses - number of the MotorXP-AFM processes initiated by the **runAssembleMXA_par** function calls currently running. Once the AFM design assembling is finished, the corresponding process is terminated. The **numberOfProcesses** variable shows how many AFM designs are currently being assembled.

5.2. Running magnetostatic FE simulations in parallel

5.2.1. *initMStimesteppingafm_par* function

The **initAssembleMXA_par** function initializes running magnetostatic FE simulations in parallel.

5.2.1.1. *Syntax*

```
initMStimesteppingafm_par()
```

5.2.2. *runMStimesteppingafm_par* function

5.2.2.1. *Syntax*

```
runMStimesteppingafm_par(Private, Geometry, Mesh, Windings,  
settingsMagnetostatic, RadMesh, Materials, SubdomainProperty,  
SwitchDutyCycle, Coefs3D)
```

5.2.2.2. *Description*

Function **runMStimesteppingafm_par** is used to run several magnetostatic FE simulations in parallel. The **runMStimesteppingafm_par** function is similar to the [runMStimesteppingafm](#) function and has the same input arguments. Each time **runMStimesteppingafm_par** is called, it initiates a new MotorXP-AFM process for running one magnetostatic FE simulation which enables parallel processing, i.e. running several magnetostatic FE simulations at the same time.

5.2.2.3. *Inputs*

Private - structure with internal data of the FE solver. Same as described in [section 4.2.3](#).

Geometry, Windings, Mesh, Materials, RadMesh, SubdomainProperty - structures defining the AFM design. Same as described in [section 4.2.3](#).

settingsMagnetostatic - magnetostatic FE simulation settings. Refer to [section 3.3](#) for more details.

SwitchDutyCycle - switch duty cycle (120 or 180 degrees) for trapezoidal current waveform. This parameter can be empty for sinusoidal current waveform. Refer to [section 4.5](#) of Part I of this user manual for more details.

Coefs3D - coefficients for considering 3D effects. Same as described in [section 4.2.3](#).

5.2.3. *getMStimesteppingafm_par* function

5.2.3.1. *Syntax*

```
[cell_Results, cell_Timesteppingdata, cell_Coefs3D, cell_field,
 cell_Private, numberOfProcesses] = getMStimesteppingafm_par()
```

5.2.3.2. *Description*

This function should be called after calling function *runMStimesteppingafm_par* to get the cell arrays with magnetostatic FE simulation results.

5.2.3.3. *Outputs*

cell_Results – cell array of the structures with average magnetostatic FE simulation results. Refer to description of **Results** in [section 4.2.4](#) for more details.

cell_Timesteppingdata – cell array of the structures with time-stepping magnetostatic FE simulation results. Refer to description of **Timesteppingdata** in [section 4.2.4](#) for more details.

cell_Coefs3D – cell array of the structures with coefficients for considering 3D effects. Refer to description of **Coefs3D** in [section 4.2.4](#) for more details.

cell_field – cell array of the structures with field data results.

cell_Private – cell array of the structures with internal data of the FE solver for each AFM design. Refer to description of **Private** in [section 4.2.4](#) for more details.

numberOfProcesses – number of the MotorXP-AFM processes initiated by the *runMStimesteppingafm_par* function calls currently running. Once one of the magnetostatic FE simulations is finished, the corresponding process is terminated. The **numberOfProcesses** variable shows how many magnetostatic FE simulations are currently running.

5.3. Parametric sweep script example with parallel processing

This section explains how to run a parametric sweep with parallel processing to speed up the analysis. This example implements the same process as presented in [section 4.3](#), but, instead of the serial processing with functions [assembleMXA](#) and [runMStimesteppingafm](#), their parallel processing analogs described above are used. To provide an overview of the process, a code diagram is shown in Figure 4, and the complete code can be found in *CustomScripts/ExampleParametricSweep_par.m*.

To perform the parametric sweep analysis, the script first calls the [initMXAscriptingAPI](#) function to initialize the scripting API and then opens the initial mxa-file project. Two arrays are created (**SlotWidth** and **StatorHeight**), each containing six values to be swept. This generates a total of 36 possible combinations of parameter variations, with each combination resulting in a simulation run and the results are saved in corresponding data arrays. Additionally, the maximum number of MotorXP-AFM processes that can be run in parallel (variable **maxNumberOfProcesses**) is specified at this stage. When defining the **maxNumberOfProcesses** value, it is recommended to consider the number of physical CPU cores and the amount of RAM. Figure 3 shows the case when the number of MotorXP-AFM processes is equal to seven.

