

THESIS

SIMULATION AND HARDWARE VALIDATION OF METHODS FOR SYNCHRONIZATION
OF CENTRAL-CONVERTER MULTI-MOTOR ELECTRIC ACTUATION SYSTEMS

Submitted by

Zane P. Miller

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2023

Master's Committee:

Advisor: James Cale

Edwin Chong

William Fairbank

Copyright by Zane P. Miller 2023

All Rights Reserved

ABSTRACT

SIMULATION AND HARDWARE VALIDATION OF METHODS FOR SYNCHRONIZATION OF CENTRAL-CONVERTER MULTI-MOTOR ELECTRIC ACTUATION SYSTEMS

Replacement of previously hydraulic and pneumatic drives with power-electronic drive systems to reduce weight and maintenance requirements is a current target of research in the aerospace industry. This includes electrification of thrust reverser actuation systems (TRAS), which redirect thrust produced by the aircraft's engines to aid with deceleration upon landing, reducing wear on the brakes. However, one challenge of developing an electromagnetic TRAS (EM-TRAS) is the requirement of speed and position synchronization of all motors in the system, despite unequal torque loading from differing wind forces. Use of a single ("central") power electronic converter to power a set of induction machines in parallel could potentially lower cost and weight requirements compared to the use of separate converters, but such a central-converter, multi-motor (CCMM) architecture requires some form of compensation for load torque differences. Previous research presented a synchronization methodology using closed-loop feedback control of variable stator resistances in parallel with each induction machine. This thesis builds on this research by presenting an alternative methodology that instead applies closed-loop feedback control to smaller-scale auxiliary converters for each motor line, coupled to the induction machines using transformers to apply adjustments to the stator voltage. This new methodology achieves similar synchronization performance with better energy efficiency, lowering power requirements for its use compared to the external resistance methodology. The author's contributions to construction of a testbed for aerospace actuation system research are also presented in this thesis, with applications including hardware validation of the external resistance CCMM EM-TRAS implementation.

TABLE OF CONTENTS

	ABSTRACT	ii
	LIST OF TABLES	iv
	LIST OF FIGURES	v
Chapter 1	Introduction	1
Chapter 2	Background	4
2.1	Induction Machines	4
2.2	External Stator Resistance Methodology	8
2.3	Auxiliary Converter Methodology	11
2.4	Diagram of Auxiliary Converter and Transformer	15
Chapter 3	Numerical Case Studies	18
3.1	Case Study 1a: Central Voltage Control, Ideal Switching	19
3.2	Case Study 1b: Central Voltage Control, Lossy Switching	23
3.3	Case Study 2a: Central Current Control, Ideal Switching	27
3.4	Case Study 2b: Central Current Control, Lossy Switching	31
3.5	Remarks	35
Chapter 4	Testbed	37
Chapter 5	Future Work	43
Chapter 6	Conclusions	44
	Bibliography	45
Appendix A	Simulation Parameters	46
Appendix B	MATLAB [®] Code for Simulation and Analysis	49
B.1	CVHz Control, External Resistor Method	49
B.2	CVHz Control, Auxiliary Converter Method	57
B.3	IDFOC Control, External Resistor Method	70
B.4	IDFOC Control, Auxiliary Converter Method	77
B.5	Auxiliary Functions	89
B.6	Data Analysis	96

LIST OF TABLES

3.1	Efficiency analysis for CVHz control, ideal switching.	23
3.2	Efficiency analysis for CVHz control, lossy switching.	26
3.3	Efficiency analysis for IDFOC control, ideal switching.	31
3.4	Efficiency analysis for IDFOC control, lossy switching.	34
A.1	Parameters for 15 hp Induction Machine.	46
A.2	Parameters for Compensated Volts-per-Hertz Control.	46
A.3	Parameters for Speed Control.	46
A.4	Parameters for Indirect Field-oriented Control.	47
A.5	Parameters for External Resistor Circuit and Control.	47
A.6	Parameters for Auxiliary Converter Circuit and Control.	47
A.7	Parameters for Auxiliary Converter Transformers.	48

LIST OF FIGURES

1.1	Basic TRAS diagram.	1
1.2	CCMM architecture.	2
2.1	Induction machine equivalent $qd0$ circuits.	8
2.2	External resistance torque vs. speed plot.	9
2.3	External stator resistance circuit.	9
2.4	Voltage adjustment torque vs. speed plot.	11
2.5	High-level auxiliary converter implementation.	12
2.6	Auxiliary converter methodology equivalent $qd0$ circuits.	15
2.7	Auxiliary converter methodology block diagram.	16
3.1	Full velocity plot, CVHz control, ideal switching.	20
3.2	Zoomed velocity plot, CVHz control, ideal switching.	20
3.3	Electromagnetic torque plot, CVHz control, ideal switching.	21
3.4	Angle error plots, CVHz control, ideal switching.	21
3.5	External resistance plots, CVHz control, ideal switching.	22
3.6	Auxiliary converter voltage plots, CVHz control, ideal switching.	22
3.7	Full velocity plot, CVHz control, lossy switching.	24
3.8	Zoomed velocity plot, CVHz control, lossy switching.	24
3.9	Electromagnetic torque plot, CVHz control, ideal switching.	25
3.10	Angle error plots, CVHz control, lossy switching.	25
3.11	External resistance plots, CVHz control, lossy switching.	26
3.12	Auxiliary converter voltage plots, CVHz control, lossy switching.	26
3.13	Full velocity plot, IDFOC control, ideal switching.	28
3.14	Zoomed velocity plot, IDFOC control, ideal switching.	28
3.15	Electromagnetic torque plot, IDFOC control, ideal switching.	29
3.16	Angle error plots, IDFOC control, ideal switching.	29
3.17	External resistance plots, IDFOC control, ideal switching.	30
3.18	Auxiliary converter voltage plots, IDFOC control, ideal switching.	30
3.19	Full velocity plot, IDFOC control, lossy switching.	31
3.20	Zoomed velocity plot, IDFOC control, lossy switching.	32
3.21	Electromagnetic torque plot, IDFOC control, lossy switching.	32
3.22	Angle error plots, IDFOC control, lossy switching.	33
3.23	External resistance plots, IDFOC control, lossy switching.	34
3.24	Auxiliary converter voltage plots, IDFOC control, lossy switching.	34
4.1	Functional block diagram of the aerospace testbed.	37
4.2	Screenshot of SCADA user interface.	38
4.3	Screenshot of the HMI.	39
4.4	Photographs of test stand and motor control cabinet.	41
4.5	Hardware position synchronization results.	42

Chapter 1

Introduction

The aerospace industry is a constant target for innovation, with researchers always seeking improvements to the efficiency, performance, and safety of aircraft. This includes the development of More Electric Aircraft (MEA): by replacing non-propulsive mechanical (e.g., hydraulic and pneumatic) systems with electrical systems, benefits can be obtained such as reduced weight and cost, improved performance, and more comprehensive diagnostics. One system in particular targeted for electrification is the thrust reverser actuation system (TRAS), which diverts the engine's thrust to reverse the force applied on the aircraft, reducing the time and length of runway required to slow down after landing and wear on the brakes. As illustrated in Figure 1.1, the exhaust is redirected by the TRAS to leave the nacelle in a forward direction, producing a backward thrust force.

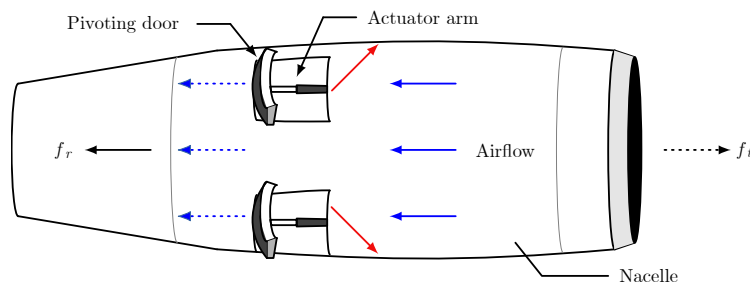


Figure 1.1: Basic diagram of a TRAS, showing normal (blue, dotted) and reversed (red, solid) airflow within the nacelle and thrust forces produced.

Operation of an electromechanical thrust reverser actuation system (EM-TRAS) requires electric drives and motors for supplying the force/torque necessary for deploying and stowing the thrust reversers. Different portions of the system would experience different external forces in operation, so the conceptually simplest method for keeping the speed and position of all motors in the system synchronized is to use a separate converter and drive for each motor, allowing for full individual control in an architecture known as distributed converter multi-motor (DCMM). This thesis builds on research exploring an alternative architecture, central converter multi-motor (CCMM), where a

single central converter is used to power all of the motors and the drive conditions the power input to each motor to achieve synchronization. A diagram of a CCMM EM-TRAS is shown in Figure 1.2. Use of CCMM architecture could reduce the size, weight, and cost of the EM-TRAS, making it an attractive option. [1]

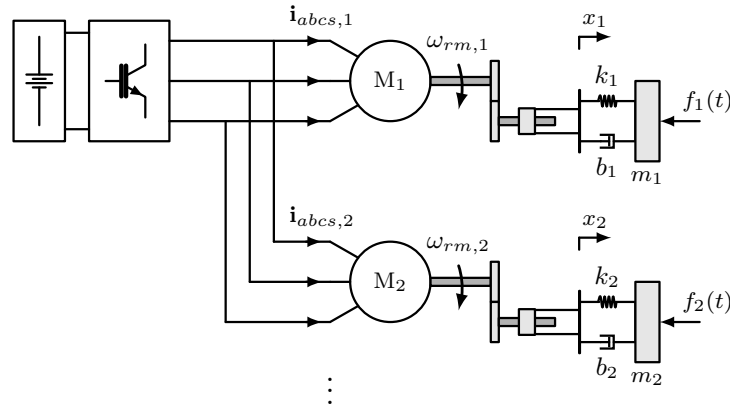


Figure 1.2: Notional depiction of CCMM architecture for EM-TRAS.

The implementation presented in [1] uses external stator resistance controlled by PWM switching to achieve speed and position synchronization. Disadvantages of this method include that rapidly switching the external resistance on and off produces large transient voltages that can damage equipment if not mitigated and a significant portion of power provided by the central converter must be dissipated in the external resistance to achieve synchronization (see Chapter 4). Better efficiency and performance could potentially be achieved through the use of external impedance including an inductive or capacitive component in addition to resistance. However, existing methods for electronic variable impedance control investigated by this researcher appear to be few in number, with some publicly documented only in patents that describe the method in terms too vague for replication, and one method requiring a separate voltage converter controlled using inline current readings [2], introducing complexity that may negate the advantages of CCMM architecture.

In the first part of this thesis, a novel CCMM implementation using low-power auxiliary converters for speed and position synchronization is presented. Chapter 2 explains the methodology for this implementation, including the principle of operation, the expected transient and steady-

state behavior, and a system diagram of a potential implementation. Chapter 3 presents results of simulations investigating this implementation using MATLAB and compares its performance and efficiency to that of the implementation presented in [1].

Chapter 4 constitutes the second part of this thesis, which details the author's contribution to a testbed designed for hardware emulation of various aerospace systems. The primary initial purpose of this testbed is hardware validation of the resistor-based CCMM EM-TRAS implementation presented in [1]. Design and construction of this testbed is explained in more detail in [3].

Plans and potential for future work building on the research presented in this thesis are detailed in Chapter 5. This includes nonlinear optimization of the novel CCMM implementation using a more detailed CCMM model and hardware validation of the implementation using the testbed. Finally, key findings and the author's specific contributions to the research presented here are summarized in Chapter 6.

Chapter 2

Background

This chapter begins with an overview of induction machines (IMs) along with the external stator resistance method for speed and position synchronization, referenced from [1]. Then, the novel auxiliary converter method is presented along with a full diagram of the circuitry required for implementation.

2.1 Induction Machines

An induction machine (IM) can be operated as a motor or a generator, though this thesis primarily focuses on motor operation. An IM contains stationary stator windings that magnetically couple with rotor windings inside the rotor when alternating current is applied, causing rotation. Roller bearings mounted on the shaft allow it to rotate freely while keeping a fixed gap distance between the stator and rotor. In this thesis, it is assumed for simplicity that the windings are sinusoidally distributed, the machine parameters are electrically symmetric, and the rotor windings are short-circuited at their ends (a “squirrel-cage” rotor).

The “electrical” angular velocity ω_r is related to the (actual) mechanical angular velocity as $\omega_r = (P/2)\omega_{rm}$ where P is the number of magnetic poles. The electrical angular position θ_r at time t is defined as

$$\theta_r(t) = \int_0^t \omega_r(\tau) d\tau + \theta_r(0), \quad (2.1)$$

where $\theta_r(0)$ is the initial electrical angular position. Voltage equations for the IM are expressed as follows:

$$\begin{aligned} \mathbf{v}_{abcs} &= \mathbf{r}_s \mathbf{i}_{abcs} + p \boldsymbol{\lambda}_{abcs} \\ \mathbf{v}_{abcr} &= \mathbf{r}_r \mathbf{i}_{abcr} + p \boldsymbol{\lambda}_{abcr}, \end{aligned} \quad (2.2)$$

where $\mathbf{v}_{abc(s(r))}$, $\mathbf{i}_{abc(s(r))}$, and $\boldsymbol{\lambda}_{abc(s(r))}$ represent the stator (motor) voltage, current, and flux linkages, respectively, while \mathbf{r}_s and \mathbf{r}_r are the stator and rotor resistance matrices

$$\mathbf{r}_s = \begin{bmatrix} r_s & 0 & 0 \\ 0 & r_s & 0 \\ 0 & 0 & r_s \end{bmatrix}, \quad \mathbf{r}_r = \begin{bmatrix} r_r & 0 & 0 \\ 0 & r_r & 0 \\ 0 & 0 & r_r \end{bmatrix}, \quad (2.3)$$

where r_s and r_r are the resistances of each stator and rotor winding, assumed to be equal for each phase (but generally $r_s \neq r_r$).

The flux linkages in (2.2) can be expressed as

$$\begin{aligned} \boldsymbol{\lambda}_{abc(s)} &= \mathbf{L}_s \mathbf{i}_{abc(s)} + \mathbf{L}_{sr} \mathbf{i}_{abc(r)} \\ \boldsymbol{\lambda}_{abc(r)} &= \mathbf{L}_r \mathbf{i}_{abc(r)} + \mathbf{L}_{sr}^\top \mathbf{i}_{abc(s)}, \end{aligned} \quad (2.4)$$

where the self-inductance matrices \mathbf{L}_s , \mathbf{L}_r are

$$\begin{aligned} \mathbf{L}_s &= \begin{bmatrix} L_{ls} + L_{ms} & -\frac{1}{2}L_{ms} & -\frac{1}{2}L_{ms} \\ -\frac{1}{2}L_{ms} & L_{ls} + L_{ms} & -\frac{1}{2}L_{ms} \\ -\frac{1}{2}L_{ms} & -\frac{1}{2}L_{ms} & L_{ls} + L_{ms} \end{bmatrix}, \\ \mathbf{L}_r &= \begin{bmatrix} L_{lr} + L_{mr} & -\frac{1}{2}L_{mr} & -\frac{1}{2}L_{mr} \\ -\frac{1}{2}L_{mr} & L_{lr} + L_{mr} & -\frac{1}{2}L_{mr} \\ -\frac{1}{2}L_{mr} & -\frac{1}{2}L_{mr} & L_{lr} + L_{mr} \end{bmatrix} \end{aligned} \quad (2.5)$$

where L_{ls} and L_{lr} are the stator and rotor leakage inductances, and L_{ms} and L_{mr} are the stator and rotor magnetizing inductances. The mutual inductance matrix is

$$\mathbf{L}_{sr} = L_{sr} \begin{bmatrix} \cos \theta_r & \cos(\theta_r + \frac{2\pi}{3}) & \cos(\theta_r - \frac{2\pi}{3}) \\ \cos(\theta_r - \frac{2\pi}{3}) & \cos \theta_r & \cos(\theta_r + \frac{2\pi}{3}) \\ \cos(\theta_r + \frac{2\pi}{3}) & \cos(\theta_r - \frac{2\pi}{3}) & \cos \theta_r \end{bmatrix}, \quad (2.6)$$

where L_{sr} is the maximum mutual inductance between stator and rotor windings. To eliminate the angular dependence in the machine equations arising from (2.6), transformation of the equations to a rotating reference frame is employed. Transformation of the stationary variables to $qd0$ variables in the arbitrary rotating reference frame is achieved by the transformation

$$\mathbf{f}_{qd0s} = \mathbf{K}_s \mathbf{f}_{abcs}, \quad (2.7)$$

where \mathbf{f} is any transformed quantity and the matrix \mathbf{K}_s is defined as

$$\mathbf{K}_s(\theta) = \frac{2}{3} \begin{bmatrix} \cos \theta & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\ \sin \theta & \sin(\theta - \frac{2\pi}{3}) & \sin(\theta + \frac{2\pi}{3}) \\ 1/2 & 1/2 & 1/2 \end{bmatrix}, \quad (2.8)$$

with the arbitrary reference frame position angle θ defined from the reference frame speed ω as

$$\theta(t) = \int_0^t \omega(\tau) d\tau + \theta(0). \quad (2.9)$$

Transformation of the rotor variables to $qd0$ variables in the arbitrary rotating reference frame is achieved by the transformation

$$\mathbf{f}_{qd0r} = \mathbf{K}_r \mathbf{f}_{abcr}, \quad (2.10)$$

where the matrix \mathbf{K}_r is defined as

$$\mathbf{K}_r(\beta) = \mathbf{K}_s(\theta - \theta_r). \quad (2.11)$$

For ease of notation, variables are referred based on the turns ratio between stator and rotor windings. The voltage, flux linkages, currents, and electrical parameters of the rotor are referred to the

stator side as

$$\begin{aligned} \mathbf{v}'_{qdr} &= \frac{N_s}{N_r} \mathbf{v}_{qdr}, & \boldsymbol{\lambda}'_{qdr} &= \frac{N_s}{N_r} \boldsymbol{\lambda}_{qdr}, & \mathbf{i}'_{qdr} &= \frac{N_r}{N_s} \mathbf{i}_{qdr}, \\ r'_r &= \left(\frac{N_s}{N_r} \right)^2 r_r, & L'_{lr} &= \left(\frac{N_s}{N_r} \right)^2 L_{lr}, \end{aligned} \quad (2.12)$$

where N_s and N_r are the equivalent turns of the stator and rotor windings, respectively. After transformation of (2.2)–(2.4) to the arbitrary rotating reference frame using (2.7)–(2.11) and the referral of variables from (2.12), the voltage equations in qd variables are expressed as

$$\begin{aligned} v_{qs} &= r_s i_{qs} + \omega \lambda_{ds} + p \lambda_{qs} \\ v_{ds} &= r_s i_{ds} - \omega \lambda_{qs} + p \lambda_{ds} \\ v_{qr} &= r'_r i'_{qr} + (\omega - \omega_r) \lambda'_{dr} + p \lambda'_{qr} \\ v_{dr} &= r'_r i'_{dr} - (\omega - \omega_r) \lambda'_{qr} + p \lambda'_{dr}, \end{aligned} \quad (2.13)$$

and flux linkages in qd variables are expressed as

$$\begin{aligned} \lambda_{qs} &= L_{ls} i_{qs} + L_M (i_{qs} + i'_{qr}) \\ \lambda_{ds} &= L_{ls} i_{ds} + L_M (i_{ds} + i'_{dr}) \\ \lambda_{qr} &= L'_{lr} i'_{qr} + L_M (i_{qs} + i'_{qr}) \\ \lambda_{dr} &= L'_{lr} i'_{dr} + L_M (i_{qs} + i'_{qr}), \end{aligned} \quad (2.14)$$

where $L_M = (3/2)L_{ms}$. Note that in (2.13-2.14) and the remainder of this thesis, a balanced operation is assumed, i.e., $0s(r)$ variables are neglected. By inspection of (2.13-2.14), the differential equations for the qd variables for the stator and rotor circuits can be represented by the equivalent circuits shown in Figure 2.1.

The electromagnetic torque produced by the induction machine can be expressed in qd variables [4] as

$$T_e = \frac{3P}{2} \frac{L_M}{L'_{rr}} (i_{qs} \lambda'_{dr} - i_{ds} \lambda'_{qr}), \quad (2.15)$$

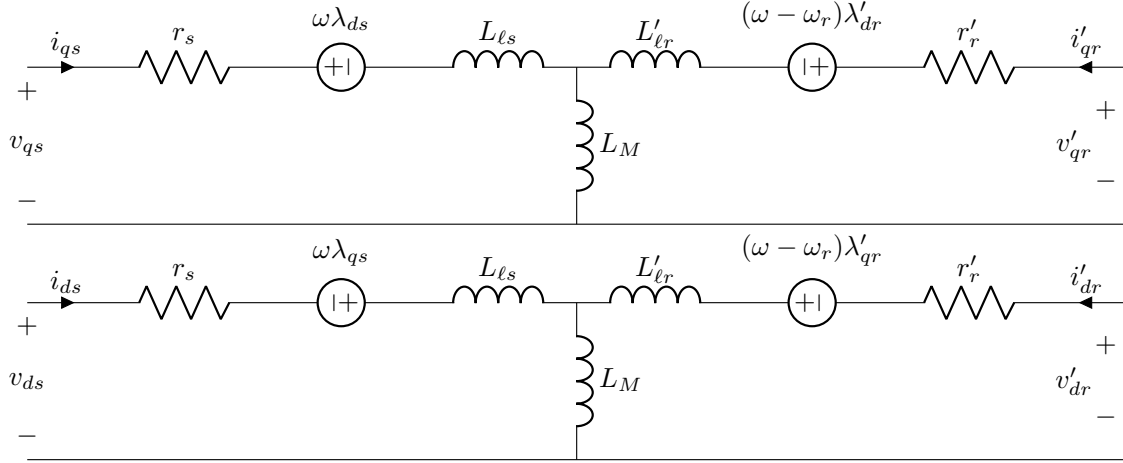


Figure 2.1: Induction machine equivalent $qd0$ circuits.

where $L'_{rr} = L_M + L'_{lr}$ is the rotor self-inductance. This leads to the mechanical motion equation

$$T_e - T_L = Jp\omega_{rm} + B_m\omega_{rm}, \quad (2.16)$$

where T_L is the mechanical load torque (defined in the opposite direction of T_e), J is the rotor's moment of inertia, and B_m is a loss coefficient representing windage and friction.

See [1] for background on the control strategies of compensated volts-per-hertz and indirect field-oriented control, which are used here for completeness of the comparison between the synchronization methodologies but not studied in detail.

2.2 External Stator Resistance Methodology

Induction machines are designed to operate under rated load at a speed slightly below the “synchronous” speed $\omega_r = \omega_e$. From (2.16), steady-state equilibrium of the motor is achieved when the difference between T_e and T_L is equal to the torque resulting from windage and friction and the derivative of T_e with respect to speed is negative [4]. To simplify explanation, this windage and friction term is often neglected, making the equilibrium point $T_e = T_L$.

As explained in [1], adding an external stator resistance to an IM decreases the slope of the equivalent T_e vs. speed curve near synchronous speed. Changing the shape of the curve in this way

moves the equilibrium point to a lower speed, as illustrated in Figure 2.2 for a 15 hp (11.19 kW) induction machine (parameters listed in Appendix A).

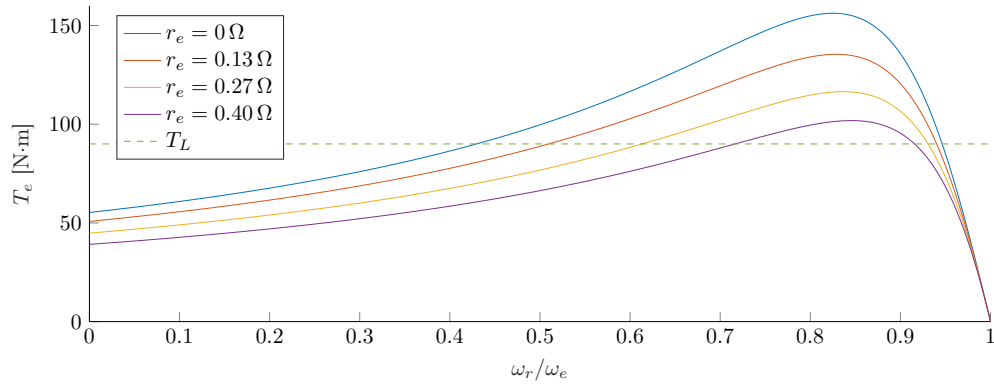


Figure 2.2: Torque vs. speed plot for selected external stator resistance values.

Note that the peak of the torque vs. speed curve moves to a higher speed as the external resistance is increased. The circuit used to implement this external stator resistance in [1] and this thesis is a power-electronic circuit with a bi-directional MOSFET arrangement shown in Figure 2.3.

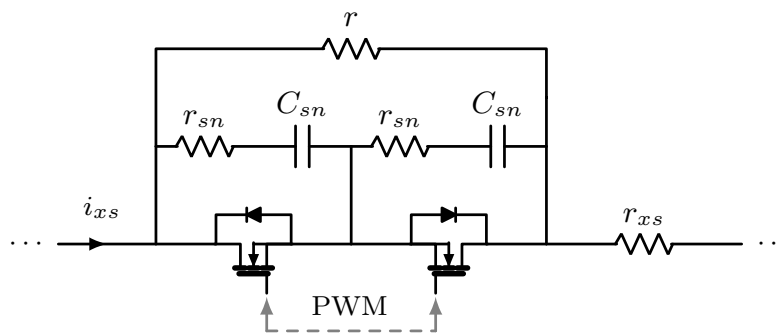


Figure 2.3: Circuit for achieving a desired average external stator resistance.

In Figure 2.3, r_s and C_s are the snubber resistance and capacitance across the back-to-back MOSFETs, i_{xs} and r_{xs} are the current and (nominal) resistance of stator phase x , and r is the ‘base’ external resistance. The MOSFETs receive the same PWM (fully on or fully off) signal;

the bi-directional arrangement allows current to flow in the stator in both directions. Neglecting snubber impedance, the circuit places either base resistance r or 0Ω in series with the nominal stator resistance. By modulation of the PWM duty cycle, a desired external resistance can be achieved in a fast-average sense, where the fast-average of a signal $f(t)$ is defined as

$$\bar{f}(t) = \frac{1}{T_s} \int_0^{T_s} f(t) dt, \quad (2.17)$$

where the bar over $f(t)$ denotes fast-average, and $T_s = 1/f_s$ is the time period of the PWM switching cycle with frequency f_s .

The control strategy used in [1] and this thesis is a proportional-integral (PI) feedback control that updates the duty cycles of the external variable resistance circuits using the angular position differences of the IMs as inputs. In this control strategy, the motor that is experiencing the highest load torque (and therefore runs at the slowest speed without synchronization) is designated as the “primary” and used for the central converter’s feedback control, while all other motors are designated as “secondary”. This designation can be changed during operation to respond to varying load torques. In this thesis, values relating to the primary motor are denoted with a subscript p , while values relating to one of the secondary motors are denoted with a subscript s_i .

The difference in angular position between a secondary motor and the primary motor is denoted

$$\delta\theta_{rm,s_i}(t) = \theta_{rm,s_i}(t) - \theta_{rm,p}(t), \quad (2.18)$$

where

$$\begin{aligned} \theta_{rm,s_i}(t) &= \int_0^t \omega_{rm,s_i}(\tau) d\tau + \theta_{rm,s_i}(0), \\ \theta_{rm,p}(t) &= \int_0^t \omega_{rm,p}(\tau) d\tau + \theta_{rm,p}(0). \end{aligned} \quad (2.19)$$

The duty cycle for the external resistance for the primary motor, $D_{e,p}$, is held at the value it had when the motor was designated as primary (typically zero), while the duty cycles for the external

resistances for the secondary motors, D_{e,s_i} are updated by PI feedback as

$$D_{e,s_i} = K_{P,s_i} \delta\theta_{rm,s_i} + K_{I,s_i} \int \delta\theta_{rm,s_i} dt, \quad (2.20)$$

where K_{P,s_i} is the proportional gain and K_{I,s_i} is the integral gain of the PI controller for motor s_i .

2.3 Auxiliary Converter Methodology

The synchronization method introduced in this thesis instead changes the shape of the T_e vs. speed curve by adjusting the voltage applied across the IM. The effect of this on the equilibrium point is illustrated in Figure 2.4 for a 15 hp (11.19 kW) induction machine (parameters listed in Appendix A), with ΔV defined as the difference between the adjusted operating condition and the rated line-neutral RMS voltage.

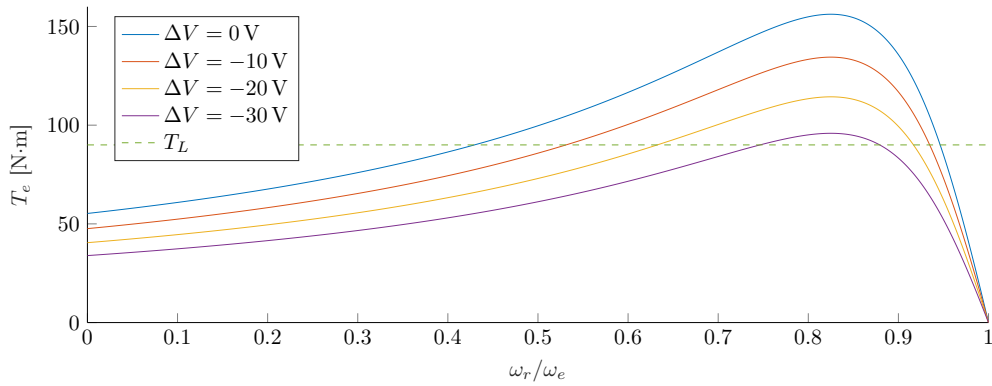


Figure 2.4: Torque vs. speed plot for selected IM input voltage adjustment values.

Note that unlike with applying an external resistance, the peak of the torque vs. speed curve remains at the same speed when adjusting the IM input voltage. The proposed implementation of this method is similar to the variable impedance implementation presented in [2], using an auxiliary converter connected to one side of a transformer, the other side being in series with one of the stator windings. However, the control strategy here is simpler, as the targeted effect is that of an ideal voltage source rather than a variable impedance, and only motor position readings are

necessary. The implementation is illustrated in Figure 2.5, which only displays one phase of the wiring for one motor for clarity.

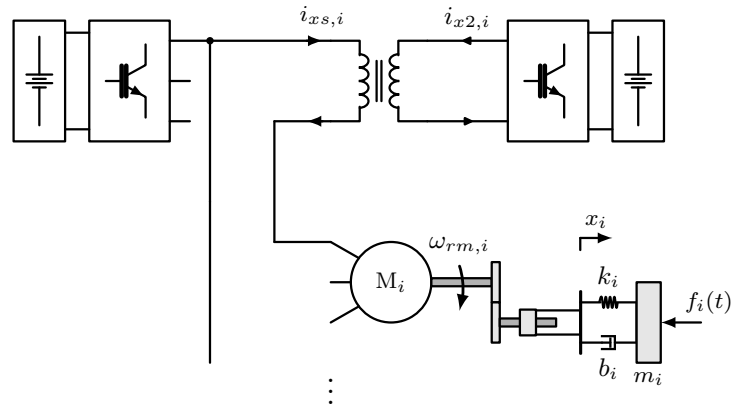


Figure 2.5: High-level per-phase, per-motor diagram of auxiliary converter implementation.

In this thesis, the side of each transformer in line with the stator is denoted the primary, while the side connected to the auxiliary converter is denoted the secondary. Note that this implementation requires independent access to both ends of the primary of each transformer. The secondary may be connected in a three-phase arrangement suitable for the auxiliary converter. (Figure 2.5 displays a neutral connection, but a delta-connected secondary should also be acceptable.)

The transformer is included for isolation between the converters and to allow for the use of an auxiliary converter running at a lower current than the IM, though this requires the converter to operate at a voltage higher than the required IM voltage adjustment. Since a power converter is used to lower the input voltage when slowing down an IM, it may be possible to recover some of the power diverted from the IM, effectively reducing the power requirement for the central converter. However, this requires the use of bidirectional auxiliary converters, which may be more expensive to build or require more complicated circuitry compared to a converter that cannot recover this power. Therefore, this thesis treats the power diverted by the auxiliary converters as “lost” for the purpose of efficiency calculations.

In a practical implementation, the auxiliary converter’s maximum voltage referred across the transformer would be lower than the voltage the central converter is operating at, so the range of

torque differentials for which speed and position synchronization can be achieved is limited. An upper bound on this range is the requirement for steady-state speed synchronization. To examine the steady-state IM voltage adjustment needed for speed synchronization, we use the expression for the steady-state electromagnetic torque for a single-fed, three-phase symmetrical IM presented in [4]. Note that in the three-phase case, we use $L_M = \frac{3}{2}L_{ms}$ in place of L_{ms} and define $L_{ss} = L_{ls} + L_M$ and $L'_{rr} = L'_{lr} + L_M$. We denote the corresponding reactance X_M .

$$T_e = \frac{2(3/2)(P/2)(X_M^2/\omega_e)r'_r s |\tilde{V}_{as}|^2}{[r_s r'_r + s(X_M^2 - X_{ss}X'_{rr})]^2 + (r'_r X_{ss} + s r_s X'_{rr})^2}, \quad (2.21)$$

where the slip $s = (\omega_e - \omega_r)/\omega_e$. Suppose unequal load torques are applied to two IMs. We calculate the required voltage adjustment to retain equal slip using two methods. The first method uses one IM as a reference and reduces the voltage of the other IM to match the slip, which is required when the IMs are operated at rated voltage. The second method adjusts the voltages of both IM to reach an intermediate shared slip, which could be used if operating below rated voltage.

If the slips are equal, the torque can be treated as proportional to the square of the applied rms voltage. We denote the proportionality constant K .

$$K = \frac{2(3/2)(P/2)(X_M^2/\omega_e)r'_r s}{[r_s r'_r + s(X_M^2 - X_{ss}X'_{rr})]^2 + (r'_r X_{ss} + s r_s X'_{rr})^2} \quad (2.22)$$

Note that K is positive in motor mode ($s > 0$) and negative in generator mode ($s < 0$). For small values of s , we have

$$K \approx \frac{3(P/2)(X_M^2/\omega_e)}{r'_r(r_s^2 + X_{ss}^2)} s. \quad (2.23)$$

Therefore, when the slip is small, the torque produced is approximately proportional to s .

We neglect windage/friction for this analysis. Using the first method, IM 1 has applied line-neutral rms voltage V_{as1} and load torque T_1 , while IM 2 has applied line-neutral rms voltage $V_{as1} + \Delta V$ and load torque $T_1 + \Delta T$.

$$\begin{aligned}
T_1 &= KV_{as1}^2, & T_1 + \Delta T &= K(V_{as1} + \Delta V)^2 \\
\frac{KV_{as1}^2 + \Delta T}{K} &= V_{as1}^2 + 2V_{as1}\Delta V + (\Delta V)^2 \\
0 &= -\frac{\Delta T}{K} + 2V_{as1}\Delta V + (\Delta V)^2 \\
\Delta V &= -V_{as1} \pm \sqrt{V_{as1}^2 + \frac{\Delta T}{K}}
\end{aligned}$$

This formula is only valid if $\Delta T = 0$ implies $\Delta V = 0$, so the + solution is required.

$$\Delta V = -V_{as1} + \sqrt{V_{as1}^2 + \frac{\Delta T}{K}} \quad (2.24)$$

Now we consider adjusting the voltages of both IMs. IM 1 has applied line-neutral rms voltage $V_{as} + (\Delta V/2)$ and load torque $T + (\Delta T/2)$, while IM 2 has applied line-neutral rms voltage $V_{as} - (\Delta V/2)$ and load torque $T - (\Delta T/2)$.

$$\begin{aligned}
T + (\Delta T/2) &= K(V_{as} + (\Delta V/2))^2, & T - (\Delta T/2) &= K(V_{as} - (\Delta V/2))^2 \\
(K(V_{as} - (\Delta V/2))^2 + (\Delta T/2)) + \Delta T/2 &= K(V_{as} + (\Delta V/2))^2 \\
\Delta T &= K[(V_{as} + (\Delta V/2))^2 - (V_{as} - (\Delta V/2))^2] \\
\Delta T &= K[V_{as}^2 + V_{as}\Delta V + (\Delta V)^2/4 - (V_{as}^2 - V_{as}\Delta V + (\Delta V)^2/4)] \\
\Delta T &= 2KV_{as}\Delta V \\
\Delta V &= \frac{\Delta T}{2KV_{as}} \quad (2.25)
\end{aligned}$$

In both cases, the required rms voltage adjustment ΔV increases with the torque difference ΔT . Since K is approximately proportional to s , which increases with higher applied load torque, the required ΔV is lower when the base torque T is higher. Additionally, since the sign of K matches the sign of s , the required polarity of the applied voltage adjustment is reversed in generator mode. (Regardless of mode, the IM with the greater applied load torque magnitude must receive a higher voltage to synchronize.)

In this thesis, PI feedback control is used to achieve speed and position synchronization using the auxiliary converter method. This uses the same definition of primary and secondary motors and angular position used for the external resistor method's PI feedback control (2.18, 2.19). The auxiliary converters all command $v_{d2}^* = 0$ (so the commanded voltage is entirely in the q component) and, for the converters for secondary motor lines,

$$v_{q2,s_i}^* = K_{P,s_i} \delta\theta_{rm,s_i} + K_{I,s_i} \int \delta\theta_{rm,s_i} dt. \quad (2.26)$$

The primary motor line's auxiliary converter is disabled, with all high switches open and all low switches closed so that the converter acts like a short.

2.4 Diagram of Auxiliary Converter and Transformer

Figure 2.6 shows the full qd equivalent circuits for one motor of the auxiliary converter methodology, treating converter outputs as voltage sources.

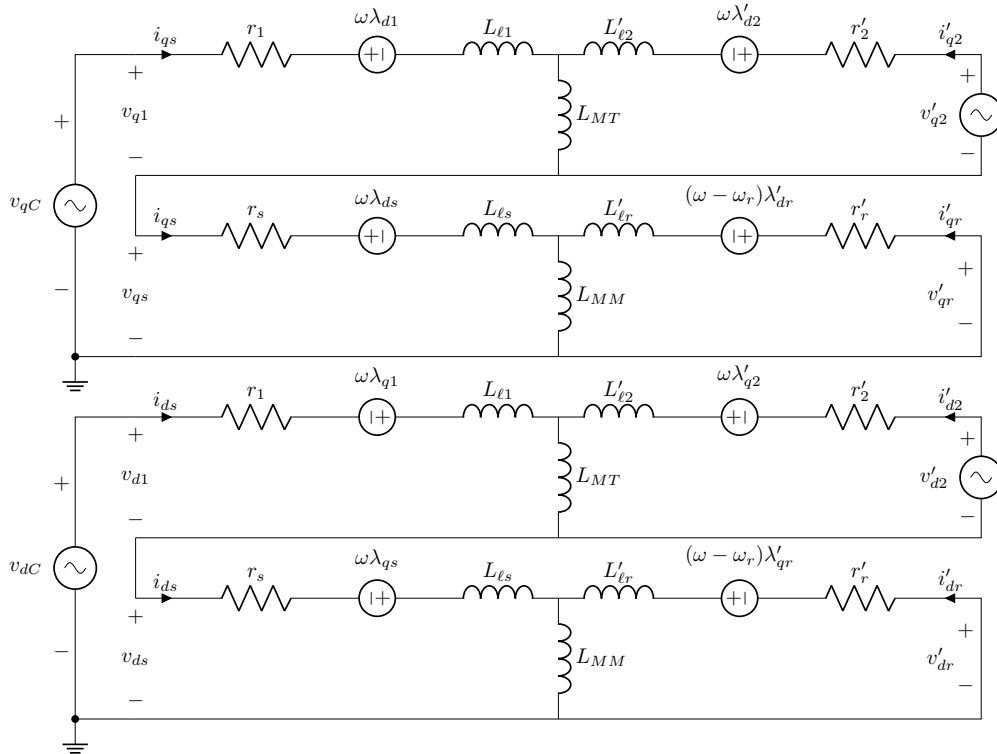


Figure 2.6: Auxiliary converter methodology equivalent $qd0$ circuits for one motor.

