



Sensorless Motor Commissioning

User Guide

Version	Date	Editor	Comment
001	2018-05-09	dg	initial draft
002	2019-08-06	dg	released version

Document AN133_SensorlessCommissioning_UserGuide_EP
Version 002
Source
Destination
Owner dg

Copyright © 2019
Triamec Motion AG
All rights reserved.

Triamec Motion AG
Industriestrasse 49
6300 Zug / Switzerland

Phone +41 41 747 4040
Email info@triamec.com
Web www.triamec.com

Disclaimer

This document is delivered subject to the following conditions and restrictions:

- This document contains proprietary information belonging to Triamec Motion AG. Such information is supplied solely for the purpose of assisting users of Triamec products.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Information in this document is subject to change without notice.

Table of Contents

1	Sensorless Mode.....	2	3.1	Current Controller.....	5
2	Preparation TAM Configuration.....	3	3.2	Position Controller.....	9
	2.1 Required Data:.....	3	4	Trouble-Shooting.....	11
	2.2 TAM Parameters.....	3	5	Appendix.....	13
3	Commissioning.....	5	5.1	Position Controller.....	13

1 Sensorless Mode

With the TSD servo-drives it is possible to run synchronous motors without a feedback of the rotor position. Instead of a position signal a model is used to calculate the rotor angle based on the applied voltage and the measured current. As the model requires a certain speed of the rotor to estimate the rotor angle, the motor is driven in sensorless mode at higher speed and in synchronous mode at lower speed.

In synchronous mode a constant current-phasor is rotated to drive the motor. When a certain speed-threshold is reached (*SensorlessTransitionStartVelocity*) the transition to sensorless-mode starts. The position of the motor and the commutation angle are now calculated based on the model, the position controller is activated and the current-phasor is reduced steadily until *SensorlessTransitionEndVelocity* is reached and the motor is controlled in sensorless mode (see Figure 1).

This document describes a procedure to commissioning a motor in sensorless mode. Depending on the characteristic of the motor, the sequence has to be adapted to get good results.

Chapter 2 describes the preparation of the TAM configuration. Chapter 3 describes the commissioning process in detail and how the main parameters can be derived.

Remark In this document it is assumed, that radian [rad] are used as unit for the position. If other units are used, the values must be converted accordingly.