The [runAssembleMXA_par](#) function is used to start assembling the AFM designs in parallel, while the [getAssembleMXA_par](#) function is used to retrieve the designs data until all AFM designs are ready. Afterward, the designs are verified to ensure that each was created correctly. The parallel processing helps to reduce the overall analysis time, as the AFM designs can be assembled simultaneously instead of sequentially.

The subsequent stage focuses on adjusting the stator currents for each AFM design through parallel simulations, using the [runMStimesteppingafm_par](#) and [getMStimesteppingafm_par](#) functions. This process retains the maximum number of parallel processes until all AFM designs achieve the target

torque. During this stage, the *Private* structure is utilized with the first call *runMStimesteppingafm_par* to reduce the overall simulation time. Finally, *runMStimesteppingafm_par* is executed with the desired accuracy for multi-step FE simulation and the results are kept in corresponding cell arrays.

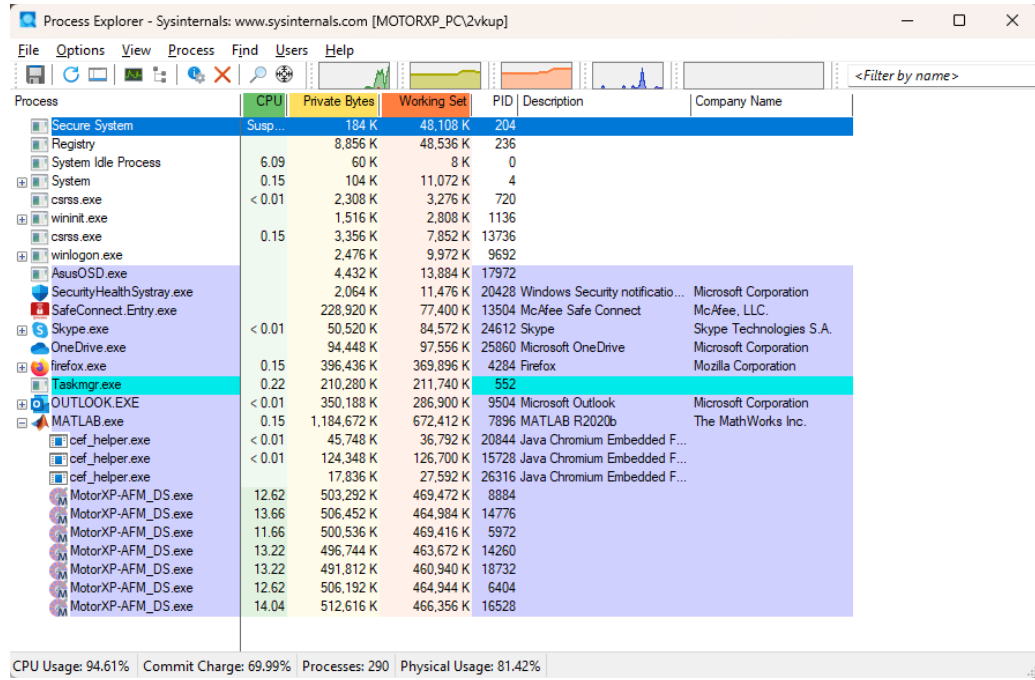


Figure 3. Seven MotorXP-AFM processes running under the host MATLAB process.

