MOTOR CONTROL FIRMWARE

Reference Manual

Revision	Date	Firmware Version	Description
01	05-20-2024	release-v1.9.0	
02	07-07-2025	release-v3.0.0	
03	10-10-2025	release-v3.1.0	

Contents

1 Introduction

This document is a reference manual for Infineon's advanced motor control firmware solution. The firmware supports many permutations of control type, controlled entity, feedback type, and startup method as seen below.

Control Type	Controlled Entity	Position Feedback	Startup Method
Open loop	Voltage	-NA-	-NA-
Open loop in RFO	Current	-NA-	-NA-
FOC in RFO	Current	Sensorless	Rotor Pre-Alignment
FOC in RFO	Current	Sensorless	Six Pulse Injection
FOC in RFO	Current	Sensorless	High Frequency Injection
FOC in RFO	Current	Sensorless	Dyno Mode
FOC in RFO	Current	Encoder	Rotor Pre-Alignment
FOC in RFO	Current	Hall Sensor	-NA-
BC in TBC	Current	Hall Sensor	-NA-
TC in TBC	Current	Hall Sensor	-NA-
FOC in SFO	Torque	Sensorless	Rotor Pre-Alignment
FOC in SFO	Torque	Sensorless	Six Pulse Injection
FOC in SFO	Torque	Sensorless	High Frequency Injection
FOC in SFO	Torque	Sensorless	Dyno Mode
FOC in RFO or SFO	Speed	Sensorless	Rotor Pre-Alignment
FOC in RFO or SFO	Speed	Sensorless	Six Pulse Injection
FOC in RFO or SFO	Speed	Sensorless	High Frequency Injection
FOC in RFO or SFO	Speed	Sensorless	Open-Loop Volt/Hz
FOC in RFO	Speed	Sensorless	Open-Loop Current
FOC in RFO	Speed	Encoder	Rotor Pre-Alignment
FOC in RFO	Speed	Hall Sensor	-NA-
FOC in RFO	Position	Encoder	Rotor Pre-Alignment
BC in TBC	Speed	Hall Sensor	-NA-
TC in TBC	Speed	Hall Sensor	-NA-

Acronym	Expansion
BC	Block Commutation
FOC	Field Oriented Control
RFO	Rotor Frame Orientation
SFO	Stator Frame Orientation
TBC	Trapezoidal or Block Commutation
TC	Trapezoidal Commutation

Fig. 1: Various permutations of control type, controlled entity, feedback type, and startup method

The firmware is deployed into multiple Infineon hardware platforms employing various microcontrollers e.g. PSOC6, PSOC-C3, XMC7200, XMC4400. However, the control algorithms are hardware agnostic, and the device dependent part of the firmware is encapsulated in a defined standard hardware interface module.

The chapters of this document describe various modules in the firmware. For each module, the following topics are presented in a sequential order as needed:

- Algorithm description and mathematical framework for technical analysis
- Block diagram describing the controller in the module
- Typical simulation and/or experimental waveforms

The last chapter includes a parameter dictionary that provides a one-to-one mapping between the materials provided in this document and their corresponding parameters in the firmware.

2 Mathematical Conventions

In this section, we set mathematical conventions for the rest of this document.

In general, three phase variables (uvw) such as voltages, currents, fluxes, etc. can be expressed using a vector representation as follows.

$$\mathbf{x}_{uvw} = \begin{bmatrix} x_u \\ x_v \\ x_w \end{bmatrix} \tag{1}$$

Same vector definition can be used for the corresponding Stationary-Reference-Frame (SRF $\alpha\beta 0$) and Rotating-Reference-Frame (RRF $qd0$) representation of such variables:

$$\mathbf{x}_{\alpha\beta 0} = \begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix} \tag{2}$$

$$\mathbf{x}_{qd0} = \begin{bmatrix} x_q \\ x_d \\ x_0 \end{bmatrix} \tag{3}$$

Fig. 2 depicts the reference frame conventions used in the rest of this document for all control methods (RFO, SFO, and TBC). It also shows the conventional hall sensor placement with respect to the motor phases in TBC. As seen in Fig. 2, the d-axis is defined as the direction of the rotor flux λ_m. The stationary $\alpha\beta$ axes are in the same direction as the rotating qd axes at $\theta = 0$.

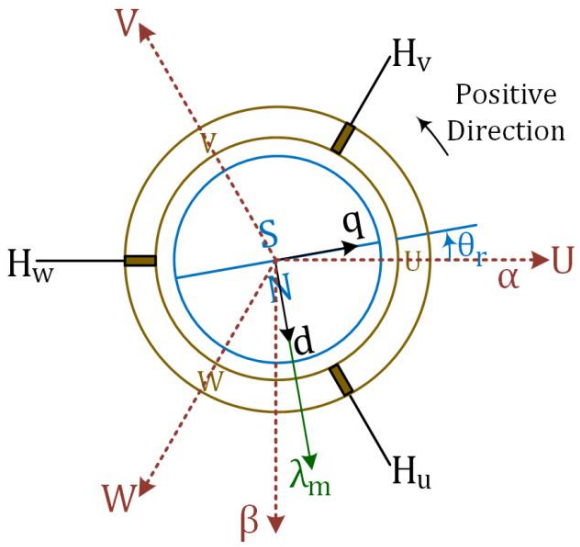

Fig. 2: Reference frame and hall placement conventions for RFO, SFO, and TBC

The transformation from three phase to SRF can be done using the following matrix (also known as Clarke transformation matrix)

$$\mathbf{x}_{\alpha\beta 0} = \boldsymbol{K}(0)\mathbf{x}_{uvw} \tag{4}$$

$$\boldsymbol{K}(0) = \frac{2}{3}\begin{bmatrix} 1 & \cos(-\alpha) & \cos(\alpha) \\ 0 & \sin(-\alpha) & \sin(\alpha) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \tag{5}$$

where $\alpha = 2\pi/3$ here. The transformation from SRF to RRF can be done using the following rotation matrix (also known as Park transformation matrix)

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

It should be noted that Park transform has certain properties that will be used later on in this document, namely

$$\boldsymbol{R}^{-1}(\theta) = \boldsymbol{R}(-\theta) = \boldsymbol{R}^{T}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

and

$$\boldsymbol{R}(\theta_1 + \theta_2) = \boldsymbol{R}(\theta_1)\boldsymbol{R}(\theta_2) = \boldsymbol{R}(\theta_2)\boldsymbol{R}(\theta_1) \quad (8)$$

One can combine Park and Clarke transformation matrices to go directly from three phase to RRF as follows

$$\mathbf{x}_{qd0} = \boldsymbol{R}(\theta)\mathbf{x}_{\alpha\beta0} = \underbrace{\boldsymbol{R}(\theta)\boldsymbol{K}(0)}_{\boldsymbol{K}(\theta)}\mathbf{x}_{uvw} \quad (9)$$

where

$$\boldsymbol{K}(\theta) = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos(\theta-\alpha) & \cos(\theta+\alpha) \\ \sin(\theta) & \sin(\theta-\alpha) & \sin(\theta+\alpha) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \quad (10)$$

Motor windings are either star connected with no ground return or delta connected. Therefore, the zero component of the currents and phase-to-neutral voltages are always zero and one can simplify the vector representations by removing the zero component:

$$\mathbf{x}_{\alpha\beta} = \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} \quad (11)$$

3 Rotor Field Oriented Control

RFO is a variant of sensor-less FOC where the motor's three phase sinusoidal currents are decomposed into q-axis and d-axis DC currents using Clark and Park transformations. These transformations reduce the complexity of the control system for AC machines. Fig. 3 illustrates the overall block diagram of RFO control method which is comprised of modules such as a speed control loop, q- and d-axis current control loops and position feedback.

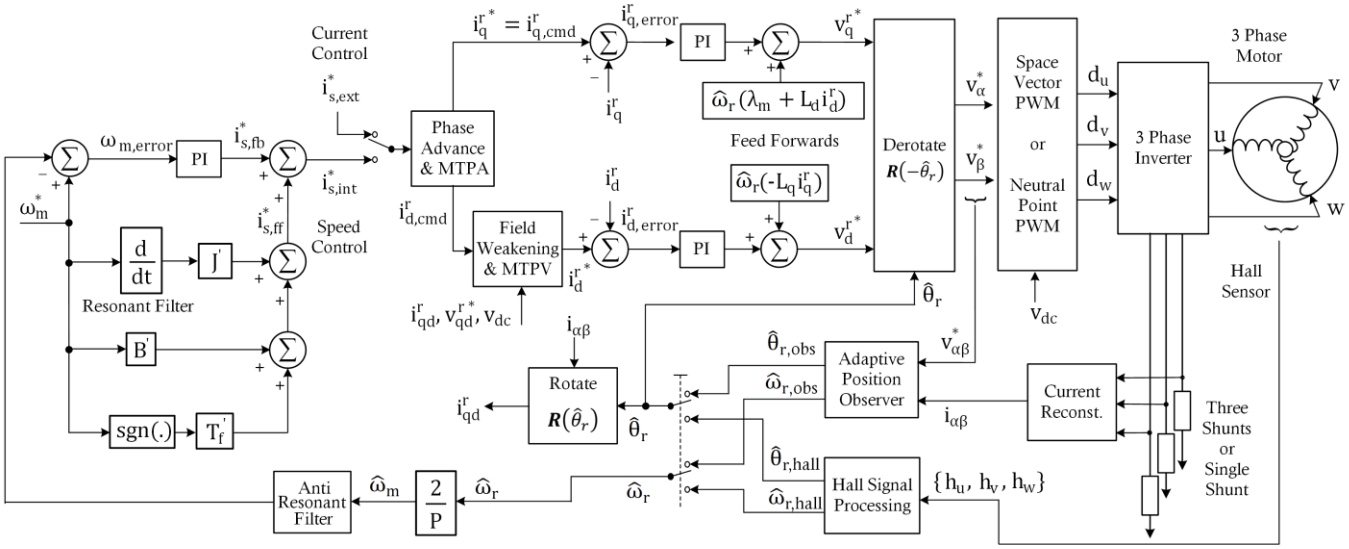

Fig. 3: RFO block diagram

The position information of the rotor is not only required to obtain the speed for speed controller, but also to control the q and d axis currents in an optimized way for both efficiency and dynamic performance. The position and speed information can be extracted by either using a position sensor (such as encoder, resolver, or hall sensors) or a sensor-less approach as it will be described extensively in section 3.5. The speed controller then uses the speed information to create a current reference using a PI controller as will be detailed in section 3.1. If the motor under control is an Interior PMSM (IPM) extra reluctance torque (beside permanent magnet torque) can be generated by injecting current in d-axis. As it will be described in section 3.2, Maximum Torque Per Ampere (MTPA) block can be used to generate the right amount of d-axis and q-axis currents to maximize the torque for a specific amount of current and overall efficiency. If operations above the base speed is required, then the Maximum Torque Per Volt (MTPV) can be used to achieve higher operating speeds as will be explained in section 3.3. MTPA and MTPV create the reference for the q and d axes which are used to control the current as will be detailed in 3.4. At the end, the voltage references created by the current controller, are applied to the inverter using a selectable modulation scheme (chapter 6.3).

3.1 Speed Controller

This section will cover the speed controller, which is common across all the three control types such as RFO, SFO, and TBC. The parameters of the speed controller are significantly impacted by the mechanical load. Fig. 4 illustrates the mathematical model of the mechanical load driven by the motor and electrical drive system. This model includes T_f, representing coulombic friction, B as the coefficient of viscous friction, and J denoting the inertia. When utilizing the firmware to operate any motor, it is crucial to accurately measure or estimate these three parameters and input them into the graphical user interface (GUI) or hard code them in the appropriate header files. The speed controller's k_p, k_i, and feedforward terms are directly derived from these parameters, as will be explained further.

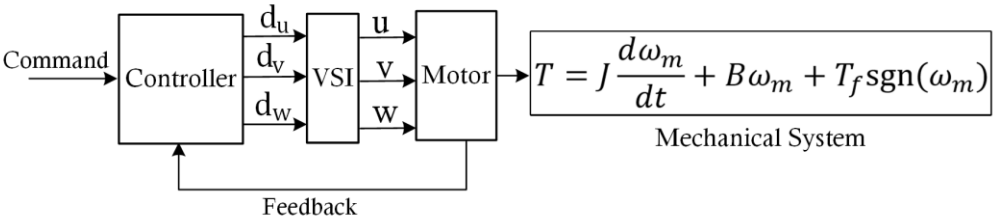

Fig. 4: Mechanical load run by motor-drive system.

To derive the value of proportional, integral, and feedforward terms of speed controller, the speed loop block diagram along with a simple model of motor and mechanical load will be used as shown in Fig. 5 where:

$$k_t \approx \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\lambda_m \tag{12}$$

Pole-zero cancellation technique is used to find the PI controller integral and proportional gain. By having

$$\frac{k_i}{k_p} = \frac{B}{J} \tag{13}$$

the controller zero will cancel the mechanical load's pole. The PI controller coefficients are also proportional to speed loop bandwidth as shown below:

$$k_i \propto B\omega_{BW} \tag{14}$$

$$k_p \propto J\omega_{BW} \tag{15}$$

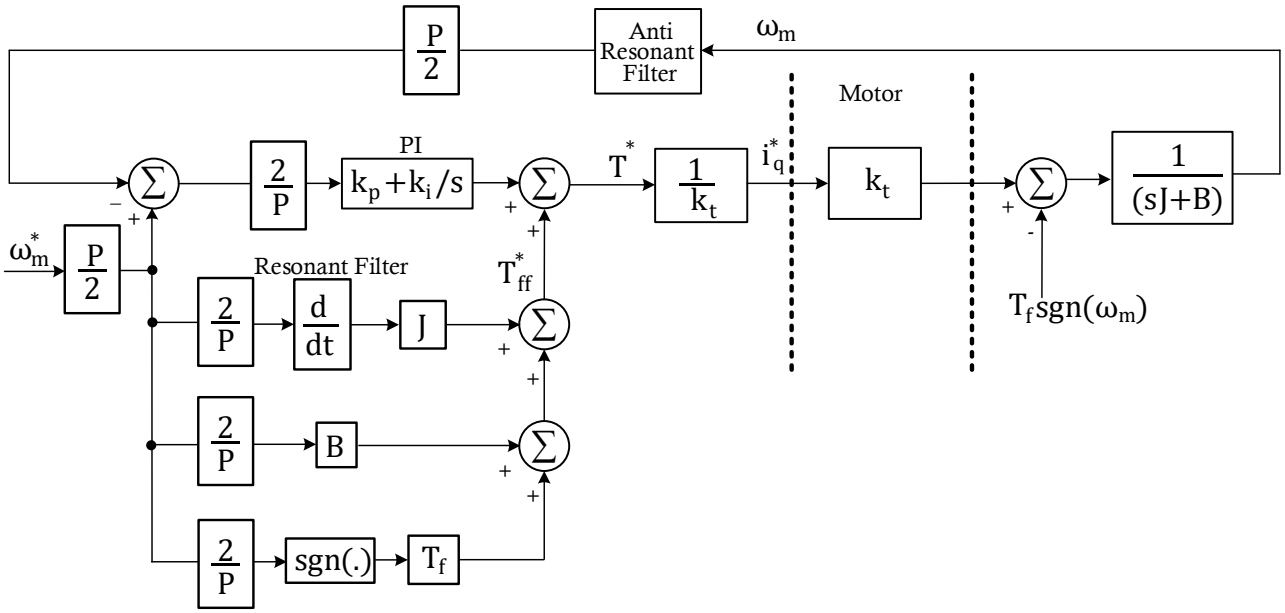

Fig. 5: RFO speed loop block diagram before rescaling the parameters

The proportional and integral gains are ultimately determined after being rescaled to account for the motor parameters k_t and P. Fig. 6 illustrates how k_t and P as were shown in Fig. 5 can be integrated into the controller parameters, leading to the new updated forms:

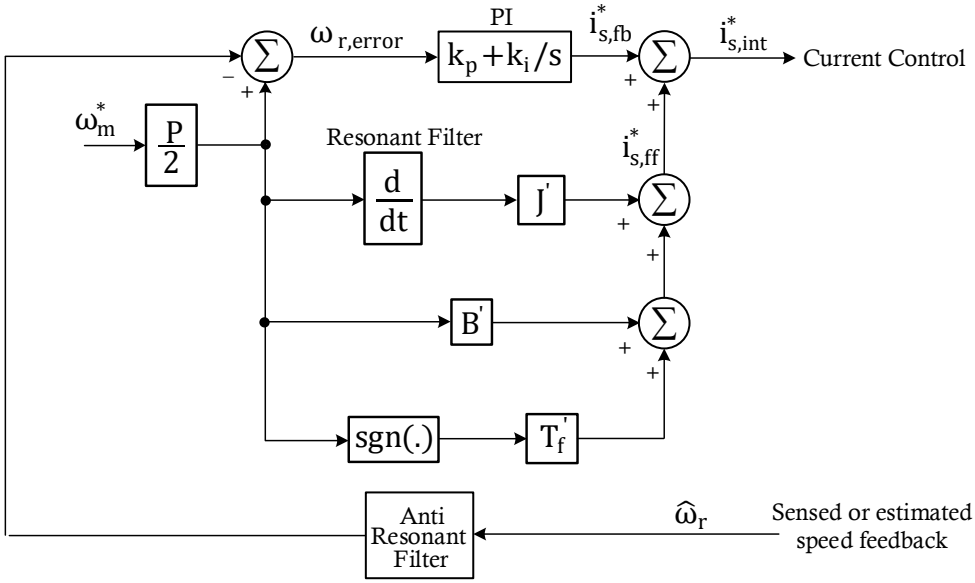

Fig. 6: RFO speed loop block diagram as implemented in the firmware

The speed contoller parameters after rescaling are as follows:

$$k_i = \left(\frac{1}{k_t}\right)\left(\frac{2}{P}\right) B\omega_{BW} = \left(\frac{8}{3}\right)\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right) B\omega_{BW} \quad (16)$$

$$k_p = \left(\frac{1}{k_t}\right)\left(\frac{2}{P}\right) J\omega_{BW} = \left(\frac{8}{3}\right)\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right) J\omega_{BW} \quad (17)$$

The feedforward terms can also be updated as depicted in the following manner:

$$B' = \left(\frac{1}{k_t}\right)\left(\frac{2}{P}\right) B = \left(\frac{8}{3}\right)\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right) B \quad (18)$$

$$J' = \left(\frac{1}{k_t}\right)\left(\frac{2}{P}\right) J = \left(\frac{8}{3}\right)\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right) J \quad (19)$$

$$T_f' = \left(\frac{1}{k_t}\right) T_f = \left(\frac{4}{3}\right)\left(\frac{1}{P}\right)\left(\frac{1}{\lambda_m}\right) T_f \quad (20)$$

The feedforward terms in the speed loop are affected by both mechanical load and motor parameters. These three feedforward terms play a role in enhancing the dynamic performance of the speed loop. The inertia term utilizes a second-order resonant filter to estimate the acceleration, aiming to mitigate the potential impact of noise that could arise if a direct derivation method had been employed.

3.2 Phase Advance & MTPA

MTPA block is responsible for generating command values for q and d axis currents (i^r_q and i^r_d) to maximize the torque for the overall current injected in the motor winding. The torque equation of the PMSM motor is written as

$$T = \frac{3P}{4}\left(\lambda_m i^r_q + (L_d - L_q) i^r_q i^r_d\right) \quad (21)$$

The d and q axis current commands $i^r_{d,cmd}$ and $i^r_{q,cmd}$ need to be obtained from the reference current magnitude i^*_s generated by the output of the speed controller as follows

$$i^r_{d,cmd} = i^*_s \sin\theta \quad (22)$$

$$i^r_{q,cmd} = i^*_s \cos\theta$$

where θ is the angle of the current i_s^* with respect to rotor q axis.

Plugging (22) into (21) and rearranging the equations yields the following torque equation as a function of θ as,

$$\mathrm{T} = \frac{3P}{4}\left(\lambda_m i_s^* \cos\theta + \frac{(L_d - L_q)}{2} i_s^{*2} \sin 2\theta\right) \tag{23}$$

To find the maximum of the torque function, the derivative of T can be set to zero:

$$\frac{dT}{d\theta} = 0 \tag{24}$$

which results in,

$$\sin\theta = \frac{\sqrt{{\lambda_m}^2 + 8(L_d - L_q)^2 i_s^{*2}} - \lambda_m}{4(L_d - L_q) i_s^*} \tag{25}$$

Plugging (25) back into (22) and rearranging, results in MTPA current command values as follows

$$\mathrm{i}_{\mathrm{d,MTPA}}^{\mathrm{r}} = \frac{\lambda_m - \sqrt{{\lambda_m}^2 + 8(L_q - L_d)^2 i_s^{*2}}}{4(L_q - L_d)} \tag{26}$$

$$\mathrm{i}_{\mathrm{q,MTPA}}^{\mathrm{r}} = \sqrt{i_s^{*2} - {\mathrm{i}_{\mathrm{d,MTPA}}^{\mathrm{r}}}^2} \cdot \mathrm{sgn}(i_s^*)$$

From (26), it can be seen that optimal $\mathrm{i}_{\mathrm{d,MTPA}}^{\mathrm{r}}$ is always negative whereas optimal $\mathrm{i}_{\mathrm{q,MTPA}}^{\mathrm{r}}$ has the same polarity as the commanded current by the speed loop.

3.3 Flux Weakening & MTPV

Flux weakening (or field weakening) is a method to increase the speed of the motor beyond its base speed. At base speed, the inverter voltage becomes saturated which means that there would not be enough voltage generated on the inverter to overcome the back EMF of the motor and increase the speed above the base speed. Therefore, the flux wakening method is needed to reduce the back EMF voltage of the motor and further increase the speed. Fig. 7 illustrates the flux weakening controller for RFO.

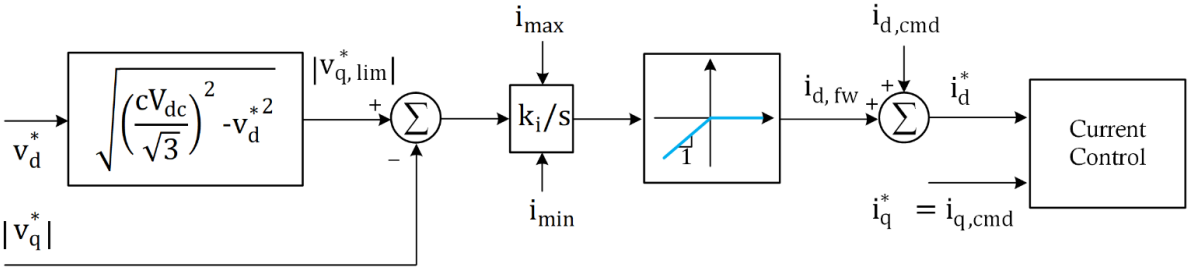

Fig. 7: Flux weakening block diagram in RFO.

As mentioned, the controller requires more voltage, v_q^* than can produced by the inverter $v_{q,\lim}^*$ above the base limit. Note that the coefficient $c = 0.90{\sim}0.95$ used for the voltage limit is needed to leave some margin before the absolute voltage limitation is reached to avoid instability. The voltage error required to increase the motor speed is passed through an integrator to generate the amount of the current in d axis for weakening the flux and allow for speed increase. Note that the current generated by the integrator needs to be limited as follows

$$i_{\max} = \min\left(KI_{d,\max}, \sqrt{I_{\lim}^2 - i_{q,\mathrm{fb}}^2}\right) - i_{d,\mathrm{cmd}} \tag{27}$$
$$i_{\min} = \max\left(-I_{d,\max}, -\sqrt{I_{\lim}^2 - i_{q,\mathrm{fb}}^2}\right) - i_{d,\mathrm{cmd}}$$

Where $\mathrm{K} = 0.05$ is a coefficient to increase the d axis current saturation level above zero, $\mathrm{I_{d,max}}$ is maximum allowable d axis current in the motor and $I_{\lim}$ is the maximum three phase allowable current in the motor windings which is determined by I^2T protection algorithm. The field weakening current must always be a negative current and be added to MTPA current command $i^r_{d,cmd}$.

In (27), $I_{d,\max}$ needs to be selected based on the maximum allowable current of the motor and at the same time it must not exceed the level where demagnetization would occur in the rotor magnets. $I_{\lim}$ is also the output of I^2T as will be discussed in section 6.12.

To design the gain of the integrator, it is necessary to derive the small signal model of the system. The dynamics equations of the system can be obtained from Fig. 7 as,

$$\varepsilon = v_{q,\lim} - v_q \rightarrow \hat{\varepsilon} = \frac{\partial v_{q,\lim}}{\partial v_d}\hat{v}_d - \hat{v}_q = \frac{-V_d}{V_q}\hat{v}_d - \hat{v}_q \tag{28}$$
$$\hat{\imath}_d = \frac{k_i}{s}\hat{\varepsilon}$$

where variables with ^ refer to the small signal values and V_d and V_q are the equilibrium points. Also, $\hat{v}_q$ and $\hat{v}_d$ can be expressed as

$$\hat{v}_q = \omega_r L_d \hat{\imath}_d\,, \qquad \hat{v}_d = r\hat{\imath}_d \tag{29}$$

Therefore,

$$\hat{\varepsilon} = \left(\frac{-V_d}{V_q}r - \omega_r L_d\right)\hat{\imath}_d \tag{30}$$

Combining (28) and (30) yields the bandwidth of the controller ω_c as,

$$\omega_c = \left(\frac{V_d}{V_q}r + \omega_r L_d\right)k_i \tag{31}$$

Assuming $V_q = V_{\max}$, $V_{\max} = \omega_r \lambda_m$, and $\frac{V_d}{V_q}r$ term is negligible, k_i can be calculated based on the desired bandwidth, ω_c, as

$$k_i = \frac{\omega_c \lambda_m}{V_{\max} L_d} \tag{32}$$

3.4 Current Controller

The d and q current commands are used as an input for the current controllers and the output would be the voltage references. The voltage references v_d^{r*} and v_q^{r*} are eventually applied to the motor using the inverter to control the current. This means that the voltages are considered as an input to the model of the PMSM machine as,

$$v_d^{r*} = R_s i_d^r + L_d\frac{di_d^r}{dt} - \omega_r\left(L_q i_q^r\right) \tag{33}$$
$$v_q^{r*} = R_s i_q^r + L_q\frac{di_q^r}{dt} + \omega_r(\lambda_m + L_d i_d^r)$$

In these equations, the d and q axis currents are not decoupled, and one has a dependency on the other. To decouple this dependency and simplify the equations for the control system, feedforward terms can be added to the output of the current controllers according to Fig. 3 and reduce the PMSM system to,

$$v_d^{r*} = R_s i_d^r + L_d\frac{di_d^r}{dt} \tag{34}$$

$$v_q^{r*} = R_s i_q^r + L_q \frac{di_q^r}{dt}$$

Now a PI controller can be easily designed for these set of equations with the equivalent control block diagram shown as followed,

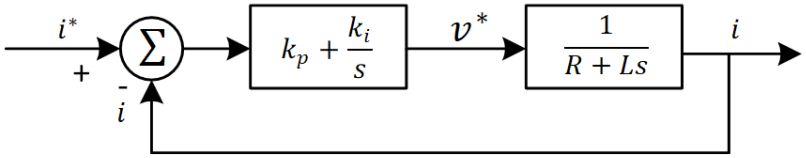

Fig. 8: Current controller block diagram.

In the closed loop system in Fig. 8, it is desired to cancel the pole of the system with the zero of the PI controller to reduce the order of closed loop system, i.e.

$$\frac{k_p}{k_i} = \frac{L}{R} \tag{35}$$

This will reduce the closed loop transfer function of the system to,

$$H_{cl}(s) = \frac{k_i}{Rs} = \frac{k_p}{Ls} \tag{36}$$

which can be used to calculate the PI controller coefficients for a given system bandwidth ω_c as,

$$k_p = \omega_c L \tag{37}$$
$$k_i = \omega_c R$$

3.5 Adaptive Sensorless Observer

Observers are control loops that are designed to estimate or “observe” some internal variables of the system without directly measuring them. In the context of motor control applications, a sensorless observer estimates the rotor angular position and speed without using an actual position sensor. Some of the advantages of sensorless observers include cost reduction by eliminating the position sensor, increasing the system reliability because the sensor can fail, and increasing the maximum operating temperature of the motor since position sensors are mounted on the motor and usually cannot operate at high temperatures.

A sensorless observer with the following characteristics is utilized in the firmware:

- Universal
  - Supporting both motor types (SPMs and IPMs)
  - Supporting both FOC methods (RFO and SFO)
- Adaptive
  - Control parameters are not constant and adapt to changes in the operating point and speed of the motor
- Zero phase shift at all speeds
  - Non-zero phase-shift in estimated position means the motor is not operating at Maximum Torque Per Amp (MTPA) condition
  - This means lower than optimal efficiency
  - The efficiency is especially critical for Battery Powered Application (BPA) where minimizing unnecessary losses is of utmost importance to maximize battery run time

- Ability to observe both
  - Rotor position and speed (for RFO and SFO)
  - Stator flux magnitude and load angle (for SFO)

This section describes the operation of this observer in RFO, whereas its operation in SFO is presented in section 4.8.

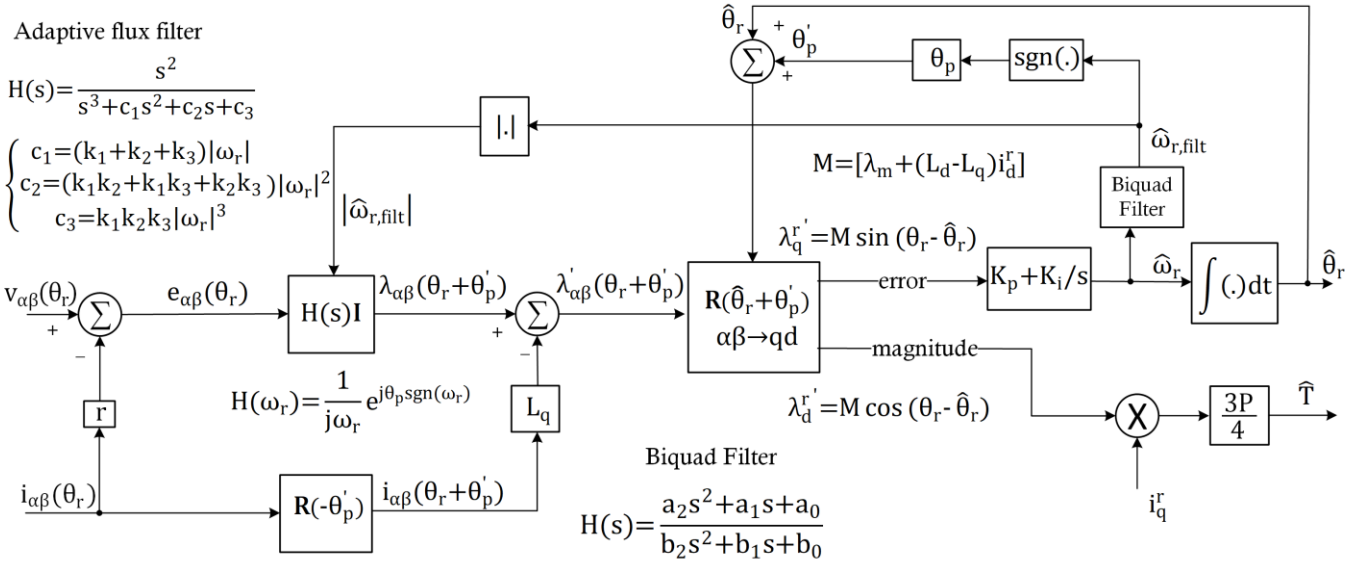

Fig. 9: Universal Adaptive Zero-Phase-Shift Position Observer in RFO

Fig. 9 depicts the block diagram of the observer in RFO. First, the motor back-EMF voltages are calculated as seen in Fig. 9. Motor back-EMF voltages, which are the derivatives of the stator fluxes, can be expressed in terms of the applied inverter voltages and currents in SRF as follows.

$$\boldsymbol{e}_{\alpha\beta}(\theta_r)=\frac{d\boldsymbol{\lambda}_{\alpha\beta}(\theta_r)}{dt}=\boldsymbol{v}_{\alpha\beta}(\theta_r)-r\boldsymbol{i}_{\alpha\beta}(\theta_r) \tag{38}$$

The back-EMF voltage vector is then fed to the transfer functions $H(s)\boldsymbol{I}$. Here, without loss of generality, we define the transfer function $H(s)$ as any transfer function that can satisfy the following criteria.

$$|H(|\omega_r|)|=\frac{1}{|\omega_r|} \qquad \measuredangle H(|\omega_r|)=-\frac{\pi}{2}+\theta_p \tag{39}$$
$$\lim_{\omega\to\infty}H(\omega)\sim\frac{1}{j\omega} \qquad \lim_{\omega\to0}H(\omega)\sim(j\omega)^2$$

Such criteria ensure that $H(s)$ acts as an integrator at high frequencies, has constant phase shift at the electrical frequency of the motor no matter the speed, and has -40dB/dec of attenuation at low frequencies to eliminate DC offset due to integration and/or ADC offset and noise.

The phase shift at positive electrical frequencies is positive:

$$H(|\omega_r|)=\frac{1}{j|\omega_r|}\cdot e^{j\theta_p} \quad (\theta_p>0) \tag{40}$$

In addition, since the impulse response $h(t)$ is always real (no imaginary component), $H(s)$ must have complex conjugate symmetry. Therefore,

$$H(-|\omega_r|) = H^*(|\omega_r|) = \frac{1}{-j|\omega_r|} \cdot e^{-j\theta_p} \quad (\theta_p > 0) \tag{41}$$

Combining (40) and (41), it can be concluded that

$$H(\omega_r) = \frac{1}{j\omega_r} \cdot e^{j\theta_p \mathrm{sgn}(\omega_r)} \quad (\theta_p > 0) \tag{42}$$

In other words, the phase shift at positive electrical frequencies is always positive and the phase shift at negative electrical frequencies is always negative.

Without loss of generalization, one can assume that a scalar input to $H(s)$ can be expressed as

$$x(\theta_r) = X\cos(\theta_r) = X\cos(\omega_r t + \varphi) = \frac{X}{2}\left(e^{j\omega_r t}e^{j\varphi} + e^{-j\omega_r t}e^{-j\varphi}\right) \tag{43}$$

where θ_r, ω_r, and φ are the rotor angle, rotor electrical frequency, and phase offset of x, respectively.

Taking the Fourier transform of (43), we have

$$x(\omega) = \left(\frac{X}{2}\right)(2\pi)\left(e^{j\varphi}\delta(\omega - \omega_r) + e^{-j\varphi}\delta(\omega + \omega_r)\right) \tag{44}$$

Therefore, the output of $H(s)$ in frequency domain would be

$$y(\omega) = x(\omega)H(\omega) = \tag{45}$$
$$\left(\frac{X}{2}\right)(2\pi)\left(e^{j\varphi}\delta(\omega - \omega_r)H(\omega_r) + e^{-j\varphi}\delta(\omega + \omega_r)H(-\omega_r)\right)$$

Replacing $H(\omega_r)$ and $H(-\omega_r)$ according to (42), we will have

$$y(\omega) = \frac{X\pi}{j\omega_r}\left(e^{j\varphi}\delta(\omega - \omega_r)e^{j\theta_p\mathrm{sgn}(\omega_r)} - e^{-j\varphi}\delta(\omega + \omega_r)e^{-j\theta_p\mathrm{sgn}(\omega_r)}\right) \tag{46}$$
$$= \frac{X\pi}{j\omega_r}\left(e^{j\left(\varphi+\theta_p\mathrm{sgn}(\omega_r)\right)}\delta(\omega - \omega_r) - e^{-j\left(\varphi+\theta_p\mathrm{sgn}(\omega_r)\right)}\delta(\omega + \omega_r)\right)$$

We can transform the output from the frequency domain back to the time domain as follows.

$$y(t) = \frac{1}{j\omega_r} \cdot \frac{X}{2}\left(e^{j\left(\varphi+\theta_p\mathrm{sgn}(\omega_r)\right)}e^{j\omega_r t} - e^{-j\left(\varphi+\theta_p\mathrm{sgn}(\omega_r)\right)}e^{-j\omega_r t}\right) \tag{47}$$
$$= \frac{X}{\omega_r} \cdot \frac{1}{2j}\left(e^{j\left(\omega_r t+\varphi+\theta_p\mathrm{sgn}(\omega_r)\right)} - e^{-j\left(\omega_r t+\varphi+\theta_p\mathrm{sgn}(\omega_r)\right)}\right)$$

Using Euler's formula, one can simplify (47) to

$$y(t) = \frac{1}{\omega_r} \cdot X\sin\left(\omega_r t + \varphi + \theta_p\mathrm{sgn}(\omega_r)\right) \tag{48}$$
$$= \int X\cos\left(\omega_r t + \varphi + \theta_p\mathrm{sgn}(\omega_r)\right)dt + c$$
$$= \int x\left(\theta_r + \theta_p\mathrm{sgn}(\omega_r)\right)dt + c$$

Let us define

$$\theta_p' = \theta_p\mathrm{sgn}(\omega_r) \tag{49}$$

Therefore, according to (48), when $\boldsymbol{e}_{\alpha\beta}(\theta_r)$ is applied as the input of $H(s)$, the output would be

$$\int \boldsymbol{e}_{\alpha\beta}\left(\theta_r + \theta_p\mathrm{sgn}(\omega_r)\right)dt = \boldsymbol{\lambda}_{\alpha\beta}\left(\theta_r + \theta_p\mathrm{sgn}(\omega_r)\right) = \boldsymbol{\lambda}_{\alpha\beta}\left(\theta_r + \theta_p'\right) \tag{50}$$

What (50) indicates is that there is always a constant positive or negative phase shift at positive or negative rotor electrical frequencies, respectively.

In the other part of the observer's block diagram in Fig. 9, a Park transform of angle $-\theta_p'$ is applied to the SRF currents as follows.

$$\boldsymbol{R}(-\theta'_p)\boldsymbol{i}_{\alpha\beta}(\theta_r) = \begin{bmatrix} \cos(\theta'_p) & \sin(\theta'_p) \\ -\sin(\theta'_p) & \cos(\theta'_p) \end{bmatrix} \begin{bmatrix} I\cos(\theta_r+\varphi) \\ -I\sin(\theta_r+\varphi) \end{bmatrix} \tag{51}$$
$$= \begin{bmatrix} I\cos(\theta_r+\varphi+\theta'_p) \\ -I\sin(\theta_r+\varphi+\theta'_p) \end{bmatrix} = \boldsymbol{i}_{\alpha\beta}(\theta_r+\theta'_p)$$

What (51) indicates is that using $\boldsymbol{R}(-\theta'_p)$, the phase shift in the current $\boldsymbol{i}_{\alpha\beta}(\theta_r+\theta'_p)$ will match that of the flux $\boldsymbol{\lambda}_{\alpha\beta}(\theta_r+\theta'_p)$.

Then, according to the block diagram, one can combine the SRF fluxes and currents as follows

$$\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r+\theta'_p) = \boldsymbol{\lambda}_{\alpha\beta}(\theta_r+\theta'_p) - L_q\boldsymbol{R}(-\theta'_p)\boldsymbol{i}_{\alpha\beta}(\theta_r) = \tag{52}$$
$$\boldsymbol{\lambda}_{\alpha\beta}(\theta_r+\theta'_p) - L_q\boldsymbol{i}_{\alpha\beta}(\theta_r+\theta'_p) =$$
$$\boldsymbol{R}(-\theta_r-\theta'_p)(\boldsymbol{\lambda}^r_{qd} - L_q\boldsymbol{i}^r_{qd})$$

where $\boldsymbol{x}^r_{qd}$ are the corresponding RRF variables in the rotor's reference frame (where d-axis is aligned with the motor's permanent magnet flux).

Expanding $\boldsymbol{\lambda}^r_{qd} - L_q\boldsymbol{i}^r_{qd}$ in (52) will result in

$$\boldsymbol{\lambda}^r_{qd} - L_q\boldsymbol{i}^r_{qd} = \begin{bmatrix} L_q i^r_q \\ \lambda_m + L_d i^r_d \end{bmatrix} - \begin{bmatrix} L_q i^r_q \\ L_q i^r_d \end{bmatrix} = \begin{bmatrix} 0 \\ \lambda_m + (L_d - L_q) i^r_d \end{bmatrix} \tag{53}$$

What (53) indicates is that $\boldsymbol{\lambda}^r_{qd} - L_q\boldsymbol{i}^r_{qd}$ has no vector component along the q-axis. Therefore,

$$\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r+\theta'_p) = \tag{54}$$
$$\boldsymbol{R}(-\theta_r-\theta'_p)(\boldsymbol{\lambda}^r_{qd} - L_q\boldsymbol{i}^r_{qd}) =$$
$$\begin{bmatrix} \cos(\theta_r+\theta'_p) & \sin(\theta_r+\theta'_p) \\ -\sin(\theta_r+\theta'_p) & \cos(\theta_r+\theta'_p) \end{bmatrix} \begin{bmatrix} 0 \\ \lambda_m + (L_d - L_q) i^r_d \end{bmatrix} =$$
$$(\lambda_m + (L_d - L_q) i^r_d) \begin{bmatrix} \sin(\theta_r+\theta'_p) \\ \cos(\theta_r+\theta'_p) \end{bmatrix}$$

Using (54), we can extract the rotor angle as follows.

$$\theta_r + \theta'_p = \tan^{-1}\left(\frac{\lambda'_\alpha}{\lambda'_\beta}\right) \tag{55}$$

As a byproduct, the magnitude of $\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r+\theta'_p)$ will be as follows

$$\left|\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r+\theta'_p)\right| = (\lambda_m + (L_d - L_q) i^r_d) \tag{56}$$

This magnitude will be shown to be useful in estimating the torque later on. The $\tan^{-1}(\cdot)$ function in (55) is implemented using a Phase Lock Loop (PLL) as seen in the block diagram in Fig. 9.

According to the block diagram, first we take the Park transform of the input $\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r+\theta'_p)$ using the estimated angle $\hat{\theta}_r$ and applying the same offset θ'_p:

$$\boldsymbol{\lambda}^{r\prime}_{qd} = \boldsymbol{R}(\hat{\theta}_r+\theta'_p)\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r+\theta'_p) = \tag{57}$$
$$[\lambda_m + (L_d - L_q) i^r_d] \begin{bmatrix} \cos(\hat{\theta}_r+\theta'_p) & -\sin(\hat{\theta}_r+\theta'_p) \\ \sin(\hat{\theta}_r+\theta'_p) & \cos(\hat{\theta}_r+\theta'_p) \end{bmatrix} \begin{bmatrix} \sin(\theta_r+\theta'_p) \\ \cos(\theta_r+\theta'_p) \end{bmatrix} =$$
$$[\lambda_m + (L_d - L_q) i^r_d] \begin{bmatrix} \sin(\theta_r-\hat{\theta}_r) \\ \cos(\theta_r-\hat{\theta}_r) \end{bmatrix}$$

According to (57), the q-axis and d-axis components of the resulting vector will be error and magnitude signals, respectively:

$$\lambda_q^{r\prime} = \left[\lambda_m + \left(L_d - L_q\right)i_d^r\right]\sin\underbrace{\left(\theta_r - \hat{\theta}_r\right)}_{\theta_{r,\text{error}}} \tag{58}$$

$$\lambda_d^{r\prime} = \left[\lambda_m + \left(L_d - L_q\right)i_d^r\right]\cos\underbrace{\left(\theta_r - \hat{\theta}_r\right)}_{\theta_{r,\text{error}}} \tag{59}$$

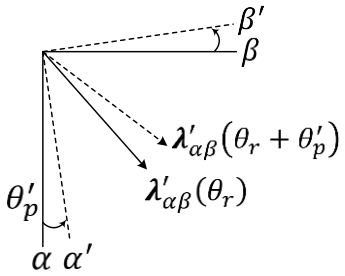

Fig. 10: Rotation in SRF coordinates and its effect on the phase-shift of the signals in the time domain

Fig. 10 illustrates how $\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r + \theta'_p)$ in $\alpha'\beta'$ frame has identical vector components to those of $\boldsymbol{\lambda}'_{\alpha\beta}(\theta_r)$ in $\alpha\beta$ frame. Therefore, one can compensate for the phase-shift in the time domain by adding θ'_p to the rotation angle $\hat{\theta}_r$ in the Park transform $\boldsymbol{R}(\hat{\theta}_r + \theta'_p)$ as seen in Fig. 9 and was mathematically shown in (57) as well.

If the PLL is stable, the steady state error is always zero $\theta_{r,\text{error}} = 0$. Therefore, the d-axis component will converge to $\left[\lambda_m + \left(L_d - L_q\right)i_d^r\right]$ and $\hat{\theta}_r$ will converge to θ_r.

The torque in PMSMs, whether IPM or SPM, can be written as

$$T = \frac{3}{2}\cdot\frac{P}{2}\left(\lambda_d^r i_q^r - \lambda_q^r i_d^r\right) = \tag{60}$$

$$\frac{3}{2}\cdot\frac{P}{2}\left(\lambda_m + \left(L_d - L_q\right)i_d^r\right)i_q^r =$$

Therefore, we can estimate the torque using the magnitude component estimated by PLL in steady state:

$$\hat{T} = \frac{3}{2}\cdot\frac{P}{2}\lambda_d^{r\prime} i_q^r \tag{61}$$

3.5.1 Adaptive Flux Extraction Filter

Without loss of generality, we defined the transfer function $H(s)$ as any transfer function that satisfies the criteria defined in (39). The fact that the phase shift of $H(s)$ is always constant at $|\omega_r|$ indicates that $H(s)$ must be an adaptive transfer function whose coefficients change according to the operating point of the system. Here, we present a specific realization of such adaptive transfer function without loss of generality:

$$H(s) = \frac{1}{A}G(s) \quad A = |\omega_r G(\omega_r)| \quad \lim_{\omega\to\infty}|j\omega G(\omega)| = 1 \tag{62}$$

$$G(s) = \frac{\frac{1}{s}}{1 + \frac{1}{s}\left(c_1 + \frac{1}{s}\left(c_2 + \frac{c_3}{s}\right)\right)} = \frac{s^2}{s^3 + c_1 s^2 + c_2 s + c_3} \tag{63}$$

Assume we can factorize the third-degree polynomial in the denominator of (63) such that each root is proportional to the absolute electrical frequency of the rotor $|\omega_r|$ as follows

$$G(s) = \frac{s^2}{(s + k_1|\omega_r|)(s + k_2|\omega_r|)(s + k_3|\omega_r|)} = \tag{64}$$

$$\frac{s^2}{s^3 + (k_1 + k_2 + k_3)|\omega_r|s^2 + (k_1k_2 + k_2k_3 + k_1k_3)|\omega_r|^2 s + (k_1k_2k_3)|\omega_r|^3}$$

By equating (63) and (64), we can find the coefficients $\{c_1, c_2, c_3\}$:

$$\begin{cases} c_1 = \underbrace{(k_1 + k_2 + k_3)}_{c_1'} |\omega_r| = c_1'|\omega_r| \\ c_2 = \underbrace{(k_1k_2 + k_2k_3 + k_1k_3)}_{c_2'} |\omega_r|^2 = c_2'|\omega_r|^2 \\ c_3 = \underbrace{(k_1k_2k_3)}_{c_3'} |\omega_r|^3 = c_3'|\omega_r|^3 \end{cases} \quad (65)$$

Now, we can evaluate $H(s)$ at $s = j\omega_r$ to see if satisfies the criteria in (39):

$$\begin{aligned} G(\omega_r) &= \frac{(j\omega_r)^2}{(j\omega_r)^3 + c_1'|\omega_r|(j\omega_r)^2 + c_2'|\omega_r|^2(j\omega_r) + c_3'|\omega_r|^3} \\ &= \frac{1}{j\omega_r} \cdot \frac{j^3}{j^3 + j^2 c_1' \mathrm{sgn}(\omega_r) + jc_2' + c_3' \mathrm{sgn}(\omega_r)} \\ &= \frac{1}{j\omega_r} \cdot \frac{1}{1 - jc_1' \mathrm{sgn}(\omega_r) - c_2' + jc_3' \mathrm{sgn}(\omega_r)} \\ &= \frac{1}{j\omega_r} \cdot \frac{1}{(1 - c_2') - j\mathrm{sgn}(\omega_r)(c_1' - c_3')} \\ &= \frac{1}{j\omega_r} \cdot \frac{(1 - c_2') + j\mathrm{sgn}(\omega_r)(c_1' - c_3')}{(1 - c_2')^2 + (c_1' - c_3')^2} \end{aligned} \quad (66)$$

If we select A such that

$$\frac{1}{A} = \sqrt{(1 - c_2')^2 + (c_1' - c_3')^2} = \quad (67)$$

$$\sqrt{\left(1 - (k_1k_2 + k_2k_3 + k_1k_3)\right)^2 + \left((k_1 + k_2 + k_3) - (k_1k_2k_3)\right)^2}$$

it would be obvious that $|H(|\omega_r|)| = 1/|\omega_r|$. Therefore, the first criteria in (39) is satisfied.

Also, the phase shift at ω_r can be found by examining (66):

$$\theta_p' = \tan^{-1}\left(\frac{\mathrm{sgn}(\omega_r)(c_1' - c_3')}{(1 - c_2')}\right) = \mathrm{sgn}(\omega_r)\tan^{-1}\left(\frac{c_1' - c_3'}{1 - c_2'}\right) = \mathrm{sgn}(\omega_r)\theta_p \quad (68)$$

where

$$\theta_p = \tan^{-1}\left(\frac{c_1' - c_3'}{1 - c_2'}\right) = \tan^{-1}\left(\frac{(k_1 + k_2 + k_3) - (k_1k_2k_3)}{1 - (k_1k_2 + k_2k_3 + k_1k_3)}\right) \quad (69)$$

Therefore, the second criteria in (39) is also satisfied.

In addition, since the nominator and denominator of $H(s)$ are second-degree and third-degree polynomials, $\lim_{\omega\to\infty} H(\omega) \sim 1/j\omega$ and $\lim_{\omega\to 0} H(\omega) \sim (j\omega)^2$. Therefore, the third and fourth criteria in (39) are also satisfied.

Fig. 11 depicts the block diagram of the adaptive transfer function $H(s)$.

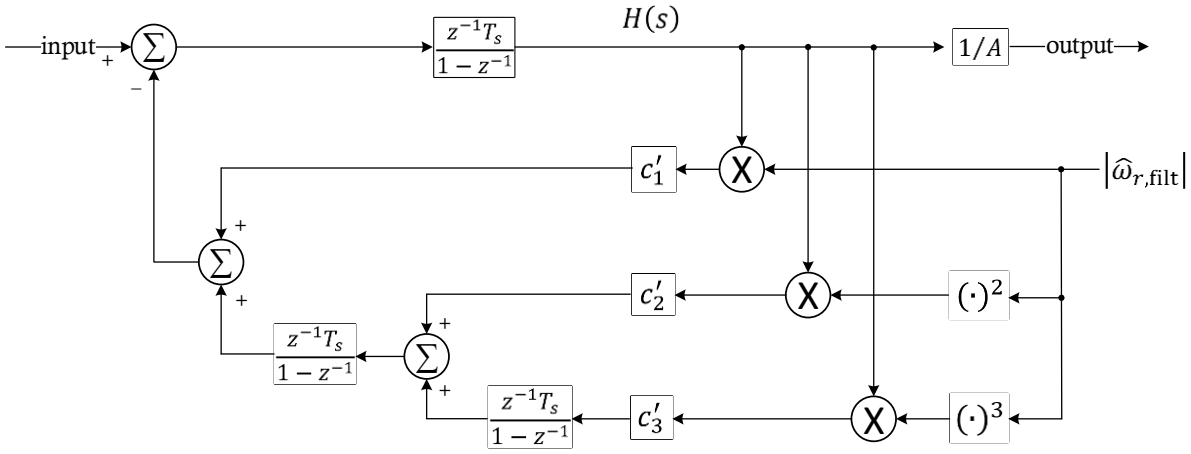

Fig. 11: Adaptive $H(s)$

Note that in the block diagram for $H(s)$, a discrete-time accumulator equivalent to $1/s$ is used since this system will be implemented in discrete-time domain inside a microcontroller:

$$G(z) = \frac{z^{-1}T_s}{1-z^{-1}} \approx \frac{1}{s} \tag{70}$$

3.5.2 Critically-Damped Biquad Low-Pass Filter

As seen in the block diagrams for RFO observers, the estimated rotor electrical frequency, ω_r, is fed back from the PLL to the adaptive flux extraction filter as well as the rotation matrix. A filter with enough attenuation at high frequencies must be applied to ω_r to reject the high frequency noise and stabilize the system. Any stabilizing low pass filter can be employed here.

Without loss of generalization, we will use a biquad filter here as described in section 6.1. This biquad filter is a critically-damped second-order low-pass-filter as follows:

$$\begin{cases} K = 1 \\ \omega_{z1} = \omega_{z2} = \infty \\ \omega_{p1} = \omega_{p2} = \omega_p \end{cases} \Rightarrow \begin{cases} a_0 = b_0 = 1 \\ a_1 = a_2 = 0 \\ b_1 = 2/\omega_p \,, b_2 = 1/\omega_p^2 \end{cases} \Rightarrow \tag{71}$$

$$\begin{cases} m_0 = \dfrac{\omega_p^2}{\omega_p^2 + 2\omega_p f_s + f_s^2} \\ m_1 = 0 \\ m_2 = 0 \end{cases} \quad \begin{cases} n_0 = 1 \\ n_1 = \dfrac{-2\left(\omega_p f_s + f_s^2\right)}{\omega_p^2 + 2\omega_p f_s + f_s^2} \\ n_2 = \dfrac{f_s^2}{\omega_p^2 + 2\omega_p f_s + f_s^2} \end{cases} \tag{72}$$

3.6 Single-Shunt Current Reconstruction

In single shunt configuration, the ADC sampling must happen according to a certain pattern that would require the duty cycles of different phase to not be in close vicinity of each other. The following figure depicts this concept.

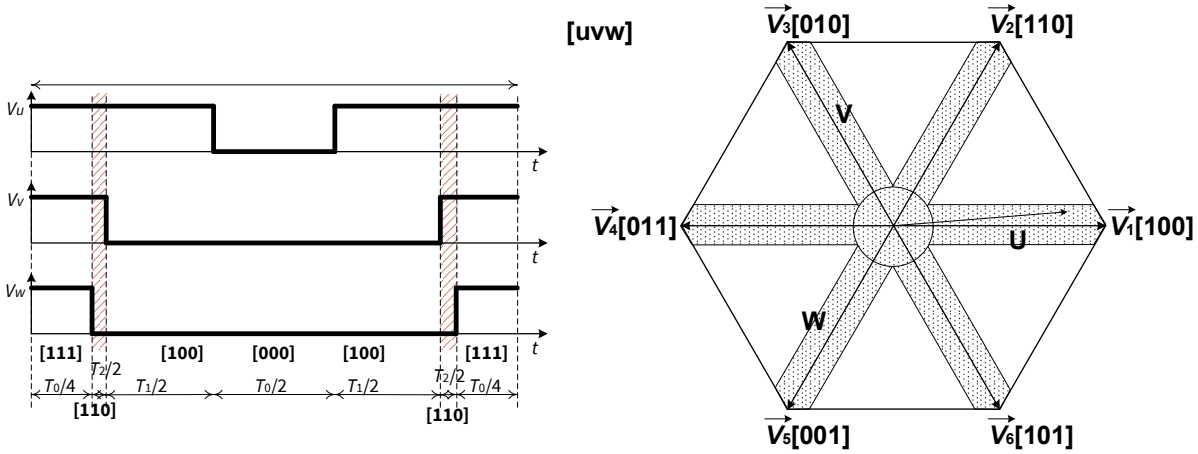

Fig. 12: Forbidden region in space-vector-modulation hexagon of single-shunt motor-drive inverters

As can be seen in Fig. 12, the reason why the shaded region in the space-vector-modulation hexagon is forbidden is that when the duty cycles of for example d_{v} and d_{w} get close to each other, there is not enough time for ADC sampling to happen (switching state 110).

3.6.1 Forbidden ranges

The shaded region in Fig. 12, corresponds to forbidden regions in the range of different variables. For example, it can correspond to the forbidden regions in duty cycles, $\{d_u^*, d_v^*, d_w^*\}$, phase-to-dc-bus voltages $\{v_{uz}^*, v_{vz}^*, v_{wz}^*\}$, phase-to-neutral voltages $\{v_{un}^*, v_{vn}^*, v_{wn}^*\}$, or voltage angle, θ. These are all equivalent representations of the same shaded region in Fig. 12.

3.6.1.1 Forbidden duty-cycle ranges

Assuming the minimum time required for ADC sampling is T_{ADC} and the switching period is T_{s}, the minimum duty cycle (e.g. duty cycle of state 110 in Fig. 12) is

$$D_{\min} = \frac{2 \times T_{\mathrm{ADC}}}{T_{\mathrm{s}}} \tag{73}$$

It can be shown that the corresponding forbidden region in the range of possible duty cycles are

$$\begin{cases} d_x \epsilon \left[\frac{1}{2}(1 + D_0 - D_{\min}), \frac{1}{2}(1 + D_0 + D_{\min})\right] \\ d_x \epsilon \left[\frac{1}{2}(1 - D_0 - D_{\min}), \frac{1}{2}(1 - D_0 + D_{\min})\right] \end{cases} \tag{74}$$

where d_x can be any of $\{d_u, d_v, d_w\}$, and D_0 can be shown to be

$$D_0 = \frac{\sqrt{3}V}{V_{\mathrm{dc}}} \sin\left(\frac{\pi}{3} - \sin^{-1}\left(\frac{D_{\min} V_{\mathrm{dc}}}{\sqrt{3}V}\right)\right) \tag{75}$$

3.6.1.2 Forbidden phase-to-dc-bus voltage ranges

It can be shown that the forbidden range of phase-to-dc-bus voltages is as follows

$$\begin{cases} v_{xz}^* \in \left[\frac{V_{\mathrm{dc}}}{2}(D_0 - D_{\min}), \frac{V_{\mathrm{dc}}}{2}(D_0 + D_{\min})\right] \\ v_{xz}^* \in \left[\frac{-V_{\mathrm{dc}}}{2}(D_0 + D_{\min}), \frac{-V_{\mathrm{dc}}}{2}(D_0 - D_{\min})\right] \end{cases} \quad (76)$$

where v_{xz}^* is any of the $\{v_{uz}^*, v_{vz}^*, v_{wz}^*\}$.

3.6.1.3 Forbidden phase-to-neutral voltage ranges

It can be shown that the forbidden range of phase-to-neutral voltages are as follows

$$\begin{cases} v_{xn}^* \in \left[\frac{V_{\mathrm{dc}}}{3}(D_0 - D_{\min}), \frac{V_{\mathrm{dc}}}{3}(D_0 + 2D_{\min})\right] \\ v_{xn}^* \in \left[\frac{-V_{\mathrm{dc}}}{3}(D_0 + 2D_{\min}), \frac{-V_{\mathrm{dc}}}{3}(D_0 - D_{\min})\right] \end{cases} \quad (77)$$

where v_{xn}^* is any of the $\{v_{un}^*, v_{vn}^*, v_{wn}^*\}$.

3.6.1.4 Forbidden voltage angle ranges

The forbidden voltage angle ranges can be shown to be

$$\theta \in \left[\frac{k\pi}{3} - \frac{\theta_{\min}}{2}, \frac{k\pi}{3} + \frac{\theta_{\min}}{2}\right], k = \{0,1,2,3,4,5\} \quad (78)$$

where

$$\theta_{\min} = 2\sin^{-1}\left(\frac{D_{\min}V_{\mathrm{dc}}}{\sqrt{3}V}\right) \quad (79)$$

As an example, for $D_{\min} = 0.05$, $V_{\mathrm{dc}} = 100\mathrm{V}$, and $V = 50\mathrm{V}$, the following figure depicts the forbidden regions in duty cycles, phase-to-dc-bus voltages, phase-to-neutral voltages, and voltage angle.

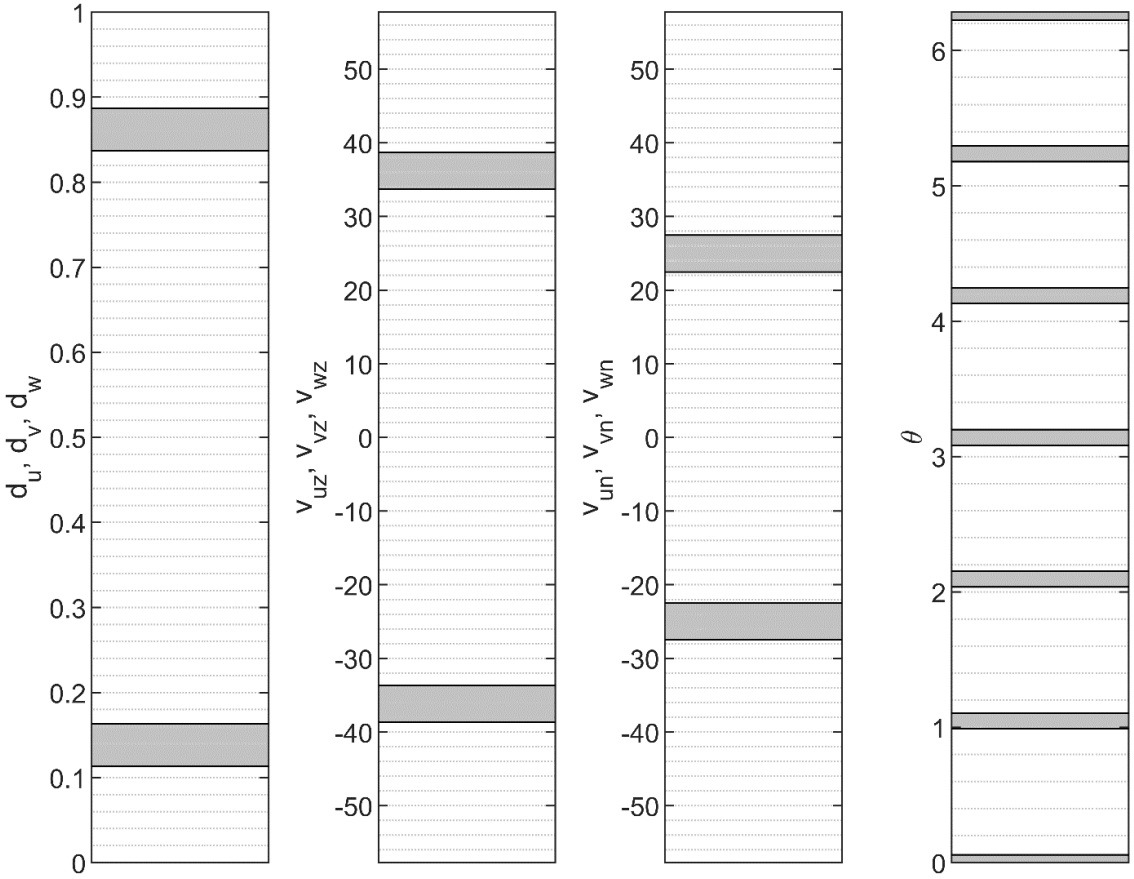

Fig. 13: Forbidden regions in duty cycles, phase-to-dc-bus voltages, phase-to-neutral voltages, and voltage angle

Without loss of generality, a hybrid nonlinear modulation method will be described in the next section that can be applied to any of the variables presented in (73)-(79) to avoid the corresponding forbidden region as seen in the Fig. 14 below.

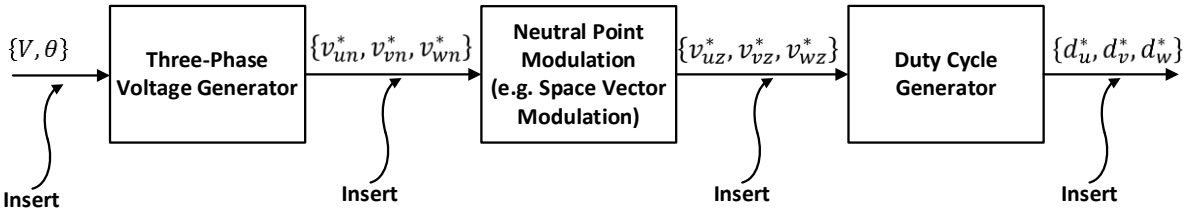

Fig. 14: The proposed solution can be inserted in any stage to avoid the forbidden operation regions

3.6.2 Hybrid Nonlinear Modulation

Without loss of generality, let us assume that the forbidden operation region in the range of possible duty cycles is to be avoided. The same method can also be used for phase-to-dc-bus voltages, phase-to-neutral voltages, and voltage angle. Fig. 15 depicts the proposed hybrid nonlinear modulator that can achieve this goal.

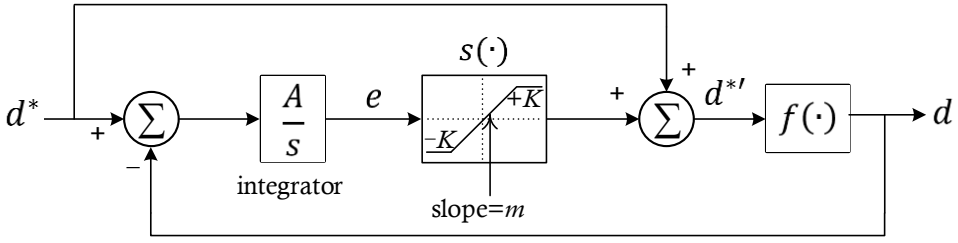

Fig. 15. The hybrid nonlinear duty-cycle modulator

In Fig. 15, $f(\cdot)$ is a nonlinear function that removes the forbidden region from the range of possible duty cycles. For example, $D_{\min} = 0.10$, $V_{\mathrm{dc}} = 100\mathrm{V}$, and $V = 50\mathrm{V}$, $f(\cdot)$ will be as follows:

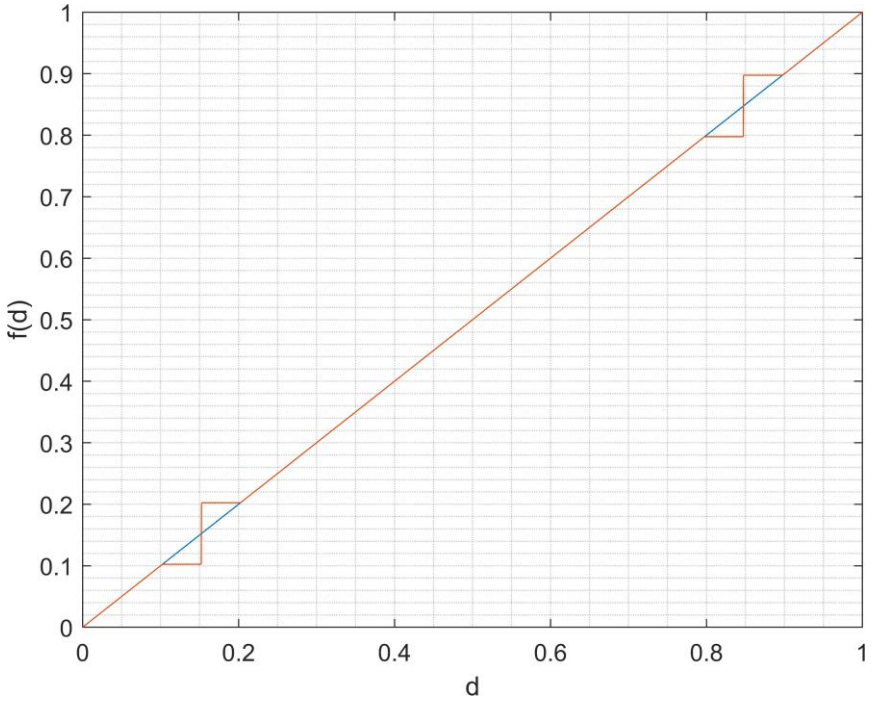

Fig. 16: A sample $f(\cdot)$ that removes the forbidden region from the range of possible duty cycles

Similarly, let us define function $g(\cdot)$ as the difference between the commanded and actual duty cycle values:

$$\begin{cases} g(d) = f(d) - d \\ -\frac{D_{\min}}{2} \le g(d) < \frac{D_{\min}}{2} \end{cases} \tag{80}$$

The $g(\cdot)$ corresponding to $f(\cdot)$ in Fig. 16 is shown in the figure bellow.

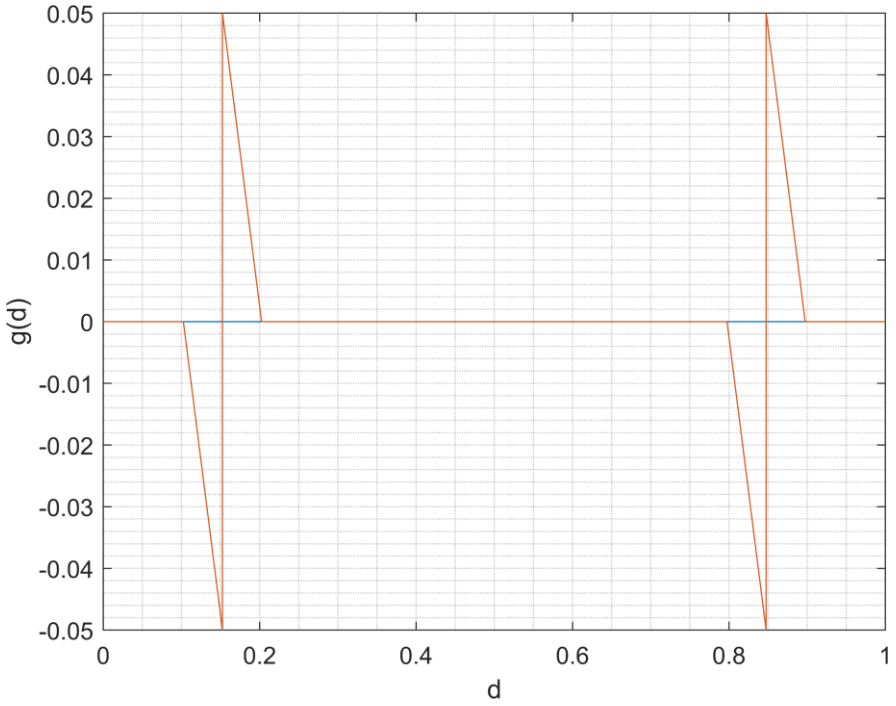

Fig. 17. The $g(\cdot)$ corresponding to $f(\cdot)$ in Fig. 16

Although extremely simple and also nonlinear, this hybrid modulator is very robust and stable in all operating conditions.

3.6.2.1 Lyapunov Stability Analysis

Here we will use nonlinear control theory to analyze the behavior of this system.

We use Lyapunov stability theorem. Let us define the following Lyapunov function.

$$V = \frac{e^2}{2} \tag{81}$$

From this definition, we can easily see that V is a positive definite function:

$$\begin{cases} \forall e \neq 0: & V > 0 \\ e = 0: & V = 0 \end{cases} \tag{82}$$

Also, V is radially unbounded as expressed below

$$e \to \infty \Leftrightarrow V \to \infty \tag{83}$$

The derivative of V with respect to time can be written as

$$\dot{V} = e\dot{e} \tag{84}$$

where $\dot{e}$ can be expanded according to the block diagram in Fig. 15 as

$$\dot{e} = A(d^* - d) = A\big(d^* - f(d^{*\prime})\big) = A\Big(d^* - f\big(d^* + s(e)\big)\Big) \tag{85}$$

Now, we express $f(\cdot)$ in terms of $g(\cdot)$ to obtain

$$\dot{e} = A\Big(d^* - \big(d^* + s(e) + g\big(d^* + s(e)\big)\big)\Big) = -A\Big(s(e) + g\big(d^* + s(e)\big)\Big) \tag{86}$$

First, let's assume that error, e, is greater than ($|e| \geq K/m$). Therefore, $s(e) = K\mathrm{sgn}(e)$, and

$$\dot{V} = e\dot{e} = -Ae\Big(K\mathrm{sgn}(e) + g\big(d^* + K\mathrm{sgn}(e)\big)\Big) = -A|e|\Big(K + \underbrace{\mathrm{sgn}(e)g\big(d + K\mathrm{sgn}(e)\big)}_{x}\Big) \tag{87}$$

The variable x always has a lower limit which is

$$-\frac{D_{\min}}{2} \leq x \tag{88}$$

Therefore, $\dot{V}$ has an upper limit as follows

$$\dot{V} = -A|e|(K + x) \leq -A|e|\left(K - \frac{D_{\min}}{2}\right) \tag{89}$$

By inspecting (89), we can conclude that if K is large enough, $\dot{V}$ is a always negative function i.e.

$$\forall K > \frac{D_{\min}}{2} \Leftrightarrow \dot{V} < 0 \tag{90}$$

This means that by choosing proper K, the error will decrease as time goes by until it reaches K/m. Once the error hits K/m, the behavior of the system depends on whether we are in the forbidden region or not. If the system is not in the forbidden region,

$$g\big(d + K\mathrm{sgn}(e)\big) = 0 \tag{91}$$

Therefore, (86) can be written as

$$\dot{e} = -A\left(\underbrace{s(e)}_{me} + \underbrace{g\big(d^* + s(e)\big)}_{0}\right) = -Ame \tag{92}$$

and

$$\dot{V} = e\dot{e} = -Ame^2 \Rightarrow \dot{V} < 0 \tag{93}$$

What (93) proves is that the error, e, will decrease and asymptotically approach zero as long as the system is not in the forbidden region. In fact, the whole system becomes a linear regulator and is no longer nonlinear in this condition. It can be shown that the linear system will have a first-degree response with cross-over frequency of $\omega_0 = Am$.