In Figure 2.6, v_{qC}, v_{dC} are the components of the voltage supplied by the central converter, $v_{q1}, v_{d1}, i_{qs}, i_{ds}, \lambda_{q1}, \lambda_{d1}$ are the components of the transformer primary voltage and flux linkage, $v'_{q2}, v'_{d2}, i'_{q2}, i'_{d2}, \lambda'_{q2}, \lambda'_{d2}$ are the components of the referred transformer secondary output voltage, current, and flux linkage (tied to the auxiliary converter), $v_{qs}, v_{ds}, i_{qs}, i_{ds}, \lambda_{qs}, \lambda_{ds}$ are the components of the stator winding voltage, current, and flux linkage, and $v'_{qr}, v'_{dr}, i'_{qr}, i'_{dr}, \lambda'_{qr}, \lambda'_{dr}$ are the components of the rotor winding voltage (zero for a squirrel-cage rotor), current, and flux linkage. Note that the transformer primary and the stator winding share the same current.

The transformer has primary resistance and leakage inductance $r_1, L_{\ell 1}$, referred secondary resistance and leakage inductance $r'_2, L'_{\ell 2}$, and three-phase magnetizing inductance L_{MT} . The motor has stator resistance and leakage inductance $r_s, L_{\ell s}$, referred rotor resistance and leakage inductance $r'_r, L'_{\ell r}$, and three-phase magnetizing inductance L_{MM} .

Figure 2.7 is a block diagram of the control structure of the auxiliary converter methodology.

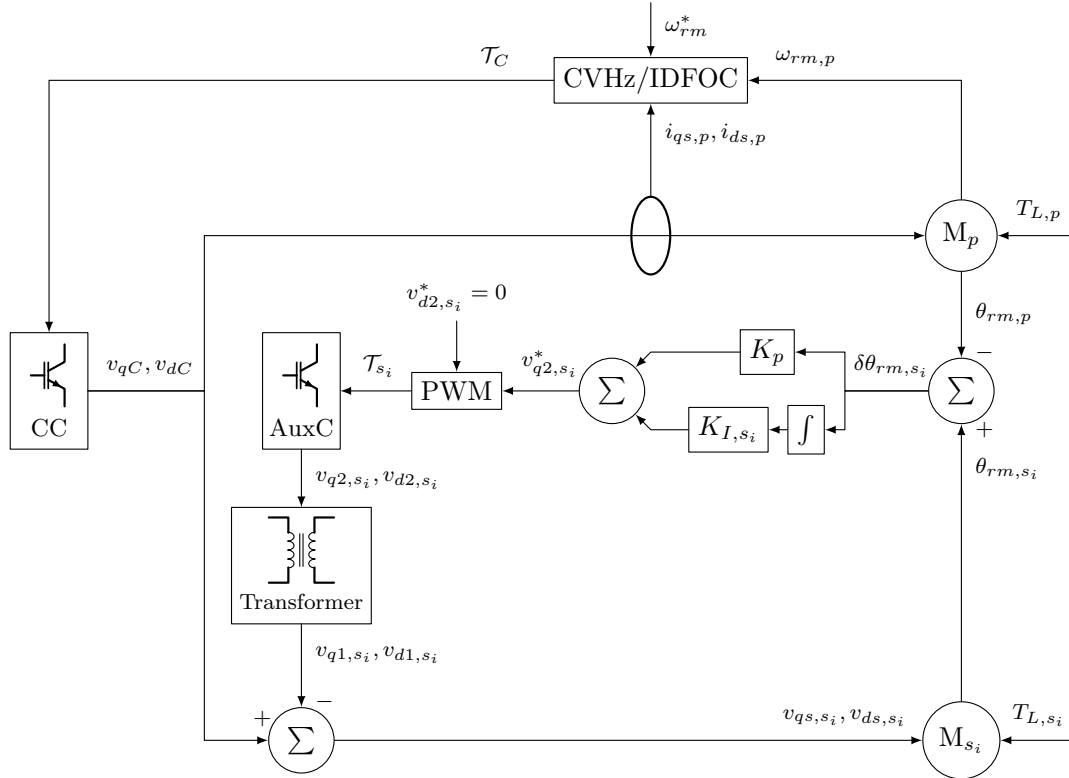


Figure 2.7: Auxiliary converter methodology block diagram, showing control for the primary motor and one secondary motor.

In Figure 2.7, the central converter is labeled as “CC” and the auxiliary converter is labeled as “AuxC”. The control strategy for the central converter and primary motor is represented as one block, as it can be changed without affecting the structure of the synchronization control. In a realistic implementation, a transformer and auxiliary converter would be included on every motor line, though the primary motor’s auxiliary converter would be disabled (acting as a short) until a new primary designation is assigned.

Chapter 3

Numerical Case Studies

This chapter describes and presents results of transient simulations used as case studies to validate the auxiliary converter methodology and compare its performance and efficiency to that of the external stator resistance methodology. Time-domain simulations were performed in MATLAB®, release R2022a. The central and auxiliary converters were modeled as six-switch, four-quadrant inverters with ideal switch states (fully on or fully off). The external resistor circuits were also modeled as switches in the same manner, neglecting snubbing. All case studies presented here include three actuation lines in the CCMM with identical 15 hp IMs used as drive motors. Motor 1 was selected to be the primary motor, with motors 2 and 3 acting as secondary motors. Parameters for the motors and control systems used in these studies are listed in Appendix A. Each case study explores the effect of varying the transformer turns ratio and auxiliary converter bus voltage.

For the auxiliary converter methodology, each motor line with transformer and auxiliary converter included was simulated using the following set of eight differential equations derived from Figure 2.6 and (2.13-2.16), where ω is an arbitrary reference frame speed:

$$p\lambda_{qC} = v_{qC} - (r_s + r_1)i_{qs} - \omega\lambda_{dC} \quad (3.1a)$$

$$p\lambda_{dC} = v_{dC} - (r_s + r_1)i_{ds} + \omega\lambda_{qC} \quad (3.1b)$$

$$p\lambda'_{qr} = v'_{qr} - r'_r i'_{qr} - (\omega - \omega_r)\lambda'_{dr} \quad (3.1c)$$

$$p\lambda'_{dr} = v'_{dr} - r'_r i'_{dr} + (\omega - \omega_r)\lambda'_{qr} \quad (3.1d)$$

$$p\lambda'_{q2} = v'_{q2} - r'_2 i'_{q2} - \omega\lambda'_{d2} \quad (3.1e)$$

$$p\lambda'_{d2} = v'_{d2} - r'_2 i'_{d2} + \omega\lambda'_{q2} \quad (3.1f)$$

$$p\omega_r = \frac{1}{J} \left(\frac{P}{2}(T_e - T_L) - B_m\omega_r \right) \quad (3.1g)$$

$$p\theta_r = \omega_r \quad (3.1h)$$

Using these differential equations requires the inductance matrix converting between current and flux linkage:

$$\begin{bmatrix} \lambda_{xC} \\ \lambda'_{xr} \\ \lambda'_{x2} \end{bmatrix} = \begin{bmatrix} L_{\ell s} + L_{\ell 1} + L_{MM} + L_{MT} & L_{MM} & L_{MT} \\ & L_{MM} & 0 \\ & L_{MT} & 0 \end{bmatrix} \begin{bmatrix} i_{xs} \\ i'_{xr} \\ i'_{x2} \end{bmatrix}. \quad (3.2)$$

The secondary resistance and leakage inductance of the transformers ($r_2, L_{\ell 2}$) are assumed to scale linearly with the secondary-to-primary turns ratio N_2/N_1 . Therefore, by the impedance referral relation $Z'_2 = (N_1/N_2)^2 Z_2$, the referred secondary resistance and leakage inductance of the transformers ($r'_2, L'_{\ell 2}$) are assumed to scale linearly with the primary-to-secondary turns ratio N_1/N_2 (i.e., the inverse of the secondary-to-primary turns ratio).

Efficiency calculations presented here consider electrical energy entering the converters, electrical energy supplied to the IMs, and mechanical energy transferred to the load. The energy required to generate the control signals is neglected, as it is relatively small and dependent on specifics of implementation.

3.1 Case Study 1a: Central Voltage Control, Ideal Switching

In this case study, all IMs were driven by the central converter using the CVHz strategy described in [1] in combination with both position synchronization schemes detailed in Chapter 2. Converter switching states for implementing the voltage control were generated using sine-triangle modulation with third harmonic injection (see [5] for implementation). All switches were modeled as ideal. In this study, the commanded speed of $\omega_{rm}^* = 1800$ rpm was first applied at $t = 0$ s, with zero torque load applied to each motor. After reaching steady-state commanded rotor speed, at $t = 2$ s, torque loads of $T_{L,1} = 1.0T_{\text{rated}}$, $T_{L,2} = 0.8T_{\text{rated}}$, and $T_{L,3} = 0.7T_{\text{rated}}$ were then applied, where $T_{\text{rated}} = 61.1$ Nm was the torque rating of the (identical) IMs.

Under these conditions, the nominal voltage applied to motor 1 is $V_{as1} = \frac{240}{\sqrt{3}}$ V and the slip of motor 1 was observed to approach a steady-state value of $s \approx 0.0374$. Therefore, the proportion-

ality constant $K \approx 0.0033$ and the voltage adjustments expected for steady-state synchronization are

$$\begin{aligned}\Delta V_2 &= -V_{as1,1} + \sqrt{V_{as1,1}^2 + \frac{-0.2T_{\text{rated}}}{K}} \approx -14.3 \text{ V}, \\ \Delta V_3 &= -V_{as1,1} + \sqrt{V_{as1,1}^2 + \frac{-0.3T_{\text{rated}}}{K}} \approx -22.1 \text{ V}.\end{aligned}\tag{3.3}$$

Transient responses are shown in Figures 3.1-3.6.

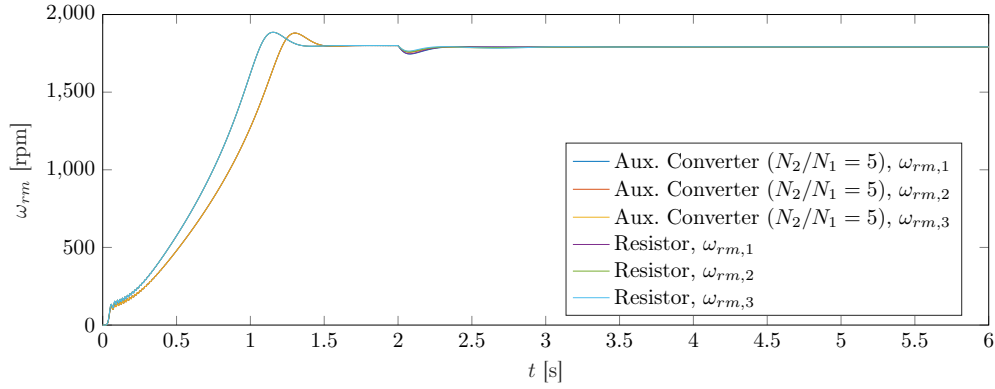


Figure 3.1: Mechanical rotor velocities including start-up, CVHz control, ideal switching.

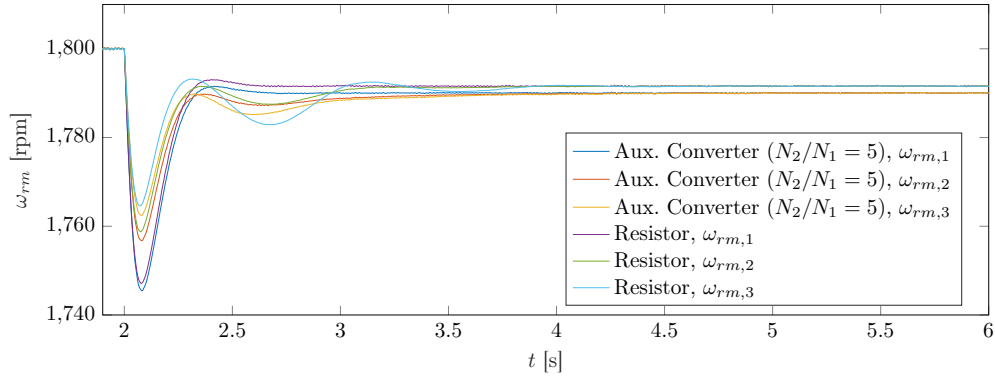


Figure 3.2: Mechanical rotor velocities (zoomed in), CVHz control, ideal switching.

Figure 3.1 shows mechanical rotor velocity for all motors including the initial start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$. Figure 3.2 provides a zoomed-in view after the torque loads are applied. The auxiliary converter circuit slows down the acceleration a bit, tak-

ing about 0.2 s longer to stabilize with this methodology. Then, when load is applied, the speed varies slightly more with the auxiliary converter methodology compared to the external resistance methodology, settling with a slightly higher slip. However, more ringing is seen in the speed response with the external resistance methodology compared to the auxiliary converter methodology.

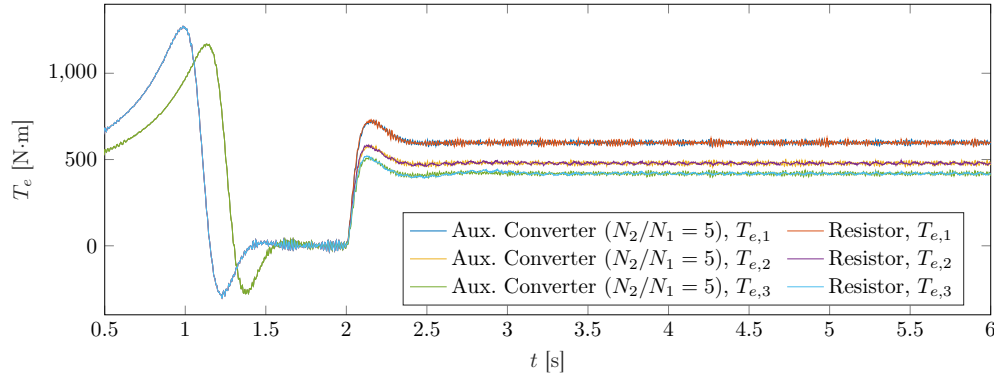


Figure 3.3: Electromagnetic torques from 0.5 s, CVHz control, ideal switching.

Figure 3.3 shows electromagnetic torques for all motors including part of the start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$. The delay introduced by the auxiliary converter circuit is visible, but after the start-up period ends, the electromagnetic torque for each motor is nearly independent of the methodology. The torque responses for different motors separate at $t = 2$ s because of the different applied load torques.

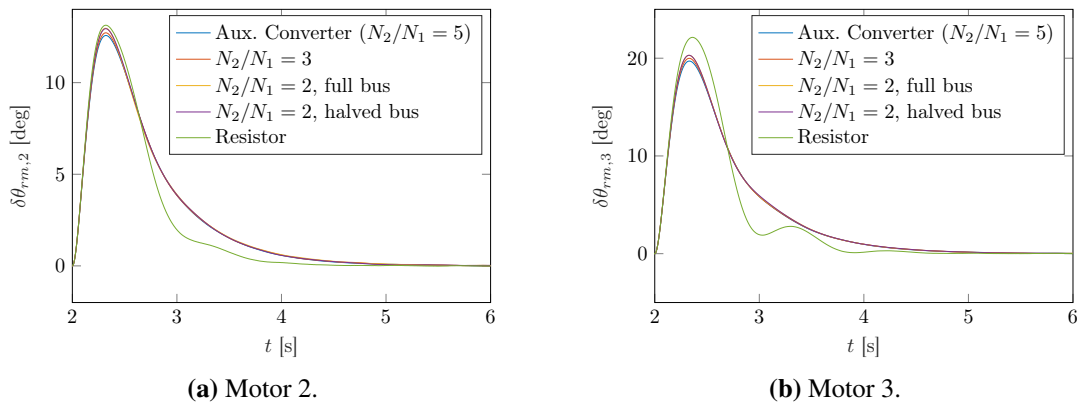


Figure 3.4: Individual angle errors, CVHz control, ideal switching.

Individual rotor angle errors for the secondary motors, referenced from the primary motor, are shown in Figure 3.4. Compared to the external resistance methodology, the auxiliary converter methodology results in less overshoot and a smoother settling response, but a slightly longer settling time. The response is marginally better with a higher secondary-to-primary turns ratio for the transformer. Reducing the auxiliary converter bus voltage to half that of the central converter bus voltage has negligible impact on the response for $N_2/N_1 = 2$.

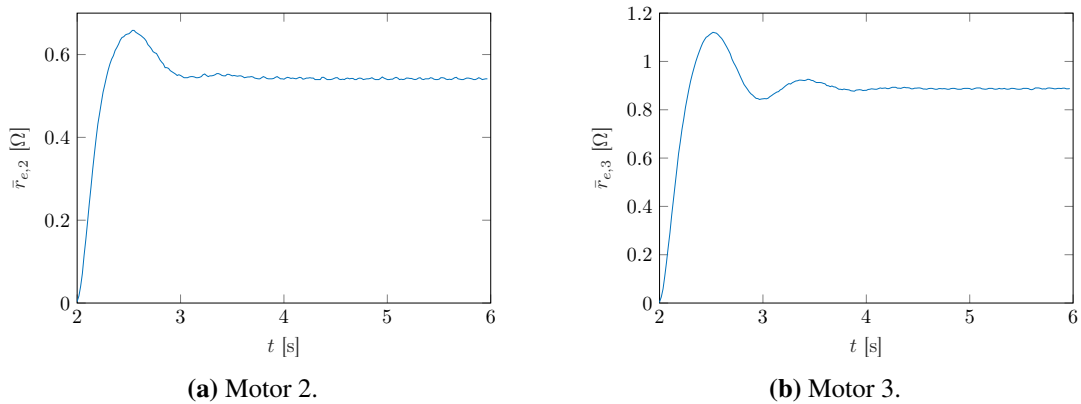


Figure 3.5: Fast-average external resistance, CVHz control, ideal switching.

Figure 3.5 shows the fast-average external resistance (phase a) applied to each secondary motor line. Ringing is visible in the response, and a higher resistance is required when the load torque difference is higher.

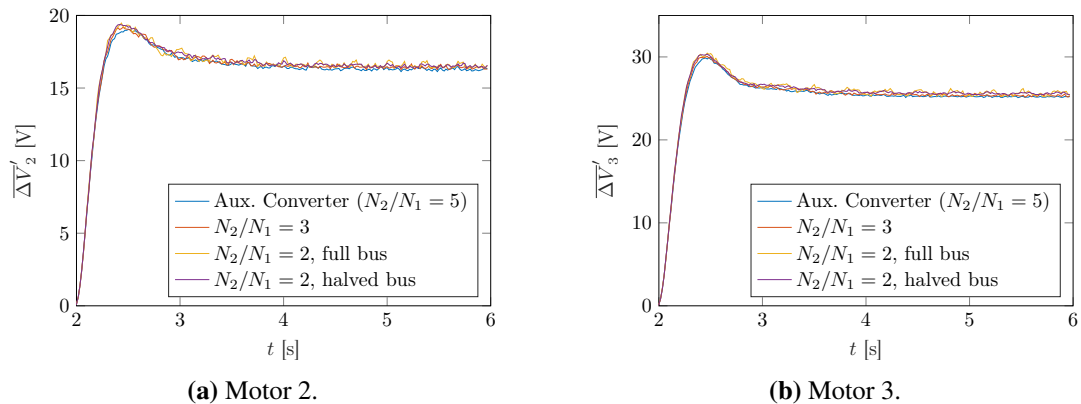


Figure 3.6: Fast-average auxiliary converter voltage, CVHz control, ideal switching.

Figure 3.6 shows the fast-average rms referred output voltage of the auxiliary converter for each secondary motor line. Less ringing occurs compared to the external resistance methodology, but a higher voltage is required when the load torque difference is higher. The output voltage is marginally lower for higher values of N_2/N_1 , but the auxiliary converter bus voltage has no noticeable effect here. Note that the voltages here approach 16.5 V for motor 2 and 25.5 V for motor 3, about 15% higher than the predicted steady-state values.

Table 3.1: Efficiency analysis over first 4 seconds of operation, CVHz control, ideal switching.

Description	Symbol	Ext. Resistor	Auxiliary Converter			
			5	3	2	2
Turns ratio	N_2/N_1	–	5	3	2	2
Auxiliary bus voltage	v_{xdc}	–	339 V	339 V	339 V	170 V
Central conv. energy in	E_{in}	137.8 kJ	134.9 kJ	134.9 kJ	134.9 kJ	135.0 kJ
Energy supp. to motors	E_{mot}	123.2 kJ	123.3 kJ	123.3 kJ	123.3 kJ	123.3 kJ
Mech. energy output	E_{T_m}	117.1 kJ	117.0 kJ	117.0 kJ	117.0 kJ	117.0 kJ
Electrical efficiency	η_e	89.40%	91.40%	91.37%	91.33%	91.31%
Mechanical efficiency	η_m	84.98%	86.78%	86.72%	86.66%	86.65%

Table 3.1 displays the energy flow measured at 3 points in the system for this case study for the first 4 seconds after load torque is applied: the electrical energy entering the central converter, the electrical energy supplied to the motors, and the mechanical energy supplied to the load by the motors. These are used to calculate values for electrical efficiency $\eta_e = E_{mot}/E_{in}$ and mechanical efficiency $\eta_m = E_{T_m}/E_{in}$. By both metrics, the auxiliary converter methodology is more energy efficient, as the central converter draws less energy while the energy output remains approximately the same. Efficiency is slightly improved by a higher secondary-to-primary turns ratio, while the auxiliary converter bus voltage appears to have no significant effect.

3.2 Case Study 1b: Central Voltage Control, Lossy Switching

In this modification of the previous case study, all switches were instead modeled with a voltage drop of 5 V in the direction of current flow, as described in [5]. The setup is otherwise identical. Transient responses are shown in Figures 3.7-3.12.

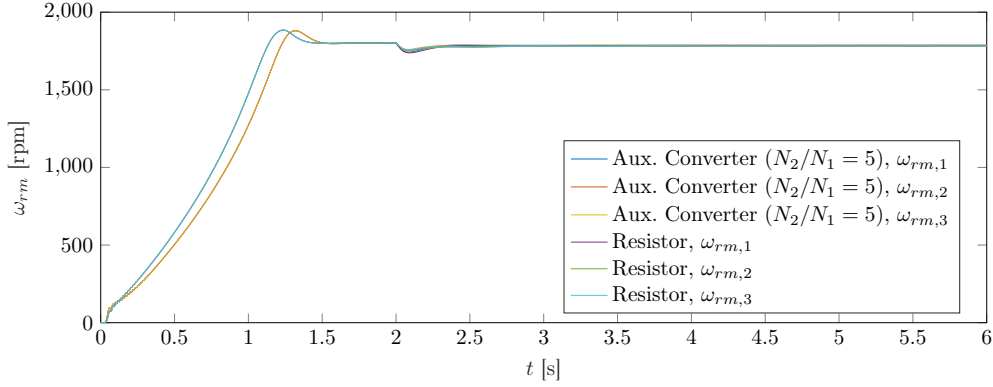


Figure 3.7: Mechanical rotor velocities including start-up, CVHz control, lossy switching.

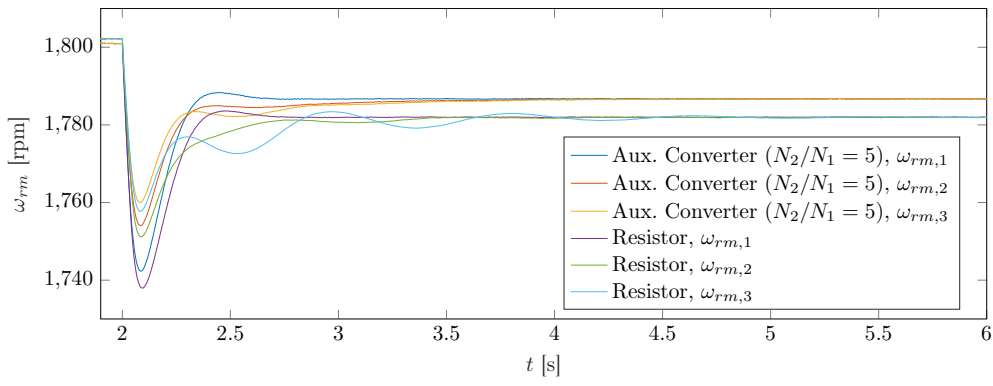


Figure 3.8: Mechanical rotor velocities (zoomed in), CVHz control, lossy switching.

Figure 3.7 shows mechanical rotor velocity for all motors including the initial start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$, now with switching losses included. Figure 3.8 provides a zoomed-in view after the torque loads are applied. The start-up period is slightly longer in this case, and the slip is noticeably higher with a wider gap between the speeds for the different methodologies. The external resistance methodology now has more ringing in its response.

Figure 3.9 shows electromagnetic torques for all motors including part of the start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$. The peak seen at about $t = 1$ s is lower, but the torque responses are not significantly changed beyond that point.

Individual rotor angle errors for the secondary motors, referenced from the primary motor, are shown in Figure 3.10. The introduction of switching losses increases overshoot and lengthens the

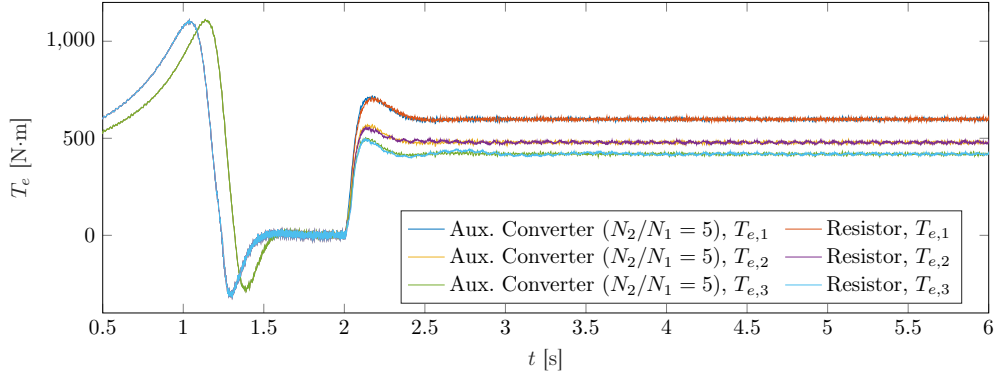


Figure 3.9: Electromagnetic torques from 0.5 s, CVHz control, lossy switching.

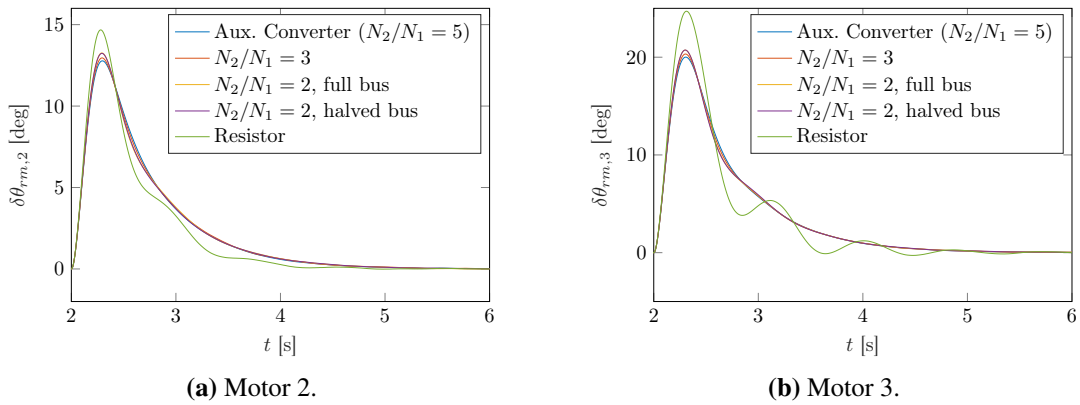
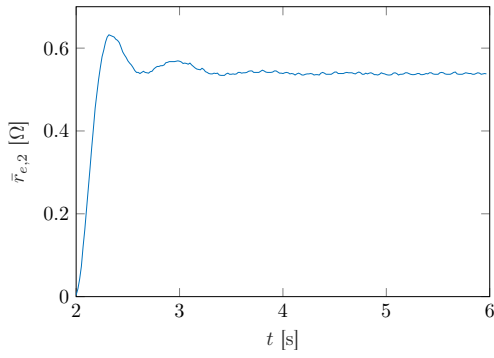


Figure 3.10: Individual angle errors, CVHz control, lossy switching.

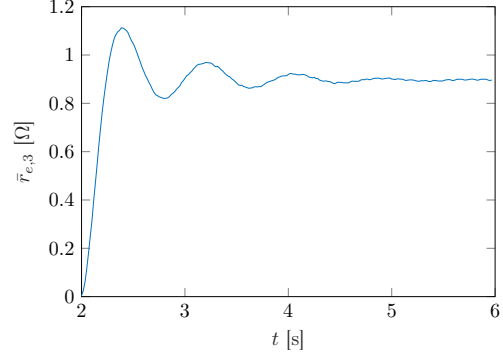
settling time, though this effect is more pronounced for the external resistance method than for the auxiliary converter method. The external resistance response also experiences more ringing in this case.

Figure 3.11 shows the fast-average external resistance (phase a) applied to each secondary motor line. The rise time is faster and ringing is more pronounced with the addition of switching losses, but peak and steady-state external resistance do not appear to have been affected significantly.

Figure 3.12 shows the fast-average rms referred output voltage of the auxiliary converter for each secondary motor line. As before, less ringing occurs compared to the external resistance methodology, but the voltage is now significantly dependent on the choice of turns ratio, with a higher secondary-to-primary turns ratio resulting in a lower voltage.

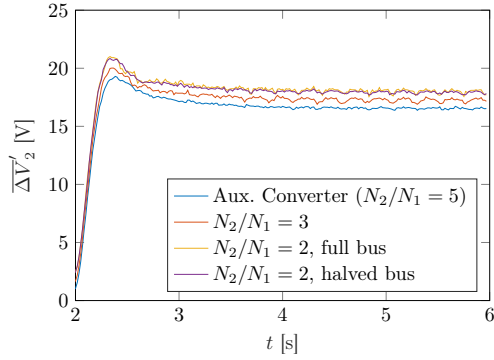


(a) Motor 2.

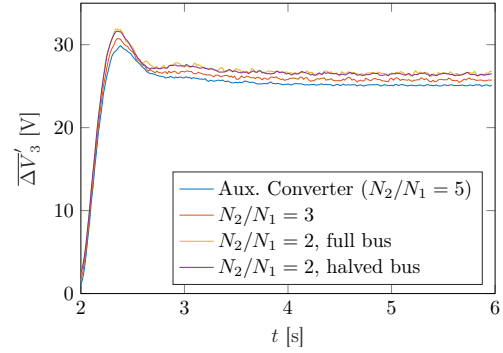


(b) Motor 3.

Figure 3.11: Fast-average external resistance, CVHz control, lossy switching.



(a) Motor 2.



(b) Motor 3.

Figure 3.12: Fast-average auxiliary converter voltage, CVHz control, lossy switching.

Table 3.2: Efficiency analysis over first 4 seconds of operation, CVHz control, lossy switching.

Description	Symbol	Ext. Resistor	Auxiliary Converter			
			5	3	2	2
Turns ratio	N_2/N_1	–	5	3	2	2
Auxiliary bus voltage	v_{xdc}	–	339 V	339 V	339 V	170 V
Central conv. energy in	E_{in}	154.7 kJ	141.5 kJ	142.3 kJ	143.2 kJ	143.3 kJ
Energy supp. to motors	E_{mot}	123.8 kJ	123.5 kJ	123.5 kJ	123.5 kJ	123.5 kJ
Mech. energy output	E_{T_m}	116.5 kJ	116.8 kJ	116.7 kJ	116.7 kJ	116.7 kJ
Electrical efficiency	η_e	80.02%	87.29%	86.81%	86.23%	86.22%
Mechanical efficiency	η_m	75.32%	82.57%	82.06%	81.43%	81.43%

Table 3.2 displays the energy flow measured at 3 points in the system for this case study for the first 4 seconds after load torque is applied: the electrical energy entering the central converter, the electrical energy supplied to the motors, and the mechanical energy supplied to the load by the

motors. These are used to calculate values for electrical efficiency $\eta_e = E_{mot}/E_{in}$ and mechanical efficiency $\eta_m = E_{T_m}/E_{in}$. Efficiency is significantly lower with the addition of switching losses, but the gap between the methodologies has widened. The efficiency improvement from increasing the secondary-to-primary turns ratio is more significant here, but the effect of changing the auxiliary converter bus voltage is still negligible.

3.3 Case Study 2a: Central Current Control, Ideal Switching

In this case study, all IMs were driven by the central converter using the IDFOC strategy described in [1] in combination with both position synchronization schemes detailed in Chapter 2. Converter switching states for implementing the voltage control were again generated using sine-triangle modulation with third harmonic injection. All switches were modeled as ideal. The same speed commands and torque loading conditions used in the previous study were applied here.

Under these conditions, the nominal voltage applied to motor 1 is $V_{as1} = \frac{240}{\sqrt{3}}$ V and the slip of motor 1 was observed to approach a steady-state value of $s \approx 0.0333$. Therefore, the proportionality constant $K \approx 0.003$ and the voltage adjustments expected for steady-state synchronization are

$$\begin{aligned}\Delta V_2 &= -V_{as1,1} + \sqrt{V_{as1,1}^2 + \frac{-0.2T_{rated}}{K}} \approx -15.9 \text{ V}, \\ \Delta V_3 &= -V_{as1,1} + \sqrt{V_{as1,1}^2 + \frac{-0.3T_{rated}}{K}} \approx -24.7 \text{ V}.\end{aligned}\tag{3.4}$$

Transient responses are shown in Figures 3.13-3.18.

Figure 3.13 shows mechanical rotor velocity for all motors including the initial start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$. Figure 3.14 provides a zoomed-in view after the torque loads are applied. All motors accelerate at a constant rate starting at about 0.6 s and stabilize at the same time with this methodology. Then, when load is applied, the speed varies marginally more with the auxiliary converter methodology compared to the external resistance methodology, but all motors settle back at 1800 rpm in both cases. As in the CVHz control case study, more

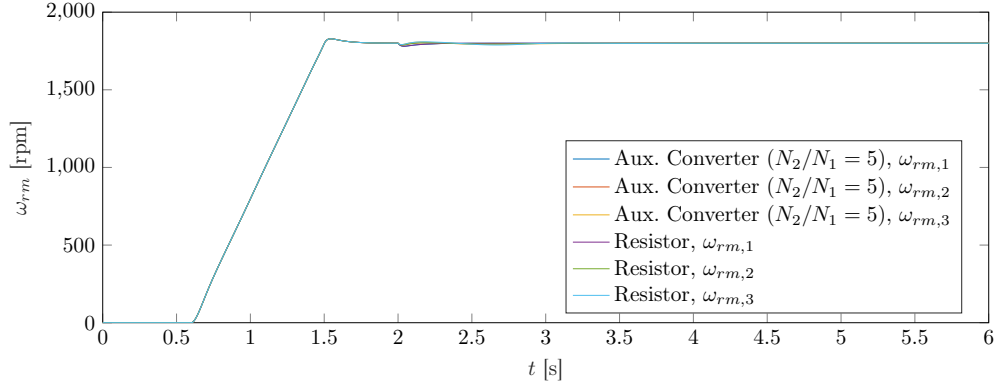


Figure 3.13: Mechanical rotor velocities including start-up, IDFOC control, ideal switching.

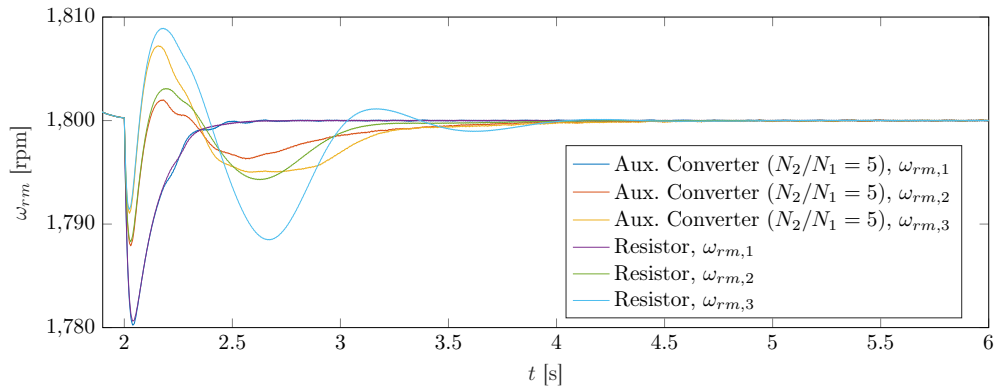


Figure 3.14: Mechanical rotor velocities (zoomed in), IDFOC control, ideal switching.

ringing is seen in the speed response with the external resistance methodology compared to the auxiliary converter methodology.

Figure 3.15 shows electromagnetic torques for all motors including part of the start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$. The electromagnetic torque for each motor is virtually independent of the methodology until load is applied. The torque responses for different motors separate at $t = 2$ s because of the different applied load torques, with the ripple patterns looking slightly different. More torque ripple is present with IDFOC control than with CVHz control.

Individual rotor angle errors for the secondary motors, referenced from the primary motor, are shown in Figure 3.16. Compared to the external resistance methodology, the auxiliary converter

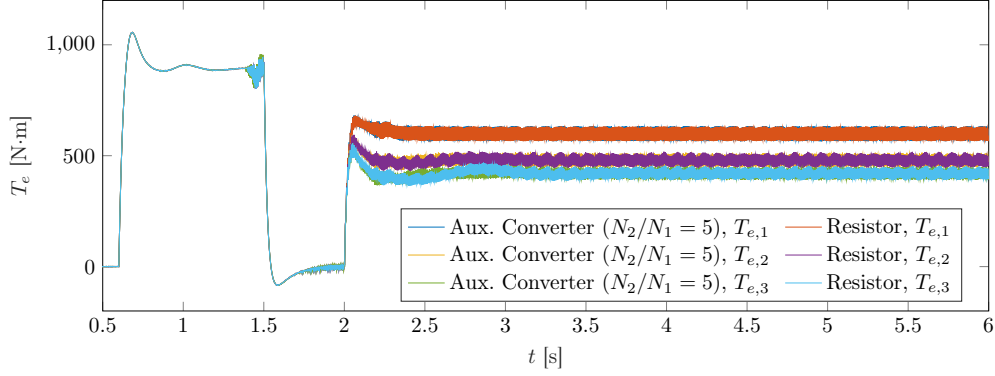


Figure 3.15: Electromagnetic torques from 0.5 s, IDFOC control, ideal switching.

