Flux coupled with Activate

2D Multiphysics Summary

This document describes Flux-Activate coupling on the example of an interior permanent magnet (IPM) motor with a simple vector/field-oriented control system. The Activate model contains different levels of fidelity of the electric machine based on results from either FluxMotor or Flux and the implementation of these different methods with Activate is the highlight of this tutorial.

The first type of model is an analytical approach with the Park model (classical first approach based on the Park/Clarke transformations into direct/quadrature directions for Vector/Field-Oriented Control). This model is based on constant values for the parameters in the model (inductance, etc.) and provides the lowest level of fidelity of these methods, but with fastest simulation speed.

The second modeling approach uses magneto-static look-up tables (LuT) calculated with finite element Altair Flux or FluxMotor, which typically improves the accuracy compared to first method. Here there are two different models – one based on FluxMotor with dependency on current (Id, Iq, direct and quadrature currents, respectively), and one based on Flux (adding rotor angle as a third input to the look-up table). Compared to using FluxMotor, by using Flux, we can add rotor-angle dependency to capture more accurate torque ripple. The LuT models are both relatively fast compared to co-simulation and have improved accuracy compared to the simplified analytical approach of the Park model.

The last modeling approach uses the highest level of accuracy via co-simulation between the finite element software in transient mode and Activate. The case of co-simulation depending on the type of the kinematics, there are two cases: with coupled load and multiphysics position. While this is the most accurate method, it also is the most expensive computationally and may take much longer to simulate compared to the LuT or Park models above.

A comparative study has been made to show what are the advantages/disadvantages of each method.

Keywords

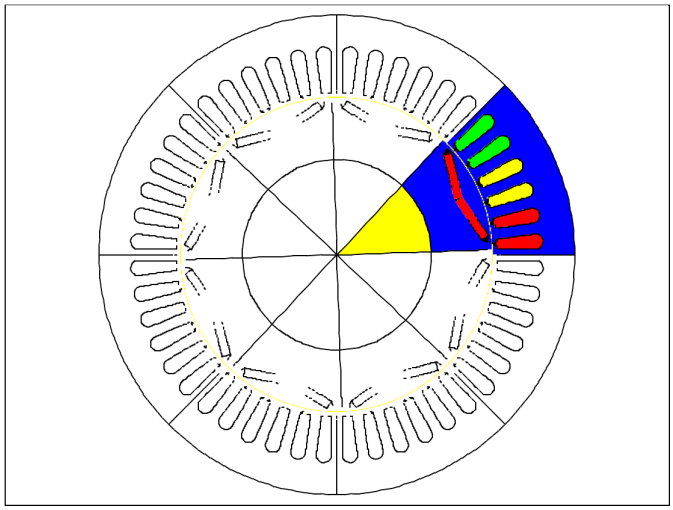
Applications	Flux main functions	Post-processed quantities
• Magneto Static • Transient Magnetic	• Rotating motion, Mechanical set • Kin. = multi-static, coupled load and multiphysics position • Generate tables • Generate Activate coupled component for co-simulation • Sensor: magnet losses. To create the sensor from: **Parameter/Quantity** > **Sensor** > **Magnet losses** > **Losses by Joule effect on face regions.**	• Magnetic quantities • Kinematic quantities, Circuit quantities • Spectral analysis, Back electromotive force (bemf)

Studied device

The studied device, a brushless AC permanent magnet synchronous machine (PMSM, particularly an interior permanent magnet machine, IPM) presented in the figure below, includes the following elements:

- a fixed part (stator) including yoke, slots, and windings
- an air gap
- a movable part (rotor) with embedded magnets

A section of the model of the studied device is presented in the figure below.



In practice

Open example = Open Flux + Run the pyFlux command file

- Recommended memory configuration (standard): 1000 MiB Numerical + 50 MiB Character + 300 MiB GUI
- Computation time: 7 min < t < 4h [64 bit - 16 GB RAM - 2.2 GHz]

Application of the method into a complex model: vector control of a PMSM

In a first step we have implemented a vector command of permanent magnet synchronous motor (PMSM) via an analytical Park model into Activate, in a second step we have implemented two types of look-up table methods, and finally we have used the full co-simulation with Flux.

The system set could be decomposed in four parts:

1. The target, or desired, speed command
2. The speed controller (vector/field-oriented control)
3. The current controller (vector/field-oriented control)
4. The PMSM (magnetic and mechanical part)

Speed Command

The regulation is done with regulation over the speed of the machine. So, the target speed command is provided as input to the control system.

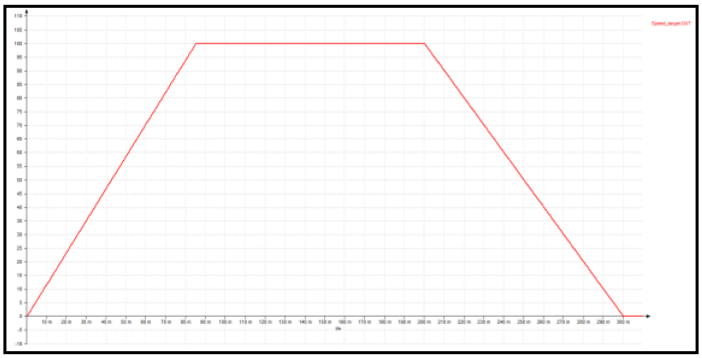


Figure 1: Speed command

The Speed and Current Controllers

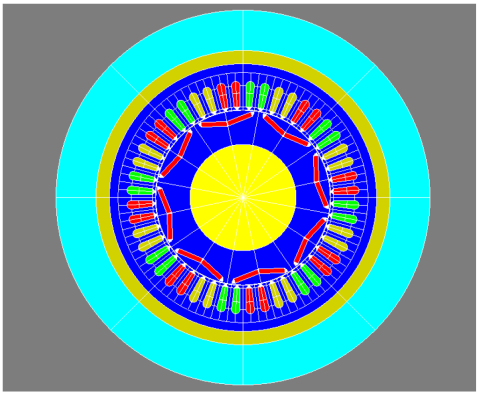
The overall aim of the regulation is to produce applied voltages to the PMSM in order to set close to zero the error between target speed and actual speed. This regulation is done with two PI controllers in cascade (speed controller provides desired current to current controller, which provides input voltages to PMSM in phases A, B, C, converted by Park/Clarke transformation from the vector control in the direct (d) and quadrature (q) frame). Actual speed and actual current are measured and provided for feedback control to the speed and current PI controllers, respectively.

The PMSM model

As we have seen above, we will apply phase voltages from the control system to a magnet synchronous machine modeled with three different methods: analytical (Park), magnetostatic look-up table, and Flux co-simulation.

We will compare those three methods with the following characteristics:

- Elaboration time to set the complete model
- Simulation time
- Accuracy



Characteristics:

Geometry:
- 8 POLES
- 48 slots
- Rotor radius=92mm
- Stator radius=141mm
- Depth=75mm

Electric characteristics:
- Power supply: DC source 550V
- Peak power rating: 50 kW at 1200-1500 rpm

Figure 2: Flux model

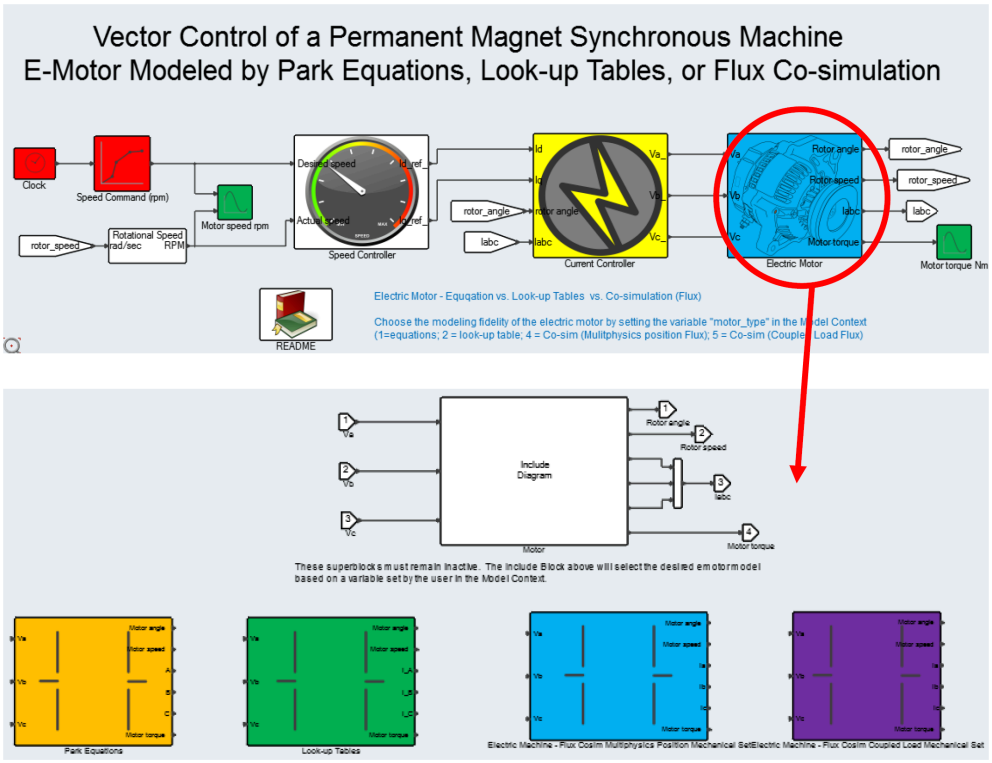
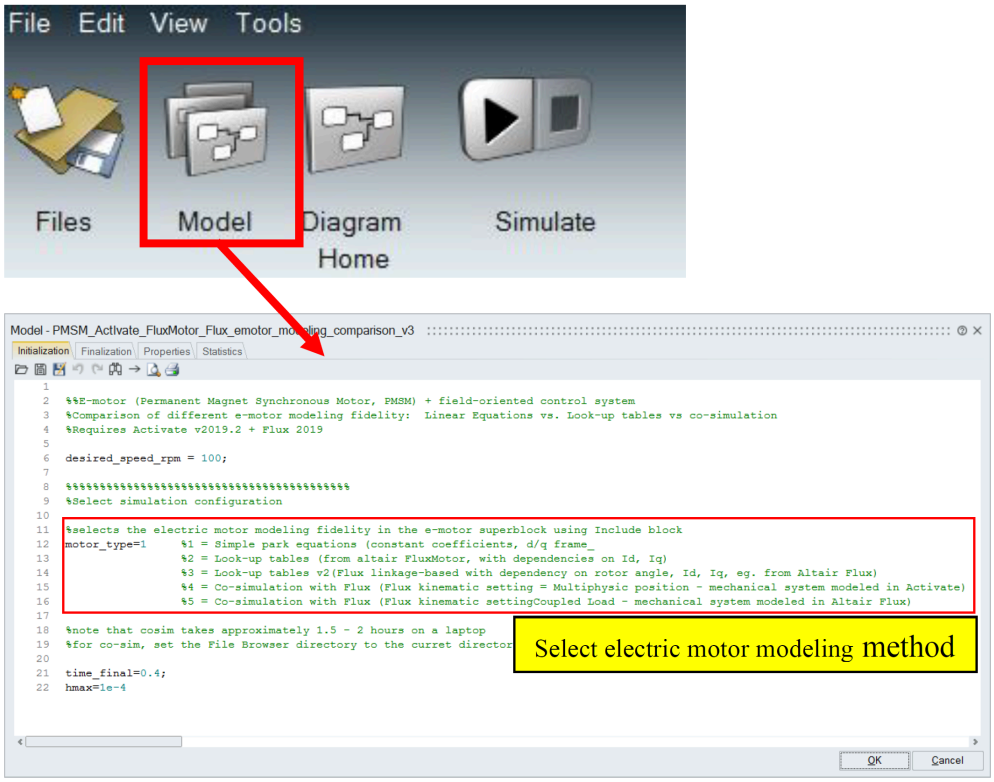