If the system is inside the forbidden region, however, it will become a nonlinear switching regulator. This nonlinear switching regulator switches back and forth between the upper and lower edges of the forbidden region in order to create an average that is inside the forbidden region. Therefore, it can be thought of as blue noise injector that will also regulates the output in average sense even if the input command is in the forbidden region.

3.6.2.2 Typical Waveforms

The following figures show example waveforms for $D_{\min} = 0.05$, $V_{\mathrm{dc}} = 100\mathrm{V}$, and $V = 50\mathrm{V}$.

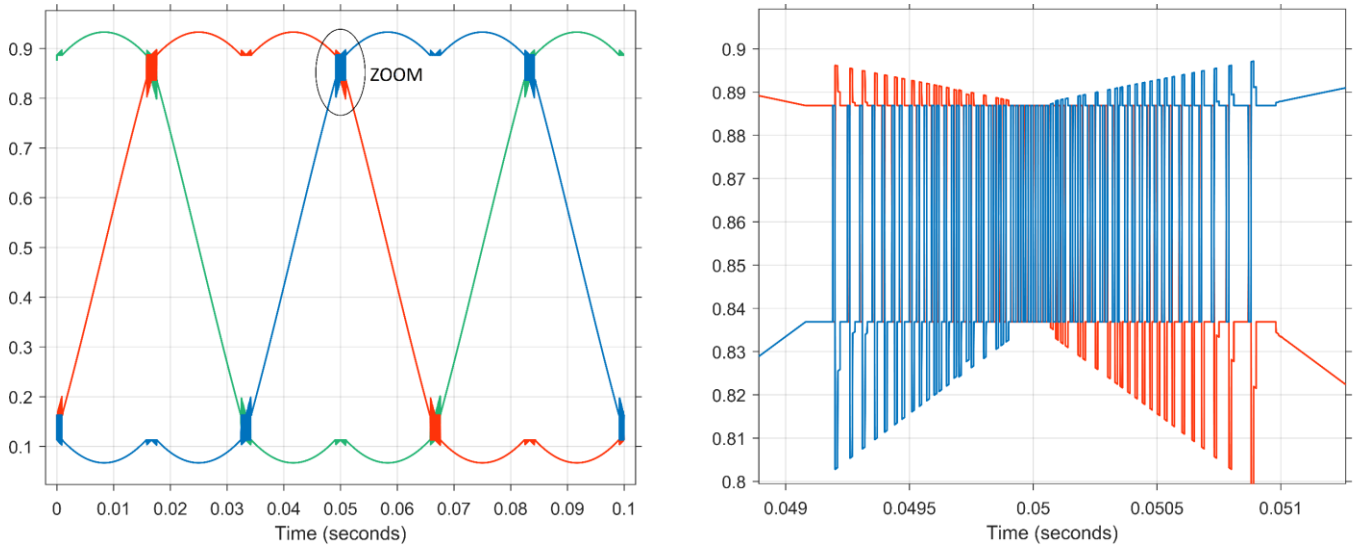

Fig. 18: Example modulated duty cycle waveforms, $\{d_u, d_v, d_w\}$

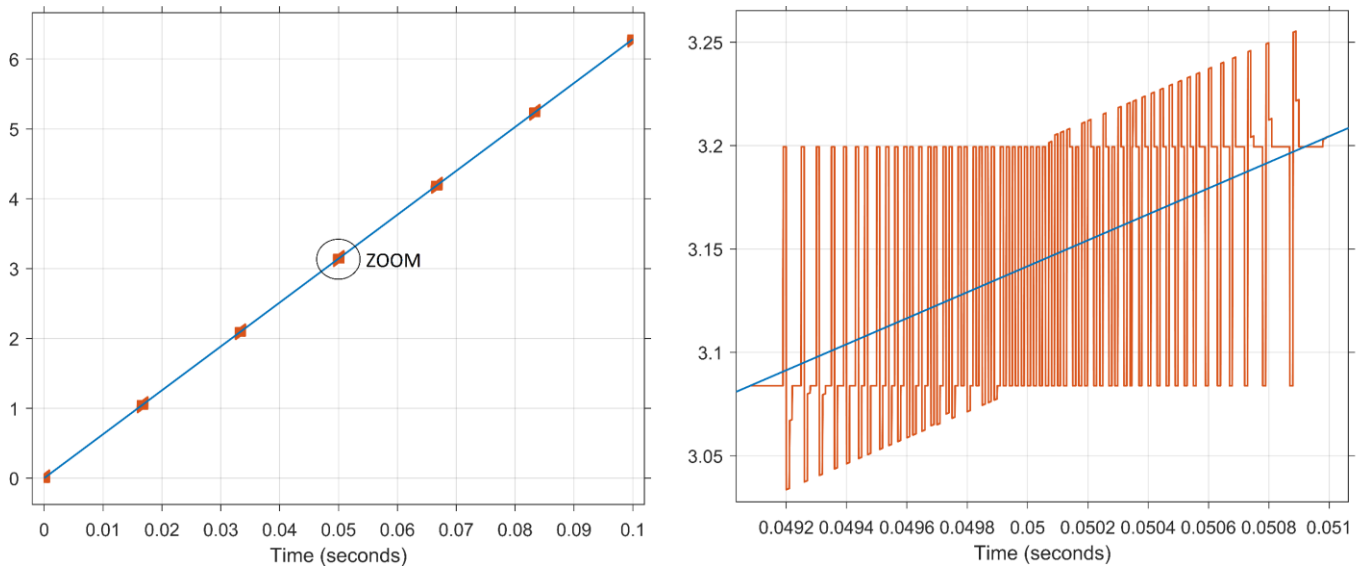

Fig. 19: Example modulated angle waveforms, θ

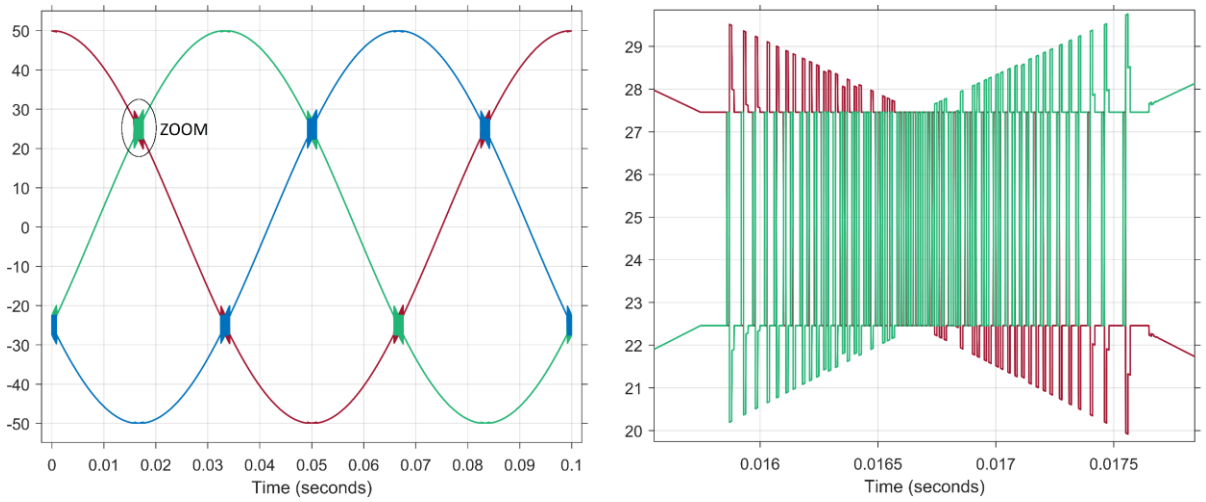

Fig. 20: Example modulated phase-to-neutral waveforms, $\{v_{un}, v_{vn}, v_{wn}\}$

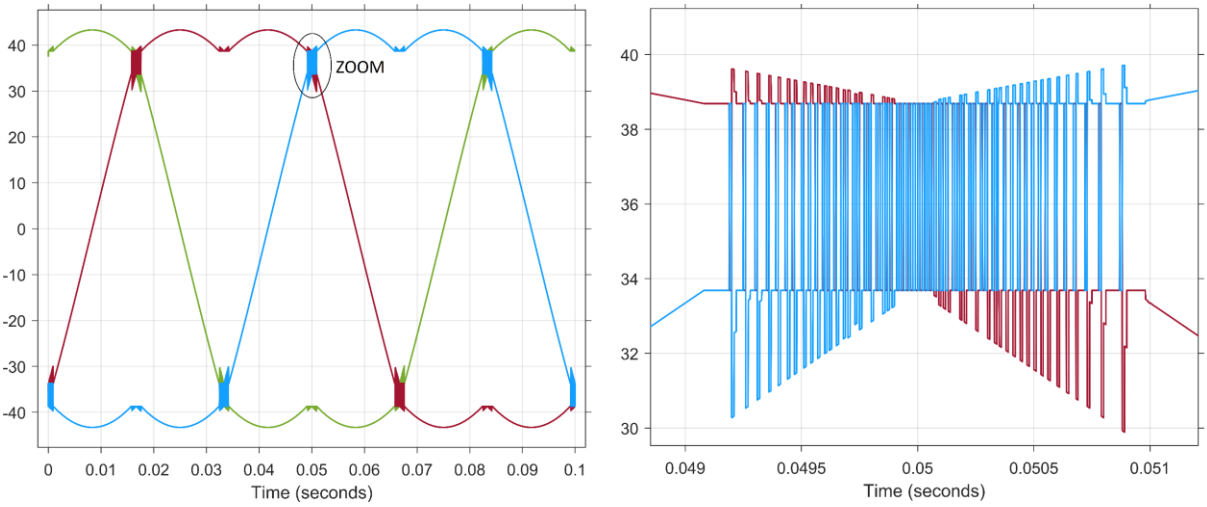

Fig. 21: Example modulated phase-to-dc-bus waveforms, $\{v_{uz}, v_{vz}, v_{wz}\}$

3.6.3 Current Reconstruction

In single shunt configuration, the current flowing through the shunt is always zero when all lower switches are on (000) or all upper switches are on (111). Therefore, the current sampling should happen at specific times during the switching period as opposed to the three-shunt configuration where current sampling always happens at switching state 000.

Based on the switching sector (S0…S5 as seen Fig. 22), one of the duty cycles is greater than the other two duty cycles. Let us call that d_x. Similarly, let us call the next greatest duty cycle and the smallest duty cycle d_y and d_z, respectively. Table 1 shows how to determine XYZ based on UVW and the switching sector. Similarly, Table 2 shows how to determine UVW based on XYZ and the switching sector. Therefore, at each switching sector, there is a unique mapping from UVW to XYZ and vice versa. The “XYZ code” and “UVW code” shown in Table 1 and Table 2 are used in the firmware to encode this one-to-one mapping.

Sector	0	1	2	3	4	5
X	U^0	V^1	V^1	W^2	W^2	U^0
Y	V^1	U^0	W^2	V^1	U^0	W^2
Z	W^2	W^2	U^0	U^0	V^1	V^1
XYZ code	012	102	120	210	201	021

Table 1: Determining XYZ based on UVW and switching sector

Sector	0	1	2	3	4	5
U	X^0	Y^1	Z^2	Z^2	Y^1	X^0
V	Y^1	X^0	X^0	Y^1	Z^2	Z^2
W	Z^2	Z^2	Y^1	X^0	X^0	Y^1
UVW code	012	102	201	210	120	021

Table 2: Determining UVW based on XYZ and switching sector

Fig. 22 depicts the typical PWM waveforms corresponding to $\{d_x, d_y, d_z\}$.

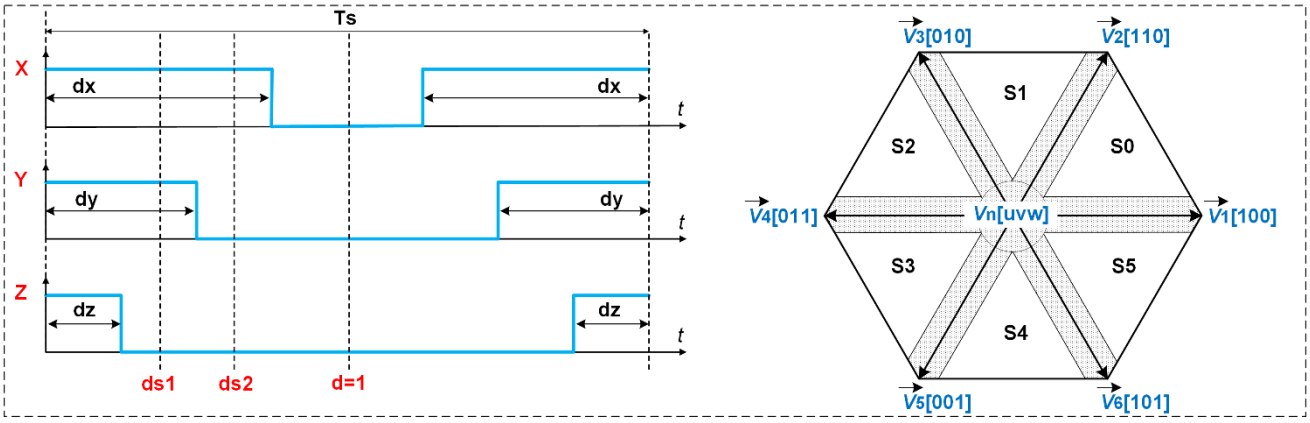

Fig. 22: SVM sectors, $\{d_x, d_y, d_z\}$ definition ($d_x > d_y > d_z$), and sample duty cycles $\{d_{s1}, d_{s2}\}$

As seen in Fig. 22, the current samples should be taken at the following duty cycles:

$$\begin{cases} d_{s1} = \dfrac{d_y + d_z}{2} \\ d_{s2} = \dfrac{d_x + d_y}{2} \end{cases} \tag{94}$$

It should be noted that $d = 1$ corresponds to $T_s/2$ since center-asymmetric triangle modulation is used here. Let us call the current samples i_{s1} and i_{s2} (corresponding to d_{s1} and d_{s2}). The direction of the current is defined as going from the motor to the dc ground here. Based on this definition,

$$\begin{cases} i_{s1} = i_x + i_y = -i_z \\ i_{s2} = -i_y - i_z = i_x \end{cases} \tag{95}$$

Hence, the currents $\{i_x, i_y, i_z\}$ can be reconstructed from measured $\{i_{s1}, i_{s2}\}$ as

$$\begin{cases} i_x = i_{s2} \\ i_y = i_{s1} - i_{s2} \\ i_z = -i_{s1} \end{cases} \tag{96}$$

Once $\{i_x, i_y, i_z\}$ are found, $\{i_u, i_v, i_w\}$ can be determined using Table 1.

The current reconstruction procedure can be summarized as follows:

- Having $\{d_u, d_v, d_w\}$ as the outputs of the hybrid modulator, determine $\{d_x, d_y, d_z\}$ based on Table 1.
- Calculate sampling duty cycles for the next switching period, $\{d_{s1}, d_{s2}\}$, based on $\{d_x, d_y, d_z\}$ and (94).
- Calculate $\{i_x, i_y, i_z\}$ for the previous switching period based on sampled currents $\{i_{s1}, i_{s2}\}$ and (96).
- Having $\{i_x, i_y, i_z\}$, determine $\{i_u, i_v, i_w\}$ based on Table 2.

3.7 Position Controller

Position control is supported in RFO control type. Rotor position (θ_p) that is estimated from encoder and position command (θ^*_{cmd}) are input to this control block. Output of the control block sets the speed command (ω^*_m) (input to the speed controller)

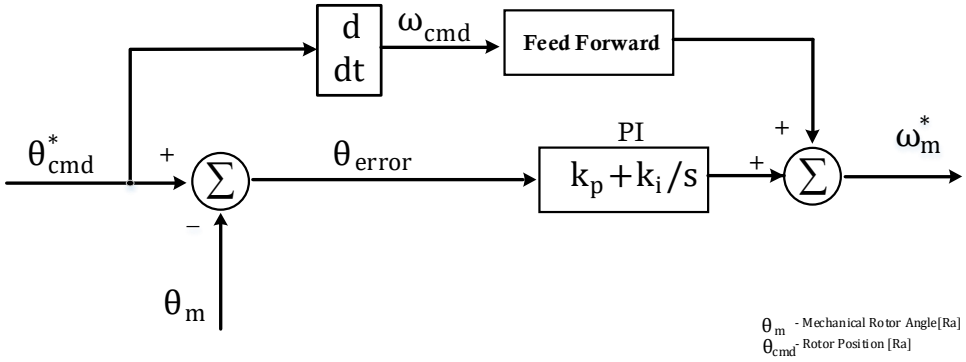

Fig. 23: Position Controller Block Diagram.

The “plant” transfer function for position controller can be approximated as follows.

$$\omega_m \propto \frac{d\theta}{dt} \tag{97}$$

The plant model can be approximate as an integrator. Then, the loop-gain transfer function would be

$$G(s) = \left(k_p + \frac{k_i}{s}\right) \cdot \frac{1}{s} \tag{98}$$

The closed-loop transfer function can be determined based on the loop-gain as

$$H(s) = \frac{G(s)}{1 + G(s)} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} \tag{99}$$

Let us place the poles of the closed-loop transfer function, $H(s)$, at

$$\begin{cases} \omega_{p,1} = \omega_0 \\ \omega_{p,2} = (r-1)\omega_0 \end{cases} \tag{100}$$

where r is the pole separation ratio ($r > 1$). Hence,

$$s^2 + k_p s + k_i = s^2 + (r\omega_0)s + (r-1)\omega_0^2 \tag{101}$$

From (168), PI controller’s parameters can be computed as

$$\begin{cases} k_p = r\omega_0 \\ k_i = (r-1)\omega_0^2 \end{cases} \tag{102}$$

based on the desired pole separation ratio, r, and bandwidth, ω_0.

3.8 Experimental Results

To demonstrate the performance of RFO control, a 1.5kW 8-poles 48V motor is tested. The control mode is set to RFO sensorless speed control. The commanded electrical speed is 750 rad/sec-elec and the load is set at 1.5 Nm. Fig. 24 shows the three-phase current waveforms. As seen here, the current waveforms are regulated and balanced. Fig. 25 shows variables’ live watch snapshot from Ozone debugger. As seen in Fig. 25, feedbacks follow command values (speed and currents).

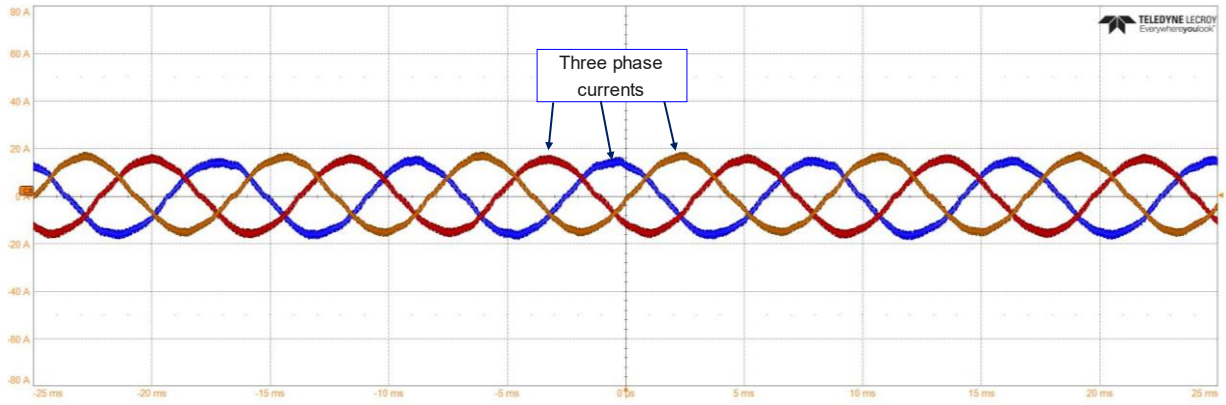

Fig. 24: Three phase current waveforms with RFO sensorless speed mode control with speed commanded at 750 rad/sec-elec and load set at 1.5 Nm.

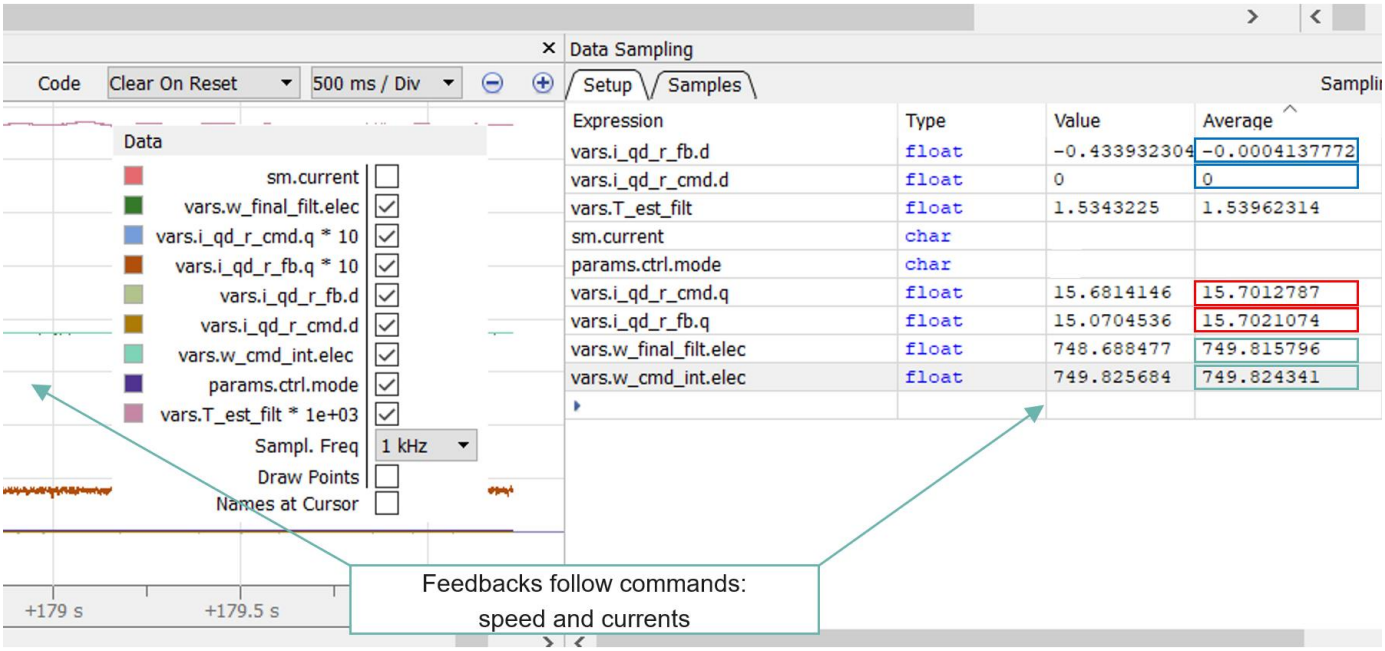

Fig. 25: Variables live watch snapshot from Ozone debugger: feedbacks follow commands.

4 Stator Field Oriented Control

SFO is a control method that is based on motor's stator reference frame rather than rotor reference frame. The main advantage of SFO is that the torque and flux can be directly controlled. In RFO torque and flux are indirectly controlled via i_q and i_d. This is not optimal for applications when direct torque control is needed such as e-bike. Although torque is proportional to current, but it also depends on rotor flux which can change significantly with temperature. This is where direct torque control with SFO can be advantageous. Also, by directly controlling the flux, better dynamic performance in the field weakening region (above base speed) can be obtained.

The relationship between SFO and RFO is illustrated in Fig. 26. SFO frame is ahead of RFO frame with a phase difference equal to δ as shown in Fig. 26. This is called load angle, and is a function of load current. Higher load current results in higher load angle. That means to get more torque larger load angle needs to be commanded. This is the main basis for the SFO control as shown in Fig. 27. To see how the control is implemented let's start with expressing the torque as follows:

$$T = \frac{3P}{4}(\lambda_d^s i_q^s - \lambda_q^s i_d^s) \quad (103)$$

Because in SFO, the d-axis of reference frame is intentionally being set to be aligned with the stator flux vector λ_s^r one can write:

$$\lambda_s^r = \sqrt{\lambda_d^{r\,2} + \lambda_q^{r\,2}}\ , \lambda_q^s = 0,\ \lambda_d^s = \lambda_s^r \quad (104)$$

Then, from (103) and (104) torque can be expressed as

$$T = \frac{3P}{4}(\lambda_d^s i_q^s) \quad (105)$$

In SFO the command can be speed or torque. To directly control the torque, the error between its estimated value and commanded value is compensated via a PI controller (section 4.5) and the output will be used as the reference for the load angle δ^*. The load angle $\hat{\delta}$ is also estimated by adaptive observer block (section 4.8). The compensated error will be given to an adaptive PI controller (section 4.7) which will generate v_q^{s*} reference voltage.

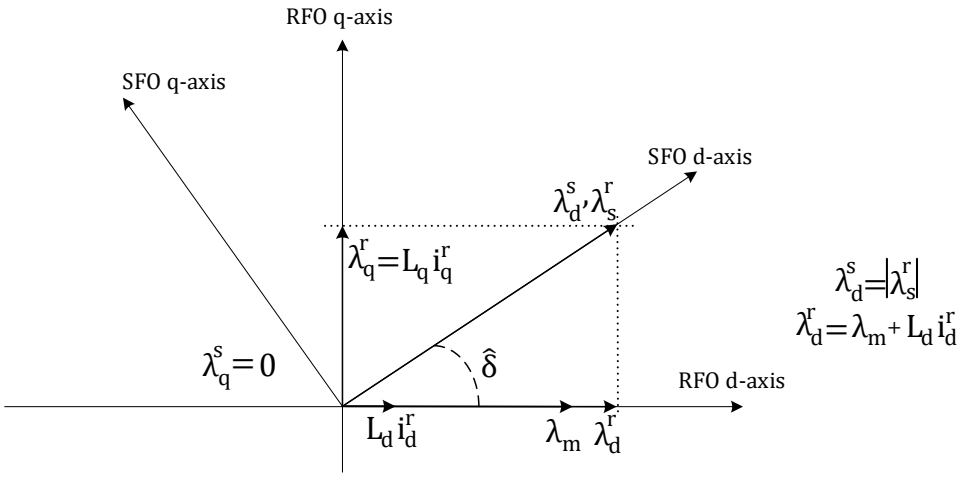

Fig. 26: Flux illustration in rotor and stator reference frame

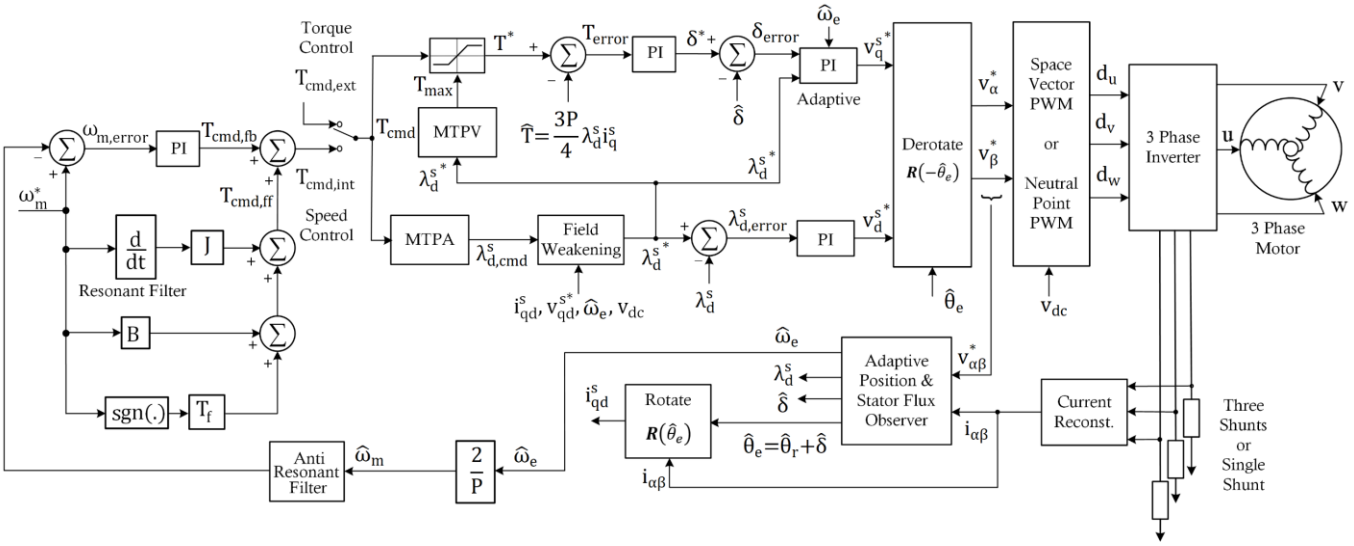

Fig. 27: SFO control block diagram

As seen in (105) torque is also influenced by λ_d^s . The flux controller is also part of the SFO control block diagram as shown in Fig. 27. To control the flux, the reference λ_d^{s*} is generated by MTPA (section 4.2) and field weakening (section 4.3) blocks. Finally, the flux controller (section 4.6) generates the v_d^{s*} reference voltage.

4.1 Speed Controller

This section will cover the SFO speed controller, which is fundamentally similar to the speed controller in RFO as described in section 3.1. The main difference between SFO and RFO speed controllers is that with RFO the speed controller's output is current reference while with SFO, it is torque reference. This means the parameters of speed controller will have a different scale factor than RFO.

To derive the value of proportional, integral, and feedforward terms of speed controller, the speed loop block diagram along with a simple model of motor and mechanical load will be used as shown in Fig. 28. Pole-zero cancellation technique is used to find the PI controller integral and proportional gain. By having

$$\frac{k_i}{k_p} = \frac{B}{J} \tag{106}$$

the controller zero will cancel the mechanical load's pole. The PI controller coefficients are also proportional to speed loop bandwidth as shown below:

$$k_i \propto B\omega_{BW} \tag{107}$$

$$k_p \propto J\omega_{BW} \tag{108}$$

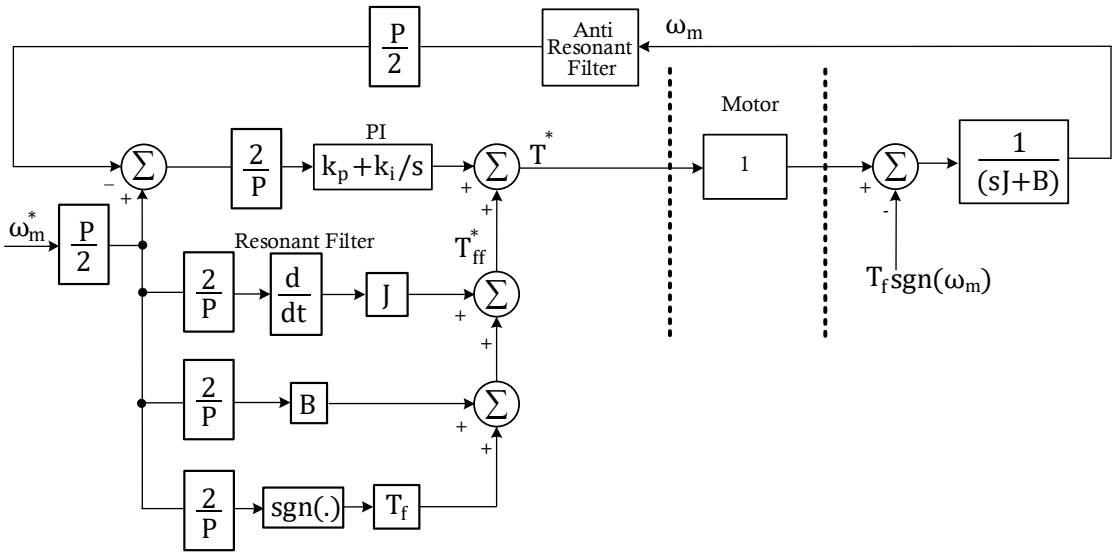

Fig. 28: SFO speed loop block diagram before rescaling the parameters

The proportional and integral gains are ultimately determined after being rescaled to account for the number of poles P. Fig. 29 illustrates the speed loop implemented for SFO in the firmware.

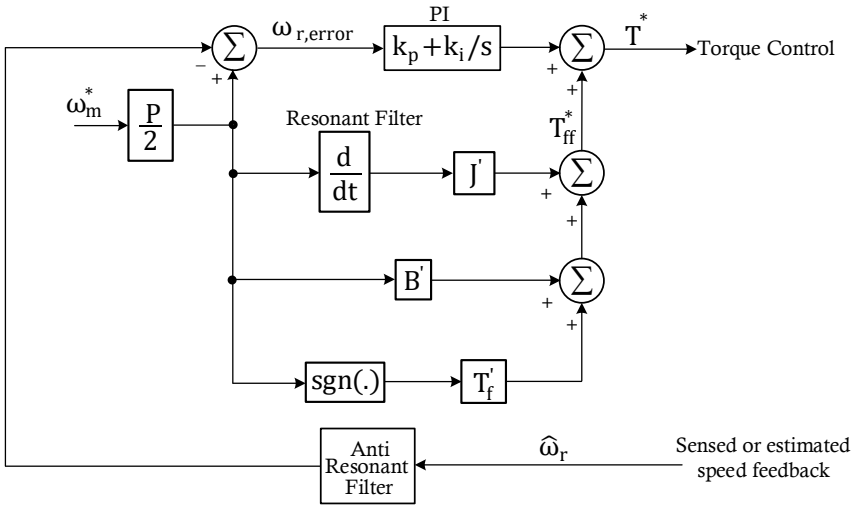

Fig. 29: SFO speed loop block diagram as implemented in the firmware

The speed contoller parameters after rescaling are as follows:

$$k_i = \left(\frac{2}{P}\right) B\omega_{BW} \quad (109)$$

$$k_p = \left(\frac{2}{P}\right) J\omega_{BW} \quad (110)$$

The feedforward terms have been also updated as depicted in the following manner:

$$B' = \left(\frac{2}{P}\right) B \quad (111)$$

$$J' = \left(\frac{2}{P}\right) J \quad (112)$$

$$T_f' = T_f \quad (113)$$

The feedforward terms in the speed loop are affected by both mechanical load and number of poles. These three feedforward terms play a role in enhancing the dynamic performance of the speed loop. The inertia term utilizes a second-order resonant filter to estimate the acceleration, aiming to mitigate the potential impact of noise that could arise if a direct derivation method had been employed.

4.2 MTPA

In order to maximize the system efficiency, defined as the ratio of input electrical power to the output mechanical power delivered at the shaft of the motor, it is important to minimize the resistive power losses. MTPA aims to achieve this goal by controlling the motor in such a way that the minimum current required is applied to obtain a certain amount of torque.

In SFO, torque is controlled by the torque controller. However, for a given torque command, there is one degree of freedom in choosing the stator flux magnitude command, λ_d^s, as well. The problem of MTPA in SFO, therefore, can be stated as finding the optimum λ_d^s command for a given T command in order to minimize i_s.

It is simpler to solve this optimization problem in terms of the projections of the flux vector onto the rotor reference frame (λ_q^r, λ_d^r) first and then calculate the magnitude of the flux vector in stator reference frame, λ_d^s, as follows.

$$\begin{cases} T^2 = T^2(\lambda_q^r, \lambda_d^r) : \text{constraint curve} \\ \min\{\, i_s^2(\lambda_q^r, \lambda_d^r) \,\} : \text{cost function} \end{cases} \tag{114}$$

$$\begin{cases} \lambda_d^s = \sqrt{(\lambda_q^r)^2 + (\lambda_d^r)^2} \\ \lambda_q^s = 0 \end{cases} \tag{115}$$

As can be seen in (114), this is a classical multivariable optimization problem under a certain constraint. We will use the well-known Lagrange multiplier method to solve this problem.

4.2.1 Lagrange Multiplier Method

Lagrange multiple method can be formulated as optimizing a certain cost function (f) under the constraint of a certain constraint curve (g):

$$\begin{cases} g(x,y) = 0 : \text{constraint curve} \\ \min\{\, f(x,y) \,\} : \text{cost function} \end{cases} \tag{116}$$

It must be noted that this method can be used for n-dimensional problems but only two dimensions are required here as presented in (114).

It can be shown that the solution to (116) can be obtained by adding additional auxiliary variable, k, to obtain three equations and three variables to solve for, as follows:

$$\begin{cases} g(x,y) = 0 \\ \boldsymbol{\nabla} f(x,y) = k \cdot \boldsymbol{\nabla} \mathrm{g}(x,y) \end{cases} \tag{117}$$

Fig. 30 illustrates the reason behind (117). The blue contours shown in Fig. 30 represent certain constant values of $f(x,y)$ in 2D. The gradient $\boldsymbol{\nabla} f(x,y)$ shows the direction of maximum increase in $f(x,y)$ at any point in 2D. Hence, these gradient vectors are always perpendicular to the contours. Similarly, the gradient $\boldsymbol{\nabla} g(x,y)$ is always perpendicular to $g(x,y)$. One must move in line with the direction of gradients $\boldsymbol{\nabla} f(x,y)$ to reach the maximum or minimum of $f(x,y)$. However, the direction of movement is also constrained by the constraint curve, $g(x,y)$. The optimum solution is always achieved when $g(x,y)$ curve is tangent to the contours of $f(x,y)$, i.e., when $\boldsymbol{\nabla} f(x,y)$ and $\boldsymbol{\nabla} g(x,y)$ gradients are in the same direction, i.e. $\boldsymbol{\nabla} f(x,y) = k \cdot \boldsymbol{\nabla} \mathrm{g}(x,y)$.

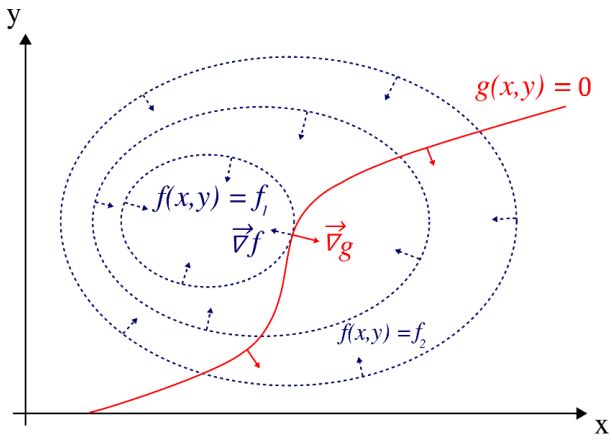

Fig. 30: Constraint curve, cost function, and gradients in Lagrange multiplier method

Equation (117) can be written as

$$\begin{cases} g(x,y) = 0 \\ \dfrac{\partial f}{\partial x} = k \cdot \dfrac{\partial g}{\partial x} \\ \dfrac{\partial f}{\partial y} = k \cdot \dfrac{\partial g}{\partial y} \end{cases} \tag{118}$$

in order to remove k from the list of the variables to solve for as follows:

$$\begin{cases} g(x,y) = 0 \\ \left(\dfrac{\partial f}{\partial x}\right) \Big/ \left(\dfrac{\partial g}{\partial x}\right) = \left(\dfrac{\partial f}{\partial y}\right) \Big/ \left(\dfrac{\partial g}{\partial y}\right) \end{cases} \tag{119}$$

Equation (119) describes a convenient system of two equations and two unknowns to solve for in order to find the solution of the MTPA optimization problem.

4.2.2 Optimum Flux Vector Curve

To simplify the computations, we express the torque in terms of rotor reference frame variables first, then solve the optimization problem, and finally calculate the stator reference frame variables based on the equivalent rotor reference frame variables. Torque in RFO can be written as

$$T = \frac{3}{2} \cdot \frac{P}{2} \left(\lambda_d^r i_q^r - \lambda_q^r i_d^r \right) \tag{120}$$

where

$$\begin{cases} \lambda_q^r = L_q i_q^r \\ \lambda_d^r = L_d i_d^r + \lambda_m \end{cases} \tag{121}$$

Combining (121) and (120) and eliminating the currents, i_q^r and i_d^r, we can obtain

$$T = \frac{3}{2} \cdot \frac{P}{2} \left(\lambda_d^r \frac{\lambda_q^r}{L_q} - \lambda_q^r \frac{\lambda_d^r - \lambda_m}{L_d} \right) \Rightarrow \tag{122}$$

$$T^2 = \left(\frac{3P}{4}\right)^2 \left(\frac{\lambda_q^r}{L_d}\right)^2 \left(\lambda_m - \frac{\xi - 1}{\xi} \lambda_d^r\right)^2$$

where ξ is the saliency ratio defined as

$$\xi = \frac{L_q}{L_d} \geq 1 \tag{123}$$

Equation (122) describes the constraint curve of the optimization problem stated in (114). Similarly, the cost function in (114) can be expressed in terms of λ_q^r and λ_d^r as

$$i_s^2 = \left(i_q^r\right)^2 + \left(i_d^r\right)^2 = \left(\frac{\lambda_q^r}{L_q}\right)^2 + \left(\frac{\lambda_d^r - \lambda_m}{L_d}\right)^2 = \tag{124}$$

$$= \frac{1}{L_d^2}\left((\lambda_d^r - \lambda_m)^2 + \left(\frac{\lambda_q^r}{\xi}\right)^2\right)$$

Next, we can compute the gradients of the constraint curve (122) and the cost function (124) as follows:

$$\begin{cases} \dfrac{\partial T^2}{\partial \lambda_q^r} = \left(\dfrac{3P}{4}\right)^2 \left(\dfrac{2\lambda_q^r}{L_d^2}\right)\left(\lambda_m - \dfrac{\xi - 1}{\xi}\lambda_d^r\right)^2 \\ \dfrac{\partial T^2}{\partial \lambda_d^r} = \left(\dfrac{3P}{4}\right)^2 \left(\dfrac{\lambda_q^r}{L_d}\right)^2 \cdot 2\left(\dfrac{1-\xi}{\xi}\right)\left(\lambda_m - \dfrac{\xi - 1}{\xi}\lambda_d^r\right) \end{cases} \tag{125}$$

$$\begin{cases} \dfrac{\partial i_s^2}{\partial \lambda_q^r} = \dfrac{2\lambda_q^r}{L_d^2 \xi^2} \\ \dfrac{\partial i_s^2}{\partial \lambda_d^r} = \dfrac{2(\lambda_d^r - \lambda_m)}{L_d^2} \end{cases} \tag{126}$$

In addition, the gradient proportionality equation in (119) can be written as

$$\left(\frac{\partial T^2}{\partial \lambda_q^r}\right) \Big/ \left(\frac{\partial i_s^2}{\partial \lambda_q^r}\right) = \left(\frac{\partial T^2}{\partial \lambda_d^r}\right) \Big/ \left(\frac{\partial i_s^2}{\partial \lambda_d^r}\right) \tag{127}$$

which in combination with (125) and (126) results in

$$\left(\lambda_q^r\right)^2 = \left(\frac{\xi^3}{1-\xi}\right)(\lambda_d^r - \lambda_m)\left(\lambda_m - \frac{\xi - 1}{\xi}\lambda_d^r\right) \tag{128}$$

By substituting $\left(\lambda_q^r\right)^2$ from (128) in (122), a compact one-variable equation can be formulated to solve the MTPA optimization problem:

$$T^2(1-\xi) = \left(\frac{3P}{4}\right)^2 \left(\frac{\xi^3}{L_d^2}\right)(\lambda_d^r - \lambda_m)\left(\lambda_m - \frac{\xi - 1}{\xi}\lambda_d^r\right)^3 \tag{129}$$

The compact MTPA solution in (129) implies that for a motor with given motor parameters and a given torque command T, there is only one optimum λ_d^r to achieve MTPA. Validity of (129) can be verified by considering a SPM where $L_q = L_d$ i.e. $\xi = 1$:

$$\xi = 1 \Rightarrow 0 = \left(\frac{3P}{4}\right)^2 \left(\frac{1}{L_d^2}\right)(\lambda_d^r - \lambda_m)\lambda_m^3 \Rightarrow \tag{130}$$

$$\lambda_d^r = \lambda_m \Rightarrow i_d^r = 0$$

As can be seen in (130), for SPMs, the solution to (129) implies that $i_d^r = 0$ which is a well know criterion for achieving MTPA in SPMs.

Fig. 31 depicts optimum flux vector components in RFO and SFO based on the torque command for a sample IPM.

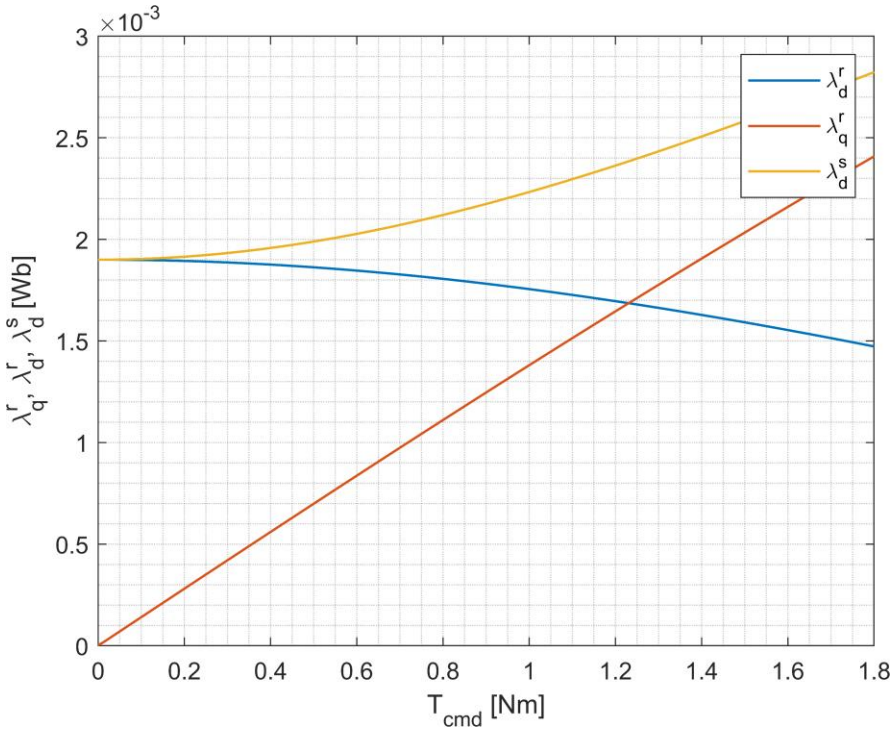

Fig. 31: Sample optimum-flux vector for a given torque command: $L_q = 28\mu\text{H}$, $L_d = 23\mu\text{H}$, $\lambda_m = 1.9\text{mWb}$, $T_{\max} = 1.8\text{Nm}$, $P = 14$

In order to obtain each point on the curves in Fig. 31, the fourth-order polynomial described in (129) is solved to first obtain λ_d^r, and then calculate λ_q^r and λ_d^s based on (128) and (115), respectively. Solving the fourth-order polynomial in (129) for each point of the optimum flux curves is not practical in real-time motor control applications that run on a microcontroller with time critical tasks that need to be periodically serviced. Next section presents a computationally efficient method to tackle this problem without having to solve a fourth-order polynomial.

4.2.3 Iterative Computation of Optimum-Flux-Vector Look Up Table

As mentioned in the previous section, solving the fourth-order polynomial in (129) for each point of the optimum flux curves is not practical in real-time motor control applications that run on a microcontroller with time critical tasks that need to be periodically serviced. Here, we use an iterative computation method to initialize a look up table based on the motor parameters at startup. Initializing this look up table does not require solving for the roots of a fourth order polynomial. Only multiplication, division, and square root are used in the computation process. All of these math operations are atomic hardware instructions for ARM microcontrollers that possess a floating-point unit.

At each step of the computation process, the next torque command T and resulting optimum stator flux magnitude λ_d^s are calculated based on the previous values in the look up table:

$$\begin{cases} T[n+1] = T[n] + \Delta T \\ \lambda_d^s[n+1] = \lambda_d^s[n] + \Delta\lambda_d^s[n] \end{cases} \tag{131}$$

The compact MTPA solution in (129) can be rewritten as

$$\underbrace{-\left(\frac{3P}{4}\right)^2\left(\frac{\xi-1}{L_d}\right)^2}_{k_1}(\lambda_d^r-\lambda_m)\left(\lambda_d^r-\underbrace{\frac{\xi}{\xi-1}}_{k_2}\lambda_m\right)^3+T^2=0 \tag{132}$$

In other words,

$$f(\lambda_d^r, T) = k_1(\lambda_d^r-\lambda_m)(\lambda_d^r-k_2\lambda_m)^3+T^2=0. \tag{133}$$

Now, we can apply a small change in the torque command ΔT and approximate the resulting $\Delta\lambda_d^r$:

$$\frac{\partial f(\lambda_d^r,T)}{\partial \lambda_d^r}\Delta\lambda_d^r + \frac{\partial f(\lambda_d^r,T)}{\partial T}\Delta T = 0 \tag{134}$$

Substituting $f(\lambda_d^r,T)$ from (133) to (134), we can obtain

$$k_1(\lambda_d^r - k_2\lambda_m)^2\big((\lambda_d^r - k_2\lambda_m) + 3(\lambda_d^r - \lambda_m)\big)\cdot\Delta\lambda_d^r + 2T\cdot\Delta T = 0 \tag{135}$$

Therefore, the resulting $\Delta\lambda_d^r$ can be computed based on ΔT as

$$\Delta\lambda_d^r = \frac{c_1}{(\lambda_d^r - c_2)^2(\lambda_d^r - c_3)}T\Delta T \tag{136}$$

where the constants c_n depend on the motor parameters only:

$$\begin{cases} c_1 = \dfrac{-1}{2k_1} = \dfrac{{L_d}^2}{2(\xi-1)^2(3P/4)^2} \\ c_2 = \dfrac{\xi}{\xi-1}\lambda_m \\ c_3 = \dfrac{k_2+3}{4}\lambda_m = \dfrac{\xi-0.75}{\xi-1}\lambda_m \\ c_4 = \xi^2 \end{cases} \tag{137}$$

When the torque command is zero, it is obvious that the optimum current magnitude would be zero. Therefore, the optimum flux would be the permanent magnet flux of the motor, λ_m. Therefore, the first iteration of the computation sequence would be as follows:

$$\begin{cases} T[0] = 0 \\ \lambda_d^r[0] = \lambda_d^s[0] = \lambda_m \\ \lambda_q^r[0] = 0 \end{cases} \tag{138}$$

Combining (136), (137), (128), and (115), the following computation method for the next iterations in the sequence can obtained:

$$\begin{cases} T[n+1] = (n+1)\Delta T \\ \lambda_d^r[n+1] = \lambda_d^r[n] + \dfrac{c_1}{(\lambda_d^r[n]-c_2)^2(\lambda_d^r[n]-c_3)}\cdot\dfrac{T[n+1]+T[n]}{2}\cdot\Delta T \\ \big(\lambda_q^r[n+1]\big)^2 = c_4(\lambda_d^r[n+1]-\lambda_m)(\lambda_d^r[n+1]-c_2) \\ \lambda_d^s[n+1] = \sqrt{\big(\lambda_q^r[n+1]\big)^2 + (\lambda_d^r[n+1])^2} \end{cases} \tag{139}$$

As seen in (139), at each iteration, λ_d^r, λ_q^r, and λ_d^s are computed based on their corresponding values in the previous iterations and some constants c_n that only depend on the motor parameters.

Fig. 32 shows the comparison between the exact solution the fourth-order polynomial described in (128) and the iterative computation method presented in (139). It is obvious that the results are practically

the same, while (139) can be easily computed by the microcontroller to initialize the optimum flux look up table at startup using basic arithmetic operations.

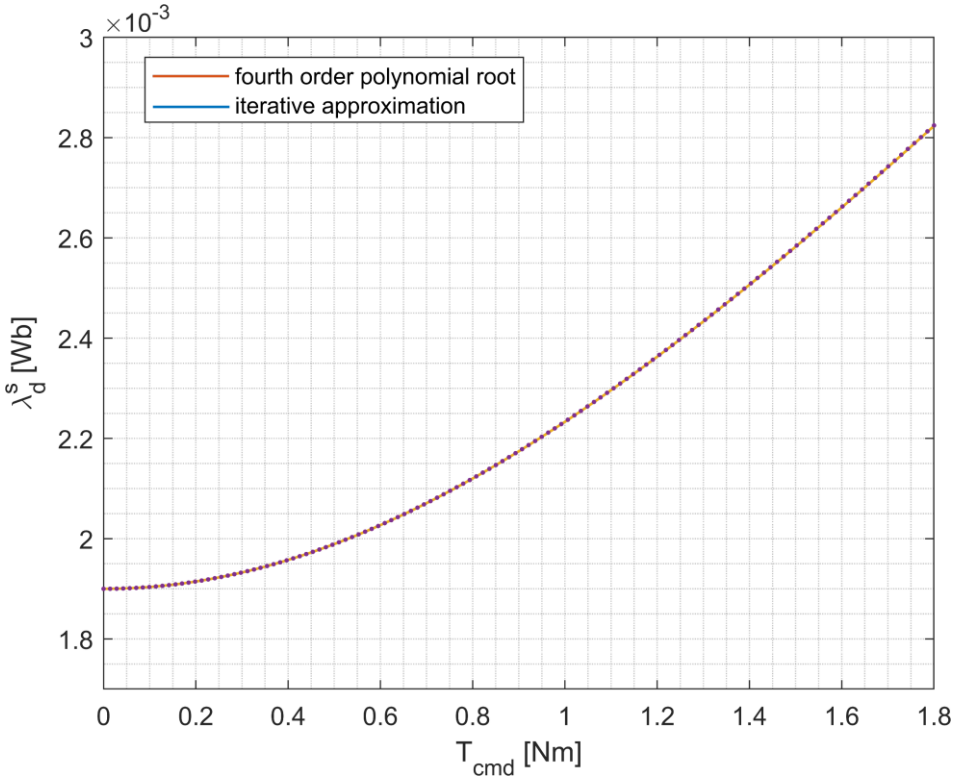

Fig. 32: Optimum flux vector computation using fourth-order polynomial root vs. iterative approximation: $L_q = 28\mu\text{H}$, $L_d = 23\mu\text{H}$, $\lambda_m = 1.9\text{mWb}$, $T_{\max} = 1.8\text{Nm}$, $P = 14$

4.3 Flux Weakening

Like RFO, Flux weakening is also used to reduce the back EMF voltage and to eventually increase the speed of the motor beyond its base speed in SFO. Fig. 33 illustrates the flux weakening block diagram in SFO.

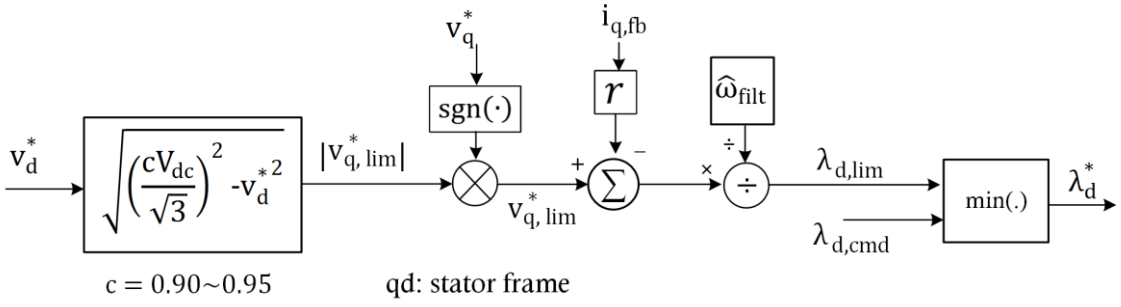

Fig. 33: Flux weakening block diagram

As mentioned in previous chapters, at above base speed the controller requires more voltage, v_q^* than can be produced by the inverter $v_{q,\text{lim}}^*$ which would impose a limit on the amount of flux that can be produced on d-axis according to steady-state q-axis voltage equation as follows,

$$v_q^{s*} = ri_q^s + \left(\omega_r + \frac{d\delta}{dt}\right)\lambda_d^s \tag{140}$$

The flux limit $\lambda_{d,\lim}$ (calculated from the equation above in steady state) is compared against the commanded flux from MTPA block and the smaller value is selected as the final d-axis flux command λ_d^* for the flux controller and MTPV block.

Note that voltage limit $v_{q,\lim}^*$ is calculated from the d-axis command voltage v_d^* where the coefficient $c = 0.90 \sim 0.95$ used for the voltage limit is needed to leave some margin before the absolute voltage limitation is reached to avoid instability.

4.4 MTPV

The voltage equations in SFO can be written as

$$\begin{cases} v_q^s = ri_q^s + \left(\omega + \frac{d\delta}{dt}\right)\lambda_d^s \\ v_d^s = ri_d^s + \frac{d\lambda_d^s}{dt} \end{cases} \tag{141}$$

Assuming that the resistance voltage-drop terms in (141) are much smaller than the back EMF terms, one can approximate (141) in steady state as

$$\begin{cases} v_q^s \approx \omega\lambda_d^s \\ v_d^s \approx 0 \end{cases} \tag{142}$$

What (142) implies is that for a given speed ω, the voltage magnitude is directly proportional to the stator flux magnitude command λ_d^s. Therefore, at a given speed, having MTPV implies obtaining maximum possible torque per flux command λ_d^s.

Fig. 34 demonstrates the projections of the flux vector onto the RFO and SFO reference frames:

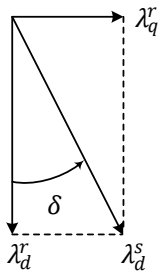

Fig. 34: Flux vector in RFO and SFO reference frames

As can be seen in Fig. 34,

$$\begin{cases} \lambda_q^r = \lambda_d^s \sin(\delta) \\ \lambda_d^r = \lambda_d^s \cos(\delta) \end{cases} \tag{143}$$

Therefore, the torque equation in (122), can be rewritten in terms of the stator flux magnitude λ_d^s and load angle δ as

$$T = \frac{3}{2}\cdot\frac{P}{2}\left(\lambda_d^r\frac{\lambda_q^r}{L_q} - \lambda_q^r\frac{\lambda_d^r - \lambda_m}{L_d}\right) = \tag{144}$$

$$\frac{3}{2}\cdot\frac{P}{2}\cdot\frac{\lambda_d^s}{L_d}\left(\lambda_m \sin(\delta) - \frac{\xi - 1}{\xi}\cdot\frac{\lambda_d^s}{2}\cdot\sin(2\delta)\right)$$

To find the maximum of $T(\delta)$, its derivative can be set to zero:

$$\frac{dT}{d\delta} = 0 \Rightarrow \lambda_m \cos(\delta) - \frac{\xi - 1}{\xi}\lambda_d^s \cos(2\delta) = 0 \quad (145)$$

Equation (145) can be solved to find $\delta_{\max}$ corresponding to $T_{\max}$. $T_{\max}$ is also known as T_{pullout} because applying load angles (δ) above $\delta_{\max}$ will result in a negative slope ($dT/d\delta < 0$) which would make the torque controller unstable. The solution to (145) is as follows.

$$\delta_{\max} = \cos^{-1}\left(k - \sqrt{k^2 + 0.5}\right) \quad (146)$$

where

$$k = \frac{\lambda_m}{4\lambda_d^s} \cdot \frac{\xi}{\xi - 1} \quad (147)$$

As the stator flux magnitude, λ_d^s, changes from 0 (in deep flux weakening region) to ∞ (at very high torque command levels), $\delta_{\max}$ goes from 90° to 135°:

$$\begin{cases} \delta_{\max}: & 90° \rightarrow 135° \\ k: & \infty \rightarrow 0 \\ \lambda_d^s: & 0 \rightarrow \infty \end{cases} \quad (148)$$

Therefore, it is safe to assume that 135° should be cap on δ and load angle should never go above this threshold.

4.4.1 Pullout-Torque Curve

Any load angle above $\delta_{\max}$ will result in a negative slope in $T(\delta)$ curve according to (145). Therefore, for load angles above $\delta_{\max}$, the torque controller becomes unstable. Hence, there is a so-called pullout torque level above which any commanded torque will make the system unstable. This is similar to the concept of pullout torque in induction motors. We can calculate this pullout torque level by substituting $\delta_{\max}$ in (144) to obtain:

$$T_{\text{pull-out}} = \frac{3P}{4} \cdot \frac{(\lambda_d^s)^2}{L_d} \cdot \frac{\xi - 1}{\xi} \sin(\delta_{\max}) \left(\frac{\lambda_m}{\lambda_d^s} \cdot \frac{\xi}{\xi - 1} - \cos(\delta_{\max})\right) \quad (149)$$

where

$$\begin{cases} \cos(\delta_{\max}) = k - \sqrt{k^2 + 0.5} \\ \sin(\delta_{\max}) = \sqrt{1 - \left(k - \sqrt{k^2 + 0.5}\right)^2} \end{cases} \quad (150)$$

Fig. 35 depicts a typical $\delta_{\max}$ and T_{pullout} curve for a motor with medium saliency $\xi = 1.22$:

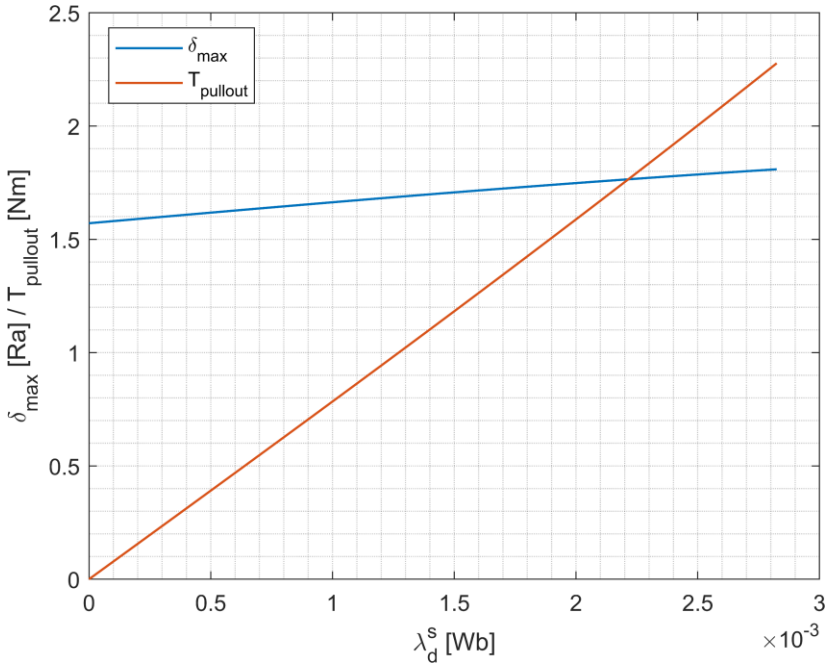

Fig. 35: Sample max load angle and pullout-torque curves for a motor with the following parameters: $L_q = 28\mu\text{H}$, $L_d = 23\mu\text{H}$, $\lambda_m = 1.9\text{mWb}$, $T_{\max} = 1.8\text{Nm}$, $P = 14$

As seen in Fig. 35, the x-axis for the MTPV curves is always the stator flux magnitude, λ_d^s, the range of which is calculated as a result of the MTPA computations discussed in section 4.2. It must also be noted that this range starts from $\lambda_d^s = 0$ since the stator flux magnitude can be reduced to very low levels by the flux weakening controller.

4.4.2 Iterative Computation of Pullout-Torque Look Up Table

Direct implementation of (149) inside the microcontroller at runtime can be computationally intensive. Here, we use a computationally efficient method to initialize a look up table for pullout torque based on the motor parameters at startup. This pullout torque table determines the saturation level of the torque PI controller at runtime.

The arguments of the pullout torque table would be the stator flux magnitude:

$$\lambda_d^s[n] = n \cdot \Delta\lambda_d^s \tag{151}$$

where $\Delta\lambda_d^s$ is the step size.

Let us define the following constants that only depend on the motor parameters:

$$\begin{cases} c_1 = \dfrac{3P}{4} \cdot \dfrac{(\Delta\lambda_d^s)^2}{L_d} \cdot \dfrac{\xi - 1}{\xi} \\ c_2 = \dfrac{\lambda_m}{4\Delta\lambda_d^s} \cdot \dfrac{\xi}{\xi - 1} \end{cases} \tag{152}$$

Then, by rearranging (149), it can be shown that

$$T_{\text{pullout}}[n] = \underbrace{\left(\frac{3P}{4}\frac{(\Delta\lambda_d^s)^2}{L_d}\frac{\xi-1}{\xi}\right)}_{c_1} n^2 \sin(\delta_{\max}[n]) \left(\underbrace{\left(\frac{\lambda_m}{n\Delta\lambda_d^s}\frac{\xi}{\xi-1}\right)}_{4c_2/n} - \cos(\delta_{\max}[n]) \right) \tag{153}$$

Therefore, a computationally efficient process can be devised to initialize the pullout-torque table as follows.

$$\text{Step } n: \begin{cases} c_3[n] = c_2/n \\ \cos(\delta_{\max}[n]) = c_3[n] - \sqrt{(c_3[n])^2 + 0.5} \\ \sin(\delta_{\max}[n]) = \sqrt{1 - (\cos(\delta_{\max}[n]))^2} \\ T_{\text{pullout}}[n] = c_1 n^2 \sin(\delta_{\max}[n])\,(4c_3[n] - \cos(\delta_{\max}[n])) \end{cases} \tag{154}$$

As can be seen in (154), there is no need to directly calculate $\sin(.)$ and $\cos(.)$ as trigonometric functions at each step, which makes the computations less intensive.

4.5 Torque Controller

The torque controller's input is the torque error, $T^* - T$, while its output is the refence load angle, δ^*, as seen in SFO system block diagram. To simplify the derivations, let us assume the gain from δ^* to T is k. The higher the load angle command, δ^*, the higher the resulting torque at the shaft of the motor, T, i.e. $k > 0$. Therefore, the loop gain of the torque control path would be

$$G(s) = \left(k_p + \frac{k_i}{s}\right) \cdot k \tag{155}$$

where torque PI controller is represented by $k_p + k_i/s$.

The closed-loop transfer function can be determined based on the loop gain in (155) as

$$H(s) = \frac{G(s)}{1 + G(s)} \tag{156}$$

which could be further simplified to

$$H(s) = \frac{1 + \frac{s}{k_i/k_p}}{1 + \frac{s}{kk_i/(kk_p + 1)}} = \frac{1 + \frac{s}{\omega_z}}{1 + \frac{s}{\omega_p}} \tag{157}$$

As can be seen in (157), the simplified closed-loop transfer function has a pole and a zero as follows:

$$\begin{cases} \omega_z = \dfrac{k_i}{k_p} \\ \omega_p = \dfrac{kk_p}{1 + kk_p}\omega_z \end{cases} \tag{158}$$

It could be seen from (158) that

$$\omega_p < \omega_z \rightarrow r = \frac{\omega_p}{\omega_z} < 1 \tag{159}$$

i.e. the pole frequency is always lower than the zero frequency, which is desirable because there will be enough attenuation at higher frequencies. The ratio, r, in (159) determines this attenuation at high frequencies and is a user variable.

Based on (158) and (159), we can compute the required k_p and k_i based on desired ω_z and r:

$$\begin{cases} k_p = \dfrac{r}{(1 - r)k} \\ k_i = k_p \omega_z \end{cases} \tag{160}$$

The gain k in (160) can be approximated for control design purposes. From (144), T can be expressed in terms of δ as

$$\begin{aligned} T &= \frac{3}{2} \cdot \frac{P}{2} \cdot \frac{\lambda_d^s}{L_d}\left(\lambda_m \sin(\delta) - \frac{\xi - 1}{\xi} \cdot \frac{\lambda_d^s}{2} \cdot \sin(2\delta)\right) \\ &= \frac{3P}{4}\lambda_d^s\left(\frac{\lambda_m}{L_d}\sin(\delta) + \left(\frac{1}{L_q} - \frac{1}{L_d}\right)\frac{\lambda_d^s}{2}\sin(2\delta)\right) \end{aligned} \tag{161}$$

Applying small-signal perturbations to (161) and linearizing the result, we will have

$$\frac{\hat{T}}{\hat{\delta}} = \frac{\partial T}{\partial \delta} = \frac{3P}{4} \cdot \lambda_d^s\left(\frac{\lambda_m}{L_d}\cos(\delta) - \left(\frac{1}{L_q} - \frac{1}{L_d}\right) \cdot \lambda_d^s \cdot \cos(2\delta)\right) \tag{162}$$

Then, k can be approximated as the gain from $\hat{\delta}$ to $\hat{T}$ around $\delta = 0$:

$$k \approx \frac{3P}{4} \cdot \lambda_m \left(\frac{\lambda_m}{L_d} - \left(\frac{1}{L_q} - \frac{1}{L_d} \right) \lambda_m \right) \tag{163}$$

This approximation usually results in a good starting point for designing the torque controller but more adjustments may be necessary to fine tune the controller.

4.6 Flux Controller

As seen in the system block diagram for SFO, the input and output of the flux controller are flux magnitude error, $\lambda_d^{s*} - \lambda_d^s$, and commanded d-axis voltage, v_d^s. The "plant" transfer function for flux controller can be approximated as follows. From (141), it can be seen that

$$v_d^s \propto \frac{d\lambda_d^s}{dt} \tag{164}$$

Here, we are considering the resistance term, ri_d^s, a negligible disturbance that the closed-loop controller can compensate for. Therefore, the plant model can be approximated as an integrator according to (164).

Then, the loop-gain transfer function would be

$$G(s) = \left(k_p + \frac{k_i}{s} \right) \cdot \frac{1}{s} \tag{165}$$

The closed-loop transfer function can be determined based on the loop-gain as

$$H(s) = \frac{G(s)}{1 + G(s)} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} \tag{166}$$

Let us place the poles of the closed-loop transfer function, $H(s)$, at

$$\begin{cases} \omega_{p,1} = \omega_0 \\ \omega_{p,2} = (r-1)\omega_0 \end{cases} \tag{167}$$

where r is the pole separation ratio ($r > 1$). Hence,

$$s^2 + k_p s + k_i = s^2 + (r\omega_0)s + (r-1)\omega_0^2 \tag{168}$$

From (168), PI controller's parameters can be computed as

$$\begin{cases} k_p = r\omega_0 \\ k_i = (r-1)\omega_0^2 \end{cases} \tag{169}$$

based on the desired pole separation ratio, r, and bandwidth, ω_0.

4.7 Load Angle Controller

As seen in the system block diagram for SFO, the input and output of the load angle controller are load angle error, $\delta^* - \delta$, and commanded q-axis voltage, v_q^s. The "plant" transfer function for flux controller can be approximated as follows. From (141), it can be seen that

$$v_q^s \propto \lambda_d^s \frac{d\delta}{dt} \tag{170}$$

Here, we are considering the resistance and back emf terms (ri_d^s and $\omega\lambda_d^s$) as disturbances that the closed-loop controller can compensate for. Therefore, the plant model can be approximated as a gain and an integrator according to (170).

Then, the loop-gain transfer function would be

$$G(s) = \left(k_p + \frac{k_i}{s} \right) \cdot \frac{1}{\lambda_d^s s} \tag{171}$$

The closed-loop transfer function can be determined based on the loop-gain as

$$H(s) = \frac{G(s)}{1+G(s)} = \frac{k_p s + k_i}{\lambda_d^s s^2 + k_p s + k_i} \tag{172}$$

Let us place the poles of the closed-loop transfer function, $H(s)$, similar to (167). Hence,

$$s^2 + \frac{k_p}{\lambda_d^s} s + \frac{k_i}{\lambda_d^s} = s^2 + (r\omega_0)s + (r-1)\omega_0^2 \tag{173}$$

From (173), PI controller's parameters can be computed as

$$\begin{cases} k_p = \lambda_d^{s*} r\omega_0 \\ k_i = \lambda_d^{s*}(r-1)\omega_0^2 \end{cases} \tag{174}$$

based on the desired pole separation ratio, r, and bandwidth, ω_0.

The dynamic response of the load angle controller can be improved by increasing its bandwidth, ω_0, as motor's electrical speed, ω, increases. Fig. 36 depicts a typical curve for a "bandwidth multiplication ratio" based on the electrical speed, ω. As can be seen in Fig. 36, the controller bandwidth goes from ω_0 to $m\omega_0$ $(m > 1)$ as the normalized electrical speed goes from ω_{L} to ω_{H}.