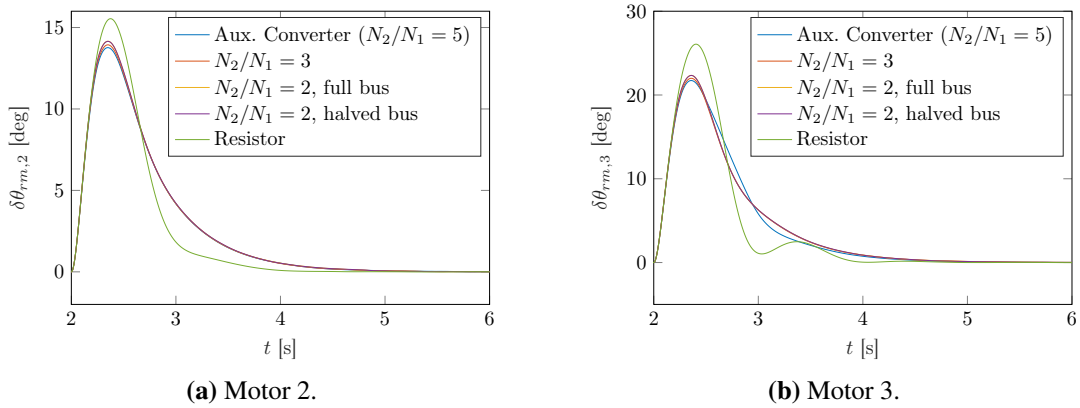


Figure 3.16: Individual angle errors, IDFOC control, ideal switching.

methodology results in less overshoot and a smoother settling response, but a slightly longer settling time. The response is marginally better with a higher secondary-to-primary turns ratio for the transformer, though the shape is a bit different for motor 3 in the $N_2/N_1 = 5$ case (due to overmodulation). Reducing the auxiliary converter bus voltage to half that of the central converter bus voltage has negligible impact on the response for $N_2/N_1 = 2$.

Figure 3.17 shows the fast-average external resistance (phase a) applied to each secondary motor line. Some ringing is visible in the response, and a higher resistance is required when the load torque difference is higher. The required resistance is slightly higher than in the CVHz control case study.

Figure 3.18 shows the fast-average rms referred output voltage of the auxiliary converter for each secondary motor line. Less ringing occurs compared to the external resistance methodology,

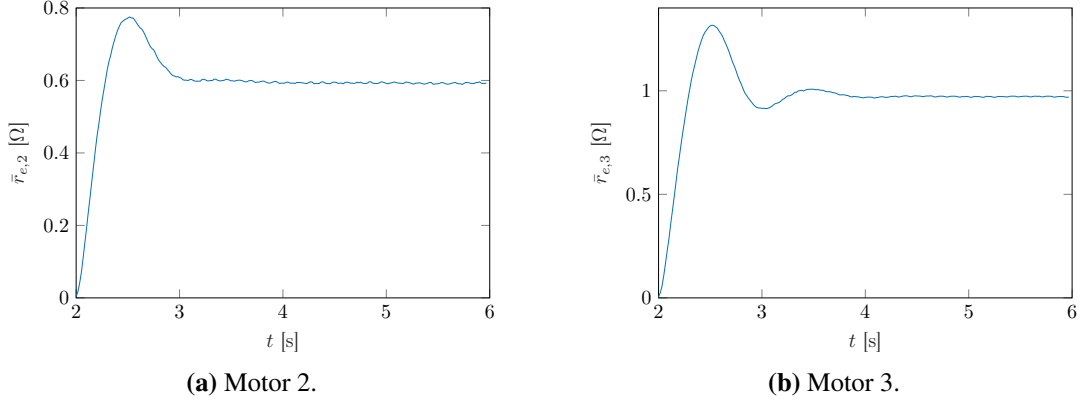


Figure 3.17: Fast-average external resistance, IDFOC control, ideal switching.

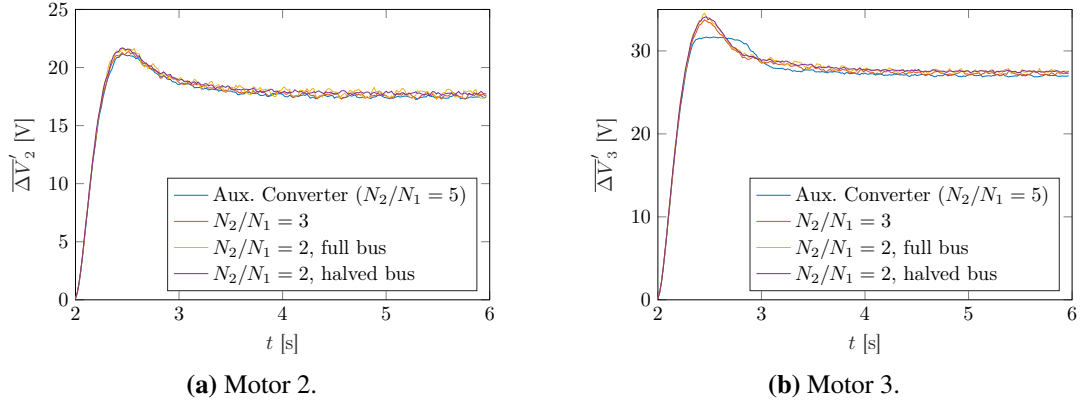


Figure 3.18: Fast-average auxiliary converter voltage, IDFOC control, ideal switching.

but a higher voltage is required when the load torque difference is higher, and this voltage is higher than in the CVHz case study. The output voltage is marginally lower for higher values of N_2/N_1 , but the auxiliary converter bus voltage has no noticeable effect here. In the $N_2/N_1 = 5$ case, the peak commanded voltage exceeds the sine-triangle PWM maximum, leading to overmodulation: the output voltage still increases with the command, but fails to track it, leading to a shorter but wider peak. Note that the voltages here approach 17.5 V for motor 2 and 27.0 V for motor 3, about 10% higher than the predicted steady-state values.

Table 3.3 displays the energy flow measured at 3 points in the system for this case study for the first 4 seconds after load torque is applied: the electrical energy entering the central converter, the electrical energy supplied to the motors, and the mechanical energy supplied to the load by the motors. These are used to calculate values for electrical efficiency $\eta_e = E_{mot}/E_{in}$ and mechanical

Table 3.3: Efficiency analysis over first 4 seconds of operation, IDFOC control, ideal switching.

Description	Symbol	Ext. Resistor	Auxiliary Converter			
			5	3	2	2
Turns ratio	N_2/N_1	–	5	3	2	2
Auxiliary bus voltage	v_{xdc}	–	339 V	339 V	339 V	170 V
Central conv. energy in	E_{in}	138.5 kJ	135.5 kJ	135.8 kJ	137.5 kJ	137.2 kJ
Energy supp. to motors	E_{mot}	123.8 kJ	122.5 kJ	122.8 kJ	124.3 kJ	124.0 kJ
Mech. energy output	E_{T_m}	117.7 kJ	117.7 kJ	117.7 kJ	117.7 kJ	117.7 kJ
Electrical efficiency	η_e	89.36%	90.41%	90.38%	90.41%	90.37%
Mechanical efficiency	η_m	84.99%	86.91%	86.67%	85.63%	85.81%

efficiency $\eta_m = E_{T_m}/E_{in}$. By both metrics, the auxiliary converter methodology is more energy efficient, as the central converter draws less energy while the energy output remains approximately the same. Efficiency is slightly improved by a higher secondary-to-primary turns ratio, while halving the auxiliary converter bus voltage appears to slightly reduce electrical efficiency while lightly improving mechanical efficiency.

3.4 Case Study 2b: Central Current Control, Lossy Switching

In this modification of the previous case study, all switches were instead modeled with a voltage drop of 5 V in the direction of current flow, as described in [5]. The setup is otherwise identical. Transient responses are shown in Figures 3.19-3.24.

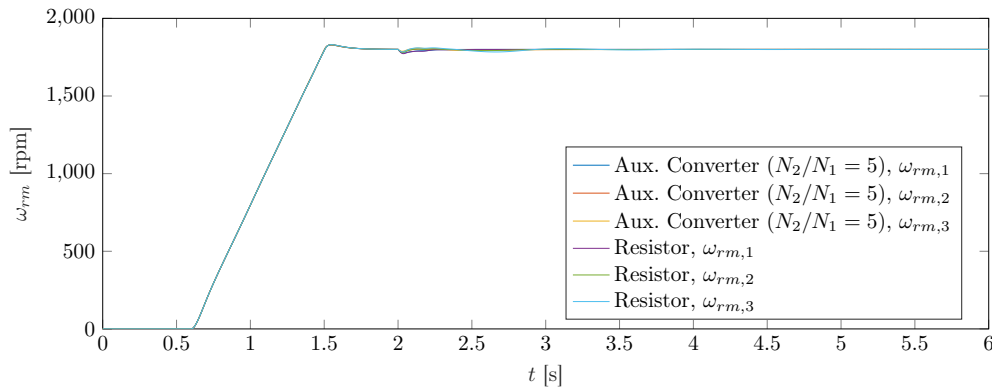


Figure 3.19: Mechanical rotor velocities including start-up, IDFOC control, lossy switching.

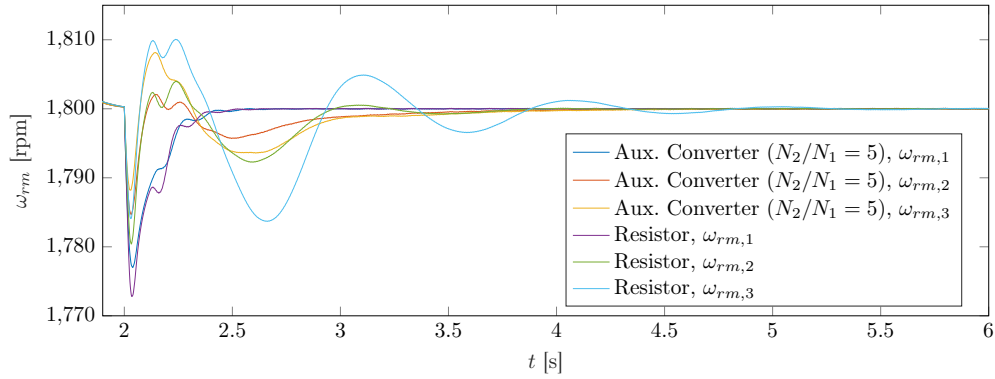


Figure 3.20: Mechanical rotor velocities (zoomed in), IDFOC control, lossy switching.

Figure 3.19 shows mechanical rotor velocity for all motors including the initial start-up period, using the external resistance methodology and the auxiliary converter methodology with all converters at the same bus voltage and $N_2/N_1 = 5$. Figure 3.20 provides a zoomed-in view after the torque loads are applied. The speed curves do not appear to be significantly affected by switching losses before load is applied. After load is applied, the speed experiences more ringing than without switching losses and the shape appears more chaotic at first, but all motors still settle back at 1800 rpm in both cases. More ringing is still seen in the speed response with the external resistance methodology compared to the auxiliary converter methodology.

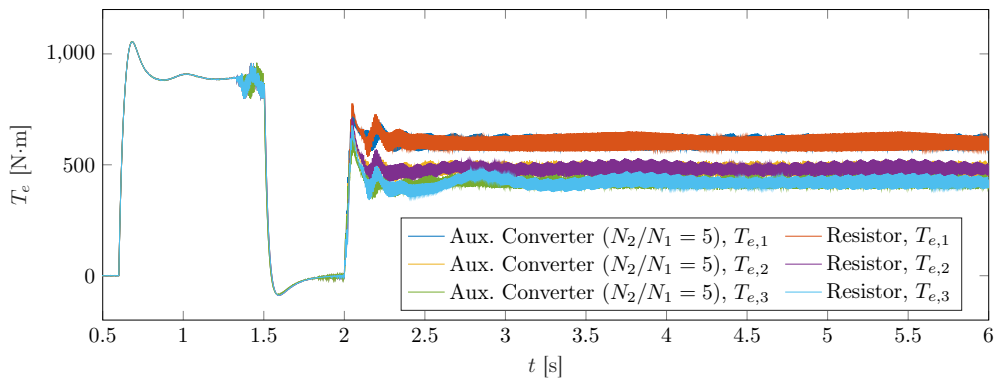


Figure 3.21: Electromagnetic torques from 0.5 s, IDFOC control, lossy switching.

Figure 3.21 shows electromagnetic torques for all motors including part of the start-up period, using the external resistance methodology and the auxiliary converter methodology with all con-

verters at the same bus voltage and $N_2/N_1 = 5$. A slight difference between the methodologies is now seen at about $t = 1.5$ s, before load is applied. The torque responses for different motors separate at $t = 2$ s because of the different applied load torques, with more ringing seen here than without switching losses.

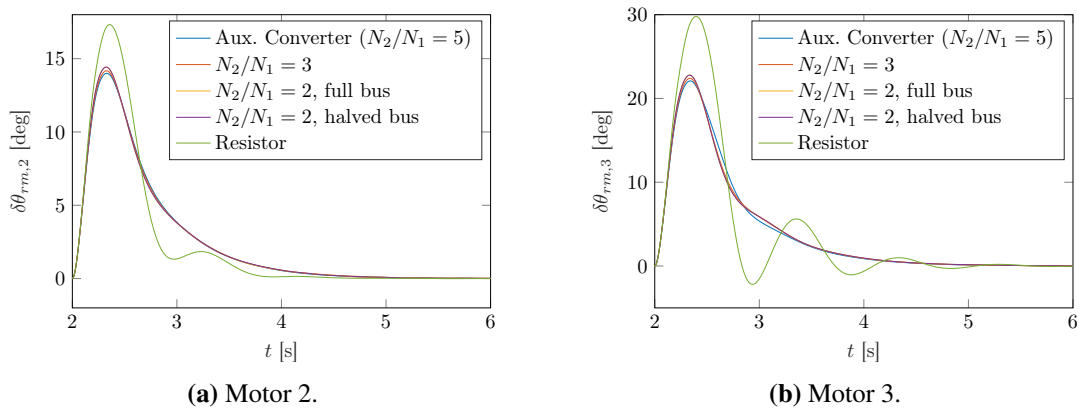


Figure 3.22: Individual angle errors, IDFOC control, lossy switching.

Individual rotor angle errors for the secondary motors, referenced from the primary motor, are shown in Figure 3.22. Compared to the external resistance methodology, the auxiliary converter methodology results in less overshoot and a smoother settling response, but a slightly longer settling time. The response is very similar to that without switching losses, except the peak angle errors are higher and more ringing is seen in the plots for the external resistance methodology.

Figure 3.23 shows the fast-average external resistance (phase a) applied to each secondary motor line. More ringing is visible than without switching losses, and the peak and steady-state resistance are higher, with the resistance for motor 3 almost reaching the maximum of 1.5Ω .

Figure 3.24 shows the fast-average rms referred output voltage of the auxiliary converter for each secondary motor line. Like with the CVHz case study, the addition of switching losses makes the voltage significantly dependent on the choice of turns ratio. As in the simulation without switching losses, overmodulation is seen for the $N_2/N_1 = 5$ case.

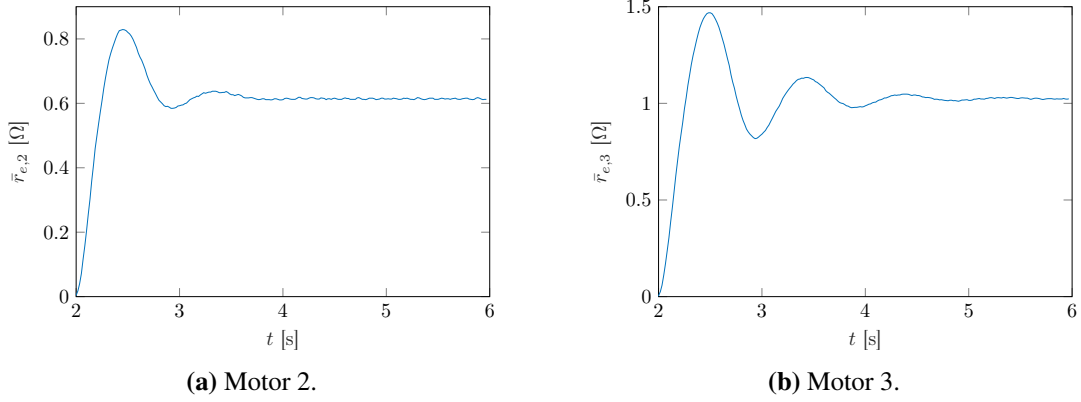


Figure 3.23: Fast-average external resistance, IDFOC control, lossy switching.

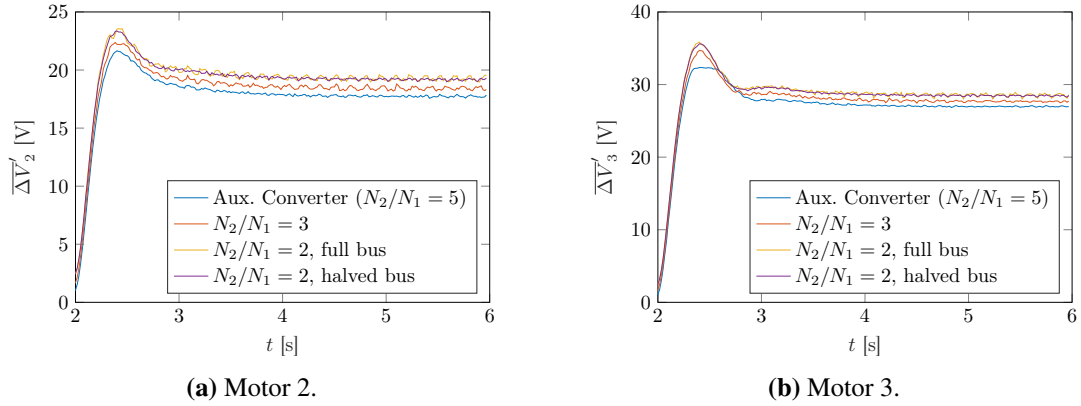


Figure 3.24: Fast-average auxiliary converter voltage, IDFOC control, lossy switching.

Table 3.4: Efficiency analysis over first 4 seconds of operation, IDFOC control, lossy switching.

Description	Symbol	Ext. Resistor	Auxiliary Converter			
			5	3	2	2
Turns ratio	N_2/N_1	–	5	3	2	2
Auxiliary bus voltage	v_{xdc}	–	339 V	339 V	339 V	170 V
Central conv. energy in	E_{in}	153.8 kJ	142.2 kJ	143.6 kJ	145.9 kJ	145.1 kJ
Energy supp. to motors	E_{mot}	123.5 kJ	123.2 kJ	123.8 kJ	125.3 kJ	124.4 kJ
Mech. energy output	E_{T_m}	117.7 kJ	117.7 kJ	117.7 kJ	117.7 kJ	117.7 kJ
Electrical efficiency	η_e	80.34%	86.66%	86.26%	85.83%	85.73%
Mechanical efficiency	η_m	76.56%	82.78%	82.02%	80.67%	81.14%

Table 3.4 displays the energy flow measured at 3 points in the system for this case study for the first 4 seconds after load torque is applied: the electrical energy entering the central converter, the electrical energy supplied to the motors, and the mechanical energy supplied to the load by

the motors. These are used to calculate values for electrical efficiency $\eta_e = E_{mot}/E_{in}$ and mechanical efficiency $\eta_m = E_{T_m}/E_{in}$. As in the CVHz control case study, efficiency is significantly lowered by switching losses, but the gap between the methodologies has widened. The efficiency improvements from increasing the primary-to-secondary turns ratio and the mechanical efficiency improvement from lowering the auxiliary converter bus voltage are both more significant here.

3.5 Remarks

Overall, the auxiliary converter methodology was able to achieve synchronization performance similar to the external resistance methodology with less ringing and better energy efficiency. Performance could be further improved with a more optimal control algorithm. The required voltage adjustments for steady-state synchronization were consistently higher than expected, likely because of losses in the transformers, inductive loading causing a delay between the voltages applied on the motor lines, and the presence of harmonics in the converter outputs.

Use of transformers with higher secondary-to-primary turns ratios further improves efficiency, but this also limits the referred auxiliary converter output voltage and therefore the range of torque differentials for which speed and position synchronization can be achieved, so transformer selection must take this tradeoff into account. The selection of auxiliary converter bus voltage has little impact on performance when using CVHz control, but using a lower bus voltage slightly improves efficiency when using IDFOC control, which could be because harmonics are less present with a voltage command closer to the maximum. The effect of selection of these parameters on the presence of voltage and current harmonics in the system merits further investigation.

No benefit was observed from simulating in a reference frame synchronized with the electrical frequency or the motor velocity, as the switching effects still cause rapid voltage changes in all reference frames. Therefore, a stationary reference frame was chosen for simulation and arbitrary reference transformation functions were replaced with fixed stationary transform functions in the main simulation loop, except for where values in other reference frames were needed (e.g., deter-

mining commands for the central converter). This produced a small but significant reduction in simulation time, allowing for faster collection of data.

An aliasing effect was observed when the switching period for the external resistance or auxiliary converter was an integer multiple of the step time, producing a low resolution in the fast-average plots for external resistance and auxiliary converter voltage. This appears to be because changes in the duty cycle that do not cross a step time boundary have no effect on the samples collected at each step. This was resolved by changing the switching period to a slightly higher value with a much larger least common multiple with the step time.

The use of ideal voltage sources to simulate switching losses described in [5] had some peculiar effects, including steady-state operation above commanded speed seen in Figure 3.8 and the chaotic shape of the speed plots soon after load is applied in 3.20. This technique appears to be primarily intended for quantifying converter efficiency and may not be well-suited for simulating the effects of switching losses on synchronization performance, though the observed effects were not large. Simulating with a more detailed transient model of switching effects would likely be preferable for studying how the presence of switching losses affects the synchronization performance of the methods studied here.

Chapter 4

Testbed

This chapter discusses the author’s contributions to a testbed for hardware emulation of various aerospace systems, including CCMM EM-TRAS implementations. The information presented here is summarized from [3]. The aerospace testbed provides a flexible environment to test different mechanical, electrical, and control architectures, is fully sensor-equipped, and includes high-speed data acquisition, allowing for anomaly detection and health prognostics investigation.

Figure 4.1 shows a functional block diagram of the testbed. It consists of three subsystems: the test stand, where the four actuation lines are mounted; the motor control cabinet, which controls the drive machines; and the supervisory control and data acquisition (SCADA) system, responsible for controlling the load machines and measuring high-speed data from the test stand.

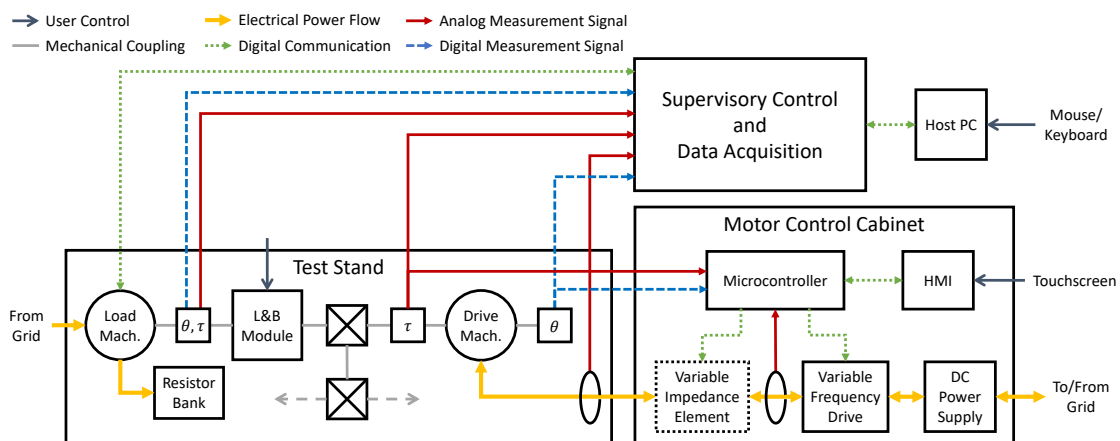


Figure 4.1: High-level functional block diagram of the aerospace testbed.

The test stand contains four actuation lines that can be mechanically coupled together by one-to-one three-way gearboxes. Each line consists of a drive machine, a load and braking module (L&B) that emulates mechanical anomalies such as jamming and sideload, a load machine responsible for emulating load torque, and several sensors, such as torque sensors (indicated by τ) and position sensors (indicated by θ). The motor control cabinet acts as the electrical drive of the

drive machines, including a custom-built variable frequency drive (VFD). It contains controller elements (HMI and microcontroller), a variable impedance element for each line responsible for position synchronization, and sensors to measure voltages and currents of each drive machine. The SCADA controls the load machines, acquires high-speed measurements from the test stand, and provides an interface for user commands. The author’s contributions to the testbed include development of the SCADA system and HMI, along with construction of the testbed’s electrical measurement equipment.

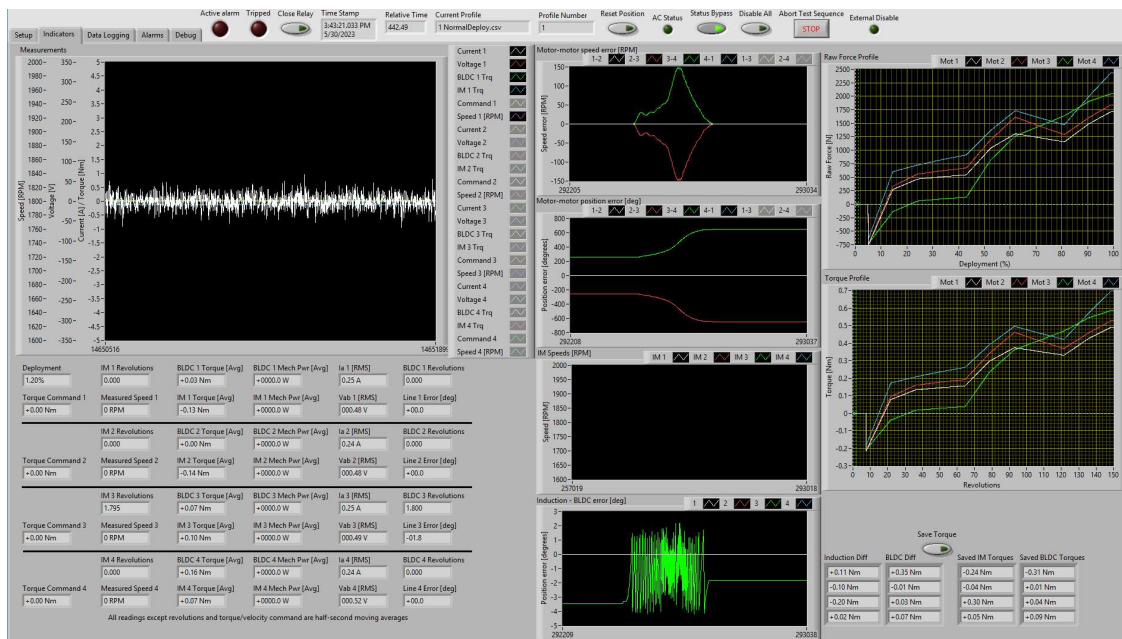


Figure 4.2: Screenshot of the “Indicators” tab of the LabVIEW SCADA user interface for the testbed.

SCADA is implemented for the testbed by a LabVIEW project consisting of an interface running on a Windows host PC, pictured in Figure 4.2, and a realtime program running on an NI PXI-1031 unit with a PXI-8016 controller. The PC interface allows for manual speed or torque control of the load motors or automated torque control based on loaded position-indexed torque profiles. Plots and data logging features for voltage, current, torque, speed, and position are also available on the PC interface, along with automated alarm and trip functionality. The PXI unit contains a PXI-6259 analog input module for voltage, current, and torque readings, a PXI-6509 digital

I/O module for load motor control, and a PXI-7833R FPGA module with digital inputs for speed and position measurement along with generation of PWM signals for the digital output module.

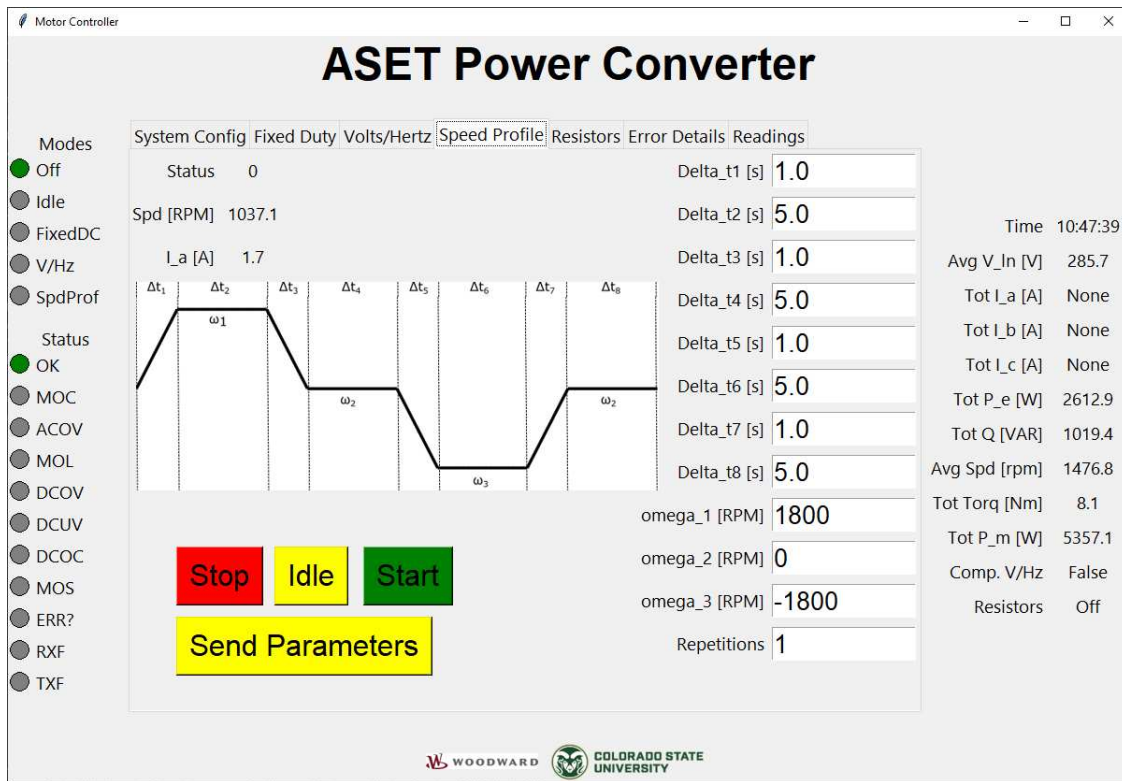


Figure 4.3: Screenshot of the “Speed Profile” tab of the HMI for the motor control cabinet.

The HMI for the motor control cabinet is programmed in Python and runs on a Raspberry Pi with a touchscreen display. Using it, the user commands the microcontroller, chooses different operation modes, sets control parameters, and accesses the system status, which includes voltage, current, speed, and torque measurements along with fault conditions. Figure 4.3 shows a screenshot of the HMI. The communication protocol used is UART/SCI, an asynchronous serial protocol that allows bidirectional, simultaneous communication between the microcontroller and the HMI.

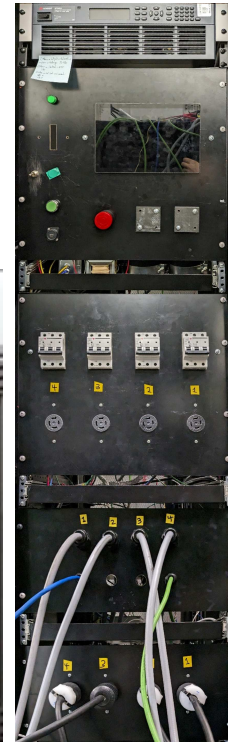
The testbed is equipped with sensors to measure mechanical variables (angular position and torque) and electrical variables (voltage and current). Each actuation line has two torque sensors measuring the torque at the shafts of the drive and load motors. The torque sensors provide an output voltage signal in the range ± 5 V corresponding to a measured torque in the range ± 20 Nm.

The driving-side torque signals are measured by the microcontroller, while both sides are measured by the SCADA. The torque sensors also provide an incremental encoder digital output with 60 pulses per revolution, used by the SCADA to quantify differences in speed and position between the load and driving sides of each line. Connected to the other end of the shaft of each driving machine is a 14-bit absolute encoder, which provides a high-resolution position measurement to the microcontroller and SCADA. Inside the cabinet are voltage and current sensors that measure the three-phase output voltage of the power converter and two phases of the currents of all the driving machines (the third current is calculated by the microcontroller based on no-neutral connection). Sensing boards are used to condition all of the sensors' output voltage signals to match the analog input range of the DSP (0 to 3.3 V) and to filter high-frequency components that could cause aliasing problems. Mounted on the testbed are voltage and current sensors that measure one phase of the line-to-line voltage and one phase of the current for each driving machine, feeding output voltage signals in the range ± 10 V into the SCADA. An optocoupler circuit is used to condition the encoder digital outputs from the load side torque sensors for the SCADA.

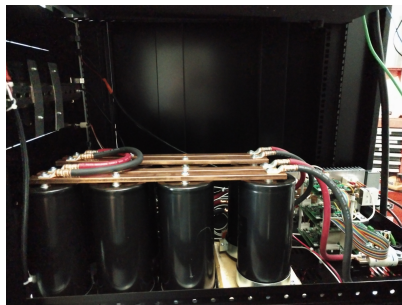
The test stand and motor control cabinet are pictured in Figure 4.4. In Figure 4.4a, the four drive motors are the induction motors on the right, the four load motors are the BLDC motors on the left, the gearboxes are visible in the middle, the 14-bit absolute encoders are on the far right, and the torque sensors are the red boxes connected to each of the eight motors. In Figure 4.4b, the DC power supply for the motor control cabinet is visible at the top. The screen below it displays the HMI, surrounded by physical controls and indicators: from the top left, a power status light, a protection bypass switch, a DC contactor status light, an enable button for the DC contactors, and an emergency stop button. The middle panel mounts breakers to control power to each motor and outlets that directly connect to the VFD output, while the lowest panel mounts outlets that connect to the VFD output through the variable impedance elements. The panel between the rows of outlets provides connections for the encoders and torque sensors.



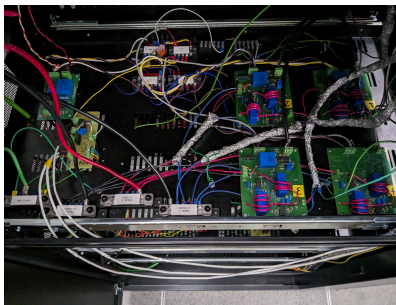
(a) Test stand.



(b) Control cabinet.



(c) Upper rack.



(d) Sensor rack.



(e) Microcontroller rack.

Figure 4.4: Photographs of (a) test stand and (b,c,d,e) motor control cabinet.

The remaining subfigures display internal views of the motor control cabinet. Figure 4.4c shows the upper rack, which contains high-voltage capacitors allowing for rapid motor acceleration along with the custom VFD. Figure 4.4d shows the sensor rack, which contains voltage and current sensors for each motor and the DC bus as well as an auxiliary board conditioning additional readings for the microcontroller. Figure 4.4e shows the microcontroller rack, which contains the microcontroller rack, the board for conditioning torque signals, and some linear power supplies

used to power the encoders and sensors. Below the microcontroller rack is the bottom rack, which mounts the external resistors and their control boards.

The currently available variable impedance elements are boards that implement the external stator resistance synchronization method described in Chapter 2.2 and [1]. Hardware tests of this methodology have recently been completed, with abridged results presented in Figure 4.5.

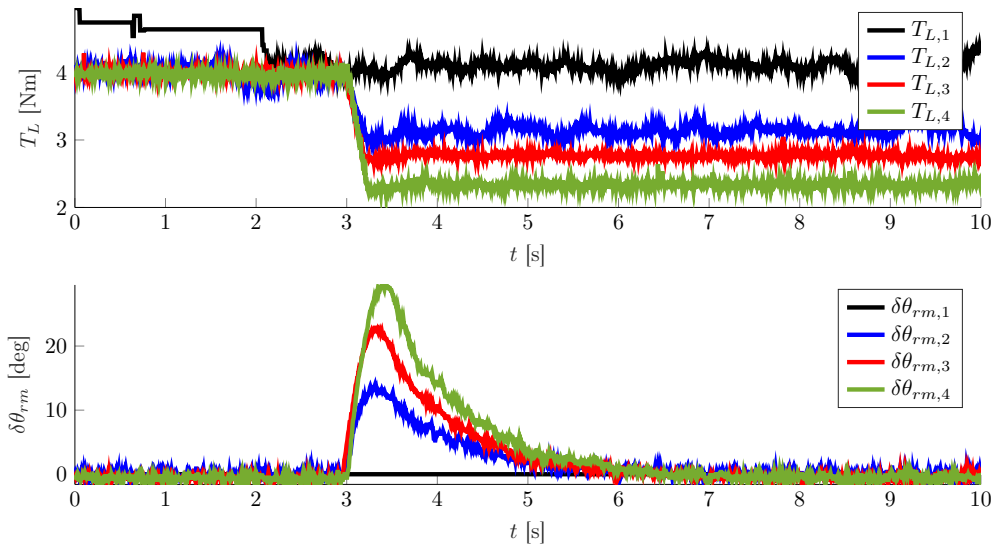


Figure 4.5: Hardware torque and position synchronization data collected from the textbed.

In the test used to generate Figure 4.5, CVHz control was used and all BLDC motors were set to apply net torque load (including windage and friction from the testbed) of 4 N·m to all IMs at first with the external resistance control keeping their positions synchronized, then the load commands were dropped by 0.8 N·m, 1.2 N·m, and 1.6 N·m for BLDC motors 2, 3, and 4 respectively. The recorded load torque reflected this change within about 0.25 s and remained fairly steady over time, while the position error rose to peaks of about 14°, 23°, and 30° for motors 2, 3, and 4 respectively before settling in about 3 s. This shows that the external resistance methodology works as expected for maintaining speed and position synchronization.

Chapter 5

Future Work

Research building on the work presented in this thesis will include the following:

- Further experimentation in simulation of the auxiliary converter methodology, including operation in generator mode, continuously varying load torques, and parameters of other motors and transformers that could realistically be used for this purpose, exploring the impact of changes to resistances and inductances.
- Investigation of the alternate auxiliary converter control strategy for operation below rated voltage, which allows an auxiliary converter to raise the voltage applied to its IM. This may achieve synchronization more quickly or more efficiently in some cases.
- Development of optimal control of both position synchronization methodologies (e.g., using H-infinity methods; see [6] for external resistance control optimization). For the auxiliary converter methodology, this could include use of a nonzero v_{d2}^* command, adding an intentional phase difference between central and auxiliary converters.
- Optimization of transformer parameters using comprehensive nonlinear models of the motors, converters, and transformers, with objectives of maximum electrical to mechanical transfer efficiency, minimum position error settling time and overshoot, and minimum cost of implementation. Obtaining a solution would require simulations performed on a computing cluster over a long period.
- Hardware validation of the auxiliary converter methodology using the testbed described in Chapter 4. This would require expansions to the testbed adding transformers and auxiliary converters in series with the drive motors, along with control capability for each auxiliary converter.