Figure 3: Illustration of the Activate model

To select the type of modeling fidelity of the PMSM, you need to follow the steps below:

Click on the **Model** Context button:



Modify the variable *motor_type* to select the modeling fidelity of the PMSM (=1, 2. Etc) – see the comments in this model which details this). This variable is used by the e-motor superblock in the Activate model to select the motor modeling fidelity. In particular, the variable is used by an Include Block in Activate, which selects the existing deactivated superblock which contains the e-motor modeling desired. The superblock is selected, it is automatically activated for the simulation. These models are "white box" models – you may open them and review and potentially customize them to your own needs and can add other models as desired.

Example 1: Analytic approaches (Park equation model)

In this case, Activate contains a model of the PMSM based on a set of equations, in this case, the Park model of the motor is implemented, which is given by the equations below:

Electrical equations:

$$V_d = R_s * I_d + \frac{d\psi_d}{dt} - \omega * \psi_q$$

$$V_q = R_s * I_q + \frac{d\psi_q}{dt} + \omega * \psi_d$$

Magnetic equations:

$$\psi_q = I_q * L_q$$

$$\psi_d = I_d * L_d + \psi_{PM}$$

Magnetic Torque:

$$\Gamma = \frac{3}{2} p * (\psi_d * I_q - \psi_q * I_d)$$

Ld, Lq and Phi_PM are calculated using The value of the Park model are given in the table below:

Parameters	Value
P (pole pair number)	4
Ld (direct axis inductance)	2.11e-3
Lq (quadrature axis inductance)	8.98 e-3
R (phase resistance)	0.088
Phi_M (magnet flux)	0.366

To get these values, in the first the magnet flux (Phi_M) form the no load test you compute the flux in one phase and you see the maximum value. For inductance in dq axis, you need to open the case5 of IPM motor from Flux supervisor example, you compute the value of Lq for 30 degree position and zero current, for the Ld you compute the value of 52.5 degree position and zero current.

The Park model is easy to use and the simulation is quick, however the accuracy could be low since the motor parameters are simplified to be constant values, which may only be accurate for an operating

point and small deviations. The motor model can be improved by adding parameter dependency on other variables in the system via look-up tables. This will be shown in the next example.

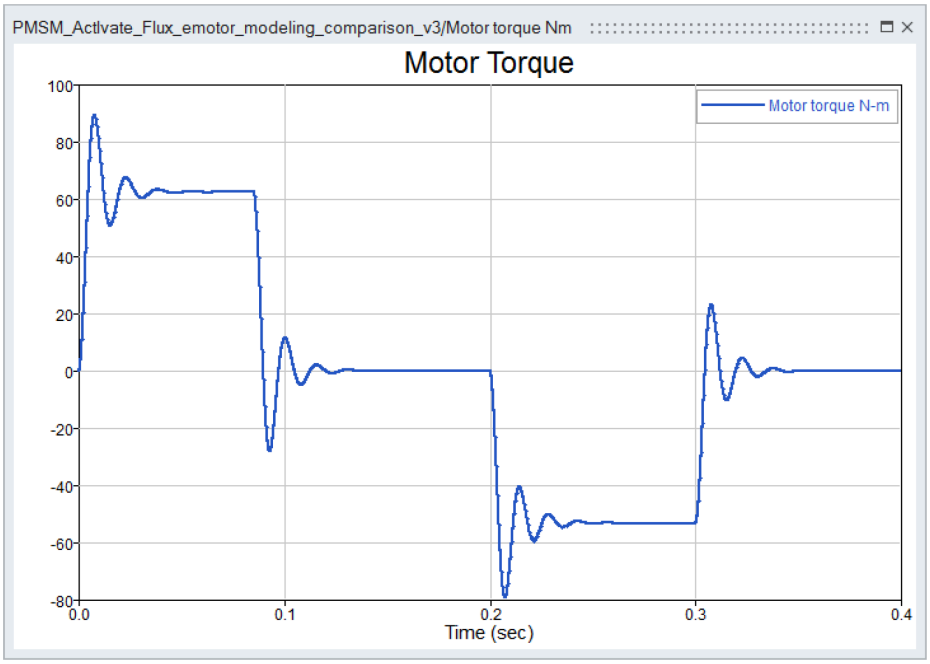


Figure 4: Torque VS time

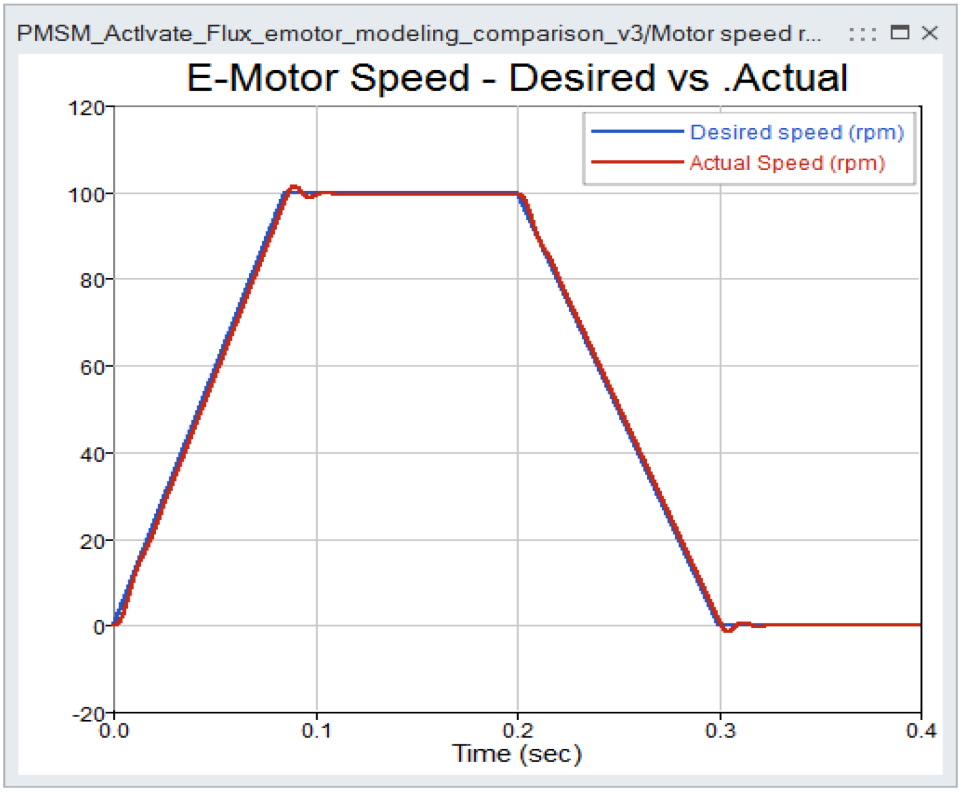


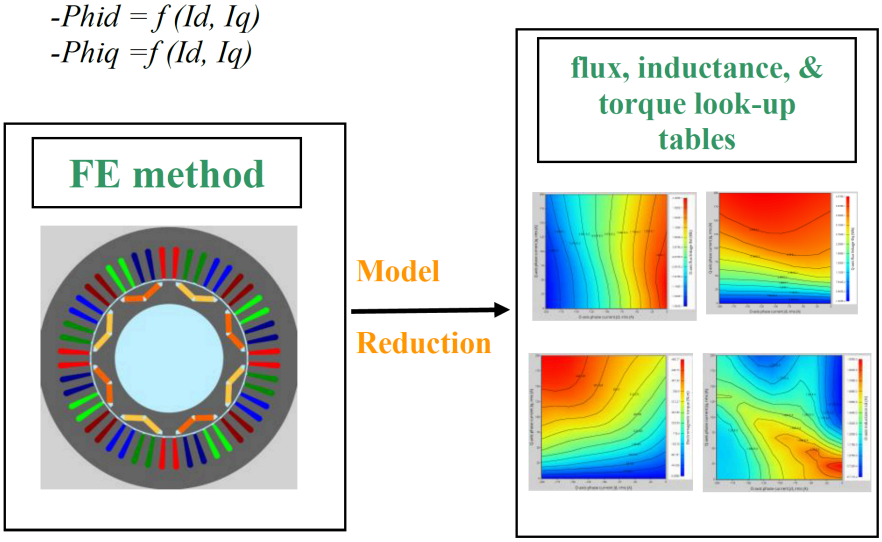
Figure 5: Speed VS time

Example 2: Reduced-order model, type 1: Look-up table models extracted from finite element model (FluxMotor)

The aim of this method is to build an accurate reduced model (based on Finite Element model) of the IPM motor: accuracy and quick simulation with Activate are the biggest advantages of the methodology. The electrical machine behavior is represented by look-up tables for flux linkage and inductance in the direct/quadrature frame, and torque (all which are dependent on current in dq axis Id, Iq) and are calculated with finite element method in FluxMotor and post-processed with a Compose script to generate the static inductances, Ld, Lq. The method in FluxMotor is quasi-static and so does not capture the effects of transient behavior like eddy currents.