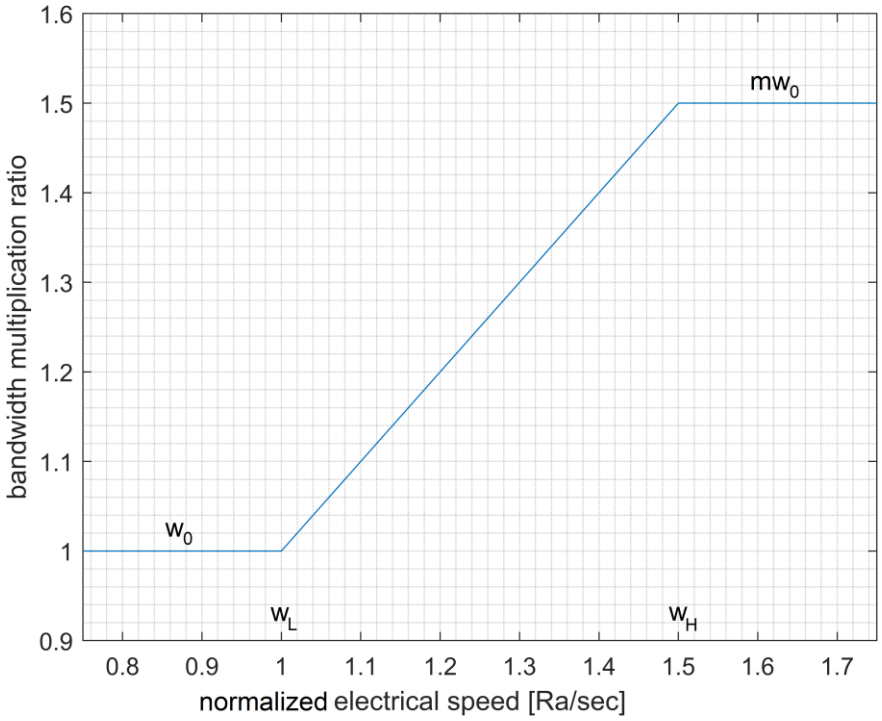

Fig. 36: Typical bandwidth multiplication ratio curve for load angle controller

4.8 Adaptive Sensorless Observer

This section describes the operation of this observer in SFO, whereas its operation in RFO was presented in section 3.5.

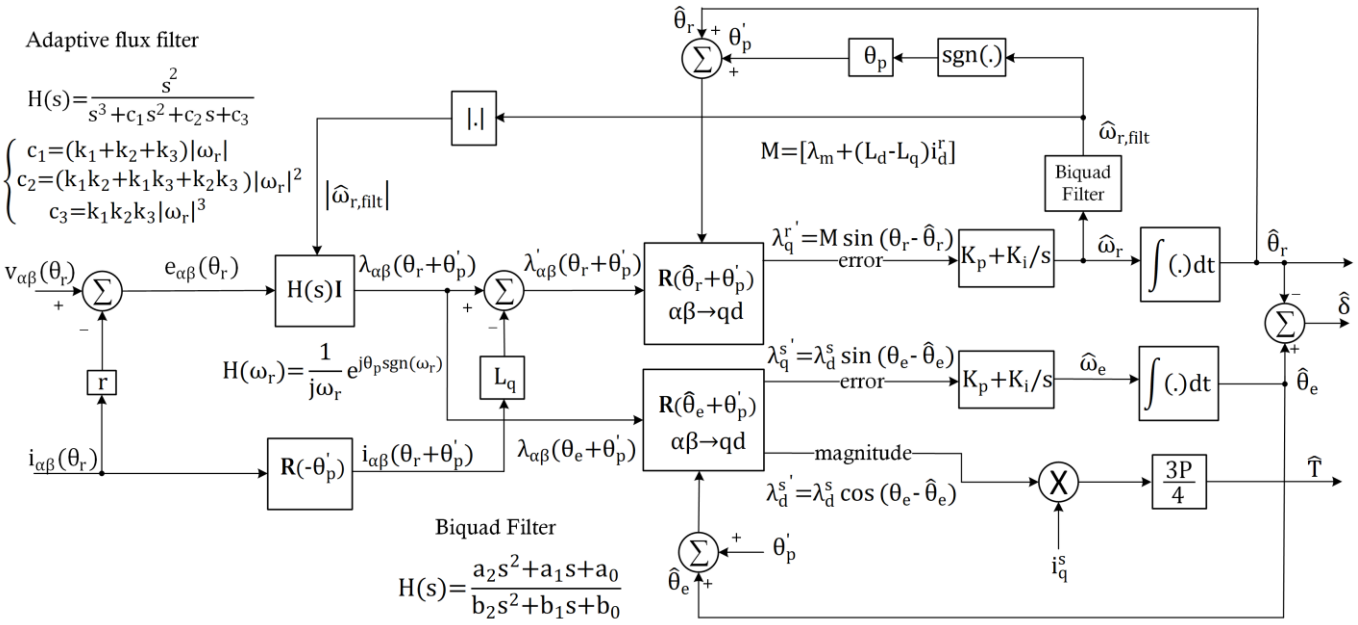

Fig. 37: Universal Adaptive Zero-Phase-Shift Position and Flux Observer in SFO

Fig. 37 depicts the block diagram of the observer in SFO. In SFO, in addition to rotor's electrical angle, stator's electrical angle, stator flux magnitude, and load angle are required to control the system. Stator's electrical angle and the corresponding stator frame are defined such that

$$\lambda_q^s = 0 \quad @\theta_e \tag{175}$$

In other words, in SFO, all of the stator flux is aligned with the d-axis.

With a change of variable from θ_r to θ_e, a deductive reasoning similar to (43)-(50) can be used to derive $\boldsymbol{\lambda}_{\alpha\beta}$ in terms of θ_e:

$$\boldsymbol{\lambda}_{\alpha\beta}(\theta_e + \theta_p') = \boldsymbol{\lambda}_{\alpha\beta}(\theta_r + \theta_p') \tag{176}$$

Fluxes in (176) can then be expressed as

$$\boldsymbol{\lambda}_{\alpha\beta}(\theta_e + \theta_p') = \boldsymbol{R}(-\theta_e - \theta_p')\boldsymbol{\lambda}_{qd}^s \tag{177}$$

$$= \begin{bmatrix} \cos(\theta_e + \theta_p') & \sin(\theta_e + \theta_p') \\ -\sin(\theta_e + \theta_p') & \cos(\theta_e + \theta_p') \end{bmatrix} \begin{bmatrix} 0 \\ \lambda_d^s \end{bmatrix}$$

$$= \lambda_d^s \begin{bmatrix} \sin(\theta_e + \theta_p') \\ \cos(\theta_e + \theta_p') \end{bmatrix}$$

Therefore, the stator electrical angle, θ_e, and stator flux magnitude, λ_d^s, can be extracted as follows

$$\theta_e + \theta_p' = \tan^{-1}\left(\frac{\lambda_\alpha}{\lambda_\beta}\right) \tag{178}$$

$$\left|\boldsymbol{\lambda}_{\alpha\beta}(\theta_e + \theta_p')\right| = \lambda_d^s \tag{179}$$

To realize the $\tan^{-1}(\cdot)$ function in (178), we use another PLL to extract θ_e.

As seen in Fig. 37, $\boldsymbol{\lambda}_{\alpha\beta}(\theta_e + \theta_p')$ is fed to such PLL with a Park transform as follows

$$\boldsymbol{\lambda}_{qd}^{s\prime} = \boldsymbol{R}(\hat{\theta}_e + \theta_p')\boldsymbol{\lambda}_{\alpha\beta}(\theta_e + \theta_p') = \tag{180}$$

$$\lambda_{ds}^s \begin{bmatrix} \cos(\hat{\theta}_e + \theta_p') & -\sin(\hat{\theta}_e + \theta_p') \\ \sin(\hat{\theta}_e + \theta_p') & \cos(\hat{\theta}_e + \theta_p') \end{bmatrix} \begin{bmatrix} \sin(\theta_e + \theta_p') \\ \cos(\theta_e + \theta_p') \end{bmatrix} =$$

$$\lambda_d^s \begin{bmatrix} \sin(\theta_e - \hat{\theta}_e) \\ \cos(\theta_e - \hat{\theta}_e) \end{bmatrix}$$

As seen in (180), the q-axis and d-axis components of such Park transform are the error and magnitude signals, respectively:

$$\lambda_q^{s\prime} = \lambda_d^s \sin\underbrace{\left(\theta_e - \hat{\theta}_e\right)}_{\theta_{e,\text{error}}} \tag{181}$$

$$\lambda_d^{s\prime} = \lambda_d^s \cos\underbrace{\left(\theta_e - \hat{\theta}_e\right)}_{\theta_{e,\text{error}}} \tag{182}$$

If the PLL is stable, the error ($\theta_{e,\text{error}}$) is zero in steady state. Therefore, $\lambda_d^{s\prime}$ and $\hat{\theta}_e$ will converge to λ_d^s and θ_e, respectively.

The torque in SFO can be expressed as

$$T = \frac{3}{2}\cdot\frac{P}{2}\left(\lambda_d^s i_q^s - \lambda_q^s i_d^s\right) = \frac{3}{2}\cdot\frac{P}{2}\lambda_d^s i_q^s \tag{183}$$

Therefore, the PLL will produce the estimated torque as a byproduct of the $\hat{\theta}_e$ estimation as well:

$$\hat{T} = \frac{3}{2}\cdot\frac{P}{2}\lambda_d^{s\prime} i_q^s \tag{184}$$

In addition, the load angle can be estimated by subtracting the rotor's electrical angle from that of the stator:

$$\hat{\delta} = \hat{\theta}_e - \hat{\theta}_r \tag{185}$$

4.8.1 Adaptive Flux Extraction Filter

The adaptive flux extraction filter for SFO (Fig. 37) is identical to that of RFO (Fig. 9). Please refer to section 3.5.1 for more details.

4.8.2 Critically-Damped Biquad Low-Pass Filter

The critically-damped biquad low-pass filter for SFO (Fig. 37) is identical to that of RFO (Fig. 9). Please refer to section 3.5.2 for more details.

4.9 Sliding Mode Current Limiter

In SFO, current is not directly controlled (instead, torque, flux, and load angle are directly controlled). However, the current magnitude must be capped such that it never goes above a certain limit. This limit is determined by the I^2T module and is set to the peak current rating of the motor, I_{peak} in normal conditions. When I^2T protection engages (see section 6.12 for more information), this limit is set to the continuous current rating of the motor, I_{cont}. Therefore, there is a need to control the current magnitude in SFO, *if and only if*, it attempts to go above the limit.

A nonlinear sliding-mode current controller for controlling the current in SFO is employed for this purpose:

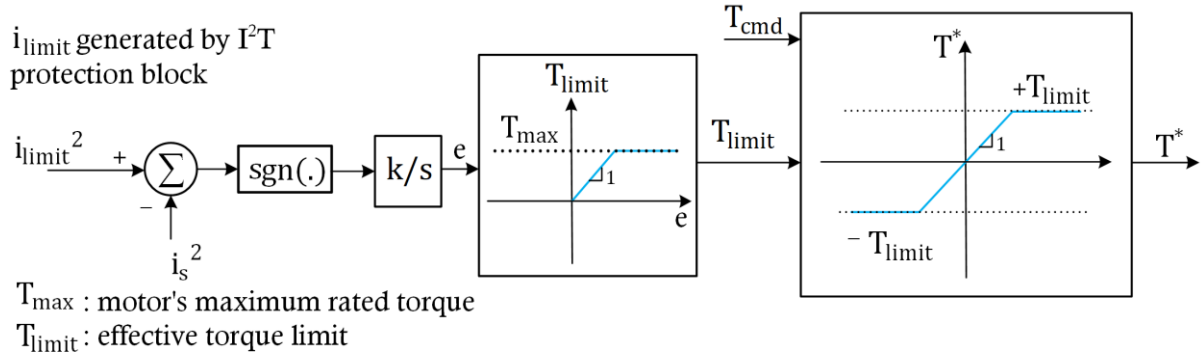

Fig. 38: Sliding mode current controller block diagram

This controller is a nonlinear sliding-mode controller. Therefore, we will use Lyapunov stability theorem to analyze it in the next section. For simplification of the derivations in the next section, let us assume that i_{limit} (whether it is I_{peak} or I_{cont}) is the command value, as shown in Fig. 38:

$$i_s^* = i_{\text{limit}} \tag{186}$$

Also, for the sign function in Fig. 38, it can be shown that

$$\text{sgn}(i_s^{*2} - i_s^2) = \text{sgn}\big((i_s^* - i_s)(i_s^* + i_s)\big) = \text{sgn}(i_s^* - i_s) \tag{187}$$

since both current magnitudes are always positive ($i_s^*, i_s \geq 0$).

It must be noted that when $i_s < i_s^*$, the integrator, k/s, eventually saturates to $T_{\max}$. Therefore, this control loop will be automatically deactivated. It is only activated when the current magnitude, i_s, tries to surpass the limit, i_s^*.

4.9.1 Lyapunov Stability Analysis

Let us define the current error as

$$\bar{\imath}_s = i_s^* - i_s \tag{188}$$

and define the Lyapunov potential function as

$$V = \frac{\bar{\imath}_s^2}{2} \tag{189}$$

It can be seen that this function is positive definite,

$$\begin{cases} V > 0 \Leftrightarrow \bar{\imath}_s \neq 0 \\ V = 0 \Leftrightarrow \bar{\imath}_s = 0 \end{cases}: \quad \text{positive definite} \tag{190}$$

and radially unbounded,

$$V \to \infty \Leftrightarrow |\bar{\imath}_s| \to 0: \quad \text{radially unbounded} \tag{191}$$

Therefore, if we can show that its time derivative is also negative definite, all three conditions for global asymptotic stability are satisfied.

The time derivative of V can be written as

$$\dot{V} = \bar{\imath}_s \frac{d\bar{\imath}_s}{dt} = \bar{\imath}_s \left(\frac{di_s^*}{dt} - \frac{di_s}{dt}\right) = -\bar{\imath}_s \frac{di_s}{dt} \tag{192}$$

where

$$i_s = cT^* \tag{193}$$

and c is a scalar positive gain that represents the positive relationship between torque command, T^*, and current magnitude, i_s. In other words, it shows that if more torque is required, more current should be applied. The exact calculation of c is not necessary for the stability analysis here, hence, it is skipped. It is only important that its sign is positive, which is the case because of the positive relationship between torque command, T^*, and current magnitude, i_s.

Based on (192), (193), (187), and Fig. 38, we can write $\dot{V}$ as

$$\dot{V} = -c\bar{\imath}_s \frac{dT^*}{dt} = -ck\bar{\imath}_s \mathrm{sgn}(i_s^{*2} - i_s^2) = -ck\bar{\imath}_s \mathrm{sgn}(\bar{\imath}_s) = -ck|\bar{\imath}_s| \quad (194)$$

The significance of (194) is that it proves

$$\begin{cases} \dot{V} < 0 \Leftrightarrow \bar{\imath}_s \neq 0 \\ \dot{V} = 0 \Leftrightarrow \bar{\imath}_s = 0 \end{cases} : \quad \text{negative definite} \quad (195)$$

or in another words, $\dot{V}$ is a negative definite function. Therefore, all of the stability conditions based on Lyapunov theorem are satisfied and the controller illustrated in Fig. 38 is globally asymptotically stable.

4.9.2 Integrator Gain and Reaching Time

In section 4.9.1, it was established that the sliding-mode regulator in Fig. 38 is stable as long as the integrator gain, k, is positive. However, section 4.9.1 sets no criteria for picking k. In this section, it will be shown that the reaching time is directly controlled by k and a method for choosing k based on the desired reaching time is presented.

When I²T protection engages, the current limit goes from I_{peak} to I_{cont}. Therefore, the worst-case initial value of $\bar{\imath}_s(0)$ would be

$$\bar{\imath}_s(0) = \left(I_{\text{cont}} - I_{\text{peak}}\right) \quad (196)$$

The rate by which $\bar{\imath}_s$ changes can be written as

$$\frac{d\bar{\imath}_s}{dt} = -ck\mathrm{sgn}(\bar{\imath}_s) = ck \quad (197)$$

Therefore,

$$\bar{\imath}_s(t) = \bar{\imath}_s(0) + \frac{d\bar{\imath}_s}{dt} \cdot t = \left(I_{\text{cont}} - I_{\text{peak}}\right) + ck \cdot t \quad (198)$$

From (198), we can calculate how much time it takes to reach the new operating point:

$$ck \cdot T_{\text{reach}} = \left(I_{\text{peak}} - I_{\text{cont}}\right) \quad (199)$$

Thus, the minimum value for integrator gain, $k_{\min}$, can be calculated based on (199) as

$$k_{\min} = \frac{\left(I_{\text{peak}} - I_{\text{cont}}\right)}{c_{\min} \cdot T_{\text{reach}}} \quad (200)$$

where $c_{\min}$ can be shown to be

$$c_{\min} = \min\left\{\frac{1}{\sqrt{(3P/8)\left(L_q - L_d\right)T_{\max}}}, \frac{1}{(3P/4)\lambda_m}\right\} \quad (201)$$

4.10 Single-Shunt Current Reconstruction

The current reconstruction procedure for SFO is similar to that for RFO. Please refer to section 3.6 for more information.

4.11 Experimental Results

To demonstrate the performance of SFO control, the same motor that was tested in section 3.7 is used here. Control mode is set to SFO sensorless torque control. The commanded torque is 1.4 Nm. Fig. 39, shows the three phase current waveforms. As seen, the current waveforms are regulated and balanced. Fig. 40 shows variables live watch snapshot from Ozone debugger. It is also seen in Fig. 40 that feedbacks follow commands (torque and flux).

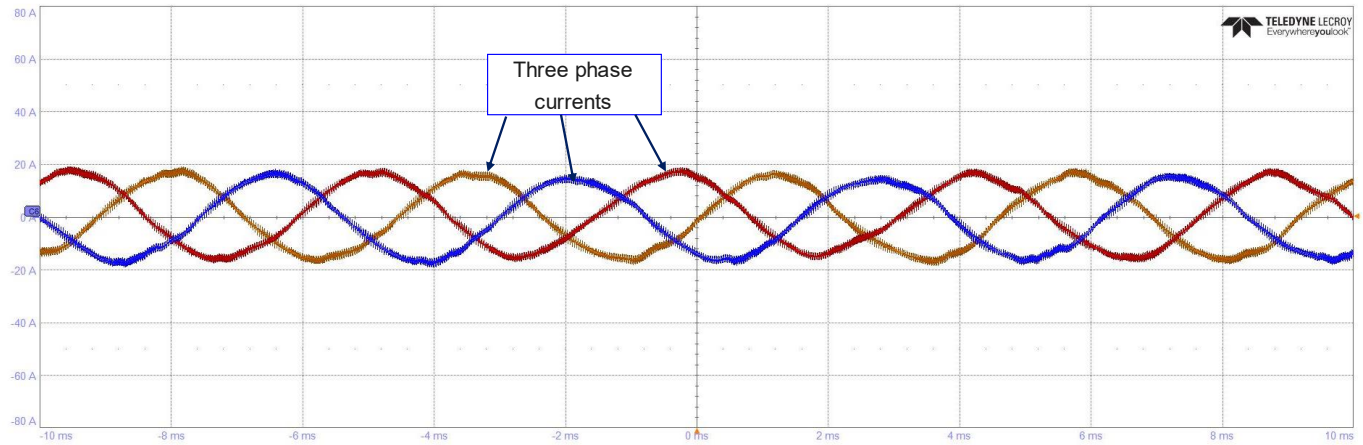

Fig. 39: Three phase current waveforms in SFO sensorless torque mode with commanded torque at 1.4 Nm.

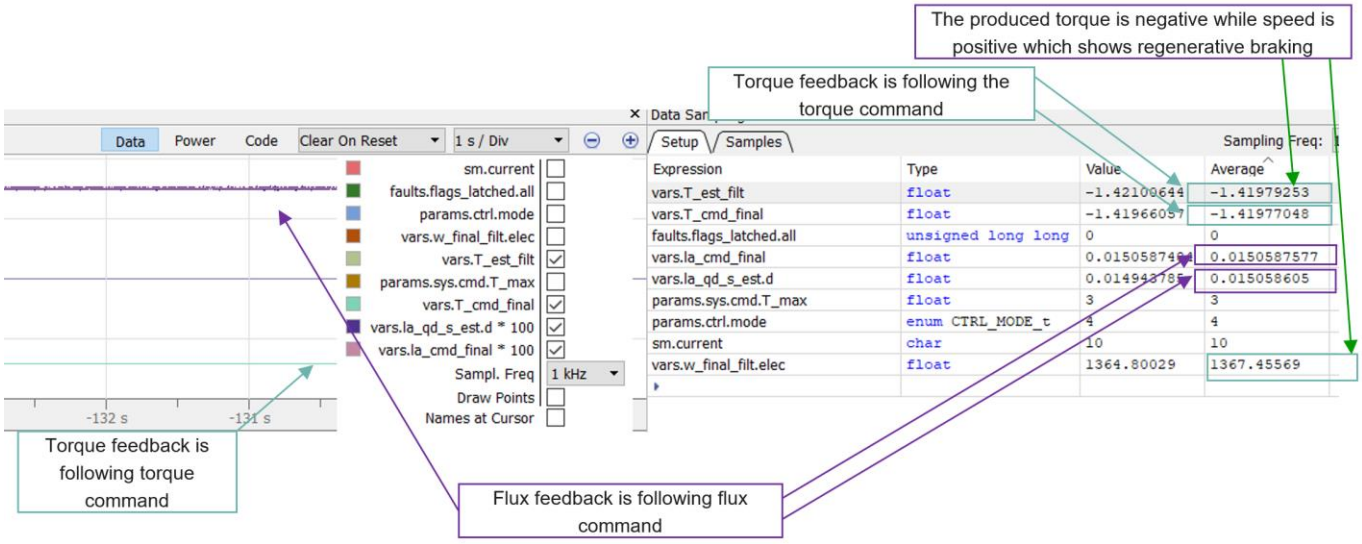

Fig. 40: Variables live watch snapshot from Ozone debugger: feedbacks follow commands.

5 Trapezoidal and Block Commutation

Trapezoidal and block commutation control methods are used to control Brushless DC (BLDC) motors with trapezoidal back EMF. These are simple control methods where a DC current is injected into motor phases to create constant torque throughout one revolution of the motor shaft. Although in theory the torque should be constant throughout one revolution, but in practice there are torque ripple spikes during the commutation transitions. This is one of the drawbacks of using the block commutation method. Fig. 41 shows the block diagram of block commutation control method. As shown in this figure, the control method uses the position feedback of the rotor from the hall sensors to calculate the speed and to close the outer-loop speed controller. The output of the speed controller is a current reference (which is proportional to torque) that is used in the current controller to generate the command voltage and eventually the switching patterns used in the block commutation block. More details on the block commutation module will be provided in the subsequent chapters. It should be also noted that current controller can be bypassed, and the voltage command can be directly generated by the speed controller. This will further simplify the implementation of this method which eliminates the need for accurate current sensors as well.

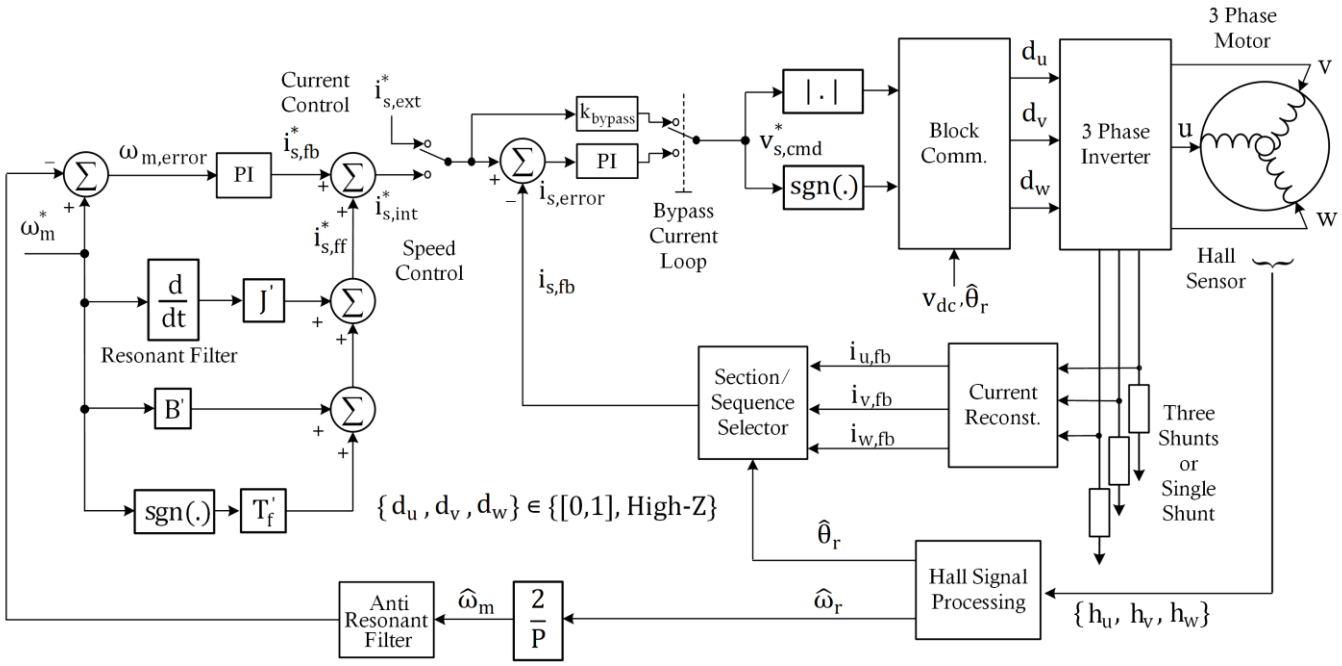

Fig. 41: Block commutation control method

5.1 Speed Controller

This section will cover the speed controller used in the TBC control type. The parameters of the speed controller are significantly impacted by the mechanical load. Fig. 42 illustrates the mathematical model of the mechanical load driven by the motor and electrical drive system. This model includes T_f, representing coulombic friction, B as the coefficient of viscous friction, and J denoting the inertia. When utilizing the firmware to operate any motor, it is crucial to accurately measure or estimate these three parameters and input them into the graphical user interface (GUI) or directly hard code them in the appropriate header files. The speed controller's k_p, k_i, and feedforward terms are directly derived from these parameters, as will be explained further.

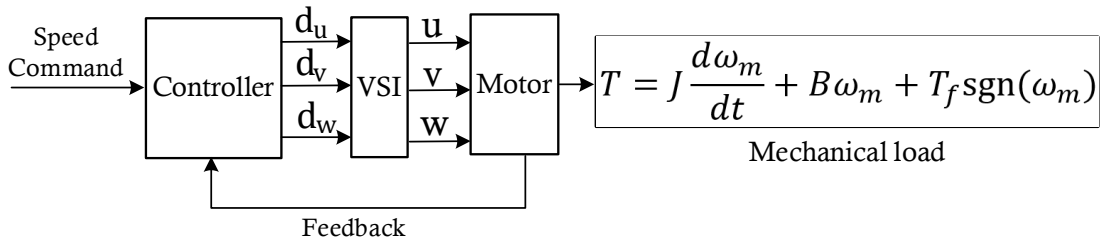

Fig. 42: Mechanical load run by the motor-drive system

To derive the value of proportional, integral, and feedforward terms of the speed controller, the speed loop block diagram along with a simple model of motor and mechanical load will be used as shown in Fig. 43. To derive the K_t value, the relationship between electromagnetic torque and motor current needs to be established. In TBC always two phases are active, and one phase is disconnected. Therefore, power can be expressed as:

$$\mathrm{P}_{out} = 2ei_{\mathrm{ph}} = 2\lambda_m\omega_e i_{\mathrm{ph}} \tag{202}$$

Where i_{ph} is the phase current which in TBC is relatively dc. Also, e is the back emf voltage with a reasonably dc form. To derive K_t as shown in Fig. 43, power and torque equations need to be considered.

$$\mathrm{P}_{out} = T\omega_m, \quad \omega_m = \frac{2}{P}\omega_e \tag{203}$$

$$T = k_t i_{\mathrm{ph}} \tag{204}$$

Hence after combining (202), (203) and (204) k_t can be written as:

$$k_t = 2\left(\frac{P}{2}\right)\lambda_m \tag{205}$$

In the firmware, all current quantities are scaled to represent $\sqrt{2}$ times their RMS values (i.e. peak values when current waveforms are sinusoidal in RFO or SFO):

$$i_{\mathrm{s}} = \sqrt{2}i_{\mathrm{ph}} = \sqrt{2}\frac{T^*}{k_t} \tag{206}$$

Multiplying both measured currents and commanded currents by $\sqrt{2}$ is just a convention that will make the common fault detection and protection modules (e.g. I^2T) work seamlessly for all control modes (RFO, SFO, TBC). It won't affect the control loops in TBC because both measured and command values are multiplied by $\sqrt{2}$ and the controller gains are adjusted to account for the difference as well.

Pole-zero cancellation technique is used to find the PI controller's integral and proportional gain. By having

$$\frac{\mathrm{k_i}}{\mathrm{k_p}} = \frac{B}{J} \tag{207}$$

the controller zero will cancel the mechanical load's pole. The PI controller coefficients are also proportional to speed loop bandwidth as shown below:

$$\mathrm{k}_i \propto B\omega_{BW} \tag{208}$$

$$\mathrm{k}_p \propto J\omega_{BW} \tag{209}$$

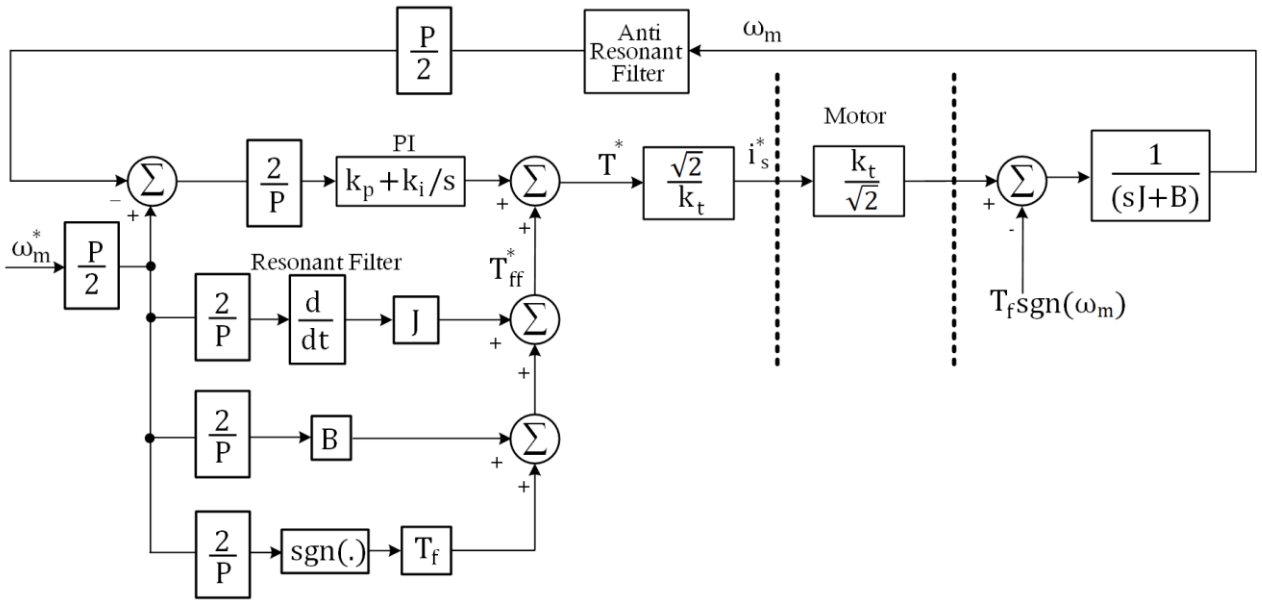

Fig. 43: TBC speed loop block diagram before rescaling the parameters

The proportional and integral gains are ultimately determined after being rescaled to account for the motor parameters k_t and P. Fig. 44 illustrates how k_t and P as were shown in Fig. 43 can be integrated into the controller parameters, leading to the new updated forms:

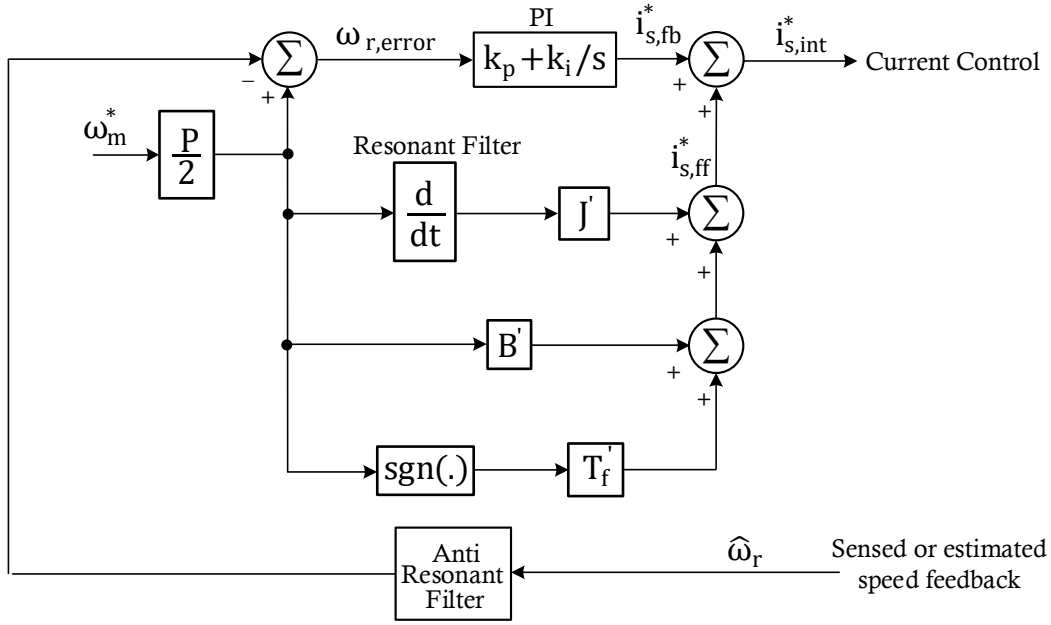

Fig. 44: TBC speed loop block diagram as implemented in the firmware.

The speed contoller parameters after rescaling are as follows:

$$k_i = \left(\frac{\sqrt{2}}{k_t}\right)\left(\frac{2}{P}\right)B\omega_{BW} = (2\sqrt{2})\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right)B\omega_{BW} \quad (210)$$

$$k_p = \left(\frac{\sqrt{2}}{k_t}\right)\left(\frac{2}{P}\right)J\omega_{BW} = (2\sqrt{2})\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right)J\omega_{BW} \quad (211)$$

The feedforward terms have been also updated as depicted in the following manner:

$$B' = \left(\frac{\sqrt{2}}{k_t}\right)\left(\frac{2}{P}\right)B = (2\sqrt{2})\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right)B \quad (212)$$

$$J' = \left(\frac{\sqrt{2}}{k_t}\right)\left(\frac{2}{P}\right)J = (2\sqrt{2})\left(\frac{1}{P^2}\right)\left(\frac{1}{\lambda_m}\right)J \quad (213)$$

$$T_f' = \left(\frac{\sqrt{2}}{k_t}\right)T_f = (\sqrt{2})\left(\frac{1}{P}\right)\left(\frac{1}{\lambda_m}\right)T_f \quad (214)$$

The feedforward terms in the speed loop are affected by both mechanical load and motor parameters. These three feedforward terms play a role in enhancing the dynamic performance of the speed loop. The inertia term utilizes a second-order resonant filter to estimate the acceleration, aiming to mitigate the potential impact of noise that could arise if a direct derivation method had been employed.

5.2 Current Controller

To design the current controller for block commutation method, first the model of the system is needed. The three-phase model of the PMSM motor assuming $L_q = L_d = L_M = \frac{3}{2}L_m$ can be written as follows

$$v_{uvw} = Ri_{uvw} + \begin{bmatrix} L_m & -L_m/2 & -L_m/2 \\ -L_m/2 & L_m & -L_m/2 \\ -L_m/2 & -L_m/2 & L_m \end{bmatrix}\frac{di_{uvw}}{dt} + e_{uvw} \quad (215)$$

At each hall state in block commutation control method, two phases conduct the current and one phase has zero current as it will be described in the next section, e.g.

$$v_u = v = Ri_u + L_m\frac{di_u}{dt} - \frac{L_m}{2}\frac{di_w}{dt} + \lambda_m\omega \quad (216)$$
$$v_w = -v = Ri_w + L_m\frac{di_w}{dt} - \frac{L_m}{2}\frac{di_u}{dt} - \lambda_m\omega$$

Assuming $i_u = -i_w = i$ and $i_v = 0$ and subtracting the voltage equations, a single dynamic equation can be obtained as,

$$v = Ri + L_M\frac{di}{dt} + \lambda_m\omega \quad (217)$$

Given the motor model, the block diagram of the current controller can be drawn as follows

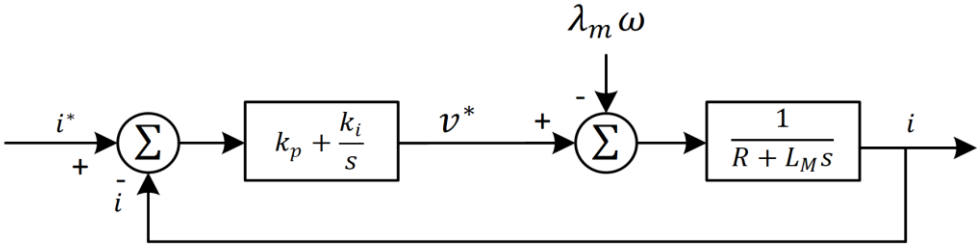

Fig. 45: Current controller block diagram.

The current command is used as an input for the current controllers and the output would be the voltage reference which is eventually applied to the motor using the inverter to control the current. This means that the voltages are considered as an input to the model of the PMSM machine as shown. Note that like FOC method, a feedforward term can be applied in the control system to decouple the speed dependent term ($\lambda_m \omega$) so the controller is only responsible for controlling the RL_M circuit's current.

In the closed loop system in Fig. 45, it is desired to cancel the pole of the system with the zero of the PI controller to reduce the order of closed loop system, i.e.

$$\frac{k_p}{k_i} = \frac{L_M}{R} \tag{218}$$

This will reduce the loop gain transfer function of the system to

$$G = \frac{k_p}{L_M s} = \frac{k_i}{Rs} \tag{219}$$

which can be used to calculate the PI controller coefficients for a given system bandwidth ω_c as,

$$k_p = \omega_c L_M \qquad k_i = \omega_c R \tag{220}$$

5.3 Block Commutation

Block commutation is a simple control method that is used to control BLDC motors with trapezoidal back EMF. This control method uses HALL sensors to determine the position of the rotor and inject current to the windings in phase with the back EMF to maximize the torque for the current injected into the phases.

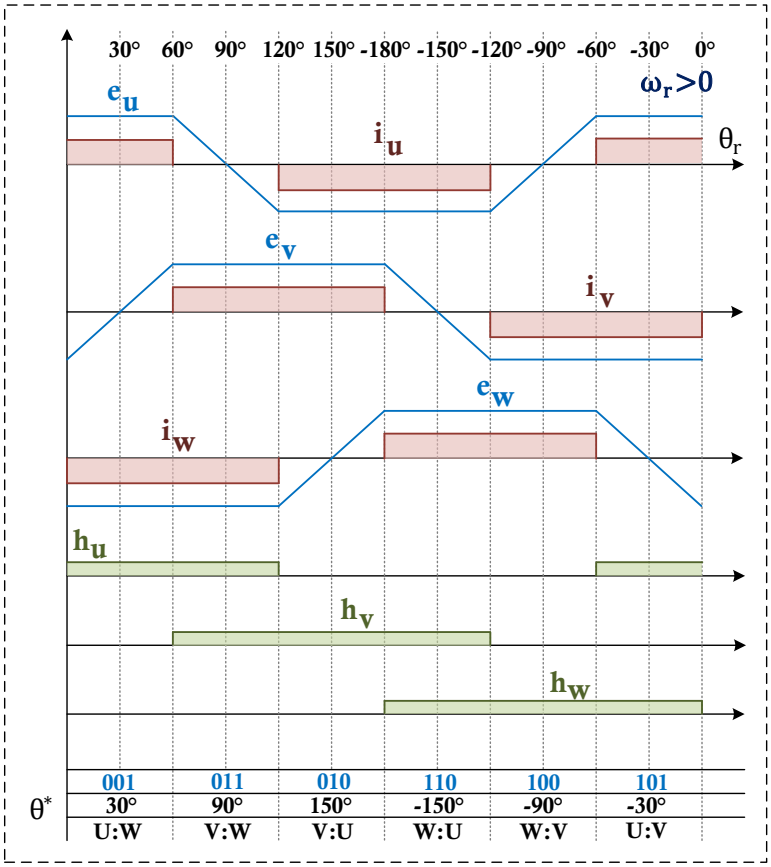

Fig. 46: Block commutation control method

As shown in Fig. 46 the positive edge of HALL transitions h_{uvw} are in line with their associated phase entering the maximum positive back EMF voltage and the negative edge transitions are in line with their associated phase entering minimum negative back EMF. The block commutation control method is responsible to force the current into the phase (positive current) when the back EMF is maximum positive and out of the phase (negative current) when the back EMF is minimum negative. The phase current needs to be driven down to zero when the back EMF is transitioning from positive to negative or the other way around. This can be done by taking the associated inverter phase to high-z by turning both high side and low side switches off.

Note that the current controller is responsible to hand out the reference voltage to block commutation block that is responsible for determining which phase has to conduct and which one has to be in high-z. Eventually, the voltage and high-z commands are routed to PWM block to create the modulation for each phase. Commutation occurs when one phase must stop carrying the current and yield it to the next phase.

For more information about the hall patterns and how they affect the block commutation at positive and negative speed, please refer to section 6.4.

5.4 Trapezoidal Commutation

As mentioned, block commutation is a simple control method for BLDC motors. However, this simple control strategy comes with a couple of drawbacks.

At the end of each HALL state, the current on one phase must go to zero and the other phase must carry the current. This is called commutation. During the commutation, current flows through the power switch body diode on the phase which must stop carrying current. This will cause significant power dissipation in the inverter. Also, the rate at which current ramps up and down in the two phases involved in commutation is not the same which creates torque ripple causing motor power loss, acoustic noise, mechanical vibration etc.

To avoid the problems associated with conventional block commutation, trapezoidal references are used in trapezoidal commutation to ramp up and ramp down the current in a controlled way for the phases involved in commutation. Fig. 47 shows the block diagram of this method. This control method uses three controllers where C_X and C_Y only operate during commutation to control the ramps and controller C_Z always operates similar to the controller in a conventional block commutation method.

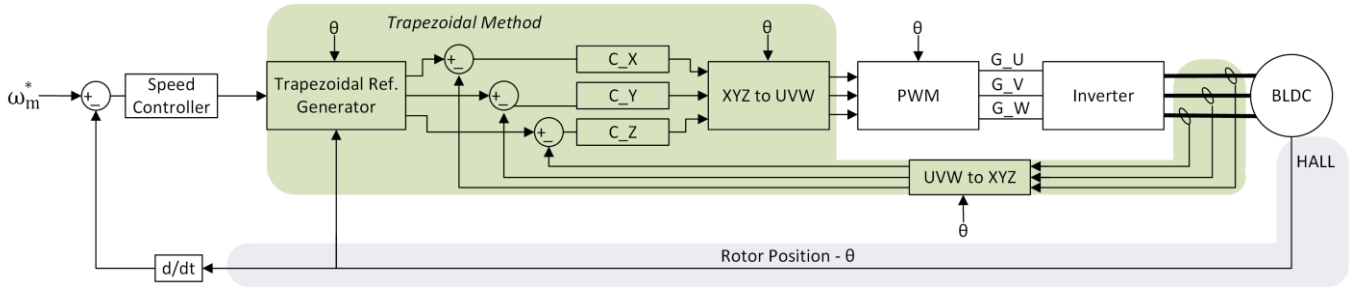

Fig. 47: Trapezoidal commutation control method

Furthermore, as shown in this figure, the block UVW to XYZ is responsible for mapping the currents of phases (UVW) to those of the trapezoidal commutation (XYZ) at each moment according to Fig. 48.

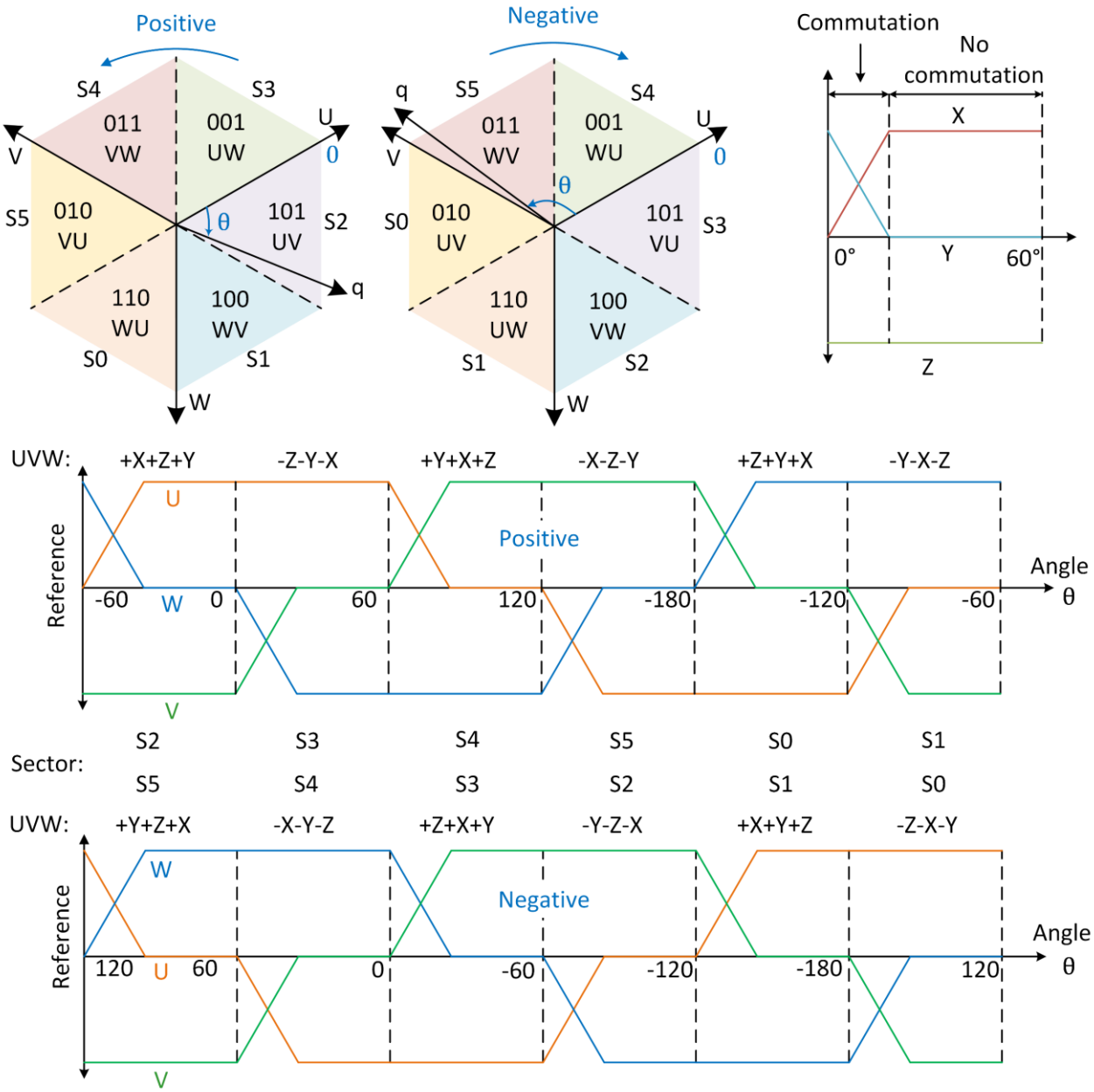

Fig. 48: Trapezoidal commutation currents, sectors, and sequences

For instance, in Fig. 48 when direction is positive, the q-axis direction vector represents the start of the commutation when the HALL sensor signal transitions from 100 to 101 (S1 to S2). At this transition, phase W has to reduce its current to zero and phase U has to pick up the current while phase V holds its current. In this scenario, phase U maps to X, W maps to Y, and phase V maps to Z. During the commutation period, C_X and C_Y control their associated phase currents to X and Y references a seen Fig. 48 while C_Z regulates the current of the phase associated with Z at a constant level. Fig. 48 also shows the phase mapping between UVW and XYZ in a full rotor rotation.

Note that the controller design for C_Z is similar to the PI controller in conventional block commutation while the controllers C_X and C_Y require higher bandwidth to achieve tight regulation during commutation. A good starting point to set C_X or C_Y bandwidth would be setting it the same as C_Z and increasing it to up to 5 times of C_Z gradually while looking at the waveforms during commutation. As shown in Fig. 49 the controller for these phases can also include a feedforward term to improve the performance of the controller. Assuming the motor resistance is negligible, the feedforward coefficient k_f can be set equal to the motor inductance L_M so the feedforward term can create the right amount of voltage to force the current follow the reference currents during commutation.

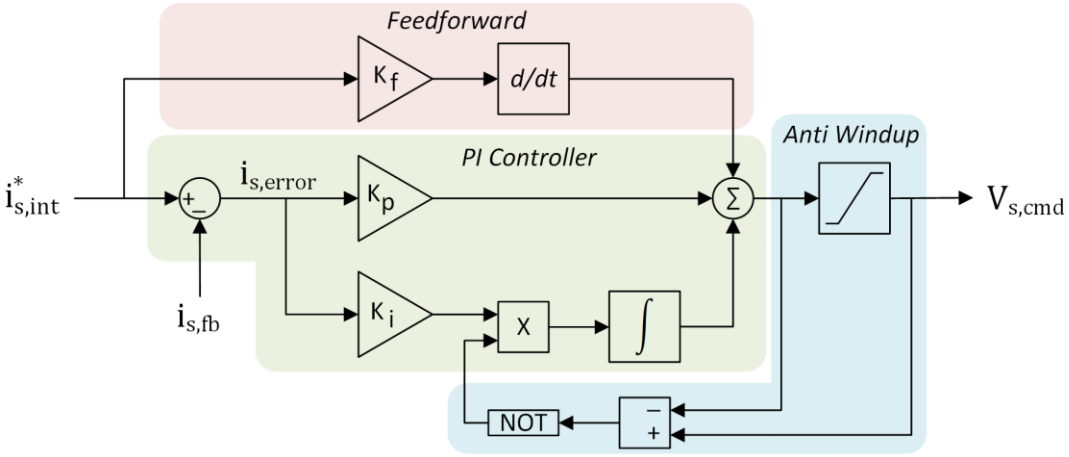

Fig. 49: Controller for X and Y phases

The XYZ to UVW block in Fig. 47 is responsible for mapping the trapezoidal control variables back to the UVW frame and generating the duty cycles for each phase to be employed by the PWM block.

5.5 Single-Shunt Current Reconstruction

In TBC, only two phases conduct current at any given time while the third phase is in high impedance state. Let us call the high impedance phase Z. Among the other two conducting phases, let us call the phase with a nonzero duty cycle X and the one with zero duty cycle Y. Fig. 50 depicts this convention.

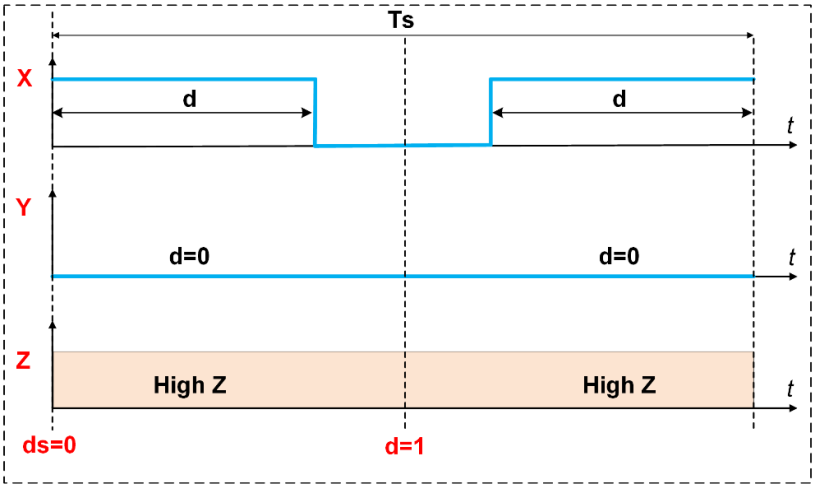

Fig. 50: $\{d_x, d_y, d_z\}$ definition ($d_x > d_y = 0, d_z = \text{High} - \text{Z}$), and sampling duty cycle $\{d_s\}$

As seen in Fig. 50, the sampling duty cycle for single shunt configuration in TBC should always be 0, since that is the middle point of the PWM signal for phase X. It should be noted that $d = 1$ corresponds to $T_s/2$ since center-asymmetric triangle modulation is used here. If two ADCs are set up to sample the current, they can both sample at the same time and the results can be averaged to reduce the noise:

$$\begin{cases} d_{s1} = 0 \\ d_{s2} = 0 \end{cases} \quad (221)$$

Then, the current feedback, i_s, can be reconstructed based on the measured i_x and the sign of applied voltage, v_s, as

$$i_s = \begin{cases} +i_x : v_s > 0 \\ -i_x : v_s < 0 \end{cases} \quad (222)$$

5.6 Experimental Results

The following figure shows the three-phase current waveforms when a BLDC motor is controlled using the trapezoidal commutation method described in section 5.4. As shown in this figure, the switching occurs during both current ramp up and ramp down which limits the diode conduction during commutation and reduces the power loss compared to traditional block commutation method. Furthermore, the phases involved in the commutation exchange the current smoothly and create almost no effect on the third phase's current. This ensures minimal commutation related torque ripple and acoustic noise in the motor.

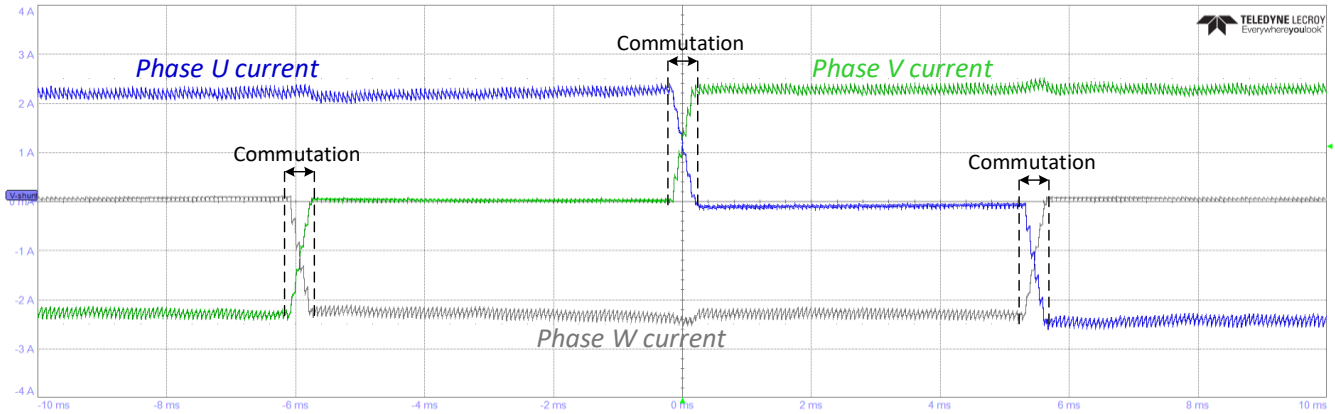

Fig. 51: Typical current waveforms using trapezoidal commutation method.

6 Common Modules

6.1 Biquad Filter

The Biquad filter is characterized by having two poles and two zeros, making it a second-order IIR filter. Both the zeros (numerator) and the poles (denominator) exhibit a flat frequency response at low frequencies. GUI can select the frequency of zeros and poles seen in (223). The biquad magnitude response will be the magnitude response of the numerator minus that of the denominator in logarithmic scale (Fig. 52). In Fig. 52, the denominator poles have lower frequency than the numerator zeros. The result is an anti-resonance low pass filter.

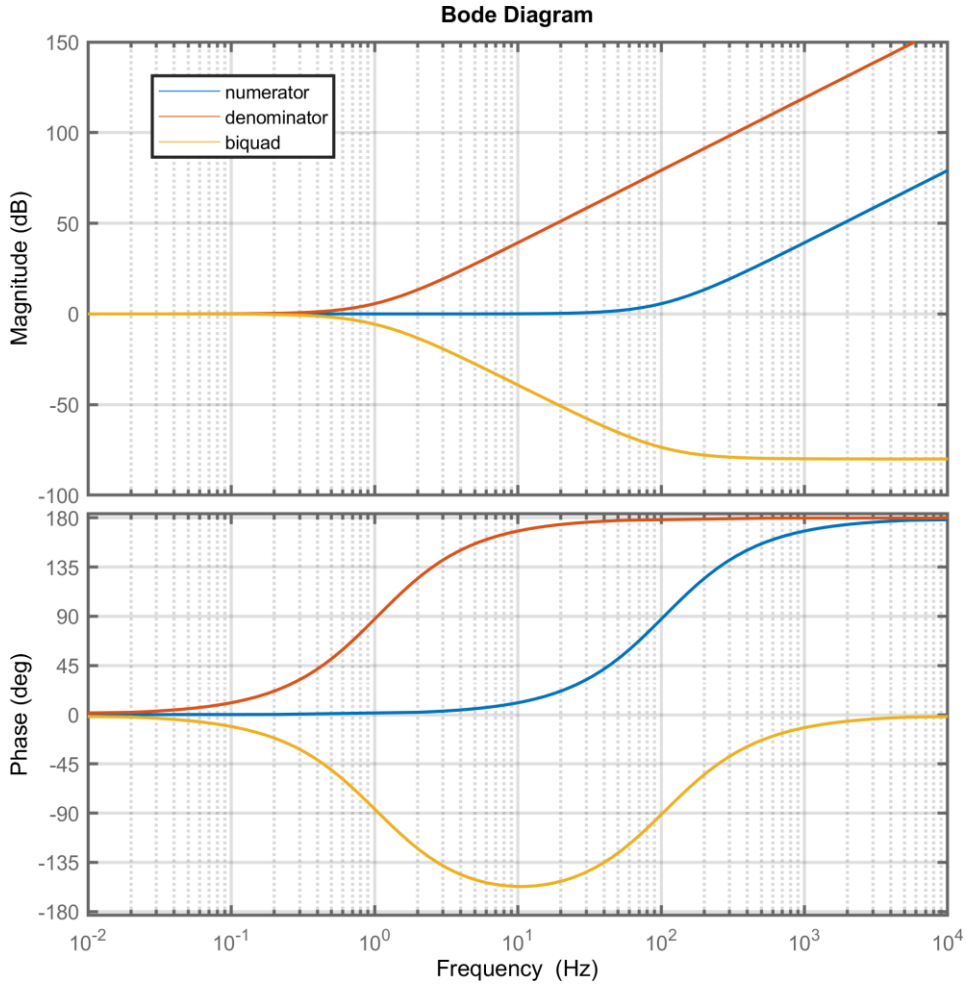

Fig. 52: A typical biquad filter's response

The transfer function of a Biquad filter with two poles and two zeros can be written as:

$$H(s)=K\frac{(s/\omega_{z1}+1)(s/\omega_{z2}+1)}{(s/\omega_{p1}+1)(s/\omega_{p2}+1)} \quad (223)$$

In order to digitally implement this filter, first (223) is rewritten in a standard quadratic polynomial form shown in (224):

$$H(s)=\frac{a_2s^2+a_1s+a_0}{b_2s^2+b_1s+b_0} \quad (224)$$

$$a_0=K,\ a_1=K(1/\omega_{z1}+1/\omega_{z2}),\ a_2=\frac{K}{\omega_{z1}\omega_{z2}}$$
$$b_0=1,\ b_1=(1/\omega_{p1}+1/\omega_{p2}),\ b_2=\frac{1}{\omega_{p1}\omega_{p2}} \quad (225)$$

To transform (224) into z-domain, (226) is incorporated into (224).

$$s=(1-z^{-1})f_s \quad (226)$$

The z-domain transfer function of the biquad filter now can be written as:

$$H(z)=\frac{m_0+m_1z^{-1}+m_2z^{-2}}{n_0+n_1z^{-1}+n_2z^{-2}} \quad (227)$$

$$m_0=a_0+a_1f_s+a_2{f_s}^2,\ m_1=-(a_1f_s+2a_2{f_s}^2),\ m_2=a_2{f_s}^2$$
$$n_0=b_0+b_1f_s+b_2{f_s}^2,\ n_1=-(b_1f_s+2b_2{f_s}^2),\ n_2=b_2{f_s}^2 \quad (228)$$

To simplify the firmware implementation, (228) can be transformed to an IIR filter of direct form II. This is done by rearranging (227) into (229) and then to (230):

$$H(z)=\frac{Y(z)}{X(z)},Y(z)=(m_0+m_1z^{-1}+m_2z^{-2})\left(\frac{X(z)}{n_0+n_1z^{-1}+n_2z^{-2}}\right) \quad (229)$$

$$D(z)=\left(\frac{X(z)}{n_0+n_1z^{-1}+n_2z^{-2}}\right),Y(z)=(m_0+m_1z^{-1}+m_2z^{-2})D(z) \quad (230)$$

Finally, Biquad filter implementation in the firmware will be based on (231).

$$\begin{cases} d_n=\frac{1}{n_0}(x_n-n_1d_{n-1}-n_2d_{n-2}) \\ y_n=m_0d_n+m_1d_{n-1}+m_2d_{n-2} \end{cases} \quad (231)$$

Fig. 53 shows the discrete time-domain block diagram of the biquad filter implemented in the firmware and normalized to have n_0=1.

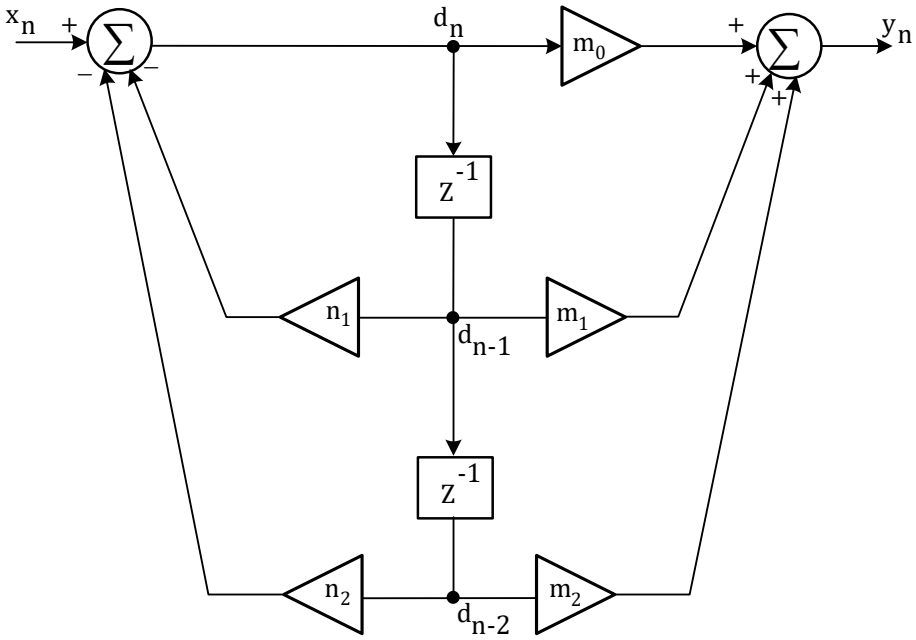

Fig. 53: The biquad filter as a discrete IIR filter of direct form II

6.2 Resonant Filter

There are three feedforward terms in the speed loop as shown in chapter 3.1, 4.1, and 5.1. For the feedforward term related to the inertia, the derivative of speed (acceleration) needs to be estimated. That is why this filter is also called acceleration estimator. In the firmware, a second-order resonant filter is utilized to estimate the acceleration, aiming to mitigate the potential impact of noise that could arise if a direct derivation method had been employed. The acceleration estimator is demonstrated below in Fig. 54 within the RFO speed loop. The same filter is also utilized in SFO and TBC speed loops.

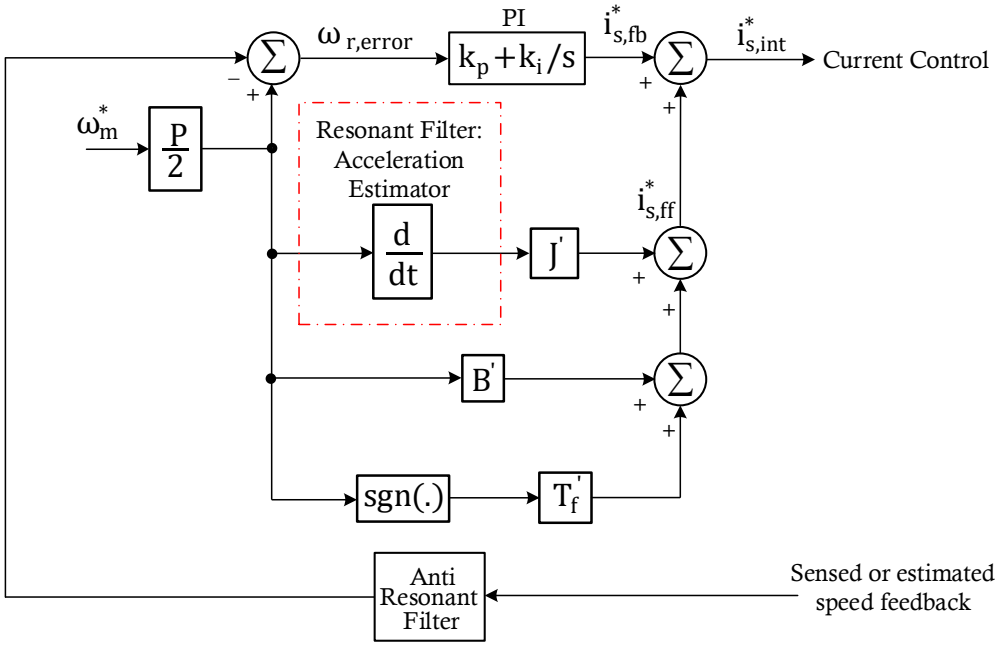

Fig. 54: Acceleration estimator filter in the RFO speed loop

The resonant filter is defined as:

$$H(s)=\frac{a_1 s}{s^2+b_1 s+b_0} \tag{232}$$

with the coefficients set as follows:

$$\begin{cases} a_1=\omega_0{}^2 \\ b_1=\omega_0 \\ b_0=\omega_0{}^2 \end{cases} \tag{233}$$

then

$$H(s)=\frac{\omega_0^2 s}{s^2+\omega_0 s+\omega_0^2} \tag{234}$$

in which

$$\zeta = 0.5, Q = 1 \tag{235}$$

where ζ is the damping factor and Q is the quality factor. Then the bandwidth of the filter will be

$$\omega_{BW}=\frac{\omega_0}{Q}=\omega_0 \tag{236}$$

For example, if the user sets the f_0 to be 1 Hz the resonant filter will look like Fig. 55. From Fig. 55 it is obvious that at lower frequency this filter works as differentiation block while at higher frequencies the gain drops sharply to attenuate the noise.

$$H(s) \overset{s\to 0}{\Longrightarrow} s \tag{237}$$

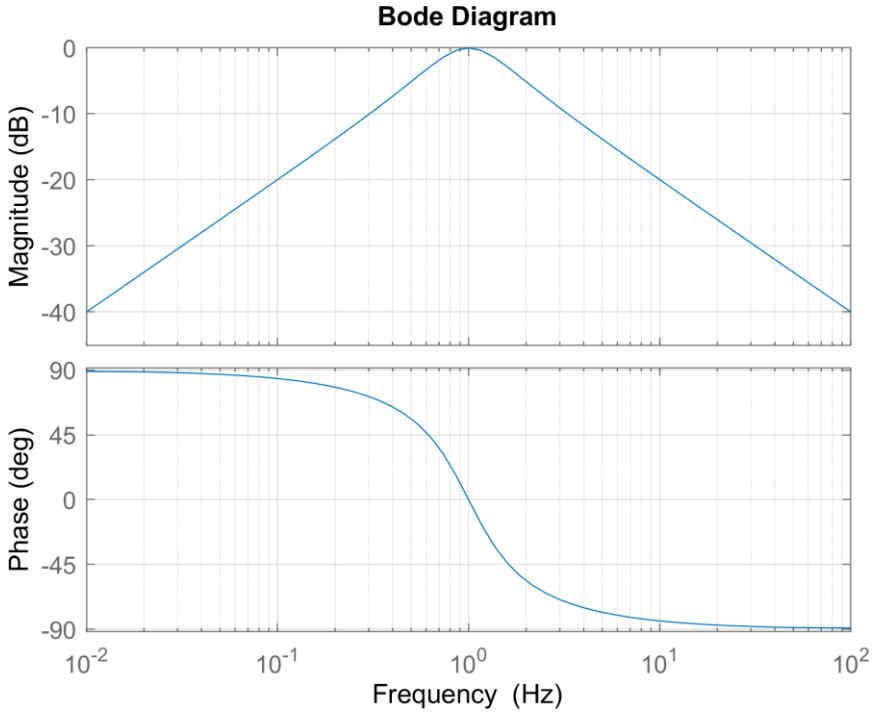

Fig. 55: A typical resonant filter acting as an acceleration estimator with $f_{BW} = 1\ Hz$.

6.3 Voltage Modulation

The objective of any motor control method is to generate a reference voltage to be applied to the motor windings using an inverter. Since the inverter is made of discrete switching devices, it cannot produce a continuous voltage on its output. Therefore, a modulation scheme is needed to create a voltage that has an average equal to the reference voltage. Fig. 56 illustrates the basics of modulation.

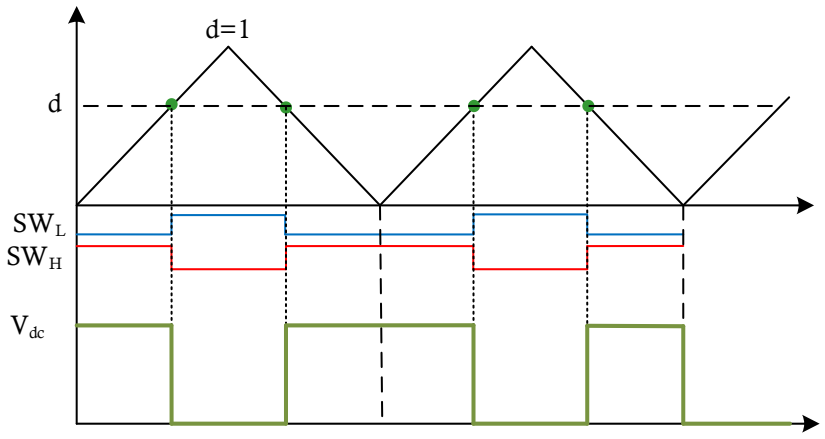

Fig. 56: Voltage modulation

As shown in Fig. 56, the reference voltage d generated by the controller is compared with a triangular waveform which creates the switching signals for low side SW_L and high side SW_H switches in the associated leg of inverter. The result would be the voltage waveform with an average of dV_{dc} in the leg of the inverter (with respect to negative DC bus).

In a three-phase inverter, the maximum voltage vector that can be generated depends on the modulation scheme. As it is shown in Fig. 57, the maximum voltage that can be generated by the inverter is $2/3 \times V_{dc}$ (with respect to the neutral point of the motor). Space Vector Modulation (SVM) and Neutral Point Modulation can both produce $1/\sqrt{3} \times V_{dc}$. These modulation schemes will be discussed in the following chapters.

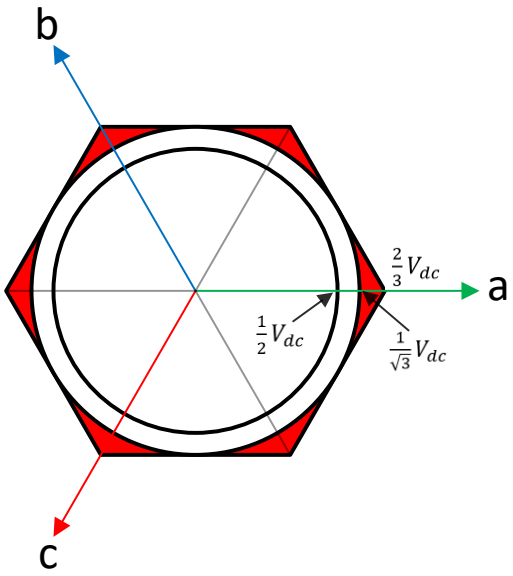

Fig. 57: Inverter voltage generation

6.3.1 Space Vector Modulation

In SVM method, the voltage reference is created using two adjacent voltage vectors shown in Fig. 58.