Chapter 6

Conclusions

Advances to the development of two methodologies for speed and position synchronization of induction machines in CCMM architecture used for applications such as EM-TRAS were presented in this thesis. The methodology introduced here, which uses auxiliary converters coupled through transformers to the lines supplying power to the IMs, has been shown in simulation to have better energy efficiency and some more desirable performance characteristics compared to the methodology presented in [1], which uses switched external resistance in series with the stators of the IMs. However, these improvements come at the expense of somewhat more complex requirements for equipment and control capability. The author also made significant contributions to the construction of a testbed used for hardware validation of the external resistance methodology, improving confidence that it would be feasible for application in a full-scale CCMM system. Both methodologies will likely have applications that are better suited for each due to their differing advantages and disadvantages, so investigation of both appears to be worthwhile.

It must be acknowledged that the simulations presented in this thesis constitute only preliminary analysis of the auxiliary converter methodology. One load torque profile was studied using a spread of transformer turns ratios and auxiliary converter bus voltages, but the resistance and inductance parameters of the transformers were chosen arbitrarily and conditions other than a step change from unloaded to positive load torque setpoints were not investigated. Simple PI feedback control was used with arbitrary parameter selection, and modeling and measurement uncertainties were not calculated. The author hopes that future study of both methodologies will provide a more complete picture of their feasibility and desirable use cases.

Bibliography

- [1] C. d. A. Lima, J. Cale, and K. E. Shahroudi, “Rotor position synchronization in central-converter multi-motor electric actuation systems,” *Energies*, vol. 14, no. 22, 2021. [Online]. Available: <https://www.mdpi.com/1996-1073/14/22/7485>
- [2] M. Song, D. Li, X. Xiao, Y. Luo, J. Hu, and K. Yang, “Novel four-quadrant variable impedance and its application,” *IET Power Electronics*, vol. 10, no. 15, pp. 2124–2132, 2017. [Online]. Available: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-pel.2017.0173>
- [3] C. d. A. Lima, Z. Miller, A. Riley, C. Lute, R. Bushue, B. Chan, A. Santos, S. Madcadi, J. Cale, M. Susman, and K. Shahroudi, “A novel electromechanical actuation testbed for emulation of aerospace actuation systems,” in *33rd Aerospace Testing Seminar*, El Segundo, California (USA), May 2023, (accepted, to appear).
- [4] P. C. Krause, O. Wasynczuk, and S. Pekarek, *Electromechanical Motion Devices*, 2nd ed. Wiley-IEEE Press, 2012.
- [5] P. C. Krause, O. Wasynczuk, and S. D. Sudhoff, *Analysis of Electric Machinery and Drive Systems*, 2nd ed. Wiley-IEEE Press, 2002.
- [6] C. d. A. Lima and J. Cale, “Control design for position synchronization in central converter multi-machine actuators,” 2023.

Appendix A

Simulation Parameters

Table A.1: Parameters for 15 hp Induction Machine.

Description	Symbol	Value
Machine poles	P	4
Rated torque	T_{rated}	61.1 [N·m]
Stator resistance	r_s	0.06 [Ω]
Rotor resistance (referred)	r'_r	0.15 [Ω]
Stator leakage inductance	$L_{\ell s}$	1.17 [mH]
Rotor leakage inductance (referred)	$L'_{\ell r}$	1.14 [mH]
Magnetizing inductance	L_{MM}	33.4 [mH]
Windage and friction loss coefficient	B_m	5.41×10^{-4} [N·m·s]
Rotor inertia	J	0.45 [kg·m ²]

Table A.2: Parameters for Compensated Volts-per-Hertz Control.

Description	Symbol	Value
Low-pass filter time constant	τ_{LPF}	0.1 [s]
Slew-rate limiter minimum	α_{\min}	-75.4 [rad/s ²]
Slew-rate limiter maximum	α_{\max}	75.4 [rad/s ²]
Base rotor speed	ω_b	377 [rad/s]
Base voltage (rms)	V_b	139 [V]
Central converter DC input voltage	v_{dc}	339 [V]
PWM switching frequency	f_s	3 [kHz]
Switch voltage drop	v_{csw}	{0,5} [V]
Diode voltage drop	v_{cd}	{0,5} [V]

Table A.3: Parameters for Speed Control.

Description	Symbol	Value
Speed control proportional gain	K_{scp}	26.7 [N·m·s/rad]
Speed control integral gain	K_{sci}	8.33 [1/s]

Table A.4: Parameters for Indirect Field-oriented Control.

Description	Symbol	Value
Torque limiter upper max	$T_{e,\max}$	$2T_{rated}$
Torque limiter lower max	$T_{e,\min}$	$-2T_{rated}$
Hysteresis band tolerance	h_b	0.1 [A]
Converter DC input voltage	v_{dc}	339 [V]
Switch voltage drop	v_{csw}	{0,5} [V]
Diode voltage drop	v_{cd}	{0,5} [V]

Table A.5: Parameters for External Resistor Circuit and Control.

Description	Symbol	Value
Base resistance	r	1.5 [Ω]
PWM switching frequency	f_{rs}	4.988 [kHz]
Resistance control proportional gain	K_{rp}	30 [1/rad]
Resistance control integral gain	K_{ri}	60 [1/(rad·s)]
Switch voltage drop	v_{rsw}	{0,5} [V]
Diode voltage drop	v_{rd}	{0,5} [V]

Table A.6: Parameters for Auxiliary Converter Circuit and Control.

Description	Symbol	Value
Auxiliary converter DC input voltage	v_{xdc}	{339,170} [V]
PWM switching frequency	f_{xs}	4.988 [kHz]
Voltage control proportional gain	K_{xp}	80 [V/rad]
Voltage control integral gain	K_{xi}	120 [V/(rad·s)]
Switch voltage drop	v_{xsw}	{0,5} [V]
Diode voltage drop	v_{xd}	{0,5} [V]

Table A.7: Parameters for Auxiliary Converter Transformers.

Description	Symbol	Value
Secondary-to-primary turns ratio	N_2/N_1	{5,3,2}
Primary resistance	r_1	0.001 [Ω]
Secondary resistance (referred)	r'_2	$0.01(N_1/N_2)$ [Ω]
Primary leakage inductance	$L_{\ell 1}$	0.1 [mH]
Secondary leakage inductance (referred)	$L'_{\ell 2}$	$0.5(N_1/N_2)$ [mH]
Magnetizing inductance	L_{MT}	10 [mH]

Appendix B

MATLAB[®] Code for Simulation and Analysis

This appendix includes all code used for simulation of both synchronization methodologies and analysis of simulation data presented in this thesis. Code provided by Cláudio Lima used for simulations performed in [1] was used as a baseline.

B.1 CVHz Control, External Resistor Method

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Multirun_3IM_CVHZ_res.m
3 % Runs 3-motor simulation of resistor sync method using CVHz control
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 clear
6 close all
7 clc
8
9 % load k_hinf
10 % [AK,Bk,Ck,Dk] = ssdata(k_hinf);
11
12 Kp = -30; % 1/rad
13 Ki = -60; % 1/(rad*s)
14 [AK,Bk,Ck,Dk] = ssdata(tf([Kp Ki],[1 0]));
15
16 % Code currently only supports 3 identical motors
17 ParametersIM = ReturnParametersIM_15hp;
18
19 rs = ParametersIM.rs;
20 rr = ParametersIM.rr;
21 Lls = ParametersIM.Lls;
22 Llr = ParametersIM.Llr;
23 Lm = ParametersIM.Lm;
24 Ls = ParametersIM.Ls;
25 Lr = ParametersIM.Lr;
26 % sigma = 1 - Lm^2/(Ls*Lr);
27 % tau_s = Ls/rs;
28 % tau_r = Lr/rr;
29 % sigma_s = tau_s*sigma;
30 % sigma_r = tau_r*sigma;
31 % mu_s = 1/sigma_s;
32 % mu_r = 1/sigma_r;
33
34 % Incorporating mu_s inside the param structure
35 % ParametersIM.mu_s = mu_s;
36
37 ParamController.Ak = Ak;
38 ParamController.Bk = Bk;
39 ParamController.Ck = Ck;
40 ParamController.Dk = Dk;
41 ParamController.order = length(Ak);
42
43 % ParamController.rext_max = ParametersIM.2.rs*100;
44 ParamController.rext_max = 1.5;
45
46 % Parameters for switching losses
47 ParamSys.vswM = 0; % Main converter switch voltage
48 ParamSys.vdM = 0; % Main converter diode voltage
49 ParamSys.vswR = 0; % Resistor board switch voltage
50 ParamSys.vdR = 0; % Resistor board diode voltage
51
52 TL1_variation = 0; % Variation in load torque in pu
53 TL2_variation = -0.2;
54 TL3_variation = -0.3;
55
56 time_variation_TL1 = 2;
57 time_variation_TL2 = 2;
```

```

58 time_variation_TL3 = 2;
59
60 ParamSys.TL1_variation = TL1_variation;
61 ParamSys.TL2_variation = TL2_variation;
62 ParamSys.TL3_variation = TL3_variation;
63 ParamSys.time_variation_TL1 = time_variation_TL1;
64 ParamSys.time_variation_TL2 = time_variation_TL2;
65 ParamSys.time_variation_TL3 = time_variation_TL3;
66
67 num_states = 20 + 2*ParamController.order;
68 x0 = zeros(1,num_states); % Initial conditions for state variables
69 % load x0_save
70 % x0 = x0_save;
71
72 t_ini = 0;
73 t_end = 6;
74
75 deltat = 10e-6;
76 tspan = t_ini:deltat:t_end;
77 options = odeset('MaxStep', deltat, 'RelTol', 1e-5, 'AbsTol', 1e-6);
78 % options = odeset();
79
80 tic
81 [t,y] = ode45(@(t,y) ThreeIM_1Converter_CVHz_res(t,y,ParametersIM, ...
82     ParamSys, ParamController), tspan, x0, options);
83 toc
84
85 % Save state
86 % x0_save = y(end,:);
87 % save x0_save x0_save
88
89 %-----%
90 %                               POST-PROCESSING
91 %-----%
92
93 %-----%
94 %                               PICKING UP THE STATES
95 %-----%
96 %                               1ST MOTOR
97 %-----%
98 lambda_ds_1 = y(:,1); % Flux ds [Wb]
99 lambda_qs_1 = y(:,2); % Flux qs [Wb]
100 lambda_dr_1 = y(:,3); % Flux dr [Wb]
101 lambda_qr_1 = y(:,4); % Flux qr [Wb]
102 wr_1 = y(:,5); % Rotor speed (electric) [rad/s]
103 theta_r_1 = y(:,6); % Rotor position (electric) [rad]
104 %-----%
105 %                               COMPENSATED VOLTS-PER-HERZ CONTROL
106 %-----%
107 R_theta_e = y(:,7); % Electrical angular position [rad]
108 % y_LPF = y(:,8); % Output of the low-pass filter
109 %-----%
110 %-----%
111 %                               2ND MOTOR
112 %-----%
113 lambda_ds_2 = y(:,9); % Flux ds [Wb]
114 lambda_qs_2 = y(:,10); % Flux qs [Wb]
115 lambda_dr_2 = y(:,11); % Flux dr [Wb]
116 lambda_qr_2 = y(:,12); % Flux qr [Wb]
117 wr_2 = y(:,13); % Rotor speed (electric) [rad/s]
118 theta_r_2 = y(:,14); % Rotor position (electric) [rad]
119 %-----%
120 %-----%
121 %                               3RD MOTOR
122 %-----%
123 lambda_ds_3 = y(:,15); % Flux ds [Wb]
124 lambda_qs_3 = y(:,16); % Flux qs [Wb]
125 lambda_dr_3 = y(:,17); % Flux dr [Wb]
126 lambda_qr_3 = y(:,18); % Flux qr [Wb]
127 wr_3 = y(:,19); % Rotor speed (electric) [rad/s]
128 theta_r_3 = y(:,20); % Rotor position (electric) [rad]
129 %-----%
130
131 %-----%
132 %                               EXTRACTING OUTPUT STRUCTURES FROM DIFF EQ FUNCTION
133 %-----%
134 [-, IM1_out, IM2_out, IM3_out, Sw_out] = cellfun(@(t,y) ...
135     ThreeIM_1Converter_CVHz_res(t, y, ParametersIM, ParamSys, ...
136     ParamController), num2cell(t), num2cell(y,2), 'uni', 0);
137
138 R_vqsC = zeros(numel(t),1);
139 R_vdsC = zeros(numel(t),1);

```

```

140
141 R_TL_1 = zeros(numel(t),1);
142 R_Te_1 = zeros(numel(t),1);
143 theta_a_1 = zeros(numel(t),1);
144 R_vqs_1 = zeros(numel(t),1);
145 R_vds_1 = zeros(numel(t),1);
146
147 R_TL_2 = zeros(numel(t),1);
148 R_Te_2 = zeros(numel(t),1);
149 theta_a_2 = zeros(numel(t),1);
150 R_vqs_2 = zeros(numel(t),1);
151 R_vds_2 = zeros(numel(t),1);
152 rext2_pwm = zeros(numel(t),1);
153 rext2_cont = zeros(numel(t),1);
154
155 R_TL_3 = zeros(numel(t),1);
156 R_Te_3 = zeros(numel(t),1);
157 theta_a_3 = zeros(numel(t),1);
158 R_vqs_3 = zeros(numel(t),1);
159 R_vds_3 = zeros(numel(t),1);
160 rext3_pwm = zeros(numel(t),1);
161 rext3_cont = zeros(numel(t),1);
162
163 R_S1 = zeros(numel(t),1);
164 R_S2 = zeros(numel(t),1);
165 R_S3 = zeros(numel(t),1);
166
167 for i=1:numel(t)
168     R_vqsC(i) = IM1_out{i}.vqsC;
169     R_vdsC(i) = IM1_out{i}.vdsC;
170
171     R_TL_1(i) = IM1_out{i}.TL;
172     R_Te_1(i) = IM1_out{i}.Te;
173     % theta_a_1(i) = IM1_out{i}.theta_a;
174
175     R_vqs_1(i) = IM1_out{i}.vqs;
176     R_vds_1(i) = IM1_out{i}.vds;
177
178     R_TL_2(i) = IM2_out{i}.TL;
179     R_Te_2(i) = IM2_out{i}.Te;
180     % theta_a_2(i) = IM2_out{i}.theta_a;
181
182     R_vqs_2(i) = IM2_out{i}.vqs;
183     R_vds_2(i) = IM2_out{i}.vds;
184
185     rext2_pwm(i) = IM2_out{i}.rext_pwm;
186     rext2_cont(i) = IM2_out{i}.rext_cont;
187
188     R_TL_3(i) = IM3_out{i}.TL;
189     R_Te_3(i) = IM3_out{i}.Te;
190     % theta_a_3(i) = IM3_out{i}.theta_a;
191
192     R_vqs_3(i) = IM3_out{i}.vqs;
193     R_vds_3(i) = IM3_out{i}.vds;
194
195     rext3_pwm(i) = IM3_out{i}.rext_pwm;
196     rext3_cont(i) = IM3_out{i}.rext_cont;
197
198     R_S1(i) = Sw_out{i}.S(1);
199     R_S2(i) = Sw_out{i}.S(2);
200     R_S3(i) = Sw_out{i}.S(3);
201
202 end
203 %-----%
204 clear y IM1_out IM2_out IM3_out Sw_out
205 %-----%
206 % Auxiliary Equations for the IMs
207 %-----%
208 % Transformation of electrical rotor speed and electrical angular position
209 % to mechanical rotor speed and mechanical angular position
210 % 1st IM
211 R_wm_1 = wr_1/(ParametersIM.P/2);
212 R_theta_m_1 = theta_r_1/(ParametersIM.P/2);
213 % 2nd IM
214 R_wm_2 = wr_2/(ParametersIM.P/2);
215 R_theta_m_2 = theta_r_2/(ParametersIM.P/2);
216 % 3rd IM
217 R_wm_3 = wr_3/(ParametersIM.P/2);
218 R_theta_m_3 = theta_r_3/(ParametersIM.P/2);
219
220 % Transformation of fluxes to currents – DQ components
221 % 1ST MOTOR

```

```

222 [R_ids_1, R_iqs_1, R_idr_1, R_iqr_1] = Fluxes2Currents(lambda_ds_1, ...
223     lambda_qs_1, lambda_dr_1, lambda_qr_1, ParametersIM);
224 % 2ND MOTOR
225 [R_ids_2, R_iqs_2, R_idr_2, R_iqr_2] = Fluxes2Currents(lambda_ds_2, ...
226     lambda_qs_2, lambda_dr_2, lambda_qr_2, ParametersIM);
227 % 3RD MOTOR
228 [R_ids_3, R_iqs_3, R_idr_3, R_iqr_3] = Fluxes2Currents(lambda_ds_3, ...
229     lambda_qs_3, lambda_dr_3, lambda_qr_3, ParametersIM);
230
231 % Calculating voltage actually reaching motor 2
232 R_vqs_2 = R_vqs_2 - (R_iqs_2 .* rext2_pwm);
233 R_vds_2 = R_vds_2 - (R_ids_2 .* rext2_pwm);
234
235 % Calculating voltage actually reaching motor 3
236 R_vqs_3 = R_vqs_3 - (R_iqs_3 .* rext3_pwm);
237 R_vds_3 = R_vds_3 - (R_ids_3 .* rext3_pwm);
238
239 % Clear variables not saved
240 cclear lambda_ds_1 lambda_qs_1 lambda_dr_1 lambda_qr_1 theta_r_1 wr_1 ...
241     lambda_ds_2 lambda_qs_2 lambda_dr_2 lambda_qr_2 theta_r_2 wr_2 ...
242     lambda_ds_3 lambda_qs_3 lambda_dr_3 lambda_qr_3 theta_r_3 wr_3 ...
243     i j tspan t_ini t_end x0 num_states deltat
244 %-----%

1 function [y, IM1, IM2, IM3, Sw] = ThreeIM_1Converter_CVHz_res(t, u, ...
2     ParametersIM, ParamSys, ParamController)
3 %-----%
4 %                               STATES
5 %-----%
6 %                               1ST MOTOR
7 %-----%
8 lambda_ds_1 = u(1); % Flux ds [Wb]
9 lambda_qs_1 = u(2); % Flux qs [Wb]
10 lambda_dr_1 = u(3); % Flux dr [Wb]
11 lambda_qr_1 = u(4); % Flux qr [Wb]
12 wr_1 = u(5); % Rotor speed (electric) [rad/s]
13 theta_r_1 = u(6); % Rotor position (electric) [rad]
14 %-----%
15 %                               COMPENSATED VOLTS-PER-HERZ CONTROL
16 %-----%
17 theta_e = u(7); % Electrical angular position [rad]
18 y_LPF = u(8); % Output of the low-pass filter
19 %-----%
20 %                               2ND MOTOR
21 %-----%
22 %-----%
23 lambda_ds_2 = u(9); % Flux ds [Wb]
24 lambda_qs_2 = u(10); % Flux qs [Wb]
25 lambda_dr_2 = u(11); % Flux dr [Wb]
26 lambda_qr_2 = u(12); % Flux qr [Wb]
27 wr_2 = u(13); % Rotor speed (electric) [rad/s]
28 theta_r_2 = u(14); % Rotor position (electric) [rad]
29 %-----%
30 %-----%
31 %                               3RD MOTOR
32 %-----%
33 lambda_ds_3 = u(15); % Flux ds [Wb]
34 lambda_qs_3 = u(16); % Flux qs [Wb]
35 lambda_dr_3 = u(17); % Flux dr [Wb]
36 lambda_qr_3 = u(18); % Flux qr [Wb]
37 wr_3 = u(19); % Rotor speed (electric) [rad/s]
38 theta_r_3 = u(20); % Rotor position (electric) [rad]
39 %-----%
40 %-----%
41 %                               RESISTANCE CONTROL OF MOTORS 2&3
42 %-----%
43 %-----%
44 %                               CONTROLLERS
45 %-----%
46 n = ParamController.order;
47 % Automatic state-building of the controllers, based on order
48 xk2 = u(21:21+n-1);
49 xk2 = reshape(xk2,n,1);
50 xk3 = u(21+n:21+2*n-1);
51 xk3 = reshape(xk3,n,1);
52 %-----%
53 %-----%
54 %-----%
55 %                               GET SYSPARAMS
56 %-----%
57 vswM = ParamSys.vswM;
58 vdM = ParamSys.vdM;

```

```

59 vswR = ParamSys.vswR;
60 vdR = ParamSys.vdR;
61
62 TL1_delta = ParamSys.TL1_variation;
63 time_delta_TL1 = ParamSys.time_variation_TL1;
64
65 TL2_delta = ParamSys.TL2_variation;
66 time_delta_TL2 = ParamSys.time_variation_TL2;
67
68 TL3_delta = ParamSys.TL3_variation;
69 time_delta_TL3 = ParamSys.time_variation_TL3;
70
71 if t > time_delta_TL1
72     TL_1 = (1+TL1_delta)*ParametersIM.Tnom;
73 else
74     % TL_1 = ParametersIM.Tnom;
75     TL_1 = 0;
76 end
77
78 if t > time_delta_TL2
79     TL_2 = (1+TL2_delta)*ParametersIM.Tnom;
80 else
81     % TL_2 = ParametersIM.Tnom;
82     TL_2 = 0;
83 end
84
85 if t > time_delta_TL3
86     TL_3 = (1+TL3_delta)*ParametersIM.Tnom;
87 else
88     % TL_3 = ParametersIM.Tnom;
89     TL_3 = 0;
90 end
91 %-----%
92 %-----%
93 %-----%
94 % Arbitrary Reference Frame
95 %-----%
96 % Stator reference
97 %-----%
98 % % 1st IM
99 % wa_1 = 0;
100 % theta_a_1 = 0;
101 %
102 % % 2nd IM
103 % wa_2 = 0;
104 % theta_a_2 = 0;
105 %
106 % % 3rd IM
107 % wa_3 = 0;
108 % theta_a_3 = 0;
109 %-----%
110 %-----%
111 %-----%
112 % Auxiliary Equations for the IMs
113 %-----%
114 % Transformation of fluxes to currents – DQ components
115 % 1ST MOTOR
116 [ids_1, iqs_1, idr_1, iqr_1] = Fluxes2Currents(lambda_ds_1, lambda_qs_1, ...
117     lambda_dr_1, lambda_qr_1, ParametersIM);
118 % 2ND MOTOR
119 [ids_2, iqs_2, idr_2, iqr_2] = Fluxes2Currents(lambda_ds_2, lambda_qs_2, ...
120     lambda_dr_2, lambda_qr_2, ParametersIM);
121 % 3RD MOTOR
122 [ids_3, iqs_3, idr_3, iqr_3] = Fluxes2Currents(lambda_ds_3, lambda_qs_3, ...
123     lambda_dr_3, lambda_qr_3, ParametersIM);
124
125 % Electromagnetic and Load Torques
126 % 1ST MOTOR
127 Te_1 = 3/4*ParametersIM.P*(lambda_ds_1.*iqs_1 - lambda_qs_1.*ids_1);
128 % 2ND MOTOR
129 Te_2 = 3/4*ParametersIM.P*(lambda_ds_2.*iqs_2 - lambda_qs_2.*ids_2);
130 % 3RD MOTOR
131 Te_3 = 3/4*ParametersIM.P*(lambda_ds_3.*iqs_3 - lambda_qs_3.*ids_3);
132
133 % Transformation of electrical rotor speed and electrical angular position
134 % to mechanical rotor speed and mechanical angular position
135 % 1st IM
136 % wm_1 = wr_1/(ParametersIM.P/2);
137 theta_m_1 = theta_r_1/(ParametersIM.P/2);
138 % 2nd IM
139 % wm_2 = wr_2/(ParametersIM.P/2);
140 theta_m_2 = theta_r_2/(ParametersIM.P/2);

```

```

141 % 3rd IM
142 % wm_2 = wr_2/(ParametersIM.P/2);
143 theta_m_3 = theta_r_3/(ParametersIM.P/2);
144
145 % Transforming stator and rotor currents from qd reference to abc variables
146 % % 1st IM
147 % [ias_1, ibs_1, ics_1] = qd2abcs(iqs_1, ids_1, theta_a_1);
148 % % 2nd IM
149 % [ias_2, ibs_2, ics_2] = qd2abcs(iqs_2, ids_2, theta_a_2);
150 % % 3rd IM
151 % [ias_3, ibs_3, ics_3] = qd2abcs(iqs_3, ids_3, theta_a_3);
152 % 1st IM
153 ias_1 = iqs_1;
154 [ibs_1, ics_1] = statbc(iqs_1, ids_1);
155 % 2nd IM
156 ias_2 = iqs_2;
157 [ibs_2, ics_2] = statbc(iqs_2, ids_2);
158 % 3rd IM
159 ias_3 = iqs_3;
160 [ibs_3, ics_3] = statbc(iqs_3, ids_3);
161
162 % Rotor Voltages (squirrel cage) — DQ components
163 % 1st IM
164 vdr_1 = 0;
165 vqr_1 = 0;
166 % 2nd IM
167 vdr_2 = 0;
168 vqr_2 = 0;
169 % 3rd IM
170 vdr_3 = 0;
171 vqr_3 = 0;
172 %
173 %
174 %
175 % Low-Pass Filter Definitions/Equations
176 %
177 % The signal x_corr = 3*P*(vqs_e_star*iqs_e - 2*rs*Is^2)/Ktv (Eq. 14.2–16
178 % from Krause) is filtered using a low-pass filter (LPF), accordingly to
179 % Krause's suggestion. In order to implement this filtering process, a new
180 % state, y, was created, from the diff. equation of a LPF:
181 % dy/dt = (x-y)/tau,
182 % where x is the filter input and y is the filter output, while tau is the
183 % filter time constant.
184 % The LPF output is the signal X_corr, used in the Volts-per-Hertz
185 % compensated control section down below.
186 % The input of the filter is set in the Volts-per-Hertz compensated control
187 % section down below, since, as seen above, it depends on vqs_e_star, which
188 % in turn is defined by the Volts-per-Hertz compensated control equations.
189 %
190 % Time constant of the filter
191 tau_LPF = 0.1;
192
193 % Output of the LPF filter
194 X_corr = y_LPF;
195 %
196 %
197 %
198 % Compensated Volts-per-Hertz Control
199 %
200 %
201 % Transforming from iabcs to iqds_e
202 [iqs_e_1, ids_e_1] = abcs2qd(ias_1, ibs_1, ics_1, theta_e);
203
204 % Rated voltage (line to neutral rms) and rated radian frequency
205 Vb = ParametersIM.Vs/sqrt(3);
206 wb = ParametersIM.f*2*pi;
207
208 % Commanded mechanical speed
209 wrm_star = DesiredSpeed(t);
210
211 % Mechanical commanded speed to electrical commanded speed
212 wr_star = wrm_star*ParametersIM.P/2;
213
214 % Eq 14.2–7 (Krause)
215 % Equation from Krause was wrong (commented version). The right one, after
216 % checking 2 times, with 2 different approaches, is the one used here!
217 % num_Ktv = 3*ParametersIM.P/2*ParametersIM.Lm^2*ParametersIM.rr*Vb^2;
218 % den_Ktv = ParametersIM.rr*2*(ParametersIM.rs^2+wb^2*ParametersIM.Ls^2);
219 num_Ktv = 3*ParametersIM.P*ParametersIM.Lm^2*Vb^2;
220 den_Ktv = ParametersIM.rr*2*(ParametersIM.rs^2+wb^2*ParametersIM.Ls^2);
221 Ktv = num_Ktv/den_Ktv;
222

```

```

223 % Eq. 14.2-12 (Krause)
224 Is = sqrt(iqs_e_1^2 + ids_e_1^2)/sqrt(2);
225
226 % Eq. 14.2-14 (Krause)
227 we = (wr_star + sqrt(max([0 wr_star^2+X_corr])))/2;
228
229 % Eq 14.2-4 (Krause)
230 Vs = (wr_star ~= 0)*Vb*sqrt((ParametersIM.rs^2+we^2*ParametersIM.Ls^2)/(ParametersIM.rs^2+wb^2*ParametersIM.Ls^2));
231
232 % Saturation on Vs
233 Vs = min(Vb,Vs);
234
235 % Placing the voltage arbitrarily in the q axis
236 vqs_e_star = Vs*sqrt(2);
237 vds_e_star = 0;
238
239 % Eq 14.2-16 (Krause)
240 x_corr = 3*ParametersIM.P*(vqs_e_star*iqs_e_1 - 2*ParametersIM.rs*Is^2)/Ktv;
241
242 % Input of the LPF filter
243 u_LPF = x_corr;
244
245 % Transforming vds_e_star and vqs_e_star to vabcs
246 [vas_star, vbs_star, vcs_star] = qd2abcs(vqs_e_star,vds_e_star,theta_e);
247
248 % Converter (voltage control)
249 f_sw = 3e3; % inverter switching frequency
250 vdc = ParametersIM.Vs*sqrt(2); % vdc got by a 3-phase full-bridge rectifier
251 [S,~,~,~,~] = my3HarmonicSineTri_v2(t, theta_e, vqs_e_star, vds_e_star, vdc, f_sw);
252 IM = [(ias_1+ias_2+ias_3) (ibs_1+ibs_2+ibs_3) (ics_1+ics_2+ics_3)];
253 [vas, vbs, vcs] = my3PhaseConverterVoltLoss(S, vdc, vswM, vdM, IM);
254
255 % Ideal voltage source
256 % vas_1 = vas_star;
257 % vbs_1 = vbs_star;
258 % vcs_1 = vcs_star;
259
260 % Transforming vabcs to vds and vqs
261 [vqsC, vdsC] = abcs2qd(vas, vbs, vcs, theta_a_1);
262 %-----%
263 %-----%
264 %-----%
265 % STATOR RESISTANCE CONTROL
266 % INDUCTION MOTORS 2 & 3
267 %-----%
268 %-----%
269 %-----%
270 % CONTROLLER EQUATIONS
271 %-----%
272 % Controller input vector uk
273 % For now, only the position error
274 Ak = ParamController.Ak;
275 Bk = ParamController.Bk;
276 Ck = ParamController.Ck;
277 Dk = ParamController.Dk;
278
279 uk2(1) = theta_m_1 - theta_m_2;
280 xk2_dot = Ak*xk2 + Bk*uk2;
281 yk2 = Ck*xk2 + Dk*uk2;
282
283 uk3(1) = theta_m_1 - theta_m_3;
284 xk3_dot = Ak*xk3 + Bk*uk3;
285 yk3 = Ck*xk3 + Dk*uk3;
286
287 % Controller output vector yk
288 % For PI controller, yk is in units of 1/rad
289 rext2_cont = yk2(1)*ParametersIM.rs;
290 rext2_cont = max(rext2_cont,0);
291 rext2_cont = min(rext2_cont,ParamController.rext_max);
292
293 rext3_cont = yk3(1)*ParametersIM.rs;
294 rext3_cont = max(rext3_cont,0);
295 rext3_cont = min(rext3_cont,ParamController.rext_max);
296 %-----%
297
298 % Controlable external resistance via PWM
299 f_sw = 4.987654321e3;
300 r_full = ParamController.rext_max;
301 d2 = rext2_cont/r_full;
302 S_rext2 = myPWM(t, d2, f_sw);
303 rext2_pwm = S_rext2 * r_full;
304

```

```

305 d3 = rext3_cont/r_full;
306 S_rext3 = myPWM(t, d3, f_sw);
307 rext3_pwm = S_rext3 * r_full;
308
309 % PWM external resistance
310 rext2 = rext2_pwm;
311 rext3 = rext3_pwm;
312
313 % Ideal external resistance
314 % rext2 = rext2_cont;
315 % rext3 = rext3_cont;
316
317 % vqs_2 = vqs_1;
318 % vds_2 = vds_1;
319 % If external resistance is off, then current flows through switches. If
320 % these are non-ideal, then voltage drop is always 1 switch + 1 diode
321 vas_2 = vas - ~S_rext2*sign(ias_2)*(vswR+vdR);
322 vbs_2 = vbs - ~S_rext2*sign(ibs_2)*(vswR+vdR);
323 vcs_2 = vcs - ~S_rext2*sign(ics_2)*(vswR+vdR);
324 % Transforming vabcs to vds and vqs
325 [vqs_2, vds_2] = abcs2qd(vas_2, vbs_2, vcs_2, theta_a_2);
326
327 vas_3 = vas - ~S_rext3*sign(ias_3)*(vswR+vdR);
328 vbs_3 = vbs - ~S_rext3*sign(ibs_3)*(vswR+vdR);
329 vcs_3 = vcs - ~S_rext3*sign(ics_3)*(vswR+vdR);
330 % Transforming vabcs to vds and vqs
331 [vqs_3, vds_3] = abcs2qd(vas_3, vbs_3, vcs_3, theta_a_3);
332
333 % Apply this loss to motor line 1 too for parity. In a real system, both
334 % would be connected to a board in case torque difference reverses
335 vas_1 = vas - sign(ias_1)*(vswR+vdR);
336 vbs_1 = vbs - sign(ibs_1)*(vswR+vdR);
337 vcs_1 = vcs - sign(ics_1)*(vswR+vdR);
338 % Transforming vabcs to vds and vqs
339 [vqs_1, vds_1] = abcs2qd(vas_1, vbs_1, vcs_1, theta_a_1);
340 %-----%
341 %-----%
342 %-----%
343 %-----%
344 %-----%
345 % Switching states
346 Sw.S = S;
347 Sw.S_rext2 = S_rext2;
348 Sw.S_rext3 = S_rext3;
349
350 % 1st motor
351 IM1.vqsC = vqsC;
352 IM1.vdsC = vdsC;
353 IM1.TL = TL_1;
354 IM1.Te = Te_1;
355 IM1.vqs = vqs_1;
356 IM1.vds = vds_1;
357 IM1.we = we;
358 IM1.theta_e = theta_e;
359 IM1.theta_a = theta_a_1;
360
361 % 2nd motor
362 IM2.TL = TL_2;
363 IM2.Te = Te_2;
364 IM2.vqs = vqs_2;
365 IM2.vds = vds_2;
366 IM2.theta_a = theta_a_2;
367 IM2.rext_pwm = rext2_pwm;
368 IM2.rext_cont = rext2_cont;
369
370 % 3rd motor
371 IM3.TL = TL_3;
372 IM3.Te = Te_3;
373 IM3.vqs = vqs_3;
374 IM3.vds = vds_3;
375 IM3.theta_a = theta_a_3;
376 IM3.rext_pwm = rext3_pwm;
377 IM3.rext_cont = rext3_cont;
378
379 %-----%
380
381 %-----%
382 %-----%
383 %-----%
384 %-----%
385 %-----%
386 %-----%

```

```

387 y(1) = vds_1 - ParametersIM.rs*ids_1 + wa_1 * lambda_qs_1; % lambda_ds_1
388 y(2) = vqs_1 - ParametersIM.rs*iqs_1 - wa_1 * lambda_ds_1; % lambda_qs_1
389 y(3) = vdr_1 - ParametersIM.rr*idr_1 + (wa_1-wr_1)*lambda_qr_1; % lambda_dr_1
390 y(4) = vqr_1 - ParametersIM.rr*iqr_1 - (wa_1-wr_1)*lambda_dr_1; % lambda_qr_1
391 y(5) = (1/ParametersIM.J)*((Te_1-TL_1)*ParametersIM.P/2 + ...
392 -ParametersIM.b*wr_1); % wr_1
393 y(6) = wr_1; % theta_r_1
394 %
395 % COMPENSATED VOLTS-PER-HERTZ CONTROL
396 %
397 y(7) = we; % Electrical angular position
398 y(8) = (u_LPF - y_LPF)/tau_LPF; % Output of the low-pass filter
399 %
400 %
401 % 2ND MOTOR
402 %
403 y(9) = vds_2 - (ParametersIM.rs + rext2)*ids_2 + wa_2 * lambda_qs_2; % lambda_ds_2
404 y(10) = vqs_2 - (ParametersIM.rs + rext2)*iqs_2 - wa_2 * lambda_ds_2; % lambda_qs_2
405 y(11) = vdr_2 - ParametersIM.rr*idr_2 + (wa_2-wr_2)*lambda_qr_2; % lambda_dr_2
406 y(12) = vqr_2 - ParametersIM.rr*iqr_2 - (wa_2-wr_2)*lambda_dr_2; % lambda_qr_2
407 y(13) = (1/ParametersIM.J)*((Te_2-TL_2)*ParametersIM.P/2 + ...
408 -ParametersIM.b*wr_2); % wr_2
409 y(14) = wr_2; % theta_r_2
410 %
411 % 3RD MOTOR
412 %
413 y(15) = vds_3 - (ParametersIM.rs + rext3)*ids_3 + wa_3 * lambda_qs_3; % lambda_ds_3
414 y(16) = vqs_3 - (ParametersIM.rs + rext3)*iqs_3 - wa_3 * lambda_ds_3; % lambda_qs_3
415 y(17) = vdr_3 - ParametersIM.rr*idr_3 + (wa_3-wr_3)*lambda_qr_3; % lambda_dr_3
416 y(18) = vqr_3 - ParametersIM.rr*iqr_3 - (wa_3-wr_3)*lambda_dr_3; % lambda_qr_3
417 y(19) = (1/ParametersIM.J)*((Te_3-TL_3)*ParametersIM.P/2 + ...
418 -ParametersIM.b*wr_3); % wr_3
419 y(20) = wr_3; % theta_r_3
420 %
421 % RESISTANCE CONTROL OF 2ND & 3RD MOTOR
422 %
423 %
424 % CONTROLLER
425 %
426 % Automatic state-building diff. equations of the controller, based on its order
427 y(21:21+n-1) = xk2_dot;
428 y(21+n:21+2*n-1) = xk3_dot;
429 %
430
431 y = y';
432 end

```

B.2 CVHz Control, Auxiliary Converter Method

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Multirun_3IM_CVHZ_volt_xf.m
3 % Runs 3-motor simulation of auxiliary converter method using CVHz control
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 clear
6 close all
7 clc
8
9 % load k_hinf
10 % [Ak,Bk,Ck,Dk] = ssdata(k_hinf);
11
12 Kp = -80; % V/rad
13 Ki = -120; % V/(rad*s)
14 [Ak,Bk,Ck,Dk] = ssdata(tf([Kp Ki],[1 0]));
15
16 % Code currently only supports 3 identical motors
17 ParametersIM = ReturnParametersIM_15hp;
18
19 rs = ParametersIM.rs;
20 rr = ParametersIM.rr;
21 Lls = ParametersIM.Lls;
22 Llr = ParametersIM.Llr;
23 Lm = ParametersIM.Lm;
24 Ls = ParametersIM.Ls;
25 Lr = ParametersIM.Lr;
26 % sigma = 1 - Lm^2/(Ls*Lr);
27 % tau_s = Ls/rs;
28 % tau_r = Lr/rr;
29 % sigma_s = tau_s*sigma;
30 % sigma_r = tau_r*sigma;