We calculate with FluxMotor the response surface of flux and torque with finite element method – this is accomplished using the options in **Test** > **Characterization** > **Model**, and selecting appropriate options to export the desired results (see FluxMotor documentation for more details: http://fluxmotordoc.altair.com/helponlines/search). Note that the inductances Ld, and Lq from FluxMotor from this test are dynamic – they show the variation in flux linkage per change in current. However, the Activate LuT model uses a "static" inductance, defined as the total flux linkage divided by the total current. In order to compute the static Ld and Lq, the Compose script `compute_emotor_LuT_data_from_FluxMotor.oml` may be used to help accomplish this step. Please see the documentation in FluxMotor for more details. The surfaces will be used as look-up tables in Activate.

-Torque= f (Id, Iq)
-Ld (static)= f (Id, Iq)
-Lq (static)= f (Id, Iq)
-Phid = f (Id, Iq)
-Phiq =f (Id, Iq)



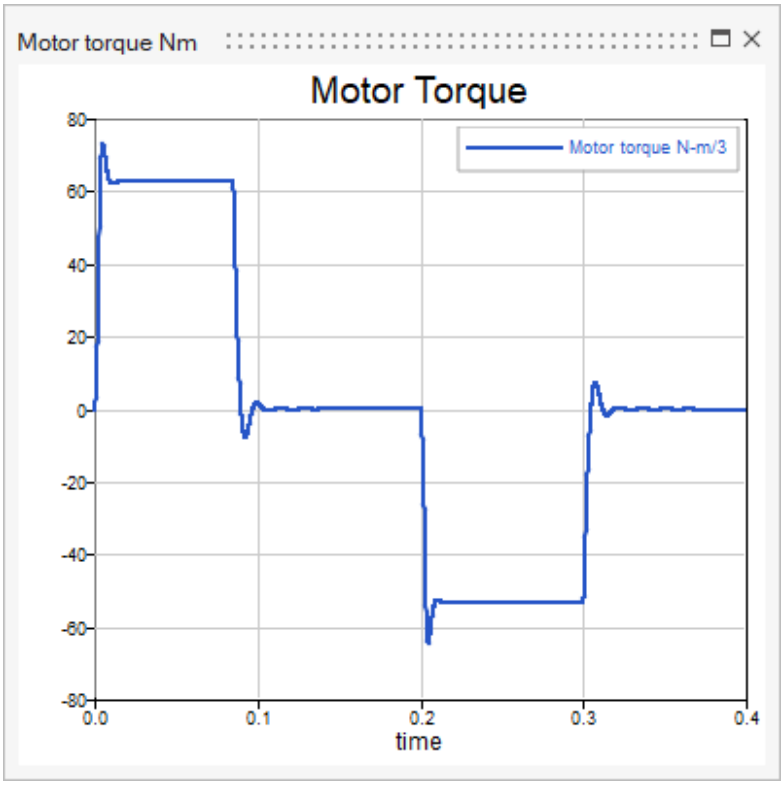


Figure 6: Torque VS time

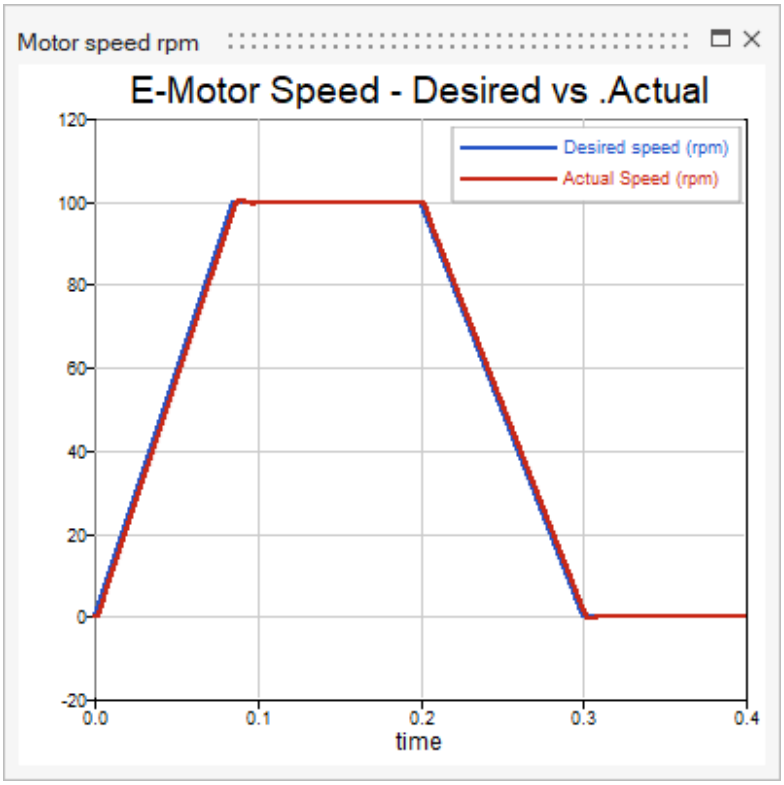


Figure 7: Speed VS time

Example 3: Reduced-order model, type 2: Look-up table models extracted from finite element model (Flux2D)

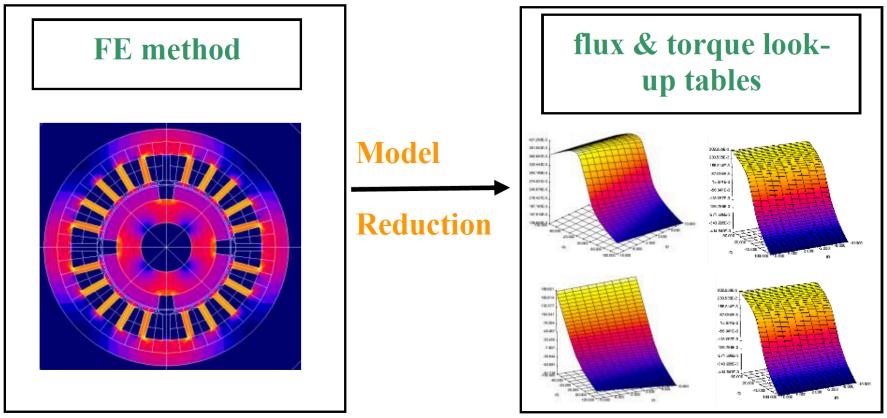
The aim of this method is to build an accurate reduced model (based on Finite Element model) of the IPM motor: accuracy and quick simulation with Activate are the biggest advantages of the methodology. The electrical machine behavior is represented by look-up tables for flux linkage for each phase and torque (dependent on current in dq axis Id, Iq, and rotor angle, θ) which are calculated with finite element method. By using Altair Flux, we can add rotor-angle dependency to capture more accurate torque ripple compared to the method with Altair FluxMotor which does not have the dependency. Like FluxMotor, the method is quasi-static and so does not capture the effects of transient behavior like eddy currents.

In a first time we calculate with Altair Flux software the response surface of flux linkage and torque with finite element method. In a first approximation, the variation parameters are the direct, quadrature current and position. These surfaces will be used as look-up tables in Activate.

- Torque= f (Id, Iq, θ)
- Flux_A= f (Id, Iq, θ)
- Flux_B= f (Id, Iq, θ)
- Flux_C= f (Id, Iq, θ)

To generate these tables, the user needs to use the macro: **CreateLookUpTableFromTMProjectDQLight.PFM**. This macro intends to create look up table of flux abc and torque versus Id, Iq and rotor position. The attached document `Create Lookup Table Case 3.pdf` in the folder case3 gives more details on how to use the macro to generate the lookup tables.

The LuT models are both relatively fast compared to co-simulation and have improved accuracy compared to the simplified analytical approach of the Park model. The co-simulation method improves on accuracy by capturing transient behavior (eddy currents) but at expense of simulation speed – this will be discussed next.



To use these tables, the user needs to run the script: `emotor_type_3_LuT_Prius_like_2004.oml`. This script generates a mat file that we used in Activate model.