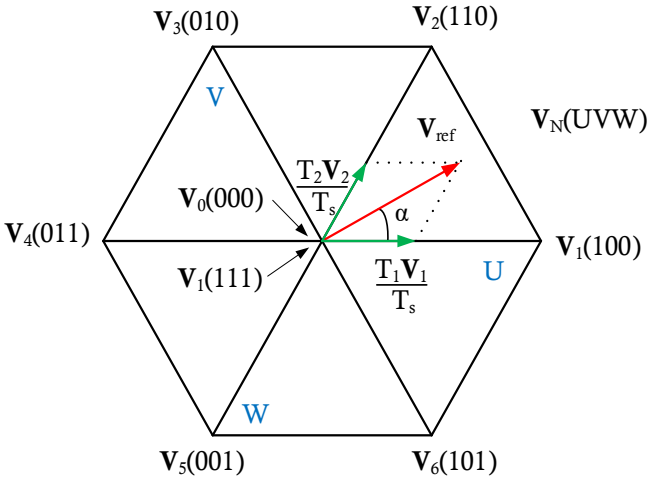

Fig. 58: SVM voltage creation

With a switching period of T_s, the reference voltage is created by applying $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ for T_1 and T_2 respectively and T_0 and T_7 for the rest of the period. Therefore,

$$\boldsymbol{V}_{\text{ref}} = \frac{T_0\boldsymbol{V}_0 + T_1\boldsymbol{V}_1 + T_2\boldsymbol{V}_2 + T_7\boldsymbol{V}_7}{T_s} = d_1\boldsymbol{V}_1 + d_2\boldsymbol{V}_2 \quad (238)$$

and the duration of each voltage vector can be calculated as

$$d_1 = \frac{V_{\text{ref}}}{V_{\text{dc}}/\sqrt{3}} \sin\left(\frac{\pi}{3} - \alpha\right) = \frac{\sqrt{3}}{2} m \sin\left(\frac{\pi}{3} - \alpha\right) \quad (239)$$

$$d_2 = \frac{V_{\text{ref}}}{V_{\text{dc}}/\sqrt{3}} \sin\alpha = \frac{\sqrt{3}}{2} m \sin\alpha$$

$$d_0 + d_7 = 1 - (d_1 + d_2)$$

where m is the modulation index and defined as

$$m = \frac{V_{\text{ref}}}{V_{\text{dc}}/2} \quad (240)$$

Fig. 59 shows the two different switching schemes for each leg of the inverter based on the timings acquired for each voltage vector. There are a few different switching schemes that can be used to create the vectors determined by the SVM method. However, 7 segment and 5 segment switching methods as shown in Fig. 59 are the most popular ($T_0 = T_7$).

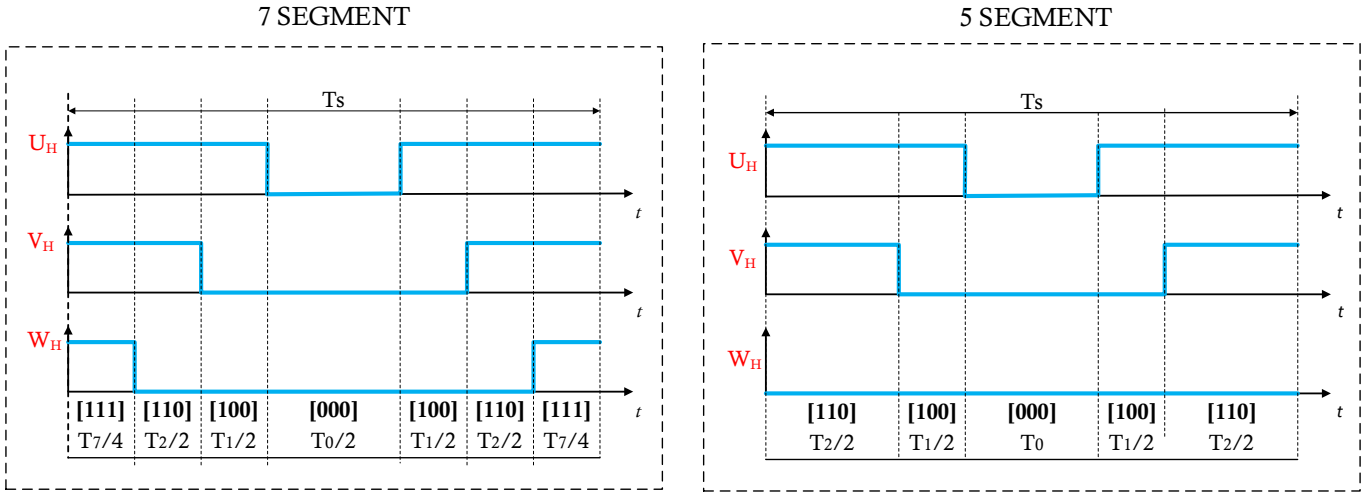

Fig. 59: SVM switching methods

The difference between the two switching methods is that in 7 segment, both $\boldsymbol{V}_0$ and $\boldsymbol{V}_7$ times are used to create zero voltage while in 5 segment only $\boldsymbol{V}_0$ is used to create zero voltage.

Note that the 5 segment switching method creates less switching compared to 7 segment which reduces the switching losses. However, the current ripple in this switching method is higher compared to 7 segment.

6.3.2 Neutral Point Modulation

Neutral point modulation is another method to increase the modulation index beyond 1.0 similar to SVM method. This modulation method can be used to increase modulation index to 1.15. Fig. 60 illustrates how this type of modulation work.

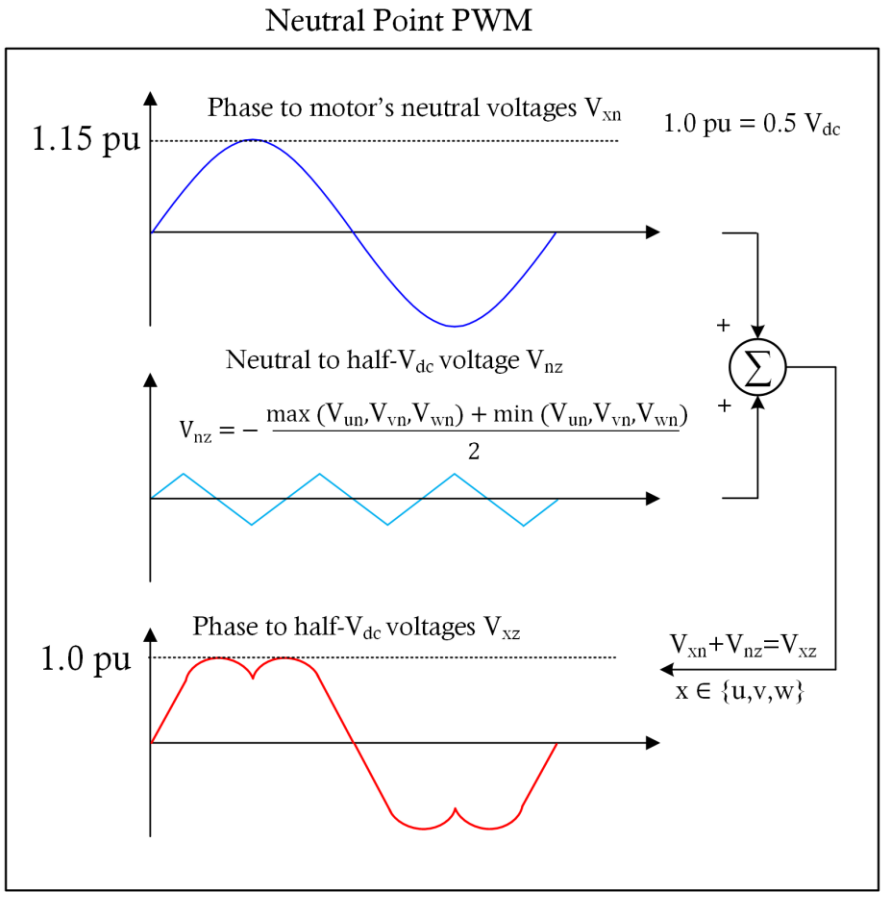

Fig. 60: Neutral point modulation scheme

In this picture V_{xn} is the phase reference voltage created by the controller. Note that this voltage is phase-to-neutral not phase-to-phase. V_{nz} is a common mode voltage that must be added to all the three phase voltage references. This voltage is calculated as,

$$V_{nz} = -\frac{\max(V_{un}, V_{vn}, V_{wn}) + \min(V_{un}, V_{vn}, V_{wn})}{2} \quad (241)$$

The voltages with respect half V_{dc} point of the inverter are denoted as V_{xz} as shown in Fig. 60.

As mentioned, this is a simple modulation strategy to increase the effective modulation index which creates results that are identical to those of SVM while being less computationally demanding. The switching can also be scheduled like SVM method according to Fig. 59.

6.4 Adaptive Tracking Loop

When using hall sensors, there are six possible permutations of hall patterns per one electrical revolution of the motor. As a result, there are six possible raw-angle estimates, θ^*, per electrical revolution. Similarly, the raw angle estimate for incremental encoders is also a discontinuous function. For example, for an incremental encoder with 512 Counts Per Revolution (CPR), there are 512 discrete raw angle estimates per mechanical revolution.

The goal of the adaptive tracking loop is to smooth out the discontinuous angle estimate curve to a more linear curve and estimate the angle of the rotor. Figure below depicts the block diagram of the fine-angle tracking loop.

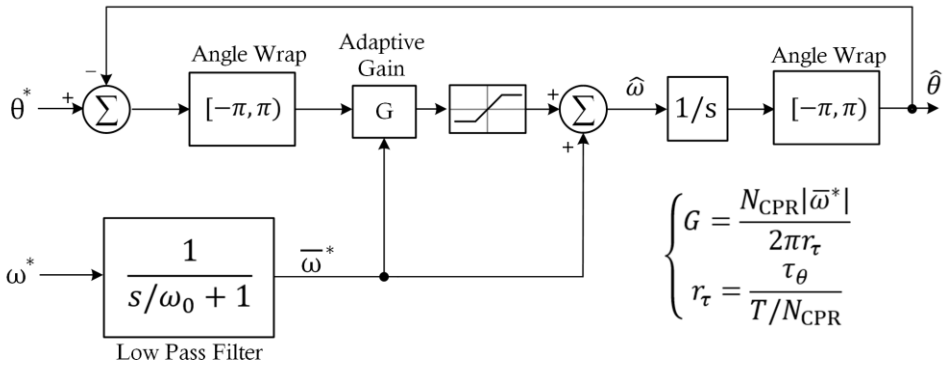

Fig. 61: Adaptive tracking loop's block diagram

In order to analyze the operation of the adaptive tracking loop, let us express the output of the loop, i.e. the estimated fine angle $\hat{\theta}$, in terms of the inputs, i.e. raw angle θ^* and speed feed forward $\bar{\omega}^*$, as follows:

$$\hat{\theta} = H_1(s)\theta^* + H_2(s)\bar{\omega}^* \tag{242}$$

Here, $H_1(s)$ can be written as

$$H_1(s) = \frac{G/s}{1+G/s} = \frac{1}{s/G+1} \tag{243}$$

Similarly, $H_2(s)$ can be written as

$$H_2(s) = \frac{1/G}{s/G+1} \tag{244}$$

Speed, ω^*, is calculated by capturing the elapsed time between two consecutive transitions, i.e. T^*/N_{CPR}. The speed feedforward term, $\bar{\omega}^*$, is then generated by applying a low pass filter, $H_3(s)$, on the captured speed, ω^*. We can express $\bar{\omega}^*$ as

$$\bar{\omega}^* = H_3(s)\theta^* \tag{245}$$

The transfer function $H_3(s)$ can be written as a derivative (s) and a low pass filter as

$$H_3(s) = \frac{s}{s/\omega_0+1} \tag{246}$$

Combining (242)-(246), the fine-angle estimate can be expressed as

$$\hat{\theta} = \theta^*\left(\frac{1}{s/G+1}\right)\left(1+\frac{s/G}{s/\omega_0+1}\right) \tag{247}$$

The angle error, θ_{error}, is

$$\theta_{\mathrm{error}} = \theta^* - \hat{\theta} \tag{248}$$

Therefore, it can be shown that

$$\theta_{\mathrm{error}}(s) = \frac{s^2/(G\omega_0)}{(s/G+1)(s/\omega_0+1)}\theta^*(s) \tag{249}$$

The steady-state error in response to a step input (i.e. a constant angle when the motor is stopped) must be zero. The Laplace transform of such step input is

$$\theta^*(t) = u(t) \Rightarrow \theta^*(s) = \frac{1}{s} \tag{250}$$

We can utilize the final value theorem to calculate this steady-state error based on the Laplace transform of the angle error as follows.

$$\lim_{t\to\infty}\theta_{\mathrm{error}}(t) = \lim_{s\to 0} s\cdot\theta_{\mathrm{error}}(s) = \lim_{s\to 0}\frac{s^2/(G\omega_0)}{(s/G+1)(s/\omega_0+1)} = 0 \tag{251}$$

The steady-state error in response to a ramp input (i.e. when the motor is spinning with constant speed) must also be zero. The Laplace transform of this ramp input is

$$\theta^*(t) = r(t) \Rightarrow \theta^*(s) = \frac{1}{s^2} \tag{252}$$

Utilizing the final value theorem again, we can calculate the steady-state error in response to the ramp input as follows:

$$\lim_{t\to\infty}\theta_{\text{error}}(t)=\lim_{s\to 0}s\cdot\theta_{\text{error}}(s)=\lim_{s\to 0}\frac{s/(G\omega_0)}{(s/G+1)(s/\omega_0+1)}=0 \tag{253}$$

Equation (253) shows the significance and the utility of the speed feedforward term. Without the speed feedforward term, $\bar{\omega}^*$, the steady-state error to ramp input would be nonzero since the closed loop system will turn into a first order loop. However, with the speed feedforward term, the loop gain will have two poles at zero which implies that, as shown in (253), the error response to both step input and ramp input would be zero in steady state.

The adaptive gain, G, can be chosen in such a way that the time constant of the tracking loop, τ_θ, is always proportional to the time duration of each transition step, T/N_{CPR}. If this requirement is satisfied, the adaptive tracking loop's time constant will adaptively change based on the motor speed such that it will always result in a smooth fine-angle estimation at all speeds. Let us define this ratio as

$$r_\tau=\frac{\tau_\theta}{T/N_{\text{CPR}}} \tag{254}$$

Thus,

$$\tau_\theta=r_\tau\frac{T}{N_{\text{CPR}}}=r_\tau\frac{2\pi/|\bar{\omega}^*|}{N_{\text{CPR}}}=\frac{2\pi r_\tau}{N_{\text{CPR}}|\bar{\omega}^*|} \tag{255}$$

Therefore, we can calculate the adaptive gain, G, as

$$G=\frac{1}{\tau_\theta}=\frac{N_{\text{CPR}}|\bar{\omega}^*|}{2\pi r_\tau} \tag{256}$$

Increasing r_τ will result in a smoother fine-angle estimation but will also make the tracking loop slower (since G becomes smaller). On the other hand, decreasing r_τ will result in a less smooth fine-angle estimation but also faster closed loop tracking. For typical applications, r_τ values between 0.8 and 1.0 is recommended.

6.5 Hall Sensor Module

6.5.1 Raw-Angle and Hall Patterns

Fig. 62 depicts the conventional way of placing hall sensors inside BLDC motors. BLDCs, as opposed to PMSMs, are constructed in a way that rotor flux is always perpendicular to the surface of the stator cylinder. This, in turn, makes the back emfs trapezoidal as opposed to sinusoidal.

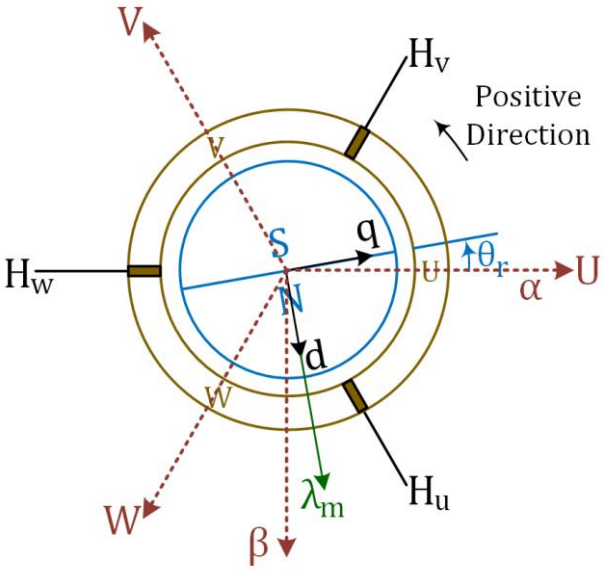

Fig. 62: Conventional hall sensor placement inside the BLDC motors

Here, the q-d reference frame and rotor angle conventions are the same as those defined for FOC methods (i.e. RFO and SFO) for consistency reasons. In addition, by keeping the same definitions, “sensored” FOC would also be seamlessly possible by changing the feedback from the sensorless observer module to the hall sensor module.

Each hall sensor shown in Fig. 62 outputs logic 1 when it is facing the rotor’s north pole and logic 0 when it is facing the rotor’s south pole. Therefore, the conventional placement of the hall sensors inside the BLDC, as shown in Fig. 62, results in the following truth table for the output hall signals as well as the raw angle estimation, θ^*:

Table 1: Hall patterns and their corresponding raw angle estimation, θ^*

hall u	1	1	0	0	0	1
hall v	0	1	1	1	0	0
hall w	0	0	0	1	1	1
[wvu]	001	011	010	110	100	101
raw angle θ^*	+30°	+90°	+150°	-150°	-90°	-30°

Typical back emfs $\{e_u, e_v, e_w\}$, phase currents $\{i_u, i_v, i_w\}$, hall patterns $[uvw]$, and raw angle estimates, θ^*, at positive speeds ($\omega_r > 0$) are shown in Fig. 63.

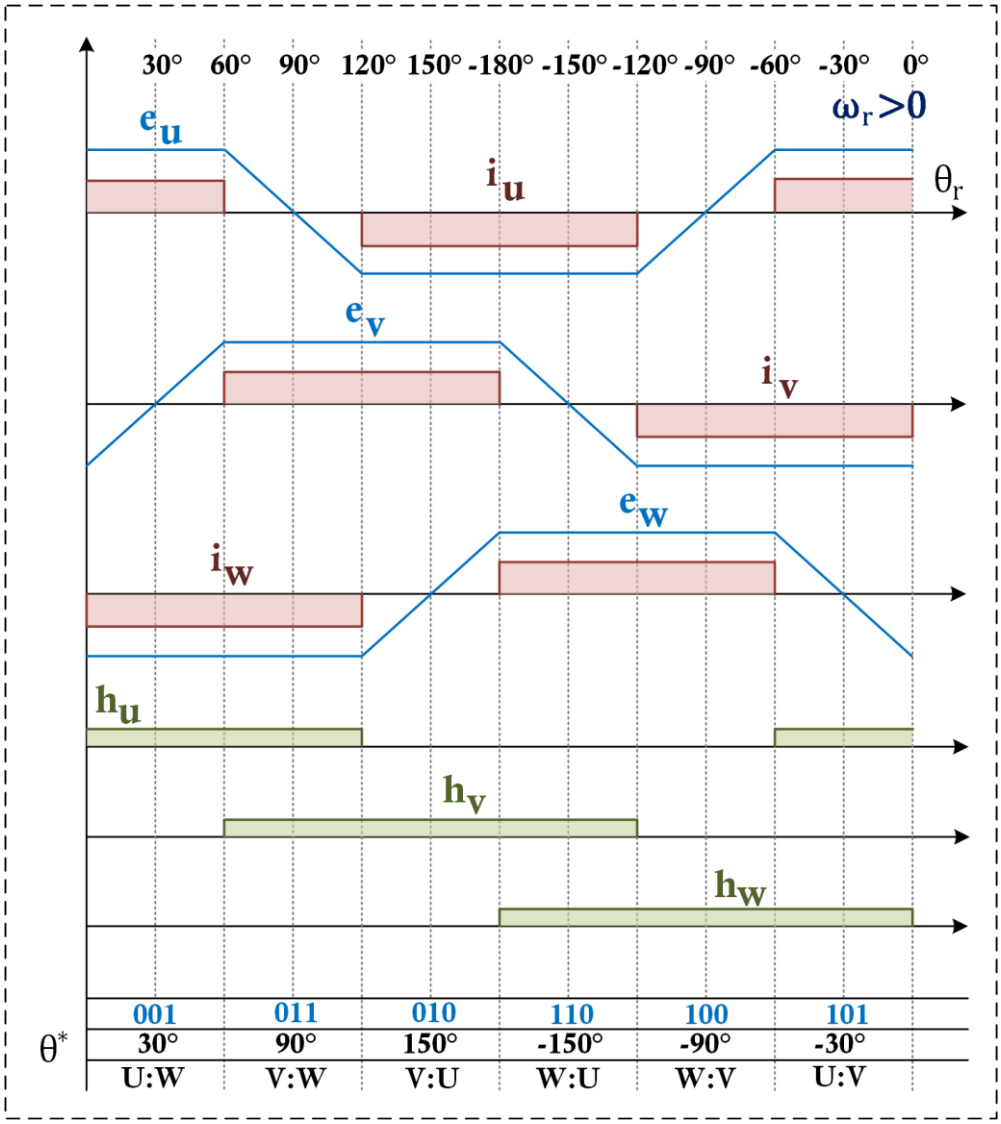

Fig. 63: Back emfs $\{e_u, e_v, e_w\}$, phase currents $\{i_u, i_v, i_w\}$, hall patterns $[uvw]$, and raw angle estimates, θ^* at positive speeds $\omega_r > 0$

Similar waveforms are shown in Fig. 64 for negative speeds ($\omega_r < 0$).

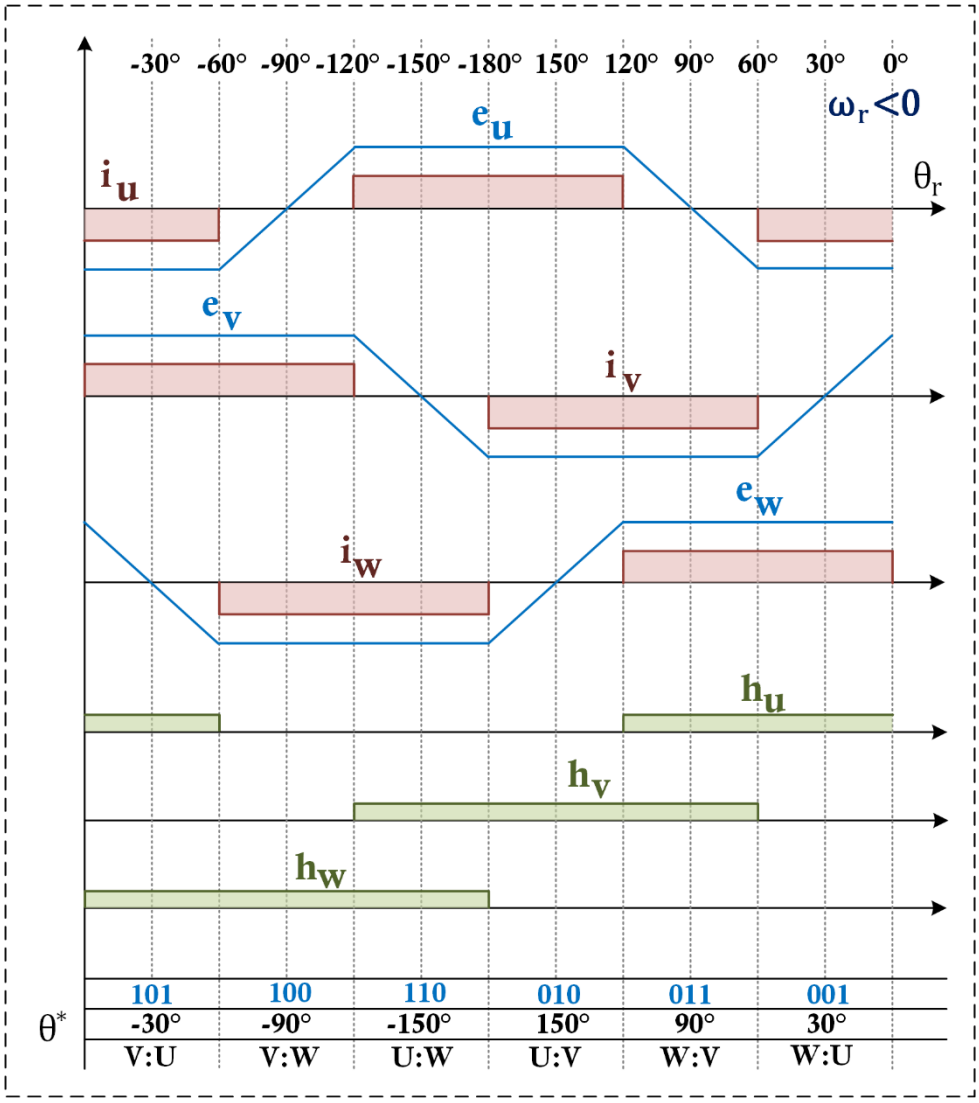

Fig. 64: Back emfs $\{e_u, e_v, e_w\}$, phase currents $\{i_u, i_v, i_w\}$, hall patterns $[uvw]$, and raw angle estimates, θ^* at negative speeds $\omega_r > 0$

By careful investigation, it can be seen that Fig. 63 will turn into Fig. 64 by reversing the x-axis direction (θ_r) and inverting the polarity of the back emf voltages and currents. This proves that the same hall module would work seamlessly for both positive and negative speed control as long as positive voltages ($v_s > 0$) are applied for positive speed and negative voltages ($v_s < 0$) are applied for negative speeds. Since the applied voltage is automatically determined by either the current control loop or the speed control loop (when bypassing the current loop), this requirement is already satisfied by the closed loop controller in the firmware.

6.5.2 Adaptive Tracking Loop

The raw angle estimates θ^* shown in Fig. 63 and Fig. 64 can be fed to the adaptive tracking loop described in section 6.4. The result would be a smooth angle estimate curve $\hat{\theta}$ that is suitable for sensored FOC.

The following figure depicts the overall block diagram of the hall sensor angle processing module.

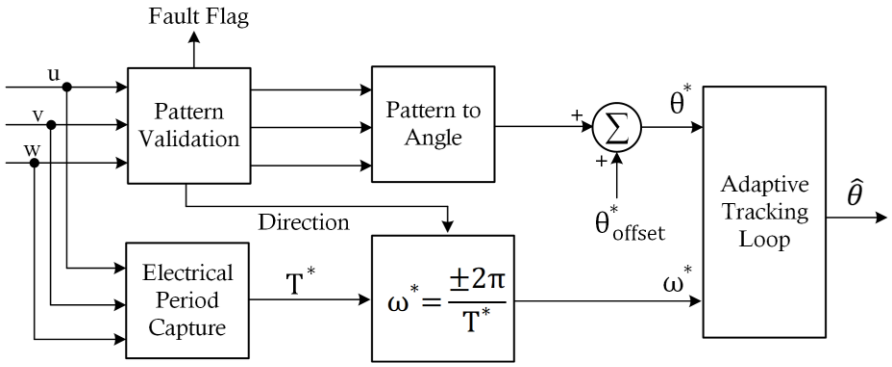

Fig. 65: Hall sensor module's block diagram

Fig. 66 shows typical raw angle (red), θ^*, and fine angle estimate (green), $\hat{\theta}$, waveforms for $r_\tau = 1$. As can be seen in Fig. 66, the fine-angle estimation is reasonably smooth and phase lag is minimal for $r_\tau = 1$.

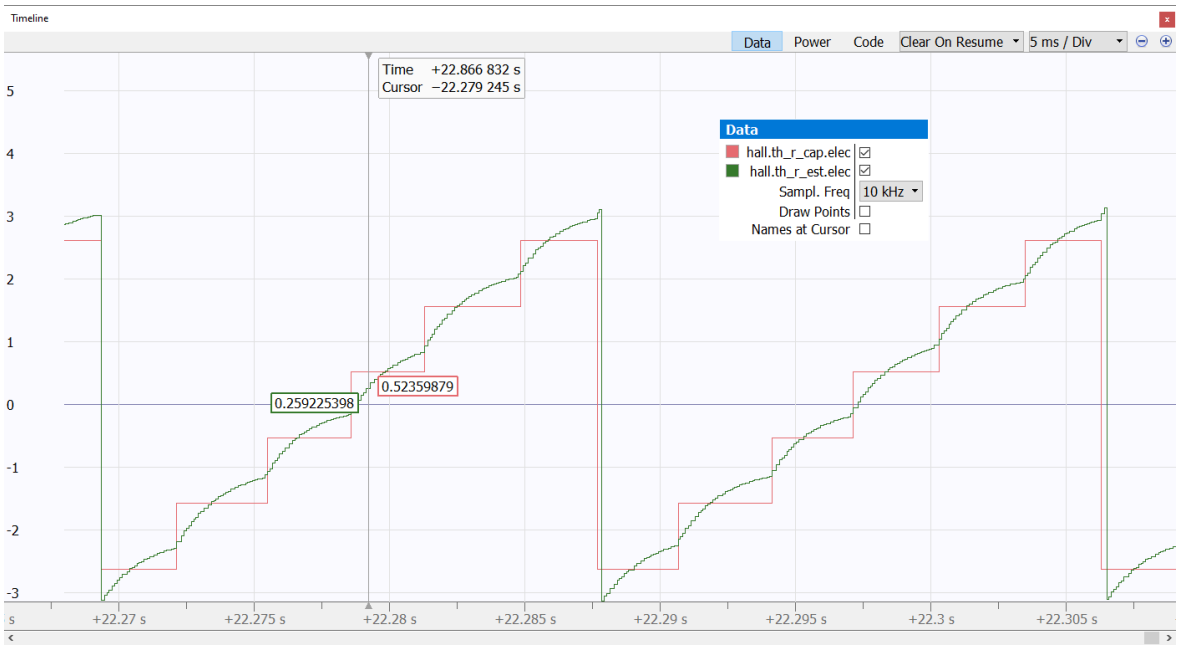

Fig. 66: Raw angle (red), θ^*, and fine angle estimate (green), $\hat{\theta}$, for $r_\tau = 1$

6.6 Incremental Encoder Module

The operation of incremental encoder module is similar to the hall sensor module since both will estimate the raw angle θ^* first and then feed it to the adaptive tracking loop to obtain the smooth estimated angle $\hat{\theta}$. The following figure depicts the incremental encoder module.

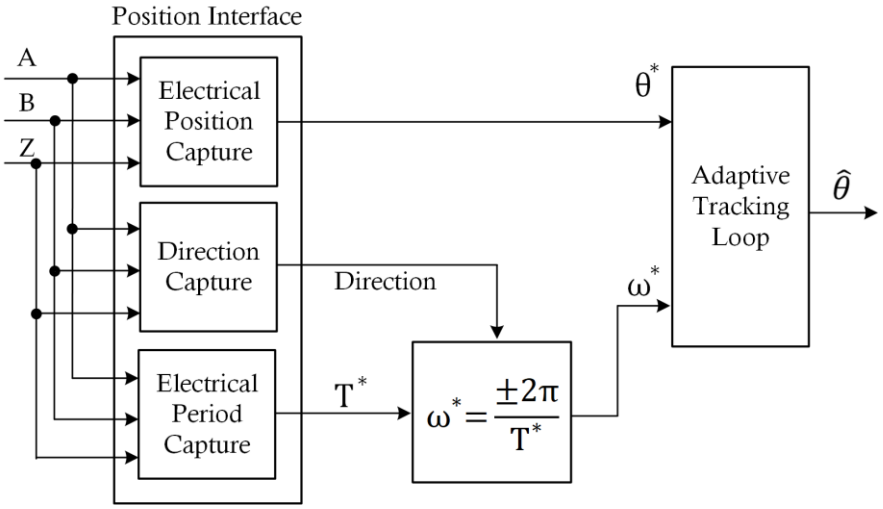

Fig. 67: Incremental encoder module's block diagram

The main difference between incremental encoder module and hall sensor module is that incremental encoders usually have higher precision (i.e. CPR) which is denoted by N_{CPR} in (254)-(256). In hall sensors N_{CPR} is 6 counts per electrical revolution, whereas in incremental encoders N_{CPR} can be up to 1024 or even 2048 counts per mechanical revolution.

6.7 Acceleration Estimator

As it was described in sections 3.1, 4.1 and 5.1 the speed control consists of some feedforward terms. For the inertia feedforward term, the acceleration needs to be estimated. The acceleration estimator is demonstrated in Fig. 54 within the RFO speed loop. In the firmware, a second-order resonant filter is utilized to estimate the acceleration, aiming to mitigate the potential impact of noise that could arise if a direct derivation method had been employed. The theory and implementation of this filter have been fully described in section 6.2.

6.8 Anti-Resonant Filter

As shown in Fig. 54, an anti-resonance filter is used in the speed controller. This is to notch out any oscillation that may be caused by the inertia mismatch between the motor and the load. A biquad filter that has a notch in the frequency range where speed oscillation is suspected can act as anti-resonant filter. The theory and implementation of biquad filters have been fully described in section 6.1.

6.9 Open-Loop V/F Controller

Open-loop V/F control, also referred to as scalar control or Volt/Hz control, is a widely used simple method for driving PMSMs. This technique operates on the principle that the ratio of applied voltage to the commanded synchronous speed remains relatively constant. Since there is no speed or current feedback involved, it is called open loop. The open loop control block diagram is shown in Fig. 68. The key is to properly calculate parameters K and V_{min} as will be described below.

The stator flux linkage can be written as:

$$\begin{cases} \lambda_q^r = L_q i_q^r \\ \lambda_d^r = L_d i_d^r + \lambda_m \end{cases} \tag{257}$$

From (257) the voltage equations of PMSM are:

$$\begin{cases} v_q^r = R_s i_q^r + \dfrac{d\lambda_q^r}{dt} + \omega_e \lambda_d^r \\ v_d^r = R_s i_d^r + \dfrac{d\lambda_d^r}{dt} - \omega_e \lambda_q^r \end{cases} \quad (258)$$

In steady state, flux linkage derivatives are zero. Resistor voltage drop can also be neglected in this control method. Then the voltages can be estimated as:

$$\begin{cases} v_q^r \simeq \omega_e \lambda_d^r \\ v_d^r \simeq -\omega_e \lambda_q^r \end{cases} \quad (259)$$

The total applied voltage is:

$$v_s = \sqrt{(v_q^r)^2 + (v_d^r)^2} = |\omega_e| \sqrt{(L_d i_d^r + \lambda_m)^2 + (L_q i_q^r)^2} \quad (260)$$

Assuming low currents, (260) will result in

$$\frac{v_s}{|\omega_e|} \simeq \lambda_m \quad (261)$$

which implies that in low torque conditions (current terms are negligible),

$$K \simeq \lambda_d^r \simeq \lambda_m \quad (262)$$

The rotor flux linkage, λ_m, can be obtained from motor datasheet or directly measured from the motor. However, the assumption in (262) will be less accurate with higher torque since v_s is also a function of the load current as shown in (260). This is why instead of simply assuming $K \simeq \lambda_m$ it is recommended to start running the motor in an open loop first with $K = \lambda_m \times 1.2$. Then run the motor at different speeds in closed-loop and measure the applied voltages to obtain a graph similar to Fig. 69. From this graph the intercept point is V_{min} and the slope will be deemed as the ratio K.

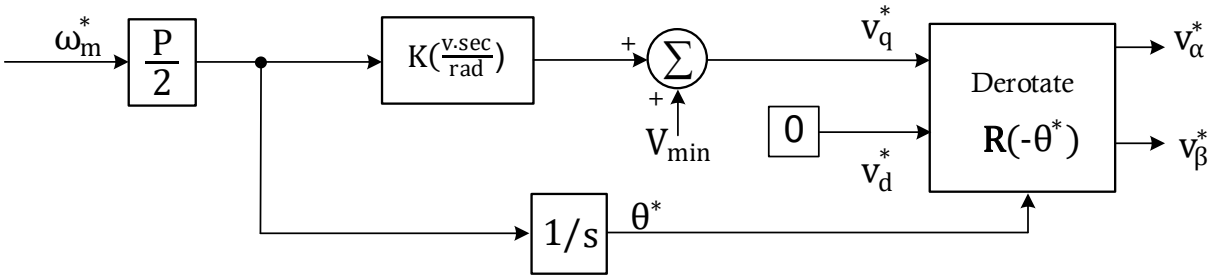

Fig. 68: Open loop V/F control block diagram

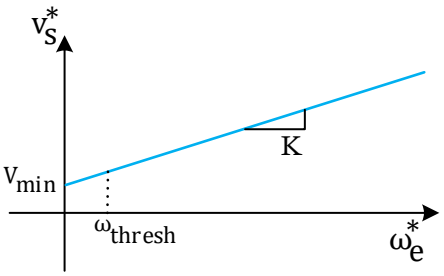

Fig. 69: Commanded voltage vs command speed in V/f control

6.10 Open-Loop I/F Controller

The current vector that is applied to the motor has a magnitude and an angle. In open-loop I/F control, the magnitude is controlled using a closed-loop controller and current feedbacks. However, the angle is determined in an open-loop fashion based on the commanded speed.

The following figure depicts the block diagram of the open-loop I/F controller.

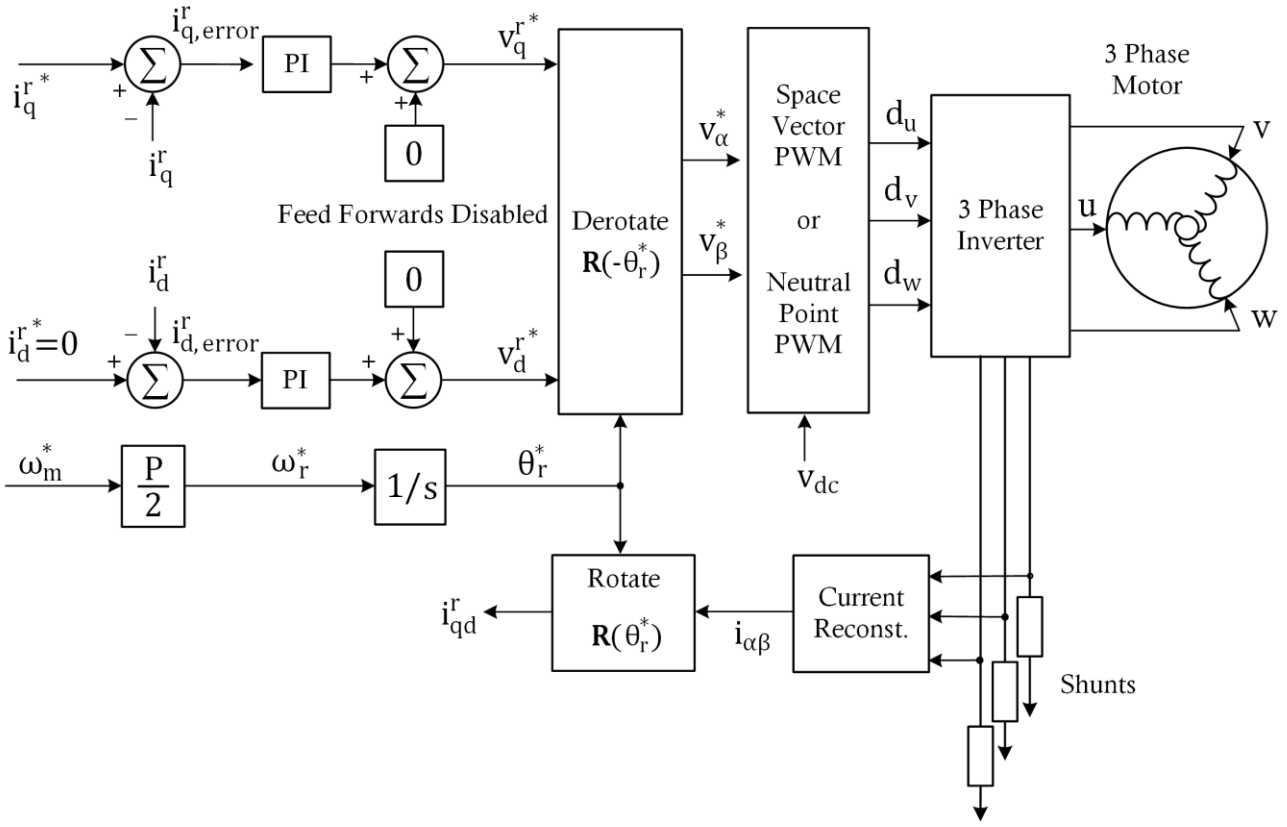

Fig. 70: Open loop I/F control block diagram

The user has to set the control parameter i_q^{r*} for open-loop I/F control. This parameter sets the magnitude of the current injected to the motor. The angle between the rotor flux and the stator current vector is determined by the load. It is important to set i_q^{r*} correctly because it determines the maximum torque the motor can deliver (when the angle between rotor flux and stator current is 90° in SPMs). The actual torque is dictated by the load as long as it is below the maximum torque achievable by i_q^{r*}. If the actual torque tries to go above the maximum achievable torque, the motor will lose synchronization and will stall.

Open-loop I/F has the following advantages compared to open-loop V/F control:

- V/F control is very sensitive to the control parameters K and $V_{\min}$ as described in section 6.9. In other words, if K and/or $V_{\min}$ are set too low, the motor won't start spinning and shake instead, and if K and/or $V_{\min}$ are set too high, the motor will start but there will be an overcurrent fault.
- I/F control is robust compared to V/F control, i.e. as long as i_q^{r*} is set such that maximum achievable torque is higher than the maximum load torque demand, the motor will never stall. In addition, if i_q^{r*} is not set higher than the peak current rating of the motor, no overcurrent fault will happen.
-

The disadvantage of open-loop I/F compared to open-loop V/F control is that it requires current sensors and a more complex controller implementation.

6.11 Sensorless Startup Methods

Sensorless startup for motor drive systems is generally challenging since the back-EMF observer method cannot be used to estimate the rotor position as there would not yet be any back-EMF built up at startup. If there is a high load torque in combination with lack of position estimation at startup, oscillations, current transients, noise and many other adverse effects are usually observed. However, there are a number of other methods that can be used to estimate the initial position of the rotor at start up as will be discussed in the next sections.

6.11.1 Rotor Pre-Alignment

In rotor pre-alignment method, a constant voltage is injected along with α axis of the motor as seen below.

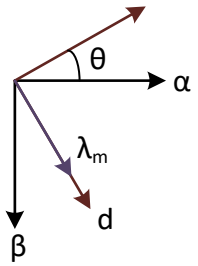

Fig. 71: Rotor pre-alignment method

The voltage applied along α axis creates a current vector aligned with this axis which forces the magnet on the rotor to align itself with this axis. Therefore, at the end of the pre-alignment d-axis and α axis would be in the same direction and the initial angle would be $\theta = 90^{\circ}$.

Note that the voltage applied to α axis, V_{α} needs to be high enough to produce the torque required to move the rotor and at the same time it shouldn't be too high to create over current and heat up the motor windings. Knowing the resistance R and the maximum current of the motor $I_{\max}$, the pre-alignment voltage can be calculated as follows as an initial estimate:

$$V_{\alpha,\mathrm{pre}} = RI_{\max} \tag{263}$$

6.11.2 Six Pulse Injection

Six pulse injection is a method of estimating the initial rotor angle in PMSMs. It relies on the inductance saturation effect. Therefore, it can only lead to reliable results in motors that show considerable saturation when moderate amount of current is applied to them (compared to their peak current rating).

6.11.2.1 Saturation Effect and Inductance

Fig. 72 depicts the inductance saturation effect as well as the space vector diagram and vector sequence used in rotor angle estimation procedure. Rotor angle estimation procedure will be explained in 6.11.2.4. This section describes how saturation affects the inductances.

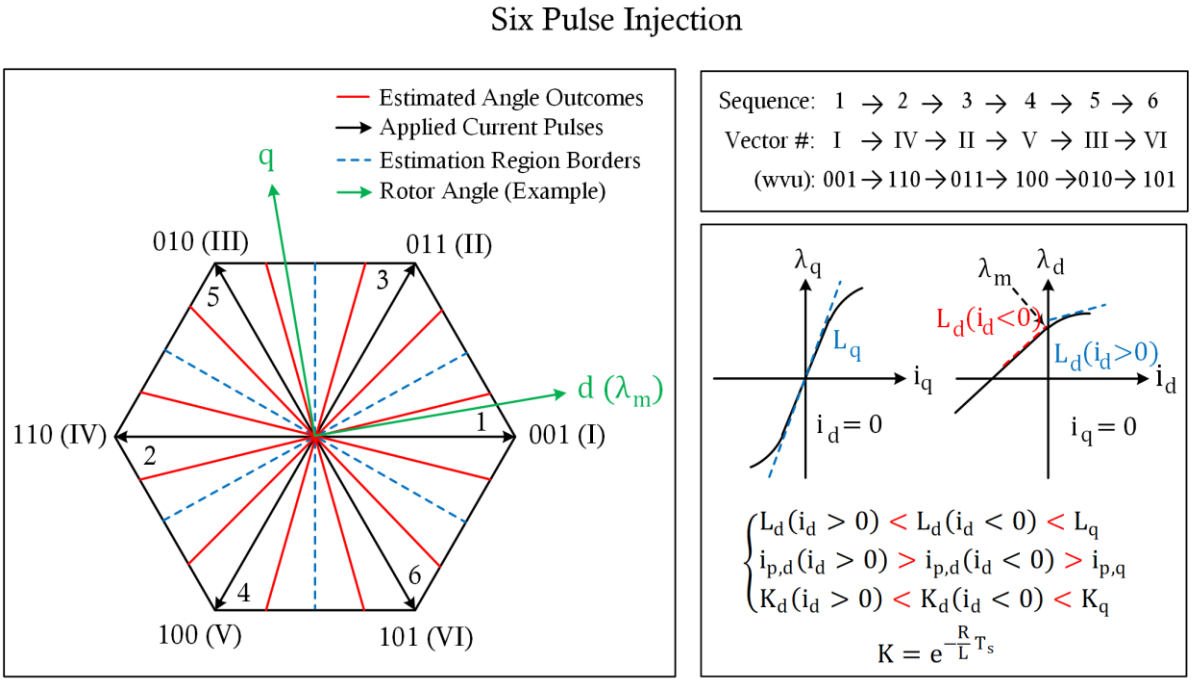

Fig. 72: Six pulse injection: inductance saturation effect, space vector diagram, and vector sequence

As can be seen in Fig. 72, the lowest inductance is observed when a current vector along the positive direction of the d-axis is applied to the motor, i.e. $L_d(i_d > 0) < L_d(i_d < 0) < L_q$. This is due to the fact that permanent magnet flux of the rotor, λ_m, which is aligned with the positive d-axis direction, always creates a flux offset that makes the flux vs. current curve seen in Fig. 72 closer to the saturation point compared to other directions. This difference in observed inductance along different directions can be utilized to estimate the initial rotor angle.

Differences in inductance will result in differences in the shape of the current going through the motor, namely, they affect the exponential decay rate of the current when no external voltage stimulus is applied, and also affect the exponential rise of the current when external voltage stimulus is applied.

6.11.2.2 Exponential Current Decay and Inductance

Assume a constant initial current, i_0, is flowing through the motor inductance. In absence of external voltage, the current curve can be expressed as

$$i(t) = i_0 e^{-\frac{r}{L}t} \tag{264}$$

Sampling the current in (264) by a discrete controller would result in

$$i_n = \underbrace{\left(e^{-\frac{r}{L}T_s}\right)}_{K}{}^n i_0 = K^n i_0 = K i_{n-1} \tag{265}$$

As can be seen in (265), the ratio between consequent samples, K, is affected by the inductance value, L. The higher the inductance value L, the higher this ratio, K, would be.

Fig. 73 illustrates a typical current exponential decay curve for a motor with $r = 20m\Omega, L = 10\mu H$, and sampling period of $T_s = 100\mu s$, which would result in $K = 0.82$.

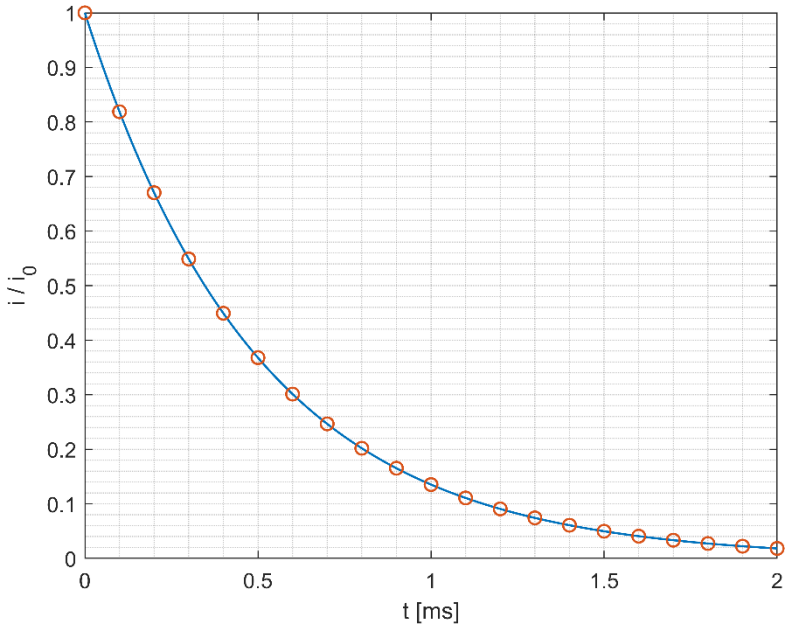

Fig. 73: Typical current exponential decay curve, $r = 20m\Omega, L = 10\mu H, T_s = 100\mu s, K = 0.82$

In order to increase the accuracy of the estimating K, multiple samples are taken similar to what is shown in Fig. 73. The best estimation of $\widehat{K}$ is calculated using the least squares method as follows.

$$\widehat{K} = \frac{\sum_{n=0}^{N} i_n i_{n+1}}{\sum_{n=0}^{N} i_n^2} \tag{266}$$

6.11.2.3 Exponential Current Rise and Inductance

When an external voltage stimulus is applied to two series R-L circuits (with $L_1 < L_2$) the current would rise according to

$$\begin{cases} i_1 = \frac{V}{R}\left(1 - e^{-\frac{r}{L_1}t}\right) \\ i_2 = \frac{V}{R}\left(1 - e^{-\frac{r}{L_2}t}\right) \end{cases}, L_1 < L_2 \tag{267}$$

Each of the applied pulses seen in Fig. 72 (I through VI) correspond to a switching state where the phase-to-neutral voltage of one phase of the motor is twice that of the other two phases. Thus, if V in (267) is the line-to-line voltage of the motor when applying the six pulses, the effective resistance, R, would be

$$R = \frac{3}{2}r \tag{268}$$

where r is the line-to-neutral resistance of the motor.

From (267), it can be seen that

$$\begin{cases} i_1(\infty) = i_2(\infty) = \frac{V}{R} = \frac{2V/3}{r} \\ \frac{di_1}{dt}(0) = \frac{2V/3}{L_1} > \frac{di_2}{dt}(0) = \frac{2V/3}{L_2} \end{cases} \tag{269}$$

In other words, two series R-L circuits with different inductance values will have the same steady-state current values, $i_1(\infty) = i_2(\infty)$, but one current rises faster than the other, $\frac{di_1}{dt}(0) > \frac{di_2}{dt}(0)$. This difference in current rise slope can be employed to determine which circuit has a higher inductance value.

Fig. 74 shows typical current exponential rise curves for two different inductance values, $L_1 = 5\mu H$ and $L_2 = 20\mu H$.

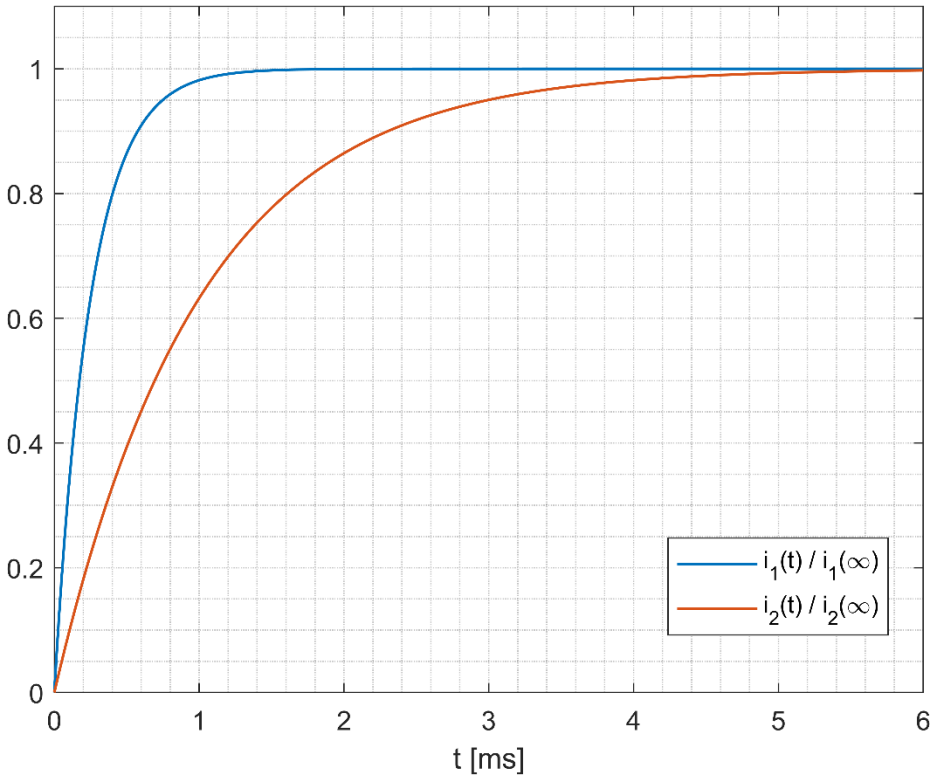

Fig. 74: Typical current exponential rise curves, $r = 20m\Omega, L_1 = 5\mu H, L_2 = 20\mu H$

As can be seen in Fig. 74, the difference between the currents is zero initially and also in steady-state. The current difference can be expressed as

$$i_1 - i_2 = \frac{-V}{R}\left(e^{-\frac{r}{L_1}t} - e^{-\frac{r}{L_2}t}\right) \tag{270}$$

For accuracy reasons, it is desired to sample the currents at a time when this difference, $i_1 - i_2$, is maximum. This optimum sampling time can be calculated by setting

$$\frac{d(i_1 - i_2)}{dt} = 0 \tag{271}$$

It can be shown that applying (271) to (270) would result in

$$t_{\text{optimal}} = \left(\frac{k \cdot \ln(k)}{k-1}\right)\tau_1 = \left(\frac{\ln(k)}{k-1}\right)\tau_2 = \left(\frac{2k \cdot \ln(k)}{k^2-1}\right)\bar{\tau} \tag{272}$$

where

$$\begin{cases} \tau_1 = \dfrac{L_1}{r} \\ \tau_2 = \dfrac{L_2}{r} \\ \bar{\tau} = \dfrac{\tau_1 + \tau_2}{2} \\ k = \dfrac{L_2}{L_1} \end{cases} \tag{273}$$

When using six-pulse injection method, only average time constant $\bar{\tau}$ is available since we don't know how much the inductances will saturate, i.e. what is the exact value of k. Nevertheless, the effect of k on t_{optimal} is insignificant for a wide range of k values as seen in Fig. 75.

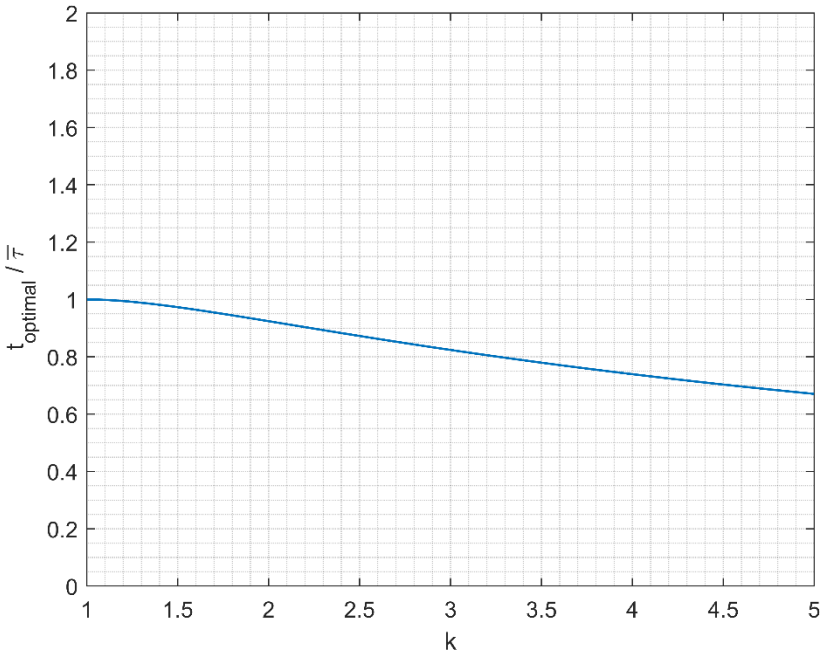

Fig. 75: Optimal pulse on-time, t_{optimal}, vs. ratio of max to min inductance, k

Therefore, it can be safely assumed that using the following choice of pulse on-time, t_{on}, would result in a near optimal situation where the difference between currents is around its maximum.

$$t_{\text{on}} = \bar{\tau} \tag{274}$$

The choice of pulse off-time, t_{off}, on the other hand, must be long enough compared to the time constant, $\bar{\tau}$, to allow for currents to go back to zero before the next pulse is applied:

$$t_{\text{off}} = 10\bar{\tau} \tag{275}$$

The applied stimulus voltage (line-to-line) must be calculated such that the peak current doesn't go above a certain prescribed threshold by the user, i_p. This peak current threshold, i_p, must be selected carefully such that it is not too high to move the rotor during the six-pulse injection procedure or too low to be able to create observable saturation. From (274), i_p can be expressed as

$$i_p = \frac{2V/3}{r}\left(1 - e^{-\frac{t_{\text{on}}}{\bar{\tau}}}\right) = \frac{2V/3}{r}(1 - e^{-1}) \tag{276}$$

Therefore, for a given desired i_p, stimulus voltage, V, can be calculated as

$$V = \frac{3}{2(1 - e^{-1})} r i_p \tag{277}$$

6.11.2.4 Rotor Angle Estimation Procedure

The rotor angle estimation procedure seen in Fig. 72 utilizes both methods described in 6.11.2.2 (exponential decay) and 6.11.2.3 (exponential rise).

First, the six pulses are applied according to the vector sequence shown in Fig. 72. This sequence is designed to maximized the angle difference between consequent pulses so that the average applied torque is minimized and rotor doesn't move during the estimation procedure. However, careful selection of i_p is of paramount importance to ensure rotor doesn't move.

After the six-pulse injection sequence is finished, the results are processed to determine which pulse (denoted by index i) resulted in the highest peak current, $i_{p,i}$, and lowest decay rate, $\hat{k}_i$. Then, among the pulses adjacent to pulse i (i.e. either $i+1$ or $i-1$), the algorithm looks for the pulse (denoted by index j) with the highest peak current, $i_{p,j}$, and lowest decay rate, $\hat{k}_j$:

$$\begin{cases} i_{p,i} > i_{p,j} \\ \hat{k}_i < \hat{k}_j \\ j = i \pm 1 \end{cases} \quad (278)$$

Once i and j are determined, the rotor angles corresponding to i and j can be calculated as

$$\begin{cases} \hat{\theta}_{\lambda_m,i} = (i-1)\frac{\pi}{3} \\ \hat{\theta}_{\lambda_m,j} = (j-1)\frac{\pi}{3} \end{cases} \quad (279)$$

The flux angle of the rotor is somewhere between pulse vector i and pulse vector j. In addition, the flux angle must be closer to pulse vector i than it is to pulse vector j. Therefore, the best estimation for flux angle, $\hat{\theta}_{\lambda_m}$, and rotor angle, $\hat{\theta}$, would be

$$\begin{cases} \hat{\theta}_{\lambda_m} = \frac{3}{4}\hat{\theta}_{\lambda_m,i} + \frac{1}{4}\hat{\theta}_{\lambda_m,j} \\ \hat{\theta} = \hat{\theta}_{\lambda_m} + \frac{\pi}{2} \end{cases} \quad (280)$$

It must be noted that the indices i and j must be unwrapped before using (279)-(280), e.g. $\{i = 6, j = 1\} \Rightarrow \{i = 6, j = 7\}$.

6.11.3 High Frequency Injection: Sine Wave

For IPMs with significant saliency ($\xi \gg 1$), inductance along the rotor d-axis is significantly lower than that of the q-axis. This difference in inductance can be utilized to estimate the rotor's electrical angle in such motors. Fig. 76 depicts the inductance observed along the α axis versus rotor angle for a typical IPM with $\xi = 1.22$.

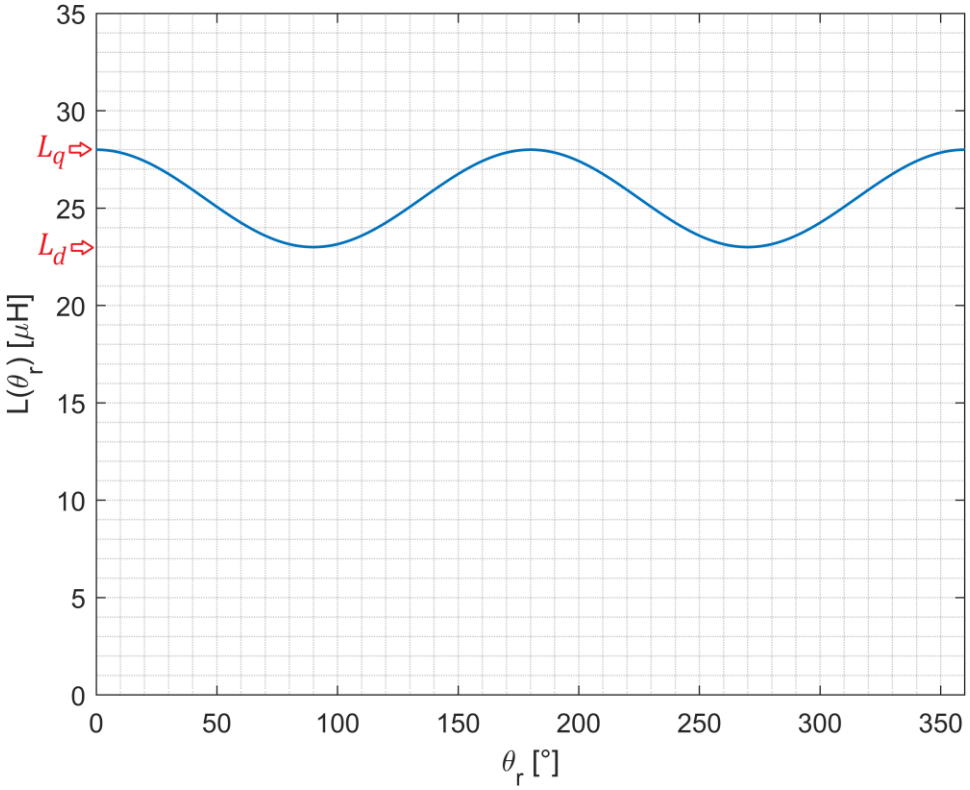

Fig. 76: Inductance versus rotor angle for a typical IPM with $L_q = 28\mu\text{H}$ and $L_d = 23\mu\text{H}$

As can be seen in Fig. 76, $L(\theta_r + \pi) = L(\theta_r)$. This is always the case for all IPMs. Therefore, there is always $\pm 180°$ ambiguity which should be resolved by using another method e.g. six pulse injection. In

other words, six pulse injection method can be used in the beginning to estimate the initial rotor angle with $\pm 15°$ accuracy. Then this preliminary estimate can be used as a seed to initialize the tracking loop in the high frequency injection module and the tracking loop will then converge to the nearest stable quiescent point which would be the correct rotor angle with $0°$ error. More information on this topic will be presented in section 6.11.3.3.

6.11.3.1 RFO

The angle-dependent inductance (Fig. 76) can be measured by injecting high-frequency voltage components to the motor and analyzing the resulting currents produced by the motor. The goal of this section is to present a framework for injecting the high frequency voltages, separating the resulting high-frequency currents from the low-frequency currents used to produce torque, and implementing an observer loop that can estimate both rotor angle and rotor speed.

The stationary reference frame voltages in PMSMs can be described as

$$\boldsymbol{v}_{\alpha\beta} = r\boldsymbol{i}_{\alpha\beta} + \frac{d\boldsymbol{\lambda}_{\alpha\beta}}{dt} + \omega\lambda_m \begin{bmatrix} \cos(\theta_r) \\ -\sin(\theta_r) \end{bmatrix} \tag{281}$$

where the flux linkages, $\boldsymbol{\lambda}_{\alpha\beta}$, depend on the currents and the rotor angle, θ_r, as follows.

$$\boldsymbol{\lambda}_{\alpha\beta} = (L_0\boldsymbol{I} + L_1\boldsymbol{C}\boldsymbol{R}(2\theta_r))\boldsymbol{i}_{\alpha\beta} \tag{282}$$

In (282), $\boldsymbol{I}$, $\boldsymbol{C}$, and $\boldsymbol{R}$ are identity, complex-conjugate, and rotation matrices, respectively:

$$\begin{cases} L_q = L_0 + L_1 \\ L_d = L_0 - L_1 \end{cases}, \boldsymbol{C} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \boldsymbol{R}(2\theta_r) = \begin{bmatrix} \cos(2\theta_r) & -\sin(2\theta_r) \\ \sin(2\theta_r) & \cos(2\theta_r) \end{bmatrix} \tag{283}$$

As seen in (283), L_0 and L_1 are used instead of L_q and L_d to simplify the derivations in this context. We can also describe the currents, $\boldsymbol{i}_{\alpha\beta}$, in terms of the flux linkages, $\boldsymbol{\lambda}_{\alpha\beta}$, as

$$\begin{aligned} \boldsymbol{i}_{\alpha\beta} &= \left(L_0\boldsymbol{I} + L_1\boldsymbol{C}\boldsymbol{R}(2\theta_r)\right)^{-1}\boldsymbol{\lambda}_{\alpha\beta} \\ &= \frac{1}{\underbrace{L_0^2 - L_1^2}_{L_qL_d}}\left(L_0\boldsymbol{I} - L_1\boldsymbol{C}\boldsymbol{R}(2\theta_r)\right)\boldsymbol{\lambda}_{\alpha\beta} \end{aligned} \tag{284}$$

It must be noted that the combination of the complex-conjugate matrix $\boldsymbol{C}$ and the rotation matrix $\boldsymbol{R}$ has the following property, which will be useful later on in this context for other derivations:

$$\boldsymbol{C}\boldsymbol{R}(2\theta_r) = \boldsymbol{R}(-2\theta_r)\boldsymbol{C} \tag{285}$$

Here, we will use both the matrix representation of the vectors and their corresponding complex number representation:

$$\begin{cases} \boldsymbol{x}_{\alpha\beta} = \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} & \Leftrightarrow & \boldsymbol{x}_{\alpha\beta} = x_\alpha - jx_\beta \\ \boldsymbol{x}_{qd} = \begin{bmatrix} x_q \\ x_d \end{bmatrix} & \Leftrightarrow & \boldsymbol{x}_{qd} = x_q - jx_d \\ \boldsymbol{x}_{qd} = \boldsymbol{R}(\hat{\theta}_r)\boldsymbol{x}_{\alpha\beta} & \Leftrightarrow & \boldsymbol{x}_{qd} = \boldsymbol{e}^{-j\hat{\theta}_r}\boldsymbol{x}_{\alpha\beta} \\ \boldsymbol{x}_{\alpha\beta} = \boldsymbol{R}(-\hat{\theta}_r)\boldsymbol{x}_{qd} & \Leftrightarrow & \boldsymbol{x}_{\alpha\beta} = \boldsymbol{e}^{+j\hat{\theta}_r}\boldsymbol{x}_{qd} \\ \boldsymbol{C}\boldsymbol{x}_{qd} & \Leftrightarrow & \boldsymbol{x}_{qd}^* \\ \boldsymbol{C}\boldsymbol{R}(2\theta_r)\boldsymbol{x}_{qd} = \boldsymbol{R}(-2\theta_r)\boldsymbol{C}\boldsymbol{x}_{qd} & \Leftrightarrow & \left(\boldsymbol{e}^{-j2\theta_r}\boldsymbol{x}_{qd}\right)^* = \boldsymbol{e}^{+j2\theta_r}\boldsymbol{x}_{qd}^* \end{cases} \tag{286}$$

This is done to simplify the derivations since in some cases complex-number representations and in some other cases the matrix representations are more straightforward. For example, the matrix properties in (285) are more obvious when using the complex number representations in (286).

We must also distinguish between θ_r, $\hat{\theta}_r$, and $\tilde{\theta}_r$ when analyzing the high frequency injection technique since the estimated, $\hat{\theta}_r$, and real rotor angle, θ_r, values are not necessarily the same and the goal is to drive the error, $\tilde{\theta}_r$, to zero:

$$\begin{cases} \theta_r & : \text{real rotor electrical angle} \\ \hat{\theta}_r & : \text{estimated rotor electrical angle} \\ \tilde{\theta}_r = \theta_r - \hat{\theta}_r & : \text{rotor electrical angle error} \end{cases} \quad (287)$$

Equation (281) describes the motor voltages and currents in terms of the actual rotor angle, θ_r. Now, let us apply a Park transform using an estimated rotor angle, $\hat{\theta}_r$, to (281) to obtain

$$\boldsymbol{v}_{qd}^{\hat{\theta}_r} = r\boldsymbol{i}_{qd}^{\hat{\theta}_r} + \frac{d\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r}}{dt} + j\hat{\omega}_r\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r} + \omega\lambda_m\boldsymbol{e}^{j\tilde{\theta}_r} \quad (288)$$

As can be seen in (288), the angle error, $\tilde{\theta}_r$, only shows up in the back emf terms, $\omega\lambda_m\boldsymbol{e}^{j\tilde{\theta}_r}$, which is expected because in the correct rotational reference frame ($\tilde{\theta}_r = 0$), the back emfs should appear to be constants.

Similarly, the flux linkage equations in (282) can be transformed a rotational reference frame with the estimated angle $\hat{\theta}_r$ as

$$\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r} = L_0\boldsymbol{i}_{qd}^{\hat{\theta}_r} + L_1\boldsymbol{e}^{j2\tilde{\theta}_r}\left(\boldsymbol{i}_{qd}^{\hat{\theta}_r}\right)^* \quad (289)$$

As can be seen in (289), the angle error, $\tilde{\theta}_r$, only shows up in $L_1\boldsymbol{e}^{j2\tilde{\theta}_r}$ term, which is expected because in the correct rotational reference frame ($\tilde{\theta}_r = 0$), the inductances of q and d axis should appear to be constants (L_q and L_d).

With the analysis framework set forth in (281)-(289), let us assume that the following high frequency voltages are injected into the motor:

$$\boldsymbol{v}_{qd}^{\hat{\theta}_r} = \begin{bmatrix} V_q \cdot \sin(\theta_h) \\ V_d \cdot \cos(\theta_h) \end{bmatrix} \Leftrightarrow \quad (290)$$

$$\boldsymbol{v}_{qd}^{\hat{\theta}_r} = V_q \sin(\theta_h) - jV_d \cos(\theta_h)$$

It is convenient to express the voltages, $\boldsymbol{v}_{qd}^{\hat{\theta}_r}$, as a superposition of positive sequence and negative sequence components and compute the effect of each sequence separately on the flux linkages, $\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r}$, i.e.

$$\boldsymbol{v}_{qd}^{\hat{\theta}_r} = -j\left(V_+\boldsymbol{e}^{+j\theta_h} + V_-\boldsymbol{e}^{-j\theta_h}\right) \quad (291)$$

By equating (290) and (291), the magnitudes of positive and negative sequences, V_+ and V_-, can be calculated in terms of V_q and V_d as follows:

$$\begin{cases} V_+ = \dfrac{V_q + V_d}{2} \\ V_- = \dfrac{V_d - V_q}{2} \end{cases} \Leftrightarrow \begin{bmatrix} V_+ \\ V_- \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix} \begin{bmatrix} V_q \\ V_d \end{bmatrix} \quad (292)$$

Now, let us compute the effect of each voltage sequence on the flux linkages separately and use the superposition principle again to compute the total flux linkage response to the voltages, i.e.

$$\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r} = \Lambda_+\boldsymbol{e}^{+j\theta_h} + \Lambda_-\boldsymbol{e}^{-j\theta_h} = \quad (293)$$

$$(\Lambda_+ + \Lambda_-)\cos(\theta_h) - j(\Lambda_- - \Lambda_+)\sin(\theta_h)$$

where

$$\begin{cases} \Lambda_q = \Lambda_+ + \Lambda_- \\ \Lambda_d = \Lambda_- - \Lambda_+ \end{cases} \Leftrightarrow \begin{bmatrix} \Lambda_q \\ \Lambda_d \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \Lambda_+ \\ \Lambda_- \end{bmatrix} \quad (294)$$

Considering that the effect of resistance term is negligible in the total voltages described in (288) and the back emf term $\omega\lambda_m\boldsymbol{e}^{j\tilde{\theta}}$ in (288) is a low frequency component (due to $\tilde{\theta}$) which gets filtered out by a frequency separation filter, it can be shown that the flux-linkage response to the injected high frequency voltages is as follows:

$$\begin{cases} \Lambda_+ = \dfrac{-V_+}{\omega_h + \hat{\omega}_r} \\ \Lambda_- = \dfrac{V_-}{\omega_h - \hat{\omega}_r} \end{cases} \Leftrightarrow \begin{bmatrix} \Lambda_+ \\ \Lambda_- \end{bmatrix} = \begin{bmatrix} \dfrac{-1}{\omega_h + \hat{\omega}_r} & 0 \\ 0 & \dfrac{1}{\omega_h - \hat{\omega}_r} \end{bmatrix} \begin{bmatrix} V_+ \\ V_- \end{bmatrix} \tag{295}$$

Consequently,

$$\begin{bmatrix} \Lambda_q \\ \Lambda_d \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \Lambda_+ \\ \Lambda_- \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \dfrac{-1}{\omega_h + \hat{\omega}_r} & 0 \\ 0 & \dfrac{1}{\omega_h - \hat{\omega}_r} \end{bmatrix} \begin{bmatrix} V_+ \\ V_- \end{bmatrix} \tag{296}$$

$$= \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \dfrac{-1}{\omega_h + \hat{\omega}_r} & 0 \\ 0 & \dfrac{1}{\omega_h - \hat{\omega}_r} \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix} \begin{bmatrix} V_q \\ V_d \end{bmatrix}$$

which is equivalent to

$$\begin{bmatrix} \Lambda_q \\ \Lambda_d \end{bmatrix} = \frac{1}{\omega_h^2 - \hat{\omega}_r{}^2} \begin{bmatrix} -\omega_h & \hat{\omega}_r \\ -\hat{\omega}_r & \omega_h \end{bmatrix} \begin{bmatrix} V_q \\ V_d \end{bmatrix} \tag{297}$$

Equation (297) describes the flux linkage response (Λ_q and Λ_d) to the injected high-frequency voltages (V_q and V_d). Now, let us inject the following high-frequency voltages (V_q and V_d) to the motor:

$$\begin{bmatrix} V_q \\ V_d \end{bmatrix} = \begin{bmatrix} \dfrac{\hat{\omega}_r}{\omega_h} V_h \\ V_h \end{bmatrix} \tag{298}$$

where V_h is a constant voltage magnitude. Fig. 77 depicts these high-frequency voltage components together with the other components of the high frequency injection block diagram.