```

```

31 % mu_s = 1/sigma_s;
32 % mu_r = 1/sigma_r;
33
34 % Incorporating mu_s inside the param structure
35 % ParametersIM.mu_s = mu_s;
36
37 ParamController.Ak = Ak;
38 ParamController.Bk = Bk;
39 ParamController.Ck = Ck;
40 ParamController.Dk = Dk;
41 ParamController.order = length(Ak);
42
43 ParamController.delv_max = 50;
44 ParamController.tstart = 2; % Time to turn on voltage drop converters
45
46 % Parameters for switching losses
47 ParamSys.vswM = 0; % Main converter switch voltage
48 ParamSys.vdM = 0; % Main converter diode voltage
49 ParamSys.vswD = 0; % Drop converter switch voltage
50 ParamSys.vdD = 0; % Drop converter diode voltage
51
52 time_variation_TL1 = 2;
53 time_variation_TL2 = 2;
54 time_variation_TL3 = 2;
55
56 ParamSys.time_variation_TL1 = time_variation_TL1;
57 ParamSys.time_variation_TL2 = time_variation_TL2;
58 ParamSys.time_variation_TL3 = time_variation_TL3;
59
60 TL1_variation = 0; % Variation in load torque in pu
61 ParamSys.TL1_variation = TL1_variation;
62
63 TL2_variation = -0.2;
64 ParamSys.TL2_variation = TL2_variation;
65
66 TL3_variation = -0.3;
67 ParamSys.TL3_variation = TL3_variation;
68
69 num_states = 26 + 2*ParamController.order;
70 x0 = zeros(1,num_states); % Initial conditions for state variables
71 % load x0_save
72 % x0 = x0_save;
73
74 t_ini = 0;
75 t_end = 6;
76
77 deltat = 10e-6;
78 tspan = t_ini:deltat:t_end;
79 options = odeset('MaxStep', deltat, 'RelTol', 1e-5, 'AbsTol', 1e-6);
80 % options = odeset();
81
82 % Set up iterations and allocate space for saved variables
83 Nlist = [1/5 1/3 1/2 1/2];
84 vdcRlist = [1 1 1 0.5];
85
86 TL_2_out = zeros(numel(tspan),numel(Nlist));
87 Te_2_out = TL_2_out;
88 delv2_vd_out = TL_2_out;
89 delv2_vq_out = TL_2_out;
90 delv2_cont_out = TL_2_out;
91 wm_2_out = TL_2_out;
92 theta_m_2_out = TL_2_out;
93 ids_2_out = TL_2_out;
94 iqs_2_out = TL_2_out;
95 idr_2_out = TL_2_out;
96 iqr_2_out = TL_2_out;
97 id2_2_out = TL_2_out;
98 iq2_2_out = TL_2_out;
99 vq1_2_out = TL_2_out;
100 vd1_2_out = TL_2_out;
101 vqs_2_out = TL_2_out;
102 vds_2_out = TL_2_out;
103 Sd1_2_out = TL_2_out;
104 Sd2_2_out = TL_2_out;
105 Sd3_2_out = TL_2_out;
106
107 TL_3_out = TL_2_out;
108 Te_3_out = TL_2_out;
109 delv3_vd_out = TL_2_out;
110 delv3_vq_out = TL_2_out;
111 delv3_cont_out = TL_2_out;
112 wm_3_out = TL_2_out;

```

```

113 theta_m_3_out = TL_2_out;
114 ids_3_out = TL_2_out;
115 iqs_3_out = TL_2_out;
116 idr_3_out = TL_2_out;
117 iqr_3_out = TL_2_out;
118 id2_3_out = TL_2_out;
119 iq2_3_out = TL_2_out;
120 vq1_3_out = TL_2_out;
121 vd1_3_out = TL_2_out;
122 vqs_3_out = TL_2_out;
123 vds_3_out = TL_2_out;
124 Sd1_3_out = TL_2_out;
125 Sd2_3_out = TL_2_out;
126 Sd3_3_out = TL_2_out;
127
128 TL_1_out = TL_2_out;
129 Te_1_out = TL_1_out;
130 wm_1_out = TL_1_out;
131 theta_m_1_out = TL_1_out;
132 ids_1_out = TL_1_out;
133 iqs_1_out = TL_1_out;
134 idr_1_out = TL_1_out;
135 iqr_1_out = TL_1_out;
136 id2_1_out = TL_1_out;
137 iq2_1_out = TL_1_out;
138 vq1_1_out = TL_1_out;
139 vd1_1_out = TL_1_out;
140 vqs_1_out = TL_1_out;
141 vds_1_out = TL_1_out;
142 S1_out = TL_1_out;
143 S2_out = TL_1_out;
144 S3_out = TL_1_out;
145
146 theta_e_out = TL_1_out;
147
148 % Iterate over transformer turns ratios & auxiliary converter bus voltages
149 for j=1:numel(Nlist)
150
151 % Transformer params: line side = primary (1), aux converter = secondary (2)
152 N = Nlist(j); % N1/N2
153 ParametersXF.N = N; % Z' = Z*N^2, but less windings = less impedance
154 ParametersXF.r1 = 0.001; % So actual secondary impedance scales with N2,
155 ParametersXF.r2 = 0.01*N; % canceling a power of N for this scaling
156 ParametersXF.Ll1 = 0.1e-3;
157 ParametersXF.Ll2 = 0.5e-3*N;
158 ParametersXF.LmT = 10e-3;
159 ParametersXF.vdcR = vdcRlist(j); % Ratio between aux converter bus and main bus
160
161 % Calculate inductance matrix inverses in setup
162 % The factor of 3/2 for magnetizing inductance (Lm -> M) is assumed included
163 Ll1 = ParametersXF.Ll1;
164 Ll2 = ParametersXF.Ll2;
165 LmT = ParametersXF.LmT;
166 L = [[Ll1+Ll1+Lm+LmT, Lm, LmT];...
167 [Lm, Ll1+Lm, 0];...
168 [LmT, 0, Ll2+LmT]];
169 ParametersXF.Linv = inv(L);
170
171 tic
172 [t,y] = ode45(@(t,y) ThreeIM_1Converter_CVHz_volt_xf(t,y,ParametersIM, ...
173 ParametersXF, ParamSys, ParamController), tspan, x0, options);
174 toc
175
176 % Save state
177 % x0_save = y(end,:);
178 % save x0_save x0_save
179
180 %-----%
181 % POST-PROCESSING
182 %-----%
183
184 %-----%
185 % PICKING UP THE STATES
186 %-----%
187 % 1ST MOTOR
188 %-----%
189 lambda_dC_1 = y(:,1); % Flux dC (main converter path) [Wb]
190 lambda_qC_1 = y(:,2); % Flux qC [Wb]
191 lambda_dr_1 = y(:,3); % Flux dr (rotor path) [Wb]
192 lambda_qr_1 = y(:,4); % Flux qr [Wb]
193 lambda_d2_1 = y(:,5); % Flux d2 (side converter path) [Wb]
194 lambda_q2_1 = y(:,6); % Flux q2 [Wb]

```

```

195 wr_1 = y(:,7); % Rotor speed (electric) [rad/s]
196 theta_r_1 = y(:,8); % Rotor position (electric) [rad]
197 %-----%
198 % COMPENSATED VOLTS-PER-HERZ CONTROL
199 %-----%
200 theta_e = y(:,9); % Electrical angular position [rad]
201 % y_LPF = y(:,10); % Output of the low-pass filter
202 %-----%
203 %-----%
204 % 2ND MOTOR
205 %-----%
206 lambda_dC_2 = y(:,11); % Flux dC (main converter path) [Wb]
207 lambda_qC_2 = y(:,12); % Flux qC [Wb]
208 lambda_dr_2 = y(:,13); % Flux dr (rotor path) [Wb]
209 lambda_qr_2 = y(:,14); % Flux qr [Wb]
210 lambda_d2_2 = y(:,15); % Flux d2 (side converter path) [Wb]
211 lambda_q2_2 = y(:,16); % Flux q2 [Wb]
212 wr_2 = y(:,17); % Rotor speed (electric) [rad/s]
213 theta_r_2 = y(:,18); % Rotor position (electric) [rad]
214 %-----%
215 %-----%
216 % 3RD MOTOR
217 %-----%
218 lambda_dC_3 = y(:,19); % Flux dC (main converter path) [Wb]
219 lambda_qC_3 = y(:,20); % Flux qC [Wb]
220 lambda_dr_3 = y(:,21); % Flux dr (rotor path) [Wb]
221 lambda_qr_3 = y(:,22); % Flux qr [Wb]
222 lambda_d2_3 = y(:,23); % Flux d2 (side converter path) [Wb]
223 lambda_q2_3 = y(:,24); % Flux q2 [Wb]
224 wr_3 = y(:,25); % Rotor speed (electric) [rad/s]
225 theta_r_3 = y(:,26); % Rotor position (electric) [rad]
226 %-----%
227 %-----%
228 % EXTRACTING OUTPUT STRUCTURES FROM DIFF EQ FUNCTION
229 %-----%
230 %-----%
231 [-, IM1_out, IM2_out, IM3_out, Sw_out] = cellfun(@(t,y) ...
232 ThreeIM_1Converter_CVHz_volt_xf(t, y, ParametersIM, ParametersXF, ...
233 ParamSys, ParamController), num2cell(t), num2cell(y,2), 'uni', 0);
234
235 TL_1 = zeros(numel(t),1);
236 Te_1 = zeros(numel(t),1);
237 % wa_1 = zeros(numel(t),1);
238 % theta_a_1 = zeros(numel(t),1);
239 vqC = zeros(numel(t),1);
240 vdC = zeros(numel(t),1);
241
242 TL_2 = zeros(numel(t),1);
243 Te_2 = zeros(numel(t),1);
244 % wa_2 = zeros(numel(t),1);
245 % theta_a_2 = zeros(numel(t),1);
246
247 delv2_vq = zeros(numel(t),1);
248 delv2_vd = zeros(numel(t),1);
249 delv2_cont = zeros(numel(t),1);
250
251 TL_3 = zeros(numel(t),1);
252 Te_3 = zeros(numel(t),1);
253 % wa_3 = zeros(numel(t),1);
254 % theta_a_3 = zeros(numel(t),1);
255
256 delv3_vq = zeros(numel(t),1);
257 delv3_vd = zeros(numel(t),1);
258 delv3_cont = zeros(numel(t),1);
259
260 S1 = zeros(numel(t),1);
261 S2 = zeros(numel(t),1);
262 S3 = zeros(numel(t),1);
263
264 Sd1_2 = zeros(numel(t),1);
265 Sd2_2 = zeros(numel(t),1);
266 Sd3_2 = zeros(numel(t),1);
267
268 Sd1_3 = zeros(numel(t),1);
269 Sd2_3 = zeros(numel(t),1);
270 Sd3_3 = zeros(numel(t),1);
271
272 for i=1:numel(t)
273 TL_1(i) = IM1_out{i}.TL;
274 Te_1(i) = IM1_out{i}.Te;
275 % wa_1(i) = IM1_out{i}.wa;
276 % theta_a_1(i) = IM1_out{i}.theta_a;

```

```

277
278     vqC(i) = IM1_out{i}.vqC;
279     vdC(i) = IM1_out{i}.vdC;
280
281     TL_2(i) = IM2_out{i}.TL;
282     Te_2(i) = IM2_out{i}.Te;
283     %     wa_2(i) = IM2_out{i}.wa;
284     %     theta_a_2(i) = IM2_out{i}.theta_a;
285
286     delv2_vq(i) = IM2_out{i}.delv2_vq;
287     delv2_vd(i) = IM2_out{i}.delv2_vd;
288     delv2_cont(i) = IM2_out{i}.delv2_cont;
289
290     TL_3(i) = IM3_out{i}.TL;
291     Te_3(i) = IM3_out{i}.Te;
292     %     wa_3(i) = IM2_out{i}.wa;
293     %     theta_a_3(i) = IM2_out{i}.theta_a;
294
295     delv3_vq(i) = IM3_out{i}.delv3_vq;
296     delv3_vd(i) = IM3_out{i}.delv3_vd;
297     delv3_cont(i) = IM3_out{i}.delv3_cont;
298
299     S1(i) = Sw_out{i}.S(1);
300     S2(i) = Sw_out{i}.S(2);
301     S3(i) = Sw_out{i}.S(3);
302
303     Sd1_2(i) = Sw_out{i}.S_d2(1);
304     Sd2_2(i) = Sw_out{i}.S_d2(2);
305     Sd3_2(i) = Sw_out{i}.S_d2(3);
306
307     Sd1_3(i) = Sw_out{i}.S_d3(1);
308     Sd2_3(i) = Sw_out{i}.S_d3(2);
309     Sd3_3(i) = Sw_out{i}.S_d3(3);
310
311 end
312
313 %-----%
314 clear y IM1_out IM2_out IM3_out Sw_out
315 %-----%
316 %           Auxiliary Equations for the IMs
317 %-----%
318
319 % Transformation of electrical rotor speed and electrical angular position
320 % to mechanical rotor speed and mechanical angular position
321
322 % 1st IM
323 wm_1 = wr_1/(ParametersIM.P/2);
324 theta_m_1 = theta_r_1/(ParametersIM.P/2);
325 % 2nd IM
326 wm_2 = wr_2/(ParametersIM.P/2);
327 theta_m_2 = theta_r_2/(ParametersIM.P/2);
328 % 3rd IM
329 wm_3 = wr_3/(ParametersIM.P/2);
330 theta_m_3 = theta_r_3/(ParametersIM.P/2);
331
332 % Transformation of fluxes to currents – DQ components
333 Linv = ParametersXF.Linv;
334 % 1ST MOTOR
335 [ids_1, iqs_1, idr_1, iqr_1, id2_1, iq2_1] = Fluxes2Currents_xf(lambda_dC_1, ...
336     lambda_qC_1, lambda_dr_1, lambda_qr_1, lambda_d2_1, lambda_q2_1, Linv);
337 % 2ND MOTOR
338 [ids_2, iqs_2, idr_2, iqr_2, id2_2, iq2_2] = Fluxes2Currents_xf(lambda_dC_2, ...
339     lambda_qC_2, lambda_dr_2, lambda_qr_2, lambda_d2_2, lambda_q2_2, Linv);
340 % 3RD MOTOR
341 [ids_3, iqs_3, idr_3, iqr_3, id2_3, iq2_3] = Fluxes2Currents_xf(lambda_dC_3, ...
342     lambda_qC_3, lambda_dr_3, lambda_qr_3, lambda_d2_3, lambda_q2_3, Linv);
343
344 % Transforming stator and rotor currents from qd reference to abc variables
345 % Disabled later to reduce size of saved data – can convert later
346 % % 1st IM
347 % % [ias_1, ibs_1, ics_1] = qd2abcs(iqs_1, ids_1, theta_a_1);
348 % % ias_1 = iqs_1;
349 % % [ibs_1, ics_1] = statbc(iqs_1, ids_1);
350 % % [iar_1, ~, ~] = qd2abcs(iqr_1, idr_1, theta_a_1–theta_r_1);
351 % % [ia2_1, ib2_1, ic2_1] = qd2abcs(iq2_1, id2_1, theta_a_1);
352 % % ia2_1 = iq2_1;
353 % % [ib2_1, ic2_1] = statbc(iq2_1, id2_1);
354 % % 2nd IM
355 % % [ias_2, ibs_2, ics_2] = qd2abcs(iqs_2, ids_2, theta_a_2);
356 % % ias_2 = iqs_2;
357 % % [ibs_2, ics_2] = statbc(iqs_2, ids_2);
358 % % [iar_2, ~, ~] = qd2abcs(iqr_2, idr_2, theta_a_2–theta_r_2);

```

```

359 %% [ia2_2, ib2_2, ic2_2] = qd2abcs(iq2_2, id2_2, theta_a_2);
360 % ia2_2 = iq2_2;
361 % [ib2_2, ic2_2] = statbc(iq2_2, id2_2);
362 %
363 %% Transforming stator voltages from qd reference to abc variables
364 %% Main converter
365 %% [vaC, vbC, vcC] = qd2abcs(vqC, vdC, theta_a_1);
366 % vaC = vqC;
367 % [vbC, vcC] = statbc(vqC, vdC);
368 %% Voltage drop converter
369 %% [delv2_va, delv2_vb, delv2_vc] = qd2abcs(delv2_vq, delv2_vd, theta_a_2);
370 % delv2_va = delv2_vq;
371 % [delv2_vb, delv2_vc] = statbc(delv2_vq, delv2_vd);
372 %
373 % Voltage reaching motors: Need to calculate based on currents. v=Ldi/dt
374 % 1st IM
375 Dids_1 = [0; diff(ids_1)/deltat];
376 Diqs_1 = [0; diff(iqs_1)/deltat];
377 Did2_1 = [0; diff(id2_1)/deltat];
378 Diq2_1 = [0; diff(iq2_1)/deltat];
379 %
380 % lambda_d1_1 = (Ll1 + LmT)*ids_1 + LmT*id2_1;
381 % lambda_q1_1 = (Ll1 + LmT)*iqs_1 + LmT*iq2_1;
382 %
383 % vd1_1 = ParametersXF.r1*ids_1 + Ll1*Dids_1 + LmT*(Dids_1+Did2_1) - wa_1.*lambda_q1_1;
384 vd1_1 = ParametersXF.r1*ids_1 + Ll1*Dids_1 + LmT*(Dids_1+Did2_1);
385 vds_1 = vdC - vd1_1;
386 % vq1_1 = ParametersXF.r1*iqs_1 + Ll1*Diqs_1 + LmT*(Diqs_1+Diq2_1) + wa_1.*lambda_d1_1;
387 vq1_1 = ParametersXF.r1*iqs_1 + Ll1*Diqs_1 + LmT*(Diqs_1+Diq2_1);
388 vqs_1 = vqC - vq1_1;
389 %
390 %% [va1_1, vb1_1, vc1_1] = qd2abcs(vq1_1, vd1_1, theta_a_1);
391 % va1_1 = vq1_1;
392 % [vb1_1, vc1_1] = statbc(vq1_1, vd1_1);
393 %% [vas_1, vbs_1, vcs_1] = qd2abcs(vqs_1, vds_1, theta_a_1);
394 % vas_1 = vqs_1;
395 % [vbs_1, vcs_1] = statbc(vqs_1, vds_1);
396 %
397 % 2nd IM
398 Dids_2 = [0; diff(ids_2)/deltat];
399 Diqs_2 = [0; diff(iqs_2)/deltat];
400 Did2_2 = [0; diff(id2_2)/deltat];
401 Diq2_2 = [0; diff(iq2_2)/deltat];
402 %
403 % lambda_d1_2 = (Ll1 + LmT)*ids_2 + LmT*id2_2;
404 % lambda_q1_2 = (Ll1 + LmT)*iqs_2 + LmT*iq2_2;
405 %
406 % vd1_2 = ParametersXF.r1*ids_2 + Ll1*Dids_2 + LmT*(Dids_2+Did2_2) - wa_2.*lambda_q1_2;
407 vd1_2 = ParametersXF.r1*ids_2 + Ll1*Dids_2 + LmT*(Dids_2+Did2_2);
408 vds_2 = vdC - vd1_2;
409 % vq1_2 = ParametersXF.r1*iqs_2 + Ll1*Diqs_2 + LmT*(Diqs_2+Diq2_2) + wa_2.*lambda_d1_2;
410 vq1_2 = ParametersXF.r1*iqs_2 + Ll1*Diqs_2 + LmT*(Diqs_2+Diq2_2);
411 vqs_2 = vqC - vq1_2;
412 %
413 %% [va1_2, vb1_2, vc1_2] = qd2abcs(vq1_2, vd1_2, theta_a_2);
414 % va1_2 = vq1_2;
415 % [vb1_2, vc1_2] = statbc(vq1_2, vd1_2);
416 %% [vas_2, vbs_2, vcs_2] = qd2abcs(vqs_2, vds_2, theta_a_2);
417 % vas_2 = vqs_2;
418 % [vbs_2, vcs_2] = statbc(vqs_2, vds_2);
419 %
420 % 3rd IM
421 Dids_3 = [0; diff(ids_3)/deltat];
422 Diqs_3 = [0; diff(iqs_3)/deltat];
423 Did2_3 = [0; diff(id2_3)/deltat];
424 Diq2_3 = [0; diff(iq2_3)/deltat];
425 %
426 vd1_3 = ParametersXF.r1*ids_3 + Ll1*Dids_3 + LmT*(Dids_3+Did2_3);
427 vds_3 = vdC - vd1_3;
428 vq1_3 = ParametersXF.r1*iqs_3 + Ll1*Diqs_3 + LmT*(Diqs_3+Diq2_3);
429 vqs_3 = vqC - vq1_3;
430 %-----%
431 %
432 % Clear variables not saved
433 cclear lambda_dC_1 lambda_qC_1 lambda_dr_1 lambda_qr_1 lambda_d2_1 ...
434 lambda_q2_1 lambda_dC_2 lambda_qC_2 lambda_dr_2 lambda_qr_2 ...
435 lambda_d2_2 lambda_q2_2 lambda_dC_3 lambda_qC_3 lambda_dr_3 ...
436 lambda_qr_3 lambda_d2_3 lambda_q2_3 theta_r_1 theta_r_2 theta_r_3 ...
437 wr_1 wr_2 wr_3 Dids_1 Diqs_1 Did2_1 Diq2_1 Dids_2 Diqs_2 Did2_2 ...
438 Diq2_2 Dids_3 Diqs_3 Did2_3 Diq2_3
439 %
440 % Save variables before next loop (motor 2, different xfmr params)

```

```

441 TL_2_out(:,j) = TL_2;
442 Te_2_out(:,j) = Te_2;
443 delv2_vd_out(:,j) = delv2_vd;
444 delv2_vq_out(:,j) = delv2_vq;
445 delv2_cont_out(:,j) = delv2_cont;
446 wm_2_out(:,j) = wm_2;
447 theta_m_2_out(:,j) = theta_m_2;
448 ids_2_out(:,j) = ids_2;
449 iqs_2_out(:,j) = iqs_2;
450 idr_2_out(:,j) = idr_2;
451 iqr_2_out(:,j) = iqr_2;
452 id2_2_out(:,j) = id2_2;
453 iq2_2_out(:,j) = iq2_2;
454 vq1_2_out(:,j) = vq1_2;
455 vd1_2_out(:,j) = vd1_2;
456 vqs_2_out(:,j) = vqs_2;
457 vds_2_out(:,j) = vds_2;
458 Sd1_2_out(:,j) = Sd1_2;
459 Sd2_2_out(:,j) = Sd2_2;
460 Sd3_2_out(:,j) = Sd3_2;
461
462 % Save variables before next loop (motor 3, different xfmr params)
463 TL_3_out(:,j) = TL_3;
464 Te_3_out(:,j) = Te_3;
465 delv3_vd_out(:,j) = delv3_vd;
466 delv3_vq_out(:,j) = delv3_vq;
467 delv3_cont_out(:,j) = delv3_cont;
468 wm_3_out(:,j) = wm_3;
469 theta_m_3_out(:,j) = theta_m_3;
470 ids_3_out(:,j) = ids_3;
471 iqs_3_out(:,j) = iqs_3;
472 idr_3_out(:,j) = idr_3;
473 iqr_3_out(:,j) = iqr_3;
474 id2_3_out(:,j) = id2_3;
475 iq2_3_out(:,j) = iq2_3;
476 vq1_3_out(:,j) = vq1_3;
477 vd1_3_out(:,j) = vd1_3;
478 vqs_3_out(:,j) = vqs_3;
479 vds_3_out(:,j) = vds_3;
480 Sd1_3_out(:,j) = Sd1_3;
481 Sd2_3_out(:,j) = Sd2_3;
482 Sd3_3_out(:,j) = Sd3_3;
483
484 % Save variables before next loop (motor 1, different xfmr params)
485 TL_1_out(:,j) = TL_1;
486 Te_1_out(:,j) = Te_1;
487 wm_1_out(:,j) = wm_1;
488 theta_m_1_out(:,j) = theta_m_1;
489 ids_1_out(:,j) = ids_1;
490 iqs_1_out(:,j) = iqs_1;
491 idr_1_out(:,j) = idr_1;
492 iqr_1_out(:,j) = iqr_1;
493 id2_1_out(:,j) = id2_1;
494 iq2_1_out(:,j) = iq2_1;
495 vq1_1_out(:,j) = vq1_1;
496 vd1_1_out(:,j) = vd1_1;
497 vqs_1_out(:,j) = vqs_1;
498 vds_1_out(:,j) = vds_1;
499 S1_out(:,j) = S1;
500 S2_out(:,j) = S2;
501 S3_out(:,j) = S3;
502
503 theta_e_out(:,j) = theta_e;
504
505 end
506
507 % Clear redundant variables left over
508 clear delv2_cont delv2_vd delv2_vq delv3_cont delv3_vd delv3_vq i id2_1 ...
509     id2_2 id2_3 idr_1 idr_2 idr_3 ids_1 ids_2 ids_3 iq2_1 iq2_2 iq2_3 ...
510     iqr_1 iqr_2 iqr_3 iqs_1 iqs_2 iqs_3 j k N num_states S1 S2 S3 ...
511     Sd1_2 Sd2_2 Sd3_2 Sd1_3 Sd2_3 Sd3_3 t_end t_ini tspan Te_1 Te_2 Te_3 ...
512     theta_m_1 theta_m_2 theta_m_3 TL_1 TL_2 TL_3 vd1_1 vd1_2 vd1_3 ...
513     vdC vds_1 vds_2 vds_3 vq1_1 vq1_2 vq1_3 vqC vqs_1 vqs_2 vqs_3 ...
514     wm_1 wm_2 wm_3 x0 deltat theta_e
515 %-----%

1 function [y, IM1, IM2, IM3, Sw] = ThreeIM_1Converter_CVHz_volt_xf(t, u, ...
2     ParametersIM, ParametersXF, ParamSys, ParamController)
3 %-----%
4 %                               STATES
5 %-----%
6 %                               1ST MOTOR

```

```

7 %-----%
8 lambda_dC_1 = u(1); % Flux dC (ds + d1) [Wb]
9 lambda_qC_1 = u(2); % Flux qC (qs + q1) [Wb]
10 lambda_dr_1 = u(3); % Flux dr [Wb]
11 lambda_qr_1 = u(4); % Flux qr [Wb]
12 lambda_d2_1 = u(5); % Flux d2 (xfmr side converter) [Wb]
13 lambda_q2_1 = u(6); % Flux q2 [Wb]
14 wr_1 = u(7); % Rotor speed (electric) [rad/s]
15 theta_r_1 = u(8); % Rotor position (electric) [rad]
16 %-----%
17 COMPENSATED VOLTS-PER-HERZ CONTROL
18 %-----%
19 theta_e = u(9); % Electrical angular position [rad]
20 y_LPF = u(10); % Output of the low-pass filter
21 %-----%
22 %-----%
23 2ND MOTOR
24 %-----%
25 lambda_dC_2 = u(11); % Flux dC (ds + d1) [Wb]
26 lambda_qC_2 = u(12); % Flux qC (qs + q1) [Wb]
27 lambda_dr_2 = u(13); % Flux dr [Wb]
28 lambda_qr_2 = u(14); % Flux qr [Wb]
29 lambda_d2_2 = u(15); % Flux d2 (xfmr side converter) [Wb]
30 lambda_q2_2 = u(16); % Flux q2 [Wb]
31 wr_2 = u(17); % Rotor speed (electric) [rad/s]
32 theta_r_2 = u(18); % Rotor position (electric) [rad]
33 %-----%
34 %-----%
35 3RD MOTOR
36 %-----%
37 lambda_dC_3 = u(19); % Flux dC (ds + d1) [Wb]
38 lambda_qC_3 = u(20); % Flux qC (qs + q1) [Wb]
39 lambda_dr_3 = u(21); % Flux dr [Wb]
40 lambda_qr_3 = u(22); % Flux qr [Wb]
41 lambda_d2_3 = u(23); % Flux d2 (xfmr side converter) [Wb]
42 lambda_q2_3 = u(24); % Flux q2 [Wb]
43 wr_3 = u(25); % Rotor speed (electric) [rad/s]
44 theta_r_3 = u(26); % Rotor position (electric) [rad]
45 %-----%
46 %-----%
47 CONTROLLER
48 %-----%
49 n = ParamController.order;
50 % Automatic state-building of the controller, based on its order
51 xk2 = u(27:27+n-1);
52 xk2 = reshape(xk2,n,1);
53 xk3 = u(27+n:27+2*n-1);
54 xk3 = reshape(xk3,n,1);
55 %-----%
56 %-----%
57 GET SYSPARAMS
58 %-----%
59 %-----%
60 %-----%
61 %-----%
62 vswM = ParamSys.vswM;
63 vdM = ParamSys.vdM;
64 vswD = ParamSys.vswD;
65 vdD = ParamSys.vdD;
66
67 TL1_delta = ParamSys.TL1_variation;
68 time_delta_TL1 = ParamSys.time_variation_TL1;
69
70 TL2_delta = ParamSys.TL2_variation;
71 time_delta_TL2 = ParamSys.time_variation_TL2;
72
73 TL3_delta = ParamSys.TL3_variation;
74 time_delta_TL3 = ParamSys.time_variation_TL3;
75
76 if t > time_delta_TL1
77     TL_1 = (1+TL1_delta)*ParametersIM.Tnom;
78 else
79     % TL_1 = ParametersIM.Tnom;
80     TL_1 = 0;
81 end
82
83 if t > time_delta_TL2
84     TL_2 = (1+TL2_delta)*ParametersIM.Tnom;
85 else
86     % TL_2 = ParametersIM.Tnom;
87     TL_2 = 0;
88 end

```

```

89
90 if t > time_delta_TL3
91     TL_3 = (1+TL3_delta)*ParametersIM.Tnom;
92 else
93     % TL_3 = ParametersIM.Tnom;
94     TL_3 = 0;
95 end
96 %-----%
97
98 %-----%
99 % Arbitrary Reference Frame (unused)
100 %-----%
101 % Stator reference
102 %-----%
103 % 1st IM
104 % wa_1 = 0;
105 % theta_a_1 = 0;
106
107 % 2nd IM
108 % wa_2 = 0;
109 % theta_a_2 = 0;
110 %-----%
111
112 %-----%
113 % Auxiliary Equations for the IMs
114 %-----%
115 % Transformation of fluxes to currents – DQ components
116 Linv = ParametersXF.Linv;
117 % 1ST MOTOR
118 [ids_1, iqs_1, idr_1, iqr_1, id2_1, iq2_1] = Fluxes2Currents_xf(lambda_dC_1, ...
119     lambda_qC_1, lambda_dr_1, lambda_qr_1, lambda_d2_1, lambda_q2_1, Linv);
120 % 2ND MOTOR
121 [ids_2, iqs_2, idr_2, iqr_2, id2_2, iq2_2] = Fluxes2Currents_xf(lambda_dC_2, ...
122     lambda_qC_2, lambda_dr_2, lambda_qr_2, lambda_d2_2, lambda_q2_2, Linv);
123 % 3RD MOTOR
124 [ids_3, iqs_3, idr_3, iqr_3, id2_3, iq2_3] = Fluxes2Currents_xf(lambda_dC_3, ...
125     lambda_qC_3, lambda_dr_3, lambda_qr_3, lambda_d2_3, lambda_q2_3, Linv);
126
127 % Electromagnetic and Load Torques
128 % Need to use rotor flux & current since stator flux isn't calculated
129 % 1ST MOTOR
130 Te_1 = 3/4*ParametersIM.P*(lambda_qr_1.*idr_1 - lambda_dr_1.*iqr_1);
131 % 2ND MOTOR
132 Te_2 = 3/4*ParametersIM.P*(lambda_qr_2.*idr_2 - lambda_dr_2.*iqr_2);
133 % 3RD MOTOR
134 Te_3 = 3/4*ParametersIM.P*(lambda_qr_3.*idr_3 - lambda_dr_3.*iqr_3);
135
136 % Transformation of electrical rotor speed and electrical angular position
137 % to mechanical rotor speed and mechanical angular position
138 % 1st IM
139 % wm_1 = wr_1/(ParametersIM.P/2);
140 theta_m_1 = theta_r_1/(ParametersIM.P/2);
141 % 2nd IM
142 % wm_2 = wr_2/(ParametersIM.P/2);
143 theta_m_2 = theta_r_2/(ParametersIM.P/2);
144 % 3rd IM
145 % wm_2 = wr_2/(ParametersIM.P/2);
146 theta_m_3 = theta_r_3/(ParametersIM.P/2);
147
148 % Transforming stator and rotor currents from qd reference to abc variables
149 % Replaced with stationary assumption for efficiency
150 % 1st IM
151 % [ias_1, ibs_1, ics_1] = qd2abcs(iqs_1, ids_1, theta_a_1);
152 [ibs_1, ics_1] = statbc(iqs_1, ids_1);
153 % 2nd IM
154 % [ias_2, ibs_2, ics_2] = qd2abcs(iqs_2, ids_2, theta_a_2);
155 [ibs_2, ics_2] = statbc(iqs_2, ids_2);
156 % 3rd IM
157 % [ias_3, ibs_3, ics_3] = qd2abcs(iqs_3, ids_3, theta_a_3);
158 [ibs_3, ics_3] = statbc(iqs_3, ids_3);
159
160 % 1st XFMR
161 % [ia2_1, ib2_1, ic2_1] = qd2abcs(iq2_1, id2_1, theta_a_1);
162 [ib2_1, ic2_1] = statbc(iq2_1, id2_1);
163 % 2nd XFMR
164 % [ia2_2, ib2_2, ic2_2] = qd2abcs(iq2_2, id2_2, theta_a_2);
165 [ib2_2, ic2_2] = statbc(iq2_2, id2_2);
166 % 3rd XFMR
167 % [ia2_3, ib2_3, ic2_3] = qd2abcs(iq2_3, id2_3, theta_a_3);
168 [ib2_3, ic2_3] = statbc(iq2_3, id2_3);
169
170 % Rotor Voltages (squirrel cage) – DQ components

```

```

171 % 1st IM
172 vdr_1 = 0;
173 vqr_1 = 0;
174 % 2nd IM
175 vdr_2 = 0;
176 vqr_2 = 0;
177 % 3rd IM
178 vdr_3 = 0;
179 vqr_3 = 0;
180
181 % Transformer secondary voltages are defined in position sync section
182 %-----%
183
184 %-----%
185 %
186 % Low-Pass Filter Definitions/Equations
187 %-----%
188 % The signal x_corr = 3*P*(vqs_e_star*iqs_e - 2*rs*Is^2)/Ktv (Eq. 14.2-16
189 % from Krause) is filtered using a low-pass filter (LPF), accordingly to
190 % Krause's suggestion. In order to implement this filtering process, a new
191 % state, y, was created, from the diff. equation of a LPF:
192 % dy/dt = (x-y)/tau,
193 % where x is the filter input and y is the filter output, while tau is the
194 % filter time constant.
195 % The LPF output is the signal X_corr, used in the Volts-per-Hertz
196 % compensated control section down below.
197 % The input of the filter is set in the Volts-per-Hertz compensated control
198 % section down below, since, as seen above, it depends on vqs_e_star, which
199 % in turn is defined by the Volts-per-Hertz compensated control equations.
200 %-----%
201 % Time constant of the filter
202 tau_LPF = 0.1;
203
204 % Output of the LPF filter
205 X_corr = y_LPF;
206 %-----%
207
208 %-----%
209 % Compensated Volts-per-Hertz Control
210 %-----%
211
212 % Transforming from iqds_s to iqds_e
213 [iqs_e_1, ids_e_1] = qd2qd(iqs_1, ids_1, theta_e);
214
215 % Rated voltage (line to neutral rms) and rated radian frequency
216 Vb = ParametersIM.Vs/sqrt(3);
217 wb = ParametersIM.f*2*pi;
218
219 % Commanded mechanical speed
220 wrm_star = DesiredSpeed(t);
221
222 % Mechanical commanded speed to electrical commanded speed
223 wr_star = wrm_star*ParametersIM.P/2;
224
225 % Eq 14.2-7 (Krause)
226 % Equation from Krause was wrong (commented version). The right one, after
227 % checking 2 times, with 2 different approaches, is the one used here!
228 num_Ktv = 3*ParametersIM.P/2*ParametersIM.Lm^2*ParametersIM.rr*Vb^2;
229 den_Ktv = ParametersIM.rr*2*(ParametersIM.rs^2+wb^2*ParametersIM.Ls^2);
230 num_Ktv = 3*ParametersIM.P*ParametersIM.Lm^2*Vb^2;
231 den_Ktv = ParametersIM.rr*2*(ParametersIM.rs^2+wb^2*ParametersIM.Ls^2);
232 Ktv = num_Ktv/den_Ktv;
233
234 % Eq. 14.2-12 (Krause)
235 Is = sqrt(iqs_e_1^2 + ids_e_1^2)/sqrt(2);
236
237 % Eq. 14.2-14 (Krause)
238 we = (wr_star + sqrt(max([0 wr_star^2+X_corr]))) / 2;
239
240 % Eq 14.2-4 (Krause)
241 Vs = (wr_star ~= 0)*Vb*sqrt((ParametersIM.rs^2+we^2*ParametersIM.Ls^2)/(ParametersIM.rs^2+wb^2*ParametersIM.Ls^2));
242
243 % Saturation on Vs
244 Vs = min(Vb,Vs);
245
246 % Placing the voltage arbitrarily in the q axis
247 vqs_e_star = Vs*sqrt(2);
248 vds_e_star = 0;
249
250 % Eq 14.2-16 (Krause)
251 x_corr = 3*ParametersIM.P*(vqs_e_star*iqs_e_1 - 2*ParametersIM.rs*Is^2)/Ktv;
252
253 % Input of the LPF filter

```

```

253 u_LPF = x_corr;
254
255 % Transforming vds_e_star and vqs_e_star to vabcs
256 % [vas_star, vbs_star, vcs_star] = qd2abcs(vqs_e_star,vds_e_star,theta_e);
257
258 % Converter (voltage control)
259 f_sw = 3e3; % inverter switching frequency
260 vdc = ParametersIM.Vs*sqrt(2); % vdc got by a 3-phase full-bridge rectifier
261 [S,~,~,~] = my3HarmonicSineTri_v2(t, theta_e, vqs_e_star, vds_e_star, vdc, f_sw);
262 % IM = [(ias_1+ias_2) (ibs_1+ibs_2) (ics_1+ics_2)];
263 IM = [(iqs_1+iqs_2+iqs_3) (ibs_1+ibs_2+ibs_3) (ics_1+ics_2+ics_3)];
264 [vaC_1, vbC_1, vcC_1] = my3PhaseConverterVoltLoss(S, vdc, vswM, vdM, IM);
265
266 % Ideal voltage source
267 % vas_1 = vas_star;
268 % vbs_1 = vbs_star;
269 % vcs_1 = vcs_star;
270
271 % Transforming vabcs to vds and vqs
272 % Replaced with stationary assumption for efficiency
273 % [vqC_1, vdC_1] = abcs2qd(vaC_1, vbC_1, vcC_1, theta_a_1);
274 [vqC_1, vdC_1] = statqd(vaC_1, vbC_1, vcC_1);
275 %-----%
276 %-----%
277 %
278 %             VOLTAGE ADJUSTMENT CONTROL
279 %             INDUCTION MOTOR 2
280 %-----%
281 %-----%
282 %
283 %             CONTROLLER EQUATIONS
284 %-----%
285 % Controller input vector uk
286 % For now, only the position error
287
288 Ak = ParamController.Ak;
289 Bk = ParamController.Bk;
290 Ck = ParamController.Ck;
291 Dk = ParamController.Dk;
292
293 uk2(1) = theta_m_1 - theta_m_2;
294
295 xk2_dot = Ak*xk2 + Bk*uk2;
296 yk2 = Ck*xk2 + Dk*uk2;
297
298 uk3(1) = theta_m_1 - theta_m_3;
299
300 xk3_dot = Ak*xk3 + Bk*uk3;
301 yk3 = Ck*xk3 + Dk*uk3;
302
303 % Controller output vector yk
304 % For now, only motor 2/3 voltage adjustment
305 % delv2_cont: continuous variable voltage drop, ideal target
306 % For PI controller, yk is in units of V/rad
307 delv2_cont = yk2(1);
308 delv2_cont = max(delv2_cont,0);
309 delv2_cont = min(delv2_cont,ParamController.delv_max);
310
311 delv3_cont = yk3(1);
312 delv3_cont = max(delv3_cont,0);
313 delv3_cont = min(delv3_cont,ParamController.delv_max);
314 %-----%
315
316 % Calculate voltage drops for motor 1 using dummy converter for symmetry
317 S_d2 = [false false false true true];
318 % ID = [ia2_1, ib2_1, ic2_1];
319 ID = [iq2_1, ib2_1, ic2_1];
320 [vad, vbd, vcd] = my3PhaseConverterVoltLoss(S_d2, 0, vswD, vdD, ID);
321 % Transforming vabcs to vds and vqs
322 % Replaced with stationary assumption for efficiency
323 % [vqd, vdd] = abcs2qd(vad, vbd, vcd, theta_a_1);
324 [vqd_1, vdd_1] = statqd(vad, vbd, vcd);
325 % Refer variable
326 vq2_1 = (vqd_1*ParametersXF.N);
327 vd2_1 = (vdd_1*ParametersXF.N);
328 if t > ParamController.tstart
329     % Controllable voltage drop via sine-triangle PWM, assuming synchronization
330     f_d = 4.987654321e3;
331     vdc_d = vdc * ParametersXF.vdcR;
332
333     % Motor 2
334     vqd_e = delv2_cont / ParametersXF.N;