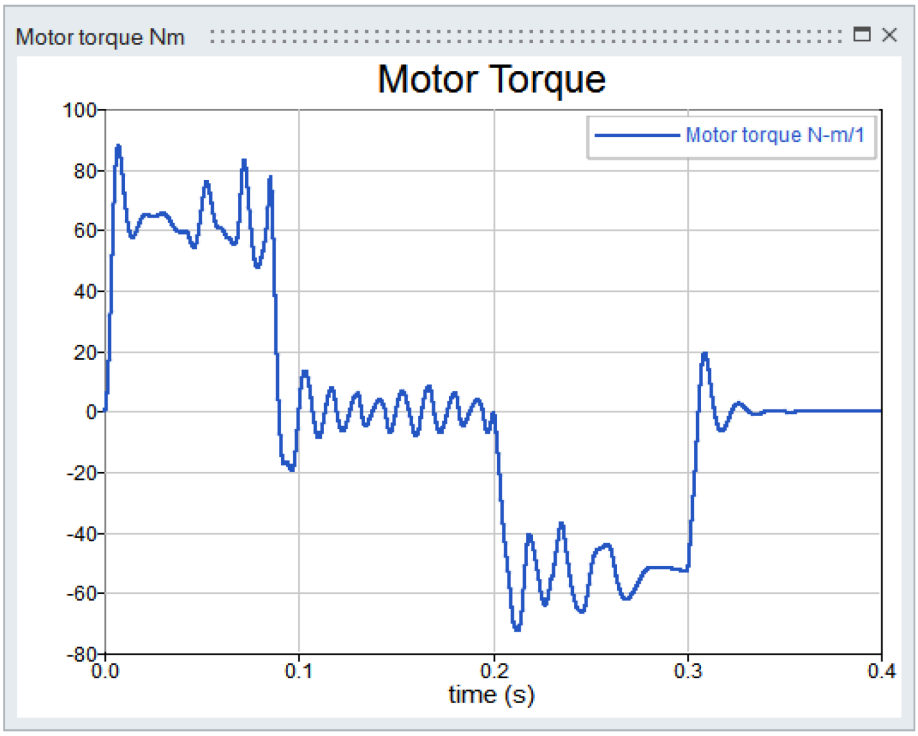


Figure 8: Torque VS time

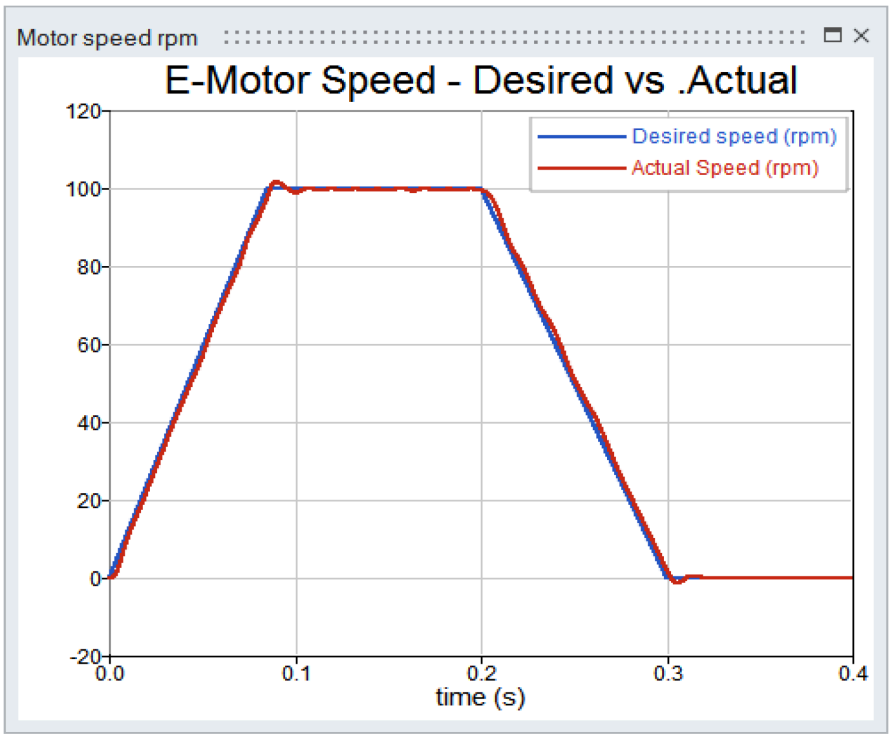


Figure 9: Speed VS time

Example 4: Co-simulation with multiphysics position kinematics

Altair Activate and Flux support coupling via co-simulation to run directly a transient (dynamic) analysis in Flux. This kind of simulation is much slower than the prior two methods but is more accurate as it takes into account eddy current in solid iron conductors. For co-simulation with Activate, the Flux kinematic set must be set to either "multiphysics position" or "coupled load" as the electric circuit is driven by voltage sources, the values of which determined by the drive system in the Activate model . The voltage sources induce a current in the electric circuit and create the torque which drives the mechanical behavior.

In addition, the value of the initial position of the rotor angle must be determined - this is used by the field-oriented controller to align the direct (d) axis to the phase A peak magnetic axis. To get the initial rotor position you can use the macro **Find_Rotor_Angle_2D.PFM**. This macro calculates the rotor angle for t=0s in order to align it with the magnetic induction generated in the airgap by a reference phase (e.g., phase A). To accomplish this goal, it demands as inputs the number of pole pairs of the device, the current source associated which each phase (the first one becomes the reference phase), the rotor mechanical set and the stator mechanical set. The output of this macro is a new variable called *POS_INI*. This parameter will be used in the mechanical set rotor.

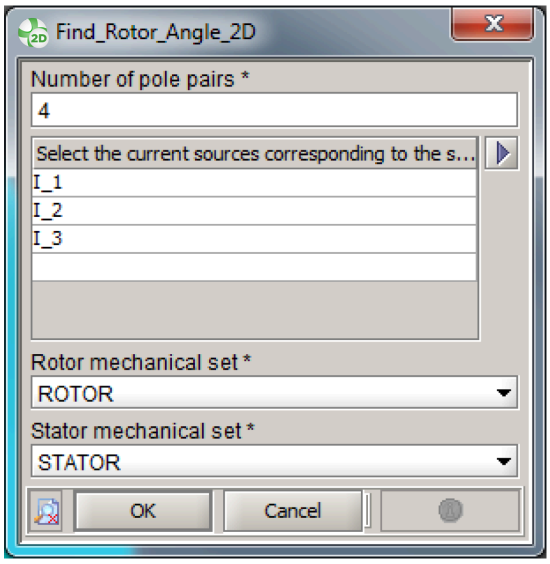
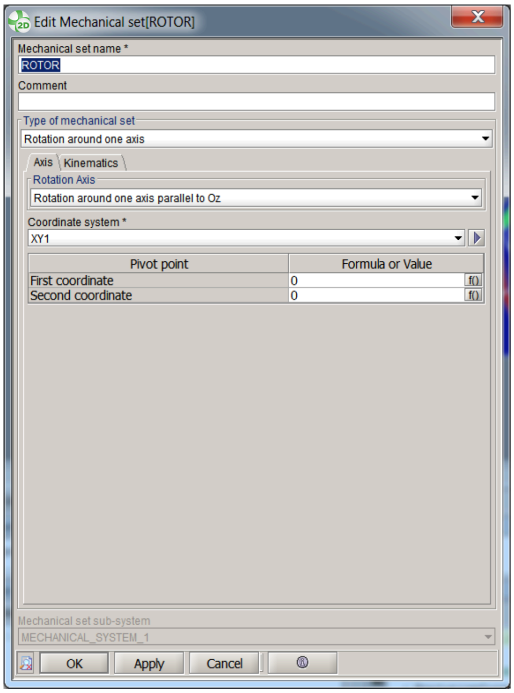
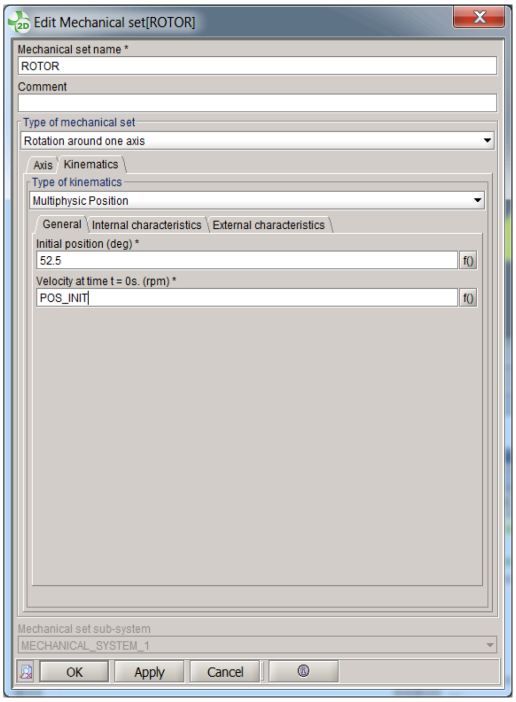


Figure 10: Interface of Find_Rotor_Angle_2D macro

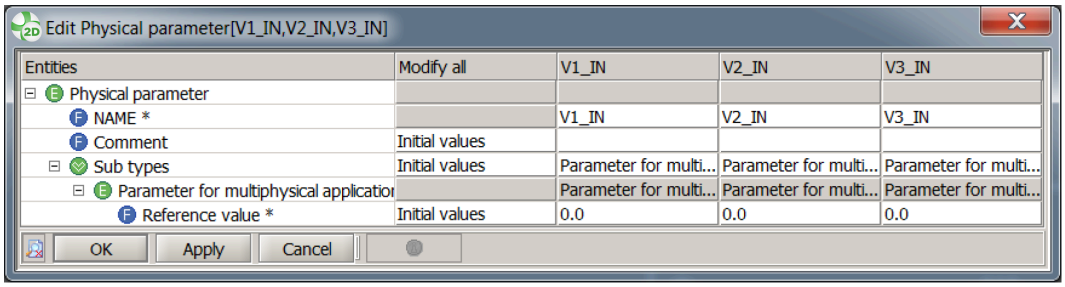
In this example, the type of kinematics in the mechanical set rotor is multiphysics position. In this case, the mechanical modeling is defined in the Activate model, and the Activate model provides the position of the moving part (rotor angle) to Flux as an input. This allows the potential to model the mechanic driveline in Activate and take advantage of modeling capabilities for more sophisticated systems.





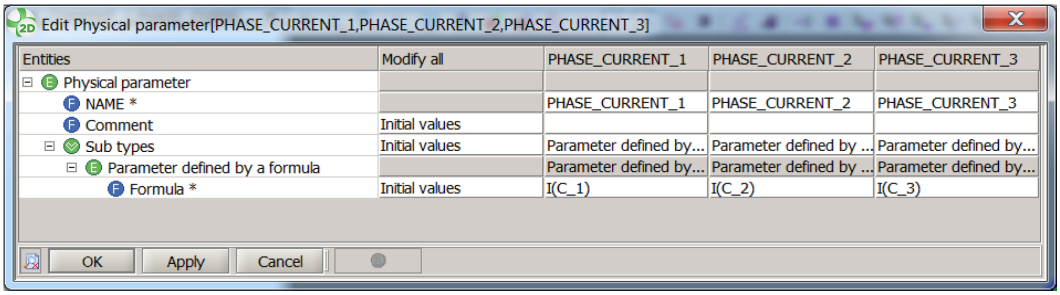
Coupling component:

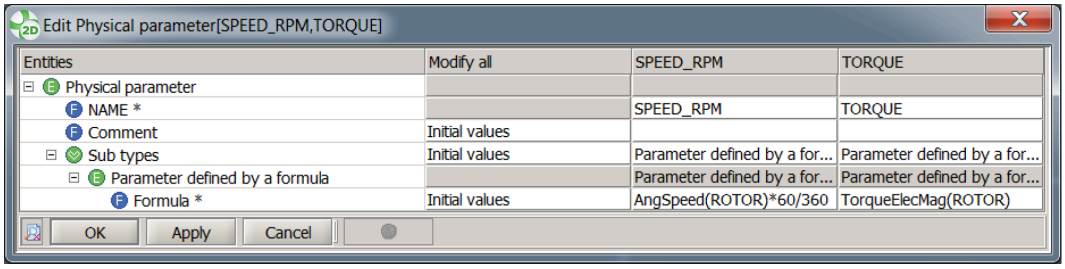
- Input parameters:



The type of these parameters is parameter for multiphysical application.