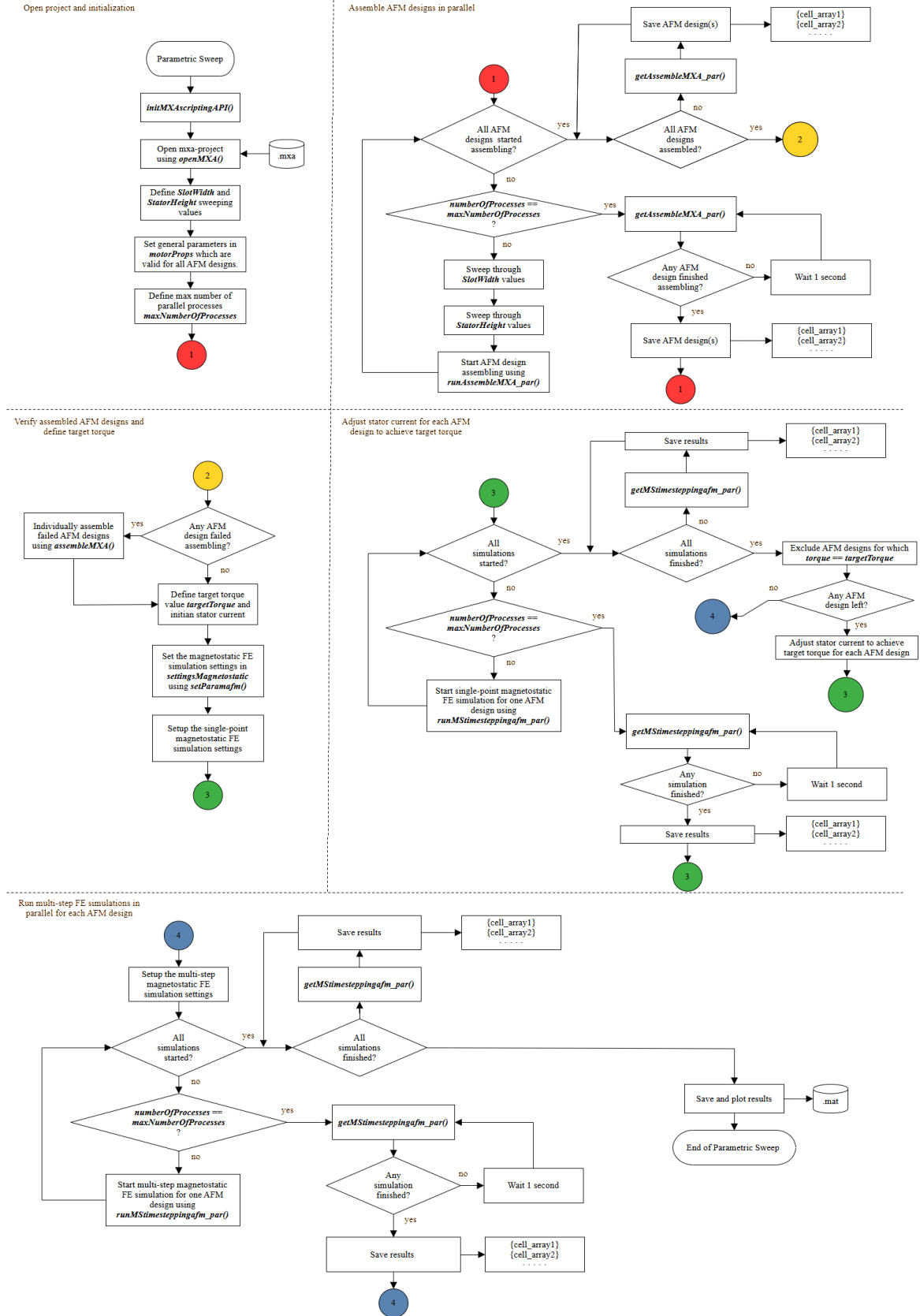


Figure 4. Parametric sweep code diagram with parallel processing.

6. Automatic optimization workflows

6.1. Overview

Electric machine automatic optimization is a process that involves using advanced software and algorithms to optimize the design of electric machines for maximum efficiency and performance. The idea behind this process is to modify the machine's geometry (and maybe other parameters) to achieve the desired performance metrics, such as torque, power, and efficiency, while also reducing weight, costs and environmental impact. The concept of electric machine automatic optimization is based on the principle that small changes to the machine's geometry can have a significant impact on its performance.

By optimizing the design of the machine, it is possible to achieve higher levels of efficiency and performance, while also reducing energy consumption and weight as well minimizing cost and environmental impact. The process of electric machine automatic optimization involves several steps. The first step is to define the performance metrics that the machine should achieve. This information is used to guide the optimization process and ensure that the machine meets the desired performance targets.