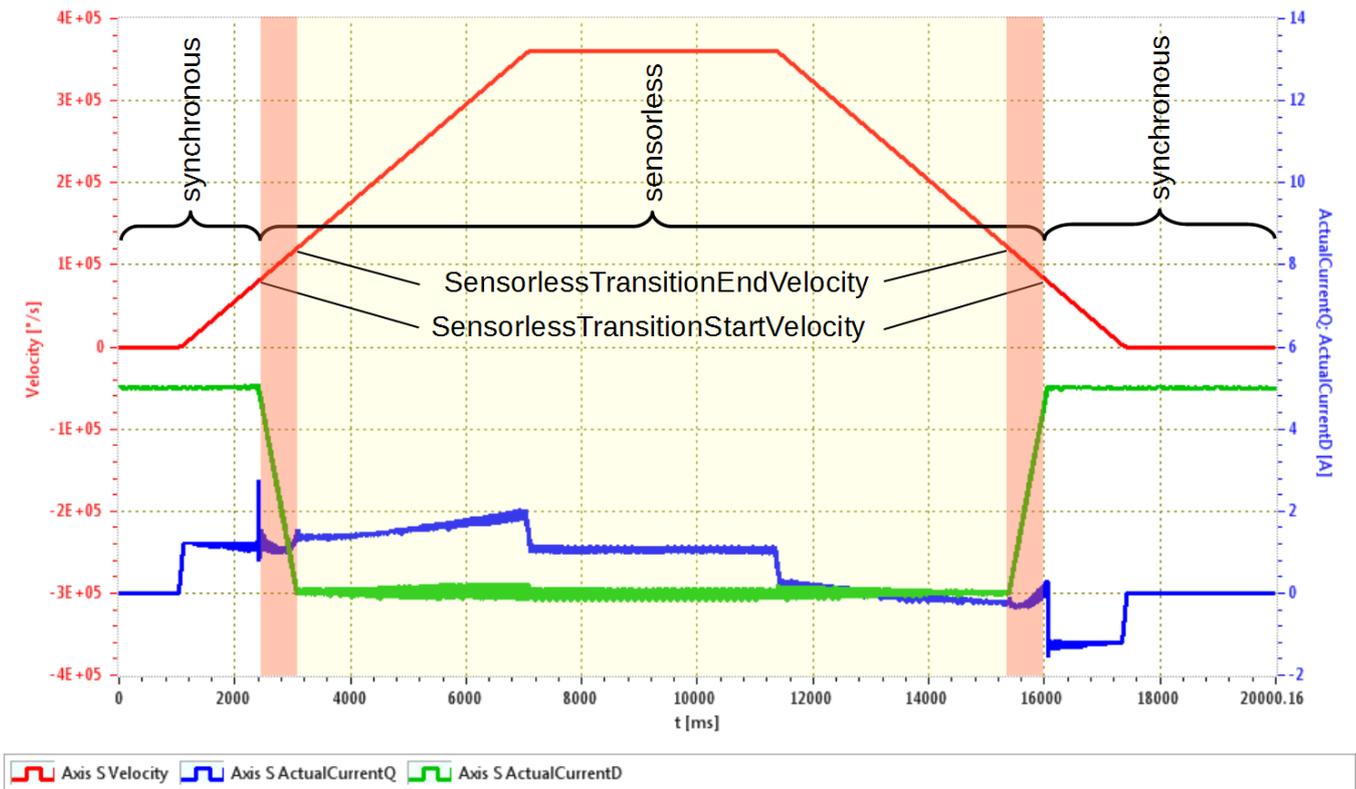


Figure 1: Full speed move with transition from synchronous operation to sensorless and back to synchronous.

2 Preparation TAM Configuration

This section describes the preparation of the TAM configuration to run a motor in sensorless-mode.

2.1 Required Data:

The following data are required for the commissioning of the motor (see data sheet of the motor):

- Peak current [A_{PK}]
- Nominal current [A_{RMS}]
- Thermal time constant [s]
- Pole pairs [] (if number of poles is given in the data-sheet: $PolePairs = NuberOfPoles/2$)
- Torque constant [Nm/ A_{PK}]
- Inertia [kgm^2]
- Resistance phase-phase [Ohm]
- Inductance phase-phase [H]
- Type of temperature sensor (if a temperature signal is available)
- Temperature limit (if a temperature signal is available)

Remark A current value given in A_{RMS} can be transformed to A_{PK} by multiplying with $\sqrt{2}$. If A_{RMS} is in the denominator (e.g. torque constant), the value has to be divided by $\sqrt{2}$.

To verify if the required dynamic for axis is reachable also the following specifications are required:

- Specified max velocity [rad/s]
- Specified max acceleration [rad/s²]
- Specified max jerk [rad/s³]

2.2 TAM Parameters

This section gives a recommendation for the setup of the TAM parameters. Depending on the characteristic of the spindle, modifications are required. Default values are used for parameters not mentioned in this section.

.../Parameter/Motor/

- *Type* = SynchronousAC
- *InvertDirection*: Set this value to 'True' if the motor turns in the wrong direction.
- *PolePairs*: Set this value according the data-sheet of the motor
- *EncoderCountsPerMotorRevolution* = 1000
- *PeakCurrent*: Set this value according the data-sheet of the motor in unit [A_{PK}].
- *NominalCurrent*: Set this value according the data-sheet of the motor in unit [A_{RMS}].

- *CurrentSquareTime*: Set this value according the data-sheet of the motor.
- *TemperatureSensorType*: Sensor type if sensor is available – else 'None'
- *TemperatureUpperLimit*: Set this value according the data-sheet of the motor.

Warning The limitation of the current (*NominalCurrent*, *PeakCurrent*, *CurrentSquareTime*) must be configured correctly before the drive is enabled the first time. Wrong values could cause damage of the motor.
It is recommended to use a temperature sensor to observe the motor temperature.

.../Parameters/PathPlanner

The internal path planner is used to generate moves with the axis module to optimize and test the commissioning. Set the dynamic limits according to the specifications of the axis. Consider that the reachable dynamic is also limited by the available bridge voltage and the restriction of the current (motor or drive) and the physical properties of the motor and the load.