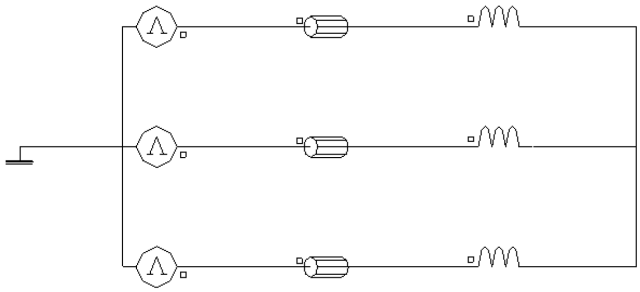
- Output parameters:





For the output parameters the type is parameter defined by formula.

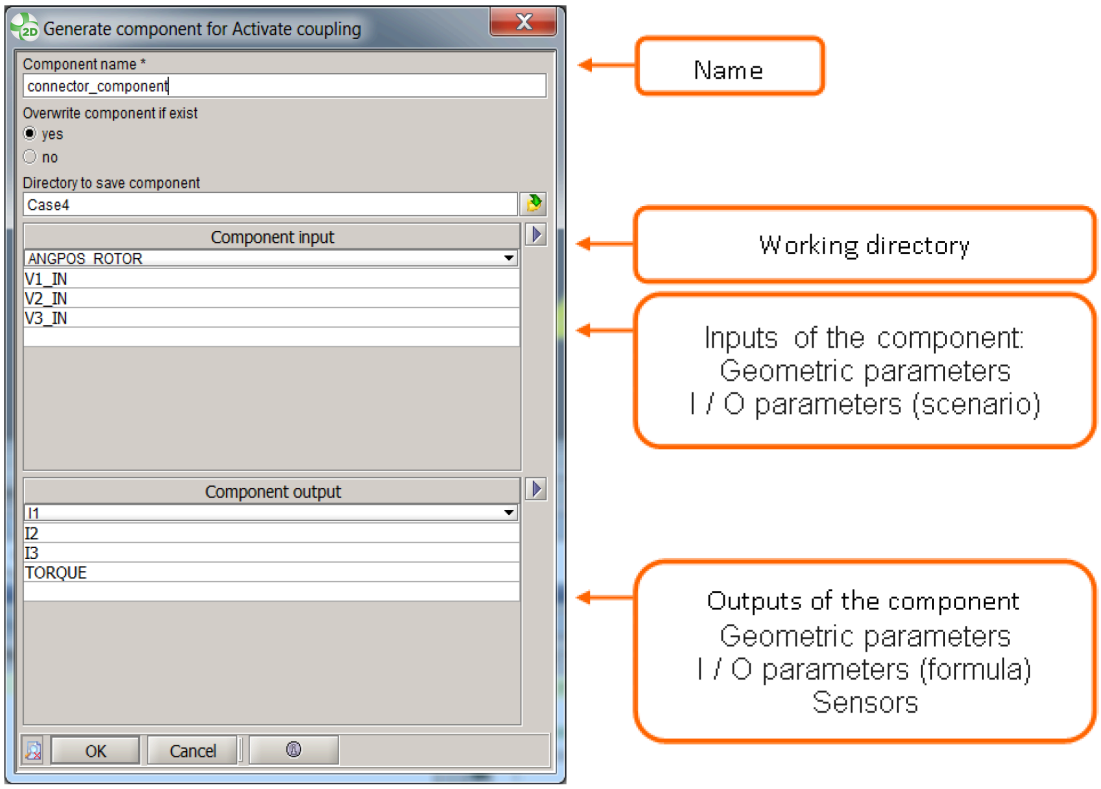
The electrical circuit is defined as:



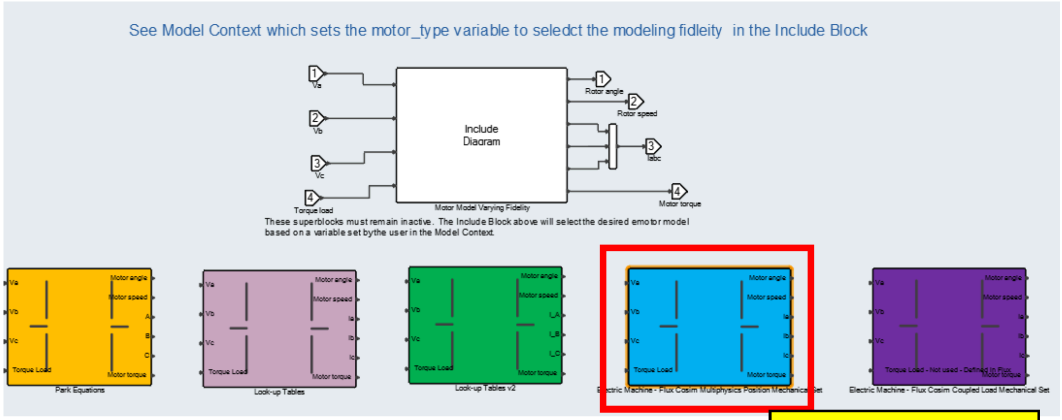
It is composed: Voltage source, Coil conductor and Inductance

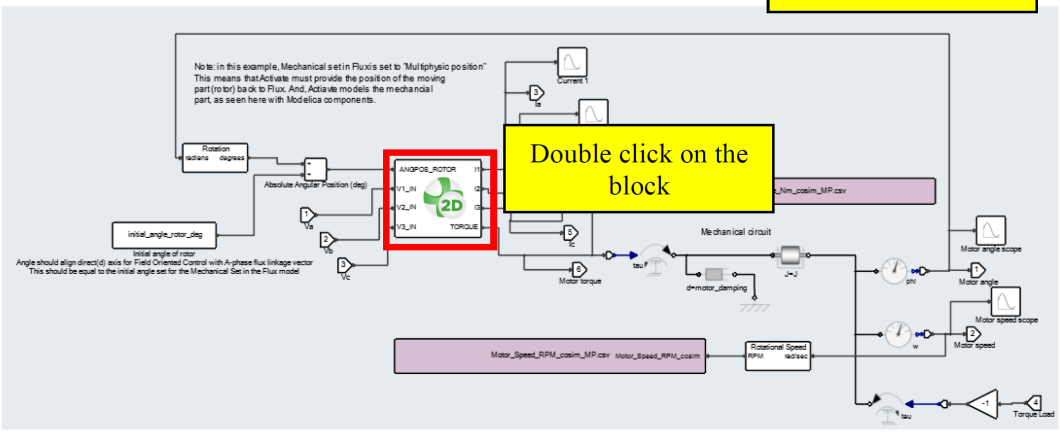
In the voltage sources we put the input parameter (V1_IN, V2_IN and V3_IN).

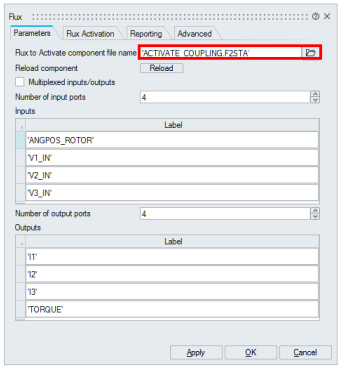
Generate component (**Solving** > **Generate component for Activate coupling**) for Activate coupling in the same working directory.



Once the coupling component is created, we load it in Activate.