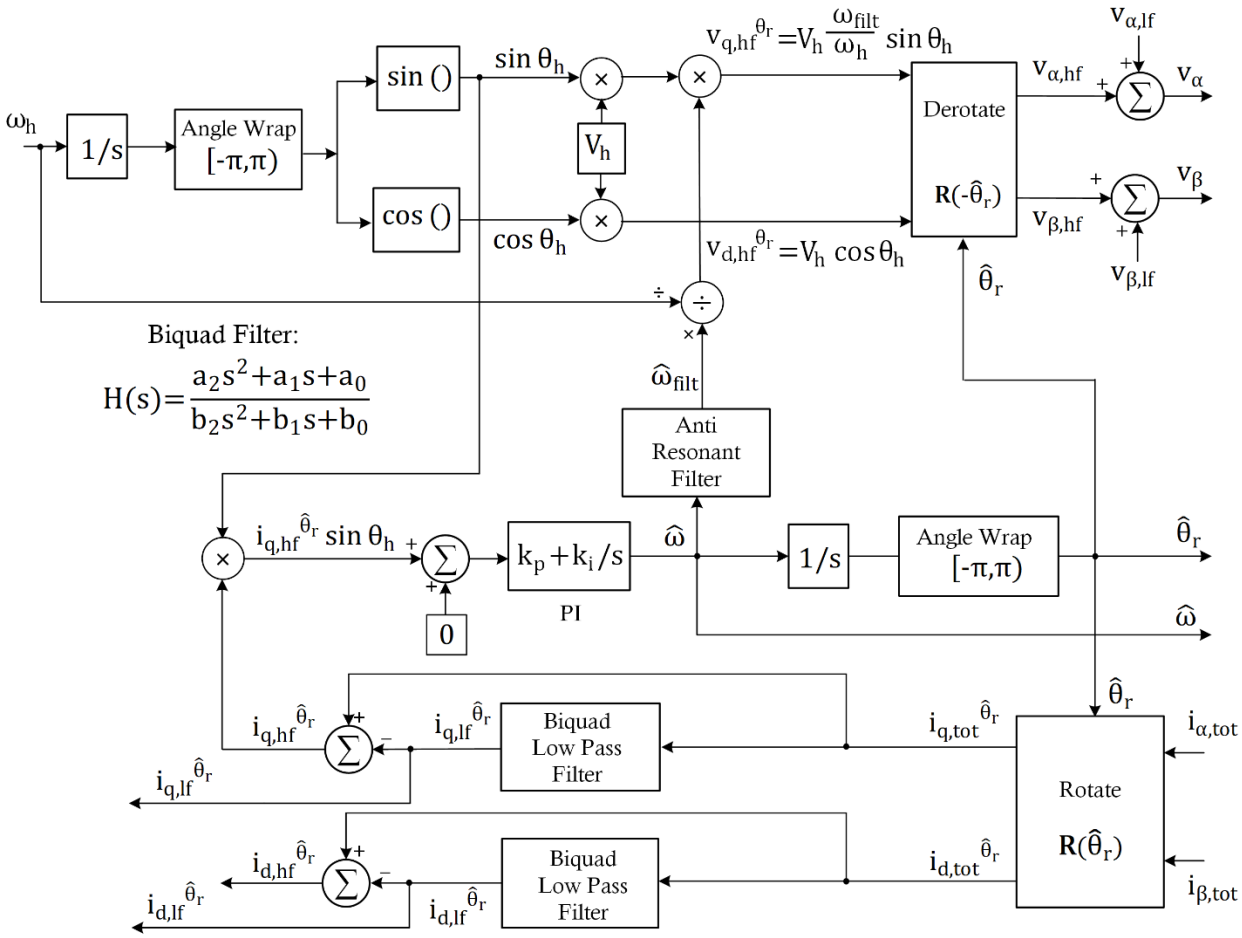

Fig. 77: High-frequency-injection block diagram in RFO

By substituting (298) into (297), we can find the resulting flux linkage response:

$$\begin{bmatrix}\Lambda_q\\ \Lambda_d\end{bmatrix} = \frac{1}{\omega_h^2 - \hat{\omega}_r{}^2}\begin{bmatrix}-\omega_h & \hat{\omega}_r\\ -\hat{\omega}_r & \omega_h\end{bmatrix}\begin{bmatrix}\hat{\omega}_r\\ \omega_h\end{bmatrix}\frac{V_h}{\omega_h} \tag{299}$$
$$= \frac{V_h}{\omega_h}\begin{bmatrix}0\\ 1\end{bmatrix}!$$

The significance of (299) is that it shows injecting the high frequency voltages described in (298) will result in a flux linkage response that is completely aligned with the d-axis. In other words,

$$\boldsymbol{v}_{qd}^{\hat{\theta}_r} = \begin{bmatrix}\frac{\hat{\omega}_r}{\omega_h}V_h \cdot \sin(\theta_h)\\ V_h \cdot \cos(\theta_h)\end{bmatrix} \Rightarrow \boldsymbol{\lambda}_{qd}^{\hat{\theta}_r} = \begin{bmatrix}0\\ \frac{V_h}{\omega_h} \cdot \sin(\theta_h)\end{bmatrix} \tag{300}$$

The flux linkages, $\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r}$, cannot be directly measured but the currents can. Therefore, we must express the currents, $\boldsymbol{i}_{qd}^{\hat{\theta}_r}$, in terms of $\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r}$ as

$$\boldsymbol{i}_{qd}^{\hat{\theta}_r} = \frac{1}{L_q L_d}\left(L_0\boldsymbol{I} - L_1\boldsymbol{C}\boldsymbol{R}(2\tilde{\theta}_r)\right)\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r} \tag{301}$$

Then, by substituting $\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r}$ from (300), the current responses can be obtained as follows:

$$\boldsymbol{i}_{qd}^{\hat{\theta}_r} = \frac{V_h \sin(\theta_h)}{L_q L_d \omega_h}\begin{bmatrix}L_1 \sin(2\tilde{\theta}_r)\\ L_0 + L_1 \cos(2\tilde{\theta}_r)\end{bmatrix} \tag{302}$$

Equation (302) is the core of high-frequency injection method. It shows that

- The q-axis current, $i_q^{\widehat{\theta}_r}$, has the angle error information, $\tilde{\theta}_r$, embedded in it.
- The q-axis current, $i_q^{\widehat{\theta}_r}$, is directly proportional to L_1. Therefore, for SPMs with $L_1 = 0$, the high-frequency injection method cannot be used.
- Both q-axis and d-axis currents, $\boldsymbol{i}_{qd}^{\widehat{\theta}_r}$, are amplitude modulated similar to an AM radios due to $\sin(\theta_h)$ term. Therefore, demodulation by multiplication with the carrier frequency is necessary to bring the signals to the base band as depicted in Fig. 77. The demodulation multiplier will create two frequency components, one at dc and one at twice the carrier frequency. Here, the tracking loop will act as a low pass filter to eliminate the twice carrier frequency component. Hence, no additional filtering is necessary, which is desirable because external low pass filters can create phase lag in the estimate rotor angle.
- The angle error term $\sin(2\tilde{\theta}_r)$ has four quiescent points, two of which are stable quiescent points because the slope is positive in their vicinity as seen in Fig. 78.

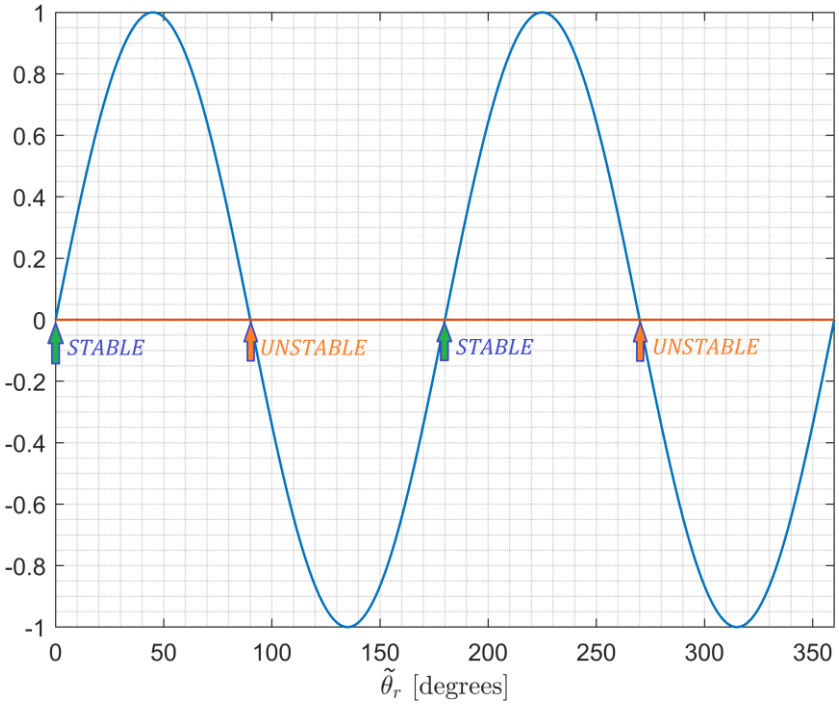

Fig. 78: Stable and unstable quiescent points in the high-frequency-injection tracking loop due to $\sin(2\tilde{\theta}_r)$

Since there are two stable points, one of which is around $\tilde{\theta}_r = 180°$, the system can have $\pm 180°$ error as also explained in section 6.11.3. This $180°$ ambiguity can be eliminated by using the six-pulse injection method in the beginning as a seed to initialize the tracking loop so that it converges to the correct stable point.

Fig. 79 illustrates the typical waveforms of voltages, currents, estimated rotor angle, and estimated rotor speed in RFO, as well as their transients. The motor is not spinning in Fig. 79. However, the algorithm presented here also works well when the motor is spinning.

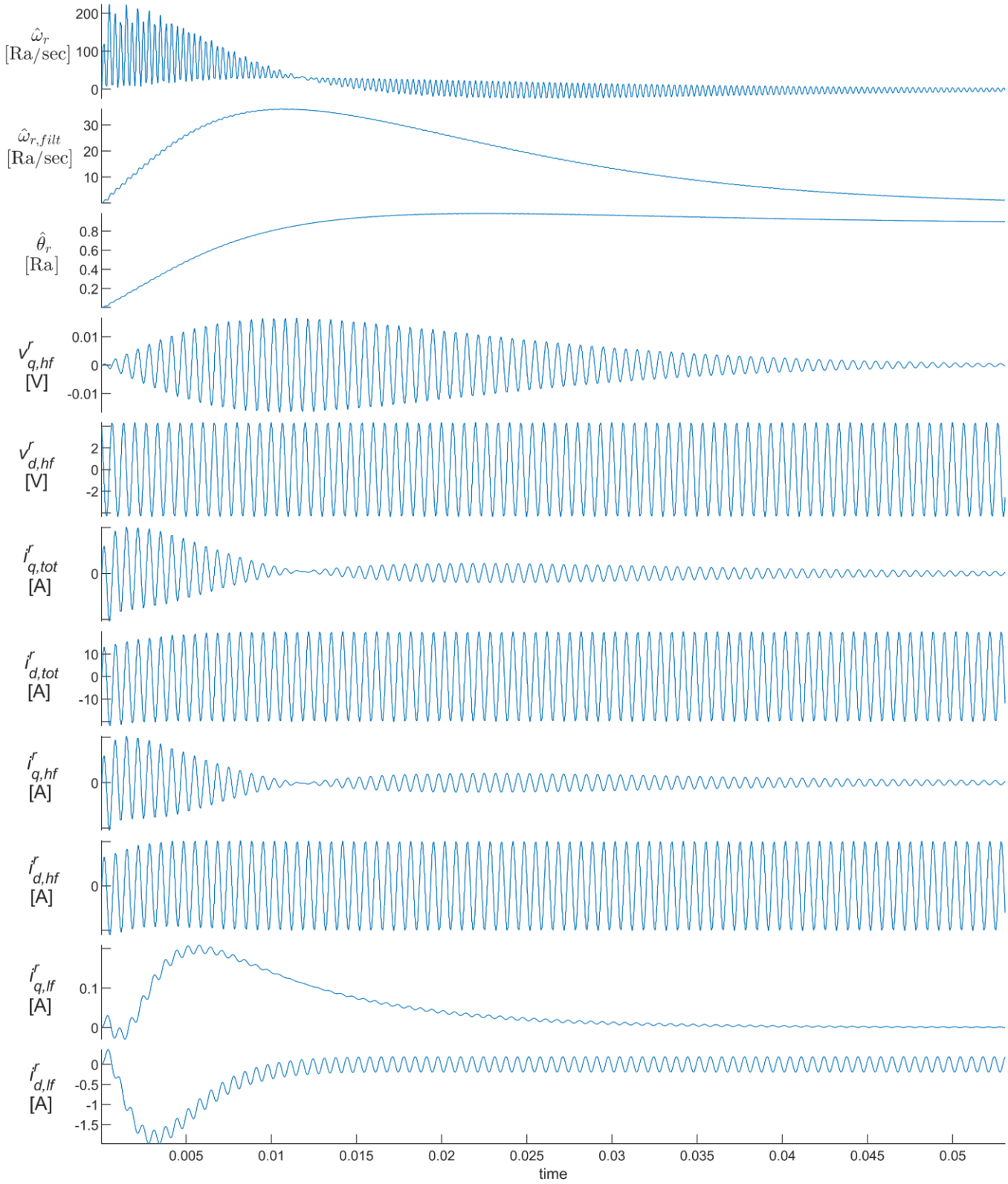

Fig. 79: Typical high frequency injection waveforms in RFO at $\omega_r = 0$

6.11.3.2 SFO

In SFO, in addition to rotor angle and speed ($\hat{\theta}_r$, $\hat{\omega}_r$), stator angle, load angle, and stator flux magnitude ($\hat{\theta}_s$, $\hat{\delta}$, $\lambda_d^{\hat{\theta}_s}$) must also be observed for control purposes. Fig. 80 depicts the block diagram of the high-frequency injection module in SFO.

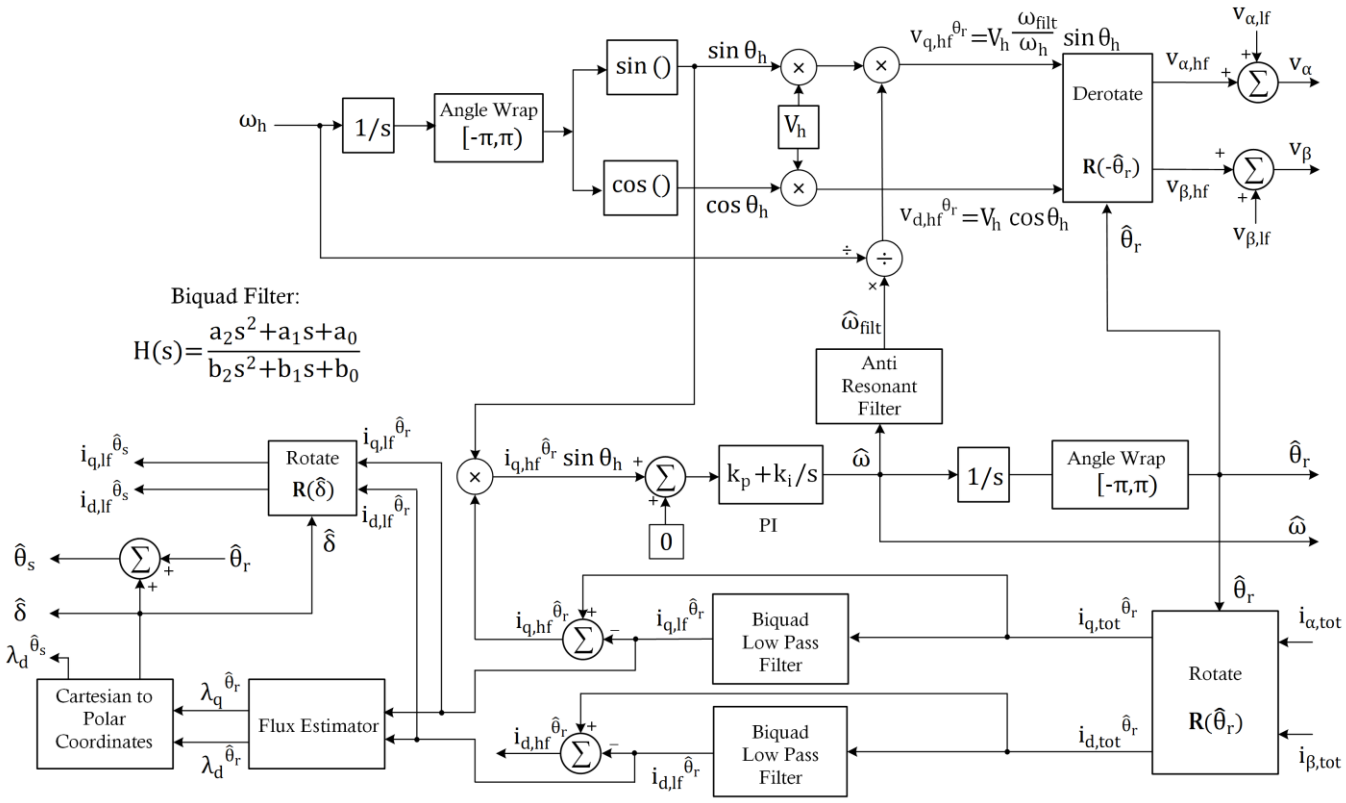

Fig. 80: High-frequency-injection block diagram in SFO

The flux estimator block in Fig. 80, estimates the flux vector based on the current vector in RFO:

$$\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r} = \begin{bmatrix} L_q i_q^{\hat{\theta}_r} \\ L_d i_d^{\hat{\theta}_r} + \lambda_m \end{bmatrix} \tag{303}$$

Then, the RFO flux vector $\boldsymbol{\lambda}_{qd}^{\hat{\theta}_r}$ in (303) is used to compute the SFO flux vector, $\boldsymbol{\lambda}_{qd}^{\hat{\theta}_s}$, load angle, $\hat{\delta}$, and stator angle, $\hat{\theta}_s$, by applying a cartesian to polar transformation as follows:

$$\boldsymbol{\lambda}_{qd}^{\hat{\theta}_s} = \begin{bmatrix} 0 \\ \sqrt{\left(\lambda_q^{\hat{\theta}_r}\right)^2 + \left(\lambda_d^{\hat{\theta}_r}\right)^2} \end{bmatrix} \tag{304}$$

$$\hat{\delta} = \tan^{-1}\left(\frac{\lambda_q^{\hat{\theta}_r}}{\lambda_d^{\hat{\theta}_r}}\right) \tag{305}$$

$$\hat{\theta}_s = \hat{\theta}_r + \hat{\delta} \tag{306}$$

In addition, all controllers operate in stator frame in SFO. Therefore, the measured currents must be projected onto the stator reference frame using the load angle as follows:

$$\boldsymbol{i}_{qd}^{\hat{\theta}_s} = \boldsymbol{e}^{-j\hat{\delta}} \boldsymbol{i}_{qd}^{\hat{\theta}_r} \tag{307}$$

Fig. 81 illustrates the typical waveforms of voltages, currents, estimated rotor angle, estimated rotor speed, estimated load angle, and estimated stator angle in SFO, as well as their transients. The motor is not spinning in. However, the algorithm presented here also works when the motor spins

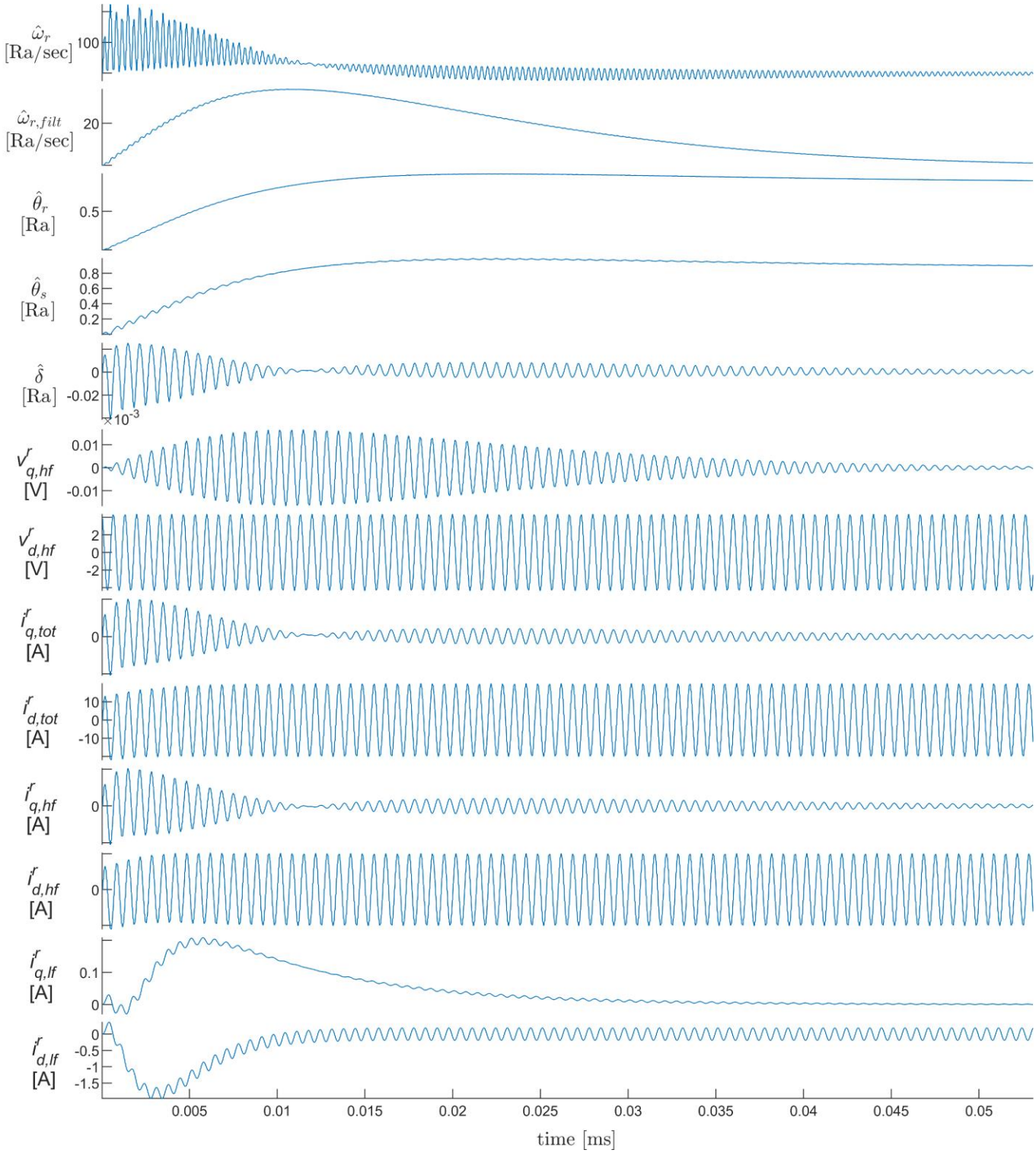

Fig. 81: Typical high frequency injection waveforms in SFO at $\omega_r = 0$

6.11.3.3 Tracking Loop

The magnitude of the injected high-frequency voltage, V_h, must be chosen carefully to ensure the magnitude of the resulting high-frequency currents doesn't go above a certain prescribed threshold. If the loop is stable, $(\tilde{\theta}_r = 0)$, the magnitude of $i_q^{\hat{\theta}_r}$ and $i_d^{\hat{\theta}_r}$ in (302) would approach

$$\begin{bmatrix} I_q \\ I_d \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{V_h}{L_q L_d \omega_h}(L_0 + L_1) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{V_h}{L_d \omega_h} \end{bmatrix} \tag{308}$$

Therefore, in order to keep the magnitude of the high-frequency current below a certain threshold, I_d, in steady state, V_h must be

$$V_h = I_d L_d \omega_h \tag{309}$$

Equation (309) can be used to calculate required voltage magnitude, V_h, based on the desired current magnitude threshold, I_d, and the injection frequency, ω_h.

The biquad low pass filters shown in Fig. 77 and Fig. 80 should provide enough attenuation at ω_h to be able to separate the high frequency and low frequency components. Here, a critically-damped second-order low pass filter the corner frequency of ω_{sep} is utilized, where ω_{sep} denotes the separation frequency which must be sufficiently lower than ω_h and sufficiently higher than the maximum frequency content of the main controller loops:

$$H_{LPF}(s) = \frac{1}{\left(s/\omega_{\text{sep}} + 1\right)^2} \tag{310}$$

The tracking loop's PI parameters (k_p and k_i) shown in Fig. 77 and Fig. 80 must also be tuned properly to ensure stability and optimal transient response. The loop gain of the tracking loop can be expressed as

$$G(s) = \frac{I_q}{2} \cdot \frac{1}{s} \cdot \left(k_{\text{p}} + \frac{k_i}{s}\right) \cdot 2 \tag{311}$$

where I_q is the maximum peak value of $i_q^{\hat{\theta}_r}$ in (302):

$$I_q = \frac{V_h L_1}{L_q L_d \omega_h} = \frac{V_h}{L_d \omega_h} \cdot \frac{\xi - 1}{2\xi} = I_d \cdot \frac{\xi - 1}{2\xi} \tag{312}$$

It must be noted that the division of I_q by 2 in (311) is due to amplitude halving effect created the demodulation stage (Fig. 77 and Fig. 80), and multiplication by 2 is due to $\sin(2\tilde{\theta}_r)$ term in (302).

Let us assume the desired bandwidth of the tracking loop is given as ω_0. By choosing the following k_p and k_i

$$\begin{cases} k_p = \frac{\omega_0}{I_q} \\ k_i = \frac{\omega_0^2}{4 I_q} \end{cases} \tag{313}$$

equation (311) will become

$$G(s) = \frac{\omega_0}{s} \cdot \left(1 + \frac{\omega_0/4}{s}\right) \tag{314}$$

It can be shown that (314) is always stable with a phase margin of $75°$, which results in a very good transient response. A normalized bode plot of (314) is shown in Fig. 82, where it can be seen that the phase response at the cross-over frequency is $\sim -105°$, which implies a phase margin of $75°$.

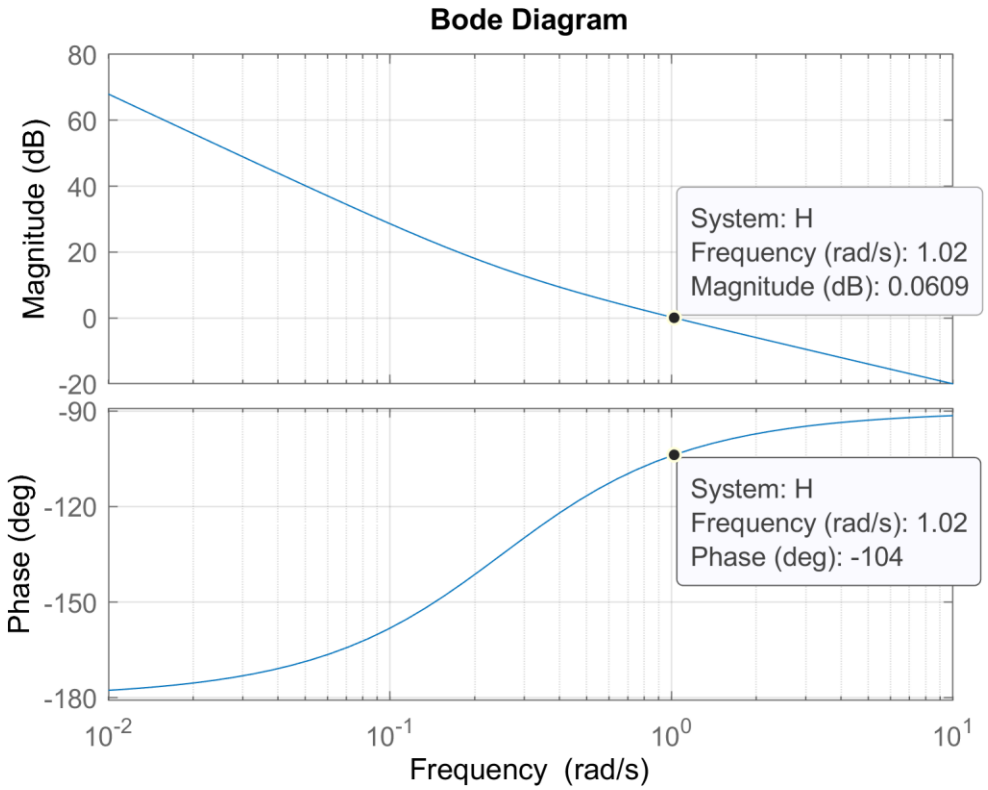

Fig. 82: Loop-gain bode plot and phase margin for the tracking loop (normalized bandwidth $\omega_0 = 1$)

6.11.4 High Frequency Injection: Square Wave

In square-wave method, the high-frequency voltages that are injected to the motor are of square-wave shape. This is in contrast to sine-wave method described in 6.11.3.

The square-wave method is simpler and less computationally intensive compared to sine-wave method because it doesn't require low pass filters and demodulation. Elimination of demodulation also allows the square-wave method to operate at higher injection frequency. However, compared to the sine-wave method, it usually creates higher audible noise and torque ripple, and is less efficient.

The user should pick the proper high-frequency-injection method based on the application requirements and motor characteristics.

6.11.4.1 RFO

The voltage equations of the PMSM in the synchronous reference frame, θ, can be written as

$$\begin{cases} v_q = ri_q + L_q \dfrac{di_q}{dt} + \omega(L_d i_d + \lambda_m) \\ v_d = ri_d + L_d \dfrac{di_d}{dt} - \omega(L_q i_q) \end{cases} \quad (315)$$

If the injected voltage is a symmetrical square wave (+/-) at a high enough frequency, the operating point of the currents (i_q, i_d) and ω in (315) can be assumed to stay intact. Therefore, only the derivative (ripple) terms in (315) are affected by the high-frequency voltage components:

$$\frac{d}{dt}\begin{bmatrix} i_q \\ i_d \end{bmatrix} \approx \begin{bmatrix} 1/L_q & 0 \\ 0 & 1/L_d \end{bmatrix}\begin{bmatrix} v_{qh} \\ v_{dh} \end{bmatrix} \quad (316)$$

These ripple terms can be transformed into stationary reference frame as

$$\frac{d}{dt}\begin{bmatrix} i_q \\ i_d \end{bmatrix} = \frac{d}{dt}\left(\boldsymbol{R}(\theta)\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}\right) \approx \boldsymbol{R}(\theta)\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{317}$$

assuming that θ's operating point stays intact. Substituting (317) to (316) would yield

$$\boldsymbol{R}(\theta)\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1/L_q & 0 \\ 0 & 1/L_d \end{bmatrix}\boldsymbol{R}(\theta-\hat{\theta})\begin{bmatrix} \hat{v_{qh}} \\ \hat{v_{dh}} \end{bmatrix} \tag{318}$$

where $\hat{v_{qh}}$ and $\hat{v_{dh}}$ denote the RRF voltages in the "estimated" rotor frame $\hat{\theta}$ (not the actual rotor frame θ). Therefore,

$$\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \boldsymbol{R}(-\theta)\begin{bmatrix} 1/L_q & 0 \\ 0 & 1/L_d \end{bmatrix}\boldsymbol{R}(\tilde{\theta})\begin{bmatrix} \hat{v_{qh}} \\ \hat{v_{dh}} \end{bmatrix} \tag{319}$$

If the injected voltages in the estimated frame $\hat{\theta}$ are always aligned with the d-axis, i.e.

$$\begin{bmatrix} \hat{v_{qh}} \\ \hat{v_{dh}} \end{bmatrix} = \begin{bmatrix} 0 \\ \pm V_h \end{bmatrix} \tag{320}$$

we can express the inputs of the PLL (p_α and p_β) based on (319) and (320) as

$$\begin{bmatrix} p_\alpha \\ p_\beta \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = (\pm V_h)\begin{bmatrix} \dfrac{-\cos(\theta)\sin(\tilde{\theta})}{L_q} + \dfrac{\sin(\theta)\cos(\tilde{\theta})}{L_d} \\ \dfrac{\sin(\theta)\sin(\tilde{\theta})}{L_q} + \dfrac{\cos(\theta)\cos(\tilde{\theta})}{L_d} \end{bmatrix} \tag{321}$$

The first observation that can be made from (321) is that for a SPM motor with no saliency,

$$\{L_q = L_d\} \Rightarrow \begin{bmatrix} p_\alpha \\ p_\beta \end{bmatrix} = \frac{\pm V_h}{L_d}\begin{bmatrix} \sin(\hat{\theta}) \\ \cos(\hat{\theta}) \end{bmatrix} \tag{322}$$

In other words, the PLL error will always be zero in SPM, which confirms that high-frequency-injection method can only be used for IPMs with significant saliency ($L_q \neq L_d$).

The second observation that can be made from (321) is that when the angle error is zero ($\tilde{\theta} = 0$), the PLL inputs only depend on the actual angel θ and L_d:

$$\{\tilde{\theta} = 0\} \Rightarrow \begin{bmatrix} p_\alpha \\ p_\beta \end{bmatrix} = \frac{\pm V_h}{L_d}\begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} \tag{323}$$

However, at $\tilde{\theta} = \pi$, the PLL inputs polarity is reversed, which means the PLL's estimated position $\hat{\theta}$ will be stable at $\tilde{\theta} = \pi$:

$$\{\tilde{\theta} = \pi\} \Rightarrow \begin{bmatrix} p_\alpha \\ p_\beta \end{bmatrix} = \frac{\mp V_h}{L_d}\begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} \tag{324}$$

This confirms the fact that there is always ±180° ambiguity when employing any high-frequency-injection method that must be resolved by another method (e.g. six pulse injection).

At near zero angle error ($\tilde{\theta} = 0$), the current ripple response to the injected square-wave voltages can be written based on (321) as

$$\begin{cases} \Delta i_\alpha^+ \approx \dfrac{V_h \Delta T}{L_d} \sin(\theta) \\ \Delta i_\beta^+ \approx \dfrac{V_h \Delta T}{L_d} \cos(\theta) \end{cases}, \quad \begin{cases} \Delta i_\alpha^- \approx \dfrac{-V_h \Delta T}{L_d} \sin(\theta) \\ \Delta i_\beta^- \approx \dfrac{-V_h \Delta T}{L_d} \cos(\theta) \end{cases} \tag{325}$$

Usually, the differentiation is done for both positive and negative halves of the square-wave and the results are computed as

$$\begin{cases} \Delta^2 i_\alpha = \Delta i_\alpha^+ - \Delta i_\alpha^- \approx \dfrac{2V_h \Delta T}{L_d} \sin(\theta) \\ \Delta^2 i_\beta = \Delta i_\beta^+ - \Delta i_\beta^- \approx \dfrac{2V_h \Delta T}{L_d} \cos(\theta) \end{cases} \tag{326}$$

to increase the signal-to-noise ratio (since differentiation always amplifies the noise). These ripple estimates in (326) are then fed to the PLL for rotor position and speed estimation as seen in the figure below.

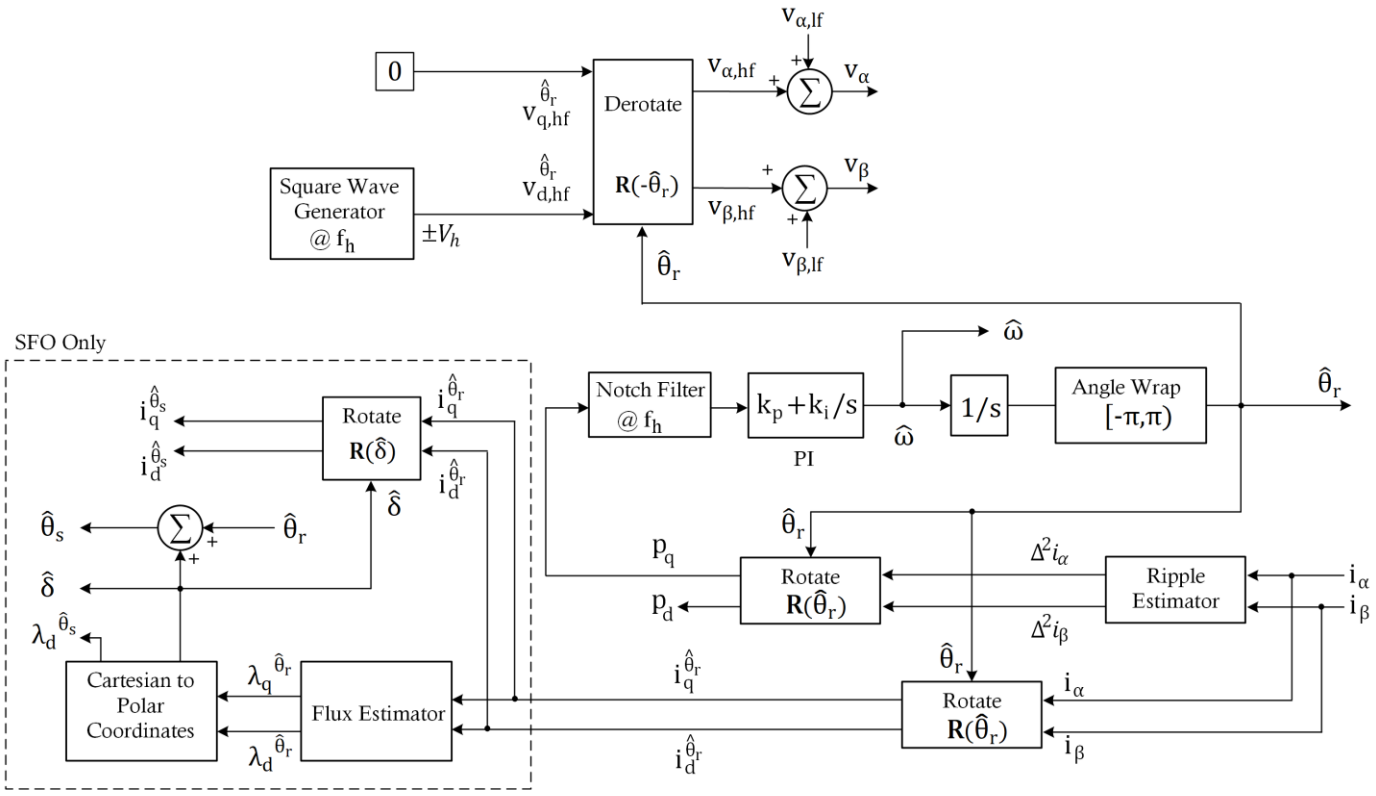

Fig. 83: Square-wave high-frequency-injection block diagram

From the block diagram above and (321), p_q and p_d can be written as

$$p_q = p_\alpha \cos(\hat{\theta}) - p_\beta \sin(\hat{\theta}) = (\pm V_h)\left(\frac{1}{L_d} - \frac{1}{L_q}\right) \underbrace{\sin(\tilde{\theta})\cos(\tilde{\theta})}_{\frac{1}{2}\sin(2\tilde{\theta})} \tag{327}$$

and

$$p_d = p_\alpha \sin(\hat{\theta}) + p_\beta \cos(\hat{\theta}) = (\pm V_h)\left(\frac{\sin^2(\tilde{\theta})}{L_q} + \frac{\cos^2(\tilde{\theta})}{L_d}\right) \tag{328}$$

The significance of (327) is that it shows p_q is directly proportional to the angle error $\sin(2\tilde{\theta})$ and hence, it can be directly fed to the PLL for position and speed estimation. In addition, (328) proves that p_d can be used as a measure of the magnitude of current ripple vector if needed.

The magnitude of the error signal p_q must be accounted for when selecting the PI parameters (k_p and k_i) for the PLL. This magnitude can be calculated as

$$M = (2V_h \Delta T)\left(\frac{1}{L_d} - \frac{1}{L_q}\right) \tag{329}$$

Therefore, the PLL PI parameters (k_p and k_i) can calculated based on the desired bandwidth ω_0 as

$$\begin{cases} k_p = \omega_0 M \\ k_i = \dfrac{\omega_0^2}{2} M \end{cases} \Rightarrow \left\{\xi = \frac{1}{\sqrt{2}} \approx 0.707, \mathrm{PM} \approx 65°\right\} \tag{330}$$

The optimal damping factor of $\xi = 0.707$ which results in a phase margin of $65°$ is utilized here.

Proper selection of V_h as a parameter is very important since a high V_h can cause excessive current ripple and noise whereas a low V_h may result in weak signals that would force the PLL out of phase lock. Based on the selected injection frequency f_h and maximum desired peak-to-peak ripple Δi_{pp}, V_h can be calculated as follows.

$$\begin{cases} f_h = \dfrac{1}{2\Delta T} \\ \Delta i_{\mathrm{pp}} = \dfrac{V_h \Delta T}{L_d} \end{cases} \Rightarrow \left\{V_h = 2\Delta i_{\mathrm{pp}} \cdot L_d \cdot f_h\right\} \tag{331}$$

6.11.4.2 SFO

Similar to sine-wave high-frequency injection, stator angle, load angle, and stator flux magnitude ($\hat{\theta}_s$, $\hat{\delta}$, $\lambda_d^{\hat{\theta}_s}$) must also be observed for control purposes in SFO. The dotted area in the block diagram of the square-wave high-frequency-injection in section 6.11.4.1 depicts the required SFO-specific computations.

6.11.5 Open-Loop V/F

As discussed in section 6.9, Volt-Hz control is a widely used simple method for driving PMSMs in open loop. This method can also be used as part of the system in closed loop sensorless control. During the startup when the back EMF information is not available for rotor position estimation, the motor can run in open loop Volt-Hz mode. Once the motor speed goes up and back EMF can be estimated by the observer, the closed loop kicks in and the motor exits Volt-Hz control. The theory and implementation in Volt-Hz is the same as what was explained in section 6.9. The user needs to follow the procedure described in section 6.9 to extract the V_{min} and K ratio. These parameters are then put in the GUI or the source files to enable the closed loop control with the Volt-Hz startup.

6.11.5.1 Experimental Results

To demonstrate how Volt-Hz startup method works, a 1.4kW 8-poles 18V PMSM motor is tested. The control mode is set to RFO speed control with Volt/Hz startup. As seen in Fig. 84, the system starts in open loop V/F mode. It can be seen that the current is increasing when the system is in open loop and suddenly decreases when it enters closed loop. This is because in closed loop MTPA is achieved i.e. the same torque is delivered by much less current. The transitions from initial rotor position to open loop and open loop to closed loop are smooth and seamless.

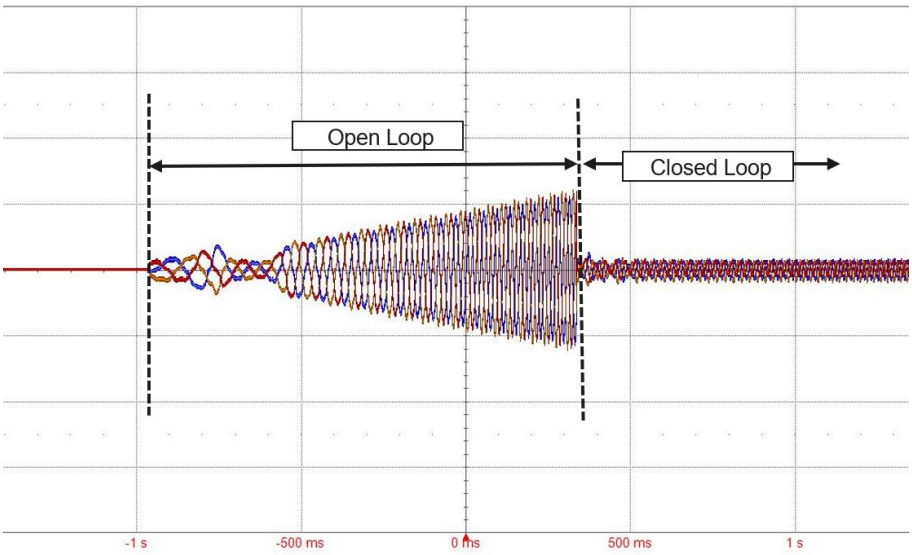

Fig. 84: FOC sensorless speed control with V/F startup.

6.11.6 Open-Loop I/F

As discussed in section 6.10, I/F control is a well-known method for driving PMSMs in open loop. This method can also be used as a part of the system in closed loop sensorless control. During the startup when the back EMF information is not available for rotor position estimation, the motor can run in open loop I/F mode. Once the motor speed goes up and back EMF can be estimated by the observer, the closed loop kicks in and the motor exits I/F control. The theory and implementation in I/F control is the same as what was explained in section 6.10. The user needs to follow the guidelines described in section 6.10 to set the parameter i_q^{r*}.

6.11.7 Dyno Catch Spin

Dyno catch spin mode can be employed to realize a back-to-back motor-generator system where one motor is the Device Under Test (DUT) and the other motor is the Dyno.

The DUT motor can be in speed control mode (RFO or SFO), current control mode (RFO), or torque control mode (SFO). The same applies to Dyno motor. However, both DUT and Dyno cannot be in the same control mode, i.e. if DUT is in speed control mode, Dyno must be in current or torque control mode and vice versa. This is due to the fact that both motors cannot compete for controlling the same physical entity.

Based on the direction of spinning and polarity of current or torque command, DUT or Dyno can be either motor or generator, i.e. if the DUT is providing positive power to mechanical shaft (motor mode), then Dyno must take that power from the shaft by providing negative power (generator mode) and vice versa.

The power would circulate in the system and the power supply only provides the power loss in the entire system. This is beneficial because it allows the user to test the motor and controller at very high power while the power supplied by the dc supply is low (only losses). In addition, there is no need to dissipate the output mechanical power as heat. Therefore, there is no need for brakes, ventilation system, noisy fans, etc.

In normal control modes, the system goes to brake and bootstrap state after the initialization (see section 6.16). This will not work for the Dyno controller because if DUT is spinning, sudden shorting of the outputs of the Dyno motor in “brake and bootstrap” state will immediately put a large mechanical load on

the coupling shaft and possibly freeze the shaft. This consequently causes instability of DUT and the whole system. In addition, the Dyno controller's observer should measure the back-emf voltages to lock before starting the controllers. Because of these reasons, in dyno control mode, the state machine transitions to "dyno lock" state right after initialization as opposed to "brake and bootstrap" state (see section 6.16). In the "dyno lock" state, the observer runs with the actual measured back-emf voltages as opposed to the commanded phase voltages. In addition, the Dyno motor phases are all in high-z state, not only to be able to measure the back-emf voltages, but also not short the Dyno motor and cause instability in the system. After Dyno's observer is locked, the state machine will transition to the normal control mode and both DUT and Dyno will continue to run while one of them is controlling the speed and the other one is controlling the current (RFO) or torque (SFO).

6.12 Motor I²T Protection

To protect the motor 'windings' from overload, I²T protection block has been implemented. The I²T cannot be considered as a replacement for OCP fault detection. I²T will not trigger any faults. Instead, it alters the allowed current command in the control loops. To understand the concept behind I²T protection, consider a simple thermal model of the motor shown in Fig. 85.

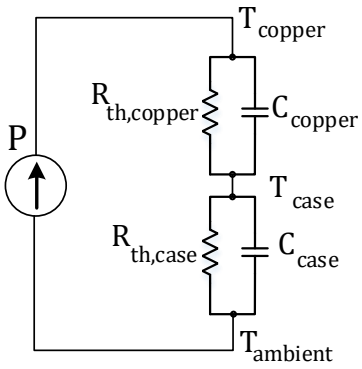

Fig. 85: Simple thermal model of a Motor

The power source in the thermal equivalent circuit is

$$P = 3\frac{Ri_s^2}{2} \quad (332)$$

where i_s is the peak phase current and R is the phase to neutral resistance. The temperature rise of the motor windings can then be calculated as:

$$\triangle T = T_{copper} - T_{case} \quad (333)$$

$$\triangle T = PR_{th,copper}(1 - e^{-t/\tau}) \quad (334)$$

$$\tau = R_{th,copper}C_{copper} \quad (335)$$

The continuous current rating of the motor, I_{cont}, is the current that motor can handle indefinitely, i.e. applying I_{cont} will result in a temperature rise ΔT that is below the maximum temperature rise, ΔT_{max}, motor windings can take without damaging the insulations. Obviously, the absolute value of the winding temperature, T_{copper}, is the limiting factor here and the temperature rise, ΔT, is usually determined by assuming a case temperature of 25 ºC.

On the other hand, the peak current rating of the motor, I_{peak}, is the current that motor can handle only for a short period depending on the time constant, τ, defined in (335).

Based on these definitions of I_{cont} and I_{peak}, the I²T protection block seen below can ensure that $\Delta T \leq \Delta T_{max}$ without needing the exact thermal model of the motor ($R_{th,copper}$ and C_{copper})

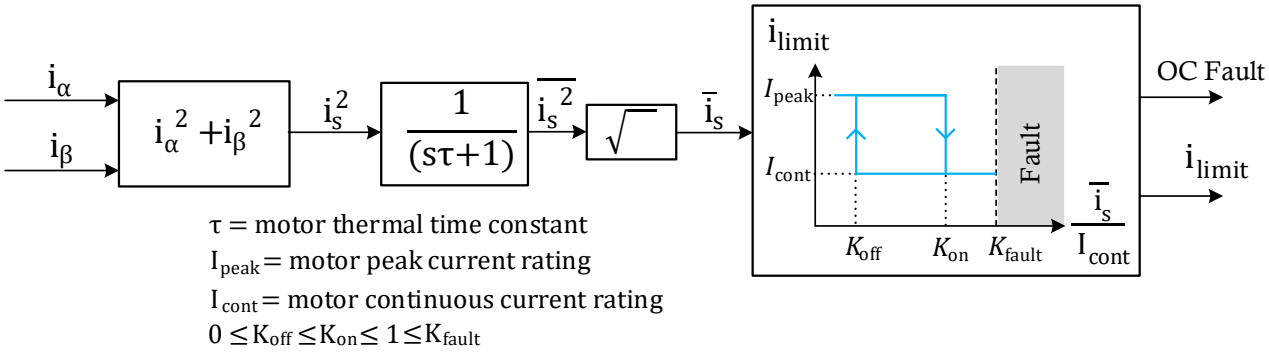

Fig. 86: I²T block block diagram

The I²T protection has been implemented with a hysteresis band in Fig. 86. If $\overline{i_s}$ is larger than $K_{on}I_{cont}$ then the allowed current limit in the control loop is reduced from I_{peak} to I_{cont}. It will only increase back to I_{peak} if $\overline{i_s}$ becomes less than $K_{off}I_{cont}$. Note that I²T protection doesn't trigger a fault. Only the current limit is going to change. However, if $\overline{i_s}$ is larger than $K_{fault}I_{cont}$ SW overcurrent fault will trigger which is independent of I²T protection.

As seen in the figure below, I_{cont} can be forever applied to the motor without exceeding the maximum temperature rise, ΔT_{max}. However, when I_{peak} is applied (e.g. $I_{peak} = 2I_{cont}$ here), the current cap must be reduced to I_{cont} at t_0 to ensure the temperature rise is below the safe threshold, ΔT_{max}.

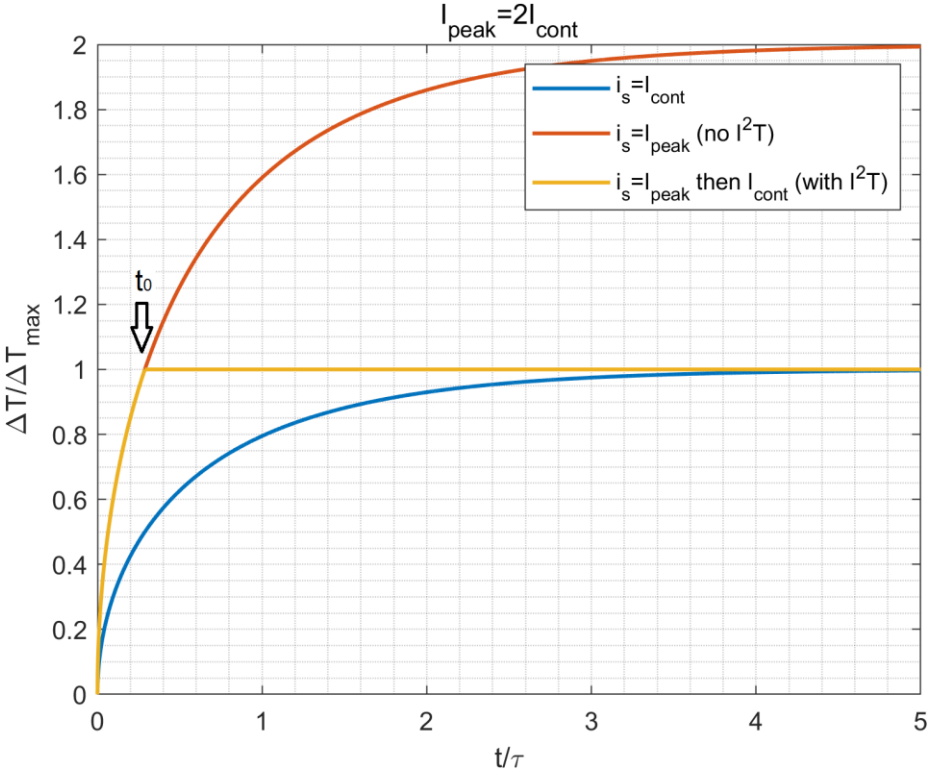

Fig. 87: Normalized temperature rise vs. normalized time when applying continuous and peak current ratings of the motor

6.12.1 Experimental Results

In this section, using experimental results, it is shown how the motor performs when I²T protection kicks in.

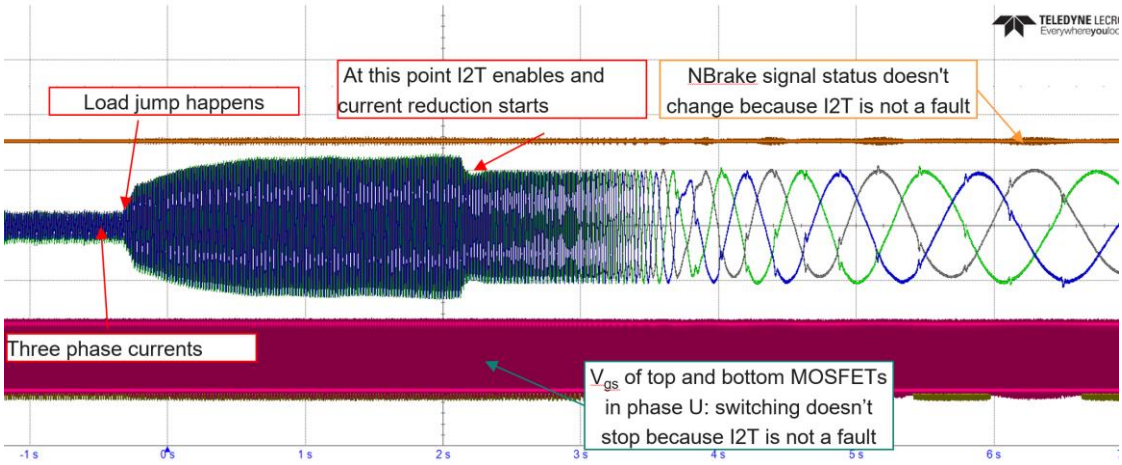

Fig. 88: I^2T protection block operation: I_{limit} is reduced from I_{peak} to I_{cont}

Initially system runs with light load. Then quickly heavy load is applied which increases the peak of phase currents. It takes around two seconds for the I^2T to get enabled which is consistent with applied **params.motor.i2t.therm_tau.** When I^2T is enabled I_{limit} is enforced to reduce from I_{peak} to I_{cont} as it can be seen from Fig. 88.

As explained in section 6.12 if $\overline{i_s}$ is larger than $K_{fault}I_{cont}$, SW overcurrent fault will trigger. This is independent of I^2T protection. This has been shown in Fig. 89. Note that the SW OC fault is independent of HW OC fault. The HW OC fault will act much faster, and it is aimed to protect against large instantaneous changes in phase current. However, the HW OC threshold is usually much higher.

Fig. 89: SW OC fault detection and MCU reaction

6.13 Fault Detections

Beside the I^2T protection and OC fault detection as discussed in the previous section, there needs to be other protections and fault detections implemented to guarantee the proper operation in the motor drive system. These faults include over temperature, over speed, under voltage and over voltage. Over temperature and over speed faults can be easily implemented by continuously monitoring and comparing the real-time values to their corresponding fault thresholds. However, for over voltage and under voltage faults, a debounce filter can be implemented according to Fig. 90.

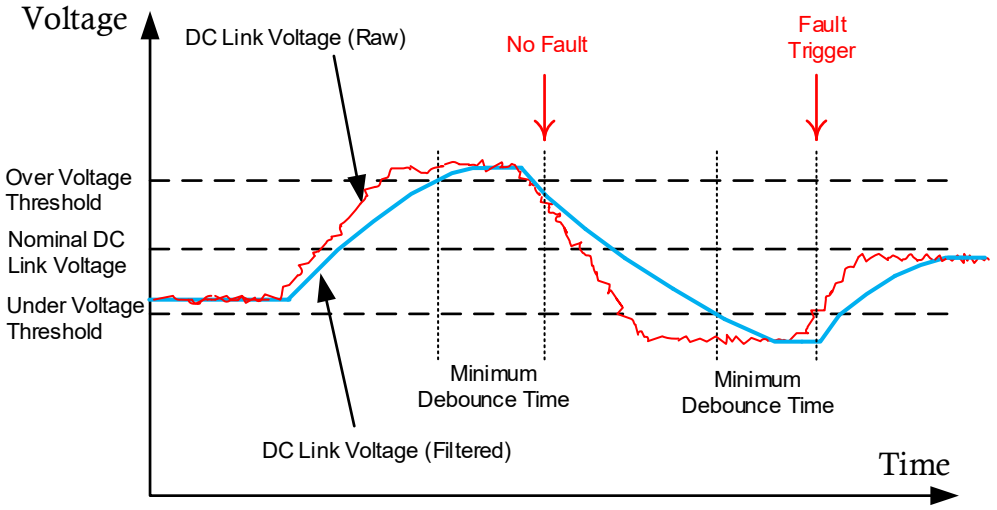

Fig. 90: Over voltage and under voltage fault detection.

As shown in this figure, a debounce filter can be implemented to avoid spurious fault triggering. A threshold for over voltage and another one for under voltage are defined. If the voltage goes above the over-voltage threshold or below the under-voltage threshold for the duration of the debounce filter time, then a fault will be detected.

After any fault occurs, an action must be taken to react to the fault and stop the motor drive operation. The fault reaction in the inverter could be either open circuit (high-z) which leaves the motor windings open or short circuit which shorts the motor windings together. Fig. 91 illustrates the possible short fault reactions in an inverter. As shown, the motor can be shorted through high side switches, low side switches or it could be alternating between high side and the low side to distribute the generated heat in the inverter.

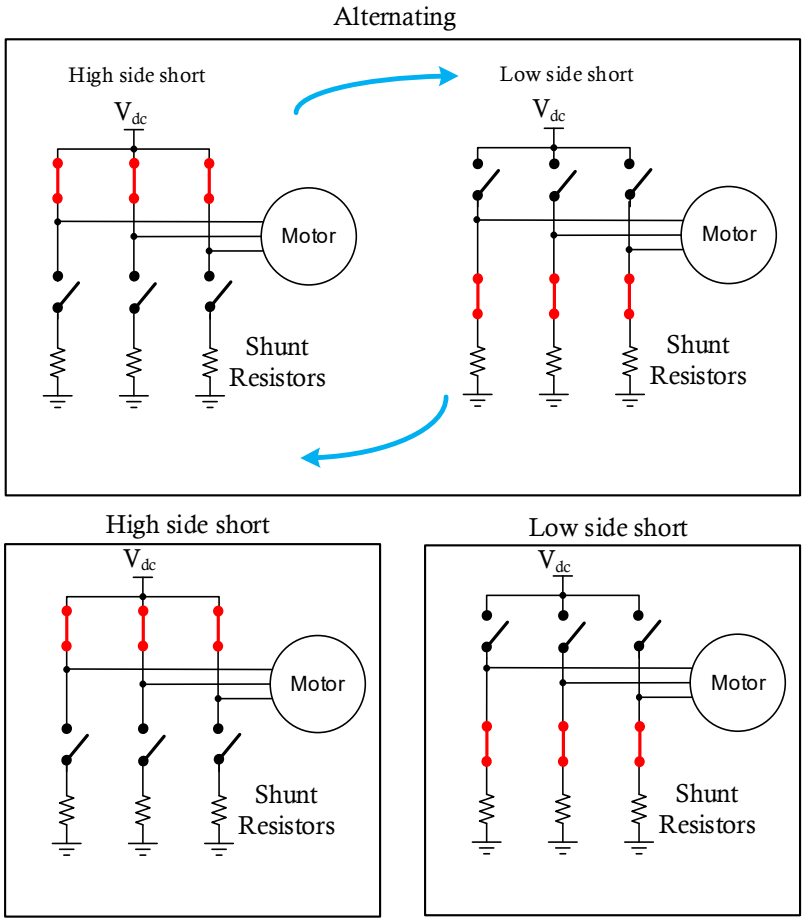

Fig. 91: Inverter fault reaction

Note that each fault will require a specific fault reaction given the requirement of the application. For instance, a fault reaction to over temperature could be open circuit as a short circuit fault reaction in such a scenario could cause even higher temperature in the inverter. Therefore, the fault reaction for any fault scenario must be selected appropriately.

6.13.1 Experimental Results

Here in this section HW OC fault detection and subsequent MCU reaction is discussed. It is shown in Fig. 92 that when the phase-V current goes below a negative threshold (here -125 A), fault is detected by HW and nFault signal is sent to MCU. As a reaction to this fault, all MOSFETs will be forced to go into high Z state right away.

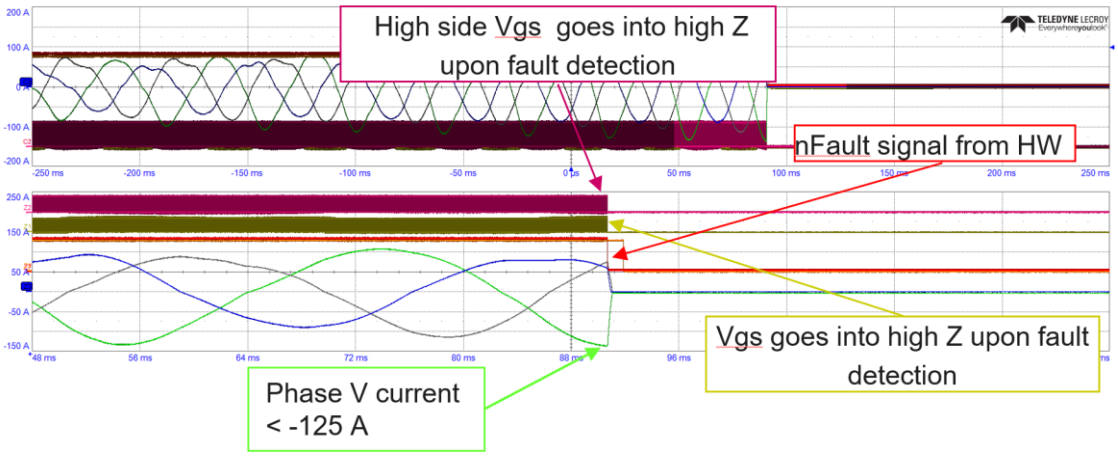

Fig. 92: HW OC Fault detection and MOSFETs going to high Z subsequently.

6.14 Analog Sensor Calibration and Filtering

The sampled current and voltage signals that are passed to the CPU from ADCs need to be calibrated and filtered. This is to compensate for the measurement error due to the hardware component tolerances and noise disturbances. Particularly for the three phase current samples offset nulling needs to be considered. This is because as the current sense opamp ages, its offset keeps changing which can lead to a significant measurement error if not compensated. Fig. 93 shows the calibration and filtering block diagram. Note that as seen in Fig. 93 offset added in the calibration unit is independent of the offsets that originate from opamps and need to be nulled.

Calibration of the sampled signal is straightforward. (336) shows the simple equation of a line with slope ***a*** and intercept ***b*** where ***a*** is the gain and ***b*** is the offset as shown in the Fig. 93.

$$y = ax + b, x = \text{Raw Signal}, y = \text{Calibrated Signal} \tag{336}$$

Each hardware design is unique. That is why user need to extract *a* and *b* parameters based on the hardware. In the case of phase currents, the nulling offset that is extracted by the automatic offset nulling loop needs to be subtracted from calibrated signals. Then the result will be filtered. With voltage signals there is no nulling offset and the result of calibration is directly filtered. The filter is a common IIR low pass filter:

$$y_n = y_{n-1} + k_0(x_n - y_{n-1}) \tag{337}$$

where k_0 is the filter coefficient which sets the bandwidth and is calculated as seen below:

$$k_0 = \frac{\omega_0}{f_s} \tag{338}$$

Here, ω_0 is the low pass filter bandwidth as shown in Fig. 93 and it is set by the user. f_s is the sampling frequency (ISR0 frequency).

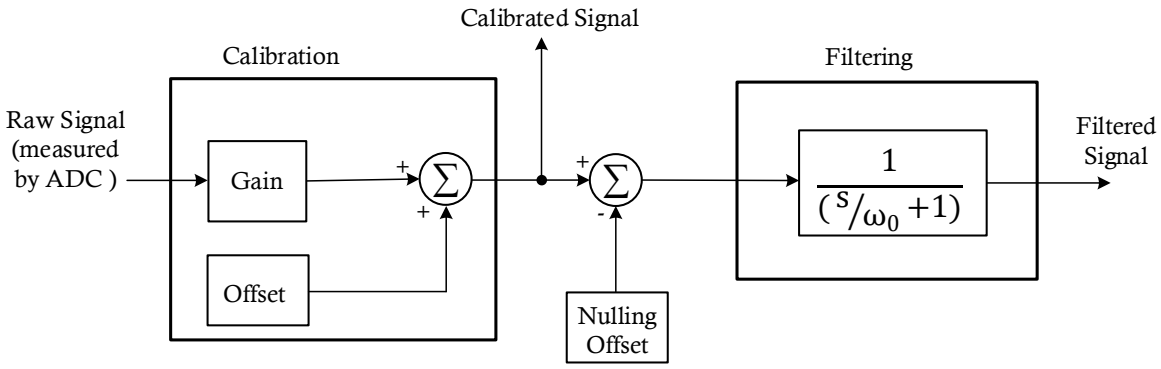

Fig. 93: Calibration and filtering block diagram

Fig. 94 shows the block diagram of the automatic offset nulling loops for current measurements. Note that this subroutine is only executing during the startup in ***Init*** state. As this control loop forces the error to zero, the nulling offset converges to the actual offset of current sense opamps.

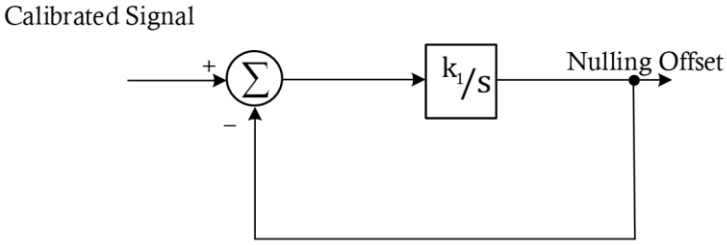

Fig. 94: Automatic current-sense offset nulling.

In this loop, k_1 is:

$$k_1 = \frac{(\tau \text{ ratio})}{(\text{offset nulling time})} \tag{339}$$

where in (339), $(\text{offset nulling time})$ is the time duration when the offset nulling loop is running during ***Init*** state. The default value for this parameter is 250 ms. $(\tau \text{ ratio})$ is the ratio that sets the number of time constants within $(\text{offset nulling time})$. The default for $(\tau \text{ ratio})$ is 5 which means the extracted offset will be at %99 of the actual value $(1 - e^{-5} = 0.9933)$ after ***Init*** state is done.

6.15 Rate Limiters

To have a controllable soft start for the system, rate limiters have been implemented and are applied to the commanded inputs including speed, torque, and current. The rates are defined in units of ***rad/(sec)²***, ***A/sec***, ***Ra/Sec*** and ***Nm/sec*** respectively for speed, current, and torque commands as shown in Fig. 95, Fig. 96 and Fig. 97. For example, with a 10 ***A/sec*** rate limit, it takes 1 second for the current to go from 0A to 10A.

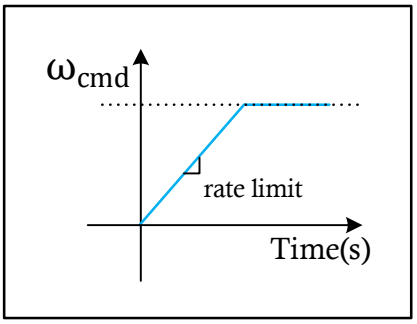

Fig. 95: Commanded speed vs time with a user selected rate limit.

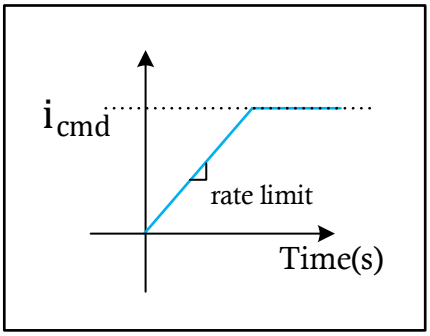

Fig. 96: Commanded current vs time with a user selected rate limit.

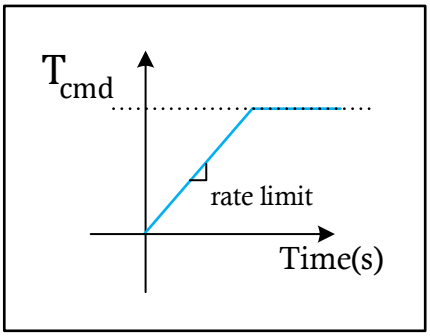

Fig. 97: Commanded Torque vs time with a user selected rate limit.

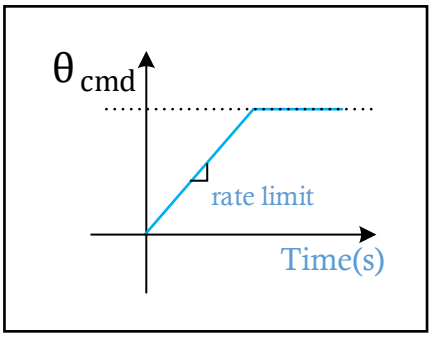

Fig. 98: Commanded position vs time with a user selected rate limit.

6.16 State Machine

The states implemented in the state machine have been listed in Table 2. Note that while most of the states are common among RFO and SFO, there are states that are unique to RFO (e.g. ***Current_CL***) or SFO (e.g. ***Torque_CL***). TBC only uses some of the states due to its simpler implementation. For example, ***Speed_OL_To_CL*** state is available in RFO and SFO, however, it is not available in TBC as seen in Table 2. In addition, Table 2 includes the description of each of these states. Note that the states associated with motor profiler including ***Prof_Finished, Prof_Rot_Lock, Prof_R, Prof_Ld*** and ***Prof_Lq*** will be discussed in section 6.18.1.

Table 2: States in the state machine.