ModuloPositionMaximum and *ModuloPositionMinimum*:

Set the modulo values if required. In normal case this is $n_E \cdot \text{EncoderPitch}$ whereas n_E is an integer and in most cases $n_E = \text{EncoderCountsPerRevolution}$. Use many decimal places to write the value. Make sure, that the modulo value does match the modulo value used by the higher level control system to generate the set-positions. Example:

- *ModuloPositionMaximum* = 6.2831853071795862
- *ModuloPositionMinimum* = 0.0

.../Parameter/PositionController

- *PositionUnit* = set the desired position unit e.g. [rad]
- *FeedForwardPosition* = 0
- *FeedForwardVelocity* = 0
- *FeedForwardAcceleration*: (see section 'Acceleration Feed Forward' on page 9)
- *OutputLimit* = $0.95 \cdot \text{PeakCurrent}$
- *Encoders[0].Type* = *Sensorless*
- *Encoders[0].Pitch* = set the encoder pitch. For example if *PositionUnit* is [rad] then $\text{Pitch} = 2\pi/n_E$ with $n_E = \dots/\text{Motor}/\text{EncoderCountsPerMotorRevolution}$ (position unit is [rad]). Use many decimal places to write the value.
- *Controller[0]*: For *Kp*, *Ki*, *Kd*, *T1* see section 3.2.
- *Controller[0]/PositionErrorLimit*: Set this value high enough to not cause errors in normal operation e.g. 20rad.
- *Controller[0]/IntegratorOutputLimit*: Rule of thumb: $0.5 \cdot \text{NominalCurrent}$. This value is just considered if the integrator part is activated ($Ki \neq 0$).

.../Parameters/Commutation

- *PhasingMethod* = *RotorAlignmentThenEncoder*
- *EnablingMethod* = *ForcePhasing*
- *StartTime* = *0s*
- *RampRiseTime* = *0.5s*
- *RampConstTime* = *1s*
- *CurrentAmplitude* = $0.75 \cdot \text{NominalCurrent}$. This is also the current used to run the spindle in synchronous mode.
- *SineAmplitude* = *0.1rad*
- *SineFrequency* = *10Hz*

.../Parameters/CurrentController

- *K_r*, *T_n*, *R*, *L*, *SensorlessTransitionStartVelocity*, *SensorlessTransitionEndVelocity* : See section 3.1
- *PwmFrequency* = *pwm100kHz*
- *CurrentErrorLimit* = $2 \cdot \text{PeakCurrent}$
- *IntegratorOutputLimit* = $0.5 \cdot \text{BridgeVoltage}$
- *ModulationMethod* = *SinusoidalPwm*
- *K_t* = *0*
- *OutputLimit* = $0.57 \cdot \text{BridgeVoltage}$

.../General/Parameters

- *DcBusVoltageUpperLimit*: Upper limit of DC bus voltage
- *DcBusVoltageLowerLimit*: Lower limit of DC bus voltage

3 Commissioning

The following section describes one possible procedure to commission a sensorless motor. Depending on the characteristic of the motor, the sequence has to be adapted to get good results.

3.1 Current Controller

Controller Parameters K_R and T_n

The tuning of the current controller with the parameters K_R and T_n can be done with the bode tool based on a bode measurement. The current controller should be stiff, with high bandwidth but without overshoot. See figure 2 and 3 for an example of the bode- and nyquist-plot.

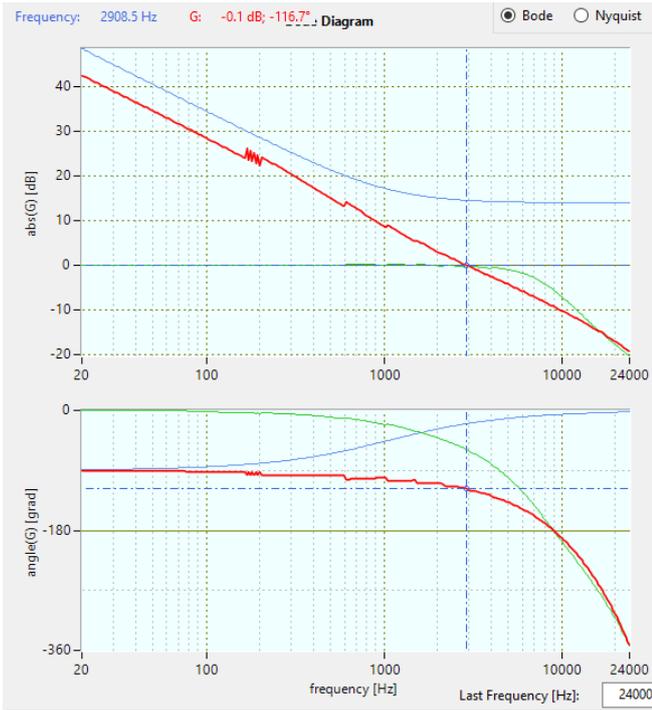


Figure 2: Bode-plot of current controller.