Load the coupling component

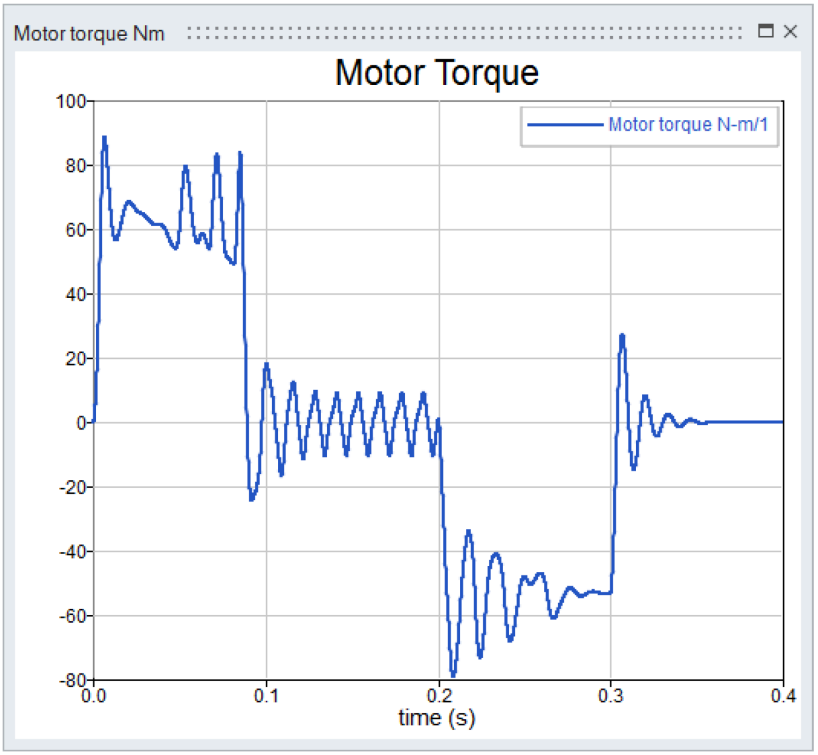


Figure 11: Torque VS time

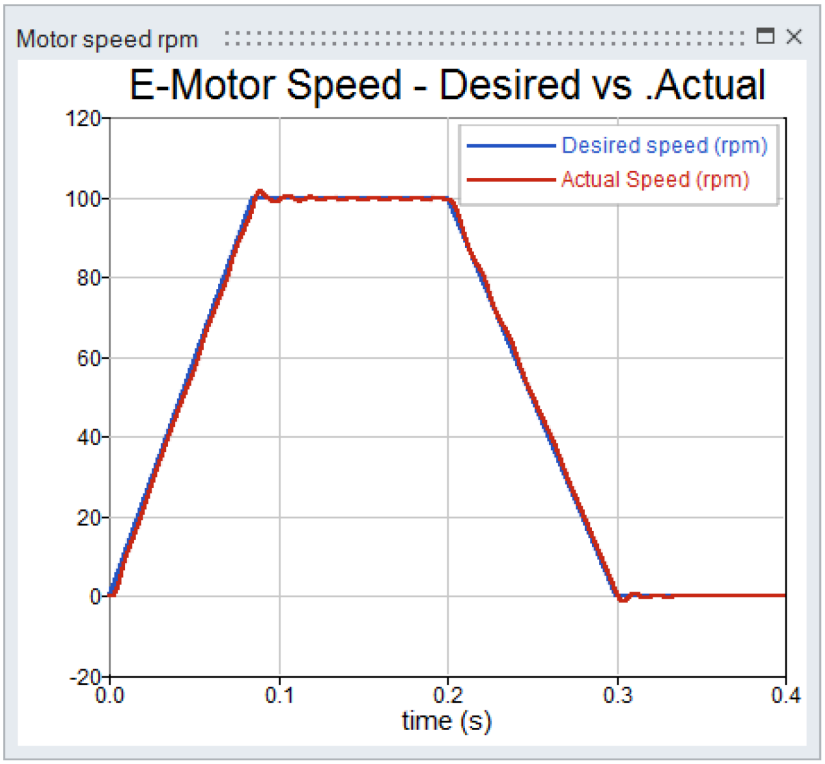


Figure 12: Speed VS time

Example 5: Co-simulation with coupled load kinematics

Activate and Flux support coupling via co-simulation to run directly a transient (dynamic) analysis in Flux. This kind of simulation is much slower than the prior two methods but is more accurate as it takes into account eddy current in solid iron conductors. For co-simulation with Activate, the Flux kinematic set must be set to either “multiphysics position” or “coupled load” as the electric circuit is driven by voltage sources, the values of which determined by the drive system in the Activate model.

In addition, the value of the initial position of the rotor angle must be determined - this is used by the field-oriented controller to align the direct (d) axis to the phase A peak magnetic axis. In addition, the value of the initial position of the rotor angle must be determined - this is used by the field-oriented controller to align the direct (d) axis to the phase A peak magnetic axis. To get the initial rotor position you can use the macro **Find_Rotor_Angle_2D.PFM**. This macro calculates the rotor angle for t=0s in order to align it with the magnetic induction generated in the airgap by a reference phase (e.g., phase A). To accomplish this goal, it demands as inputs the number of pole pairs of the device, the current source associated which each phase (the first one becomes the reference phase), the rotor mechanical set and the stator mechanical set. The output of this macro is a new variable called *POS_INI*. This parameter will be used in the mechanical set rotor.

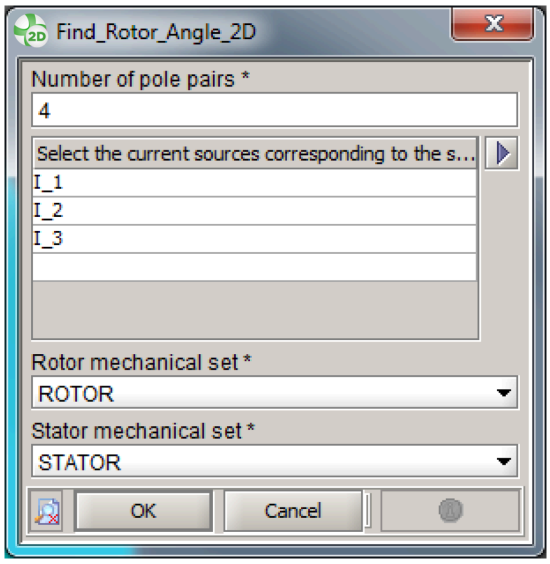
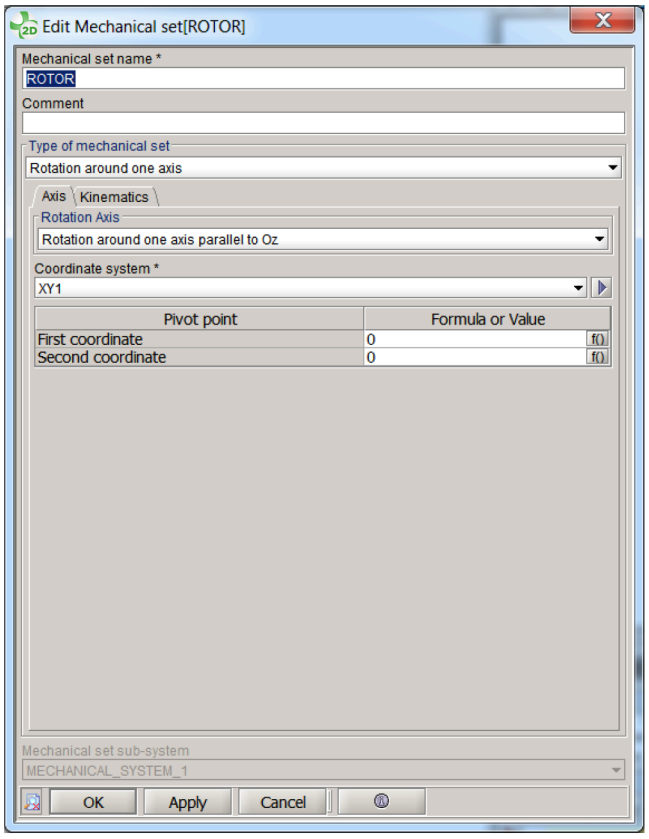
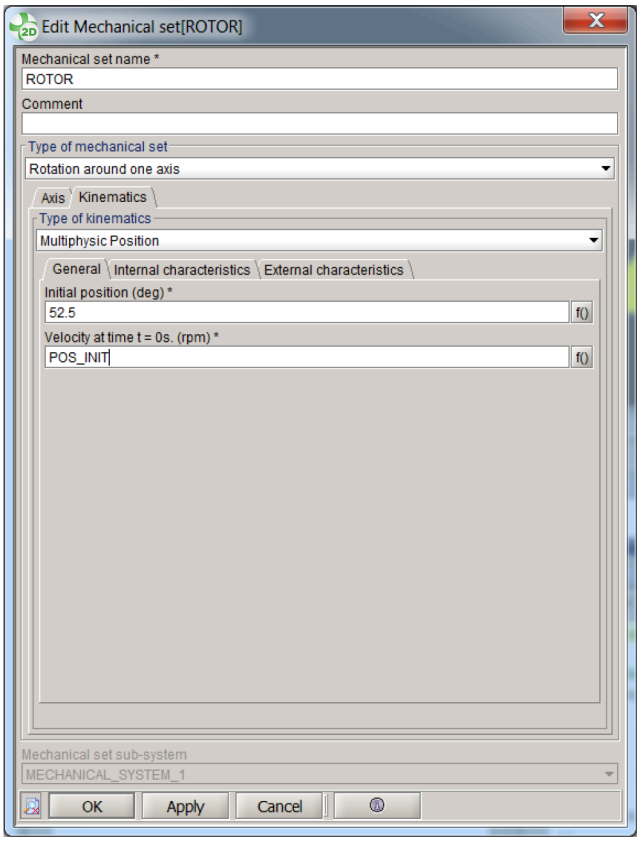


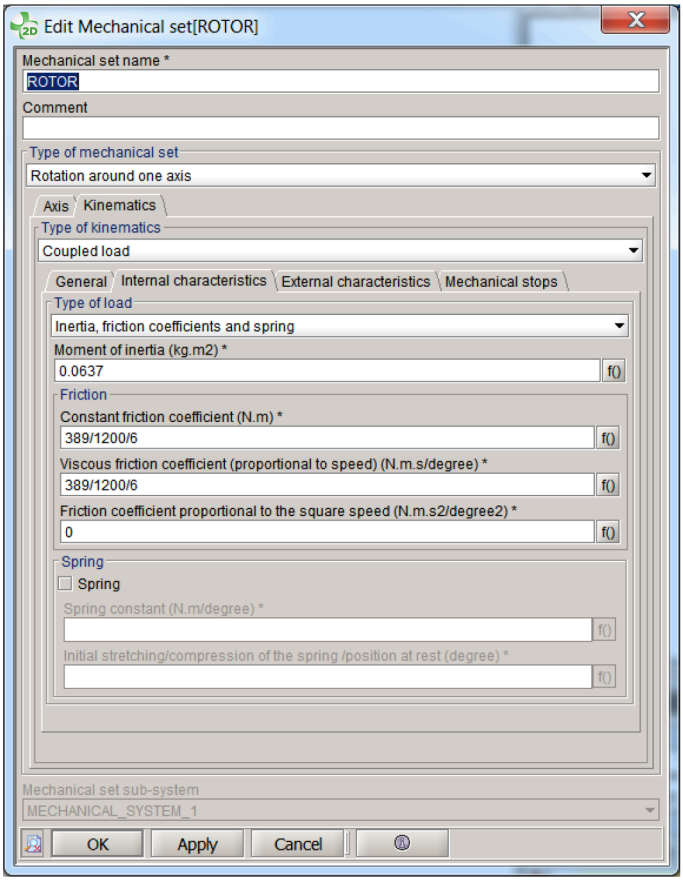
Figure 13: Interface of Find_Rotor_Angle_2D macro

In this case, the type of kinematics in the mechanical set rotor is coupled load. Here, the mechanical modeling is defined in the Flux model, rather than in the Activate model as in the prior example for kinematic set = multiphysics position.