```

```

335     vdd_e = 0;
336     [S_d2,~,~,~] = my3HarmonicSineTri_v2(t, theta_e, vqd_e, vdd_e, vdc_d, f_d);
337     % ID = [ia2_2, ib2_2, ic2_2];
338     ID = [iq2_2, ib2_2, ic2_2];
339     [vad, vbd, vcd] = my3PhaseConverterVoltLoss(S_d2, vdc_d, vswD, vdD, ID);
340
341     % Transforming vabcs to vds and vqs
342     % Replaced with stationary assumption for efficiency
343     % [vqd, vdd] = abcs2qd(vad, vbd, vcd, theta_a_2);
344     [vqd_2, vdd_2] = statqd(vad, vbd, vcd);
345
346     % Refer variable
347     vq2_2 = (vqd_2*ParametersXF.N);
348     vd2_2 = (vdd_2*ParametersXF.N);
349
350     % Motor 3
351     vqd_e = delv3_cont / ParametersXF.N;
352     vdd_e = 0;
353     [S_d3,~,~,~] = my3HarmonicSineTri_v2(t, theta_e, vqd_e, vdd_e, vdc_d, f_d);
354     % ID = [ia2_3, ib2_3, ic2_3];
355     ID = [iq2_3, ib2_3, ic2_3];
356     [vad, vbd, vcd] = my3PhaseConverterVoltLoss(S_d3, vdc_d, vswD, vdD, ID);
357
358     % Transforming vabcs to vds and vqs
359     % Replaced with stationary assumption for efficiency
360     % [vqd, vdd] = abcs2qd(vad, vbd, vcd, theta_a_2);
361     [vqd_3, vdd_3] = statqd(vad, vbd, vcd);
362
363     % Refer variable
364     vq2_3 = (vqd_3*ParametersXF.N);
365     vd2_3 = (vdd_3*ParametersXF.N);
366 else
367     % Voltage drop is only active when torques are different
368     % Since motors are identical in this state, use same voltages
369     S_d3 = S_d2;
370     vqd_2 = vqd_1;
371     vdd_2 = vdd_1;
372     vqd_3 = vqd_1;
373     vdd_3 = vdd_1;
374     vq2_2 = vq2_1;
375     vd2_2 = vd2_1;
376     vq2_3 = vq2_1;
377     vd2_3 = vd2_1;
378 end
379 %-----%
380
381 %-----%
382 %                               OUTPUT STRUCTURES
383 %-----%
384 % Switching states
385 Sw.S = S;
386 Sw.S_d2 = S_d2;
387 Sw.S_d3 = S_d3;
388
389 % 1st motor
390 IM1.TL = TL_1;
391 IM1.Te = Te_1;
392 IM1.vqC = vqC_1;
393 IM1.vdC = vdC_1;
394 IM1.we = we;
395 % IM1.wa = wa_1;
396 IM1.theta_e = theta_e;
397 % IM1.theta_a = theta_a_1;
398 IM1.delv1_vq = vqd_1;
399 IM1.delv1_vd = vdd_1;
400
401 % 2nd motor
402 IM2.TL = TL_2;
403 IM2.Te = Te_2;
404 % IM2.wa = wa_2;
405 % IM2.theta_a = theta_a_2;
406 IM2.delv2_vq = vqd_2;
407 IM2.delv2_vd = vdd_2;
408 IM2.delv2_cont = delv2_cont;
409
410 % 3rd motor
411 IM3.TL = TL_3;
412 IM3.Te = Te_3;
413 % IM3.wa = wa_3;
414 % IM3.theta_a = theta_a_3;
415 IM3.delv3_vq = vqd_3;
416 IM3.delv3_vd = vdd_3;

```

```

417 IM3.delv3_cont = delv3_cont;
418
419 %-----%
420
421 %-----%
422 %
423 % DIFF. STATES EQUATIONS
424 %-----%
425 % Extracting parameters here, assuming identical motors
426 rs = ParametersIM.rs;
427 rr = ParametersIM.rr;
428 r1 = ParametersXF.r1;
429 r2 = ParametersXF.r2;
430 J = ParametersIM.J;
431 P = ParametersIM.P;
432 b = ParametersIM.b;
433
434 % Equations used are altered to assume stationary ref. frame for efficiency
435 %-----%
436 % 1ST MOTOR
437 %-----%
438 % y(1) = vdC_1 - (rs+r1)*ids_1 + wa_1 * lambda_qC_1; % lambda_dC_1
439 % y(2) = vqC_1 - (rs+r1)*iqs_1 - wa_1 * lambda_dC_1; % lambda_qC_1
440 % y(3) = vdr_1 - rr*idr_1 + (wa_1-wr_1)*lambda_qr_1; % lambda_dr_1
441 % y(4) = vqr_1 - rr*iqr_1 - (wa_1-wr_1)*lambda_dr_1; % lambda_qr_1
442 % y(5) = vd2_1 - r2*id2_1 + wa_1 * lambda_q2_1; % lambda_d2_1
443 % y(6) = vq2_1 - r2*iq2_1 - wa_1 * lambda_d2_1; % lambda_q2_1
444 y(1) = vdC_1 - (rs+r1)*ids_1; % lambda_dC_1
445 y(2) = vqC_1 - (rs+r1)*iqs_1; % lambda_qC_1
446 y(3) = vdr_1 - rr*idr_1 - wr_1*lambda_qr_1; % lambda_dr_1
447 y(4) = vqr_1 - rr*iqr_1 + wr_1*lambda_dr_1; % lambda_qr_1
448 y(5) = vd2_1 - r2*id2_1; % lambda_d2_1
449 y(6) = vq2_1 - r2*iq2_1; % lambda_q2_1
450 y(7) = (1/J)*((Te_1-TL_1)*P/2 - b*wr_1); % wr_1
451 y(8) = wr_1; % theta_r_1
452 %-----%
453 % COMPENSATED VOLTS-PER-HERTZ CONTROL
454 %-----%
455 y(9) = we; % Electrical angular position
456 y(10) = (u_LPF - y_LPF)/tau_LPF; % Output of the low-pass filter
457 %-----%
458 %
459 % 2ND MOTOR
460 %-----%
461 % y(11) = vdC_1 - (rs+r1)*ids_2 + wa_2 * lambda_qC_2; % lambda_dC_2
462 % y(12) = vqC_1 - (rs+r1)*iqs_2 - wa_2 * lambda_dC_2; % lambda_qC_2
463 % y(13) = vdr_2 - rr*idr_2 + (wa_2-wr_2)*lambda_qr_2; % lambda_dr_2
464 % y(14) = vqr_2 - rr*iqr_2 - (wa_2-wr_2)*lambda_dr_2; % lambda_qr_2
465 % y(15) = vd2_2 - r2*id2_2 + wa_2 * lambda_q2_2; % lambda_d2_2
466 % y(16) = vq2_2 - r2*iq2_2 - wa_2 * lambda_d2_2; % lambda_q2_2
467 y(11) = vdC_1 - (rs+r1)*ids_2; % lambda_dC_2
468 y(12) = vqC_1 - (rs+r1)*iqs_2; % lambda_qC_2
469 y(13) = vdr_2 - rr*idr_2 - wr_2*lambda_qr_2; % lambda_dr_2
470 y(14) = vqr_2 - rr*iqr_2 + wr_2*lambda_dr_2; % lambda_qr_2
471 y(15) = vd2_2 - r2*id2_2; % lambda_d2_2
472 y(16) = vq2_2 - r2*iq2_2; % lambda_q2_2
473 y(17) = (1/J)*((Te_2-TL_2)*P/2 - b*wr_2); % wr_2
474 y(18) = wr_2; % theta_r_2
475 %-----%
476 % 3RD MOTOR
477 %-----%
478 y(19) = vdC_1 - (rs+r1)*ids_3; % lambda_dC_3
479 y(20) = vqC_1 - (rs+r1)*iqs_3; % lambda_qC_3
480 y(21) = vdr_3 - rr*idr_3 - wr_3*lambda_qr_3; % lambda_dr_3
481 y(22) = vqr_3 - rr*iqr_3 + wr_3*lambda_dr_3; % lambda_qr_3
482 y(23) = vd2_3 - r2*id2_3; % lambda_d2_3
483 y(24) = vq2_3 - r2*iq2_3; % lambda_q2_3
484 y(25) = (1/J)*((Te_3-TL_3)*P/2 - b*wr_3); % wr_3
485 y(26) = wr_3; % theta_r_3
486 %-----%
487 % CONTROLLER
488 %-----%
489 % Automatic state-building diff. equations of the controller, based on its order
490 y(27:27+n-1) = xk2_dot;
491 y(27+n:27+2*n-1) = xk3_dot;
492 %-----%
493
494 y = y';
495 end

```

B.3 IDFOC Control, External Resistor Method

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Multirun_3IM_IFOC_res.m
3 % Runs 3-motor simulation of resistor sync method using IDFOC control
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 clear
6 close all
7 clc
8
9 % load k_hinf
10 % [Ak,Bk,Ck,Dk] = ssdata(k_hinf);
11
12 Kp = -30; % 1/rad
13 Ki = -60; % 1/(rad*s)
14 [Ak,Bk,Ck,Dk] = ssdata(tf([Kp Ki],[1 0]));
15
16 % Code currently only supports 3 identical motors
17 ParametersIM = ReturnParametersIM_15hp;
18
19 rs = ParametersIM.rs;
20 rr = ParametersIM.rr;
21 Lls = ParametersIM.Lls;
22 LLr = ParametersIM.LLr;
23 Lm = ParametersIM.Lm;
24 Ls = ParametersIM.Ls;
25 Lr = ParametersIM.Lr;
26 % sigma = 1 - Lm^2/(Ls*Lr);
27 % tau_s = Ls/rs;
28 % tau_r = Lr/rr;
29 % sigma_s = tau_s*sigma;
30 % sigma_r = tau_r*sigma;
31 % mu_s = 1/sigma_s;
32 % mu_r = 1/sigma_r;
33
34 % Incorporating mu_s inside the param structure
35 % ParametersIM.mu_s = mu_s;
36
37 ParamController.h = 0.1; % Hysteresis band
38
39 ParamController.Ak = Ak;
40 ParamController.Bk = Bk;
41 ParamController.Ck = Ck;
42 ParamController.Dk = Dk;
43 ParamController.order = length(Ak);
44
45 % ParamController.rext_max = ParametersIM.2.rs*100;
46 ParamController.rext_max = 1.5;
47
48 % Parameters for switching losses
49 ParamSys.vswM = 0; % Main converter switch voltage
50 ParamSys.vdM = 0; % Main converter diode voltage
51 ParamSys.vswR = 0; % Resistor board switch voltage
52 ParamSys.vdR = 0; % Resistor board diode voltage
53
54 TL1_variation = 0; % Variation in load torque in pu
55 TL2_variation = -0.2;
56 TL3_variation = -0.3;
57
58 time_variation_TL1 = 2;
59 time_variation_TL2 = 2;
60 time_variation_TL3 = 2;
61
62 ParamSys.TL1_variation = TL1_variation;
63 ParamSys.TL2_variation = TL2_variation;
64 ParamSys.TL3_variation = TL3_variation;
65 ParamSys.time_variation_TL1 = time_variation_TL1;
66 ParamSys.time_variation_TL2 = time_variation_TL2;
67 ParamSys.time_variation_TL3 = time_variation_TL3;
68
69 num_states = 20 + 2*ParamController.order;
70 x0 = zeros(1,num_states); % Initial conditions for state variables
71 % load x0_save
72 % x0 = x0_save;
73
74 t_ini = 0;
75 t_end = 6;
76
77 deltat = 10e-6;
78 tspan = t_ini:deltat:t_end;
79 options = odeset('MaxStep', deltat, 'RelTol', 1e-5, 'AbsTol', 1e-6);
80 % options = odeset();

```

```

81
82 tic
83 [t,y] = ode45(@(t,y) ThreeIM_1Converter_IFOC_res(t,y,ParametersIM, ...
84     ParamSys, ParamController), tspan, x0, options);
85 toc
86
87 % Save state
88 % x0_save = y(end,:);
89 % save x0_save x0_save
90
91 %-----%
92 %                POST-PROCESSING
93 %-----%
94
95 %-----%
96 %                PICKING UP THE STATES
97 %-----%
98 %                1ST MOTOR
99 %-----%
100 lambda_ds_1 = y(:,1); % Flux ds [Wb]
101 lambda_qs_1 = y(:,2); % Flux qs [Wb]
102 lambda_dr_1 = y(:,3); % Flux dr [Wb]
103 lambda_qr_1 = y(:,4); % Flux qr [Wb]
104 wr_1 = y(:,5); % Rotor speed (electric) [rad/s]
105 theta_r_1 = y(:,6); % Rotor position (electric) [rad]
106 %-----%
107 %                INDIRECT FIELD ORIENTED CONTROL (1ST MOTOR)
108 %-----%
109 R.theta_e = y(:,7); % Rotor flux angular position (electric) [rad]
110 % I_Action_CL_w_1 = y(:,8); % Integral action of the closed loop control of wm
111 %-----%
112 %-----%
113 %                2ND MOTOR
114 %-----%
115 lambda_ds_2 = y(:,9); % Flux ds [Wb]
116 lambda_qs_2 = y(:,10); % Flux qs [Wb]
117 lambda_dr_2 = y(:,11); % Flux dr [Wb]
118 lambda_qr_2 = y(:,12); % Flux qr [Wb]
119 wr_2 = y(:,13); % Rotor speed (electric) [rad/s]
120 theta_r_2 = y(:,14); % Rotor position (electric) [rad]
121 %-----%
122 %-----%
123 %                3RD MOTOR
124 %-----%
125 lambda_ds_3 = y(:,15); % Flux ds [Wb]
126 lambda_qs_3 = y(:,16); % Flux qs [Wb]
127 lambda_dr_3 = y(:,17); % Flux dr [Wb]
128 lambda_qr_3 = y(:,18); % Flux qr [Wb]
129 wr_3 = y(:,19); % Rotor speed (electric) [rad/s]
130 theta_r_3 = y(:,20); % Rotor position (electric) [rad]
131 %-----%
132 %-----%
133 %-----%
134 %                EXTRACTING OUTPUT STRUCTURES FROM DIFF EQ FUNCTION
135 %-----%
136 [~, IM1_out, IM2_out, IM3_out, Sw_out] = cellfun(@(t,y) ...
137     ThreeIM_1Converter_IFOC_res(t, y, ParametersIM, ParamSys, ...
138     ParamController), num2cell(t), num2cell(y,2), 'uni',0);
139
140 R_vqsC = zeros(numel(t),1);
141 R_vdsC = zeros(numel(t),1);
142
143 R_TL_1 = zeros(numel(t),1);
144 R_Te_1 = zeros(numel(t),1);
145 theta_a_1 = zeros(numel(t),1);
146 R_vqs_1 = zeros(numel(t),1);
147 R_vds_1 = zeros(numel(t),1);
148
149 R_TL_2 = zeros(numel(t),1);
150 R_Te_2 = zeros(numel(t),1);
151 theta_a_2 = zeros(numel(t),1);
152 R_vqs_2 = zeros(numel(t),1);
153 R_vds_2 = zeros(numel(t),1);
154 rext2_pwm = zeros(numel(t),1);
155 rext2_cont = zeros(numel(t),1);
156
157 R_TL_3 = zeros(numel(t),1);
158 R_Te_3 = zeros(numel(t),1);
159 theta_a_3 = zeros(numel(t),1);
160 R_vqs_3 = zeros(numel(t),1);
161 R_vds_3 = zeros(numel(t),1);
162 rext3_pwm = zeros(numel(t),1);

```

```

163 rext3_cont = zeros(numel(t),1);
164
165 R_S1 = zeros(numel(t),1);
166 R_S2 = zeros(numel(t),1);
167 R_S3 = zeros(numel(t),1);
168
169 for i=1:numel(t)
170     R_vqsC(i) = IM1_out{i}.vqsC;
171     R_vdsC(i) = IM1_out{i}.vdsC;
172
173     R_TL_1(i) = IM1_out{i}.TL;
174     R_Te_1(i) = IM1_out{i}.Te;
175     %   theta_a_1(i) = IM1_out{i}.theta_a;
176
177     R_vqs_1(i) = IM1_out{i}.vqs;
178     R_vds_1(i) = IM1_out{i}.vds;
179
180     R_TL_2(i) = IM2_out{i}.TL;
181     R_Te_2(i) = IM2_out{i}.Te;
182     %   theta_a_2(i) = IM2_out{i}.theta_a;
183
184     R_vqs_2(i) = IM2_out{i}.vqs;
185     R_vds_2(i) = IM2_out{i}.vds;
186
187     rext2_pwm(i) = IM2_out{i}.rext_pwm;
188     rext2_cont(i) = IM2_out{i}.rext_cont;
189
190     R_TL_3(i) = IM3_out{i}.TL;
191     R_Te_3(i) = IM3_out{i}.Te;
192     %   theta_a_3(i) = IM3_out{i}.theta_a;
193
194     R_vqs_3(i) = IM3_out{i}.vqs;
195     R_vds_3(i) = IM3_out{i}.vds;
196
197     rext3_pwm(i) = IM3_out{i}.rext_pwm;
198     rext3_cont(i) = IM3_out{i}.rext_cont;
199
200     R_S1(i) = Sw_out{i}.S(1);
201     R_S2(i) = Sw_out{i}.S(2);
202     R_S3(i) = Sw_out{i}.S(3);
203
204 end
205 %-----%
206 clear y IM1_out IM2_out IM3_out Sw_out
207 %-----%
208 %           Auxiliary Equations for the IMs
209 %-----%
210
211 % Transformation of electrical rotor speed and electrical angular position
212 % to mechanical rotor speed and mechanical angular position
213
214 % 1st IM
215 R_wm_1 = wr_1/(ParametersIM.P/2);
216 R_theta_m_1 = theta_r_1/(ParametersIM.P/2);
217
218 % 2nd IM
219 R_wm_2 = wr_2/(ParametersIM.P/2);
220 R_theta_m_2 = theta_r_2/(ParametersIM.P/2);
221
222 % 3rd IM
223 R_wm_3 = wr_3/(ParametersIM.P/2);
224 R_theta_m_3 = theta_r_3/(ParametersIM.P/2);
225
226 % Transformation of fluxes to currents — DQ components
227 % 1ST MOTOR
228 [R_ids_1, R_iqs_1, R_idr_1, R_iqr_1] = Fluxes2Currents(lambda_ds_1, ...
229     lambda_qs_1, lambda_dr_1, lambda_qr_1, ParametersIM);
230 % 2ND MOTOR
231 [R_ids_2, R_iqs_2, R_idr_2, R_iqr_2] = Fluxes2Currents(lambda_ds_2, ...
232     lambda_qs_2, lambda_dr_2, lambda_qr_2, ParametersIM);
233 % 3RD MOTOR
234 [R_ids_3, R_iqs_3, R_idr_3, R_iqr_3] = Fluxes2Currents(lambda_ds_3, ...
235     lambda_qs_3, lambda_dr_3, lambda_qr_3, ParametersIM);
236
237 % Calculating voltage actually reaching motor 2
238 R_vqs_2 = R_vqs_2 - (R_iqs_2 .* rext2_pwm);
239 R_vds_2 = R_vds_2 - (R_ids_2 .* rext2_pwm);
240
241 % Calculating voltage actually reaching motor 3
242 R_vqs_3 = R_vqs_3 - (R_iqs_3 .* rext3_pwm);
243 R_vds_3 = R_vds_3 - (R_ids_3 .* rext3_pwm);
244

```

```

245 % Clear variables not saved
246 clear lambda_ds_1 lambda_qs_1 lambda_dr_1 lambda_qr_1 theta_r_1 wr_1 ...
247     lambda_ds_2 lambda_qs_2 lambda_dr_2 lambda_qr_2 theta_r_2 wr_2 ...
248     lambda_ds_3 lambda_qs_3 lambda_dr_3 lambda_qr_3 theta_r_3 wr_3 ...
249     i j tspan t_ini t_end x0 num_states deltat
250 %-----%

1 function [y, IM1, IM2, IM3, Sw] = ThreeIM_1Converter_IFOC_res(t, u, ...
2     ParametersIM, ParamSys, ParamController)
3 %-----%
4 %                               STATES
5 %-----%
6 %                               1ST MOTOR
7 %-----%
8 lambda_ds_1 = u(1); % Flux ds [Wb]
9 lambda_qs_1 = u(2); % Flux qs [Wb]
10 lambda_dr_1 = u(3); % Flux dr [Wb]
11 lambda_qr_1 = u(4); % Flux qr [Wb]
12 wr_1 = u(5); % Rotor speed (electric) [rad/s]
13 theta_r_1 = u(6); % Rotor position (electric) [rad]
14 %-----%
15 %                               INDIRECT FIELD ORIENTED CONTROL (1ST MOTOR)
16 %-----%
17 theta_e = u(7); % Rotor flux angular position (electric) [rad]
18 I_Action_CL_w_1 = u(8); % Integral action of the closed loop control of wm
19 %-----%
20 %-----%
21 %                               2ND MOTOR
22 %-----%
23 lambda_ds_2 = u(9); % Flux ds [Wb]
24 lambda_qs_2 = u(10); % Flux qs [Wb]
25 lambda_dr_2 = u(11); % Flux dr [Wb]
26 lambda_qr_2 = u(12); % Flux qr [Wb]
27 wr_2 = u(13); % Rotor speed (electric) [rad/s]
28 theta_r_2 = u(14); % Rotor position (electric) [rad]
29 %-----%
30 %-----%
31 %                               3RD MOTOR
32 %-----%
33 lambda_ds_3 = u(15); % Flux ds [Wb]
34 lambda_qs_3 = u(16); % Flux qs [Wb]
35 lambda_dr_3 = u(17); % Flux dr [Wb]
36 lambda_qr_3 = u(18); % Flux qr [Wb]
37 wr_3 = u(19); % Rotor speed (electric) [rad/s]
38 theta_r_3 = u(20); % Rotor position (electric) [rad]
39 %-----%
40 %-----%
41 %                               RESISTANCE CONTROL OF MOTORS 2&3
42 %-----%
43 %-----%
44 %                               CONTROLLERS
45 %-----%
46 n = ParamController.order;
47 % Automatic state-building of the controllers, based on order
48 xk2 = u(21:21+n-1);
49 xk2 = reshape(xk2,n,1);
50 xk3 = u(21+n:21+2*n-1);
51 xk3 = reshape(xk3,n,1);
52 %-----%
53 %-----%
54 %                               GET SYSPARAMS
55 %-----%
56 %-----%
57 %-----%
58 %-----%
59 vswM = ParamSys.vswM;
60 vdM = ParamSys.vdM;
61 vswR = ParamSys.vswR;
62 vdR = ParamSys.vdR;
63
64 TL1_delta = ParamSys.TL1_variation;
65 time_delta_TL1 = ParamSys.time_variation_TL1;
66
67 TL2_delta = ParamSys.TL2_variation;
68 time_delta_TL2 = ParamSys.time_variation_TL2;
69
70 TL3_delta = ParamSys.TL3_variation;
71 time_delta_TL3 = ParamSys.time_variation_TL3;
72
73 if t > time_delta_TL1
74     TL_1 = (1+TL1_delta)*ParametersIM.Tnom;
75 else

```

```

76     % TL_1 = ParametersIM.Tnom;
77     TL_1 = 0;
78 end
79
80 if t > time_delta_TL2
81     TL_2 = (1+TL2_delta)*ParametersIM.Tnom;
82 else
83     % TL_2 = ParametersIM.Tnom;
84     TL_2 = 0;
85 end
86
87 if t > time_delta_TL3
88     TL_3 = (1+TL3_delta)*ParametersIM.Tnom;
89 else
90     % TL_3 = ParametersIM.Tnom;
91     TL_3 = 0;
92 end
93 %-----%
94 %-----%
95 % GET CONTROLLER PARAMS
96 %-----%
97 % Hysteresis control
98 h = ParamController.h; % Hysteresis band
99
100 % r_ext control
101 Ak = ParamController.Ak;
102 Bk = ParamController.Bk;
103 Ck = ParamController.Ck;
104 Dk = ParamController.Dk;
105 %-----%
106 % Arbitrary Reference Frame
107 %-----%
108 % Stator reference
109 %-----%
110 % % 1st IM
111 % wa_1 = 0;
112 % theta_a_1 = 0;
113 %
114 % % 2nd IM
115 % wa_2 = 0;
116 % theta_a_2 = 0;
117 %
118 % % 3rd IM
119 % wa_3 = 0;
120 % theta_a_3 = 0;
121 %-----%
122 %-----%
123 % Auxiliary Equations for the IMs
124 %-----%
125 %-----%
126 % Transformation of fluxes to currents — DQ components
127 % 1ST MOTOR
128 [ids_1, iqs_1, idr_1, iqr_1] = Fluxes2Currents(lambda_ds_1, lambda_qs_1, ...
129     lambda_dr_1, lambda_qr_1, ParametersIM);
130 % 2ND MOTOR
131 [ids_2, iqs_2, idr_2, iqr_2] = Fluxes2Currents(lambda_ds_2, lambda_qs_2, ...
132     lambda_dr_2, lambda_qr_2, ParametersIM);
133 % 3RD MOTOR
134 [ids_3, iqs_3, idr_3, iqr_3] = Fluxes2Currents(lambda_ds_3, lambda_qs_3, ...
135     lambda_dr_3, lambda_qr_3, ParametersIM);
136
137 % Electromagnetic and Load Torques
138 % 1ST MOTOR
139 Te_1 = 3/4*ParametersIM.P*(lambda_ds_1.*iqs_1 - lambda_qs_1.*ids_1);
140 % 2ND MOTOR
141 Te_2 = 3/4*ParametersIM.P*(lambda_ds_2.*iqs_2 - lambda_qs_2.*ids_2);
142 % 3RD MOTOR
143 Te_3 = 3/4*ParametersIM.P*(lambda_ds_3.*iqs_3 - lambda_qs_3.*ids_3);
144
145 % Transformation of electrical rotor speed and electrical angular position
146 % to mechanical rotor speed and mechanical angular position
147 % 1st IM
148 % wm_1 = wr_1/(ParametersIM.P/2);
149 % theta_m_1 = theta_r_1/(ParametersIM.P/2);
150 % 2nd IM
151 % wm_2 = wr_2/(ParametersIM.P/2);
152 % theta_m_2 = theta_r_2/(ParametersIM.P/2);
153 % 3rd IM
154 % wm_3 = wr_3/(ParametersIM.P/2);
155 % theta_m_3 = theta_r_3/(ParametersIM.P/2);
156
157 % Transforming stator and rotor currents from qd reference to abc variables

```

```

158 % % 1st IM
159 % [ias_1, ibs_1, ics_1] = qd2abcs(iqs_1, ids_1, theta_a_1);
160 % % 2nd IM
161 % [ias_2, ibs_2, ics_2] = qd2abcs(iqs_2, ids_2, theta_a_2);
162 % % 3rd IM
163 % [ias_3, ibs_3, ics_3] = qd2abcs(iqs_3, ids_3, theta_a_3);
164 % 1st IM
165 ias_1 = iqs_1;
166 [ibs_1, ics_1] = statbc(iqs_1, ids_1);
167 % 2nd IM
168 ias_2 = iqs_2;
169 [ibs_2, ics_2] = statbc(iqs_2, ids_2);
170 % 3rd IM
171 ias_3 = iqs_3;
172 [ibs_3, ics_3] = statbc(iqs_3, ids_3);
173
174 % Rotor Voltages (squirrel cage) — DQ components
175 % 1st IM
176 vdr_1 = 0;
177 vqr_1 = 0;
178 % 2nd IM
179 vdr_2 = 0;
180 vqr_2 = 0;
181 % 3rd IM
182 vdr_3 = 0;
183 vqr_3 = 0;
184 %-----%
185
186 %-----%
187 %
188 %                INDIRECT (ROTOR) FIELD ORIENTED CONTROL
189 %                INDUCTION MOTOR 1
190 %-----%
191 [lambda_dr_star, wm_star] = RefLambdaAndwm(t, ParametersIM);
192
193 % Indirect field oriented control drive
194 [iqs_star, ids_star, di_wm_dt, wrf_est] = IFOCDrive(ParametersIM, ...
195     lambda_dr_star, wm_star, wr_1, I_Action_CL_w_1);
196
197 % Converting currents to abc variables for Hysteresis current control
198 % Commanded currents
199 [ias_star, ibs_star, ics_star] = qd2abcs(iqs_star, ids_star, theta_e);
200 i_abcs_star = [ias_star, ibs_star, ics_star];
201
202 % Measured/real currents
203 i_abcs = [iqs_1, ibs_1, ics_1];
204
205 % Hysteresis current control
206 [S] = myHysteresisCurrentControl(i_abcs', i_abcs_star', h);
207
208 % 3phase bridge converter voltages
209 vdc = ParametersIM.Vs*sqrt(2); % peak of line-to-line voltage
210 IM = [(ias_1+ias_2+ias_3) (ibs_1+ibs_2+ibs_3) (ics_1+ics_2+ics_3)];
211 [vas, vbs, vcs] = my3PhaseConverterVoltLoss(S', vdc, vswM, vdM, IM);
212
213 % Converting vabcs voltages to vqds
214 [vqsC, vdsC] = statqd(vas, vbs, vcs);
215 %-----%
216
217 %-----%
218 %
219 %                STATOR RESISTANCE CONTROL
220 %                INDUCTION MOTOR 2
221 %-----%
222
223 %-----%
224 %                CONTROLLER EQUATIONS
225 %-----%
226 % Controller input vector uk
227 % For now, only the position error
228 uk2(1) = theta_m_1 - theta_m_2;
229 xk2_dot = Ak*xk2 + Bk*uk2;
230 yk2 = Ck*xk2 + Dk*uk2;
231
232 uk3(1) = theta_m_1 - theta_m_3;
233 xk3_dot = Ak*xk3 + Bk*uk3;
234 yk3 = Ck*xk3 + Dk*uk3;
235
236 % Controller output vector yk
237 % For PI controller, yk is in units of 1/rad
238 rext2_cont = yk2(1)*ParametersIM.rs;
239 rext2_cont = max(rext2_cont, 0);

```

```

240 rext2_cont = min(rext2_cont,ParamController.rext_max);
241
242 rext3_cont = yk3(1)*ParametersIM.rs;
243 rext3_cont = max(rext3_cont,0);
244 rext3_cont = min(rext3_cont,ParamController.rext_max);
245 %-----%
246
247 % Controlable external resistance via PWM
248 f_sw = 4.987654321e3;
249 r_full = ParamController.rext_max;
250 d2 = rext2_cont/r_full;
251 S_rext2 = myPWM(t, d2, f_sw);
252 rext2_pwm = S_rext2 * r_full;
253
254 d3 = rext3_cont/r_full;
255 S_rext3 = myPWM(t, d3, f_sw);
256 rext3_pwm = S_rext3 * r_full;
257
258 % PWM external resistance
259 rext2 = rext2_pwm;
260 rext3 = rext3_pwm;
261
262 % Ideal external resistance
263 % rext2 = rext2_cont;
264 % rext3 = rext3_cont;
265
266 % vqs_2 = vqs_1;
267 % vds_2 = vds_1;
268 % If external resistance is off, then current flows through switches. If
269 % these are non-ideal, then voltage drop is always 1 switch + 1 diode
270 vas_2 = vas - ~S_rext2*sign(ias_2)*(vswR+vdR);
271 vbs_2 = vbs - ~S_rext2*sign(ibs_2)*(vswR+vdR);
272 vcs_2 = vcs - ~S_rext2*sign(ics_2)*(vswR+vdR);
273 % Transforming vabcs to vds and vqs
274 [vqs_2, vds_2] = statqd(vas_2, vbs_2, vcs_2);
275
276 vas_3 = vas - ~S_rext3*sign(ias_3)*(vswR+vdR);
277 vbs_3 = vbs - ~S_rext3*sign(ibs_3)*(vswR+vdR);
278 vcs_3 = vcs - ~S_rext3*sign(ics_3)*(vswR+vdR);
279 % Transforming vabcs to vds and vqs
280 [vqs_3, vds_3] = statqd(vas_3, vbs_3, vcs_3);
281
282 % Apply this loss to motor line 1 too for parity. In a real system, both
283 % would be connected to a board in case torque difference reverses
284 vas_1 = vas - sign(ias_1)*(vswR+vdR);
285 vbs_1 = vbs - sign(ibs_1)*(vswR+vdR);
286 vcs_1 = vcs - sign(ics_1)*(vswR+vdR);
287 % Transforming vabcs to vds and vqs
288 [vqs_1, vds_1] = statqd(vas_1, vbs_1, vcs_1);
289 %-----%
290
291 %-----%
292 %
293 %-----%
294 % Switching states
295 Sw.S = S;
296 Sw.S_rext2 = S_rext2;
297 Sw.S_rext3 = S_rext3;
298
299 % 1st motor
300 IM1.vqsC = vqsC;
301 IM1.vdsC = vdsC;
302 IM1.TL = TL_1;
303 IM1.Te = Te_1;
304 IM1.vqs = vqs_1;
305 IM1.vds = vds_1;
306 IM1.theta_e = theta_e;
307 % IM1.we = we;
308 % IM1.theta_e = theta_e;
309 % IM1.theta_a = theta_a_1;
310 IM1.wm_star = wm_star;
311
312 % 2nd motor
313 IM2.TL = TL_2;
314 IM2.Te = Te_2;
315 IM2.vqs = vqs_2;
316 IM2.vds = vds_2;
317 % IM2.theta_a = theta_a_2;
318 IM2.rext_pwm = rext2_pwm;
319 IM2.rext_cont = rext2_cont;
320
321 % 3rd motor

```

```

322 IM3.TL = TL_3;
323 IM3.Te = Te_3;
324 IM3.vqs = vqs_3;
325 IM3.vds = vds_3;
326 % IM3.theta_a = theta_a_3;
327 IM3.rext_pwm = rext3_pwm;
328 IM3.rext_cont = rext3_cont;
329
330 %-----%
331
332 %-----%
333 %
334 %          DIFF. STATES EQUATIONS
335 %-----%
336 %
337 %          1ST MOTOR
338 %-----%
338 % y(1) = vds_1 - ParametersIM.rs*ids_1 + wa_1 * lambda_qs_1;          % lambda_ds_1
339 % y(2) = vqs_1 - ParametersIM.rs*iqs_1 - wa_1 * lambda_ds_1;          % lambda_qs_1
340 % y(3) = vdr_1 - ParametersIM.rr*idr_1 + (wa_1-wr_1)*lambda_qr_1; % lambda_dr_1
341 % y(4) = vqr_1 - ParametersIM.rr*iqr_1 - (wa_1-wr_1)*lambda_dr_1; % lambda_qr_1
342 y(1) = vds_1 - ParametersIM.rs*ids_1;          % lambda_ds_1
343 y(2) = vqs_1 - ParametersIM.rs*iqs_1;          % lambda_qs_1
344 y(3) = vdr_1 - ParametersIM.rr*idr_1 - wr_1*lambda_qr_1; % lambda_dr_1
345 y(4) = vqr_1 - ParametersIM.rr*iqr_1 + wr_1*lambda_dr_1; % lambda_qr_1
346 y(5) = (1/ParametersIM.J)*((Te_1-TL_1)*ParametersIM.P/2 + ...
347          -ParametersIM.b*wr_1);          % wr_1
348 y(6) = wr_1;          % theta_r_1
349 %-----%
350 %          INDIRECT FIELD ORIENTED CONTROL (1ST MOTOR)
351 %-----%
352 y(7) = wrf_est; % Rotor flux angular position (electric)
353 y(8) = di_wm_dt; % Integral action of the closed loop control of wm
354 %-----%
355 %
356 %          2ND MOTOR
357 %-----%
358 % y(9) = vds_2 - (ParametersIM.rs + rext2)*ids_2 + wa_2 * lambda_qs_2; % lambda_ds_2
359 % y(10) = vqs_2 - (ParametersIM.rs + rext2)*iqs_2 - wa_2 * lambda_ds_2; % lambda_qs_2
360 % y(11) = vdr_2 - ParametersIM.rr*idr_2 + (wa_2-wr_2)*lambda_qr_2; % lambda_dr_2
361 % y(12) = vqr_2 - ParametersIM.rr*iqr_2 - (wa_2-wr_2)*lambda_dr_2; % lambda_qr_2
362 y(9) = vds_2 - (ParametersIM.rs + rext2)*ids_2; % lambda_ds_2
363 y(10) = vqs_2 - (ParametersIM.rs + rext2)*iqs_2; % lambda_qs_2
364 y(11) = vdr_2 - ParametersIM.rr*idr_2 - wr_2*lambda_qr_2; % lambda_dr_2
365 y(12) = vqr_2 - ParametersIM.rr*iqr_2 + wr_2*lambda_dr_2; % lambda_qr_2
366 y(13) = (1/ParametersIM.J)*((Te_2-TL_2)*ParametersIM.P/2 + ...
367          -ParametersIM.b*wr_2);          % wr_2
368 y(14) = wr_2;          % theta_r_2
369 %-----%
370 %          3RD MOTOR
371 %-----%
372 % y(15) = vds_3 - (ParametersIM.rs + rext3)*ids_3 + wa_3 * lambda_qs_3; % lambda_ds_3
373 % y(16) = vqs_3 - (ParametersIM.rs + rext3)*iqs_3 - wa_3 * lambda_ds_3; % lambda_qs_3
374 % y(17) = vdr_3 - ParametersIM.rr*idr_3 + (wa_3-wr_3)*lambda_qr_3; % lambda_dr_3
375 % y(18) = vqr_3 - ParametersIM.rr*iqr_3 - (wa_3-wr_3)*lambda_dr_3; % lambda_qr_3
376 y(15) = vds_3 - (ParametersIM.rs + rext3)*ids_3; % lambda_ds_3
377 y(16) = vqs_3 - (ParametersIM.rs + rext3)*iqs_3; % lambda_qs_3
378 y(17) = vdr_3 - ParametersIM.rr*idr_3 - wr_3*lambda_qr_3; % lambda_dr_3
379 y(18) = vqr_3 - ParametersIM.rr*iqr_3 + wr_3*lambda_dr_3; % lambda_qr_3
380 y(19) = (1/ParametersIM.J)*((Te_3-TL_3)*ParametersIM.P/2 + ...
381          -ParametersIM.b*wr_3);          % wr_3
382 y(20) = wr_3;          % theta_r_3
383 %-----%
384 %          RESISTANCE CONTROL OF 2ND MOTOR
385 %-----%
386 %
387 %          CONTROLLER
388 %-----%
389 % Automatic state-building diff. equations of the controller, based on its order
390 y(21:21+n-1) = xk2_dot;
391 y(21+n:21+2*n-1) = xk3_dot;
392 %-----%
393
394 y = y';
395 end

```

B.4 IDFOC Control, Auxiliary Converter Method

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Multirun_3IM_IFOC_volt_xf.m