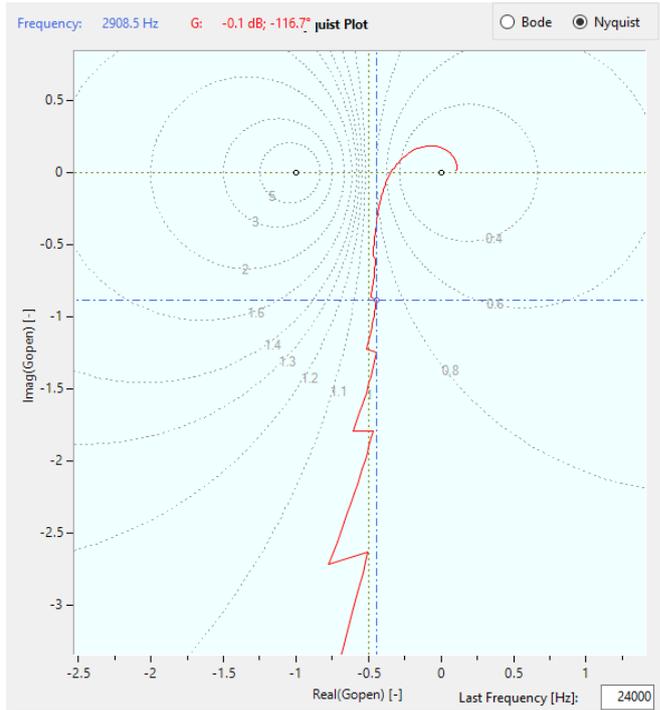


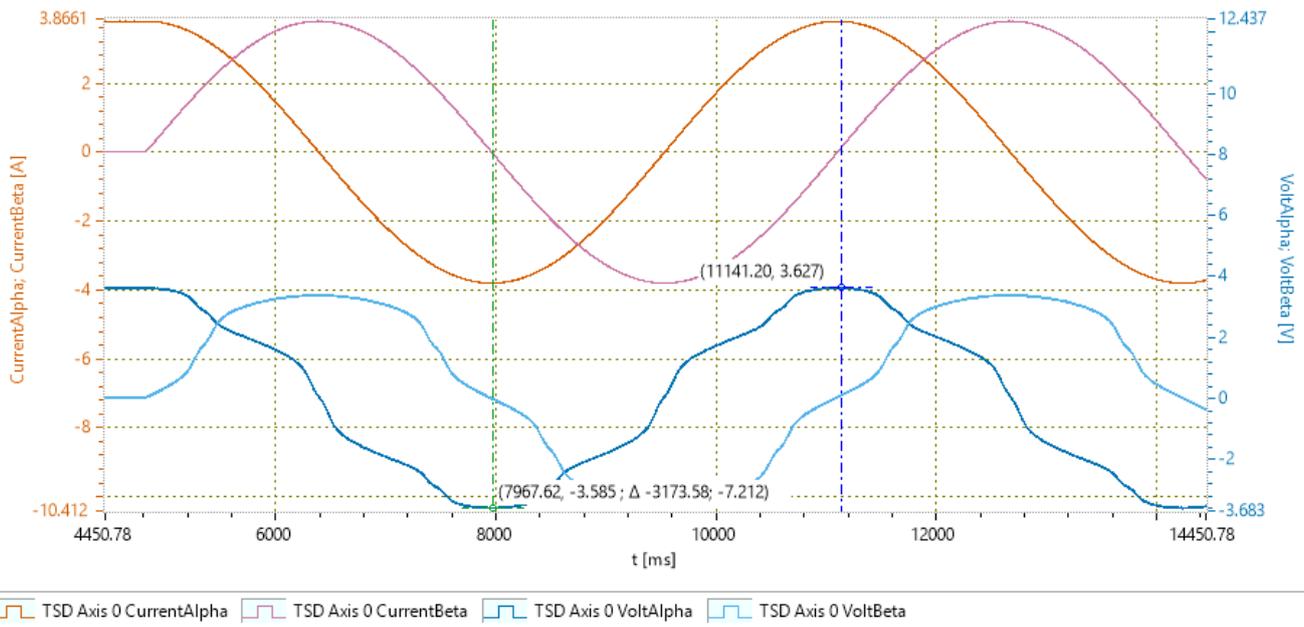
Figure 3: Nyquist-plot of current controller

When then the tuning of the current controller is done and the parameters are set as described in section 2.2 it should be possible to enable and to turn the axis in synchronous mode which means at speeds below `.../Parameters/CurrentController/SensorlessTransitionStartVelocity` and `.../Parameters/CurrentController/SensorlessTransitionStartVelocity` by using the Axis Module in TAM System Explorer.