State	Used in			Description
Init	RFO	SFO	TBC	This is the first state by default and parameter initialization and offset nulling happen here.
	✓	✓	✓	
Brake_Boot	RFO	SFO	TBC	In this state, bottom switches turn on to charge bootstrap capacitors before switching starts.
	✓	✓	✓	
Align	RFO	SFO	TBC	If the align mode is selected by the user as the startup method, the state machine waits for the rotor to align with the stator field in this state.
	✓	✓		
Six_Pulse	RFO	SFO	TBC	At startup, the state machine comes here to estimate rotor position before getting into a closed loop or high frequency state.
	✓	✓		
High_Freq	RFO	SFO	TBC	If high frequency is selected for startup, during the startup state machine comes here to estimate rotor position before getting into the closed loop.
	✓	✓		
Speed_OL_To_CL	RFO	SFO	TBC	The state machine comes here when transitioning from open loop V/F to closed loop. It stays here for a predefined time set by user for the observer to lock.
	✓	✓		
Dyno_Lock	RFO	SFO	TBC	Instead of Brake_Boot, the state machine comes here in dyno mode state so that the observer can lock.
	✓	✓		
Prof_Finished	RFO	SFO	TBC	When motor profiler is done, the state machine comes here and saves all the estimated parameters.
	✓	✓		
Prof_Rot_Lock	RFO	SFO	TBC	When the motor profiling mode is active, state machine comes here to lock the rotor and start the motor profiler.
	✓	✓		
Prof_R	RFO	SFO	TBC	Motor parameter R is calculated in this state.
	✓	✓		
Prof_Ld	RFO	SFO	TBC	Motor parameter L_d is calculated in this state.
	✓	✓		
Prof_Lq	RFO	SFO	TBC	Motor parameter L_q is calculated in this state.
	✓	✓		
Current_OL	RFO	SFO	TBC	In this state motor spins in an open loop based on the I/F method. The state machine also comes here when transitioning to a closed loop if the I/F startup is selected.
	✓			
Volt_Hz_OL	RFO	SFO	TBC	In this state motor spins in an open loop based on the V/F method. The state machine also comes here when transitioning to a closed loop if the V/F startup is selected.
	✓	✓	✓	
Speed_CL	RFO	SFO	TBC	Motor works in speed closed loop control.
	✓	✓	✓	
Fault	RFO	SFO	TBC	In all states, if a fault is detected state machine comes here.
	✓	✓	✓	
Torque_CL	RFO	SFO	TBC	The state machine stays here when the system is in torque control. This state is specific to only SFO.
		✓		
Current_CL	RFO	SFO	TBC	The state machine stays here when the system is in the current control.
	✓		✓	

Position_CL	RFO	SFO	TBC	The state machine stays here when the system is in the position control. This state is specific to only RFO.
	✓			

In general, each state has four or fewer functions associated with it. All states have an ***Entry()*** function while only some of them have ***Exit()*** function. All states except ***Fault*** have a fast-running function that runs in ***ISR0***. On the other hand, few states have a slow running function that runs in ***ISR1***. For each state where the function isn't available ***EmptyFcn()*** is used as seen in Table 3.

Table 3: Functions associated with each state.

State	Entry() Function	Exit() Function	RunISR0() Function	RunISR1() Function
Init	InitEntry	EmptyFcn	InitISR0	EmptyFcn
Brake_Boot	BrakeBootEntry	EmptyFcn	BrakeBootISR0	BrakeBootISR1
Volt_Hz_OL	VoltHzOLEntry	VoltHzOLExit	VoltHzOLISR0	VoltHzOLISR1
Speed_CL	SpeedCLEntry	EmptyFcn	SpeedCLISR0	SpeedCLISR1
Fault	FaultEntry	FaultExit	EmptyFcn	FaultISR1
Align	AlignEntry	AlignExit	AlignISR0	EmptyFcn
Six_Pulse	SixPulseEntry	SixPulseExit	SixPulseISR0	SixPulseISR1
High_Freq	HighFreqEntry	HighFreqExit	HighFreqISR0	HighFreqISR1
Speed_OL_To_CL	SpeedOLToCLEntry	SpeedOLToCLExit	SpeedOLToCLISR0	SpeedOLToCLISR1
Dyno_Lock	DynoLockEntry	DynoLockExit	DynoLockISR0	DynoLockISR1
Prof_Rot_Lock	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_R	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_Ld	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_Lq	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_Finished	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Torque_CL	TorqueCLEntry	EmptyFcn	TorqueCLISR0	TorqueCLISR1
Current_CL	CurrentCLEntry	EmptyFcn	CurrentCLISR0	CurrentCLISR1
Current_OL	CurrentOLEntry	CurrentOLExit	CurrentOLISR0	CurrentOLISR1
Position_CL	PositionCLEntry	EmptyFcn	PositionCLISR0	PositionCLISR1

In both Table 2 and Table 3 many of the states and associated functions are seen which may indicate complex state machine diagrams as a result. However, only some of these states are relevant for each control type. The state machine diagram is derived per control type. Table 4 lists all the available control types in the firmware that the user can select and the reference to their associated state machine diagram. For example, Fig. 99 shows the state machine diagram for the open loop control as shown in Table 4.

Table 4: Control types available to users and references to their associated state machine diagrams.

Control Type	Controlled Entity	Feedback Type	Startup Method	State Machine Diagram
Open-loop (all types)	Voltage	N.A.	N.A.	Fig. 99
Open-loop in RFO	Current	N.A.	N.A.	Fig. 97
FOC in RFO	Current	Sensorless	Rotor Pre-Alignment	Fig. 101
FOC in RFO	Current	Sensorless	Six Pulse Injection	Fig. 102
FOC in RFO	Current	Sensorless	High Frequency Injection	Fig. 103
FOC in RFO	Current	Sensorless	Dyno Mode	Fig. 104
FOC in RFO	Current	Encoder	Rotor Pre-Alignment	Fig. 102
FOC in RFO	Current	Hall Sensor	N.A.	Fig. 106
TBC in BC	Current	Hall Sensor	N.A.	Fig. 107
TBC in TC	Current	Hall Sensor	N.A.	Fig. 108
FOC in SFO	Torque	Sensorless	Rotor Pre-Alignment	Fig. 109
FOC in SFO	Torque	Sensorless	Six Pulse Injection	Fig. 110
FOC in SFO	Torque	Sensorless	High frequency Injection	Fig. 111
FOC in SFO	Torque	Sensorless	Dyno Mode	Fig. 112
FOC in RFO or SFO	Speed	Sensorless	Rotor Pre-Alignment	Fig. 113
FOC in RFO or SFO	Speed	Sensorless	Six Pulse Injection	Fig. 114
FOC in RFO or SFO	Speed	Sensorless	High Frequency Injection	Fig. 115
FOC in RFO or SFO	Speed	Sensorless	Open-Loop V/F	Fig. 116
FOC in RFO	Speed	Sensorless	Open-Loop I/F	Fig. 114
FOC in RFO	Speed	Encoder	Rotor Pre-Alignment	Fig. 115
FOC in RFO	Speed	Hall Sensor	N.A.	Fig. 119
FOC in RFO	Position	Encoder	Rotor Pre-Alignment	Fig. 120
TBC in BC	Speed	Hall Sensor	N.A.	Fig. 121
TBC in TC	Speed	Hall Sensor	N.A.	Fig. 122

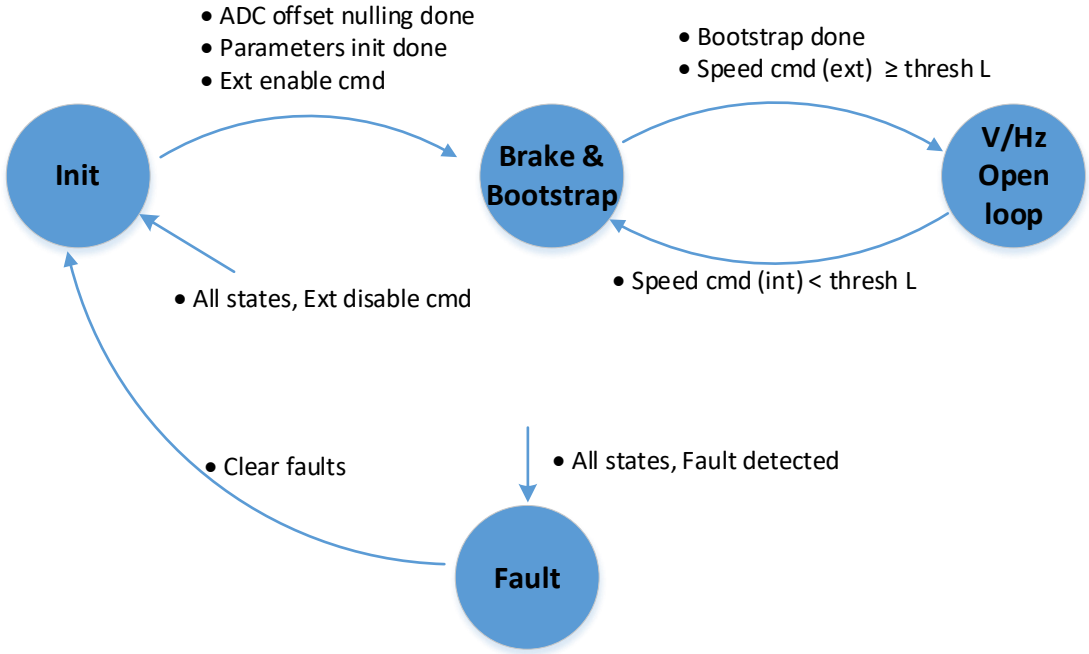

Fig. 99: V/F open loop state machine diagram.

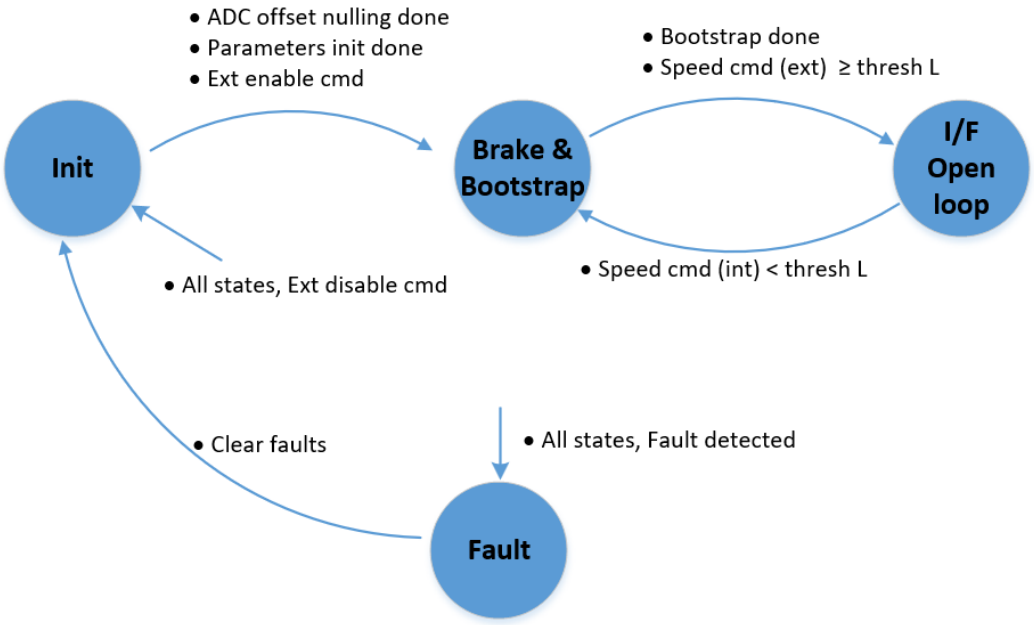

Fig. 100: I/F open loop state machine diagram.

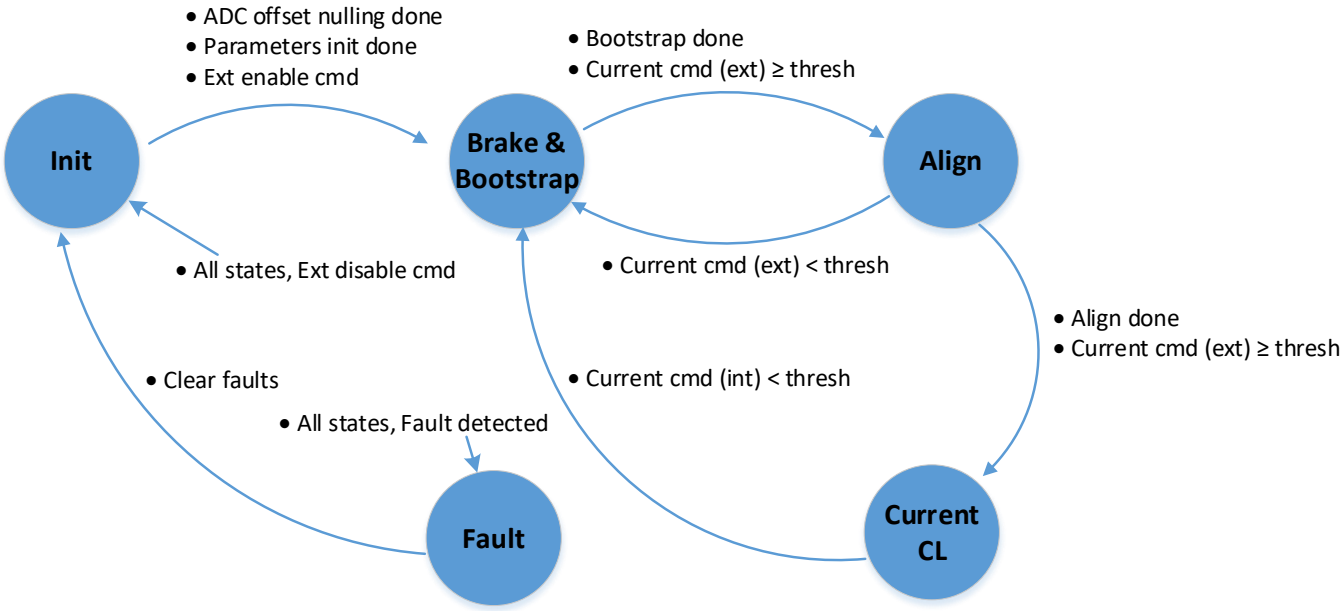

Fig. 101: RFO current mode FOC sensorless with align startup state machine diagram.

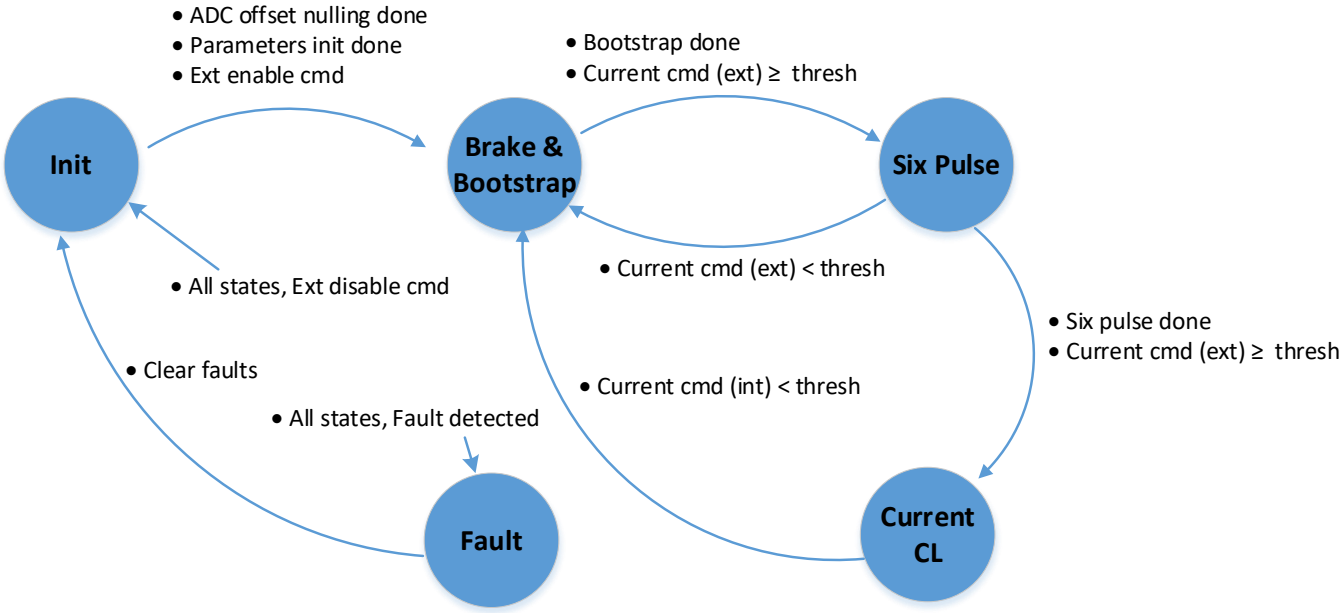

Fig. 102: RFO current mode FOC sensorless with six pulse startup state machine diagram.

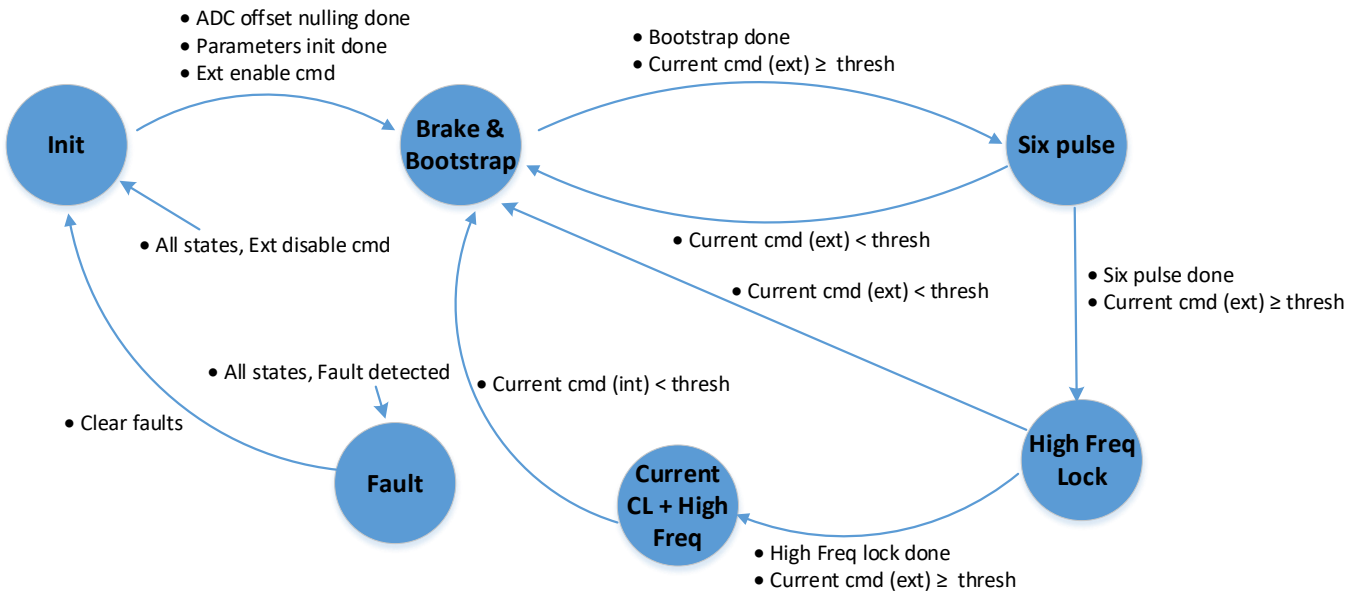

Fig. 103: RFO current mode FOC sensorless with high frequency startup state machine diagram.

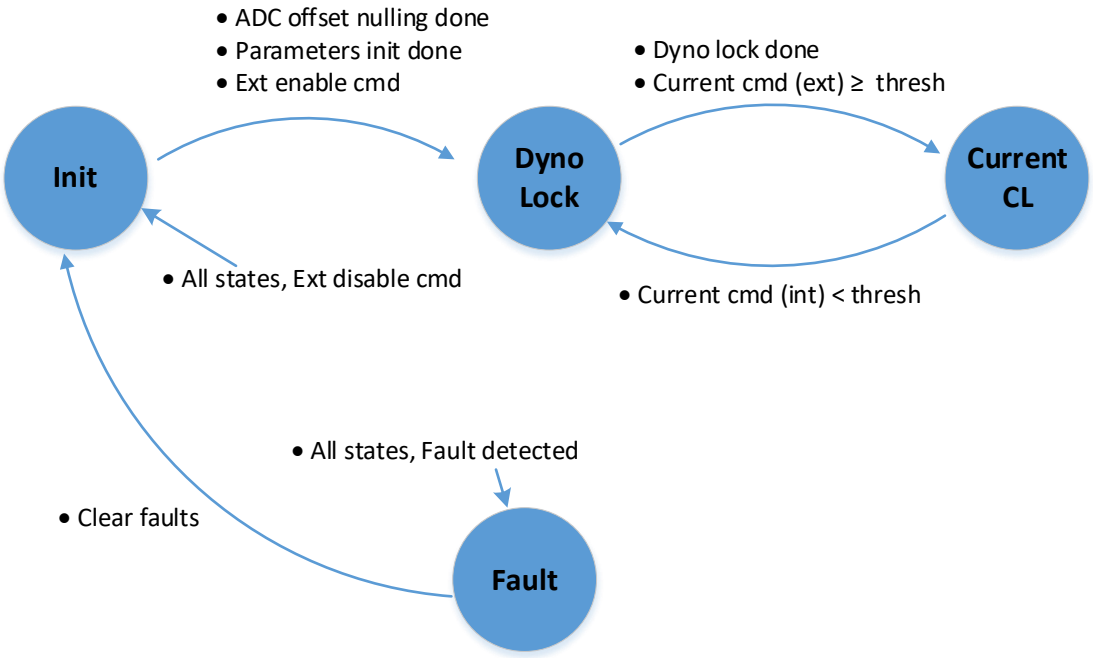

Fig. 104: RFO dyno current mode FOC sensorless state machine diagram.

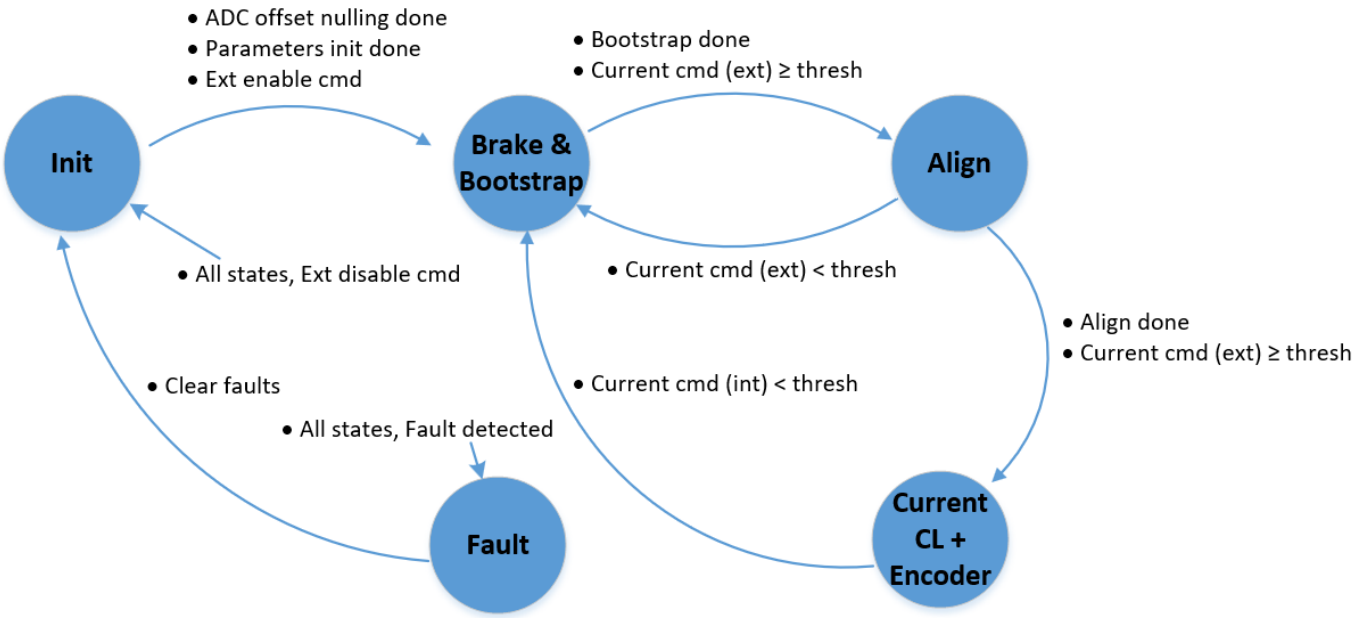

Fig. 105: RFO current mode using encoder with rotor pre-alignment startup state machine diagram.

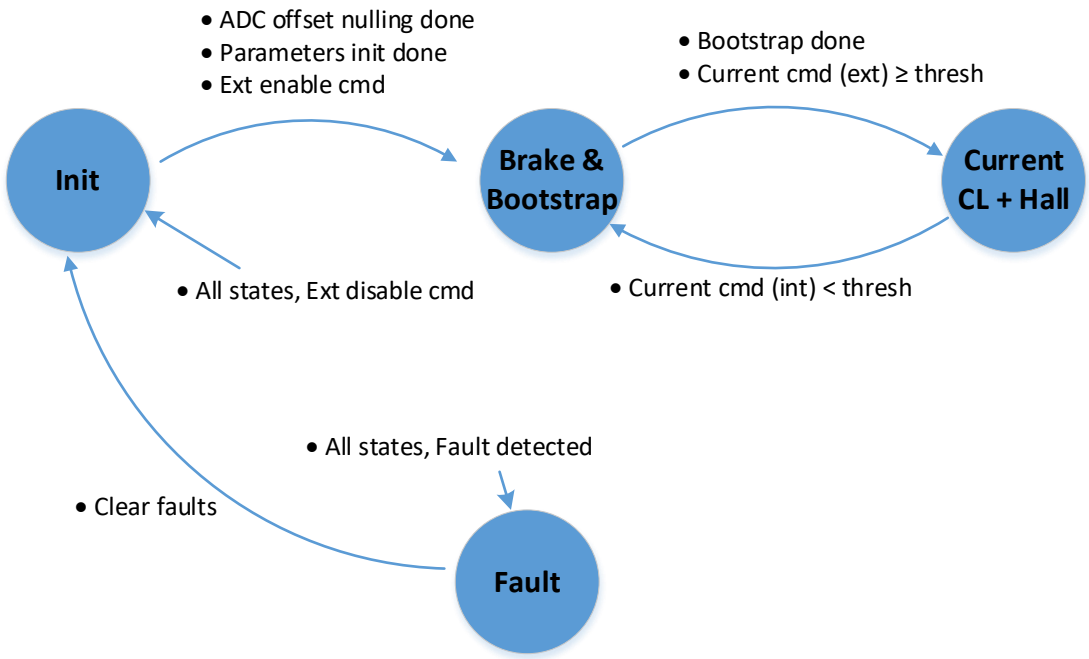

Fig. 106: RFO current mode FOC with hall sensor state machine diagram.

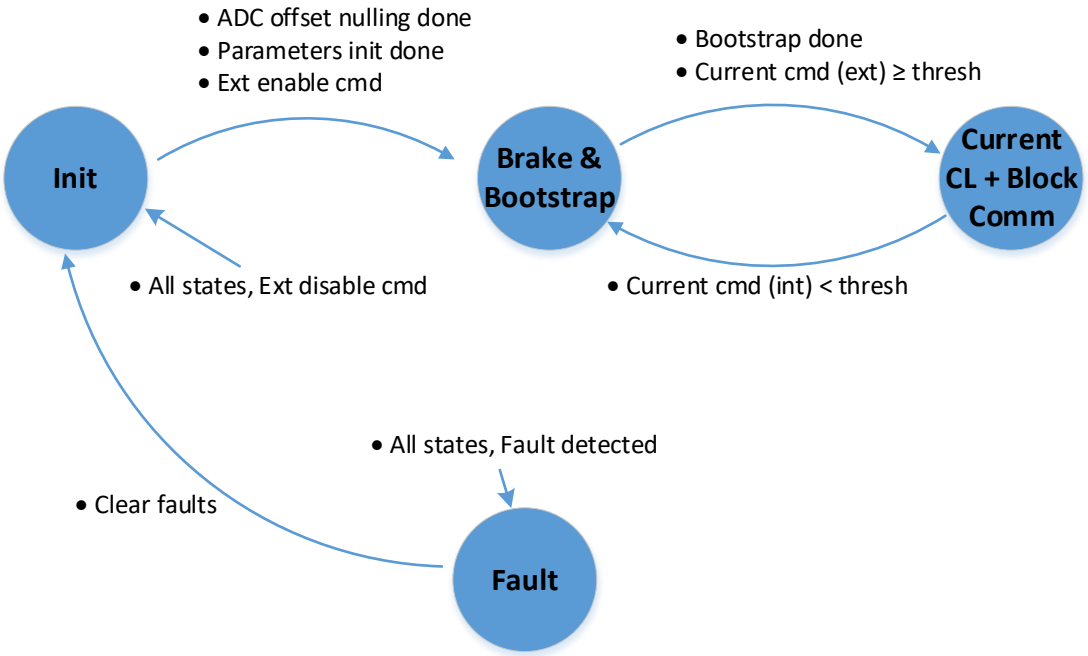

Fig. 107: TBC block commutation current control state machine diagram.

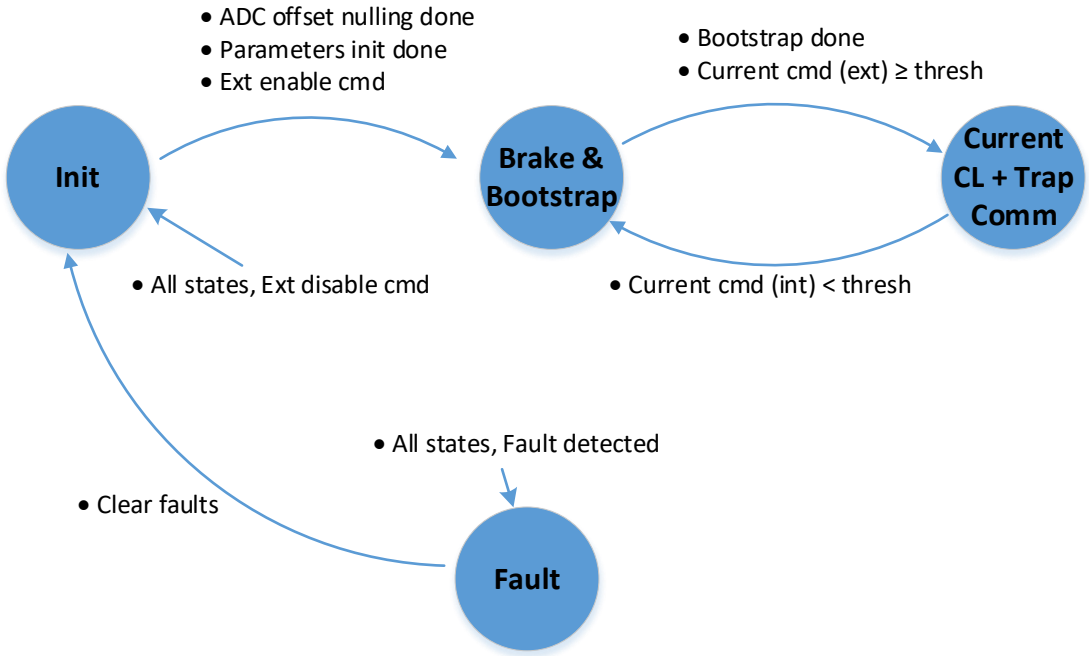

Fig. 108: TBC trapezoidal commutation current control state machine diagram.

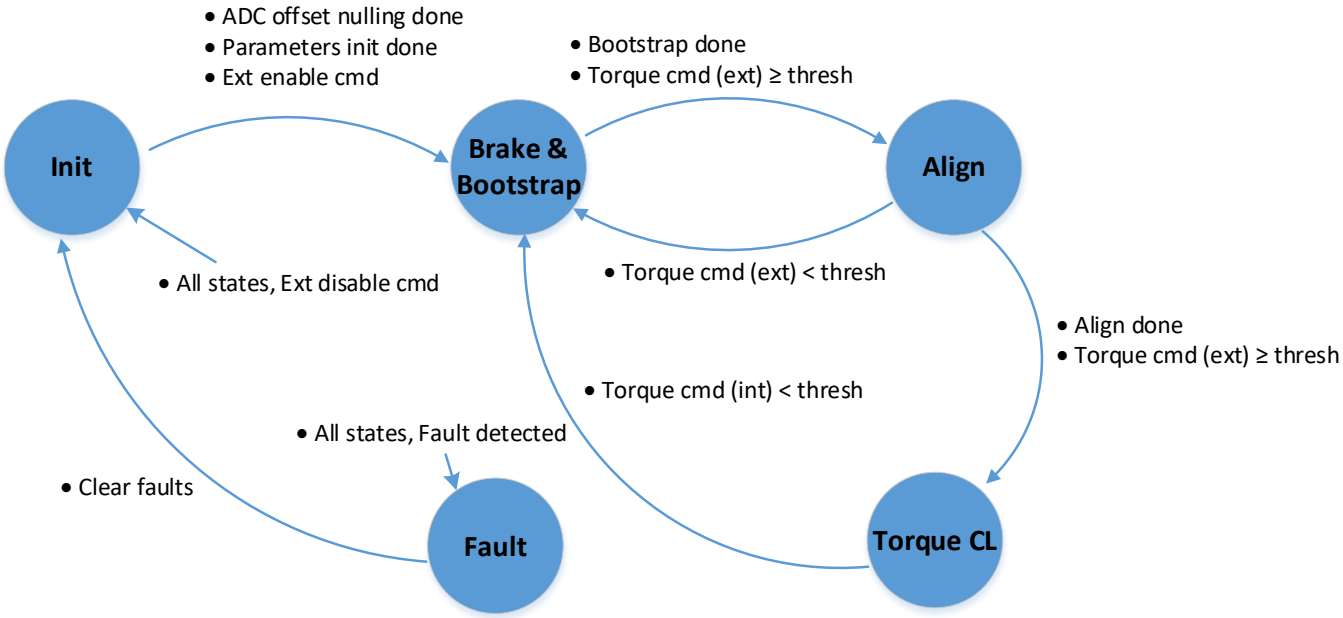

Fig. 109: SFO torque control FOC sensorless with align startup state machine diagram.

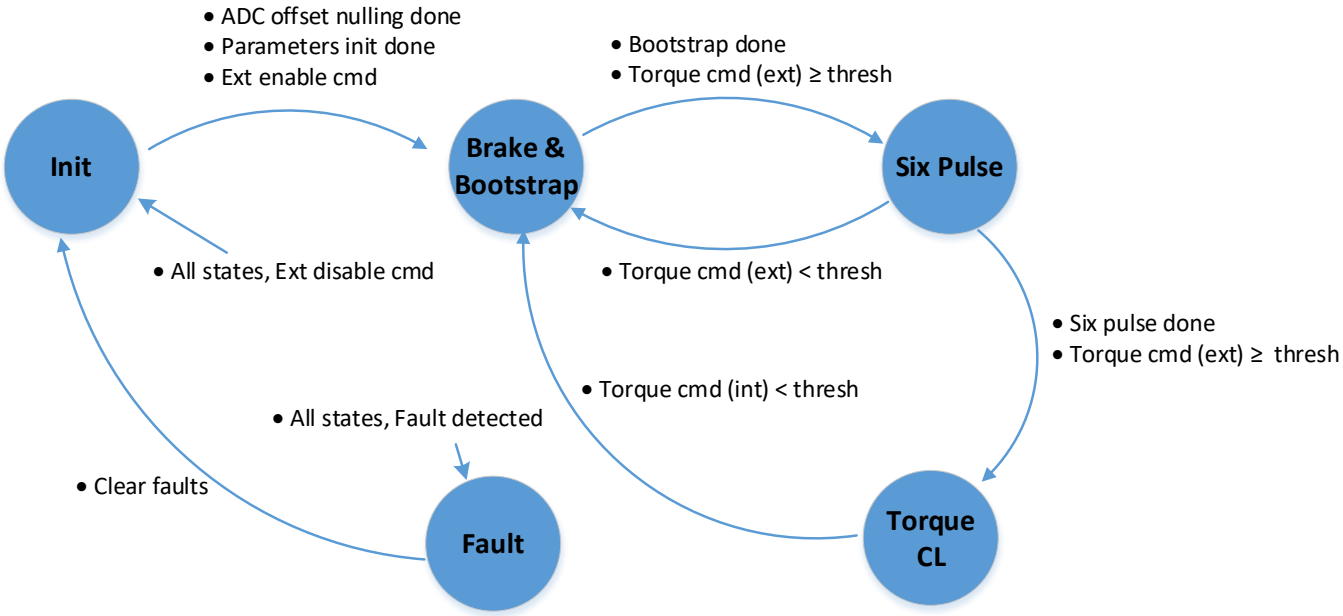

Fig. 110: SFO torque control FOC sensorless with six pulse startup state machine diagram.

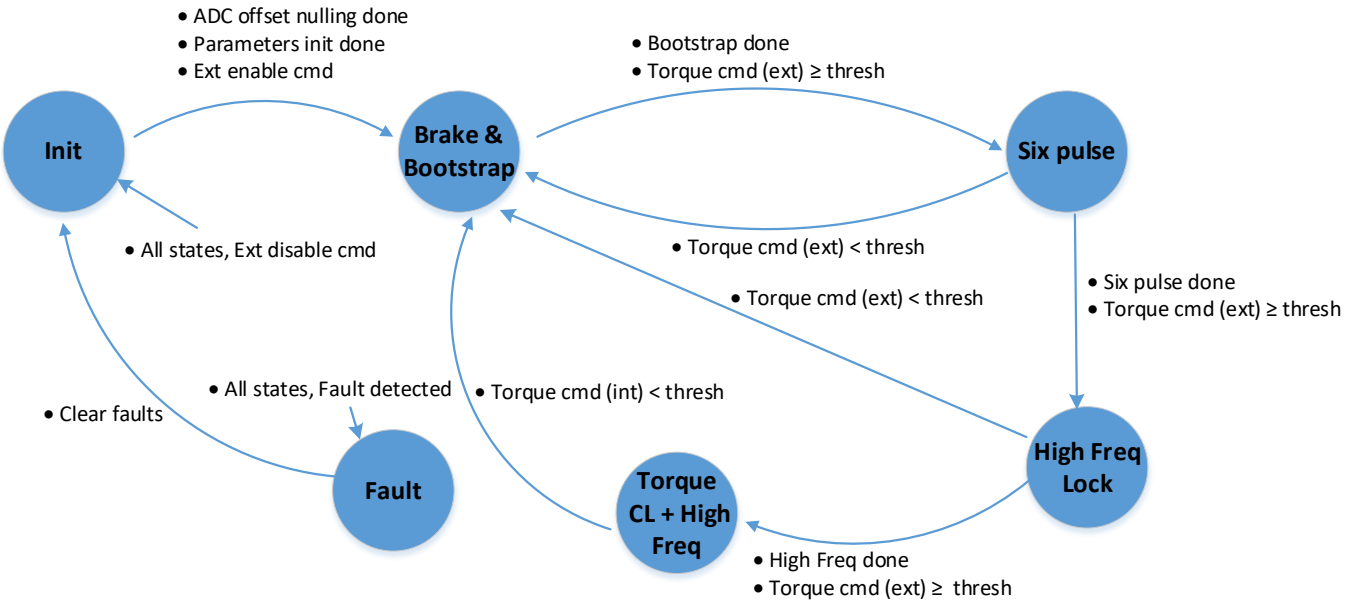

Fig. 111: SFO torque control FOC sensorless with high frequency startup state machine diagram.

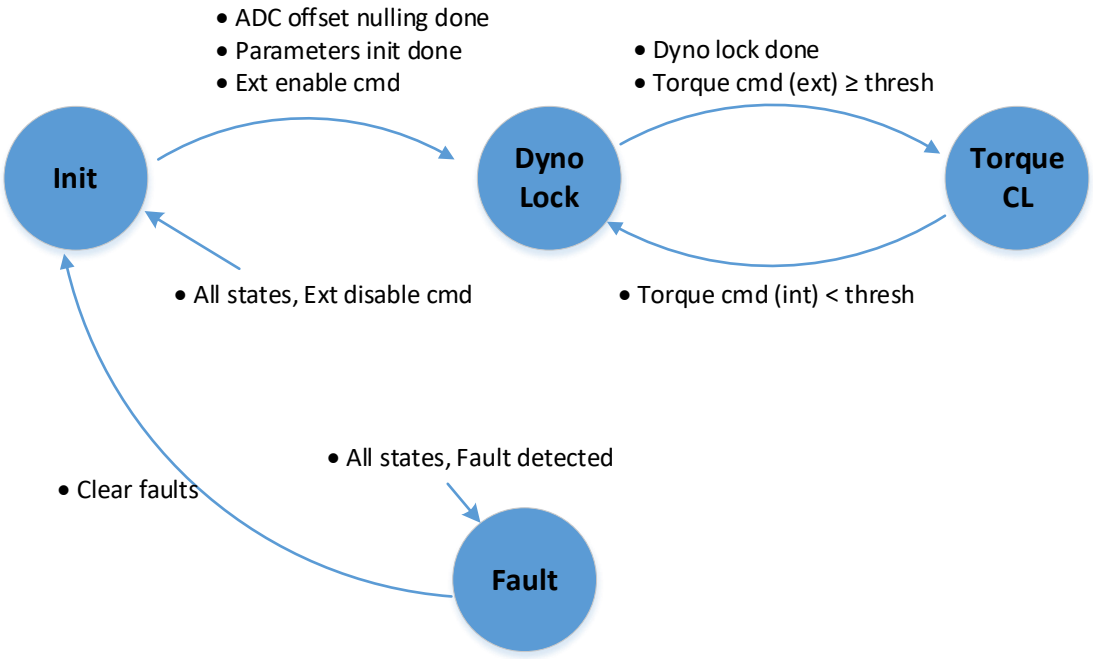

Fig. 112: SFO dyno torque control sensorless FOC state machine diagram.

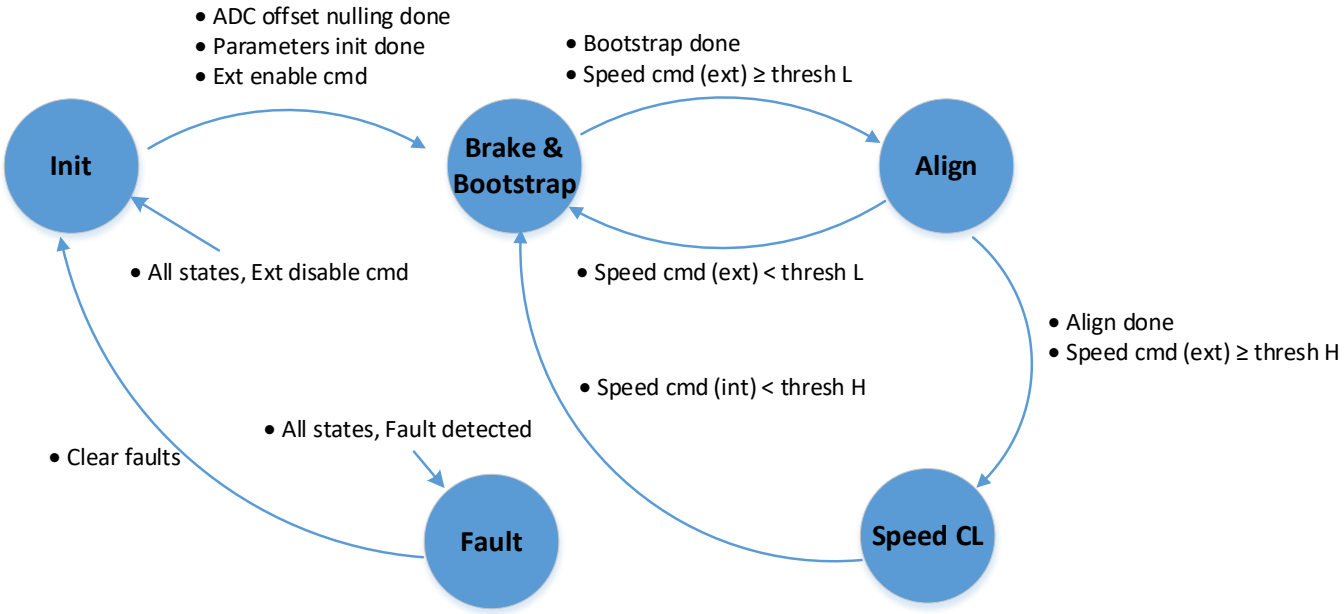

Fig. 113: RFO & SFO speed control FOC sensorless with align startup state machine diagram.

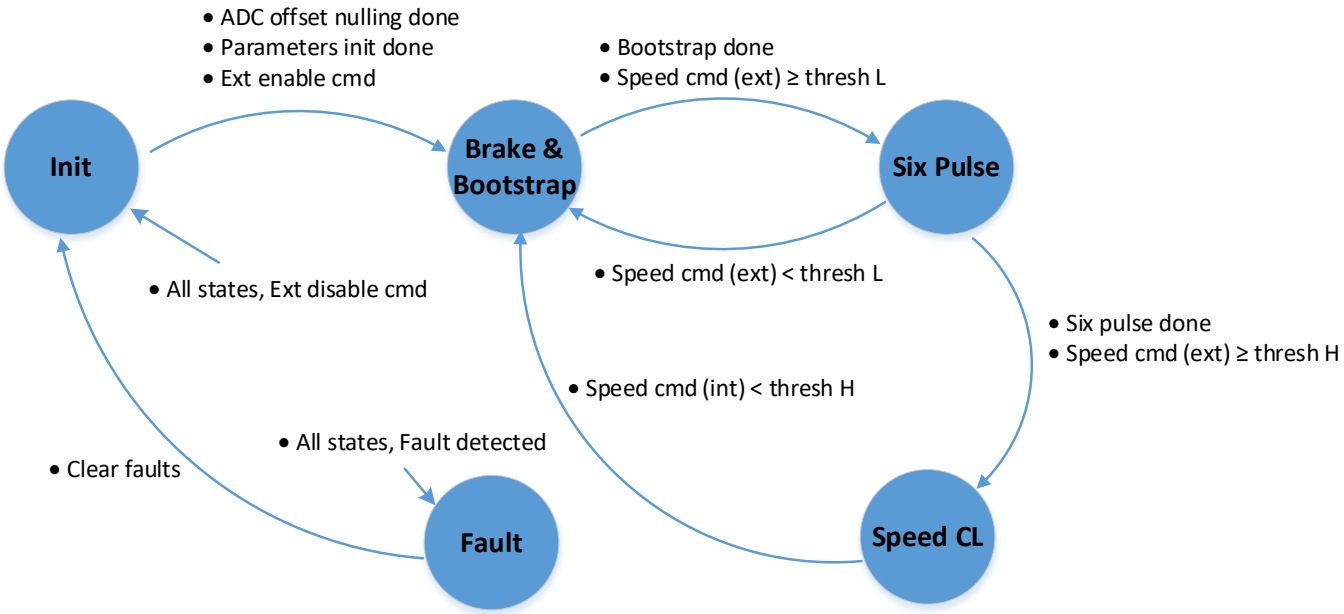

Fig. 114: RFO & SFO speed control FOC sensorless with six pulse startup state machine diagram.

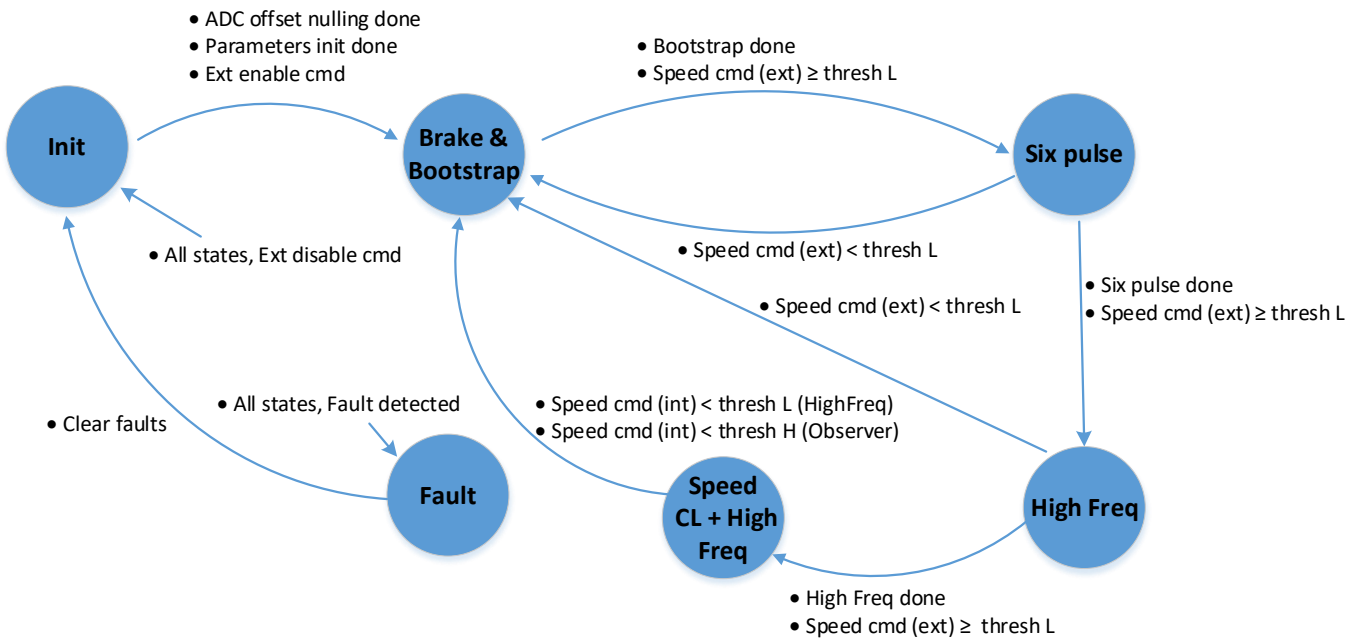

Fig. 115: RFO & SFO speed control FOC sensorless with high frequency startup state machine diagram.

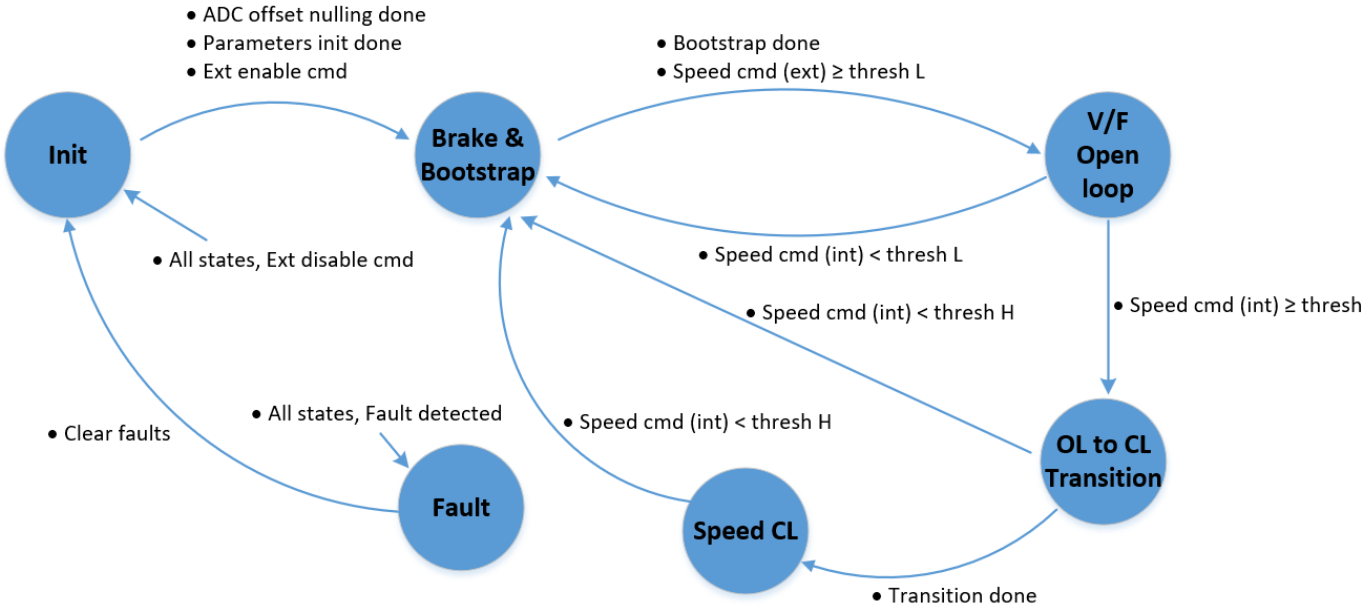

Fig. 116: RFO & SFO speed control FOC sensorless with V/F startup state machine diagram.

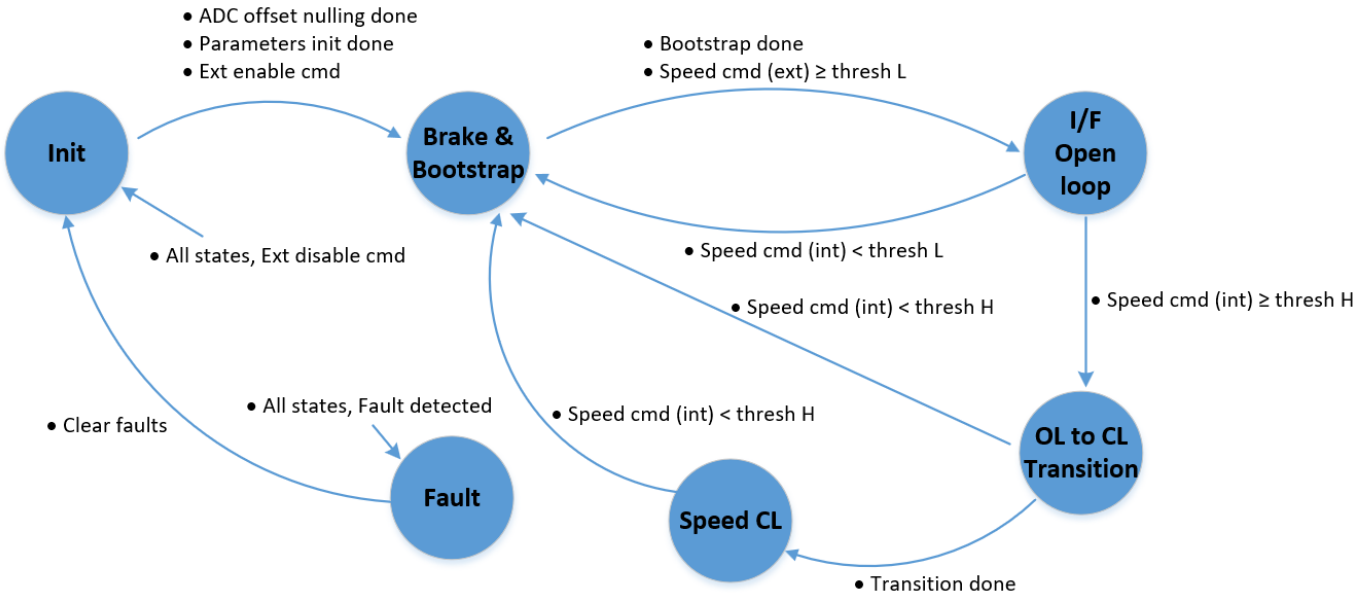

Fig. 117: RFO & SFO speed control FOC sensorless with I/F startup state machine diagram.

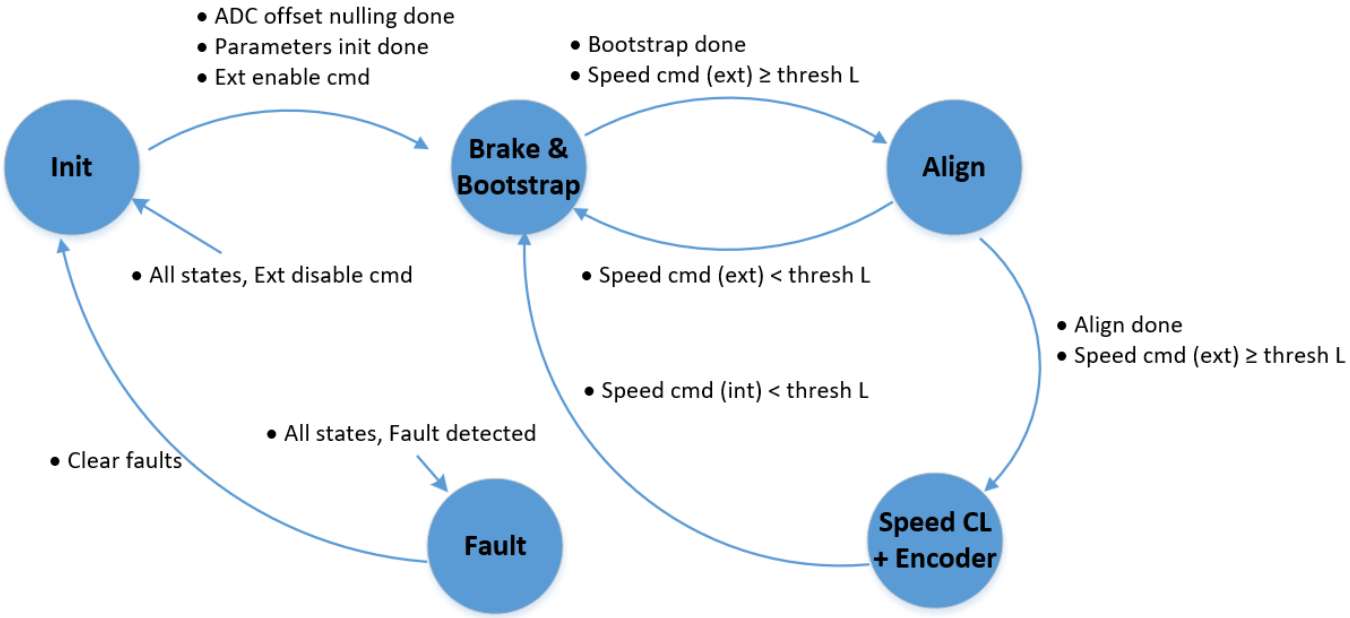

Fig. 118: RFO speed control using encoder with rotor pre-alignment startup diagram.

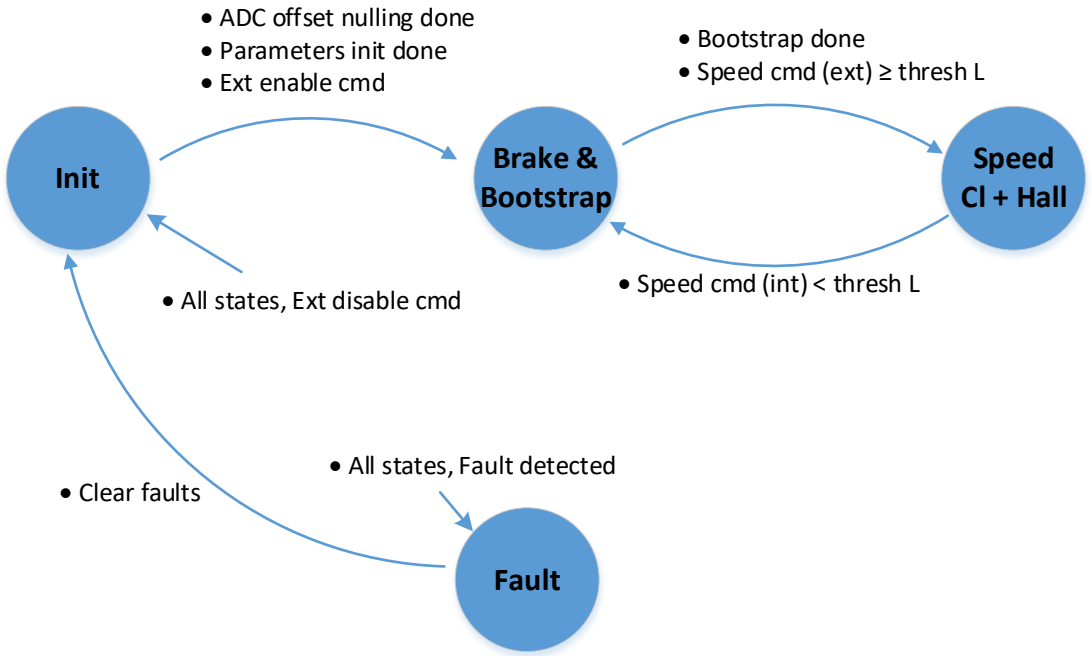

Fig. 119: RFO speed control FOC hall state machine diagram.

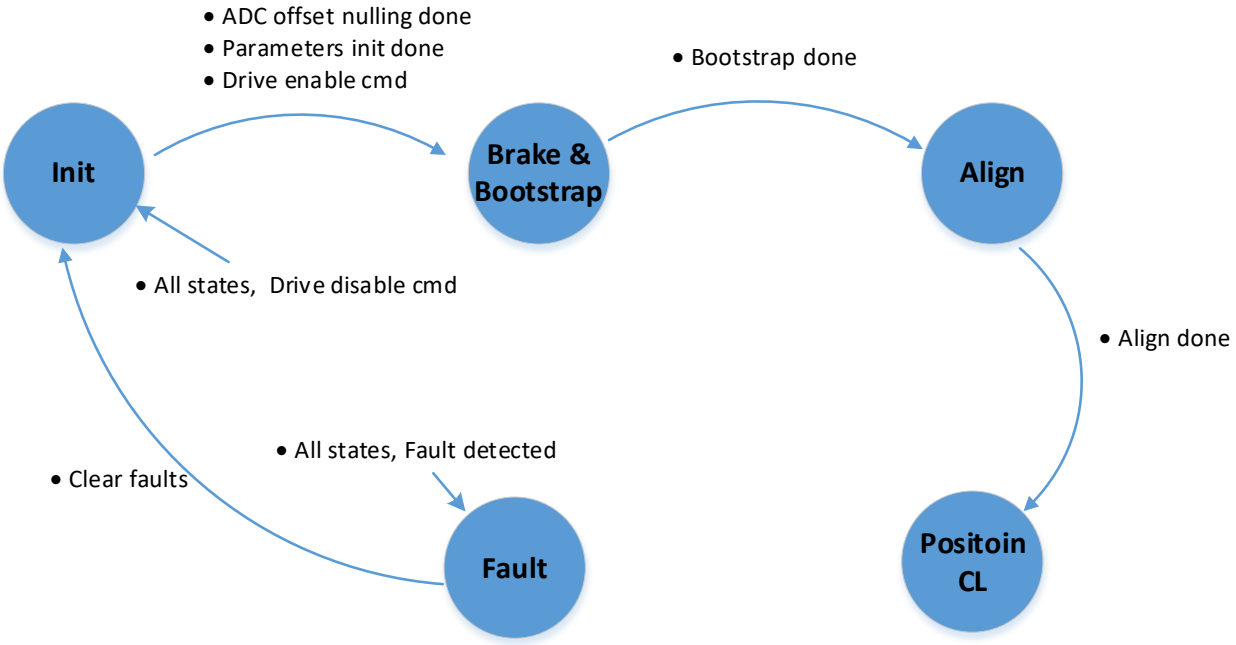

Fig. 120: RFO position control FOC encoder state machine diagram.

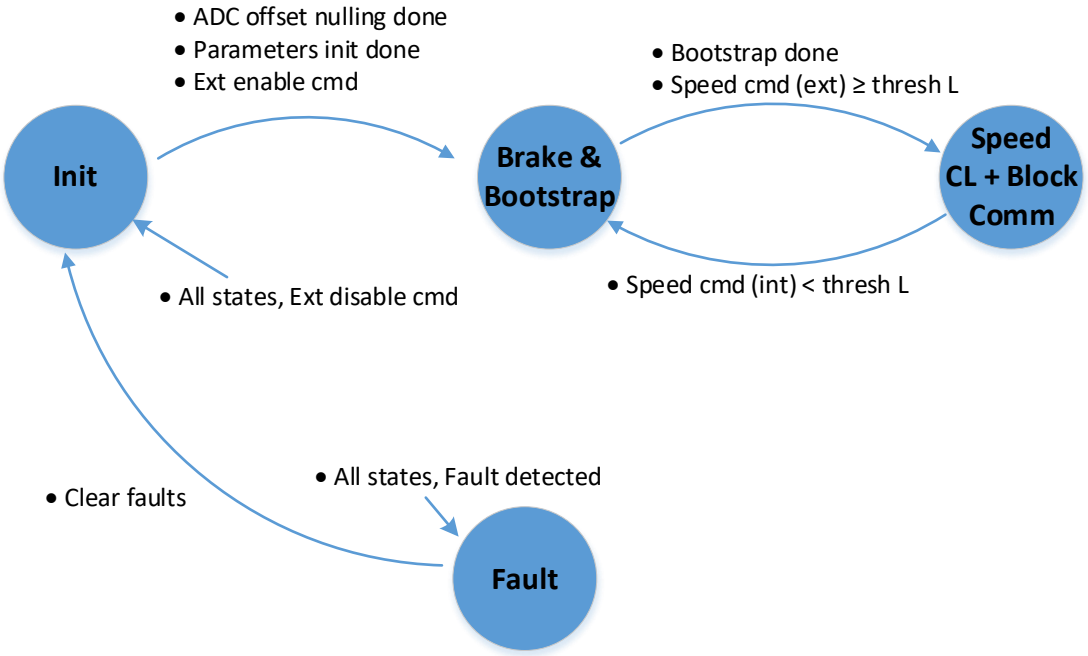

Fig. 121: TBC block commutation speed control state machine diagram.

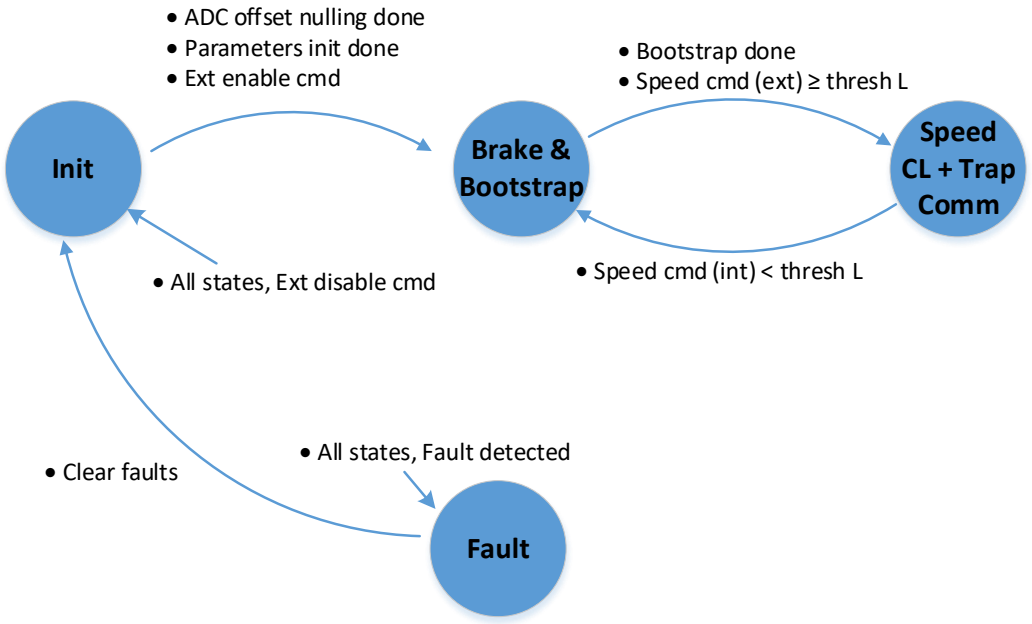

Fig. 122: TBC trapezoidal commutation speed control state machine diagram.

6.17 Peripherals

In this section, the peripherals and their interconnections which firmware configures and relies on will be discussed. The peripherals explained here belong to PSOC6, XMC7200, XMC4400 and PSOC-C3. The same functionalities are expected among these four MCUs. However, the way that hardware is set up to achieve these functionalities is flexible and changes from one MCU to another (e.g. ADC sampling, pin numbers, etc.). All such hardware dependencies should be encapsulated and addressed by the BSP that is created for that specific hardware kit (which includes and describes both MCU board and power board).

Main peripherals such as TCPWMs, ADCs, SCBs, DMAs, etc. are discussed. Functionalities of MCU pins that have been used in the firmware such as hall inputs, encoder input, general debugging pins, etc. are also described. A diagram can be found for each group below. These diagrams display the peripheral descriptions and their given names in the board support package.

6.17.1 TCPWM

The TCPWM block is configured so that it covers both single-shunt and three-shunt hardware configurations. Therefore, one board support package will work for both, and the user doesn't need to regenerate the peripheral configuration through the device configurator if they choose to switch to a single- shunt. Fig. 123 depicts the TCPWM block's device configuration for PSOC6:

- The six output gate driver signals that run the inverter are generated by TCPWM unit and shown in Fig. 123 as ***PWM_U*** (L&H), ***PWM_V*** (L&H) and ***PWM_W*** (L&H).
- Three other TCPWM units are defined for ADC sampling, synchronization, and ISR generation including ***ADC0_ISR0*** (same frequency as ISR0), ***PWM_SYNC*** (same frequency as ISR0), and ***SYNC_ISR1*** (same frequency as ISR1).
- ***SYNC_ISR1*** PWM is used as a reload signal to synchronize all PWMs[UVW], ***ADC0_ISR0*** and ***PWM_SYNC***, in addition to generating ISR1 interrupt.
- Using ***ADC0_ISR0***, ADC sampling of currents at ***CC0*** and ***CC1*** match are synchronized with generated PWMs[UVW]. The ADCs individually trigger a ***DMA*** transaction upon completion. ***ISR0*** starts after both ***DMA***s are done. This way CPU starts ISR0 only after both sample chains are ready in either single or three-shunt configurations.
- Swapping the updated duty cycle values (***CC0*** and ***CC1*** of each PWMs[UVW]) is synchronized with generated PWMs[UVW] using ***PWM_SYNC***.
- For single-shunt configuration, the timing of samples (***ADC0_ISR0 CC0*** and ***CC1*** match) is calculated based on the duty cycle of PWMs[UVW] in the latest ***ISR0*** interrupt.
- For three-shunt solution ***ADC0_ISR0 CC0*** and ***CC1*** are identical to ***CC0*** of ***PWM_SYNC*** which is half PWMs' period.
- For both single-shunt and three-shunt configurations, swapping the updated duty cycle values (***CC0*** and ***CC1*** of each PWM[UVW]) happens at ***PWM_SYNC CC0***.
- The TCPWM device configuration for XMC7200, XMC4400, and PSOC-C3 are almost the same as the one implemented for PSOC6 and shown in Fig. 123. Note that the PWM output pin numbers assigned are different among these MCUs (see Table 6, Table 7, and Table 8).
- PSoC-C3 TCPWM device configuration in general follows PSoC6 deign with one difference. In PSoC-C3 it is not possible to route the first trigger output of a TCPWM to the ADC module. That means to have two triggers for ADCs, another timer needs to be defined other than ***ADC0_ISR***. That is why for PSoC-C3 a second timer called ***ADC1_ISR*** is added with a period equal to ***ADC0_ISR*** timer and in phase with it.

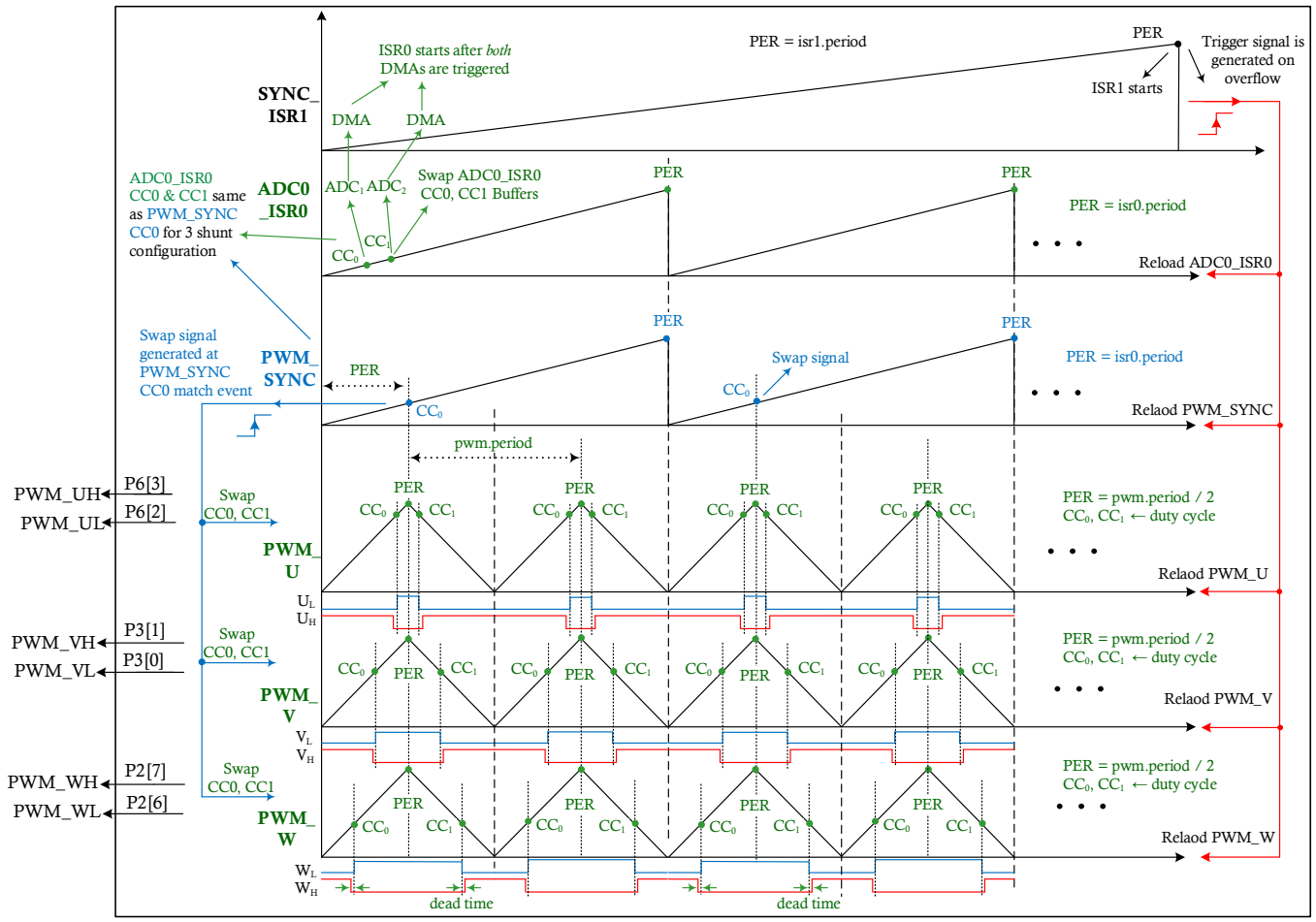

Fig. 123: TCPWM device configuration for PSoC6.