```

```

3 % Runs 3-motor simulation of auxiliary converter method using CVHz control
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 clear
6 close all
7 clc
8
9 % load k_hinf
10 % [Ak,Bk,Ck,Dk] = ssdata(k_hinf);
11
12 Kp = -80; % V/rad
13 Ki = -120; % V/(rad*s)
14 [Ak,Bk,Ck,Dk] = ssdata(tf([Kp Ki],[1 0]));
15
16 % Code currently only supports 3 identical motors
17 ParametersIM = ReturnParametersIM_15hp;
18
19 rs = ParametersIM.rs;
20 rr = ParametersIM.rr;
21 Lls = ParametersIM.Lls;
22 LLr = ParametersIM.LLr;
23 Lm = ParametersIM.Lm;
24 Ls = ParametersIM.Ls;
25 Lr = ParametersIM.Lr;
26 % sigma = 1 - Lm^2/(Ls*Lr);
27 % tau_s = Ls/rs;
28 % tau_r = Lr/rr;
29 % sigma_s = tau_s*sigma;
30 % sigma_r = tau_r*sigma;
31 % mu_s = 1/sigma_s;
32 % mu_r = 1/sigma_r;
33
34 % Incorporating mu_s inside the param structure
35 % ParametersIM.mu_s = mu_s;
36
37 ParamController.h = 0.1; % Hysteresis band
38
39 ParamController.Ak = Ak;
40 ParamController.Bk = Bk;
41 ParamController.Ck = Ck;
42 ParamController.Dk = Dk;
43 ParamController.order = length(Ak);
44
45 ParamController.delv_max = 50;
46 ParamController.tstart = 2; % Time to turn on voltage drop converter
47
48 % Parameters for switching losses
49 ParamSys.vswM = 0; % Main converter switch voltage
50 ParamSys.vdM = 0; % Main converter diode voltage
51 ParamSys.vswD = 0; % Drop converter switch voltage
52 ParamSys.vdD = 0; % Drop converter diode voltage
53
54 time_variation_TL1 = 2;
55 time_variation_TL2 = 2;
56 time_variation_TL3 = 2;
57
58 ParamSys.time_variation_TL1 = time_variation_TL1;
59 ParamSys.time_variation_TL2 = time_variation_TL2;
60 ParamSys.time_variation_TL3 = time_variation_TL3;
61
62 TL1_variation = 0; % Variation in load torque in pu
63 ParamSys.TL1_variation = TL1_variation;
64
65 TL2_variation = -0.2;
66 ParamSys.TL2_variation = TL2_variation;
67
68 TL3_variation = -0.3;
69 ParamSys.TL3_variation = TL3_variation;
70
71 num_states = 26 + 2*ParamController.order;
72 x0 = zeros(1,num_states); % Initial conditions for state variables
73 % load x0_save
74 % x0 = x0_save;
75
76 t_ini = 0;
77 t_end = 6;
78
79 deltat = 10e-6;
80 tspan = t_ini:deltat:t_end;
81 options = odeset('MaxStep', deltat, 'RelTol', 1e-5, 'AbsTol', 1e-6);
82 % options = odeset();
83
84 % Set up iterations and allocate space for saved variables

```

```

85 Nlist = [1/5 1/3 1/2 1/2];
86 vdcRlist = [1 1 1 0.5];
87
88 TL_2_out = zeros(numel(tspan),numel(Nlist));
89 Te_2_out = TL_2_out;
90 delv2_vd_out = TL_2_out;
91 delv2_vq_out = TL_2_out;
92 delv2_cont_out = TL_2_out;
93 wm_2_out = TL_2_out;
94 theta_m_2_out = TL_2_out;
95 ids_2_out = TL_2_out;
96 iqs_2_out = TL_2_out;
97 idr_2_out = TL_2_out;
98 iqr_2_out = TL_2_out;
99 id2_2_out = TL_2_out;
100 iq2_2_out = TL_2_out;
101 vq1_2_out = TL_2_out;
102 vd1_2_out = TL_2_out;
103 vqs_2_out = TL_2_out;
104 vds_2_out = TL_2_out;
105 Sd1_2_out = TL_2_out;
106 Sd2_2_out = TL_2_out;
107 Sd3_2_out = TL_2_out;
108
109 TL_3_out = TL_2_out;
110 Te_3_out = TL_2_out;
111 delv3_vd_out = TL_2_out;
112 delv3_vq_out = TL_2_out;
113 delv3_cont_out = TL_2_out;
114 wm_3_out = TL_2_out;
115 theta_m_3_out = TL_2_out;
116 ids_3_out = TL_2_out;
117 iqs_3_out = TL_2_out;
118 idr_3_out = TL_2_out;
119 iqr_3_out = TL_2_out;
120 id2_3_out = TL_2_out;
121 iq2_3_out = TL_2_out;
122 vq1_3_out = TL_2_out;
123 vd1_3_out = TL_2_out;
124 vqs_3_out = TL_2_out;
125 vds_3_out = TL_2_out;
126 Sd1_3_out = TL_2_out;
127 Sd2_3_out = TL_2_out;
128 Sd3_3_out = TL_2_out;
129
130 TL_1_out = TL_2_out;
131 Te_1_out = TL_1_out;
132 wm_1_out = TL_1_out;
133 theta_m_1_out = TL_1_out;
134 ids_1_out = TL_1_out;
135 iqs_1_out = TL_1_out;
136 idr_1_out = TL_1_out;
137 iqr_1_out = TL_1_out;
138 id2_1_out = TL_1_out;
139 iq2_1_out = TL_1_out;
140 vq1_1_out = TL_1_out;
141 vd1_1_out = TL_1_out;
142 vqs_1_out = TL_1_out;
143 vds_1_out = TL_1_out;
144 S1_out = TL_1_out;
145 S2_out = TL_1_out;
146 S3_out = TL_1_out;
147
148 theta_e_out = TL_1_out;
149
150 % Iterate over transformer turns ratios & auxiliary converter bus voltages
151 for j=1:numel(Nlist)
152
153 % Transformer params: line side = primary (1), aux converter = secondary (2)
154 N = Nlist(j); % N1/N2
155 ParametersXF.N = N; % Z' = Z*N^2, but less windings = less impedance
156 ParametersXF.r1 = 0.001; % So actual secondary impedance scales with N2,
157 ParametersXF.r2 = 0.01*N; % canceling a power of N for this scaling
158 ParametersXF.Ll1 = 0.1e-3;
159 ParametersXF.Ll2 = 0.5e-3*N;
160 ParametersXF.LmT = 10e-3;
161 ParametersXF.vdcR = vdcRlist(j); % Ratio between aux converter bus and main bus
162
163 % Calculate inductance matrix inverses in setup
164 % The factor of 3/2 for magnetizing inductance (Lm -> M) is assumed included
165 Ll1 = ParametersXF.Ll1;
166 Ll2 = ParametersXF.Ll2;

```

```

167 LmT = ParametersXF.LmT;
168 L = [[Lls+Ll1+Lm+LmT, Lm, LmT];...
169 [Lm, Llr+Lm, 0];...
170 [LmT, 0, Ll2+LmT]];
171 ParametersXF.Linv = inv(L);
172
173 tic
174 [t,y] = ode45(@(t,y) ThreeIM_1Converter_IFOC_volt_xf(t,y,ParametersIM, ...
175 ParametersXF, ParamSys, ParamController), tspan, x0, options);
176 toc
177
178 % Save state
179 % x0_save = y(end,:);
180 % save x0_save x0_save
181
182 %-----%
183 % POST-PROCESSING
184 %-----%
185
186 %-----%
187 % PICKING UP THE STATES
188 %-----%
189 % 1ST MOTOR
190 %-----%
191 lambda_dC_1 = y(:,1); % Flux dC (main converter path) [Wb]
192 lambda_qC_1 = y(:,2); % Flux qC [Wb]
193 lambda_dr_1 = y(:,3); % Flux dr (rotor path) [Wb]
194 lambda_qr_1 = y(:,4); % Flux qr [Wb]
195 lambda_d2_1 = y(:,5); % Flux d2 (side converter path) [Wb]
196 lambda_q2_1 = y(:,6); % Flux q2 [Wb]
197 wr_1 = y(:,7); % Rotor speed (electric) [rad/s]
198 theta_r_1 = y(:,8); % Rotor position (electric) [rad]
199 %-----%
200 % INDIRECT FIELD ORIENTED CONTROL (1ST MOTOR)
201 %-----%
202 theta_e = y(:,9); % Rotor flux angular position (electric) [rad]
203 I_Action_CL_w_1 = y(:,10); % Integral action of the closed loop control of wm
204 %-----%
205 %-----%
206 % 2ND MOTOR
207 %-----%
208 lambda_dC_2 = y(:,11); % Flux dC (main converter path) [Wb]
209 lambda_qC_2 = y(:,12); % Flux qC [Wb]
210 lambda_dr_2 = y(:,13); % Flux dr (rotor path) [Wb]
211 lambda_qr_2 = y(:,14); % Flux qr [Wb]
212 lambda_d2_2 = y(:,15); % Flux d2 (side converter path) [Wb]
213 lambda_q2_2 = y(:,16); % Flux q2 [Wb]
214 wr_2 = y(:,17); % Rotor speed (electric) [rad/s]
215 theta_r_2 = y(:,18); % Rotor position (electric) [rad]
216 %-----%
217 %-----%
218 % 3RD MOTOR
219 %-----%
220 lambda_dC_3 = y(:,19); % Flux dC (main converter path) [Wb]
221 lambda_qC_3 = y(:,20); % Flux qC [Wb]
222 lambda_dr_3 = y(:,21); % Flux dr (rotor path) [Wb]
223 lambda_qr_3 = y(:,22); % Flux qr [Wb]
224 lambda_d2_3 = y(:,23); % Flux d2 (side converter path) [Wb]
225 lambda_q2_3 = y(:,24); % Flux q2 [Wb]
226 wr_3 = y(:,25); % Rotor speed (electric) [rad/s]
227 theta_r_3 = y(:,26); % Rotor position (electric) [rad]
228 %-----%
229 %-----%
230 %-----%
231 % EXTRACTING OUTPUT STRUCTURES FROM DIFF EQ FUNCTION
232 %-----%
233 [-, IM1_out, IM2_out, IM3_out, Sw_out] = cellfun(@(t,y) ...
234 ThreeIM_1Converter_IFOC_volt_xf(t, y, ParametersIM, ParametersXF, ...
235 ParamSys, ParamController), num2cell(t), num2cell(y,2), 'uni',0);
236
237 TL_1 = zeros(numel(t),1);
238 Te_1 = zeros(numel(t),1);
239 % wa_1 = zeros(numel(t),1);
240 % theta_a_1 = zeros(numel(t),1);
241 vqC = zeros(numel(t),1);
242 vdC = zeros(numel(t),1);
243
244 TL_2 = zeros(numel(t),1);
245 Te_2 = zeros(numel(t),1);
246 % wa_2 = zeros(numel(t),1);
247 % theta_a_2 = zeros(numel(t),1);
248

```

```

249 delv2_vq = zeros(numel(t),1);
250 delv2_vd = zeros(numel(t),1);
251 delv2_cont = zeros(numel(t),1);
252
253 TL_3 = zeros(numel(t),1);
254 Te_3 = zeros(numel(t),1);
255 % wa_3 = zeros(numel(t),1);
256 % theta_a_3 = zeros(numel(t),1);
257
258 delv3_vq = zeros(numel(t),1);
259 delv3_vd = zeros(numel(t),1);
260 delv3_cont = zeros(numel(t),1);
261
262 S1 = zeros(numel(t),1);
263 S2 = zeros(numel(t),1);
264 S3 = zeros(numel(t),1);
265
266 Sd1_2 = zeros(numel(t),1);
267 Sd2_2 = zeros(numel(t),1);
268 Sd3_2 = zeros(numel(t),1);
269
270 Sd1_3 = zeros(numel(t),1);
271 Sd2_3 = zeros(numel(t),1);
272 Sd3_3 = zeros(numel(t),1);
273
274 for i=1:numel(t)
275     TL_1(i) = IM1_out{i}.TL;
276     Te_1(i) = IM1_out{i}.Te;
277     % wa_1(i) = IM1_out{i}.wa;
278     % theta_a_1(i) = IM1_out{i}.theta_a;
279
280     vqC(i) = IM1_out{i}.vqC;
281     vdC(i) = IM1_out{i}.vdC;
282
283     TL_2(i) = IM2_out{i}.TL;
284     Te_2(i) = IM2_out{i}.Te;
285     % wa_2(i) = IM2_out{i}.wa;
286     % theta_a_2(i) = IM2_out{i}.theta_a;
287
288     delv2_vq(i) = IM2_out{i}.delv2_vq;
289     delv2_vd(i) = IM2_out{i}.delv2_vd;
290     delv2_cont(i) = IM2_out{i}.delv2_cont;
291
292     TL_3(i) = IM3_out{i}.TL;
293     Te_3(i) = IM3_out{i}.Te;
294     % wa_3(i) = IM2_out{i}.wa;
295     % theta_a_3(i) = IM2_out{i}.theta_a;
296
297     delv3_vq(i) = IM3_out{i}.delv3_vq;
298     delv3_vd(i) = IM3_out{i}.delv3_vd;
299     delv3_cont(i) = IM3_out{i}.delv3_cont;
300
301     S1(i) = Sw_out{i}.S(1);
302     S2(i) = Sw_out{i}.S(2);
303     S3(i) = Sw_out{i}.S(3);
304
305     Sd1_2(i) = Sw_out{i}.S_d2(1);
306     Sd2_2(i) = Sw_out{i}.S_d2(2);
307     Sd3_2(i) = Sw_out{i}.S_d2(3);
308
309     Sd1_3(i) = Sw_out{i}.S_d3(1);
310     Sd2_3(i) = Sw_out{i}.S_d3(2);
311     Sd3_3(i) = Sw_out{i}.S_d3(3);
312
313 end
314
315 %-----%
316 clear y IM1_out IM2_out IM3_out Sw_out
317 %-----%
318 % Auxiliary Equations for the IMs
319 %-----%
320
321 % Transformation of electrical rotor speed and electrical angular position
322 % to mechanical rotor speed and mechanical angular position
323
324 % 1st IM
325 wm_1 = wr_1/(ParametersIM.P/2);
326 theta_m_1 = theta_r_1/(ParametersIM.P/2);
327 % 2nd IM
328 wm_2 = wr_2/(ParametersIM.P/2);
329 theta_m_2 = theta_r_2/(ParametersIM.P/2);
330 % 3rd IM

```

```

331 wm_3 = wr_3/(ParametersIM.P/2);
332 theta_m_3 = theta_r_3/(ParametersIM.P/2);
333
334 % Transformation of fluxes to currents — DQ components
335 Linv = ParametersXF.Linv;
336 % 1ST MOTOR
337 [ids_1, iqs_1, idr_1, iqr_1, id2_1, iq2_1] = Fluxes2Currents_xf(lambda_dC_1, ...
338     lambda_qC_1, lambda_dr_1, lambda_qr_1, lambda_d2_1, lambda_q2_1, Linv);
339 % 2ND MOTOR
340 [ids_2, iqs_2, idr_2, iqr_2, id2_2, iq2_2] = Fluxes2Currents_xf(lambda_dC_2, ...
341     lambda_qC_2, lambda_dr_2, lambda_qr_2, lambda_d2_2, lambda_q2_2, Linv);
342 % 3RD MOTOR
343 [ids_3, iqs_3, idr_3, iqr_3, id2_3, iq2_3] = Fluxes2Currents_xf(lambda_dC_3, ...
344     lambda_qC_3, lambda_dr_3, lambda_qr_3, lambda_d2_3, lambda_q2_3, Linv);
345
346 % Transforming stator and rotor currents from qd reference to abc variables
347 % Disabled to reduce size of saved data — can convert later
348 % % 1st IM
349 % % [ias_1, ibs_1, ics_1] = qd2abcs(iqs_1, ids_1, theta_a_1);
350 % ias_1 = iqs_1;
351 % [ibs_1, ics_1] = statbc(iqs_1, ids_1);
352 % % [iar_1, ~, ~] = qd2abcs(iqr_1, idr_1, theta_a_1-theta_r_1);
353 % % [ia2_1, ib2_1, ic2_1] = qd2abcs(iq2_1, id2_1, theta_a_1);
354 % ia2_1 = iq2_1;
355 % [ib2_1, ic2_1] = statbc(iq2_1, id2_1);
356 % % 2nd IM
357 % % [ias_2, ibs_2, ics_2] = qd2abcs(iqs_2, ids_2, theta_a_2);
358 % ias_2 = iqs_2;
359 % [ibs_2, ics_2] = statbc(iqs_2, ids_2);
360 % % [iar_2, ~, ~] = qd2abcs(iqr_2, idr_2, theta_a_2-theta_r_2);
361 % % [ia2_2, ib2_2, ic2_2] = qd2abcs(iq2_2, id2_2, theta_a_2);
362 % ia2_2 = iq2_2;
363 % [ib2_2, ic2_2] = statbc(iq2_2, id2_2);
364 %
365 % % Transforming stator voltages from qd reference to abc variables
366 % % Main converter
367 % % [vaC, vbC, vcC] = qd2abcs(vqC, vdC, theta_a_1);
368 % vaC = vqC;
369 % [vbC, vcC] = statbc(vqC, vdC);
370 % % Voltage drop converter
371 % % [delv2_va, delv2_vb, delv2_vc] = qd2abcs(delv2_vq, delv2_vd, theta_a_2);
372 % delv2_va = delv2_vq;
373 % [delv2_vb, delv2_vc] = statbc(delv2_vq, delv2_vd);
374 %
375 % Voltage reaching motors: Need to calculate based on currents. v=Ldi/dt
376 % 1st IM
377 Dids_1 = [0; diff(ids_1)/deltat];
378 Diqs_1 = [0; diff(iqs_1)/deltat];
379 Did2_1 = [0; diff(id2_1)/deltat];
380 Diq2_1 = [0; diff(iq2_1)/deltat];
381
382 % lambda_d1_1 = (Ll1 + LmT)*ids_1 + LmT*id2_1;
383 % lambda_q1_1 = (Ll1 + LmT)*iqs_1 + LmT*iq2_1;
384
385 % vd1_1 = ParametersXF.r1*ids_1 + Ll1*Dids_1 + LmT*(Dids_1+Did2_1) - wa_1.*lambda_q1_1;
386 vd1_1 = ParametersXF.r1*ids_1 + Ll1*Dids_1 + LmT*(Dids_1+Did2_1);
387 vds_1 = vdC - vd1_1;
388 % vq1_1 = ParametersXF.r1*iqs_1 + Ll1*Diqs_1 + LmT*(Diqs_1+Diq2_1) + wa_1.*lambda_d1_1;
389 vq1_1 = ParametersXF.r1*iqs_1 + Ll1*Diqs_1 + LmT*(Diqs_1+Diq2_1);
390 vqs_1 = vqC - vq1_1;
391
392 % % [va1_1, vb1_1, vc1_1] = qd2abcs(vq1_1, vd1_1, theta_a_1);
393 % va1_1 = vq1_1;
394 % [vb1_1, vc1_1] = statbc(vq1_1, vd1_1);
395 % % [vas_1, vbs_1, vcs_1] = qd2abcs(vqs_1, vds_1, theta_a_1);
396 % vas_1 = vqs_1;
397 % [vbs_1, vcs_1] = statbc(vqs_1, vds_1);
398
399 % 2nd IM
400 Dids_2 = [0; diff(ids_2)/deltat];
401 Diqs_2 = [0; diff(iqs_2)/deltat];
402 Did2_2 = [0; diff(id2_2)/deltat];
403 Diq2_2 = [0; diff(iq2_2)/deltat];
404
405 % lambda_d1_2 = (Ll1 + LmT)*ids_2 + LmT*id2_2;
406 % lambda_q1_2 = (Ll1 + LmT)*iqs_2 + LmT*iq2_2;
407
408 % vd1_2 = ParametersXF.r1*ids_2 + Ll1*Dids_2 + LmT*(Dids_2+Did2_2) - wa_2.*lambda_q1_2;
409 vd1_2 = ParametersXF.r1*ids_2 + Ll1*Dids_2 + LmT*(Dids_2+Did2_2);
410 vds_2 = vdC - vd1_2;
411 % vq1_2 = ParametersXF.r1*iqs_2 + Ll1*Diqs_2 + LmT*(Diqs_2+Diq2_2) + wa_2.*lambda_d1_2;
412 vq1_2 = ParametersXF.r1*iqs_2 + Ll1*Diqs_2 + LmT*(Diqs_2+Diq2_2);

```

```

413 vqs_2 = vqC - vq1_2;
414
415 % % [va1_2, vb1_2, vc1_2] = qd2abcs(vq1_2, vd1_2, theta_a_2);
416 % va1_2 = vq1_2;
417 % [vb1_2, vc1_2] = statbcs(vq1_2, vd1_2);
418 % % [vas_2, vbs_2, vcs_2] = qd2abcs(vqs_2, vds_2, theta_a_2);
419 % vas_2 = vqs_2;
420 % [vbs_2, vcs_2] = statbcs(vqs_2, vds_2);
421
422 % 3rd IM
423 Dids_3 = [0; diff(ids_3)/deltat];
424 Diqs_3 = [0; diff(iqs_3)/deltat];
425 Did2_3 = [0; diff(id2_3)/deltat];
426 Diq2_3 = [0; diff(iq2_3)/deltat];
427
428 vd1_3 = ParametersXF.r1*ids_3 + L11*Dids_3 + LmT*(Dids_3+Did2_3);
429 vds_3 = vdC - vd1_3;
430 vq1_3 = ParametersXF.r1*iqs_3 + L11*Diqs_3 + LmT*(Diqs_3+Diq2_3);
431 vqs_3 = vqC - vq1_3;
432 %-----%
433
434 % Clear variables not saved
435 clear lambda_dC_1 lambda_qC_1 lambda_dr_1 lambda_qr_1 lambda_d2_1 ...
436     lambda_q2_1 lambda_dC_2 lambda_qC_2 lambda_dr_2 lambda_qr_2 ...
437     lambda_d2_2 lambda_q2_2 lambda_dC_3 lambda_qC_3 lambda_dr_3 ...
438     lambda_qr_3 lambda_d2_3 lambda_q2_3 theta_r_1 theta_r_2 theta_r_3 ...
439     wr_1 wr_2 wr_3 Dids_1 Diqs_1 Did2_1 Diq2_1 Dids_2 Diqs_2 Did2_2 ...
440     Diq2_2 Dids_3 Diqs_3 Did2_3 Diq2_3
441
442 % Save variables before next loop (motor 2, different xfmr params)
443 TL_2_out(:,j) = TL_2;
444 Te_2_out(:,j) = Te_2;
445 delv2_vd_out(:,j) = delv2_vd;
446 delv2_vq_out(:,j) = delv2_vq;
447 delv2_cont_out(:,j) = delv2_cont;
448 wm_2_out(:,j) = wm_2;
449 theta_m_2_out(:,j) = theta_m_2;
450 ids_2_out(:,j) = ids_2;
451 iqs_2_out(:,j) = iqs_2;
452 idr_2_out(:,j) = idr_2;
453 iqr_2_out(:,j) = iqr_2;
454 id2_2_out(:,j) = id2_2;
455 iq2_2_out(:,j) = iq2_2;
456 vq1_2_out(:,j) = vq1_2;
457 vd1_2_out(:,j) = vd1_2;
458 vqs_2_out(:,j) = vqs_2;
459 vds_2_out(:,j) = vds_2;
460 Sd1_2_out(:,j) = Sd1_2;
461 Sd2_2_out(:,j) = Sd2_2;
462 Sd3_2_out(:,j) = Sd3_2;
463
464 % Save variables before next loop (motor 3, different xfmr params)
465 TL_3_out(:,j) = TL_3;
466 Te_3_out(:,j) = Te_3;
467 delv3_vd_out(:,j) = delv3_vd;
468 delv3_vq_out(:,j) = delv3_vq;
469 delv3_cont_out(:,j) = delv3_cont;
470 wm_3_out(:,j) = wm_3;
471 theta_m_3_out(:,j) = theta_m_3;
472 ids_3_out(:,j) = ids_3;
473 iqs_3_out(:,j) = iqs_3;
474 idr_3_out(:,j) = idr_3;
475 iqr_3_out(:,j) = iqr_3;
476 id2_3_out(:,j) = id2_3;
477 iq2_3_out(:,j) = iq2_3;
478 vq1_3_out(:,j) = vq1_3;
479 vd1_3_out(:,j) = vd1_3;
480 vqs_3_out(:,j) = vqs_3;
481 vds_3_out(:,j) = vds_3;
482 Sd1_3_out(:,j) = Sd1_3;
483 Sd2_3_out(:,j) = Sd2_3;
484 Sd3_3_out(:,j) = Sd3_3;
485
486 % Save variables before next loop (motor 1, different xfmr params)
487 TL_1_out(:,j) = TL_1;
488 Te_1_out(:,j) = Te_1;
489 wm_1_out(:,j) = wm_1;
490 theta_m_1_out(:,j) = theta_m_1;
491 ids_1_out(:,j) = ids_1;
492 iqs_1_out(:,j) = iqs_1;
493 idr_1_out(:,j) = idr_1;
494 iqr_1_out(:,j) = iqr_1;

```

```

495 id2_1_out(:,j) = id2_1;
496 iq2_1_out(:,j) = iq2_1;
497 vq1_1_out(:,j) = vq1_1;
498 vd1_1_out(:,j) = vd1_1;
499 vqs_1_out(:,j) = vqs_1;
500 vds_1_out(:,j) = vds_1;
501 S1_out(:,j) = S1;
502 S2_out(:,j) = S2;
503 S3_out(:,j) = S3;
504
505 theta_e_out(:,j) = theta_e;
506
507 end
508
509 % Clear redundant variables left over
510 clear delv2_cont delv2_vd delv2_vq delv3_cont delv3_vd delv3_vq i id2_1 ...
511 id2_2 id2_3 idr_1 idr_2 idr_3 ids_1 ids_2 ids_3 iq2_1 iq2_2 iq2_3 ...
512 iqr_1 iqr_2 iqr_3 iqs_1 iqs_2 iqs_3 j k N num_states S1 S2 S3 ...
513 Sd1_2 Sd2_2 Sd3_2 Sd1_3 Sd2_3 Sd3_3 t_end t_ini tspan Te_1 Te_2 Te_3 ...
514 theta_m_1 theta_m_2 theta_m_3 TL_1 TL_2 TL_3 vd1_1 vd1_2 vd1_3 ...
515 vdc vds_1 vds_2 vds_3 vq1_1 vq1_2 vq1_3 vqC vqs_1 vqs_2 vqs_3 ...
516 wm_1 wm_2 wm_3 x0 deltat theta_e
517 %-----%

1 function [y, IM1, IM2, IM3, Sw] = ThreeIM_1Converter_IFOC_volt_xf(t, ...
2 u, ParametersIM, ParametersXF, ParamSys, ParamController)
3 %-----%
4 % STATES
5 %-----%
6 % 1ST MOTOR
7 %-----%
8 lambda_dC_1 = u(1); % Flux dC (ds + d1) [Wb]
9 lambda_qC_1 = u(2); % Flux qC (qs + q1) [Wb]
10 lambda_dr_1 = u(3); % Flux dr [Wb]
11 lambda_qr_1 = u(4); % Flux qr [Wb]
12 lambda_d2_1 = u(5); % Flux d2 (xfmr side converter) [Wb]
13 lambda_q2_1 = u(6); % Flux q2 [Wb]
14 wr_1 = u(7); % Rotor speed (electric) [rad/s]
15 theta_r_1 = u(8); % Rotor position (electric) [rad]
16 %-----%
17 % INDIRECT FIELD ORIENTED CONTROL (1ST MOTOR)
18 %-----%
19 theta_e = u(9); % Rotor flux angular position (electric) [rad]
20 I_Action_CL_w1 = u(10); % Integral action of the closed loop control of wm
21 %-----%
22 %
23 % 2ND MOTOR
24 %-----%
25 lambda_dC_2 = u(11); % Flux dC (ds + d1) [Wb]
26 lambda_qC_2 = u(12); % Flux qC (qs + q1) [Wb]
27 lambda_dr_2 = u(13); % Flux dr [Wb]
28 lambda_qr_2 = u(14); % Flux qr [Wb]
29 lambda_d2_2 = u(15); % Flux d2 (xfmr side converter) [Wb]
30 lambda_q2_2 = u(16); % Flux q2 [Wb]
31 wr_2 = u(17); % Rotor speed (electric) [rad/s]
32 theta_r_2 = u(18); % Rotor position (electric) [rad]
33 %-----%
34 %
35 % 3RD MOTOR
36 %-----%
37 lambda_dC_3 = u(19); % Flux dC (ds + d1) [Wb]
38 lambda_qC_3 = u(20); % Flux qC (qs + q1) [Wb]
39 lambda_dr_3 = u(21); % Flux dr [Wb]
40 lambda_qr_3 = u(22); % Flux qr [Wb]
41 lambda_d2_3 = u(23); % Flux d2 (xfmr side converter) [Wb]
42 lambda_q2_3 = u(24); % Flux q2 [Wb]
43 wr_3 = u(25); % Rotor speed (electric) [rad/s]
44 theta_r_3 = u(26); % Rotor position (electric) [rad]
45 %-----%
46 %
47 % CONTROLLER
48 %-----%
49 n = ParamController.order;
50 % Automatic state-building of the controller, based on its order
51 xk2 = u(27:27+n-1);
52 xk2 = reshape(xk2,n,1);
53 xk3 = u(27+n:27+2*n-1);
54 xk3 = reshape(xk3,n,1);
55 %-----%
56 %
57 %
58 % GET SYSPARAMS

```

```

59 %-----%
60
61 %-----%
62 vswM = ParamSys.vswM;
63 vdM = ParamSys.vdM;
64 vswD = ParamSys.vswD;
65 vdD = ParamSys.vdD;
66
67 TL1_delta = ParamSys.TL1_variation;
68 time_delta_TL1 = ParamSys.time_variation_TL1;
69
70 TL2_delta = ParamSys.TL2_variation;
71 time_delta_TL2 = ParamSys.time_variation_TL2;
72
73 TL3_delta = ParamSys.TL3_variation;
74 time_delta_TL3 = ParamSys.time_variation_TL3;
75
76 if t > time_delta_TL1
77     TL_1 = (1+TL1_delta)*ParametersIM.Tnom;
78 else
79     % TL_1 = ParametersIM.Tnom;
80     TL_1 = 0;
81 end
82
83 if t > time_delta_TL2
84     TL_2 = (1+TL2_delta)*ParametersIM.Tnom;
85 else
86     % TL_2 = ParametersIM.Tnom;
87     TL_2 = 0;
88 end
89
90 if t > time_delta_TL3
91     TL_3 = (1+TL3_delta)*ParametersIM.Tnom;
92 else
93     % TL_3 = ParametersIM.Tnom;
94     TL_3 = 0;
95 end
96 %-----%
97 %-----%
98 % GET CONTROLLER PARAMS
99 %-----%
100 % Hysteresis control
101 h = ParamController.h; % Hysteresis band
102
103 % r_ext control
104 Ak = ParamController.Ak;
105 Bk = ParamController.Bk;
106 Ck = ParamController.Ck;
107 Dk = ParamController.Dk;
108 %-----%
109 % Arbitrary Reference Frame (unused)
110 %-----%
111 % Stator reference
112 %-----%
113 % 1st IM
114 % wa_1 = 0;
115 % theta_a_1 = 0;
116
117 % 2nd IM
118 % wa_2 = 0;
119 % theta_a_2 = 0;
120 %-----%
121
122 %-----%
123 % Auxiliary Equations for the IMs
124 %-----%
125 % Transformation of fluxes to currents — DQ components
126 Linv = ParametersXF.Linv;
127 % 1ST MOTOR
128 [ids_1, iqs_1, idr_1, iqr_1, id2_1, iq2_1] = Fluxes2Currents_xf(lambda_dC_1, ...
129     lambda_qC_1, lambda_dr_1, lambda_qr_1, lambda_d2_1, lambda_q2_1, Linv);
130 % 2ND MOTOR
131 [ids_2, iqs_2, idr_2, iqr_2, id2_2, iq2_2] = Fluxes2Currents_xf(lambda_dC_2, ...
132     lambda_qC_2, lambda_dr_2, lambda_qr_2, lambda_d2_2, lambda_q2_2, Linv);
133 % 3RD MOTOR
134 [ids_3, iqs_3, idr_3, iqr_3, id2_3, iq2_3] = Fluxes2Currents_xf(lambda_dC_3, ...
135     lambda_qC_3, lambda_dr_3, lambda_qr_3, lambda_d2_3, lambda_q2_3, Linv);
136
137 % Electromagnetic and Load Torques
138 % Need to use rotor flux & current since stator flux isn't calculated
139 % 1ST MOTOR
140 Te_1 = 3/4*ParametersIM.P*(lambda_qr_1.*idr_1 - lambda_dr_1.*iqr_1);

```

```

141 % 2ND MOTOR
142 Te_2 = 3/4*ParametersIM.P*(lambda_qr_2.*idr_2 - lambda_dr_2.*iqr_2);
143 % 3RD MOTOR
144 Te_3 = 3/4*ParametersIM.P*(lambda_qr_3.*idr_3 - lambda_dr_3.*iqr_3);
145
146 % Transformation of electrical rotor speed and electrical angular position
147 % to mechanical rotor speed and mechanical angular position
148 % 1st IM
149 % wm_1 = wr_1/(ParametersIM.P/2);
150 theta_m_1 = theta_r_1/(ParametersIM.P/2);
151 % 2nd IM
152 % wm_2 = wr_2/(ParametersIM.P/2);
153 theta_m_2 = theta_r_2/(ParametersIM.P/2);
154 % 3rd IM
155 % wm_2 = wr_2/(ParametersIM.P/2);
156 theta_m_3 = theta_r_3/(ParametersIM.P/2);
157
158 % Transforming stator and rotor currents from qd reference to abc variables
159 % Replaced with stationary assumption for efficiency
160 % 1st IM
161 % [ias_1, ibs_1, ics_1] = qd2abcs(iqs_1, ids_1, theta_a_1);
162 [ibs_1, ics_1] = statbc(iqs_1, ids_1);
163 % 2nd IM
164 % [ias_2, ibs_2, ics_2] = qd2abcs(iqs_2, ids_2, theta_a_2);
165 [ibs_2, ics_2] = statbc(iqs_2, ids_2);
166 % 3rd IM
167 % [ias_3, ibs_3, ics_3] = qd2abcs(iqs_3, ids_3, theta_a_3);
168 [ibs_3, ics_3] = statbc(iqs_3, ids_3);
169
170 % 1st XFMR
171 % [ia2_1, ib2_1, ic2_1] = qd2abcs(iq2_1, id2_1, theta_a_1);
172 [ib2_1, ic2_1] = statbc(iq2_1, id2_1);
173 % 2nd XFMR
174 % [ia2_2, ib2_2, ic2_2] = qd2abcs(iq2_2, id2_2, theta_a_2);
175 [ib2_2, ic2_2] = statbc(iq2_2, id2_2);
176 % 3rd XFMR
177 % [ia2_3, ib2_3, ic2_3] = qd2abcs(iq2_3, id2_3, theta_a_3);
178 [ib2_3, ic2_3] = statbc(iq2_3, id2_3);
179
180 % Rotor Voltages (squirrel cage) — DQ components
181 % 1st IM
182 vdr_1 = 0;
183 vqr_1 = 0;
184 % 2nd IM
185 vdr_2 = 0;
186 vqr_2 = 0;
187 % 3rd IM
188 vdr_3 = 0;
189 vqr_3 = 0;
190
191 % Transformer secondary voltages are defined in position sync section
192 %-----%
193
194 %-----%
195 % INDIRECT (ROTOR) FIELD ORIENTED CONTROL
196 % INDUCTION MOTOR 1
197 %-----%
198 [lambda_dr_star, wm_star] = RefLambdaAndwm(t, ParametersIM);
199
200 % Indirect field oriented control drive
201 [iqs_star, ids_star, di_wm_dt, wrf_est] = IFOCDrive(ParametersIM, ...
202 lambda_dr_star, wm_star, wr_1, I_Action_CL_w_1);
203
204 % Converting currents to abc variables for Hysteresis current control
205 % Commanded currents
206 [ias_star, ibs_star, ics_star] = qd2abcs(iqs_star, ids_star, theta_e);
207 i_abcs_star = [ias_star, ibs_star, ics_star];
208
209 % Measured/real currents
210 i_abcs = [iqs_1, ibs_1, ics_1];
211
212 % Hysteresis current control
213 [S] = myHysteresisCurrentControl(i_abcs', i_abcs_star', h);
214
215 % 3phase bridge converter voltages
216 vdc = ParametersIM.Vs*sqrt(2); % peak of line-to-line voltage
217 IM = [(iqs_1+iqs_2+iqs_3) (ibs_1+ibs_2+ibs_3) (ics_1+ics_2+ics_3)];
218 [vas, vbs, vcs] = my3PhaseConverterVoltLoss(S', vdc, vswM, vdM, IM);
219
220 % Converting vabcs voltages to vqds
221 [vqsC, vdsC] = statqd(vas, vbs, vcs);
222