From the example 4, we modify the mechanical set and create again a new coupling component.

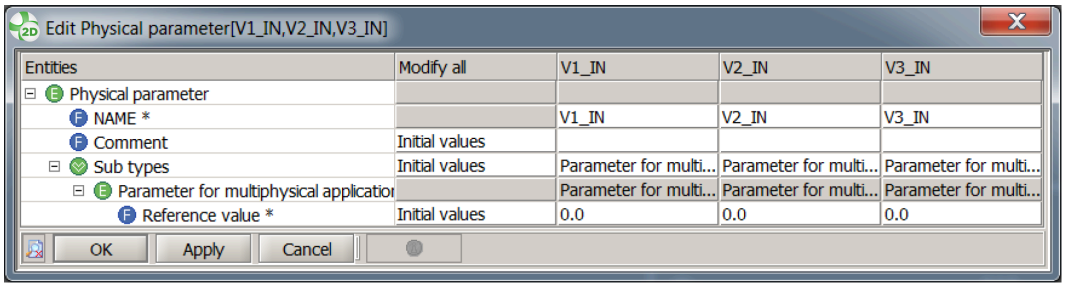

Edit Mechanical set[ROTOR]
Mechanical set name *
ROTOR
Comment
Type of mechanical set
Rotation around one axis
Axis
Kinematics
Rotation Axis
Rotation around one axis parallel to Oz
Coordinate system *
XY1
Pivot point
Formula or Value
First coordinate
0
Second coordinate
0
Mechanical set sub-system
MECHANICAL_SYSTEM_1
OK
Apply
Cancel


Edit Mechanical set[ROTOR]
Mechanical set name *
ROTOR
Comment
Type of mechanical set
Rotation around one axis
Axis
Kinematics
Type of kinematics
Multiphysic Position
General
Internal characteristics
External characteristics
Initial position (deg) *
52.5
Velocity at time t = 0s. (rpm) *
POS_INIT
Mechanical set sub-system
MECHANICAL_SYSTEM_1
OK
Apply
Cancel


Edit Mechanical set[ROTOR]
Mechanical set name *
ROTOR
Comment
Type of mechanical set
Rotation around one axis
Axis
Kinematics
Type of kinematics
Coupled load
General
Internal characteristics
External characteristics
Mechanical stops
Type of load
Inertia, friction coefficients and spring
Moment of inertia (kg.m2) *
0.0637
Friction
Constant friction coefficient (N.m) *
389/1200/6
Viscous friction coefficient (proportional to speed) (N.m.s/degree) *
389/1200/6
Friction coefficient proportional to the square speed (N.m.s2/degree2) *
0
Spring
Spring
Spring constant (N.m/degree) *
Initial stretching/compression of the spring /position at rest (degree) *
Mechanical set sub-system
MECHANICAL_SYSTEM_1
OK
Apply
Cancel

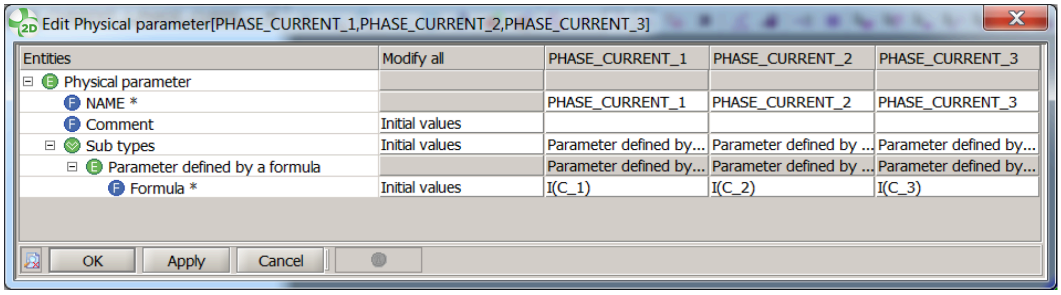
Coupling component:

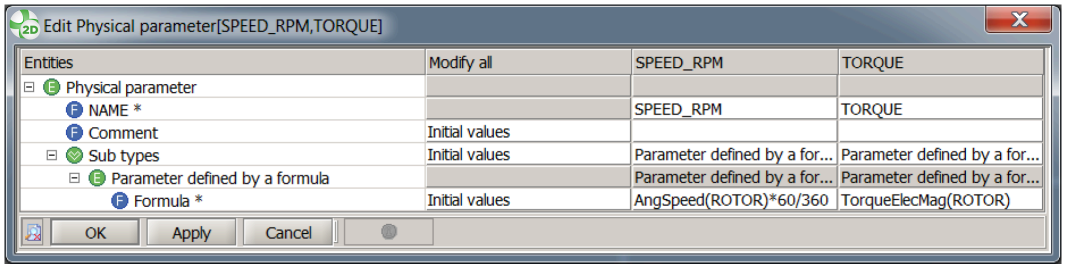
- Input parameters:



The type of these parameters is parameter for multiphysical application.

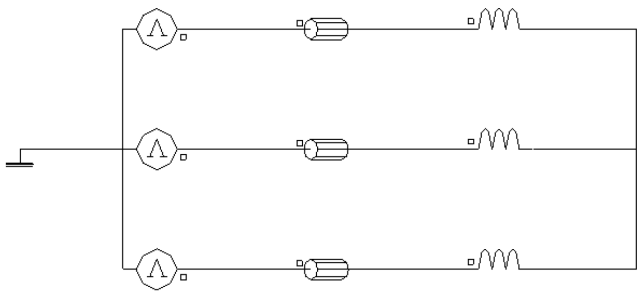
- Output parameters:





For the output parameters the type is parameter defined by formula.

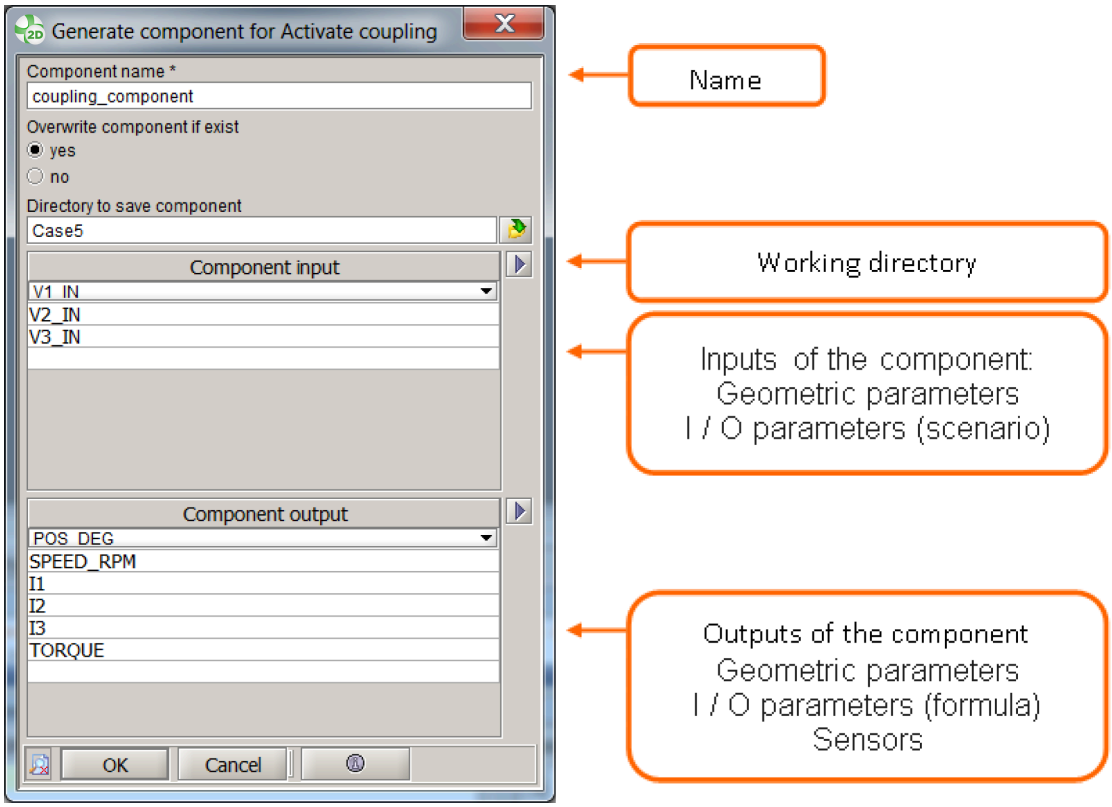
The electrical circuit is defined as:



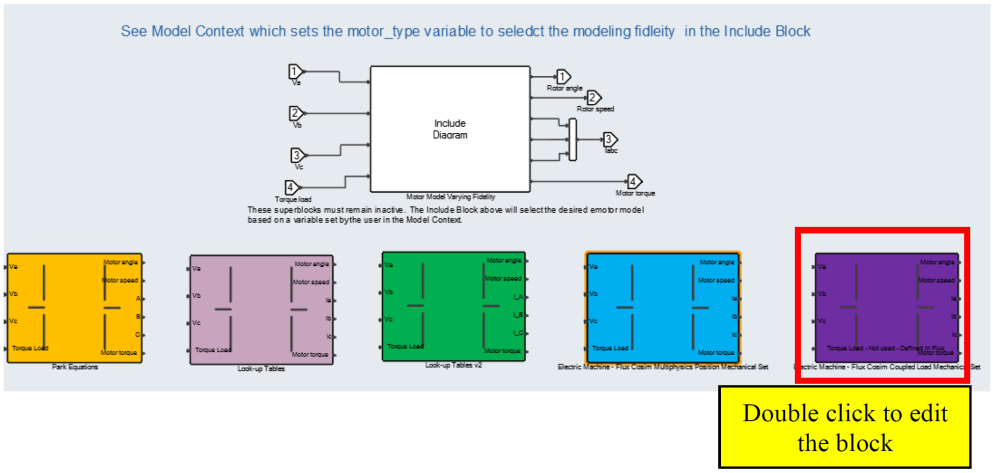
It is composed: Voltage source, Coil conductor and Inductance