6.17.2 ADC

The ADC configuration is explained for the three-shunt and single-shunt configurations. There are two SAR ADC units in PSOC6 which enable simultaneous sampling of 2 phase currents, phase U and phase V as shown in Fig. 124. This is important for three-shunt system since simultaneous sampling of 2 phase currents is necessary. For the single-shunt system there is only one current that needs to be sampled (IDC) as shown in Fig. 125. For both three-shunt and single-shunt, there are also voltage signals that are sampled using the two SAR ADCs. Fig. 124 and Fig. 125 respectively depict the ADC block's device configuration for the three-shunt and the single-shunt:

- ADC sampling happens every ISR0 period.
- As described in Fig. 123 and in the TCPWM section, ***ADC0_ISR0*** PWM is used to trigger sampling at different points in the period. The SOC (start of conversion) for ***SAR ADC_0*** and ***SAR ADC_1*** is respectively ***ADC0_ISR0 CC0*** match and ***CC1*** match.
- For three-shunt, ***ADC0_ISR0 CC0*** and ***CC1*** are happening at the same time, which is half of PWM [UVW] period. For single-shunt, values of ***ADC0_ISR0 CC0*** and ***CC1*** are calculated every sampling cycle at ***ISR0*** by firmware.
- For both ***SAR ADC_0*** and ***SAR ADC_1,*** EOS (end of sampling) signal is generated once sampling is done which will trigger the DMA transfers.

Other than three current signals, there is no need to sample at every ISR0 period for other signals. That is one of the reasons why the firmware currently changes the analog routing MUX so that it can ping pong between two different sample sets (as shown in Fig. 124 and Fig. 125). Another reason for such time-multiplexing scheme is limited analog routing resources in PSOC6, i.e. not all pins can be routed to the ADC channels at the same time. Also, the device configurator only generates one routing configuration

and cannot generate multiple routing configurations that can be switched at runtime. Therefore, the device configurator is used to generate the routing configuration denoted by ***ADC_Routing_MUXA*** for three-shunt and ***ADC_Routing_MUXB*** for the single-shunt. Each of these configurations contains two routing options: ***ADC_Routing_MUXA0***, ***ADC_Routing_MUXA1*** for three-shunt and ***ADC_Routing_MUXB0***, ***ADC_Routing_MUXB1*** for single-shunt. These two options are alternatively selected by SW in each ISR0 cycle.

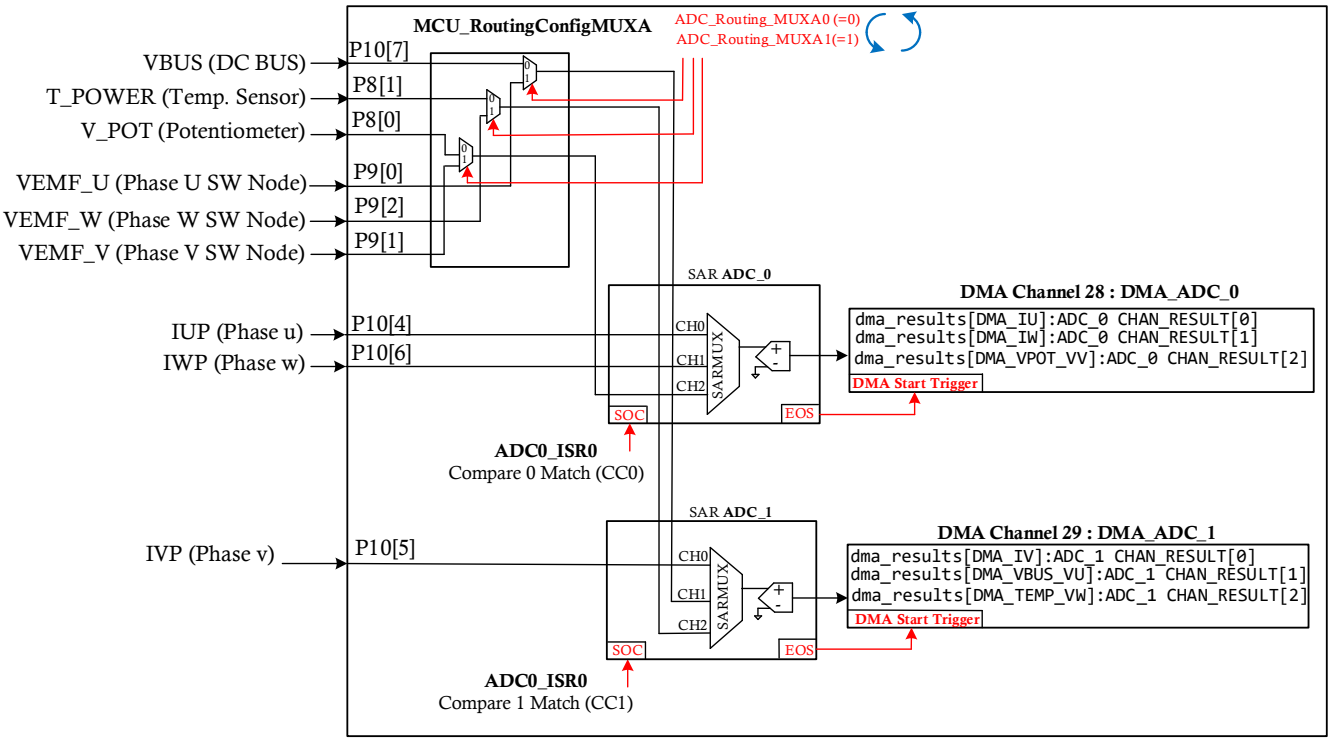

Fig. 124: Three-shunt ADC device configuration for PSoC6.

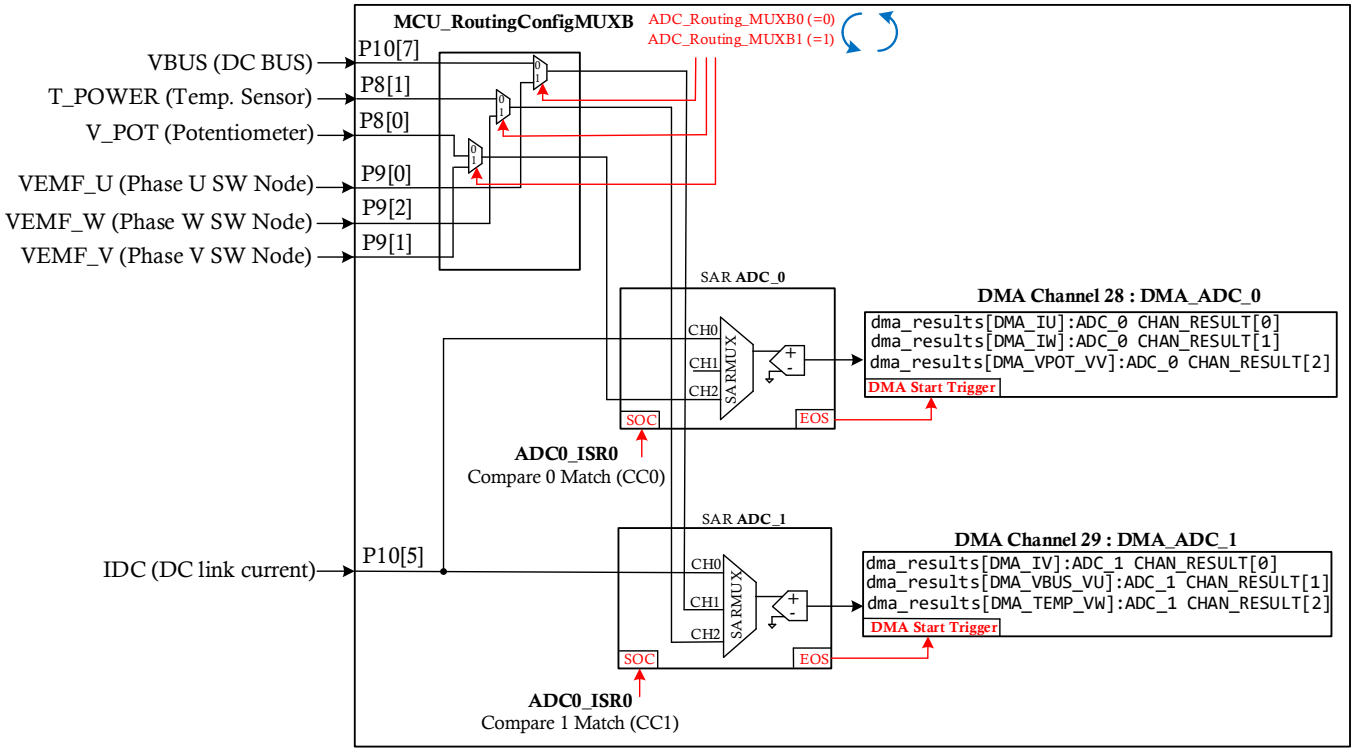

Fig. 125: Single-shunt ADC device configuration for PSoC6.

For the other microcontrollers including XMC7200, XMC4400 and PSOC-C3, there is no need for “ping pong type” analog routing MUX depicted in Fig. 124 and Fig. 125 because all analog pins can be routed to all ADC channels. Nevertheless, the routing multiplexer must be part of the BSP and whatever configuration is used (either MUXing the analog routings or not) should be encapsulated in the BSP. The firmware’s HW-agnostic library only expects the ADC samples to be captured and ready, e.g. ***sensor_iface.i_u, sensor_iface.i_v, sensor_iface.i_w, sensor_iface.v_dc***, etc.

Fig. 126 shows the three-shunt ADC device configurations for XMC4400. The same configuration is used for the single-shunt system.

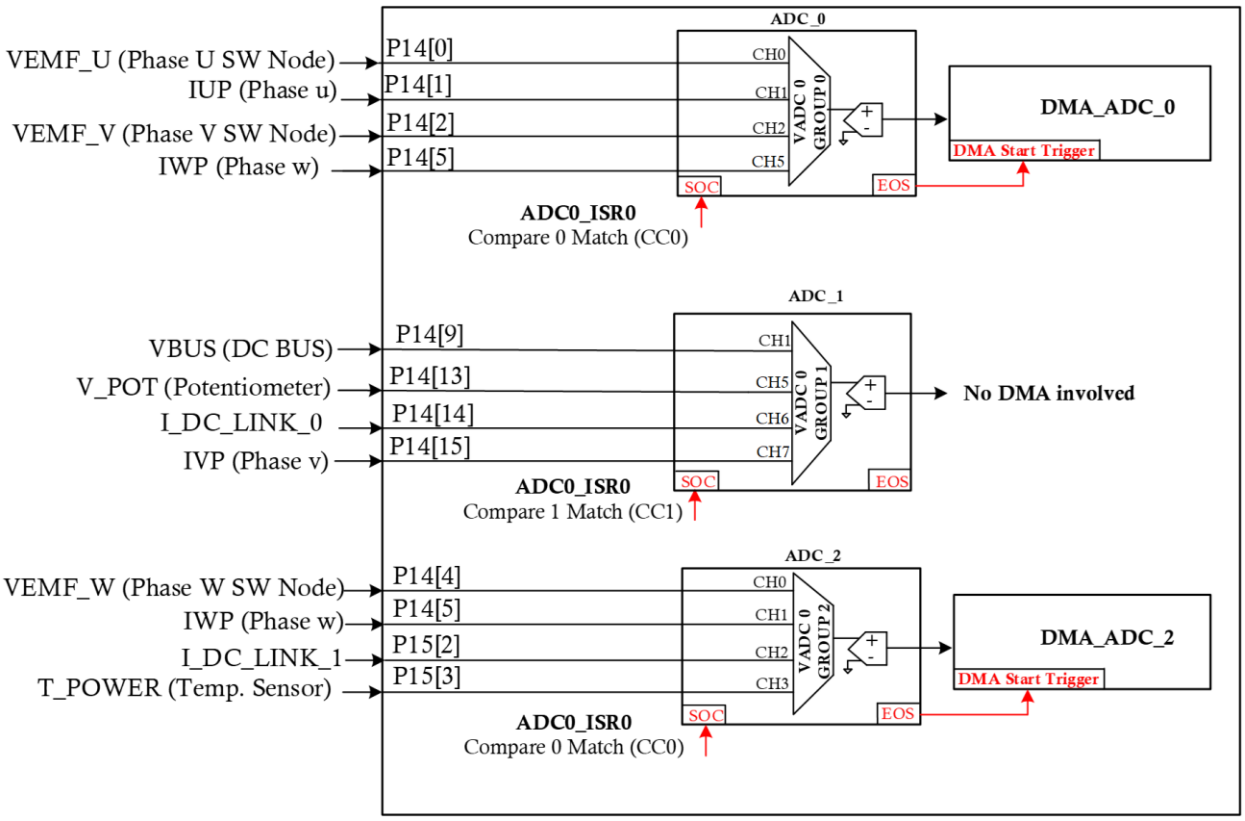

Fig. 126: Three-shunt and single-shunt systems ADC device configuration for XMC4400.

Fig. 127 and Fig. 128 respectively show the three-shunt and single-shunt ADC device configurations for XMC7200.

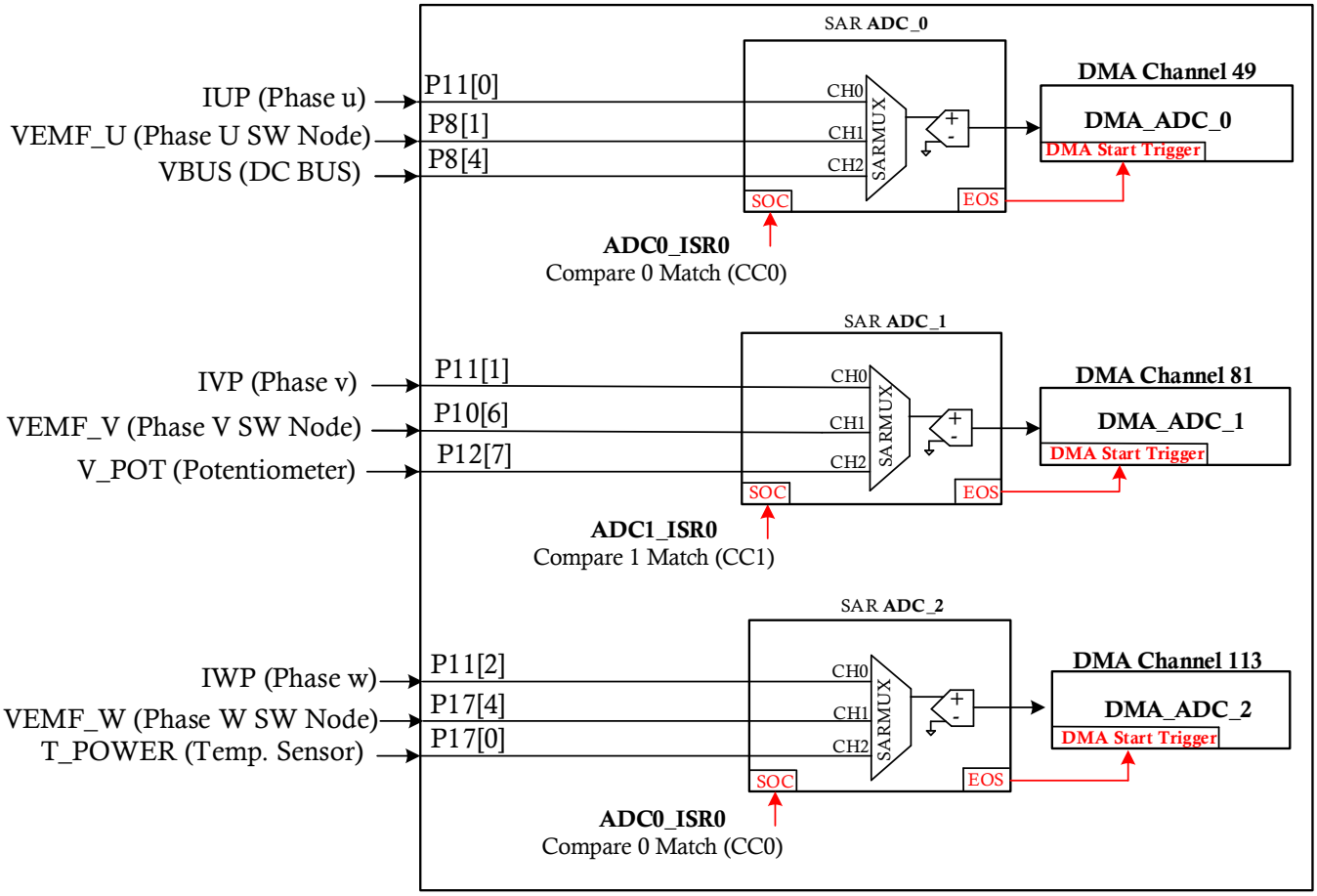

Fig. 127: Three-shunt ADC device configuration for XMC7200.

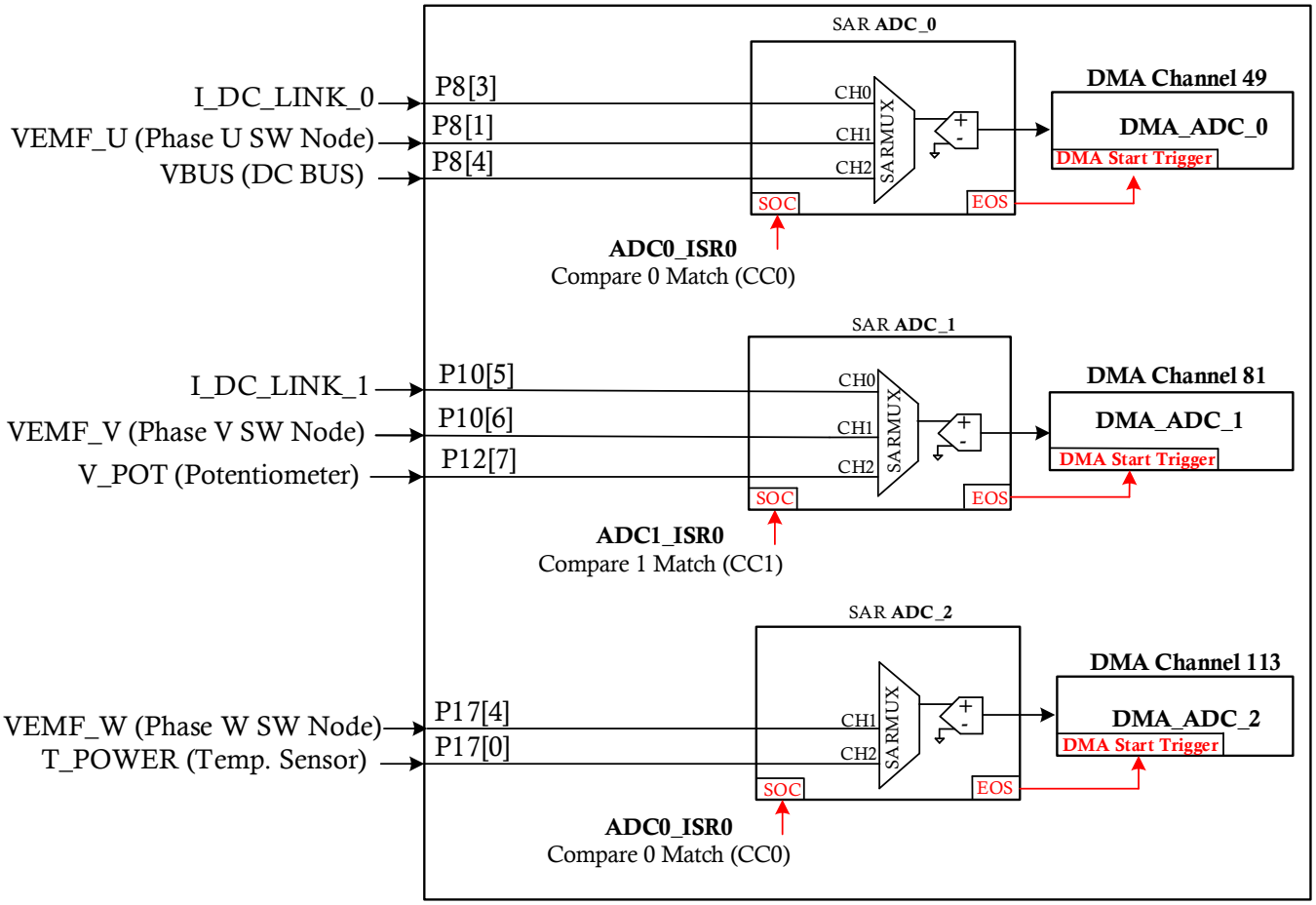

Fig. 128: Single-shunt system ADC device configuration for XMC7200.

Fig. 129, shows the three-shunt ADC device configurations for PSoC-C3. The same configuration can used for the single-shunt if I_DC_LINK is sampled twice by ADC_SEQ0 and ADC_SEQ1 as opposed to sampling IU, IV, and IW. The PSoC-C3 device has one 12-bit ADC module with up to 16 parallel sampling channels. The ADC supports multiple S/H, which enables synchronous (simultaneous) sampling on several channels. This is a great advantage particularly in motor control applications.

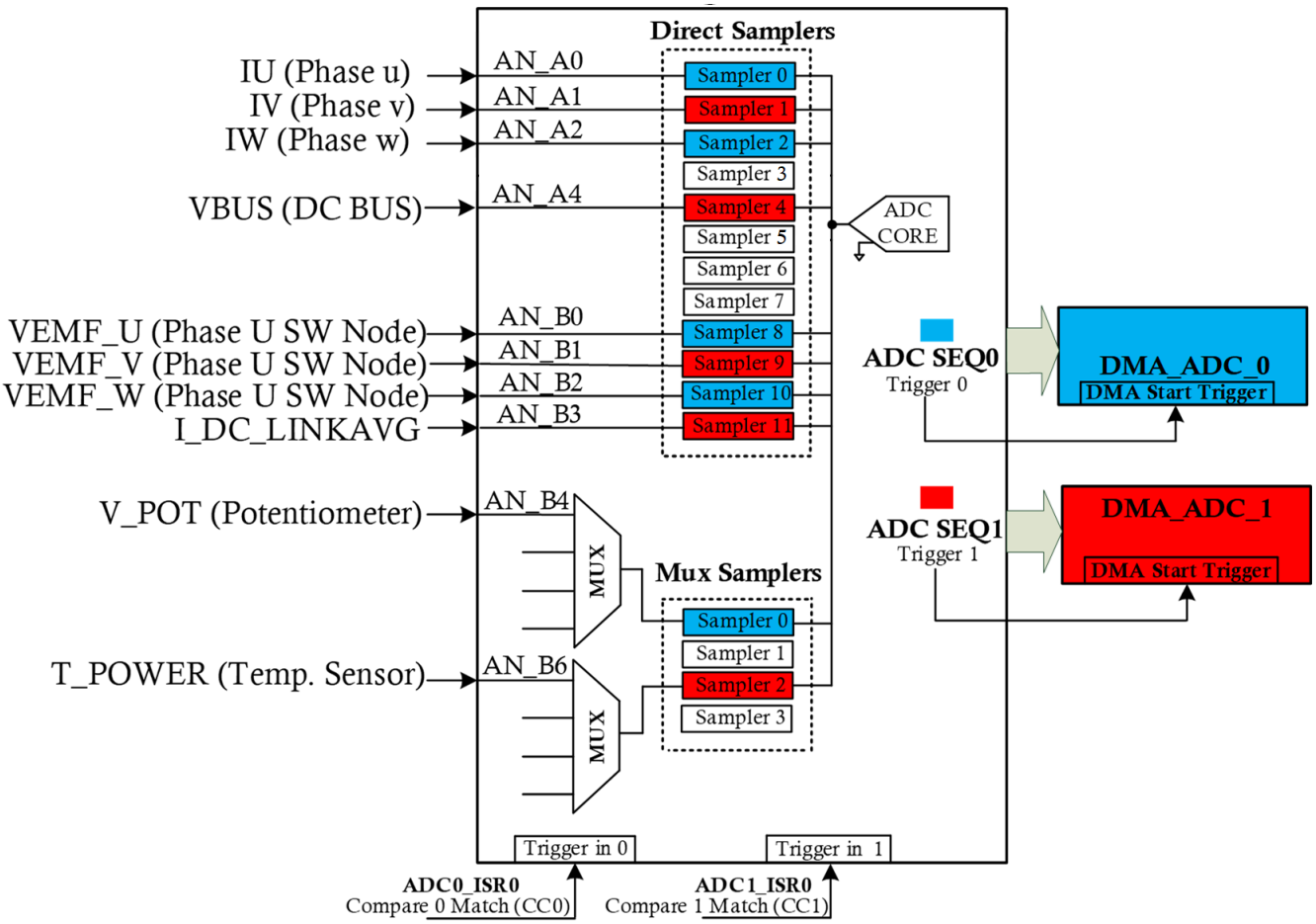

Fig. 129: Three-shunt ADC device configuration for PSoC-C3.

6.17.3 SPI

To communicate with smart gate driver 6EDL7141, the serial communication block 5 in PSOC6 is configured as SPI. As shown in Fig. 130, four pins of MCU are dedicated to SPI. Fig. 130 depicts the SPI device configuration:

- A 16-bit TCPWM unit is configured (SPI_TRANSCATIONS) to generate trigger signals at 125 kHz = 75MHz/600 (or every 8 microseconds) for DMA to transmit data (3 bytes batches) to MCU SCB (and then 6ELD7141).
- The SPI_TRANSACTIONS PWM is free running at 125 kHz. However, for the DMA TX transfer to happen every ISR1, DMA should also be enabled at the beginning of the ISR1 interrupt to kick off the transfer.

The trigger to receive data from 6EDL7141 (RX trigger below) is generated by SCB and consumed by DMA to read the 7 status registers of 6EDL7141.

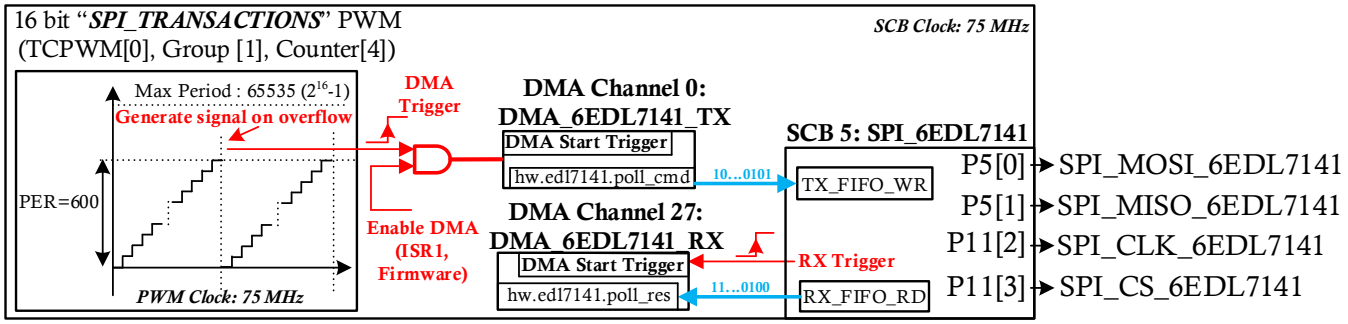

Fig. 130: SPI device configuration.

The SPI and the corresponding DMA described in this section have not been implemented for XMC4400, XMC7200 and PSOC-C3 kits since they don't have smart gate drivers (6EDL7141) on their power boards, i.e., there's no SPI communication between the microcontroller and a smart gate driver.

6.17.4 Hall inputs

There are three pins configured to read hall sensors' output signals that come from the motor. These three signals are connected to the MCU PINs as seen in Fig. 131 (PSOC6). This hall signal information is used for pattern validation and angle calculation inside the MCU. In addition, to measure speed, the hall counter is read on the edges of a signal that is generated by a SW based edge detector. The edge detector triggers any time each one of the three hall signals rises or falls. This happens six times per revolution which means each period will be equivalent to $(\frac{2\pi}{6} =)\frac{\pi}{3}$ rad. The reason to use a 32-bit counter (not 16-bit) is to accommodate very low speed measurements.

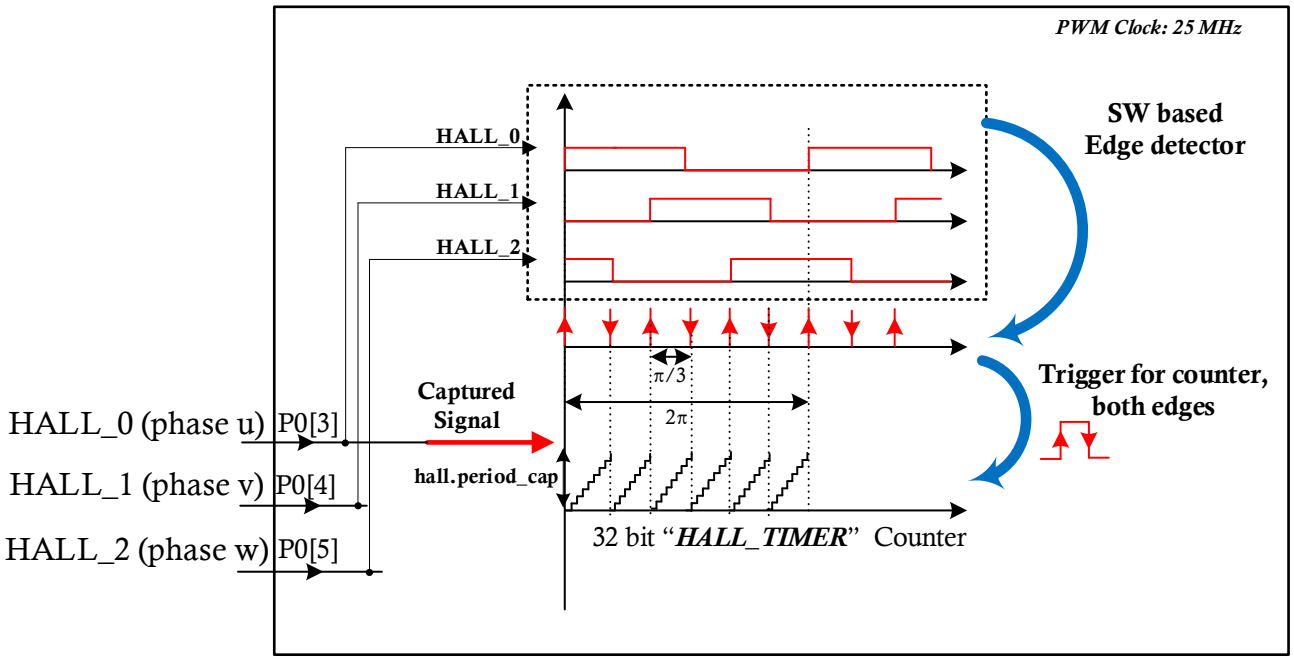

Fig. 131: Hall sensor device configuration.

Currently for the other microcontrollers including XMC7200, XMC4400 and PSOC-C3 hall sensor device configurations have been implemented similarly using TCPWM/CCU. The firmware library expects ***hall.period_cap*** to be updated by either using SW emulation or POSIF.

6.17.5 Miscellaneous GPIOs

Other than the described peripheral blocks and their configuration, other GPIOs are defined in the firmware and intended for miscellaneous purposes. This information is shown in Table 5, Table 6, Table 7 and Table 8 respectively for PSoC6, XMC4400, XMC7200 and PSoC-C3.

Table 5: GPIO assigned for PSoC6.

Pin Number	Pin Name	Pin Mode	Description
P0[0]	**ENC_A**	Digital High-Z	Pulse A is generated from the external encoder and goes to the MCU
P0[1]	**ENC_B**	Digital High-Z	Pulse B is generated from the external encoder and goes to the MCU
P0[2]	**ENC_Z**	Digital High-Z	Pulse Z is generated from the external encoder and goes to MCU
P1[0]	**DIR_SWITCH**	Resistive pull-up	To change the direction of the motor this pin is pulled down externally
P1[1]	**N_BRK_SWITCH**	Resistive pull-up	External switch used for motor braking (active low)
P1[2]	**N_FAULT_SGD**	Resistive pull-up	The fault signal generated by SGD (6EDL7141) and sent to MCU
P2[4]	**N_BRK_SGD**	Strong Drive	MCU generated signal used for motor braking (active low)
P6[4]	**SWO_TDO**	Strong Drive	To be connected to the SWO pin of the J-trace pro connector
P6[5]	**TDI**	Resistive pull-up	To be connected to the TDI pin of the J-trace pro connector
P6[6]	**SWDIO_TMS**	Resistive pull-up	To be connected to the SWDIO pin of the J-trace pro connector

P6[7]	**SWDCK_TCK**	Resistive pull-down	To be connected to the SWD clock pin of the J-trace pro connector
P7[0]	**TRACECLK**	Strong Drive	To be connected to the trace clock pin of the J-trace pro connector
P7[4]	**EN_DRV_SGD**	Strong Drive	The signal generated by MCU to enable/disable 6EDL7141 gate drivers
P7[5]	**WD_CLK**	Strong Drive	Watchdog signal generated by MCU and goes to 6EDL7141, not used currently
P10[0]	**TRACEDATA_3**	Strong Drive	To be connected to trace data [3] pin of J-trace pro connector
P10[1]	**TRACEDATA_2**	Strong Drive	To be connected to trace data [2] pin of J-trace pro connector
P10[2]	**TRACEDATA_1**	Strong Drive	To be connected to trace data [1] pin of J-trace pro connector
P10[3]	**TRACEDATA_0**	Strong Drive	To be connected to trace data [0] pin of J-trace pro connector
P11[1]	**N_FAULT_LED_SW**	Strong Drive	SW Fault LED indicator
P12[6]	**XTAL1**	Analog High-Z	MCU external clock
P12[7]	**XTAL2**	Analog High-Z	MCU external clock

Table 6: GPIO assigned for XMC4400.

Pin Number	Pin Name	Pin Mode	Description
P0[1]	**PWMVL**	Push-pull	Phase V low side PWM
P0[2]	**PWMUL**	Push-pull	Phase U low side PWM
P0[4]	**PWMVH**	Push-pull	Phase V high side PWM
P0[5]	**PWMUH**	Push-pull	Phase U high side PWM
P0[6]	**PWMWH**	Push-pull	Phase W high side PWM
P0[7]	**N_FAULT_HW**	Tristate	HW fault signal to MCU
P0[11]	**PWMWL**	Push-pull	Phase W low side PWM
P0[12]	**TEST_PIN0**	Push-pull	General purpose debugging
P1[0]	**N_HALL_EN, ENC_EN**	Push-pull	Hall enable switch (active low)/ Encoder (active high)
P1[1]	**HALL_2, ENC_Z**	Tristate	Hall input 2/ ENC input Z
P1[4]	**TEST_PIN2**	Push-pull	General purpose debugging
P1[5]	**TEST_PIN3**	Push-pull	General purpose debugging
P2[2]	**DIR_LED**	Push-pull	Direction status led indicator
P2[15]	**FAULT_LED_ALL**	Push-pull	Fault LED indicator
P4[1]	**TEST_PIN1**	Push-pull	General purpose debugging
P5[0]	**CYBSP_DEBUG_UART_RX**	Push-pull	UART RX
P5[1]	**CYBSP_DEBUG_UART_TX**	Push-pull	UART TX
P14[6]	**HALL_1, ENC_B**	Tristate	Hall input 1/ ENC input B
P14[7]	**HALL_0, ENC_A**	Tristate	Hall input 0/ ENC input A

Table 7: GPIO assigned for XMC7200.

Pin Number	Pin Name	Pin Mode	Description
P1[2]	**N_FAULT_HW**	Digital High-Z	HW fault signal to MCU
P3[0]	**SPI_MISO**	Strong Drive	SPI MISO pin
P3[1]	**SPI_MOSI**	Digital High-Z	SPI MOSI pin
P3[2]	**SPI_CLK**	Digital High-Z	SPI Clock pin
P3[3]	**SPI_CS**	Digital High-Z	SPI CS pin
P5[2]	**HALL_0**	Digital High-Z	Hall input 0
P5[3]	**HALL_2**	Digital High-Z	Hall input 2
P5[4]	**HALL_1**	Digital High-Z	Hall input 1
P5[5]	**ENC_Z**	Digital High-Z	ENC input Z
P6[0]	**PWMUH**	Strong Drive	Phase U high side PWM
P6[1]	**PWMUL**	Strong Drive	Phase U low side PWM
P6[2]	**PWMVH**	Strong Drive	Phase V high side PWM
P6[3]	**PWMVL**	Strong Drive	Phase V low side PWM
P6[4]	**PWMWH**	Strong Drive	Phase W high side PWM
P6[5]	**PWMWL**	Strong Drive	Phase W low side PWM
P6[6]	**ENC_A**	Digital High-Z	ENC input A
P6[7]	**ENC_B**	Digital High-Z	ENC input B
P7[0]	**TEST_PIN0**	Strong Drive	General purpose debugging
P7[1]	**TEST_PIN1**	Strong Drive	General purpose debugging
P7[2]	**TEST_PIN2**	Strong Drive	General purpose debugging
P7[3]	**TEST_PIN3**	Strong Drive	General purpose debugging
P7[4]	**TEST_PIN4**	Strong Drive	General purpose debugging
P14[2]	**DIR_LED**	Strong Drive	Direction status led indicator

P19[0]	**PWM_SYNC_TEST**	Strong Drive	Test pin
P21[0]	**CYBSP_WCO_IN**	Analog High-Z	External RTC IN pin
P21[1]	**CYBSP_WCO_OUT**	Analog High-Z	External RTC OUT pin
P21[2]	**CYBSP_ECO_IN, XTAL1**	Analog High-Z	External Crytal IN pin
P21[3]	**CYBSP_ECO_OUT, XTAL2**	Analog High-Z	External Crystal OUT pin
P21[7]	**FAULT_LED_ALL**	Strong Drive	Fault LED indicator
P23[2]	**N_DIR_PUSHBTN**	Resistive pull-up	Direction change Push Button
P23[4]	**CYBSP_SWO**	Strong Drive	Debugger SWO pin
P23[5]	**CYBSP_SWDCK**	Resistive pull-down	Debugger SWDCK pin
P23[6]	**CYBSP_SWDIO**	Resistive pull-up	Debugger SWDIO pin

Table 8: GPIO assigned for PSoC-C3.

Pin Number	Pin Name	Pin Mode	Description
P1[2]	**CYBSP_SWDCK**	Resistive pull-down	Debugger SWD clock pin
P1[3]	**CYBSP_SWDIO**	Resistive pull-up	Debugger SWDIO pin
P4[0]	**PWMUH**	Strong Drive	Phase U high side PWM
P4[1]	**PWMUL**	Strong Drive	Phase U low side PWM
P4[2]	**PWMVH**	Strong Drive	Phase V high side PWM
P4[3]	**PWMVL**	Strong Drive	Phase V low side PWM
P4[4]	**PWMWH**	Strong Drive	Phase W high side PWM
P4[5]	**PWMWL**	Strong Drive	Phase W low side PWM
P4[6]	**N_DIR_PUSHBTN**	Digital High-Z	Direction change Push button
P4[7]	**N_HALL_EN, ENC_EN**	Strong Drive	Hall enable switch (active low)/ Encoder (active high)
P7[4]	**HALL_0, ENC_A**	Digital High-Z	Hall input 0/ ENC input A
P7[5]	**HALL_1, ENC_B**	Digital High-Z	Hall input 1/ ENC input B
P7[6]	**HALL_2, ENC_Z**	Digital High-Z	Hall input 2/ ENC input Z
P8[0]	**N_FAULT_HW**	Digital High-Z	HW fault signal to MCU
P8[2]	**TEST_PIN4**	Strong Drive	General purpose debugging
P8[3]	**TEST_PIN5**	Strong Drive	General purpose debugging
P8[4]	**TEST_PIN6**	Strong Drive	General purpose debugging
P8[5]	**TEST_PIN7**	Strong Drive	General purpose debugging
P9[0]	**TEST_PIN0**	Strong Drive	General purpose debugging
P9[1]	**TEST_PIN1**	Strong Drive	General purpose debugging
P9[2]	**TEST_PIN2**	Strong Drive	General purpose debugging
P9[3]	**TEST_PIN3**	Strong Drive	General purpose debugging
P9[4]	**DIR_LED**	Strong Drive	Direction status led indicator
P9[5]	**FAULT_LED_ALL**	Strong Drive	Fault LED indicator

6.18 Profiler

For a motor control system to run smoothly, the motor parameters, mechanical load parameters, and sometimes open-loop V/F parameters (for sensorless V/F startup) should be properly extracted. The profiler module automates extraction of these parameters so that the user can tune the system when using a new motor and/or mechanical load.

It is important to note that the profiler is not a substitute for engineering judgement and expertise in tuning motor control systems. It is merely a tool to help facilitate that process.

For example, the profiler makes some simplified assumptions about the nature of the system such as a first order mechanical load (inertia and friction) whereas some mechanical loads have resonances (where there is a high inertia mismatch present between the motor and the load). Therefore, the application engineer should always fine tune the parameters in the end based on the nature of the application.

The following sections describe the parameter extraction algorithms for motor, mechanical load, and open-loop V/F. The terms “motor profiler”, “mechanical profiler”, and “V/F profiler” refer to the parameter extraction for each of these subsystems, respectively. All of these “subsystem profilers” operate under the umbrella of the “profiler” module.

6.18.1 Motor Profiler

It is of utmost importance to accurately estimate the motor parameters in order to tune the controllers, e.g. current controller, flux controller, torque controller, speed controller, etc.

The main motor parameters that need to be extracted are as follows

- Stator resistance, r
- Stator q-axis inductance, L_q
- Stator d-axis inductance, L_d
- Rotor permanent-magnet flux linkage, λ_m

6.18.1.1 Parameters Extraction Procedure

The stator voltages of a PMSM, whether IPM or SPM, can be expressed by the so-called algebraic-differential equations as

$$\begin{cases} \boldsymbol{v}_{qd}^{\hat{\theta}} = r\boldsymbol{i}_{qd}^{\hat{\theta}} + \dfrac{d\boldsymbol{\lambda}_{qd}^{\hat{\theta}}}{dt} + \hat{\omega}\begin{bmatrix} \lambda_d^{\hat{\theta}} \\ -\lambda_q^{\hat{\theta}} \end{bmatrix} + \omega\lambda_m \begin{bmatrix} \cos(\tilde{\theta}) \\ -\sin(\tilde{\theta}) \end{bmatrix}, \\ \boldsymbol{\lambda}_{qd}^{\hat{\theta}} = \left(L_0\boldsymbol{I} + L_1\boldsymbol{C}\boldsymbol{R}(2\tilde{\theta})\right)\boldsymbol{i}_{qd}^{\hat{\theta}} \end{cases} \quad (340)$$

where $\boldsymbol{x}_{qd}^{\hat{\theta}}$ denotes a vector of q- and d-axis variables in an arbitrary reference frame $\hat{\theta}$,

$$\boldsymbol{x}_{qd}^{\hat{\theta}} = \begin{bmatrix} x_q^{\hat{\theta}} \\ x_d^{\hat{\theta}} \end{bmatrix}, \quad (341)$$

θ is the real angle of the rotor and $\tilde{\theta}$ is the difference between the real angle of the rotor and the chosen reference-frame angle, $\hat{\theta}$:

$$\tilde{\theta} = \theta - \hat{\theta} \quad (342)$$

According to (340), the difference between the real angle of the rotor and the chosen reference frame angle, $\tilde{\theta}$, determines the back-emf voltages, $\omega\lambda_m\cos(\tilde{\theta})$ and $-\omega\lambda_m\sin(\tilde{\theta})$, in the chosen reference frame.

For simplifying the equations, instead of L_q and L_d, L_0 and L_1 are used in (340):

$$\begin{cases} L_q = L_0 + L_1 \\ L_d = L_0 - L_1 \end{cases} \leftrightarrow \begin{cases} L_0 = \dfrac{L_q + L_d}{2} \\ L_1 = \dfrac{L_q - L_d}{2} \end{cases} \quad (343)$$

The matrices $\boldsymbol{I}$ and $\boldsymbol{C}$ are respectively the identity and complex-conjugate matrices:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (344)$$

(345)

The rotation matrix $\boldsymbol{R}(2\tilde{\theta})$, also known as, Park Transform is defined as below:

$$\boldsymbol{R}(2\tilde{\theta}) = \begin{bmatrix} \cos(2\tilde{\theta}) & -\sin(2\tilde{\theta}) \\ \sin(2\tilde{\theta}) & \cos(2\tilde{\theta}) \end{bmatrix} \quad (346)$$

Let us assume that the chosen reference frame is the stationary reference frame ($\hat{\theta} = 0$), which is also known as αβ frame:

$$\alpha\beta \text{ stationary frame: } \begin{cases} \hat{\theta} = 0 \\ \hat{\omega} = 0 \end{cases} \quad (347)$$

Thus, we can simplify (340) to

$$\begin{cases} \boldsymbol{v}_{\alpha\beta} = r\boldsymbol{i}_{\alpha\beta} + \dfrac{d\boldsymbol{\lambda}_{\alpha\beta}}{dt} + \omega\lambda_m \begin{bmatrix} \cos(\theta) \\ -\sin(\theta) \end{bmatrix} \\ \boldsymbol{\lambda}_{\alpha\beta} = \left(L_0\boldsymbol{I} + L_1\boldsymbol{C}\boldsymbol{R}(2\theta)\right)\boldsymbol{i}_{\alpha\beta} \end{cases} \quad (348)$$

6.18.1.2 Rotor Locking Stage

In order to extract the motor parameters, first we need to lock the rotor at a certain angular position. The rotor can be locked by applying a large enough dc-current along the α axis and allowing enough time for the rotor to rotate and settle. Once the rotor is settled, the qd and αβ frames will be as shown below:

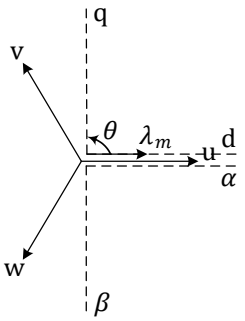

Fig. 132: uvw, qd, and αβ frames after rotor is locked

As can be seen in Fig. 132, after rotor is locked, the rotor angle and rotor angular speed would be:

$$\begin{cases}\theta = \dfrac{\pi}{2} \\ \omega = 0\end{cases} \tag{349}$$

By substituting $\theta = \pi/2$ and $\omega = 0$ in (348), we can obtain:

$$\begin{cases} \boldsymbol{v}_{\alpha\beta} = r\boldsymbol{i}_{\alpha\beta} + \dfrac{d\boldsymbol{\lambda}_{\alpha\beta}}{dt} \\ \boldsymbol{\lambda}_{\alpha\beta} = \left(L_0\boldsymbol{I} + L_1\boldsymbol{CR}(\pi)\right)\boldsymbol{i}_{\alpha\beta} \end{cases} \tag{350}$$

The stator flux linkages, $\boldsymbol{\lambda}_{\alpha\beta}$, can thus be simplified as follows

$$\boldsymbol{\lambda}_{\alpha\beta} = \left(L_0\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + L_1\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}\right)\boldsymbol{i}_{\alpha\beta} = \begin{bmatrix}L_0 - L_1 & 0\\ 0 & L_0 + L_1\end{bmatrix}\boldsymbol{i}_{\alpha\beta} = \begin{bmatrix}L_d & 0\\ 0 & L_q\end{bmatrix}\boldsymbol{i}_{\alpha\beta} \tag{351}$$

The significance of (351) is that it shows once the rotor is locked, the inductance observed along the α axis is indeed the d-axis inductance of the motor, L_d, whereas the inductance observed along the β axis is the q-axis inductance of the motor, L_q.

Hence, the voltage equations in (350) can be rewritten as

$$\begin{cases} v_\alpha = ri_\alpha + L_d\dfrac{di_\alpha}{dt} \\ v_\beta = ri_\beta + L_q\dfrac{di_\beta}{dt} \end{cases} \tag{352}$$

6.18.1.3 Stator Resistance Estimation Stage

When rotor is locked and the dc-currents reach their steady-state value, the resistance of the motor can be estimated using the applied dc voltage, V_α, and the commanded dc current, I_α^*, as follows:

$$r = \frac{V_\alpha}{I_\alpha} = \frac{V_\alpha}{I_\alpha^*} \tag{353}$$

6.18.1.4 Stator Q/D-Axes Inductance Estimation Stage

Now, let us assume that some high-frequency components are also injected to the $\alpha\beta$-axes in addition to the dc component applied to α-axis:

$$\begin{cases} i_\alpha = I_\alpha + \hat{\imath}_\alpha \\ i_\beta = 0 + \hat{\imath}_\beta \end{cases} \tag{354}$$

Therefore, from (352) we would have

$$\begin{cases} V_\alpha + \hat{v}_\alpha = rI_\alpha + r\hat{\imath}_\alpha + L_d\dfrac{d\hat{\imath}_\alpha}{dt} \\ \hat{v}_\beta = r\hat{\imath}_\beta + L_q\dfrac{d\hat{\imath}_\beta}{dt} \end{cases} \tag{355}$$

If the injected high-frequency voltages constitute a counter-clock-wise rotating vector with an angular frequency ω_h, i.e.

$$\hat{\boldsymbol{v}}_{\alpha\beta} = \begin{bmatrix} |\hat{v}_\alpha|\cos(\omega_h t) \\ -|\hat{v}_\beta|\sin(\omega_h t) \end{bmatrix}, \tag{356}$$

then, the resulting high-frequency current magnitudes would be

$$\begin{cases} |\hat{\imath}_\alpha|^2 = \dfrac{|\hat{v}_\alpha|^2}{r^2 + (L_d\omega_h)^2} \\ |\hat{\imath}_\beta|^2 = \dfrac{|\hat{v}_\beta|^2}{r^2 + (L_q\omega_h)^2} \end{cases} \tag{357}$$

In practice, we first inject the high-frequency voltage to the α-axis and measure its corresponding current and then inject the high-frequency current to the β-axis and measure its corresponding current:

$$\begin{cases} \hat{v}_\alpha = |\hat{v}_\alpha|\cos(\omega_h t) \\ \hat{v}_\beta = 0 \end{cases} \Rightarrow \begin{cases} |\hat{\imath}_\alpha|^2 = \dfrac{|\hat{v}_\alpha|^2}{r^2 + (L_d\omega_h)^2} \\ |\hat{\imath}_\beta|^2 = 0 \end{cases} \Rightarrow L_d = ? \tag{358}$$

$$\begin{cases} \hat{v}_\alpha = 0 \\ \hat{v}_\beta = -|\hat{v}_\beta|\sin(\omega_h t) \end{cases} \Rightarrow \begin{cases} |\hat{\imath}_\alpha|^2 = 0 \\ |\hat{\imath}_\beta|^2 = \dfrac{|\hat{v}_\beta|^2}{r^2 + (L_q\omega_h)^2} \end{cases} \Rightarrow L_q = ? \tag{359}$$

As can be seen in (358) and (359), it would be possible to estimate L_d and L_q by processing the measured current magnitudes, $|\hat{\imath}_\alpha|$ and $|\hat{\imath}_\beta|$.

The following figure depicts the block diagram of the system, including the dc-current regulators, the resistance estimation, the injected ac-voltage components, the resulting ac-current components, and the ac-current magnitude extractors:

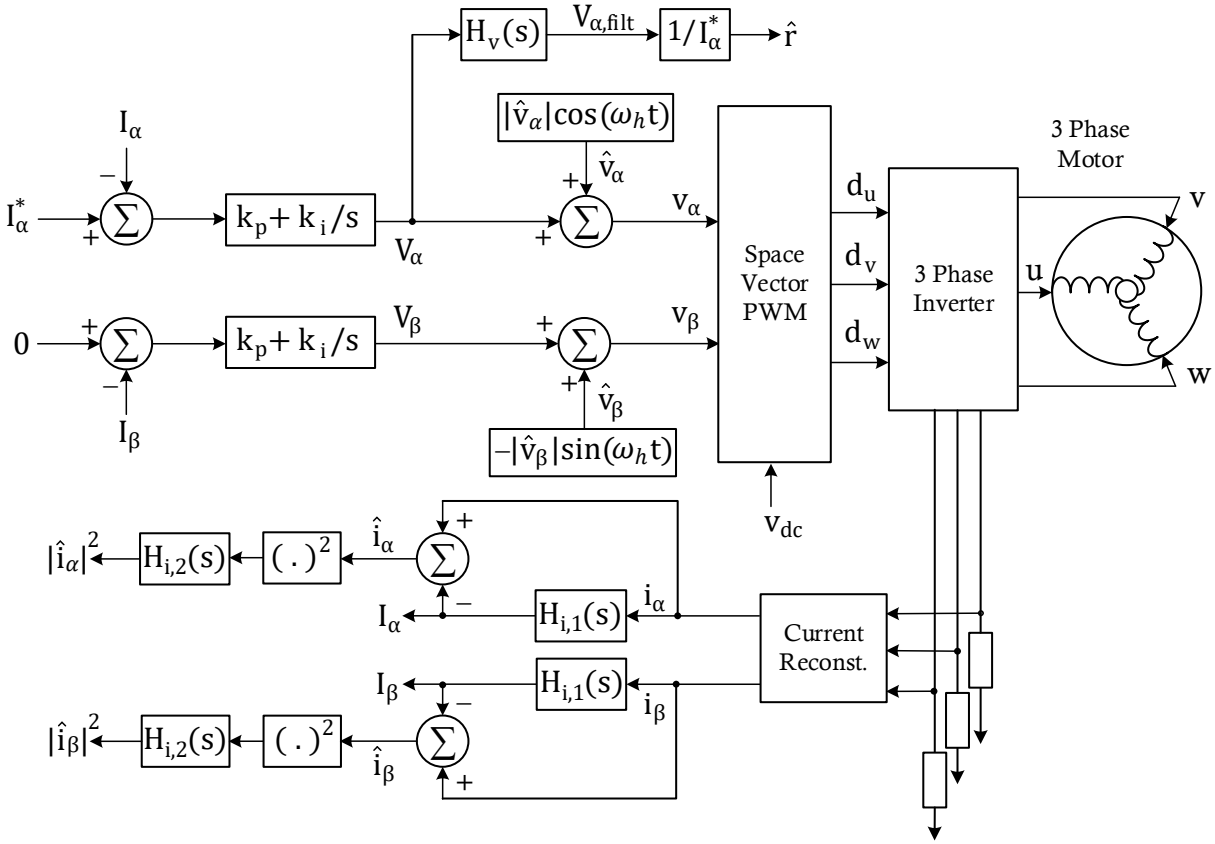

Fig. 133: Motor profiler block diagram

Without loss of generality, equations (358) and (359) can be written in general terms as

$$|\hat{\imath}|^2 = \frac{|\hat{v}|^2}{r^2 + (L\omega)^2} \tag{360}$$

where { $|\hat{\imath}|$, $|\hat{v}|$, and L } are either { $|\hat{\imath}_\alpha|$, $|\hat{v}_\alpha|$, and L_d } or { $|\hat{\imath}_\beta|$, $|\hat{v}_\beta|$, and L_q } respectively, and ω is ω_h.

To increase the estimation accuracy, it is preferable to use multiple injection frequencies to estimate L {i.e. L_q or L_d}. At each frequency point, ω_k, the injected high-frequency voltage, $|\hat{v}|_k$, will result in a measured high-frequency current, $|\hat{\imath}|_k$. Then, the best estimation of L at that particular frequency can be obtained as follows:

$$|\hat{z}|_k^2 = \frac{|\hat{v}|_k^2}{|\hat{\imath}|_k^2} = r^2 + \omega_k^2 L^2 \Rightarrow \omega_k^2 L^2 = |\hat{z}|_k^2 - r^2 \tag{361}$$

$$k = 1, \dots, n$$

Therefore, by collecting all of the measurement results in a vector format, we would have

$$\underbrace{\begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_n^2 \end{bmatrix}}_{\boldsymbol{A}_{n\times 1}} \underbrace{L^2}_{\boldsymbol{x}_{1\times 1}} = \underbrace{\begin{bmatrix} |\hat{z}|_1^2 - r^2 \\ \vdots \\ |\hat{z}|_n^2 - r^2 \end{bmatrix}}_{\boldsymbol{b}_{n\times 1}} \tag{362}$$

$$\boldsymbol{A}_{n\times 1}\boldsymbol{x}_{1\times 1} = \boldsymbol{b}_{n\times 1} \tag{363}$$

As can be seen in (362) and (363), there are n equations and one variable to solve for i.e. L^2. Therefore, the best approximate solution can be found using the "least squares method" by multiplying both sides of (362) and (363) with $\boldsymbol{A}_{1\times n}^T$:

$$\boldsymbol{A}_{1\times n}^T\boldsymbol{A}_{n\times 1}\boldsymbol{x}_{1\times 1} = \boldsymbol{A}_{1\times n}^T\boldsymbol{b}_{n\times 1} \tag{364}$$

In other words,

$$[\omega_1^2 \quad \cdots \quad \omega_n^2]\begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_n^2 \end{bmatrix} L^2 = [\omega_1^2 \quad \cdots \quad \omega_n^2]\begin{bmatrix} |\hat{z}|_1^2 - r^2 \\ \vdots \\ |\hat{z}|_n^2 - r^2 \end{bmatrix}. \tag{365}$$

Every time, a new $|\hat{v}|_n$ is applied and consequently, a new $|\hat{\imath}|_n$ is measured, we can compute the resulting $|\hat{z}|_n$ based on (361) and update the L estimation (L_n). The following equation summarizes this process based on (365):

$$L_n^2 = \frac{a_n}{b_n} \tag{366}$$

where

$$\begin{cases} a_n = \sum_{k=1}^{n} \left(|\hat{z}|_k^2 - r^2\right)\omega_k^2 \\ b_n = \sum_{k=1}^{n} \omega_k^4 \end{cases} \tag{367}$$

An iterative method can be developed here to update the best approximation of L at every iteration, L_n, while trying to control the current magnitude, $|\hat{\imath}|_n$, at a constant level, $|\hat{\imath}|^*$.

The reason for trying to control the current magnitude is because if the same voltage, $|\hat{v}|_n$, is applied at each iteration step, the current magnitude, $|\hat{\imath}|_n$, will drop significantly as ω_n increases. This will cause accuracy and precision issues when measuring the current magnitude because

1) Shunt resistors are usually sized for the peak current rating of the motor, i.e. they result in the full-scale ADC voltage at the peak current of the motor plus some margin to allow for over current detection.

2) For measuring L_q and L_d, the injected dc and ac currents must be much lower than the peak current rating of the motor otherwise the measurements would not be accurate due to saturation of the inductances along both axes (L_q and L_d). It is recommended that the injected currents don't go above 5%~10% of the peak current rating of the motor. This means that the ADC voltages would be already at a level below 5%~10% of the full-scale ADC voltage.

3) The ADCs of the digital controller have limited precision even at full-scale voltage (usually 12 bits per full-scale voltage covering both positive and negative current values). Thus, at 5% of full-scale voltage, we would have $0.05\times(2^{12}/2\times 0.7) = 71$ ticks. Here the division by 2 is because of positive/negative current coverage and 0.7 represents the margin for over-current detection. Thus, we can see that the ADCs' precision and signal-to-noise-ratio is very limited in such conditions at 5% of peak current rating of the motor.

4) If the applied voltage $|\hat{v}|_n$ results in 5% of peak current rating of the motor at low frequencies where the effect of $\omega_k^2 L^2$ on $|\hat{z}|_k^2$ is minimal, then the resulting current at higher frequencies where the

effect of $\omega_k^2 L^2$ on $|\hat{z}|_k^2$ becomes significant enough for us to be able to estimate L is too small and well below the current measurement capabilities of the ADCs. For example, at one decade above the corner frequency of the motor (r/L), the ADC reading would be ~0.1×71 ticks = 7 ticks, which will obviously result in inadequate precision and very bad signal to noise ratio.

Because of the reasons mentioned above, it is desirable to control the current magnitude (e.g. at 5% of the peak current rating of the motor) at each iteration step, while updating the inductance estimation.

We can achieve these goals by using the following iterative method which results in convergence of both L_n to the real inductance value and current magnitude $|\hat{\imath}|_n$ to the commanded current magnitude set by the user $|\hat{\imath}|^*$:

$$\begin{cases} n^{\text{th}} \text{ iteration} \\ n = 2, 3, \dots \end{cases} : \begin{cases} \text{Step 1: } |\hat{z}|_{n-1}^2 = \dfrac{|\hat{v}|_{n-1}^2}{|\hat{\imath}|_{n-1}^2} \\ \text{Step 2: } a_{n-1} = a_{n-2} + (|\hat{z}|_{n-1}^2 - r^2)\omega_{n-1}^2 \\ \text{Step 3: } b_{n-1} = b_{n-2} + \omega_{n-1}^4 \\ \text{Step 4: } L_{n-1}^2 = \dfrac{a_{n-1}}{b_{n-1}} \\ \text{Step 5: } |\hat{v}|_n^2 = (|\hat{\imath}|^*)^2 (r^2 + \omega_n^2 L_{n-1}^2) \end{cases} \tag{368}$$

The starting point of this iteration sequence is as follows:

$$\begin{cases} 1^{\text{st}} \text{ iteration} \\ n = 1 \end{cases} : \begin{cases} \text{Step 1: } |\hat{z}|_0^2 = 0 \\ \text{Step 2: } a_0 = 0 \\ \text{Step 3: } b_0 = 0 \\ \text{Step 4: } L_0^2 = 0 \\ \text{Step 5: } |\hat{v}|_1^2 = (|\hat{\imath}|^*)^2 (r^2) \end{cases} \tag{369}$$

According to (368), at each iteration step, we first calculate the resulting impedance of the previous step $|\hat{z}|_{n-1}$ from the applied voltage $|\hat{v}|_{n-1}$ and the measured current $|\hat{\imath}|_{n-1}$ of the previous step. Then, we update the numerator a_{n-1} and denominator b_{n-1} of the inductance estimation ratio and estimate the inductance value result of the previous step L_{n-1}. Then, we calculate the required voltage for the current step $|\hat{v}|_n$ which would result in the current command that we want $|\hat{\imath}|^*$ in the current step.

We can prove that the iterative sequence described in (368) and (369) will result in convergence of both estimated inductance L_n to the actual inductance L, and resulting currents $|\hat{\imath}|_n$ to the current command $|\hat{\imath}|^*$ at the same time. Based on (368) and (361):

$$\begin{cases} |\hat{\imath}|_n^2 = \dfrac{|\hat{v}|_n^2}{r^2 + \omega_n^2 L^2} \\ |\hat{v}|_n^2 = (|\hat{\imath}|^*)^2 (r^2 + \omega_n^2 L_{n-1}^2) \end{cases} \tag{370}$$

Thus,

$$|\hat{\imath}|_n^2 = (|\hat{\imath}|^*)^2 \frac{r^2 + \omega_n^2 L_{n-1}^2}{r^2 + \omega_n^2 L^2} \tag{371}$$

As indicated by (371), when L_n converges to L, $|\hat{\imath}|_n$ will converge to $|\hat{\imath}|^*$ and vice versa:

$$\{L_n^2 = L_{n-1}^2 = \cdots = L^2\} \Leftrightarrow \{|\hat{\imath}|_n^2 = |\hat{\imath}|_{n-1}^2 = \cdots = (|\hat{\imath}|^*)^2\} \tag{372}$$

The PI controller parameters (k_p and k_i) in Fig. 133 must be chosen by considering the overall loop-gain of the current controllers, which will also include $H_{i,1}(s)$ in their paths. Without loss of generality, $H_{i,1}(s)$ can be any low pass filter that provides enough attenuation for the high-frequency current components to prevent them from entering the dc-current loops, e.g.

$$H_{i,1}(s) = \left(\frac{\omega_{\text{sep}}}{s + \omega_{\text{sep}}} \right) \tag{373}$$

where ω_{sep} is the separation frequency. Similarly, $H_v(s)$ and $H_{i,2}(s)$ can be any low pass filter with significant attenuation of high-frequency voltage or current components. It is recommended that $H_v(s)$ and $H_{i,2}(s)$ are m-order filters ($m \geq 2$) to provide $-(20m)\text{dB/dec}$ attenuation at high frequencies, e.g.

$$H_{i,2}(s) = 2\left(\frac{\omega_{\text{sep}}}{s+\omega_{\text{sep}}}\right)^m \quad (374)$$

$$H_v(s) = \left(\frac{\omega_{\text{sep}}}{s+\omega_{\text{sep}}}\right)^m \quad (375)$$

Note that there is a multiplication by 2 in $H_{i,2}(s)$ to account for the demodulation gain (0.5) resulting from $(.)^2$ blocks in Fig. 133.

6.18.1.5 Rotor Permanent-Magnet Flux Linkage Estimation Stage

Depending on whether Rotor-Frame-Orientation (RFO) or Stator-Frame-Orientation (SFO) is used for Field Oriented Control (FOC), we can estimate the rotor flux linkage, λ_m, using the following equations

$$\begin{cases}\text{RFO: } \hat{\lambda}_m = \lambda_d^{r\prime} - (L_d - L_q) i_d^r \\ \text{SFO: } \hat{\lambda}_m = \lambda_d^s \cos(\delta) + L_d\left(i_q^s \sin(\delta) - i_d^s \cos(\delta)\right)\end{cases} \quad (376)$$

where $\lambda_d^{r\prime}$ is the magnitude component of the observer's Phase-Lock-Loop (PLL) in RFO, λ_d^s is the magnitude component of the observer's PLL in SFO, and δ is estimated load angle in SFO.

The estimated rotor flux linkage, $\hat{\lambda}_m$, is then filtered by an m-order low pass filter such as

$$H(s) = \left(\frac{\omega_{0,\lambda_m}}{s+\omega_{0,\lambda_m}}\right)^m \quad (377)$$

6.18.1.6 Typical Simulation Waveforms

Fig. 134 depicts the rotor locking stage for a typical IPM with an initial rotor angle of $\theta = \pi/4$. As can be seen in Fig. 134, by regulating and applying a constant dc current along the α-axis, the rotor flux linkage will align itself with the α-axis and after settling, the rotor angle would be $\theta = \pi/2$.

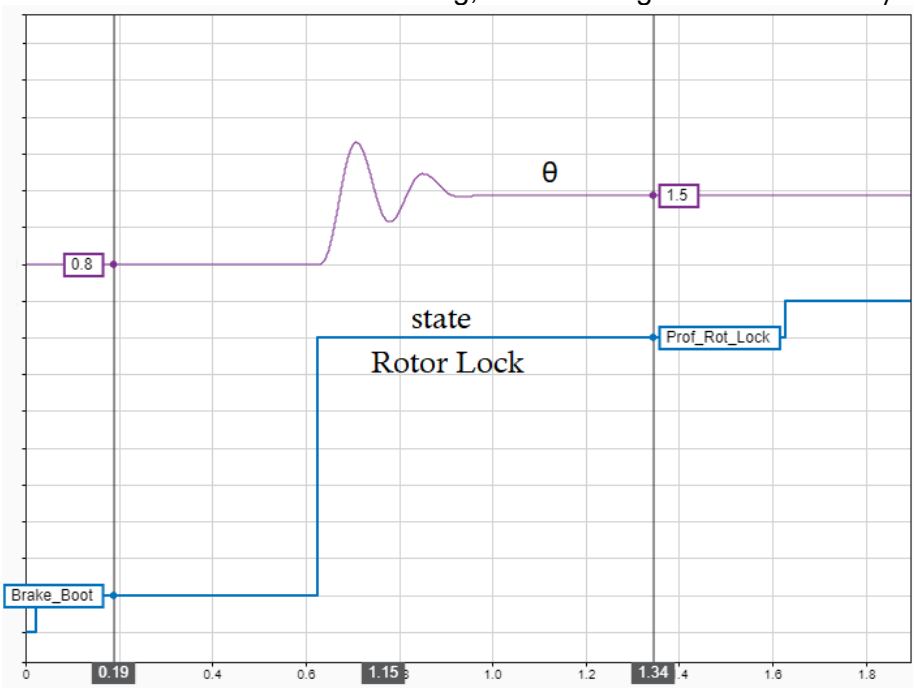

Fig. 134: Rotor locking stage, starting from $\theta = \pi/4$ as the initial rotor angle and locking at $\theta = \pi/2$

Fig. 135 illustrates the resistance-estimation stage where $H_v(s)$ is applied on V_α in order to filter out the voltage noise and compute the estimated resistance based on (353) (see the block diagram in Fig. 133)

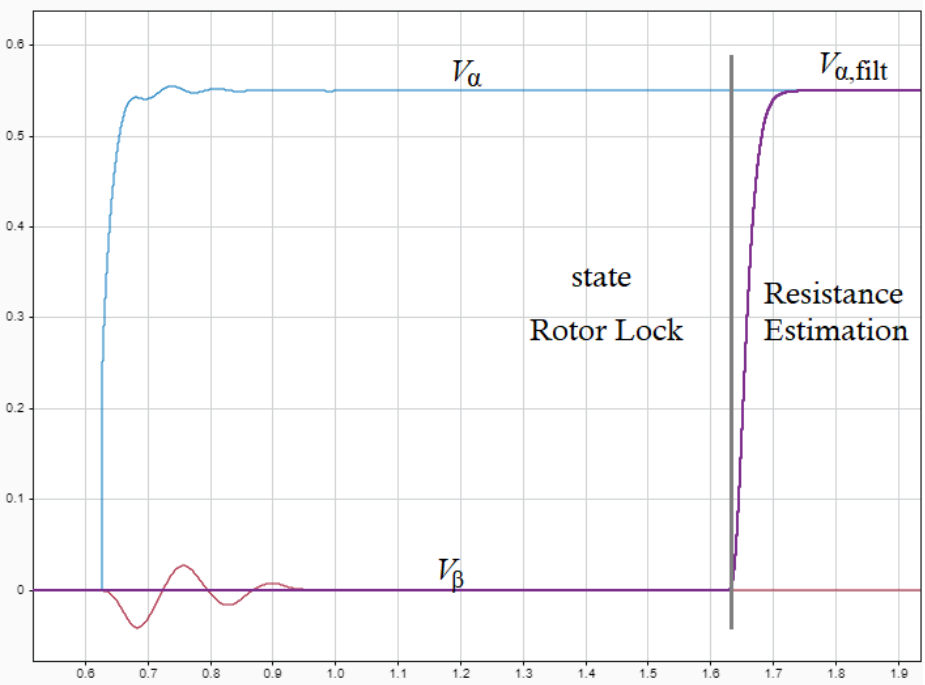

Fig. 135: Stator resistance estimation stage

Fig. 136 depicts the iterative current control and inductance estimation for L_q and L_d. As can be seen in Fig. 136, L_q and L_d estimations will converge to their corresponding real values while the current magnitudes converge to the commanded current ($|\hat{\imath}|^*$).

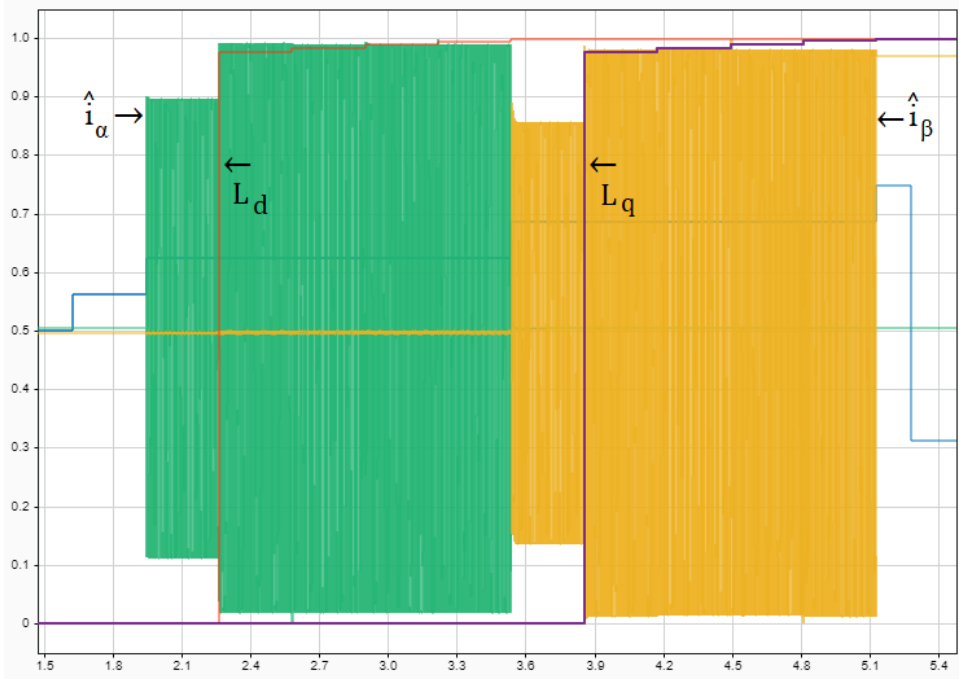

Fig. 136: Iterative current control and inductance estimation in inductance-measurement stage

Fig. 137 illustrates the calculated and applied high-frequency voltages ($|\hat{v}|_n$) in order to achieve constant current magnitudes at $|\hat{\imath}|^*$ while simultaneously estimating the inductance L_q and L_d.