Resistance R

As initial values the phase to phase resistance from the data sheet can be used. An other option to measure or verify the resistance R is by running the spindle at very low velocity in synchronous mode and measure the ratio between voltage and current:

1. Prepare the scope with following signals and set sampling time to 0.02ms:
 - `.../Signals/CurrentController/Sensorless/CurrentAlpha`
 - `.../Signals/CurrentController/Sensorless/CurrentBeta`
 - `.../SignalsCurrentController/Sensorless/VoltAlpha`
 - `.../SignalsCurrentController/Sensorless/VoltBeta`
2. Run the spindle with low velocity in synchronous mode e.g. with 1rad/s and measure the signals for at least one electrical turn.
3. Measure the relation between *CurrentAlpha* to *VoltageAlpha* and *CurrentBeta* to *VoltageBeta* to determine the resistance (see figure 4). And write the result to register `.../Parameters/CurrentController/R`.



Printed on 5/18/2018 12:39:20 PM

Figure 4: Measurement used to evaluate the resistance. $R = 3.6V / 3.8A = 0.85\Omega$.

Remark To scope signals from register `../Signals/CurrentController/Sensorless/` a sampling rate of 50 kHz or 100kHz is required. The values will be displayed as zero in scope if the *Sampling Time* is 10kHz or smaller.

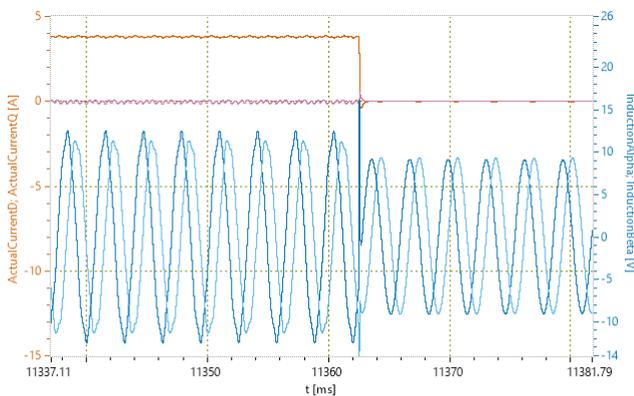
Inductance L

The parameter `../Parameters/CurrentController/L` can be determined with the following measurement:

1. Prepare the scope with following signals and set sampling time to 0.02ms:
 - `../Signals/CurrentController/Sensorless/InductionAlpha`
 - `../Signals/CurrentController/Sensorless/InductionBeta`
 - `../Signals/CurrentController/ActualCurrentQ`
 - `../Signals/CurrentController/ActualCurrentD`
2. Run the spindle at a certain speed v_m (e.g. $v_m = 0.5 v_{Max}$) in synchronous mode. This probably requires to temporary set the following parameters to a higher value than v_m :
 - `../Parameters/CurrentController/SensorlessTransitionStartVelocity`
 - `../Parameters/CurrentController/SensorlessTransitionEndVelocity`
3. Start the scope.
4. In this step, the parameters are changed in such a way that the spindle is in free-run. To achieve this, the value of `../Parameters/Commutation/CurrentAmplitude` has to be set to zero temporarily. This causes the controller to regulate the current to zero.

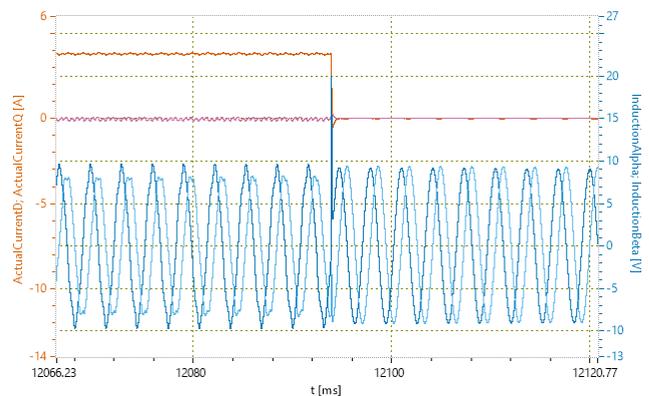
5. Now the values of *InductionAlpha* and *InductionBeta* can be compared at the transition from synchronous mode to free-run. If the parameter *L* is set correctly, the amplitudes and phases of both signals should not change at the transition. If the amplitude declines in free-run, *L* has to be increased and vice versa.
6. Repeat from step 2 until the condition at 5 is fulfilled and the optimal value L_{opt} is found. Figure 5 and 6 show the transition without and with optimization of *L*. Don't forget to reset *.../Parameters/Commutation/CurrentAmplitude* to the initial value and to stop the running move.
7. As the model behaves robust also if *L* is set smaller than the optimal value L_{opt} but gets unstable if *L* is chosen slightly bigger than L_{opt} , the value of *.../Parameters/CurrentController/L* should be set to a smaller value than L_{opt} e.g. $L = (0.0...0.75)L_{opt}$.