The next step is to identify the design variables that can be modified to achieve the desired performance metrics. These variables may include the shape and size of the rotor and stator, the number of poles, the winding configuration, and other factors. Once the design variables have been identified, a model of the machine is created using simulation and analysis software. Optimization algorithms are then used to find the optimal combination of design variables that will achieve the desired performance metrics. After the optimization process is complete, the optimized machine design is verified and validated using simulation and analysis software. This step involves simulating the machine's performance under various operating conditions to ensure that it meets the desired performance criteria.

Electric machine automatic optimization has numerous benefits, including increased efficiency, improved performance, reduced mass and cost. This process also leads to a positive impact on the environment, making it an essential tool in the push towards sustainability and energy efficiency. Overall, electric machine automatic optimization is a powerful process that can help to optimize the performance and efficiency of electric machines, leading to significant benefits for both manufacturers and end-users alike.

6.2. Functions for automatic optimization workflow development

There are several functions in the MotorXP-AFM MATLAB scripting API for developing the custom automatic optimization workflows and visualizing the optimization results. By default, the “Thompson sampling efficient multiobjective optimization” (TSEMO) algorithm is used adapted for the electric machine optimization tasks with parallel processing. The original version of the algorithm can be found at <https://www.mathworks.com/matlabcentral/fileexchange/66588-multi-objective-optimization-algorithm-for-expensive-to-evaluate-function>. The adapted version of the optimization algorithm function can be found in `\m\scriptapi\TSEMO\TSEMO_V4.m`.

6.2.1. *Suggested optimization workflow structure*

The suggested optimization workflow structure includes the main script defining basic parameters such as number of processes (FE simulations) running in parallel, number of optimization algorithm iterations, total number of AFM designs to be evaluated with FEA, initial or basic AFM design which needs to be optimized, design variable bounds and optimization objectives. The main script also specifies the function implementing the FEA evaluation of the generated AFM designs defined as a MATLAB function handle `@designFunction`. The *designFunction* is called on each iteration of the optimization algorithm receiving the design variable values and returning the objective values for each AFM design. Finally, the main script calls the *runTSEMOOptimizationafm* function executing the optimization algorithm.

6.2.2. *runTSEMOptimizationafm* function

6.2.2.1. *Syntax*

```
runTSEMOptimizationafm(designFunction, initdesignfile, ub, lb,  
no_outputs, init_dataset_size, maxeval, niter, maxNumberOfProcesses,  
save2filename, y_refdesign, labelObjective)
```

6.2.2.2. *Description*

This function executes the optimization algorithm and displays the results as the Pareto plot. The corresponding optimization algorithm diagram is shown in Figure 5. The optimization algorithm is a combination of Gaussian process surrogate models with NSGA-II genetic algorithm which demonstrates very good optimization performance. The optimization results are saved to a mat-file on each iteration of the algorithm. It allows to continue the interrupted optimization from the point where it previously left off. If, however, this is the first run, the latin hypercube sampling is employed to produce a random collection of initial designs. Then, the **@designFunction** is invoked to evaluate AFM designs generated on each iteration of the optimization algorithm.

6.2.2.3. *Inputs*

designFunction – MATLAB function handle processing generated AFM designs.

initdesignfile – initial mxa-file (base design).

ub and **lb** – lower and upper bounds on design variables.

no_outputs – number of the optimization objectives.

init_dataset – number of initial AFM designs randomly generated using latin hypercube sampling.

maxeval – total number of generated AFM designs to be processed with **designFunction**.

niter – number of optimization algorithm iterations.

maxNumberOfProcesses – maximum number of MotorXP-AFM processes that can be run in parallel; the same as described in [section 5.3](#).

save2filename – save optimization algorithm results to this mat-file.

y_refdesign – (optional) reference design objectives to be displayed on the Pareto plot as a red **x**.

labelObjective – (optional) objective names to be displayed on the Pareto plot.