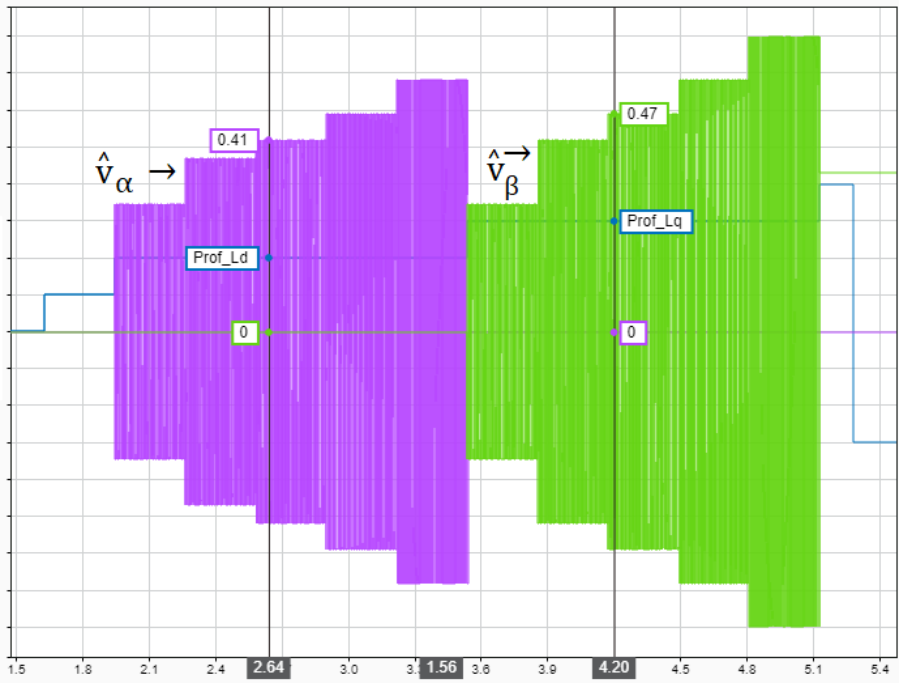

Fig. 137: Resulting voltage commands in inductance-measurement stage

The following figure depicts the flux-linkage estimation stage based on (376). A high-order critically-damped low pass filter is also applied on the estimated ($\hat{\lambda}_m$) to increase the signal-to-noise ratio.

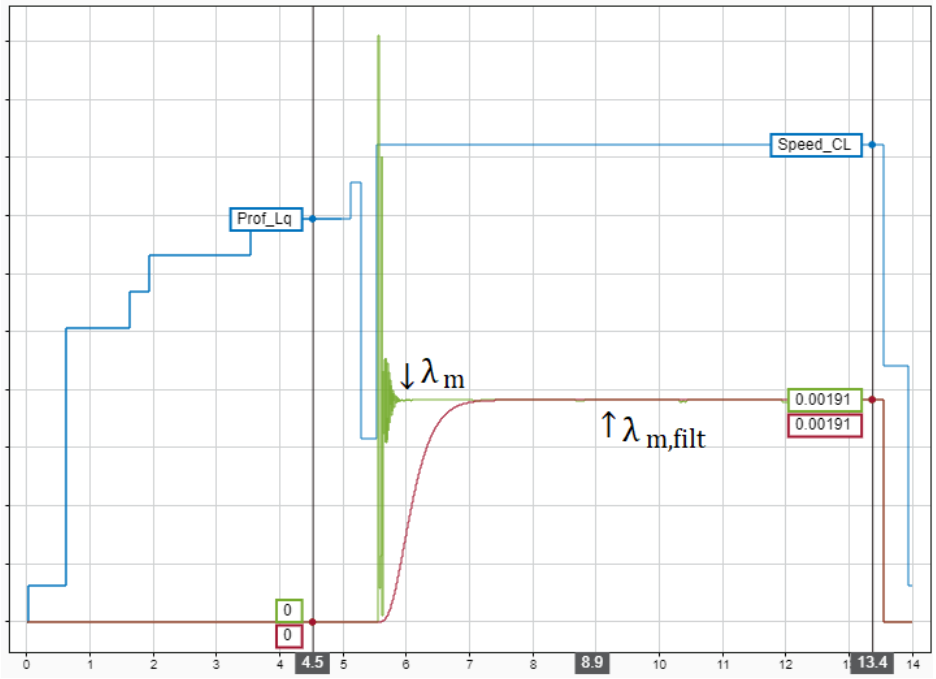

Fig. 138: Flux-linkage estimation stage (in closed-loop speed control mode)

6.18.1.7 Typical Bench Test Waveforms

Fig. 139 illustrates the resistance-estimation stage where $H_v(s)$ is applied on V_α in order to filter out the voltage noise and compute the estimated resistance based on (353) (see the block diagram in Fig. 133)

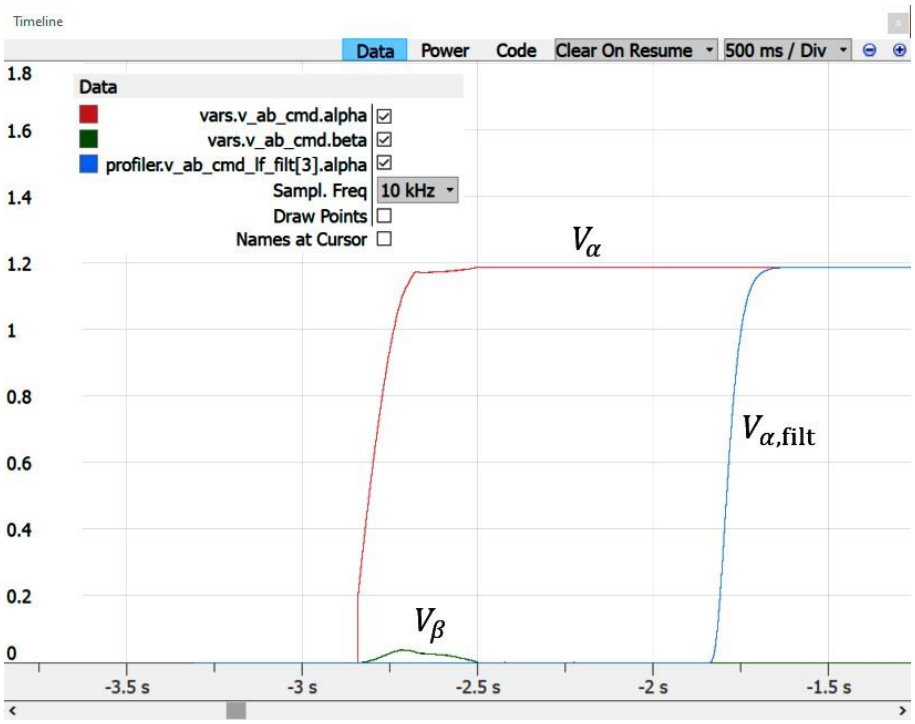

Fig. 139: Stator resistance estimation stage

Fig. 140 depicts the iterative current control during inductance estimation stage. As can be seen in Fig. 140, the high-frequency current magnitudes converge to the commanded current value ($|\hat{\imath}|^*$).

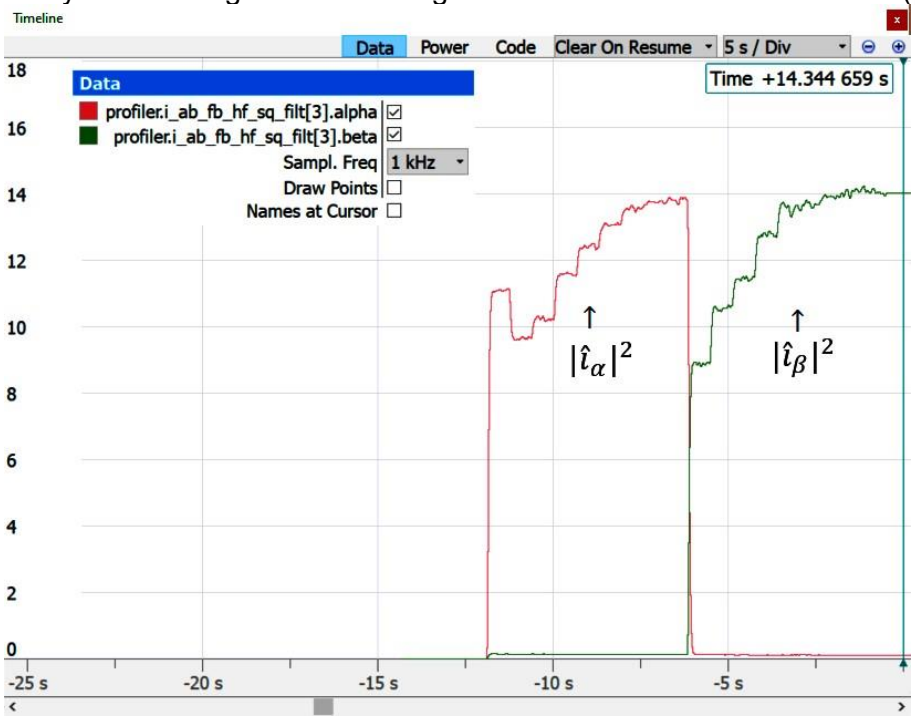

Fig. 140: Iterative current control during inductance-measurement stage

Fig. 141 depicts the iterative inductance estimation for both q- and d-axis inductances (L_q and L_d). As can be seen in Fig. 141, L_q and L_d estimations will converge to their corresponding real values.

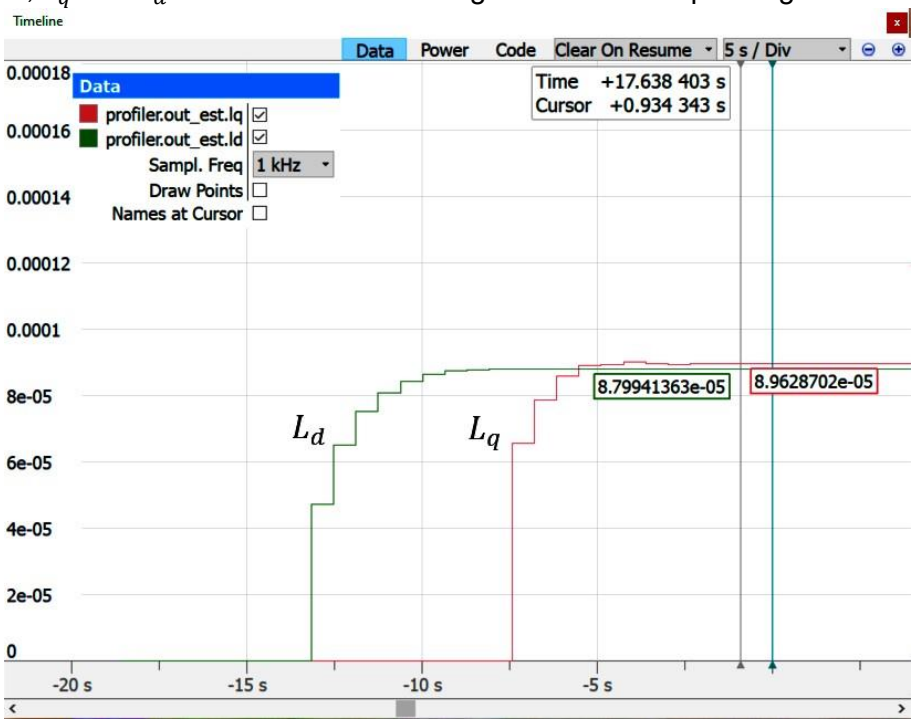

Fig. 141: Iterative inductance estimation during inductance-measurement stage

The following figure depicts the flux-linkage estimation stage based on (376). A high-order critically-damped low pass filter is also applied on the estimated ($\hat{\lambda}_m$) to increase the signal-to-noise ratio.

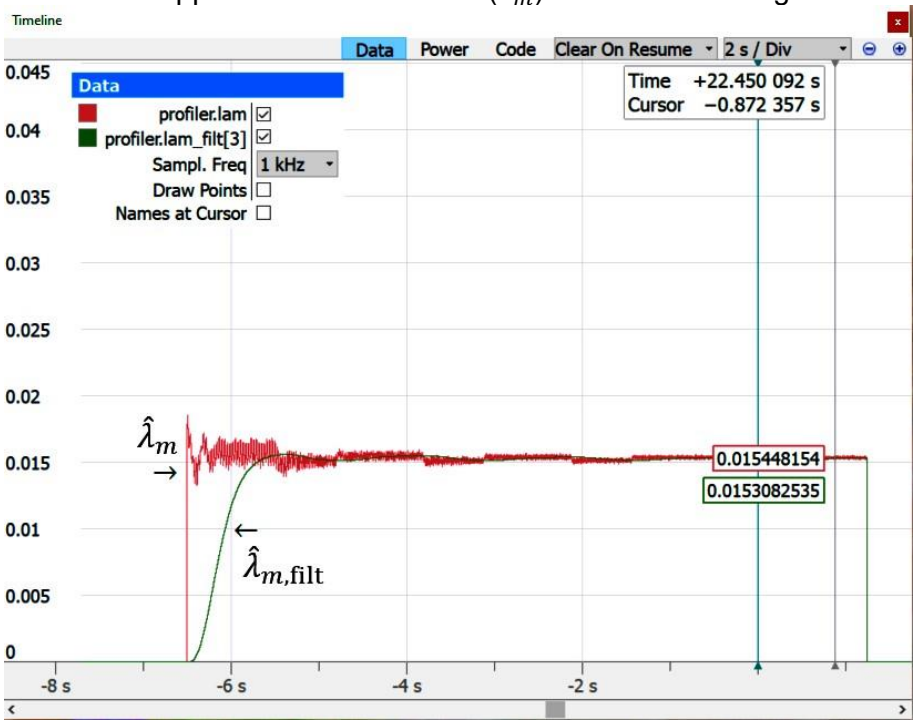

Fig. 142: Flux-linkage estimation stage (in closed-loop speed control mode)

6.18.2 Mechanical Profiler

The purpose of the mechanical profiler is to extract the parameters of the mechanical load, i.e. inertia J, viscous friction coefficient B, and static friction T_f. The inherent assumption here is that the mechanical system is a first order damped system in the low frequency range where the speed controller operates.

This assumption may not hold for motor/load systems with high inertia mismatch that show resonances in the low frequency range.

The first order mechanical system can be described as

$$T = T_{\mathrm{f}} \cdot \mathrm{sgn}(\omega_m) + B \cdot \omega_m + J \cdot \underbrace{\frac{d\omega_m}{dt}}_{0} \tag{378}$$

In order to extract viscous friction coefficient B and static friction T_{f}, the motor is run at different speed levels until it reaches steady-state, i.e. $d\omega_m/dt = 0$, and the estimate for torque T is recorded at each speed level. After collecting the data, B and T_{f} can be estimated using linear regression and least squares method, i.e.

$$y = mx + b \tag{379}$$

where $y = T$ and $x = \omega_m$ are the output and input data points, respectively. The parameters $m = B$ and $b = T_{\mathrm{f}}$ would be the viscous friction coefficient and the static friction that we want to extract here.

The linear regression algorithm runs at every ISR cycle to update $\sum x$, $\sum y$, $\sum xy$, $\sum x^2$, and $\sum y^2$ in order to calculate the variances, S_{xx}, S_{yy}, and covariance, S_{xy}, of the data later on:

$$\begin{cases} S_{xx} = \dfrac{N(\sum x^2) - (\sum x)^2}{N^2} \\ S_{yy} = \dfrac{N(\sum y^2) - (\sum y)^2}{N^2} \\ S_{xy} = \dfrac{N(\sum xy) - (\sum x)(\sum y)}{N^2} \end{cases} \tag{380}$$

Then, once the data recording is finished, the parameters can be extracted by calculating and utilizing the variances and covariances of the data as follows

$$\begin{cases} m = \dfrac{S_{xy}}{S_{xx}} \\ r = \dfrac{S_{xy}}{\sqrt{S_{xx}S_{yy}}} \\ b = \overline{y} - m\overline{x} \end{cases} \tag{381}$$

For extracting the inertia J, the following equation is used while motor decelerates from the final target speed, ω_{m0}, in a controlled manner:

$$J = \frac{\int_0^t (T - T_{\mathrm{f}} + B \cdot \omega_m) d\tau}{\omega_m - \omega_{m0}} \tag{382}$$

As can be seen in (382), the parameters T_{f} and B are required for inertia estimation, which is why this estimation happens after extracting T_{f} and B.

Figure below shows the profiler window in GUI and parameter extraction stages during motor acceleration and deceleration cycle.

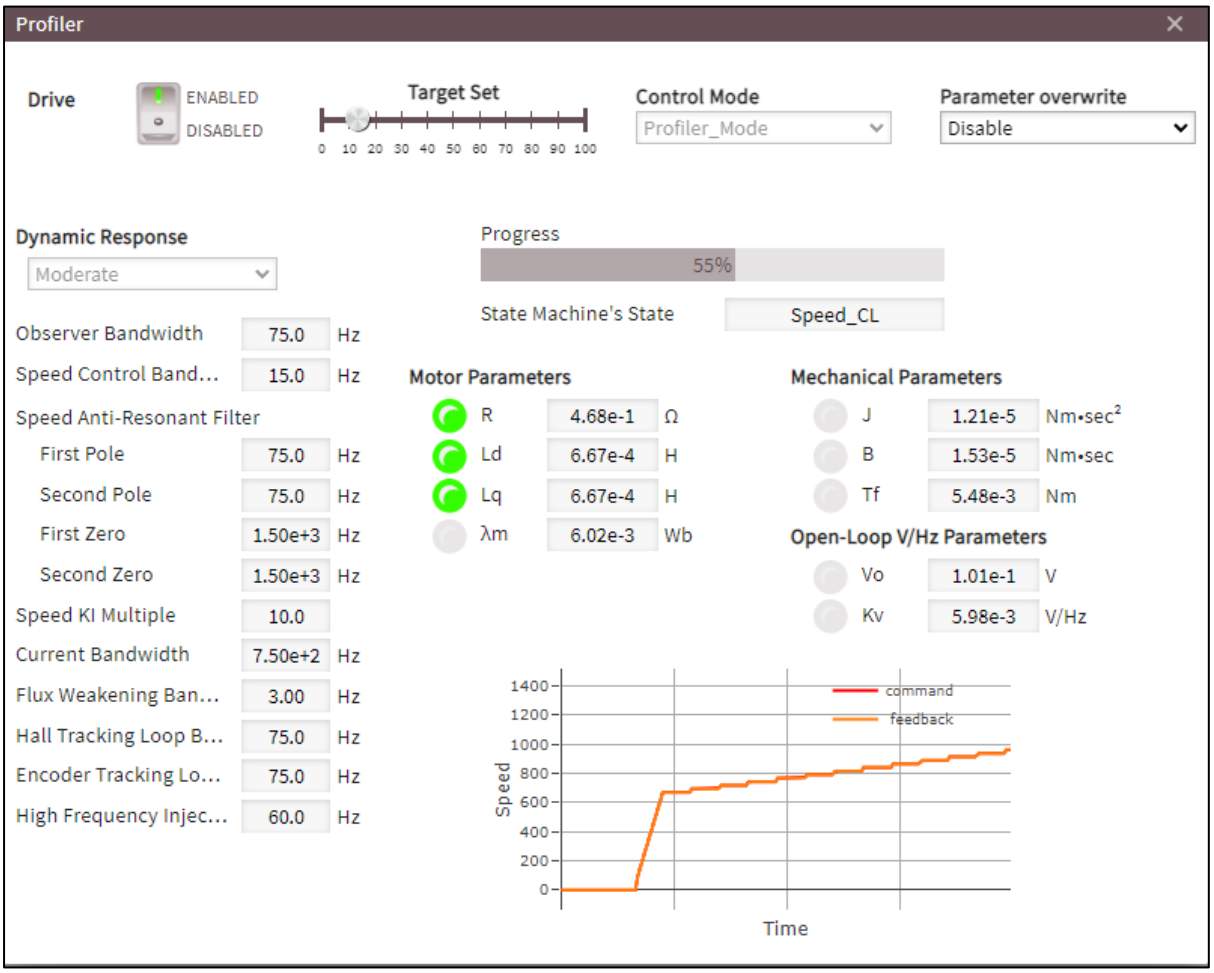

Fig. 143: GUI's profiler overview: motor parameters, mechanical parameters, and open-loop V/Hz (V/F) parameters

6.18.3 V/F Profiler

The goal of the V/F profiler is to extract the V/F parameters K and $V_{\min}$ as described in section 6.9. The same linear regression algorithm as described in previous section is used here with the slope and intercept being $m = K$ and $b = V_{\min}$:

$$v_s = m.\omega_e + b \tag{383}$$

The data is recorded together with the linear regression data for mechanical profiler while the motor is in the acceleration stage.

7 Quick Start Guide

In this section, the basic information that users need to know to start spinning a motor is presented. If the user is modifying the firmware directly via Modus Toolbox (not using GUI), then all sections in this chapter are applicable. If the user is working with GUI, then some of the information in this section about the source code may not be relevant. Nonetheless, the information here can help with understanding the important parameters and how to tune them.

7.1 Preparation and Preliminaries

Before starting, one should make sure:

- Modustoolbox v3.2 is installed.
- Most updated Segger J-Link software (free downloads from Link) is installed.
- Ozone J-Link debugger is installed for general debugging and monitoring of variables (Link).
- If not using GUI, make sure **PARAMS_ALWAYS_OVERWRITE** in ***param.h*** is changed from (false) to (true). Otherwise, any change in parameters in ***param.c*** will not take effect. This is because the parameters are stored and retrieved from the flash memory every time the micro boots up. This feature allows the user to update the firmware as new versions get released without overwriting and resetting the parameters that are already tuned for a specific application.

```
#define PARAMS_ALWAYS_OVERWRITE      (false)
```

- If using a second compiler like IAR, once the project is built using the second compiler, there may be an error that complains about the existence of duplicate output files. To resolve this, make sure to run the batch file /Auto/DeleteAllOutputFiles.bat.
- To compile code via IAR, copy the IAR workspace files in \IARWorkspaces\... to the root application directory.
- While there are different main control types available including RFO, SFO, and TBC, it is recommended to start with RFO to spin and control the motor for the first time and get familiar with the firmware. Later, users can refer to the main document to learn more about SFO and TBC and apply them. Therefore, here in this chapter, the assumption is that the code is being built using RFO configuration.

7.2 Basic and Advanced Parameters

There are some basic parameters that if understood well and set properly user can almost spin and control any PMSM or BLDC motor. These basic parameters are explained in the following sections. There are also a set of parameters called advanced parameters that are function of basic parameters and are auto calculated in ***PARAMS_InitAutoCalc()***. Basic parameters are assigned values in ***PARAMS_InitManual()*** located in ***param.c***. The basic parameters are divided to multiple categories: Such as Motor parameters (***params.motor***), System parameter (***params.sys***), observer parameters (***params.obs***), motor mechanical parameters (***params.mech***), filter parameters (***params.filt***) and control parameters (***params.ctrl***).

7.2.1 Motor Parameters

A snapshot of the motor parameter from ***param.c*** is shown below:

- Note that ***motor.P*** represents the number of poles, not pole pairs.
- The current values are based on the peak values, not rms.
- ***motor.i_cont*** is the continuous current rating of the motor.
- ***motor.i_peak*** is the peak current rating of the motor. Set this value to 2.5 to 3x of ***motor.i_cont***, if not specified in the motor datasheet.

- ***motor.id_max*** is the maximum d-axis current rating of the motor that doesn't result in demagnetization of the permanent magnet. Set ***motor.id_max*** to half of ***motor.i_cont***, if not specified in the motor datasheet.
- ***motor.v_nom*** is the motor nominal voltage. e.g. for a 24V-dc motor, set it to 24V-dc.

```
// Motor Parameters (42BL61 Motor):
params.motor.P = 8.0f; // [#]
params.motor.lq = 600.0E-6f; // [H]
params.motor.ld = 600.0E-6f; // [H]
params.motor.lam = 6.0E-3f; // [Wb]
params.motor.r = 400.0E-3f; // [Ohm]
params.motor.T_max = 0.390f; // [Nm]
params.motor.i_peak = 10.80f; // [A]
params.motor.i_cont = 3.50f; // [A]
params.motor.id_max = 1.75f; // [A]
params.motor.v_nom = LINE_TO_PHASE(24.0f); // [Vpk-ln]
params.motor.w_nom.elec = MECH_TO_ELEC(HZ_TO_RADSEC(RPM_TO_HZ(4000.0f)), params.motor.P)
params.motor.w_max.elec = MECH_TO_ELEC(HZ_TO_RADSEC(RPM_TO_HZ(6000.0f)), params.motor.P)
```

7.2.2 System Parameters

- Set the shunt type to three shunt or single shunt based on your inverter setup.
- Set the current sensing opamp gain based on your inverter setup.
- ***hyb_mod.adc_t_min*** and ***hyb_mod.ki*** belong to single shunt-based control. Refer to section 3.6 to understand and tune these parameters.
- ***shunt.res*** is a key parameter and represents shunt value for both single shunt and three shunt control. The original value is set in ***MotorCtrlHWConfig.h*** file since this parameter is board-specific and should be independent of the hardware-agnostic library.

```
params.sys.analog.shunt.type = Three_Shunt;
params.sys.analog.shunt.opamp_gain = Gain_12;
params.sys.analog.shunt.hyb_mod.adc_t_min = 3.0E-6f; // [sec]
params.sys.analog.shunt.hyb_mod.ki = 0.5f; // [#]
params.sys.analog.shunt.res = ADC_CS_SHUNT_RES; // [Ohm]
```

- ***cmd.w_max.mech*** is the maximum speed the user wants to command via potentiometer. For example, ***motor.w_max.elec*** may have been set to 6000 rpm but the user wishes to set ***cmd.w_max.mech*** to 3000 rpm. Then if the user turns the potentiometer all the way up motor will spin at max speed of 3000 rpm rather than 6000 rpm.

```
 params.sys.cmd.w_max.mech = HZ_TO_RADSEC(RPM_TO_HZ(4000.0f)); //[Ra/sec-elec]
 defined(CTRL_METHOD_RFO) || defined(CTRL_METHOD_TBC)
 params.sys.cmd.i_max = 3.50f * 0.5f; // [A]
if defined(CTRL_METHOD_SFO)
 params.sys.cmd.T_max = 0.390f * 0.5f; // [Nm]
```

- ***vdc_nom*** is the power supply voltage. Ideally, this number is the same as ***motor.v_nom***. However, just as an example, the user may want to apply 24V to a 48V motor. So, ***vdc_nom*** needs to be set as 24V rather than 48V in this case.

```
params.sys.vdc_nom = 24.0f; // [V]
```

7.2.3 Observer Parameters

Setting the observer parameters are the key for sensorless FOC operation:

- ***w_thresh.elec*** is the speed at which the transition from open loop to close loop FOC happens. This is a key parameter to tune if the transition to closed loop doesn't happen smoothly.
- ***lock_time*** is the time expected for the observer to lock to the rotor angle, to be tunned as needed.

```
params.obs.w_thresh.elec = params.motor.w_nom.elec * 0.20f; // [Ra/sec-elec]
params.obs.w_hyst.elec = params.obs.w_thresh.elec * 0.25f; // [Ra/sec-elec]
params.obs.lock_time = 0.200f; // [sec], increase when transition to closed-
```

7.2.4 Mechanical Parameters

The mechanical parameters may not be found on motor datasheet. They depend mostly on the mechanical load attached to the motor's shaft. It is best to derive and set them properly. These parameters were explained in 3.1 and depicted in Fig. 4. In the next section, it will be explained how to derive them.

```
// Mechanical System Parameters:
params.mech.inertia = 11.0E-6f; // [kg.m^2]=[(N.m)/(Ra/sec-mech).sec], =1/2*m*r^2
params.mech.viscous = 1.2E-5f; // [kg.m^2/sec]=[(N.m)/(Ra/sec-mech)]
params.mech.friction = 6.1E-3f; // [kg.m^2/sec^2]=[N.m]
```

7.2.5 Control Parameters

Control parameters are divided into some categories as explained below:

7.2.5.1 Control Mode

Multiple control modes are available for user to select in the firmware. The modes are listed in Table 4. Users can set the control mode in ***param.c***:

```
// Control Mode:
defined(CTRL_METHOD_RFO) || defined(CTRL_METHOD_SFO)
params.ctrl.mode = Speed_Mode_FOC_Sensorless_Volt_Startup; // [#]
if defined(CTRL_METHOD_TBC)
params.ctrl.mode = Speed_Mode_Block_Comm_Hall; // [#]
```

from one of the multiple options available in ***param.h***:

```
    Volt_Mode_Open_Loop = 0,                    // Open-lo
#if defined(CTRL_METHOD_RFO)
    Curr_Mode_FOC_Sensorless_Align_Startup,     // Closed-
    Curr_Mode_FOC_Sensorless_SixPulse_Startup,  // Closed-
    Curr_Mode_FOC_Sensorless_HighFreq_Startup,  // Closed-
    Curr_Mode_FOC_Sensorless_Dyno,              // Closed-
    Curr_Mode_FOC_Hall,                         // Closed-
#elif defined(CTRL_METHOD_TBC)
    Curr_Mode_Block_Comm_Hall,                  // Closed-
#elif defined(CTRL_METHOD_SFO)
    Trq_Mode_FOC_Sensorless_Align_Startup,      // Closed-
    Trq_Mode_FOC_Sensorless_SixPulse_Startup,   // Closed-
    Trq_Mode_FOC_Sensorless_HighFreq_Startup,   // Closed-
    Trq_Mode_FOC_Sensorless_Dyno,               // Closed-
#endif
#if defined(CTRL_METHOD_RFO) || defined(CTRL_METHOD_SFO)
    Speed_Mode_FOC_Sensorless_Align_Startup,    // Closed-
    Speed_Mode_FOC_Sensorless_SixPulse_Startup, // Closed-
    Speed_Mode_FOC_Sensorless_HighFreq_Startup, // Closed-
    Speed_Mode_FOC_Sensorless_Volt_Startup,     // Closed-
#endif
#if defined(CTRL_METHOD_RFO)
    Speed_Mode_FOC_Hall,                        // Closed-
#elif defined(CTRL_METHOD_TBC)
    Speed_Mode_Block_Comm_Hall,                 // Closed-
#endif
#if defined(CTRL_METHOD_RFO) || defined(CTRL_METHOD_SFO)
    Motor_Profiler_Mode,                        // Motor p
#endif
```

7.2.5.2 Speed Controller

speed.bw is a key parameter user need to provide for the speed loop controller. k_p and k_i for the controller are automatically calculated from the given bandwidth and motor mechanical parameters in **PARAMS_InitAutoCalc()** function.

```
// Speed Control Parameters:
params.ctrl.speed.bw = HZ_TO_RADSEC(5.0f); // [Ra/sec]
params.ctrl.speed.ol_cl_tr_coeff = 0.4f; // [%]
```

7.2.5.3 Current Controller

ctrl.curr.bw is a key parameter user needs to provide for the current loop controller. k_p and k_i for the controller are automatically calculated from motor electrical parameters including stator inductance and resistance in ***PARAMS_InitAutoCalc()*** function. Set the value of ***ctrl. curr.bw*** at least 3~5 times lower than switching frequency. In addition, it must be set to be higher than ***speed.bw.***

```
defined(CTRL_METHOD_RFO) || defined(CTRL_METHOD_TBC)
params.ctrl.curr.bw = HZ_TO_RADSEC(600.0f); // [Ra/sec]
```

7.2.5.4 Voltage Controller

Below parameters are key to start spinning the motor. ***w_thresh.elec*** is the minimum speed that will activate open loop voltage control. Any speed commanded below this threshold will not spin the motor. ***v_min*** is the minimum voltage need to apply when speed is still zero and right before motor starts

spinning in open loop. ***v_to_f ratio*** is the slope of the V/F curve that the motor is required to follow in an open loop. In section 7.3 it is shown how to determine these parameters for a given motor.

```
// Voltage Control Parameters:
params.ctrl.volt.w_thresh.elec = params.motor.w_nom.elec * 0.05f; // [
params.ctrl.volt.w_hyst.elec = params.ctrl.volt.w_thresh.elec * 0.5f;
params.ctrl.volt.v_min = 7.0E-2f * 2.0f; // [Vpk]
params.ctrl.volt.v_to_f_ratio = 5.8E-3f * 1.5f; // [Vpk/(Ra/sec-elec)]
```

7.3 Extracting Parameters

Some of the motor parameters are given by motor datasheet. It is recommended to derive some of the parameters directly for more optimized and smooth performance.

7.3.1 Extracting Voltage Control Parameters

To extract optimal values for ***v_min*** and ***v_to_f ratio***, first it is required to spin the motor in ***Speed_Mode_FOC_Sensorless_Volt_Startup*** control mode. Obviously, to spin the motor, initial values for the ***v_min*** and ***v_to_f ratio*** are required that don't need to be optimal. Determine these tentative starting values from the following equations:

$$\text{params.ctrl.volt.v\_min} = \frac{(\text{params.motor.r} * \text{params.motor.i\_cont})}{k} \qquad (384)$$

$$k: 3 \sim 10$$

$$\text{params.ctrl.volt.v\_to\_f\_ratio} = \frac{\text{params.motor.v\_nom}}{\text{params.motor.w\_nom.elec}} \qquad (385)$$

Once the motor is spinning in closed loop, use the Ozone debugger to find out corresponding commanded voltages for two arbitrary speeds:

x1=electrical speed at the first arbitrary point, y1=commanded voltage corresponding to x1

x2=electrical speed at the second arbitrary point, y2=commanded voltage corresponding to x2

Then derive the equation based on (x1, y1) and (x2, y2) that represents a linear graph as shown in Fig. 144. The intercept and the slope of the graph are respectively ***v_min*** and ***v_to_f*** ratio (K). These two derived values are the more optimized compared to what is obtained from (384) and (385). Update ***param.c*** with these new values which will help with the smoother function of the motor in the open loop startup.

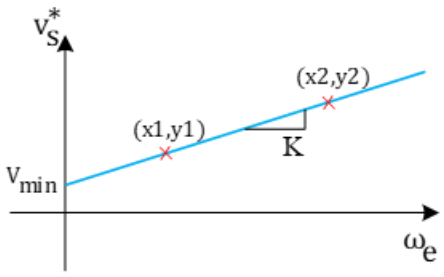

Fig. 144: Graph to extract V_{min} and v/f ratio.

7.3.2 Mechanical Parameters

To derive motor mechanical parameters, first run the motor at two arbitrary mechanical speeds of x1 and x2. Then for each case observe the corresponding filtered torque (**vars.T_est_filt** in **Trq.c**) from Ozone debugger and assign them to y1 and y2. Derive the equation based on (x1, y1) and (x2, y2) that represents a linear graph as shown in Fig. 145 . The intercept of this graph is T_f and the slope is B (see sections 3.1 and 7.2.4).

To estimate J, first increase the speed with a constant rate (from potentiometer or GUI). Note the correspondent change in T_{est} (**vars.T_est_filt**) due to change in speed from the Ozone debugger as shown in Fig. 146. Then J is calculated as:

$$J = \frac{T_{est2} - T_{est1}}{\frac{d\omega_m}{dt}} \quad (386)$$

```
// Mechanical System Parameters:
params.mech.inertia = 11.0E-6f; // [kg.m^2]=[(N.m
params.mech.viscous = 1.2E-5f; // [kg.m^2/sec]=[(
params.mech.friction = 6.1E-3f; // [kg.m^2/sec^2]
```

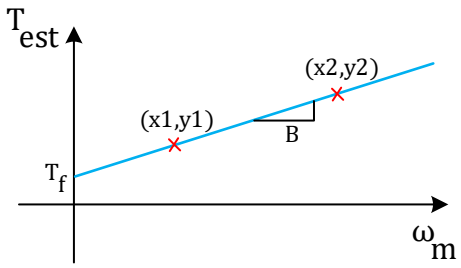

Fig. 145: Graph to extract T_f and B.

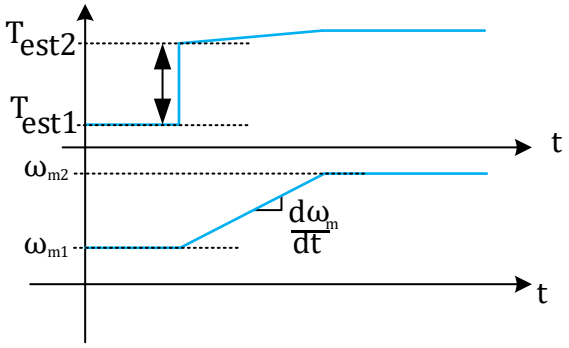

Fig. 146: Graph to extract J.

8 Parameters Dictionary

The following table presents the list of tunable parameters for various build configurations (i.e. RFO, SFO, TBC). The firmware symbol, as well as the parameter type, its SI unit, the corresponding section in this document, and the corresponding mathematical symbol in that section are listed. This table provides a one-to-one mapping between the materials provided in this document and their corresponding parameters in the firmware.

There are two parameter types, Basic and Advanced. Advanced parameters don't need to be entered by the user. They can be auto-calculated by the firmware by invoking the auto-calculate function through the debugger or the GUI. The user can also set the advanced parameters directly if fine tuning is required. Usually, auto-calculation results in close-to-optimal results for initial evaluation but the advanced parameters can be further tuned directly if desired.

In addition, not all basic parameters need to be touched for the motor control system to work. Only a few basic parameters usually need to be touched to get a motor-drive system up and running. Please see the quick start guide in section 7 for more information. The firmware is very flexible and versatile, and all of the parameters can be finely tuned but that doesn't mean that they must be changed for the motor-drive system to work.

Table 9: Parameters dictionary

Build	FW Symbol	Type	Unit	Section	Symbol
ALL	`params.motor.P`	B	-	7.2.1	P
ALL	`params.motor.lq`	B	H	7.2.1	L_q
ALL	`params.motor.ld`	B	H	7.2.1	L_d
ALL	`params.motor.lam`	B	Wb	7.2.1	λ_m
ALL	`params.motor.r`	B	Ohm	7.2.1	r
ALL	`params.motor.T_max`	B	Nm	7.2.1	$T_{\max}$
ALL	`params.motor.i_peak`	B	A	7.2.1	I_{peak}
ALL	`params.motor.i_cont`	B	A	7.2.1	I_{cont}
ALL	`params.motor.id_max`	B	A	7.2.1	$I_{\text{d,max}}$
ALL	`params.motor.v_nom`	B	V	7.2.1	V_{nom}
ALL	`params.motor.w_nom.elec`	B	Hz(elec.)	7.2.1	$\omega_{e,\text{nom}}$
ALL	`params.motor.w_max.elec`	B	Hz(elec.)	7.2.1	$\omega_{e,\max}$
SFO	`params.motor.mtpv_margin`	B	%	4.4.2	K_{MTPV}
ALL	`params.motor.zeta`	A	-	4.2.1, 4.2.2	ξ
ALL	`params.motor.i2t.therm_tau`	B	Sec	6.12	τ
ALL	`params.motor.i2t.on_level`	B	%	6.12	K_{on}
ALL	`params.motor.i2t.off_level`	B	%	6.12	K_{off}
SFO	`params.motor.mtpa_lut.x_min`	A	Nm	4.2.3	0
SFO	`params.motor.mtpa_lut.x_max`	A	Nm	4.2.3	$T_{\max}$
SFO	`params.motor.mtpa_lut.x_step`	A	Nm	4.2.3	ΔT
SFO	`params.motor.mtpa_lut.x_step_inv`	A	1/Nm	4.2.3	$1/\Delta T$
SFO	`params.motor.mtpa_lut.y`	A	Wb	4.2.3	$\lambda_d^s[n]$
SFO	`params.motor.mtpv_lut.x_min`	A	Wb	4.4.2	0
SFO	`params.motor.mtpv_lut.x_max`	A	Wb	4.2.3, 4.4.2	$\lambda_d^s[N-1]$
SFO	`params.motor.mtpv_lut.x_step`	A	Wb	4.4.2	$\Delta\lambda_d^s$
SFO	`params.motor.mtpv_lut.x_step_inv`	A	1/Wb	4.4.2	$1/\Delta\lambda_d^s$
SFO	`params.motor.mtpv_lut.y`	A	Nm	4.4.2	$T_{\text{pullout}}[n]$
ALL	`params.sys.vdc_nom`	B	V	6.13	V_{dc}
ALL	`params.sys.boot_time`	B	Sec	6.16	T_{boot}

SFO, RFO	params.sys.dyno_lock_time	B	Sec	6.11.7	$T_{\text{dyno,lock}}$
ALL	params.sys.samp.fpwm_fs0_ratio	B	-	6.17.1	$-\text{NA}$
ALL	params.sys.samp.fs0	B	Hz	6.17.1	f_{s0}
ALL	params.sys.samp.fs0_fs1_ratio	B	-	6.17.1	$-\text{NA}$
ALL	params.sys.samp.ts0	A	Sec	6.17.1	T_{s0}
ALL	params.sys.samp.fpwm	A	Hz	6.17.1	f_{pwm}
ALL	params.sys.samp.tpwm	A	Sec	6.17.1	T_{pwm}
ALL	params.sys.samp.fs1	A	Hz	6.17.1	f_{s1}
ALL	params.sys.samp.ts1	A	Sec	6.17.1	T_{s1}
ALL	params.sys.analog.offset_null_time	B	Sec	6.14	$T_{\text{offset,null}}$
ALL	params.sys.analog.calib.i_u.gain	B	%	6.14	a
ALL	params.sys.analog.calib.i_u.offset	B	A	6.14	b
ALL	params.sys.analog.calib.i_v.gain	B	%	6.14	a
ALL	params.sys.analog.calib.i_v.offset	B	A	6.14	b
ALL	params.sys.analog.calib.i_w.gain	B	%	6.14	a
ALL	params.sys.analog.calib.i_w.offset	B	A	6.14	b
ALL	params.sys.analog.calib.v_uz.gain	B	%	6.14	a
ALL	params.sys.analog.calib.v_uz.offset	B	V	6.14	b
ALL	params.sys.analog.calib.v_vz.gain	B	%	6.14	a
ALL	params.sys.analog.calib.v_vz.offset	B	V	6.14	b
ALL	params.sys.analog.calib.v_wz.gain	B	%	6.14	a
ALL	params.sys.analog.calib.v_wz.offset	B	V	6.14	b
ALL	params.sys.analog.calib.v_dc.gain	B	%	6.14	a
ALL	params.sys.analog.calib.v_dc.offset	B	V	6.14	b
ALL	params.sys.analog.calib.temp_ps.gain	B	%	6.14	a
ALL	params.sys.analog.calib.temp_ps.offset	B	C	6.14	b
ALL	params.sys.analog.calib.pot.gain	B	%	6.14	a
ALL	params.sys.analog.calib.pot.offset	B	%	6.14	b
ALL	params.sys.analog.filt.w0_i_ph	B	Hz	6.14	ω_0
ALL	params.sys.analog.filt.w0_v_ph	B	Hz	6.14	ω_0
ALL	params.sys.analog.filt.w0_v_dc	B	Sec	6.14	ω_0
ALL	params.sys.analog.filt.w0_temp_ps	B	Sec	6.14	ω_0
ALL	params.sys.analog.filt.w0_pot	B	Sec	6.14	ω_0
ALL	params.sys.analog.shunt.type	B	-	3.6, 4.10, 5.5	$-\text{NA}$
ALL	params.sys.analog.shunt.res	B	Ohm	3.6, 4.10, 5.5	R_{shunt}
ALL	params.sys.analog.shunt.opamp_gain	B	-	3.6, 4.10, 5.5	G_{opamp}
ALL	params.sys.analog.shunt.hyb_mod.adc_t_min	B	Sec	3.6.1.1	T_{ADC}
ALL	params.sys.analog.shunt.hyb_mod.ki	B	-	3.6.2.2	$A \cdot m \cdot T_{s0}$
ALL	params.sys.analog.shunt.hyb_mod.adc_d_min	A	%	3.6.1.1	D_{min}
ALL	params.sys.rate_lim.w_cmd.elec	B	Hz(elec.)/Sec	6.15	$\lvert d\omega^*/dt\rvert_{\text{lim}}$
RFO, TBC	params.sys.rate_lim.i_cmd	B	A/Sec	6.15	$\lvert dI^*/dt\rvert_{\text{lim}}$
SFO	params.sys.rate_lim.T_cmd	B	Nm/Sec	6.15	$\lvert dT^*/dt\rvert_{\text{lim}}$
ALL	params.sys.faults.oc_thresh	B	%	6.12, 6.13	K_{fault}
ALL	params.sys.faults.vdc_time	B	Sec	6.13	$T_{\text{dc,fault}}$
ALL	params.sys.faults.temp_ps_thresh	B	Celsius	6.13	$T_{\text{temp,thresh}}$
ALL	params.sys.faults.short_method	B	-	6.13	$-\text{NA}$
ALL	params.sys.faults.max_clr_tries	B	-	6.13	$N_{\text{fault,tries}}$
ALL	params.sys.faults.watchdog_time	B	mSec	6.13	$T_{\text{wd,fault}}$

ALL	params.sys.faults.vdc_thresh.min	A	V	6.13	$V_{\text{dc,min}}$
ALL	params.sys.faults.vdc_thresh.max	A	V	6.13	$V_{\text{dc,max}}$
ALL	params.sys.faults.w_thresh.elec	A	Hz(elec.)	6.13	$\omega_{\text{thresh,fault}}$
ALL	params.sys.faults.cmd_clr_thresh	A	%	6.13	$-\text{NA}$
ALL	params.sys.cmd.source	B		7.2.2	$-\text{NA}$
ALL	params.sys.cmd.w_max.mech	B	Hz(mech.)	7.2.2	ω^*_{max}
RFO, TBC	params.sys.cmd.i_max	B	A	7.2.2	I^*_{max}
SFO	params.sys.cmd.T_max	B	Nm	7.2.2	T^*_{max}
SFO, RFO	params.sys.fb.mode	A	-	6.5	$-\text{NA}$
TBC	params.sys.fb.mode	A	-	6.5	$-\text{NA}$
ALL	params.sys.fb.hall.track_loop.cpr	B	counts / rev (mech.)	6.4	$N_{\text{CPR}}.P/2$
ALL	params.sys.fb.hall.track_loop.w0_w	B	Hz	6.4	ω_0
ALL	params.sys.fb.hall.track_loop.w0_th.min	B	Hz	6.4	G_{min}
ALL	params.sys.fb.hall.track_loop.w0_th.max	B	Hz	6.4	G_{max}
ALL	params.sys.fb.hall.track_loop.tau_ratio	B	-	6.4	r_τ
ALL	params.sys.fb.hall.w_zsd_thresh.elec	B	Hz(elec.)	6.5.2	ω_{thresh}
ALL	params.sys.fb.hall.th_r_offset.elec	B	Deg	6.5.2	θ_{offset}
ALL	params.sys.fb.hall.block_comm_offset_comp	B	-	6.5.2	$-\text{NA}$
ALL	params.sys.fb.encoder.track_loop.cpr	B	counts / rev (mech.)	6.4	$N_{\text{CPR}}.P/2$
RFO	params.sys.fb.encoder.track_loop.w0_w	B	Hz	6.4	ω_0
RFO	params.sys.fb.encoder.track_loop.w0_th.min	B	Hz	6.4	G_{min}
RFO	params.sys.fb.encoder.track_loop.w0_th.max	B	Hz	6.4	G_{max}
RFO	params.sys.fb.encoder.track_loop.tau_ratio	B	-	6.4	r_τ
RFO	params.sys.fb.encoder.w_zsd_thresh.elec	B	Hz(elec.)	6.6	ω_{thresh}
RFO	params.obs.w_thresh.elec	B	Hz(elec.)	3.5, 6.16	$\omega^*_{\text{obs,thresh}}$
SFO	params.obs.w_thresh.elec	B	Hz(elec.)	4.8, 6.16	$\omega^*_{\text{obs,thresh}}$
RFO	params.obs.w_hyst.elec	B	Hz(elec.)	3.5, 6.16	$\omega^*_{\text{obs,hyst}}$
SFO	params.obs.w_hyst.elec	B	Hz(elec.)	4.8, 6.16	$\omega^*_{\text{obs,hyst}}$
RFO	params.obs.lock_time	B	Sec	6.16	$T_{\text{obs,lock}}$
SFO	params.obs.lock_time	B	Sec	6.16	$T_{\text{obs,lock}}$
RFO	params.obs.flux_filt.k1	B	-	3.5.1	k_1
SFO	params.obs.flux_filt.k1	B	-	4.8.1	k_1
RFO	params.obs.flux_filt.k2	B	-	3.5.1	k_2
SFO	params.obs.flux_filt.k2	B	-	4.8.1	k_2
RFO	params.obs.flux_filt.k3	B	-	3.5.1	k_3
SFO	params.obs.flux_filt.k3	B	-	4.8.1	k_3
RFO	params.obs.flux_filt.c1_coeff	A	Rad/Sec	3.5.1	c_1'
SFO	params.obs.flux_filt.c1_coeff	A	Rad/Sec	4.8.1	c_1'
RFO	params.obs.flux_filt.c2_coeff	A	(Rad/Sec)^2	3.5.1	c_2'
SFO	params.obs.flux_filt.c2_coeff	A	(Rad/Sec)^2	4.8.1	c_2'
RFO	params.obs.flux_filt.c3_coeff	A	(Rad/Sec)^3	3.5.1	c_3'
SFO	params.obs.flux_filt.c3_coeff	A	(Rad/Sec)^3	4.8.1	c_3'
RFO	params.obs.flux_filt.gain	A	-	3.5.1	$1/A$
SFO	params.obs.flux_filt.gain	A	-	4.8.1	$1/A$
RFO	params.obs.flux_filt.th_p.elec	A	Deg	3.5.1	θ_p
SFO	params.obs.flux_filt.th_p.elec	A	Deg	4.8.1	θ_p
RFO	params.obs.flux_filt.phase_lead.sine	A		3.5.1	$-\sin(\theta_p)$
SFO	params.obs.flux_filt.phase_lead.sine	A		4.8.1	$-\sin(\theta_p)$

RFO	params.obs.flux_filt.phase_lead.cosine	A		3.5.1	$\cos(\theta_p)$
SFO	params.obs.flux_filt.phase_lead.cosine	A		4.8.1	$\cos(\theta_p)$
RFO	params.obs.biquad_a[0]	B	-	3.5.2	a_0
SFO	params.obs.biquad_a[0]	B	-	4.8.2	a_0
RFO	params.obs.biquad_a[1]	B	1/(Rad/Sec)	3.5.2	a_1
SFO	params.obs.biquad_a[1]	B	1/(Rad/Sec)	4.8.2	a_1
RFO	params.obs.biquad_a[2]	B	1/((Rad/Sec)^2)	3.5.2	a_2
SFO	params.obs.biquad_a[2]	B	1/((Rad/Sec)^2)	4.8.2	a_2
RFO	params.obs.biquad_b[0]	B	-	3.5.2	b_0
SFO	params.obs.biquad_b[0]	B	-	4.8.2	b_0
RFO	params.obs.biquad_b[1]	B	1/(Rad/Sec)	3.5.2	b_1
SFO	params.obs.biquad_b[1]	B	1/(Rad/Sec)	4.8.2	b_1
RFO	params.obs.biquad_b[2]	B	1/((Rad/Sec)^2)	3.5.2	b_2
SFO	params.obs.biquad_b[2]	B	1/((Rad/Sec)^2)	4.8.2	b_2
RFO	params.obs.pll.w0	B	Hz	3.5	ω_0
SFO	params.obs.pll.w0	B	Hz	4.8	ω_0
RFO	params.obs.pll.w_max.elec	B	Hz(elec.)	3.5	ω_{max}
SFO	params.obs.pll.w_max.elec	B	Hz(elec.)	4.8	ω_{max}
RFO	params.obs.pll.kp	A	(Ra/sec-elec)/Wb	3.5	k_p
SFO	params.obs.pll.kp	A	(Ra/sec-elec)/Wb	4.8	k_p
RFO	params.obs.pll.ki	A	(Ra/sec).(Ra/sec-elec)/Wb	3.5	k_i
SFO	params.obs.pll.ki	A	(Ra/sec).(Ra/sec-elec)/Wb	4.8	k_i
RFO	params.obs.pll.th_offset.elec	A	Deg	3.5	θ'_p
SFO	params.obs.pll.th_offset.elec	A	Deg	4.8	θ'_p
ALL	params.mech.inertia	B	kg.m^2	3.1, 4.1, 5.1	J
ALL	params.mech.viscous	B	kg.m^2/sec	3.1, 4.1, 5.1	B
ALL	params.mech.friction	B	Nm	3.1, 4.1, 5.1	T_f
RFO, TBC	params.filt.acc_w0	B	Hz	3.1, 5.1, 6.2	ω_0
SFO	params.filt.acc_w0	B	Hz	4.1, 6.2	ω_0
ALL	params.filt.trq_w0	B	Hz	6	ω_0
RFO, TBC	params.filt.spd_ar_en	B	-	6.1	$-NA$
SFO	params.filt.spd_ar_en	B	-	6.1	$-NA$
RFO, TBC	params.filt.spd_ar_wp[0]	B	Hz	3.1, 5.1, 6.1	ω_{p1}
SFO	params.filt.spd_ar_wp[0]	B	Hz	4.1, 6.1	ω_{p1}
RFO, TBC	params.filt.spd_ar_wp[1]	B	Hz	3.1, 5.1, 6.1	ω_{p2}
SFO	params.filt.spd_ar_wp[1]	B	Hz	4.1, 6.1	ω_{p2}
RFO, TBC	params.filt.spd_ar_wz[0]	B	Hz	3.1, 5.1, 6.1	ω_{z1}
SFO	params.filt.spd_ar_wz[0]	B	Hz	4.1, 6.1	ω_{z1}
RFO, TBC	params.filt.spd_ar_wz[1]	B	Hz	3.1, 5.1, 6.1	ω_{z2}
SFO	params.filt.spd_ar_wz[1]	B	Hz	4.1, 6.1	ω_{z2}
RFO	params.ctrl.mode	B	-	0	$-NA$
SFO	params.ctrl.mode	B	-	0	$-NA$
TBC	params.ctrl.mode	B	-	0	$-NA$
RFO	params.ctrl.speed.bw	B	Hz	3.1	ω_{BW}
SFO	params.ctrl.speed.bw	B	Hz	4.1	ω_{BW}
TBC	params.ctrl.speed.bw	B	Hz	5.1	ω_{BW}
RFO	params.ctrl.speed.ki_multiple	B	%	3.1	$-NA$
SFO	params.ctrl.speed.ki_multiple	B	%	4.1	$-NA$

TBC	params.ctrl.speed.ki_multiple	B	%	5.1	$-NA$
RFO	params.ctrl.speed.ol_cl_tr_coeff	B	%	3.1	k_{ol-cl}
SFO	params.ctrl.speed.ol_cl_tr_coeff	B	%	4.1	k_{ol-cl}
TBC	params.ctrl.speed.ol_cl_tr_coeff	B	%	5.1	k_{ol-cl}
RFO	params.ctrl.speed.kp	A	A/(Ra/sec-elec)	3.1	k_p
RFO	params.ctrl.speed.ki	A	A/(Ra/sec-elec).(Ra/sec)	3.1	k_i
TBC	params.ctrl.speed.kp	A	A/(Ra/sec-elec)	5.1	k_p
TBC	params.ctrl.speed.ki	A	A/(Ra/sec-elec).(Ra/sec)	5.1	k_i
SFO	params.ctrl.speed.kp	A	Nm/(Ra/sec-elec)	4.1	k_p
SFO	params.ctrl.speed.ki	A	Nm/(Ra/sec-elec).(Ra/sec)	4.1	k_i
RFO	params.ctrl.speed.ff_k_inertia	A	A/(Ra/sec-elec).sec	3.1	J'
RFO	params.ctrl.speed.ff_k_viscous	A	A/(Ra/sec-elec)	3.1	B'
RFO	params.ctrl.speed.ff_k_friction	A	A	3.1	T_f'
TBC	params.ctrl.speed.ff_k_inertia	A	A/(Ra/sec-elec).sec	5.1	J'
TBC	params.ctrl.speed.ff_k_viscous	A	A/(Ra/sec-elec)	5.1	B'
TBC	params.ctrl.speed.ff_k_friction	A	A	5.1	T_f'
SFO	params.ctrl.speed.ff_k_inertia	A	Nm/(Ra/sec-elec).sec	4.1	J'
SFO	params.ctrl.speed.ff_k_viscous	A	Nm/(Ra/sec-elec)	4.1	B'
SFO	params.ctrl.speed.ff_k_friction	A	Nm	4.1	T_f'
TBC	params.ctrl.curr.bypass	B		5.2	$-NA$
TBC	params.ctrl.curr.k_bypass	A	V/A	5.2	$r/\sqrt{2}$
RFO	params.ctrl.curr.bw	B	Hz	3.4	ω_c
TBC	params.ctrl.curr.bw	B	Hz	5.2	ω_c
RFO	params.ctrl.curr.ff_coef	B	%	3.4	k_{ff}
TBC	params.ctrl.curr.ff_coef	B	%	5.2	k_{ff}
RFO	params.ctrl.curr.i_cmd_thresh	B	A	3.4, 6.16	I^*_{thresh}
TBC	params.ctrl.curr.i_cmd_thresh	B	A	5.2, 6.16	I^*_{thresh}
RFO	params.ctrl.curr.i_cmd_hyst	B	A	3.4, 6.16	I^*_{hyst}
TBC	params.ctrl.curr.i_cmd_hyst	B	A	5.2, 6.16	I^*_{hyst}
RFO	params.ctrl.curr.i_cmd_ol	B	A	6.10	i_q^{r*}
TBC	params.ctrl.curr.i_cmd_ol	B	A	6.10	i_q^{r*}
RFO	params.ctrl.curr.kp.q	A	V/A	3.4	k_p
RFO	params.ctrl.curr.kp.d	A	V/A	3.4	k_p
RFO	params.ctrl.curr.ki.q	A	V/A.(Ra/sec)	3.4	k_i
RFO	params.ctrl.curr.ki.d	A	V/A.(Ra/sec)	3.4	k_i
RFO	params.ctrl.curr.v_max.q	A	V	3.4	V_{max}
RFO	params.ctrl.curr.v_max.d	A	V	3.4	V_{max}
TBC	params.ctrl.curr.kp	A	V/A	5.2	$k_p/\sqrt{2}$
TBC	params.ctrl.curr.ki	A	V/A.(Ra/sec)	5.2	$k_i/\sqrt{2}$
TBC	params.ctrl.curr.v_max	A	V	5.2	V_{max}
SFO	params.ctrl.trq.w_p	B	Hz	4.5	ω_p
SFO	params.ctrl.trq.w_ratio	B	%	4.5	r
SFO	params.ctrl.trq.delta_max.elec	B	Deg	4.5	δ_{max}
SFO	params.ctrl.trq.T_cmd_thresh	B	Nm	4.5, 6.16	T^*_{thresh}
SFO	params.ctrl.trq.T_cmd_hyst	B	Nm	4.5, 6.16	T^*_{hyst}
SFO	params.ctrl.trq.curr_lmt_t_reach	B	Sec	4.9, 4.9.1, 4.9.2	T_{reach}
SFO	params.ctrl.trq.kp	A	(Ra-elec)/(Nm)	4.5	k_p
SFO	params.ctrl.trq.ki	A	(Ra/sec).(Ra-elec)/(Nm)	4.5	k_i

SFO	params.ctrl.trq.curr_lmt_ki	A	Nm/A.Ra/sec	4.9, 4.9.1, 4.9.2	kT_{s0}
SFO	params.ctrl.flux.bw	B	Hz	4.6	ω_0
SFO	params.ctrl.flux.pole_sep	B	-	4.6	r
SFO	params.ctrl.flux.kp	A	V/Wb	4.6	k_p
SFO	params.ctrl.flux.ki	A	(Ra/sec).V/Wb	4.6	k_i
SFO	params.ctrl.flux.vd_max	A	V	4.6	$V_{\mathrm{d,max}}$
SFO	params.ctrl.delta.bw	B	Hz	4.7	ω_0
SFO	params.ctrl.delta.bw_mult	B	-	4.7	m
SFO	params.ctrl.delta.bw_mult_wl.elec	B	Hz(elec.)	4.7	ω_L
SFO	params.ctrl.delta.bw_mult_wh.elec	B	Hz(elec.)	4.7	ω_H
SFO	params.ctrl.delta.pole_sep	B	-	4.7	r
SFO	params.ctrl.delta.bw_mult_slope	A	1/(Rad/Sec)	4.7	$-\mathrm{NA}$
SFO	params.ctrl.delta.bw_mult_inter	A	-	4.7	$-\mathrm{NA}$
SFO	params.ctrl.delta.vq_max	A	V	4.7	$V_{\mathrm{q,max}}$
ALL	params.ctrl.volt.w_thresh.elec	B	Hz(elec.)	6.9, 6.16	$\omega^*_{\mathrm{thresh}}$
ALL	params.ctrl.volt.w_hyst.elec	B	Hz(elec.)	6.9, 6.16	ω^*_{hyst}
ALL	params.ctrl.volt.v_min	B	V	6.9	V_{min}
ALL	params.ctrl.volt.v_to_f_ratio	B	V/(Rad/Sec)	6.9	K
ALL	params.ctrl.volt.mod_method	B	-	6.3	$-\mathrm{NA}$
ALL	params.ctrl.volt.five_seg.en	B	-	6.3.1	$-\mathrm{NA}$
ALL	params.ctrl.volt.five_seg.active_mi	B	%	6.3.1	$m \times 4/3$
ALL	params.ctrl.volt.five_seg.inactive_mi	B	%	6.3.1	$m \times 4/3$
ALL	params.ctrl.volt.five_seg.w0_filt	B	Hz	6.3.1	$\omega_{0,\mathrm{filt}}$
RFO	params.ctrl.flux_weaken.en	B	-	3.3	$-\mathrm{NA}$
SFO	params.ctrl.flux_weaken.en	B	-	4.3	$-\mathrm{NA}$
RFO	params.ctrl.flux_weaken.vdc_coeff	B	-	3.3	$c/\sqrt{3}$
SFO	params.ctrl.flux_weaken.vdc_coeff	B	-	4.3	$c/\sqrt{3}$
RFO	params.ctrl.flux_weaken.bw	B	Hz	3.3	ω_c
RFO	params.ctrl.flux_weaken.ki	A	Ra/sec.A/V	3.3	k_i
SFO	params.ctrl.flux_weaken.w_min.elec	A	Hz(elec.)	4.3	ω_{min}
SFO, RFO	params.ctrl.align.time	B	Sec	6.11.1	T_{align}
SFO, RFO	params.ctrl.align.voltage	B	V	6.11.1	$V_{\alpha,\mathrm{pre}}$
SFO, RFO	params.ctrl.six_pulse_inj.i_peak	B	A	6.11.2.3	i_p
SFO, RFO	params.ctrl.six_pulse_inj.t_on	A	Sec	6.11.2.3	t_{on}
SFO, RFO	params.ctrl.six_pulse_inj.t_off	A	Sec	6.11.2.3	t_{off}
SFO, RFO	params.ctrl.six_pulse_inj.v_pulse	A	V	6.11.2.3	V
RFO	params.ctrl.high_freq_inj.w_h	B	Hz	6.11.3.1, 6.11.3.3	ω_h
SFO	params.ctrl.high_freq_inj.w_h	B	Hz	6.11.3.2, 6.11.3.3	ω_h
RFO	params.ctrl.high_freq_inj.w_sep	B	Hz	6.11.3.1, 6.11.3.3	ω_{sep}
SFO	params.ctrl.high_freq_inj.w_sep	B	Hz	6.11.3.2, 6.11.3.3	ω_{sep}
RFO	params.ctrl.high_freq_inj.i_qd_r_peak.d	B	A	6.11.3.1, 6.11.3.3	I_d
SFO	params.ctrl.high_freq_inj.i_qd_r_peak.d	B	A	6.11.3.2, 6.11.3.3	I_d
RFO	params.ctrl.high_freq_inj.v_qd_r_coeff.q	A	V/(Ra/sec)	6.11.3.1, 6.11.3.3	$I_d L_d$
SFO	params.ctrl.high_freq_inj.v_qd_r_coeff.q	A	V/(Ra/sec)	6.11.3.2, 6.11.3.3	$I_d L_d$
RFO	params.ctrl.high_freq_inj.v_qd_r_coeff.d	A	V	6.11.3.1, 6.11.3.3	$I_d L_d \omega_h$

SFO	params.ctrl.high_freq_inj.v_qd_r_coeff.d	A	V	6.11.3.2, 6.11.3.3	$I_d L_d \omega_h$
RFO	params.ctrl.high_freq_inj.i_qd_r_peak.q	A	A	6.11.3.1, 6.11.3.3	I_q
SFO	params.ctrl.high_freq_inj.i_qd_r_peak.q	A	A	6.11.3.2, 6.11.3.3	I_q
RFO	params.ctrl.high_freq_inj.bw_red_coeff	A	%	6.11.3.1, 6.11.3.3	$k_{\omega,\mathrm{red}}$
SFO	params.ctrl.high_freq_inj.bw_red_coeff	A	%	6.11.3.2, 6.11.3.3	$k_{\omega,\mathrm{red}}$
RFO	params.ctrl.high_freq_inj.lock_time	A	Sec	6.11.3.1, 6.16	$T_{\mathrm{HFI,lock}}$
SFO	params.ctrl.high_freq_inj.lock_time	A	Sec	6.11.3.2, 6.16	$T_{\mathrm{HFI,lock}}$
RFO	params.ctrl.high_freq_inj.pll.w0	B	Hz	6.11.3.1, 6.11.3.3	ω_0
SFO	params.ctrl.high_freq_inj.pll.w0	B	Hz	6.11.3.2, 6.11.3.3	ω_0
RFO	params.ctrl.high_freq_inj.pll.w_max.elec	B	Hz(elec.)	6.11.3.1, 6.11.3.3	$\omega_{\max}$
SFO	params.ctrl.high_freq_inj.pll.w_max.elec	B	Hz(elec.)	6.11.3.2, 6.11.3.3	$\omega_{\max}$
RFO	params.ctrl.high_freq_inj.pll.kp	A	(Ra/sec-elec)/A	6.11.3.1, 6.11.3.3	k_p
SFO	params.ctrl.high_freq_inj.pll.kp	A	(Ra/sec-elec)/A	6.11.3.2, 6.11.3.3	k_p
RFO	params.ctrl.high_freq_inj.pll.ki	A	(Ra/sec).(Ra/sec-elec)/A	6.11.3.1, 6.11.3.3	k_i
SFO	params.ctrl.high_freq_inj.pll.ki	A	(Ra/sec).(Ra/sec-elec)/A	6.11.3.2, 6.11.3.3	k_i
RFO	params.ctrl.high_freq_inj.pll.th_offset.elec	A	Deg	6.11.3.1, 6.11.3.3	θ_{offset}
SFO	params.ctrl.high_freq_inj.pll.th_offset.elec	A	Deg	6.11.3.2, 6.11.3.3	θ_{offset}
RFO	params.ctrl.high_freq_inj.lpf_biquad_a[0]	A	-	6.11.3.1, 6.11.3.3	a_0
SFO	params.ctrl.high_freq_inj.lpf_biquad_a[0]	A	-	6.11.3.2, 6.11.3.3	a_0
RFO	params.ctrl.high_freq_inj.lpf_biquad_a[1]	A	1/(Ra/sec)	6.11.3.1, 6.11.3.3	a_1
SFO	params.ctrl.high_freq_inj.lpf_biquad_a[1]	A	1/(Ra/sec)	6.11.3.2, 6.11.3.3	a_1
RFO	params.ctrl.high_freq_inj.lpf_biquad_a[2]	A	1/((Ra/sec)^2)	6.11.3.1, 6.11.3.3	a_2
SFO	params.ctrl.high_freq_inj.lpf_biquad_a[2]	A	1/((Ra/sec)^2)	6.11.3.2, 6.11.3.3	a_2
RFO	params.ctrl.high_freq_inj.lpf_biquad_b[0]	A	-	6.11.3.1, 6.11.3.3	b_0
SFO	params.ctrl.high_freq_inj.lpf_biquad_b[0]	A	-	6.11.3.2, 6.11.3.3	b_0
RFO	params.ctrl.high_freq_inj.lpf_biquad_b[1]	A	1/(Ra/sec)	6.11.3.1, 6.11.3.3	b_1
SFO	params.ctrl.high_freq_inj.lpf_biquad_b[1]	A	1/(Ra/sec)	6.11.3.2, 6.11.3.3	b_1
RFO	params.ctrl.high_freq_inj.lpf_biquad_b[2]	A	1/((Ra/sec)^2)	6.11.3.1, 6.11.3.3	b_2
SFO	params.ctrl.high_freq_inj.lpf_biquad_b[2]	A	1/((Ra/sec)^2)	6.11.3.2, 6.11.3.3	b_2
SFO, RFO	params.profiler.overwrite	B	-	6.18.1	$-\mathrm{NA}$
SFO, RFO	params.profiler.cmd_thresh	B	%	6.18.1	$-\mathrm{NA}$
SFO, RFO	params.profiler.cmd_hyst	B	%	6.18.1	$-\mathrm{NA}$
SFO, RFO	params.profiler.i_cmd_dc	B	A	6.18.1.2, 6.18.1.3	I_α^*
SFO, RFO	params.profiler.i_cmd_ac	B	A	6.18.1.4	$\lvert\hat{\imath}\rvert^*$
SFO, RFO	params.profiler.w_cmd_elec.min	B	Hz(elec.)	6.18.1.5	$\omega_{\min}^*$
SFO, RFO	params.profiler.w_cmd_elec.max	B	Hz(elec.)	6.18.1.5	$\omega_{\max}^*$
SFO, RFO	params.profiler.time_rot_lock	B	Sec	6.18.1.2	T_{lock}
SFO, RFO	params.profiler.time_spd	B	Sec	6.18.1.5	T_{spd}
SFO, RFO	params.profiler.w_h	A	Hz	6.18.1.4	ω_h

SFO, RFO	params.profiler.w_sep	A	Hz	6.18.1.3, 6.18.1.4	ω_{sep}
SFO, RFO	params.profiler.w0_idc	A	Hz	6.18.1.3	$\omega_{0,i}$
SFO, RFO	params.profiler.kp_idc	A	V/A	6.18.1.3	k_p
SFO, RFO	params.profiler.ki_idc	A	(V/A).(Ra/sec)	6.18.1.3	k_i
SFO, RFO	params.profiler.time_res	A	Sec	6.18.1.3	T_{res}
SFO, RFO	params.profiler.time_ind	A	Sec	6.18.1.4	T_{ind}
SFO, RFO	params.profiler.w0_flux	A	Hz	6.18.1.5	ω_{0,λ_m}
TBC	params.ctrl.tbc.mode	B	-	5.3, 5.4	$-\text{NA}$
TBC	params.ctrl.tbc.trap.ramp_cnt	B	-	5.4	N_{ramp}
TBC	params.ctrl.tbc.trap.ramp_main_bw_ratio	B	-	5.4	k_ω
TBC	params.ctrl.tbc.trap.ramp_ff_coef	B	V/A	5.4	$k_f f_{s0}$
TBC	params.ctrl.tbc.trap.main_ff_coef	B	-	5.4	1
TBC	params.ctrl.tbc.trap.ramp_kp	A	V/A	5.4	k_p
TBC	params.ctrl.tbc.trap.ramp_ki	A	(V/A).(Ra/sec)	5.4	k_i
TBC	params.ctrl.tbc.trap.ramp_cnt_inv	A	-	5.4	$1/N_{\text{ramp}}$
RFO	Params.ctrl.position.bw	B	Hz	3.7	ω_0
RFO	Params.ctrl.position.pole_sep	B	-	3.7	r
RFO	Params.ctrl.position.ff_coef	B	%	3.7	-
RFO	Params.position.pi_output_limit	B	Ra/Sec	3.7	-
RFO	Params.ctrl.position.kp	A	(Ra/Sec-elec)/(Ra)	3.7	k_p
RFO	Params.ctrl.position.ki	A	(Ra/sec). (Ra/Sec-elec)/(Ra)	3.7	k_i