Remark In some cases most robust results were achieved, if the inductance is set to zero: $L = 0.0H$



Printed on 5/18/2018 2:00:37 PM

Figure 5: Estimated voltage before optimization of induction L ($L=0$).



Printed on 5/18/2018 2:04:01 PM

Figure 6: Estimated voltage after optimization of induction L

Transition Velocities

The transition velocities

- *.../Parameters/CurrentController/SensorlessTransitionStartVelocity*
- *.../Parameters/CurrentController/SensorlessTransitionEndVelocity*

define the transition range between synchronous mode and sensorless mode. The bandwidth of the output filter (see section 'Controller Output Filter' on page 9) should be lower than the first harmonic oscillation of the induction model. Therefore the parameters should be set according the following rule of thumb:

$$v_{SLStart} \geq \frac{4\pi f_{OPF}}{p}; \quad v_{SLEnd} = 1.1 v_{SLStart} \quad (1)$$

whereas

- $v_{SLStart}$ is *SensorlessTransitionStartVelocity* [rad/s],

- v_{SL_End} is *SensorlessTransitionEndVelocity* [rad/s],
- p is the number of pole pairs (*Parameters/Motor/PolePairs*),
- f_{0PF} is the bandwidth of the position controller output filter (recommend: 100Hz).

For example with $f_{0PF} = 100\text{Hz}$ and $p = 1$ the velocities are set to

- *.../Parameters/CurrentController/SensorlessTransitionStartVelocity* = 1250rad/s
- *.../Parameters/CurrentController/SensorlessTransitionEndVelocity* = 1380rad/s

3.2 Position Controller

Initial Setup

As bode tuning is not available for the position controller, we derive an initial guess for the controller parameters based on a physical model. For this, the inertia of the rotor J and the torque constant K_t must be known.

$$K_p = (2\pi f_{0PC})^2 \frac{J}{K_t}; \quad K_i = 0; \quad K_D = \frac{4 D \pi f_{0PC} J}{K_t}; \quad T_1 = \frac{D}{10\pi f_{0PC}}. \quad (2)$$

As a first estimate the following values for the bandwidth and the damping are recommended: $D = 0.7$ and $f_{0PC} = 10\text{Hz}$:

$$K_p \approx 4000 \frac{1}{s^2} \frac{J}{K_t}; \quad K_i = 0; \quad K_D \approx 90 \frac{1}{s} \frac{J}{K_t}; \quad T_1 = 0.002s \quad (3)$$

Controller Output Filter

To avoid that the position controller disturbs the induction model, a low pass filter of second order is applied to the controller output with following bandwidth and damping:

$$f_{0PF} \approx 10 f_{0PC} \approx 100\text{Hz}; \quad D = 0.7; \quad (4)$$

Consider also the dependency with the *SensorlessTransitionStartVelocity* (see equation 1).

The configuration of the filter is done by setting the following parameters:

- *.../PositionController/Controllers[0]/Filters[0]/Type* = Lowpass2
- *.../PositionController/Controllers[0]/Filters[0]/EdgeFrequencyDenominator* = 100 Hz
- *.../PositionController/Controllers[0]/Filters[0]/DampingDenominator* = 0.7