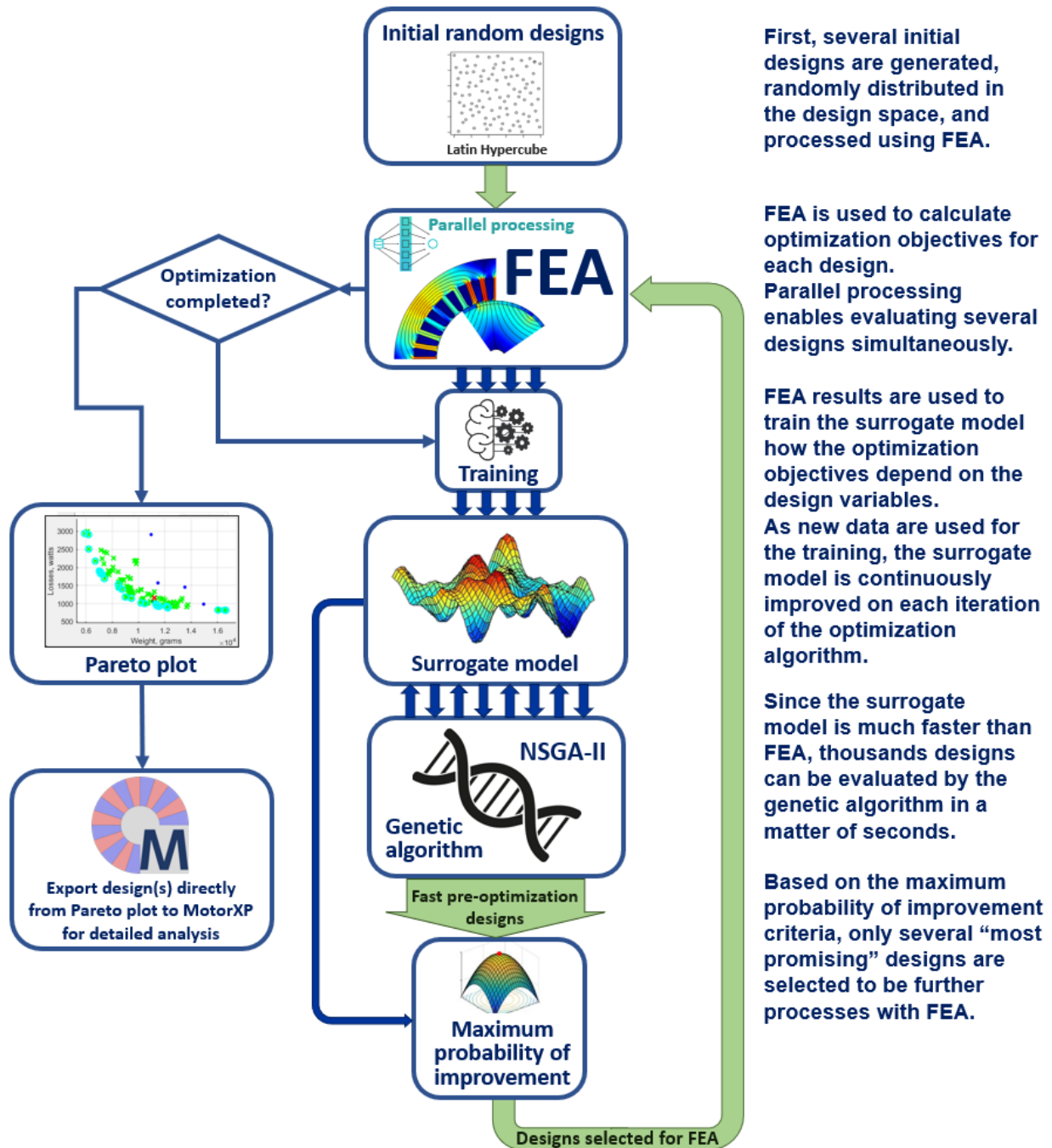


Figure 5. Optimization algorithm flowchart.

6.2.3. Design function (on an example of the *evalDesigns_emrax228* function)

6.2.3.1. *Syntax*

```
[Objectives, motorProps] = evalDesigns_emrax228(motorProps,
settingsMagnetostatic, file, maxNumberOfProcesses, designParams, msg0)
```

6.2.3.2. *Description*

This function is used for evaluating every generated AFM design. The ***designFunction*** is called on each iteration of the optimization algorithm receiving the design variable values and returning the objective values for each AFM design.