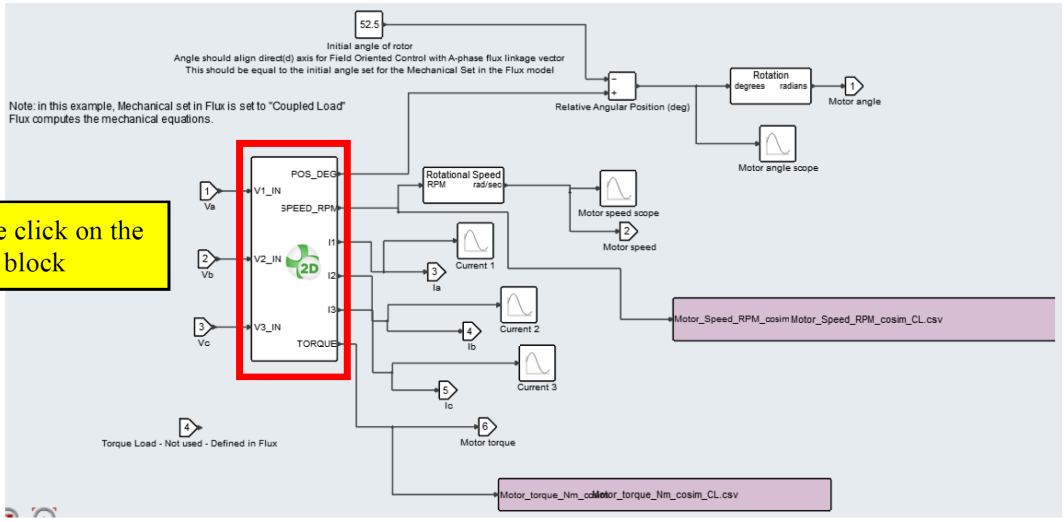
In the voltage sources we put the input parameter (V1_IN, V2_IN and V3_IN).

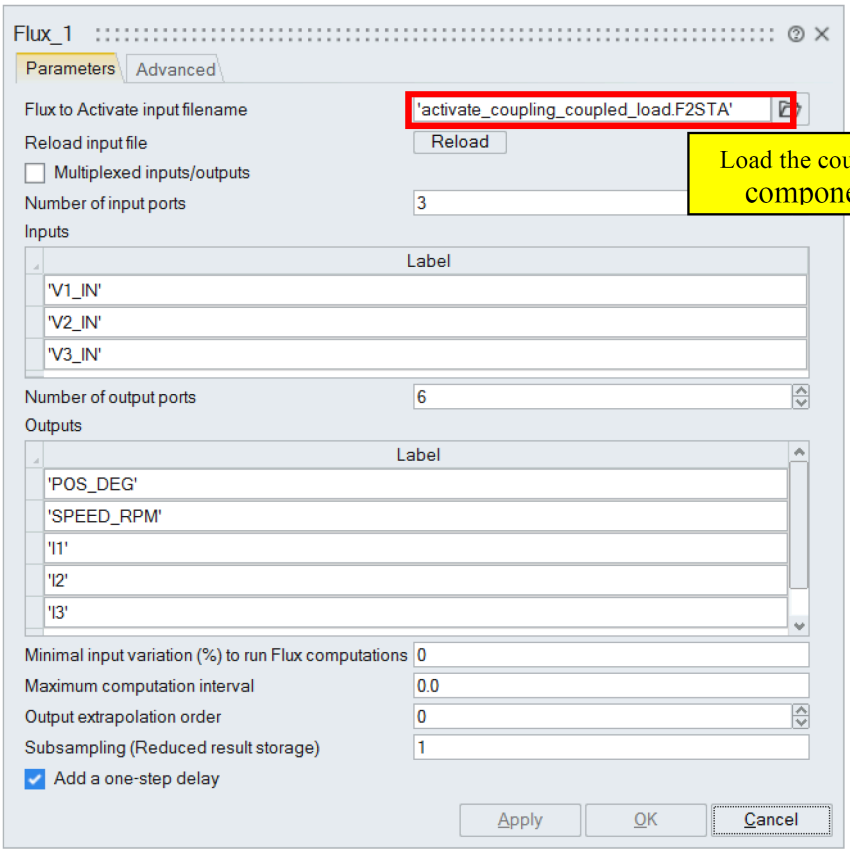
Generate component (**Solving** > **Generate component for Activate coupling**) for Activate coupling in the same working directory.



Once the coupling component is created, we load it in Activate.




52.5
Initial angle of rotor
Angle should align direct(d) axis for Field Oriented Control with A-phase flux linkage vector
This should be equal to the initial angle set for the Mechanical Set in the Flux model
Note: in this example, Mechanical set in Flux is set to "Coupled Load"
Flux computes the mechanical equations.
Relative Angular Position (deg)
Rotation
degrees radians
Motor angle
Motor angle scope
Double click on the block
POS_DEG
V1_IN
SPEED_RPM
V2_IN
V3_IN
TORQUE
Va
Vb
Vc
Rotational Speed
RPM rad/sec
Motor speed scope
Motor speed
Current 1
Ia
Current 2
Ib
Current 3
Ic
Motor_Speed_RPM_cosim
Motor_Speed_RPM_cosim_CL.csv
Torque Load - Not used - Defined in Flux
Motor torque
Motor_torque_Nm_cosim_CL.csv


Flux_1
Parameters
Advanced
Flux to Activate input filename
'activate_coupling_coupled_load.F2STA'
Reload input file
Reload
Load the coupling component
Multiplexed inputs/outputs
Number of input ports
3
Inputs
Label
'V1_IN'
'V2_IN'
'V3_IN'
Number of output ports
6
Outputs
Label
'POS_DEG'
'SPEED_RPM'
'I1'
'I2'
'I3'
Minimal input variation (%) to run Flux computations
0
Maximum computation interval
0.0
Output extrapolation order
0
Subsampling (Reduced result storage)
1
Add a one-step delay
Apply
OK
Cancel

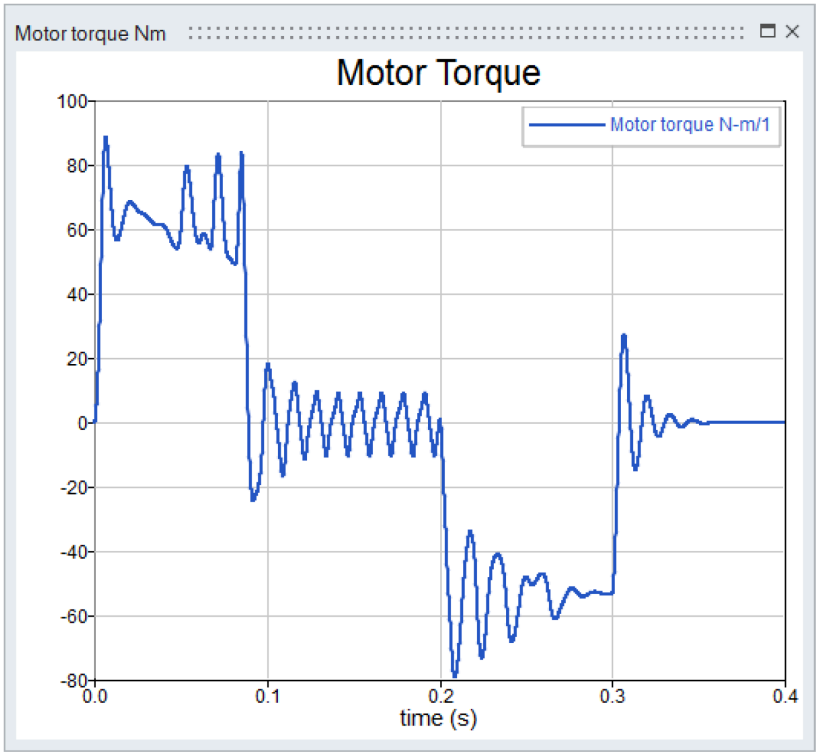


Figure 14: Torque VS time

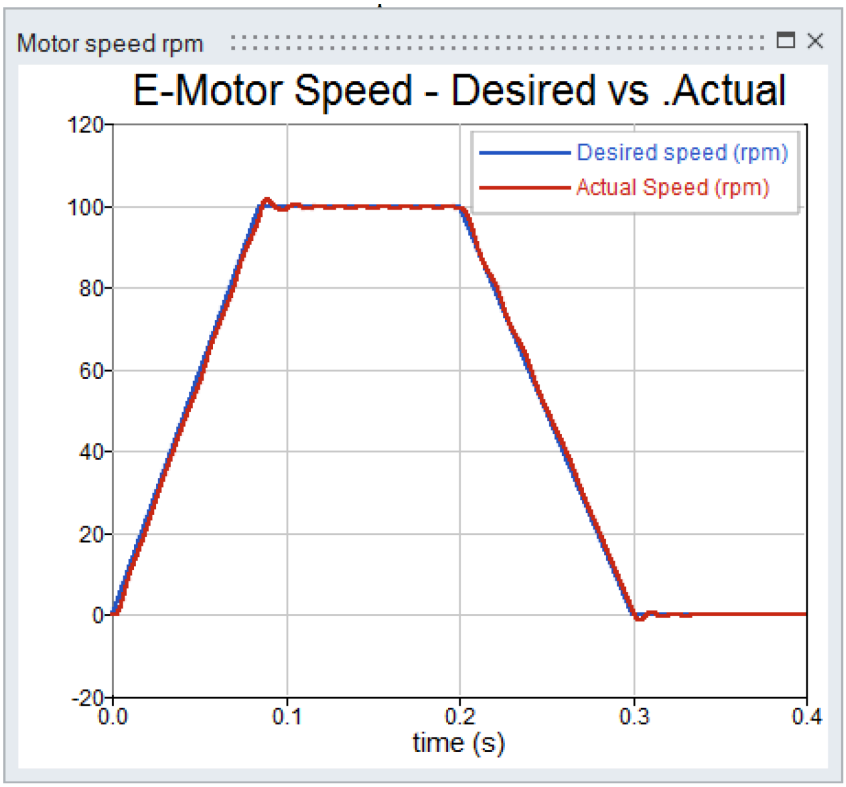


Figure 15: Speed VS time