Acceleration Feed Forward

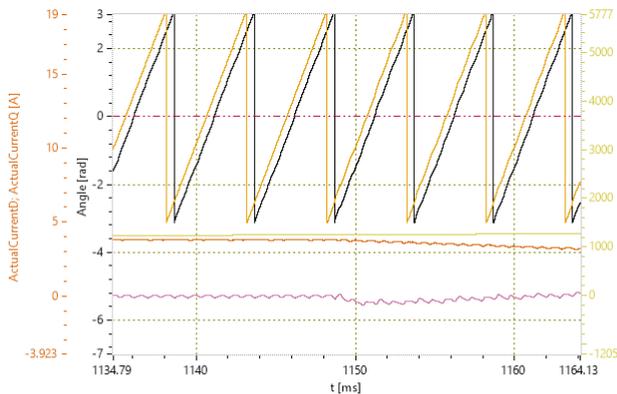
As an initial guess, the acceleration feed forward a_{ff} (*Parameters/PositionController/FeedForwardAcceleration*) can be calculated based on the inertia J and the torque constant K_t of the motor:

$$a_{ff} = \frac{J}{K_t} \quad (5)$$

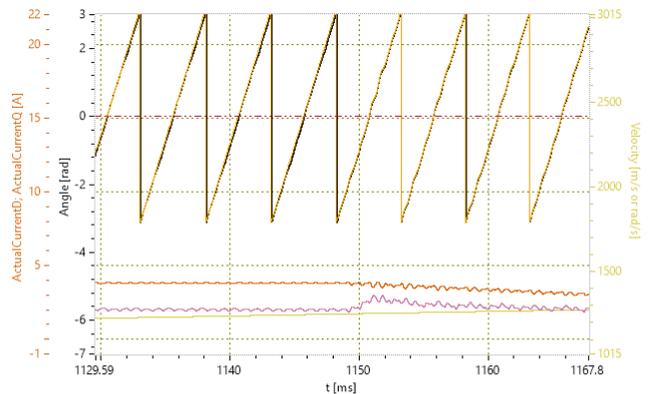
To verify the feed forward settings, the difference between the commutation angle and the sensorless angle can be observed while the spindle is accelerating in synchronous mode (velocity \leq `.../CurrentController/SensorlessTransitionStartVelocity`):

1. Add the following signals to the scope and set sampling time to 0.02ms
 - `.../Signals/Commutation/Angle`
 - `.../Signals/CurrentController/Sensorless/Angle`
 - `.../Signals/CurrentController/ActualCurrentQ`
 - `.../Signals/CurrentController/ActualCurrentD`
 - `.../Signals/PathPlanner/Velocit`
2. Start a acceleration/deceleration move with a maximum velocity $<$ `.../CurrentController/SensorlessTransitionStartVelocity`.
3. Minimize the difference between the commutation angle and the sensorless angle during the acceleration by adjusting the acceleration feed forward. Figure 7 and 8 show the plots before and after the optimization.

Remark In some cases more robust results were achieved, if a_{ff} is set slightly bigger than the value found by the measurement.



Printed on 5/18/2018 2:22:30 PM
 Figure 7: Before optimization of acceleration feed forward.



Printed on 5/18/2018 2:24:17 PM
 Figure 8: After optimization of acceleration feed forward.

Tuning of the Position Controller

To verify the stability of the position controller and to optimize the parameters, a test signal is applied while the spindle is running fast enough to reach the sensorless mode (velocity $>$ `.../CurrentController/SensorlessTransitionEndVelocity`).

1. Prepare the scope for the measurement: Add the following signals to the scope:
 - `.../Signals/PositionController/Controllers[0]/PositionError`

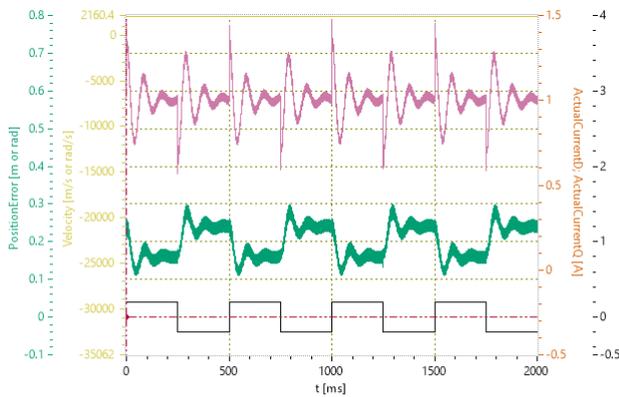
- .../Signals/CurrentController/ActualCurrentQ
- .../Signals/TestGenerator/Output

Use the output signal as trigger and activate repeat mode .

2. Run the spindle fast enough to reach the sensorless mode.
3. Apply a test-signal. Figure 9 shows a possible setup of the test-signal generator used for the optimization of the position controller (.../Commands/TestGenerator). The square signal is added to the current setpoint as soon as the command is set to *StartCurrentSquare*.
4. Now the transient response of the position error can be optimized. Figure 9 and 10 show the plots before and after the optimization.