The source code of *evalDesigns_emrax228* can be found in `\CustomScripts\evalDesigns_emrax228.m`. The algorithm implemented in the *evalDesigns_emrax228* function is similar to that illustrated in Figure 4. Initially, all the AFM designs are assembled using the parallel processing functions described in [section 5](#). Then, the target mechanical speed is set up for each AFM design and the stator current of each design is adjusted to produce the target torque. To do that, the functions for running the magnetostatic FE simulations in parallel described in [section 5.2](#) are used. Finally, after the stator currents are determined, the multi-step magnetostatic FE simulations are performed for each AFM design. To speed up the analysis, the number of rotor positions for the simulation is set to 6 and the simulation time is set to 1/6th of the electrical period. It was previously determined for the motor under study that these simulation settings provide reasonable accuracy for the motor parameters used for the analysis such as torque, eddy current iron losses and magnet losses, except the hysteresis iron losses. For the hysteresis iron losses, the empirical correction coefficient of 2.75 is used.

6.2.3.3. *Inputs*

motorProps – see [section 3.2](#).

settingsMagnetostatic – see [section 3.3](#).

file – initial mxa-file (base design).

maxNumberOfProcesses – maximum number of MotorXP-AFM processes that can be run in parallel; the same as described in [section 5.3](#).

designParams – design variable values for each AFM design generated by the optimization algorithm.

msg0 – text message showing the optimization algorithm current iteration number to be displayed in the MATLAB command window.

6.2.3.4. *Outputs*

motorProps – see [section 3.2](#).

Objectives – optimization objective values for each AFM design.



6.2.4. *plotParetoFromFile* function

6.2.4.1. *Syntax*

```
plotParetoFromFile(fileOptimizationData)
```

6.2.4.2. *Description*

This function is used to display the Pareto plot for the optimization algorithm results saved in mat-file ***fileOptimizationData***. Inside the ***plotParetoFromFile*** the original ***plotPareto*** function is called. Figure 6 shows an example of the Pareto plot. The green **x** are the AFM designs generated by the optimization algorithm, the blue dots **•** are the initial AFM designs randomly generated using Latin hypercube

sampling before the optimization started, the cyan circles  are indicating the Pareto front and the red  is the reference design, which can be optionally displayed on the Pareto plot for comparison purposes. Right-clicking on the Pareto plot design point opens the context menu allowing to save the selected design to the mxa-file for further analysis, as shown in Figure 6. The full source code how this feature is implemented can be found in:

```
\m\scriptapi\plotPareto.m
\m\scriptapi\ParetoMXArighclickmenu.m
```

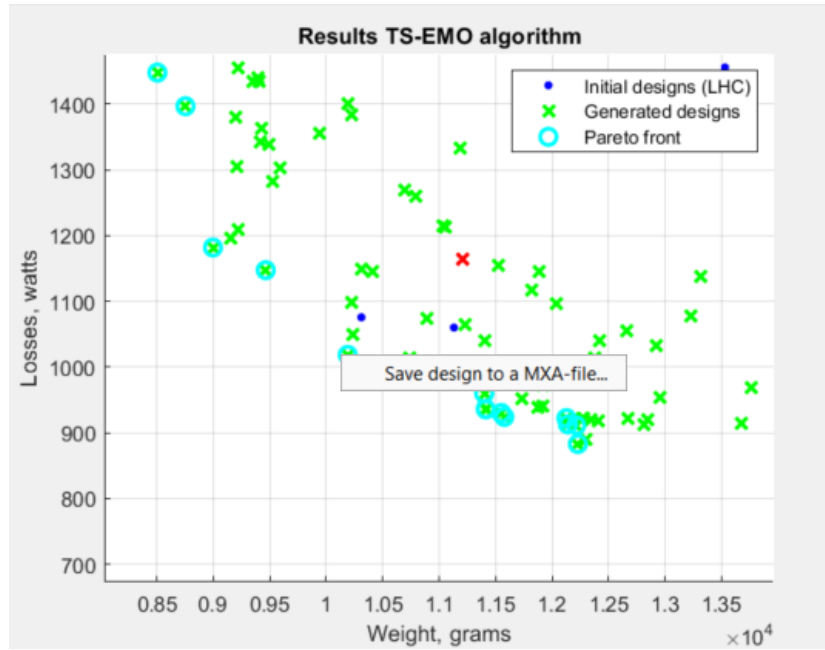


Figure 6. Example of the Pareto plot with the “Save design to a MXA-file” right-click context menu option.