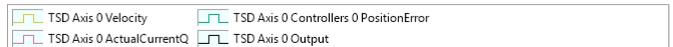
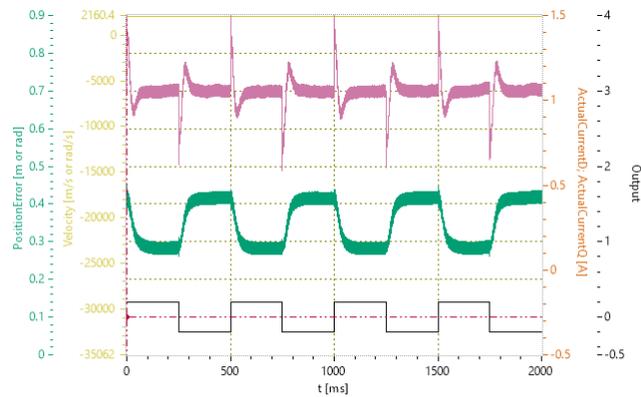
Register	Value	Unit	Description
Command	StartCurrentSquare	-	Test signal generator command
Frequency	2	Hz	Frequency of test signal. (Modulation frequency in case of vector length...
Amplitude	0.5		Amplitude of test signal after ramp-up. (Modulation amplitude in case ...
Offset	0		Offset of test signal after ramp-up. (Start vector length in case of vector ...
RampTime	0	s	Ramp-up time of test signal sequence
CommutationAngle	0	rad	Vector angle in case of vector length modulation. (Don't care with other...

Figure 9: Test-signal generator setup.



Printed on 5/18/2018 2:40:29 PM

Figure 10: Before optimization of the position controller.



Printed on 5/18/2018 2:39:36 PM

Figure 11: After the optimization of the position controller.

4 Trouble-Shooting

Motor peak current error during acceleration or deceleration:

- Reduce jerk and/or acceleration.
- Check stability of the position controller (see section 3.2).

- Check setting of acceleration feed forward(see section 3.2).
- The model tends to be unstable if the inductance L is set too big. In some cases most robust results were achieved if L is set to zero.

Bridge voltage out of range during deceleration:

- During deceleration the energy of the spindle is transferred back to the DC-bus which causes an increase of the bridge-voltage. Check if the power-supply is able to handle this increase of the voltage.
- Reduce the deceleration parameter.

Motor runs in the wrong direction:

- Change value of `.../Parameter/Motor/InvertDirection..`

Motor temperature is too high:

- If the motor has a low inductance, the ripple-current caused by the PWM could lead to additional losses in the motor. The ripple could be analyzed by attaching a current probe to a motor-phase and comparing the signal-to-ripple ratio with an oscilloscope.
- Check `Parameters/CurrentController/PwmFrequency` is set to `pwm100kHz`.
- Add additional inductance with a pre-filter (see <https://www.triamec.com/de/sinus-filter-TF.html>).
- Check the stability of position and current controller.
- Check the cooling of the motor.
- Check the data sheet of to verify that motor is operated within the specification.

Spindle does not turn at start of move (synchronous mode):

- Check if `.../Parameters/Commutation/CurrentAmplitude` is configured correctly.
- Check the parametrization of the current controller.

Value of signals is displayed as zero in scope:

- `.../Signals/CurrentController/Sensorless/` require a sampling rate of 50 kHz or 100kHz.
- Check if plot setting `Sampling Time` is set accordingly.

5 Appendix

5.1 Position Controller

With a PD-controller we get the following closed-loop transfer-function:

$$G_c = \frac{posAct}{posCmd} = \frac{K_t}{J} \frac{K_D s + K_P}{s^2 + \frac{K_t}{J} K_D s + \frac{K_t}{J} K_P}. \quad (6)$$

By comparing the coefficients of the denominator with a second order system with natural frequency $f_{0PC} = 2D\omega_0$ and damping D we get:

$$K_P = \omega_0^2 \frac{J}{K_t}; \quad K_D = 2D\omega_0 \frac{J}{K_t} \quad (7)$$

or with time constant representation of the controller:

$$K_R = \omega_0^2 \frac{J}{K_t}; \quad T_v = 2 \frac{D}{\omega_0} \quad (8)$$

Additionally we set bandwidth limitation of the differentiation to

$$T_1 = \frac{T_v}{10} = \frac{D}{5\omega_0}. \quad (9)$$