6.3. Example of the automatic optimization workflow implementation

This section outlines the steps required to implement an optimization workflow. Figure 7 provides a schematic overview of the code, and the full source code can be found in:

```
CustomScripts\runTSEMO_MXPAFM_emrax228.m
CustomScripts\evalDesigns_emrax228.m
```

This example implements the automatic optimization of an axial flux motor minimizing two objectives (motor weight and motor losses for the target operating point of 150 Nm and 1600 RPM) by varying several geometry parameters of the motor. The EMRAX-228 motor is used as a reference design for this example (see file *SimFiles/EMRAX-228/EMRAX-228_forOptimization.mxa*). The parallel processing is used to speed up the optimization process allowing running FE simulations for several designs at the same time. The *evalDesigns_emrax228* function, previously described in [section 6.2.3](#), is used by the optimization algorithm for processing generated AFM designs and calculating the objective values.

As can be seen from Figure 7, some AFM designs are punished (motor weight is set to 18kg) if the motor losses exceed 3000W. The punishment is especially useful if the design space is wide enough, i.e., geometry parameters vary in a wide range. In this case, the optimization algorithm tends to minimize the motor weight rather than losses as the relationship between motor weight and its geometry is much more straightforward. Purposely increasing the weight as a punishment for too high losses provides additional “motivation” for the optimization algorithm to minimize losses.

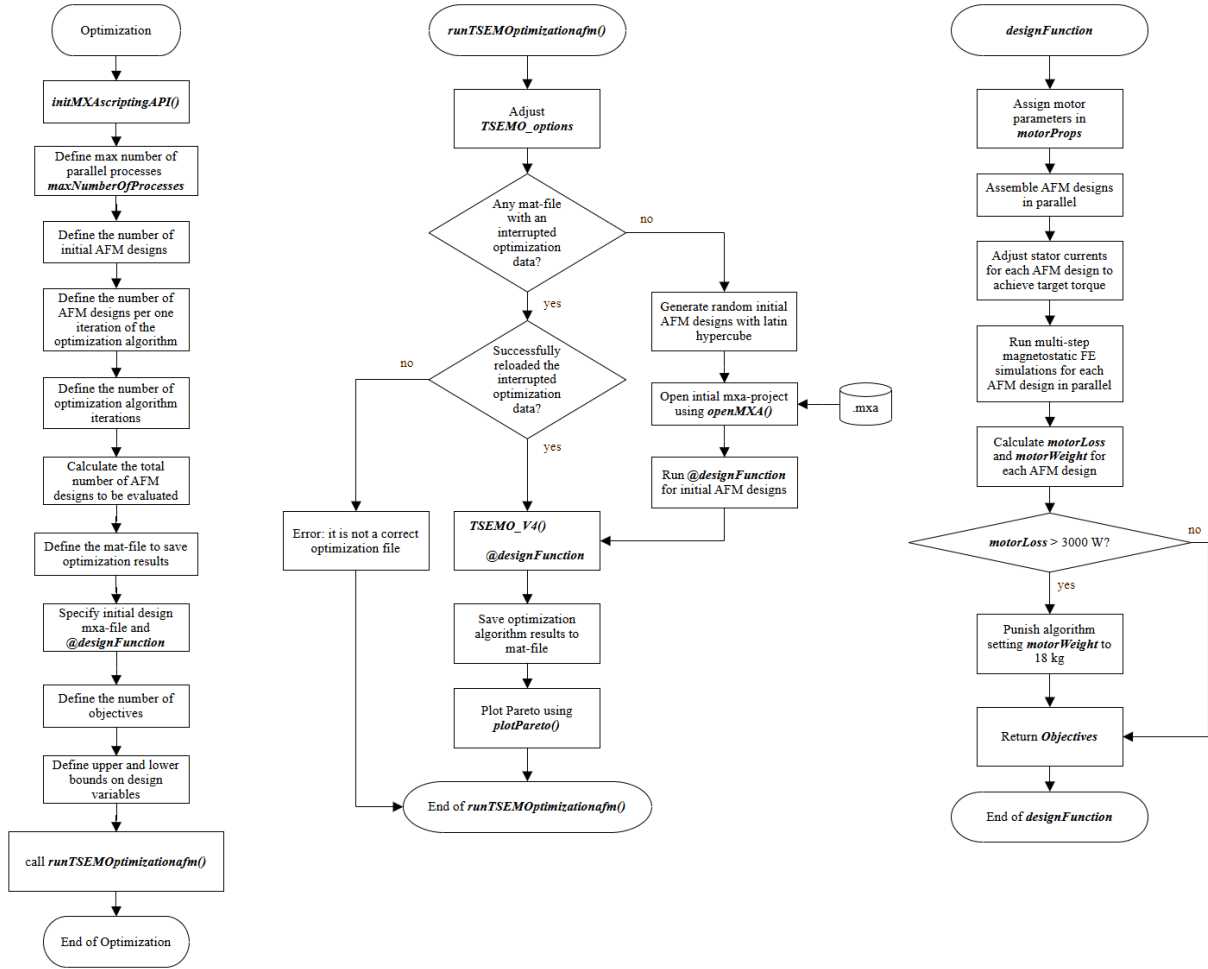


Figure 7. Code diagram of the optimization workflow example.

7. MATLAB code debugging tips

There can be some difficulties while debugging the MATLAB code that is using the parallel processing functions of the MotorXP-AFM scripting API. In case an error occurs, most of the time the [getAssembleMXA par](#) and [getMStimesteppingafm par](#) functions retrieve empty data structures without showing the actual error message. It allows continuing the execution of the code without interruption but creates difficulties understanding the reason for the error. The good programming practice in this case is to check for empty data and repeat the operation with an analog serial processing function. For example, the following code is used in the [evalDesigns emrax228](#) function to handle the AFM designs which failed assembling with parallel processing:

```
% check if any design failed assembling and try again
for i = 1:nDesigns
    if isempty(cell_Geometry_ready{i})
        motorProps = getMotorProps(motorProps,designParams,i);
        [Geometry, Windings, Mesh, Materials, RadMesh, SubdomainProperty, Error, motorProps_output, Weight] = assembleMXA(file,motorProps);
        if isempty(Geometry)
            cell_Error_ready{i} = 'Unknown error';
        else
            cell_Geometry_ready{i} = Geometry;
            cell_Windings_ready{i} = Windings;
            cell_Mesh_ready{i} = Mesh;
            cell_Materials_ready{i} = Materials;
            cell_RadMesh_ready{i} = RadMesh;
            cell_SubdomainProperty_ready{i} = SubdomainProperty;
            cell_Error_ready{i} = Error;
            cell_motorProps_output_ready{i} = motorProps_output;
            cell_Weight_ready{i} = Weight;
        end
    end
end
end
```

The MATLAB *try-catch* construction with break point put on the *catch* statement can also be used for code debugging and analyzing errors. The following code and break point are used for debugging the possible errors while running magnetostatic FE simulations in the [evalDesigns emrax228](#) function:

```
323 % check if any simulation failed and try again
324 for i = 1:nDesigns
325     Error = cell_Error{i};
326     if isempty(Error) && enableRun(i)
327         if isempty(cell_Results_ready{i})
328             Geometry = cell_Geometry{i};
329             Windings = cell_Windings{i};
330             Mesh = cell_Mesh{i};
331             Materials = cell_Materials{i};
332             RadMesh = cell_RadMesh{i};
333             SubdomainProperty = cell_SubdomainProperty{i};
334             settingsMagnetostatic=setParamafm(settingsMagnetostatic,'RMS supply current',arr_Current(i));
335             try
336                 [Results, Timesteppingdata] = runMStimesteppingafm([],Geometry,Mesh,Windings,settingsMagnetostatic,RadMesh,Materials,SubdomainProperty,0);
337             catch ME
338                 rethrow(ME)
339             end
340             cell_Results_ready{i} = Results;
341             cell_Timesteppingdata_ready{i} = Timesteppingdata;
342         end
343     end
344 end
```

It should be kept in mind that not all combinations of design variables result in valid design and not all invalid designs can be automatically recognized. It is the responsibility of the user of the current API to check for results validity, such empty results, NaN or zero values.

Finally, there are log files recording all the operations of the program, which can also help identifying possible problems. They can be found in:

```
C:\Users\[username]\AppData\Local\VepcoTech\MotorXP-AFM Design Studio\logs
C:\Users\[username]\AppData\Local\VepcoTech\MotorXP-AFM_Stub\logs
```