```

```

223 %-----%
224
225 %-----%
226 %           VOLTAGE ADJUSTMENT CONTROL
227 %           INDUCTION MOTOR 2
228 %-----%
229
230 %-----%
231 %           CONTROLLER EQUATIONS
232 %-----%
233 % Controller input vector uk
234 % For now, only the position error
235
236 uk2(1) = theta_m_1 - theta_m_2;
237 xk2_dot = Ak*xk2 + Bk*uk2;
238 yk2 = Ck*xk2 + Dk*uk2;
239
240 uk3(1) = theta_m_1 - theta_m_3;
241 xk3_dot = Ak*xk3 + Bk*uk3;
242 yk3 = Ck*xk3 + Dk*uk3;
243
244 % Controller output vector yk
245 % For now, only motor 2/3 voltage adjustment
246 % delv2_cont: continuous variable voltage drop, ideal target
247 % For PI controller, yk is in units of V/rad
248 delv2_cont = yk2(1);
249 delv2_cont = max(delv2_cont,0);
250 delv2_cont = min(delv2_cont,ParamController.delv_max);
251
252 delv3_cont = yk3(1);
253 delv3_cont = max(delv3_cont,0);
254 delv3_cont = min(delv3_cont,ParamController.delv_max);
255 %-----%
256
257 % Calculate voltage drops for motor 1 using dummy converter for symmetry
258 S_d2 = [false false false true true true];
259 % ID = [ia2_1, ib2_1, ic2_1];
260 ID = [iq2_1, ib2_1, ic2_1];
261 [vad, vbd, vcd] = my3PhaseConverterVoltLoss(S_d2, 0, vswD, vdD, ID);
262 % Transforming vabcs to vds and vqs
263 % Replaced with stationary assumption for efficiency
264 % [vqd, vdd] = abcs2qd(vad, vbd, vcd, theta_a_1);
265 [vqd_1, vdd_1] = statqd(vad, vbd, vcd);
266 % Refer variable
267 vq2_1 = (vqd_1*ParametersXF.N);
268 vd2_1 = (vdd_1*ParametersXF.N);
269 if t > ParamController.tstart
270     % Controlable voltage drop via sine-triangle PWM, assuming synchronization
271     f_d = 4.987654321e3;
272     vdc_d = vdc * ParametersXF.vdcR;
273
274     % Motor 2
275     vqd_e = delv2_cont / ParametersXF.N;
276     vdd_e = 0;
277     [S_d2,~,~,~,~] = my3HarmonicSineTri_v2(t, theta_e, vqd_e, vdd_e, vdc_d, f_d);
278     % ID = [ia2_2, ib2_2, ic2_2];
279     ID = [iq2_2, ib2_2, ic2_2];
280     [vad, vbd, vcd] = my3PhaseConverterVoltLoss(S_d2, vdc_d, vswD, vdD, ID);
281
282     % Transforming vabcs to vds and vqs
283     % Replaced with stationary assumption for efficiency
284     % [vqd, vdd] = abcs2qd(vad, vbd, vcd, theta_a_2);
285     [vqd_2, vdd_2] = statqd(vad, vbd, vcd);
286
287     % Refer variable
288     vq2_2 = (vqd_2*ParametersXF.N);
289     vd2_2 = (vdd_2*ParametersXF.N);
290
291     % Motor 3
292     vqd_e = delv3_cont / ParametersXF.N;
293     vdd_e = 0;
294     [S_d3,~,~,~,~] = my3HarmonicSineTri_v2(t, theta_e, vqd_e, vdd_e, vdc_d, f_d);
295     % ID = [ia2_3, ib2_3, ic2_3];
296     ID = [iq2_3, ib2_3, ic2_3];
297     [vad, vbd, vcd] = my3PhaseConverterVoltLoss(S_d3, vdc_d, vswD, vdD, ID);
298
299     % Transforming vabcs to vds and vqs
300     % Replaced with stationary assumption for efficiency
301     % [vqd, vdd] = abcs2qd(vad, vbd, vcd, theta_a_2);
302     [vqd_3, vdd_3] = statqd(vad, vbd, vcd);
303
304     % Refer variable

```

```

305     vq2_3 = (vqd_3*ParametersXF.N);
306     vd2_3 = (vdd_3*ParametersXF.N);
307 else
308     % Voltage drop is only active when torques are different
309     % Since motors are identical in this state, use same voltages
310     S_d3 = S_d2;
311     vqd_2 = vqd_1;
312     vdd_2 = vdd_1;
313     vqd_3 = vqd_1;
314     vdd_3 = vdd_1;
315     vq2_2 = vq2_1;
316     vd2_2 = vd2_1;
317     vq2_3 = vq2_1;
318     vd2_3 = vd2_1;
319 end
320 %-----%
321 %-----%
322 %-----%
323 %                               OUTPUT STRUCTURES
324 %-----%
325 % Switching states
326 Sw.S = S;
327 Sw.S_d2 = S_d2;
328 Sw.S_d3 = S_d3;
329
330 % 1st motor
331 IM1.TL = TL_1;
332 IM1.Te = Te_1;
333 IM1.vqC = vqsC;
334 IM1.vdC = vdsC;
335 % IM1.wa = wa_1;
336 % IM1.theta_a = theta_a_1;
337 IM1.delv1_vq = vqd_1;
338 IM1.delv1_vd = vdd_1;
339 IM1.wm_star = wm_star;
340
341 % 2nd motor
342 IM2.TL = TL_2;
343 IM2.Te = Te_2;
344 % IM2.wa = wa_2;
345 % IM2.theta_a = theta_a_2;
346 IM2.delv2_vq = vqd_2;
347 IM2.delv2_vd = vdd_2;
348 IM2.delv2_cont = delv2_cont;
349
350 % 3rd motor
351 IM3.TL = TL_3;
352 IM3.Te = Te_3;
353 % IM3.wa = wa_3;
354 % IM3.theta_a = theta_a_3;
355 IM3.delv3_vq = vqd_3;
356 IM3.delv3_vd = vdd_3;
357 IM3.delv3_cont = delv3_cont;
358
359 %-----%
360 %-----%
361 %-----%
362 %                               DIFF. STATES EQUATIONS
363 %-----%
364 %-----%
365 % Extracting parameters here, assuming identical motors
366 rs = ParametersIM.rs;
367 rr = ParametersIM.rr;
368 r1 = ParametersXF.r1;
369 r2 = ParametersXF.r2;
370 J = ParametersIM.J;
371 P = ParametersIM.P;
372 b = ParametersIM.b;
373
374 % Equations used are altered to assume stationary ref. frame for efficiency
375 %-----%
376 %                               1ST MOTOR
377 %-----%
378 % y(1) = vdC_1 - (rs+r1)*ids_1 + wa_1 * lambda_qC_1;           % lambda_dC_1
379 % y(2) = vqC_1 - (rs+r1)*iqs_1 - wa_1 * lambda_dC_1;           % lambda_qC_1
380 % y(3) = vdr_1 - rr*idr_1 + (wa_1-wr_1)*lambda_qr_1;           % lambda_dr_1
381 % y(4) = vqr_1 - rr*iqr_1 - (wa_1-wr_1)*lambda_dr_1;           % lambda_qr_1
382 % y(5) = vd2_1 - r2*id2_1 + wa_1 * lambda_q2_1;               % lambda_d2_1
383 % y(6) = vq2_1 - r2*iq2_1 - wa_1 * lambda_d2_1;               % lambda_q2_1
384 y(1) = vdsC - (rs+r1)*ids_1;                                     % lambda_dC_1
385 y(2) = vqsC - (rs+r1)*iqs_1;                                     % lambda_qC_1
386 y(3) = vdr_1 - rr*idr_1 - wr_1*lambda_qr_1;                     % lambda_dr_1

```

```

387 y(4) = vqr_1 - rr*iqr_1 + wr_1*lambda_dr_1; % lambda_qr_1
388 y(5) = vd2_1 - r2*id2_1; % lambda_d2_1
389 y(6) = vq2_1 - r2*iq2_1; % lambda_q2_1
390 y(7) = (1/J)*((Te_1-TL_1)*P/2 - b*wr_1); % wr_1
391 y(8) = wr_1; % theta_r_1
392 %
393 %-----
394 % INDIRECT FIELD ORIENTED CONTROL (1ST MOTOR)
395 %-----
396 y(9) = wrf_est; % Rotor flux angular position (electric)
397 y(10) = di_wm_dt; % Integral action of the closed loop control of wm
398 %
399 %-----
400 % 2ND MOTOR
401 %-----
401 % y(11) = vdc_1 - (rs+r1)*ids_2 + wa_2 * lambda_qC_2; % lambda_dC_2
402 % y(12) = vqc_1 - (rs+r1)*iqs_2 - wa_2 * lambda_dC_2; % lambda_qC_2
403 % y(13) = vdr_2 - rr*idr_2 + (wa_2-wr_2)*lambda_qr_2; % lambda_dr_2
404 % y(14) = vqr_2 - rr*iqr_2 - (wa_2-wr_2)*lambda_dr_2; % lambda_qr_2
405 % y(15) = vd2_2 - r2*id2_2 + wa_2 * lambda_q2_2; % lambda_d2_2
406 % y(16) = vq2_2 - r2*iq2_2 - wa_2 * lambda_d2_2; % lambda_q2_2
407 y(11) = vdsC - (rs+r1)*ids_2; % lambda_dC_2
408 y(12) = vqsC - (rs+r1)*iqs_2; % lambda_qC_2
409 y(13) = vdr_2 - rr*idr_2 - wr_2*lambda_qr_2; % lambda_dr_2
410 y(14) = vqr_2 - rr*iqr_2 + wr_2*lambda_dr_2; % lambda_qr_2
411 y(15) = vd2_2 - r2*id2_2; % lambda_d2_2
412 y(16) = vq2_2 - r2*iq2_2; % lambda_q2_2
413 y(17) = (1/J)*((Te_2-TL_2)*P/2 - b*wr_2); % wr_2
414 y(18) = wr_2; % theta_r_2
415 %
416 %-----
417 % 3RD MOTOR
418 %-----
418 y(19) = vdsC - (rs+r1)*ids_3; % lambda_dC_3
419 y(20) = vqsC - (rs+r1)*iqs_3; % lambda_qC_3
420 y(21) = vdr_3 - rr*idr_3 - wr_3*lambda_qr_3; % lambda_dr_3
421 y(22) = vqr_3 - rr*iqr_3 + wr_3*lambda_dr_3; % lambda_qr_3
422 y(23) = vd2_3 - r2*id2_3; % lambda_d2_3
423 y(24) = vq2_3 - r2*iq2_3; % lambda_q2_3
424 y(25) = (1/J)*((Te_3-TL_3)*P/2 - b*wr_3); % wr_3
425 y(26) = wr_3; % theta_r_3
426 %
427 %-----
428 % CONTROLLER
429 %-----
429 % Automatic state-building diff. equations of the controller, based on its order
430 y(27:27+n-1) = xk2_dot;
431 y(27+n:27+2*n-1) = xk3_dot;
432 %
433 %
434 y = y';
435 end

```

B.5 Auxiliary Functions

```

1 % Inputs are the abc phase variables in the stationary reference frame and
2 % the angle of the arbitrary rotating reference frame theta.
3 % Outputs are the qd variables in the (arbitrary) rotating reference
4 % frame.
5 function [x_q, x_d] = abcs2qd(x_as, x_bs, x_cs, theta)
6
7 % Reshaping the vectors so all of them are column vectors
8 n = numel(theta);
9 theta = reshape(theta,n,1);
10 x_as = reshape(x_as,n,1);
11 x_bs = reshape(x_bs,n,1);
12 x_cs = reshape(x_cs,n,1);
13
14 x_q = 2/3*(x_as.*cos(theta)+x_bs.*cos(theta-2*pi/3)+x_cs.*cos(theta+2*pi/3));
15 x_d = 2/3*(x_as.*sin(theta)+x_bs.*sin(theta-2*pi/3)+x_cs.*sin(theta+2*pi/3));
16 end

```

```

1 function wrm_star = DesiredSpeed(t)
2
3 NumPoints = length(t);
4 wrm_star = zeros(NumPoints,1);
5
6 % Commanded speed set as a ramp from t=0.1 to t=2.6 with final value of
7 % 188.5 rad/s
8 t_1 = 0;
9 t_2 = 0.1;

```

```

10 wrm_final = 188.5;
11 slope = wrm_final/(t_2-t_1);
12
13 for i = 1:NumPoints
14     if t(i) >= t_1 && t(i) < t_2
15         wrm_star(i) = slope*(t(i)-t_1);
16     elseif t(i) >= t_2
17         wrm_star(i) = wrm_final;
18     end
19 end
20
21 end

1 % [Translated from original description in Portugese]
2 % Function that transforms fluxes to currents using passed machine
3 % parameters. The fluxes and currents are in DQ components.
4 % Flux can be passed as 4 vectors, one for each flux value,
5 % in order: lambda_ds, lambda_qs, lambda_dr, lambda_qr; or a matrix,
6 % with 4 columns, referring to each of the fluxes, and as many
7 % rows as needed to describe time behavior of fluxes. The output currents
8 % will match the inputs, and may be scalars or vectors.
9 function [ids, iqs, idr, iqr]= Fluxes2Currents(lambda_ds, lambda_qs, ...
10         lambda_dr, lambda_qr, ParamatersIM)
11
12 NumPoints = length(lambda_ds);
13
14 ids = zeros(NumPoints,1);
15 iqs = zeros(NumPoints,1);
16 idr = zeros(NumPoints,1);
17 iqr = zeros(NumPoints,1);
18
19 for i = 1:NumPoints
20
21     D = ParamatersIM.Ls*ParamatersIM.Lr-ParamatersIM.Lm^2;
22     MatrizTransformation = 1/D*[ParamatersIM.Lr -ParamatersIM.Lm;
23                               -ParamatersIM.Lm ParamatersIM.Ls];
24
25     Vetor_Correntes_d = MatrizTransformation*[lambda_ds(i); lambda_dr(i)];
26     Vetor_Correntes_q = MatrizTransformation*[lambda_qs(i); lambda_qr(i)];
27
28     ids(i,1) = Vetor_Correntes_d(1); % i_ds
29     iqs(i,1) = Vetor_Correntes_q(1); % i_qs
30     idr(i,1) = Vetor_Correntes_d(2); % i_dr
31     iqr(i,1) = Vetor_Correntes_q(2); % i_qr
32 end
33 end

1 function [ids, iqs, idr, iqr, id2, iq2]= Fluxes2Currents_xf(lambda_dC, ...
2     lambda_qC, lambda_dr, lambda_qr, lambda_d2, lambda_q2, Linv)
3
4 NumPoints = numel(lambda_dC);
5
6 ids = zeros(NumPoints,1);
7 iqs = zeros(NumPoints,1);
8 idr = zeros(NumPoints,1);
9 iqr = zeros(NumPoints,1);
10 id2 = zeros(NumPoints,1);
11 iq2 = zeros(NumPoints,1);
12
13 for i = 1:NumPoints
14     Currents_d = Linv*[lambda_dC(i); lambda_dr(i); lambda_d2(i)];
15     Currents_q = Linv*[lambda_qC(i); lambda_qr(i); lambda_q2(i)];
16
17     ids(i,1) = Currents_d(1); % i_ds
18     iqs(i,1) = Currents_q(1); % i_qs
19     idr(i,1) = Currents_d(2); % i_dr
20     iqr(i,1) = Currents_q(2); % i_qr
21     id2(i,1) = Currents_d(3); % i_d2
22     iq2(i,1) = Currents_q(3); % i_q2
23 end
24 end

1 function [iqs_star, ids_star, di_wm_dt, we, Tem_ref] = IFOCDrive(ParametersIM, ...
2     lambda_dr_star, wm_star, wr, I_Action_CL_w, t)
3 % Serves for both single time and time vector
4
5 %-----%
6 %                CLOSE-LOOP SPEED CONTROL
7 %-----%
8 % Tunning kp_wm and ki_wm of the PI controller
9

```

```

10 tau_wm = 0.1;
11 % tau_wm = 0.02;
12 p1 = 1/(tau_wm);
13 p2 = 5*p1;
14 kp_wm = (p1+p2)*ParametersIM.J_est - ParametersIM.b_est;
15 ki_wm = p1*p2*ParametersIM.J_est;
16
17 % Speed error to feed the PI controller
18 wm = wr/(ParametersIM.P/2); % rotor electrical speed to mechanical speed
19 wm_error = wm_star-wm;
20
21 % PI action on speed error to generate commanded torque
22 Tem_ref = kp_wm*wm_error + I_Action_CL_w;
23
24 % Saturation and anti-windup effect
25 % Max and min torque values (saturation):
26 Te_max = 10*ParametersIM.Tnom;
27 Te_min = -Te_max;
28
29 di_wm_dt = zeros(1,length(lambda_dr_star));
30 for i = 1:length(Tem_ref)
31
32     % Saturation at Te_max
33     Tem_ref(i) = min(Tem_ref(i), Te_max);
34
35     % Saturation at Te_min
36     Tem_ref(i) = max(Tem_ref(i), Te_min);
37
38     % Anti-windup integration:
39     if (Tem_ref(i) == Te_max) && (wm_error(i)*I_Action_CL_w(i))>0
40         di_wm_dt(i) = 0;
41
42     elseif (Tem_ref(i) == Te_min) && (wm_error(i)*I_Action_CL_w(i))>0
43         di_wm_dt(i) = 0;
44
45     else
46         di_wm_dt(i) = ki_wm*wm_error(i);
47
48     end
49
50 end
51 %-----%
52 %-----%
53 %-----%
54 % iqs_star and ids_star Calculations
55 %-----%
56
57 % ids_star depending on the commanded flux (algebraic relationship)
58 ids_star = lambda_dr_star/ParametersIM.Lm_est;
59
60 % iqs_star depending on the commanded torque (algebraic relationship)
61 k_Tem = 3/4*ParametersIM.P*ParametersIM.Lm_est/ParametersIM.Lr_est;
62 iqs_star = Tem_ref/k_Tem./(lambda_dr_star+eps);
63 %-----%
64 %-----%
65 %-----%
66 % Slip/we Calculations
67 %-----%
68 % Rotor flux speed estimation (indirect control)
69 we = ParametersIM.rr_est/ParametersIM.Lr_est*(iqs_star./(ids_star+eps))+wr;
70
71 %-----%
72 %-----%
73 end

1 function [S, da, db, dc, tri] = my3HarmonicSineTri_v2(t, theta_e, vqs_e, vds_e, Vdc, f_sw)
2 %-----%
3 %
4 % FUNCTION DESCRIPTION
5 %-----%
6 % Function that implements sine-triangle modulation with third-harmonic
7 % injection
8 %
9 % Reference: Analysis of Electric Machinery and Drive Systems, P.C Krause,
10 % 2nd Edition, Chapter 13
11 %
12 % INPUTS:
13 % t: simulation time [s]
14 % theta_e: electrical angle of the commanded voltages [rad]
15 % vqs_e, vds_e: qds input voltages in the synchronous reference frame (commanded values) [V]
16 % Vdc: power converter DC voltage (peak of the line-to-line voltage) [V]
17 % f_sw: switching frequency [Hz]
18 %

```

```

18 % OUTPUTS:
19 % S: vector with the 6 switching states: T1 – T6
20 %-----%
21
22 %-----%
23 % AUXILIARY EQUATIONS
24 %-----%
25 % Converter angle (electrical angle minus the angle of the complex vector
26 % vqds_e)
27 theta_c = theta_e - angle(vqs_e - 1i*vds_e);
28
29 % Limits the angle in the interval [0, 2*pi) TODO: Maybe this is dispensable...
30 theta_c = mod(theta_c, 2*pi);
31
32 % Frequency of the triangle waveform
33 omega_tri = 2*pi*f_sw; % [rad/s]
34
35 % Amplitude of the main voltage
36 d = sqrt(vqs_e.^2 + vds_e.^2)/(0.5*Vdc);
37
38 % Amplitude of third-harmonic injection
39 d3 = d/6;
40 %-----%
41
42 % 1st: Generate sawtooth (triangle waveform), tri (between -1 and 1)
43 tri = sawtooth(omega_tri*t, 0.5);
44
45 % 2nd: Third harmonic injection
46 da = d*cos(theta_c) - d3*cos(3*theta_c);
47 db = d*cos(theta_c - 2*pi/3) - d3*cos(3*theta_c);
48 dc = d*cos(theta_c + 2*pi/3) - d3*cos(3*theta_c);
49
50 % 3rd: Generate the output logical signals: T1 – T6
51 % a-phase
52 T1 = da>= tri;
53 T4 = ~T1;
54
55 % b-phase
56 T2 = db>= tri;
57 T5 = ~T2;
58
59 % c-phase
60 T3 = dc>= tri;
61 T6 = ~T3;
62
63 % 4th: Build output vector S with the logical signals T1–T6
64 S = [T1, T2, T3, T4, T5, T6];
65
66 end

```

```

1 function [vas, vbs, vcs] = my3PhaseConverterVoltLoss(S, vdc, vsw, vd, I)
2 %-----%
3 % FUNCTION DESCRIPTION
4 %-----%
5 % Function that determines the 3-phase bridge converter output voltages.
6 % Incorporates switching losses.
7 %
8 % Reference: Analysis of Electric Machinery and Drive Systems, P.C Krause,
9 % 2nd Edition, Chapter 13
10 %
11 % INPUTS:
12 % S: vector with the 6 switching states: T1 – T6
13 % vdc: dc voltage applied to the converter bridge [V]
14 % vsw: switch voltage drop (for "forward" current)
15 % vd: diode voltage drop (for "backward" current)
16 % I: vector with the 3 currents (ias, ibs, ics)
17
18 % OUTPUTS:
19 % vas, vbs, vcs: line-to-neutral 3-phase voltages
20 %-----%
21
22 % 1st: Generate line-to-ground voltages
23 if S(1)
24     vag = vdc - (vsw * (I(1)>0)) + (vd * (I(1)<0));
25 else
26     vag = - (vd * (I(1)>0)) + (vsw * (I(1)<0));
27 end
28 if S(2)
29     vbg = vdc - (vsw * (I(2)>0)) + (vd * (I(2)<0));
30 else
31     vbg = - (vd * (I(2)>0)) + (vsw * (I(2)<0));
32 end

```

```

33 if S(3)
34     vcg = vdc - (vsw * (I(3)>0)) + (vd * (I(3)<0));
35 else
36     vcg = - (vd * (I(3)>0)) + (vsw * (I(3)<0));
37 end
38
39 % 2nd: Generate line-to-neutral voltages
40 vas = 2/3*vag - 1/3*(vbg + vcg);
41 vbs = 2/3*vbg - 1/3*(vag + vcg);
42 vcs = 2/3*vcg - 1/3*(vag + vbg);
43
44 end

1 function [S] = myHysteresisCurrentControl(i_abcs, i_abcs_star, h)
2 % Serves for both single time and time vector
3
4 % Way to hold previous state
5 persistent S_old;
6 if isempty(S_old)
7     S_old = [0; 1; 0; 1; 0; 1];
8 end
9
10 % In case of time vector case, has to allocate memory
11 num_points = numel(i_abcs)/3;
12 T1 = zeros(1,num_points);
13 T2 = zeros(1,num_points);
14 T3 = zeros(1,num_points);
15 T4 = zeros(1,num_points);
16 T5 = zeros(1,num_points);
17 T6 = zeros(1,num_points);
18 S = zeros(6,num_points);
19
20
21 for i=1:num_points
22     % Current i_as and switches T1 and T4
23     if i_abcs(1,i) > i_abcs_star(1,i) + h
24         T1(i) = 0;
25         T4(i) = 1;
26     elseif i_abcs(1,i) < i_abcs_star(1,i) - h
27         T1(i) = 1;
28         T4(i) = 0;
29     else
30         T1(i) = S_old(1);
31         T4(i) = S_old(4);
32     end
33
34     % Current i_bs and switches T2 and T5
35     if i_abcs(2,i) > i_abcs_star(2,i) + h
36         T2(i) = 0;
37         T5(i) = 1;
38     elseif i_abcs(2,i) < i_abcs_star(2,i) - h
39         T2(i) = 1;
40         T5(i) = 0;
41     else
42         T2(i) = S_old(2);
43         T5(i) = S_old(5);
44     end
45
46     % Current i_cs and switches T3 and T6
47     if i_abcs(3,i) > i_abcs_star(3,i) + h
48         T3(i) = 0;
49         T6(i) = 1;
50     elseif i_abcs(3,i) < i_abcs_star(3,i) - h
51         T3(i) = 1;
52         T6(i) = 0;
53     else
54         T3(i) = S_old(3);
55         T6(i) = S_old(6);
56     end
57
58     % Build output vector S from the logical signals T1-T6
59     S(:,i) = [T1(i); T2(i); T3(i); T4(i); T5(i); T6(i)];
60
61     S_old = S(:,i);
62 end
63
64 end

1 function [S] = myPWM(t, d, f_sw)
2 %-----%
3 %                FUNCTION DESCRIPTION
4 %-----%

```

```

5 % Function that implements pulse-width modulation.
6 %
7 % Reference: Analysis of Electric Machinery and Drive Systems, P.C Krause,
8 % 2nd Edition, Chapter 13
9 %
10 % INPUTS:
11 % t: time value [s]
12 % omega_c: converter frequency [rad/s]
13 % d: duty cycle
14 % f_sw: switching frequency [Hz]
15 %
16 % OUTPUTS:
17 % S: vector with the 6 switching states: T1 – T6
18 %-----%
19 %-----%
20 %-----%
21 % AUXILIARY EQUATIONS
22 %-----%
23 % Frequency of the triangle waveform
24 omega_tri = 2*pi*f_sw; % [rad/s]
25 %-----%
26 %
27 % 2nd: Generate sawtooth (triangle waveform), tri
28 tri = 0.5 + 0.5*sawtooth(omega_tri * t, 0.5);
29 %
30 % 3rd: Generate comparator output, c
31 threshold = 1e-4;
32 c = d > (tri + threshold);
33 %
34 % 4th: Generate the output logical signals: T1 – T6
35 T1 = c;
36 %
37 % 5th: Build output vector S from the logical signals T1–T6
38 S = T1;
39 %
40 end

1 % Inputs are the qd variables in the arbitrary reference frame and angle
2 % of the arbitrary rotating reference frame theta.
3 % Outputs are the abc variables in the stationary reference frame
4 function [x_as, x_bs, x_cs] = qd2abcs(x_q, x_d, theta)
5 %
6 % Reshaping the vectors so all of them are column vectors
7 n = numel(theta);
8 theta = reshape(theta,n,1);
9 x_q = reshape(x_q,n,1);
10 x_d = reshape(x_d,n,1);
11 %
12 x_as = x_q.*cos(theta) + x_d.*sin(theta);
13 x_bs = x_q.*cos(theta-2*pi/3) + x_d.*sin(theta-2*pi/3);
14 x_cs = x_q.*cos(theta+2*pi/3) + x_d.*sin(theta+2*pi/3);
15 end

1 % Inputs are the qd variables in the old arbitrary reference frame and
2 % difference in angle between the reference frames (new – old).
3 % Outputs are the qd variables in the new reference frame
4 function [y_q, y_d] = qd2qd(x_q,x_d,theta)
5 %
6 n = numel(theta);
7 theta = reshape(theta,n,1);
8 x_q = reshape(x_q,n,1);
9 x_d = reshape(x_d,n,1);
10 %
11 y_q = x_q.*cos(theta) - x_d.*sin(theta);
12 y_d = x_q.*sin(theta) + x_d.*cos(theta);
13 end

1 function [lambda_dr_ref, wm_ref] = RefLambdaAndwm(time, ParametersIM)
2 %
3 lambda_nom = sqrt(2/3)*ParametersIM.Vs/(2*pi*ParametersIM.f);
4 t_lambda_nom = 0.5;
5 lambda_ref_slope = lambda_nom/t_lambda_nom;
6 %
7 wm = 1800*pi/30;
8 %
9 t_1_wm = 0.6;
10 t_2_wm = 1.5;
11 %
12 % t_1_wm = 1;
13 % t_2_wm = 1.5;
14 %

```

```

15  wm_slope_acc = wm/(t_2_wm - t_1_wm);
16
17  t_3_wm = 100;
18  t_4_wm = 200;
19  wm_slope_des = wm/(t_4_wm - t_3_wm);
20
21  NumPoints = length(time);
22
23  lambda_dr_ref = zeros(NumPoints,1);
24  wm_ref = zeros(NumPoints,1);
25
26
27  for i = 1:NumPoints
28      if time(i) < t_lambda_nom
29          lambda_dr_ref(i) = lambda_ref_slope*time(i);
30          wm_ref(i) = 0;
31      elseif time(i) >= t_lambda_nom && time(i) < t_1_wm
32          lambda_dr_ref(i) = lambda_nom;
33          wm_ref(i) = 0;
34      elseif time(i) >= t_1_wm && time(i) < t_2_wm
35          lambda_dr_ref(i) = lambda_nom;
36          wm_ref(i) = wm_slope_acc*(time(i)-t_1_wm);
37      elseif time(i) >= t_2_wm && time(i) < t_3_wm
38          lambda_dr_ref(i) = lambda_nom;
39          wm_ref(i) = wm;
40      elseif time(i) >= t_3_wm && time(i) < t_4_wm
41          lambda_dr_ref(i) = lambda_nom;
42          wm_ref(i) = wm - wm_slope_des*(time(i)-t_3_wm);
43      elseif time(i) >= t_4_wm
44          lambda_dr_ref(i) = lambda_nom;
45          wm_ref(i) = 0;
46      end
47  end
48
49  end

1  function ParametersIM = ReturnParametersIM_15hp
2  % 15 hp Motor – Kirtley (Electric power principles: sources, conversion,
3  % distribution and use, Wiley, 2010) p. 293
4  ParametersIM.rs = 0.06; % Stator resistance [ohms]
5  ParametersIM.rr = 0.15; % Rotor resistance [ohms]
6  ParametersIM.Lls = 0.44/(2*pi*60); % Stator leakage inductance [Henry]
7  ParametersIM.Llr = 0.43/(2*pi*60); % Rotor leakage inductance [Henry]
8  ParametersIM.Lm = 12.6/(2*pi*60); % Magnetizing inductance [Henry]
9  ParametersIM.J = 0.445; % Inertia [kg.m^2]
10 ParametersIM.P = 4; % Pole numbers
11 ParametersIM.Vs = 240; % Rated voltage (line-to-line rms) [V]
12 ParametersIM.Ls = ParametersIM.Lls + ParametersIM.Lm;
13 ParametersIM.Lr = ParametersIM.Llr + ParametersIM.Lm;
14 ParametersIM.Lsprime = ParametersIM.Ls - ParametersIM.Lm^2/ParametersIM.Lr;
15 ParametersIM.rsprime = ParametersIM.rs + ParametersIM.rr*ParametersIM.Lm^2/ParametersIM.Lr^2;
16 % ParametersIM.b = 0.00; % Friction factor [kg.m^2/s]
17 ParametersIM.b = 5.41e-4; % Friction factor [kg.m^2/s]
18
19 ParametersIM.Pnom = 15*746; % Nominal motor power [W]
20 ParametersIM.f = 60; % Nominal frequency [Hz]
21 ParametersIM.speed = 1710*pi/30; % Nominal rotor speed [rad/s]
22 ParametersIM.Tnom = ParametersIM.Pnom/(ParametersIM.speed); % Nominal torque [N.m]
23
24 %-----%
25 % ESTIMATED PARAMETERS
26 %-----%
27
28 ParametersIM.rs_est = ParametersIM.rs;
29 ParametersIM.rr_est = ParametersIM.rr;
30 ParametersIM.Lls_est = ParametersIM.Lls;
31 ParametersIM.Llr_est = ParametersIM.Llr;
32 ParametersIM.Lm_est = ParametersIM.Lm;
33 ParametersIM.J_est = ParametersIM.J;
34 ParametersIM.b_est = ParametersIM.b;
35
36 ParametersIM.Ls_est = ParametersIM.Lls_est + ParametersIM.Lm_est;
37 ParametersIM.Lr_est = ParametersIM.Llr_est + ParametersIM.Lm_est;
38 ParametersIM.Lsprime_est = ParametersIM.Ls_est - ParametersIM.Lm_est^2/ParametersIM.Lr_est;
39 ParametersIM.rsprime_est = ParametersIM.rs_est + ParametersIM.rr_est*ParametersIM.Lm_est^2/ParametersIM.Lr_est^2;
40 %-----%
41
42
43 end

1 % Inputs are the qd variables in the stationary reference frame.
2 % Outputs are the b and c variables. In this ref. frame, a=q

```

```

3 function [x_b, x_c] = statbc(x_q,x_d)
4
5 n = numel(x_q);
6 x_q = reshape(x_q,n,1);
7 x_d = reshape(x_d,n,1);
8
9 x_b = x_q.*-0.5 - x_d.*sqrt(3)/2;
10 x_c = x_q.*-0.5 + x_d.*sqrt(3)/2;
11 end

1 % Inputs are the abc phase variables in the stationary reference frame.
2 % Outputs are the qd variables in the stationary reference frame.
3 function [x_q, x_d] = statqd(x_as, x_bs, x_cs)
4
5 % Reshaping the vectors so all of them are column vectors
6 n = numel(x_as);
7 x_as = reshape(x_as,n,1);
8 x_bs = reshape(x_bs,n,1);
9 x_cs = reshape(x_cs,n,1);
10
11 x_q = 2/3*(x_as+x_bs.*-0.5+x_cs.*-0.5);
12 x_d = 2/3*(x_bs.*-sqrt(3)/2+x_cs.*sqrt(3)/2);
13 end

```

B.6 Data Analysis

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % FullPowerAnalysis.m
3 % Analyze data for synchronization methods and generate plots.
4 % Run after loading data from runs of both sync methods.
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 close all
7
8 %% Steady state slip calculation and voltage drop estimation
9 P = ParametersIM.P;
10 we = [diff(R_theta_e) diff(theta_e_out)]/(t(2)-t(1));
11 we = mean(we(500000:end,:));
12 % Average estimate across runs to account for losses
13 wr = mean([R_wm_1(500000:end) wm_1_out(500000:end,:)]*P/2);
14 s = (we-wr)./we;
15 % Calculate reactances
16 Xm = Lm*we;
17 Xss = (Lls+Lm)*we;
18 Xrr = (Llr+Lm)*we;
19 % Proportionality constant
20 K = 3*P/2.*(Xm.^2/we).*rr.*s./((rs*rr+s.*(Xm.^2-Xss.*Xrr)).^2+(rr*Xss+s.*rs.*Xrr).^2);
21 % Steady-state voltage drops, using nominal voltage as base
22 Vas1 = ParametersIM.Vs/sqrt(3);
23 DelV2 = -Vas1+sqrt(Vas1^2+TL2_variation.*ParametersIM.Tnom./K);
24 DelV3 = -Vas1+sqrt(Vas1^2+TL3_variation.*ParametersIM.Tnom./K);
25
26 % Energy measurements are over sync period (excluding startup)
27 %% Compute power and energy measurements for resistor method
28 % Power into main converter
29 [R_ias_1, R_ibs_1, R_ics_1] = statabcMult(R_iqs_1, R_ids_1);
30 [R_ias_2, R_ibs_2, R_ics_2] = statabcMult(R_iqs_2, R_ids_2);
31 [R_ias_3, R_ibs_3, R_ics_3] = statabcMult(R_iqs_3, R_ids_3);
32 R_ias_tot = R_ias_1+R_ias_2+R_ias_3;
33 R_ibs_tot = R_ibs_1+R_ibs_2+R_ibs_3;
34 R_ics_tot = R_ics_1+R_ics_2+R_ics_3;
35 R_idc = R_S1.*R_ias_tot + R_S2.*R_ibs_tot + R_S3.*R_ics_tot;
36 vdc = ParametersIM.Vs*sqrt(2);
37 R_pdc = R_idc.*vdc;
38 R_Edc = trapz(t(200001:end), R_pdc(200001:end));
39
40 % Power output of main converter
41 [R_vasC, R_vbsC, R_vcsC] = statabcMult(R_vqsC, R_vdsC);
42 R_pCout = R_vasC.*R_ias_tot + R_vbsC.*R_ibs_tot + R_vcsC.*R_ics_tot;
43 R_ECout = trapz(t(200001:end), R_pCout(200001:end));
44
45 % Power input to motors
46 [R_vas_1, R_vbs_1, R_vcs_1] = statabcMult(R_vqs_1, R_vds_1);
47 [R_vas_2, R_vbs_2, R_vcs_2] = statabcMult(R_vqs_2, R_vds_2);
48 [R_vas_3, R_vbs_3, R_vcs_3] = statabcMult(R_vqs_3, R_vds_3);
49 R_pmot = (R_vas_1.*R_ias_1) + (R_vas_2.*R_ias_2) + (R_vas_3.*R_ias_3) + ...
50         (R_vbs_1.*R_ibs_1) + (R_vbs_2.*R_ibs_2) + (R_vbs_3.*R_ibs_3) + ...
51         (R_vcs_1.*R_ics_1) + (R_vcs_2.*R_ics_2) + (R_vcs_3.*R_ics_3);
52 R_Emot = trapz(t(200001:end), R_pmot(200001:end));

```

```

53
54 % Electrical efficiency
55 R_effE = R_Emot/R_Edc;
56
57 % Power lost in resistors (neglecting switching losses)
58 R_ploss2 = ((R_ias_2).^2 + (R_ibs_2).^2 + (R_ics_2).^2) .* ...
59     rext2_pwm;
60 R_ploss3 = ((R_ias_3).^2 + (R_ibs_3).^2 + (R_ics_3).^2) .* ...
61     rext3_pwm;
62 R_ploss = R_ploss2 + R_ploss3;
63 R_Eloss = trapz(t(200001:end),R_ploss(200001:end));
64
65 % Power output of motors (mechanical)
66 R_pTm = (R_TL_1.*R_wm_1) + (R_TL_2.*R_wm_2) + (R_TL_3.*R_wm_3);
67 R_ETm = trapz(t(200001:end), R_pTm(200001:end));
68
69 % Motor efficiency
70 R_effMot = R_ETm/R_Emot;
71
72 % Full system electromechanical efficiency
73 R_effTm = R_ETm/R_Edc;
74
75 %% Compute power and energy measurements for aux converter method
76 % Power into main converter
77 [ias_1, ibs_1, ics_1] = statabcMult(iqs_1_out, ids_1_out);
78 [ias_2, ibs_2, ics_2] = statabcMult(iqs_2_out, ids_2_out);
79 [ias_3, ibs_3, ics_3] = statabcMult(iqs_3_out, ids_3_out);
80 ias_tot = ias_1 + ias_2 + ias_3;
81 ibs_tot = ibs_1 + ibs_2 + ibs_3;
82 ics_tot = ics_1 + ics_2 + ics_3;
83 idc = S1_out.*ias_tot + S2_out.*ibs_tot + S3_out.*ics_tot;
84 % vdc = ParametersIM.Vs*sqrt(2);
85 pdc = idc.*vdc;
86 Edc = trapz(t(200001:end), pdc(200001:end,:));
87
88 % Power output of main converter
89 [vaC, vbC, vcC] = statabcMult(vq1_1_out+vqs_1_out, vd1_1_out+vds_1_out);
90 pCout = vaC.*ias_tot + vbC.*ibs_tot + vcC.*ics_tot;
91 ECout = trapz(t(200001:end), pCout(200001:end,:));
92
93 % Power input to motors
94 [vas_1, vbs_1, vcs_1] = statabcMult(vqs_1_out, vds_1_out);
95 [vas_2, vbs_2, vcs_2] = statabcMult(vqs_2_out, vds_2_out);
96 [vas_3, vbs_3, vcs_3] = statabcMult(vqs_3_out, vds_3_out);
97 pmot = (vas_1.*ias_1) + (vas_2.*ias_2) + (vas_3.*ias_3) + ...
98     (vbs_1.*ibs_1) + (vbs_2.*ibs_2) + (vbs_3.*ibs_3) + ...
99     (vcs_1.*ics_1) + (vcs_2.*ics_2) + (vcs_3.*ics_3);
100 Emot = trapz(t(200001:end), pmot(200001:end,:));
101
102 % Electrical efficiency
103 effE = Emot./Edc;
104
105 % Power input to auxiliary converters
106 [ia2_1, ib2_1, ic2_1] = statabcMult(iq2_1_out, id2_1_out);
107 [ia2_2, ib2_2, ic2_2] = statabcMult(iq2_2_out, id2_2_out);
108 [ia2_3, ib2_3, ic2_3] = statabcMult(iq2_3_out, id2_3_out);
109 delv2_idc = zeros(size(ia2_2));
110 delv3_idc = zeros(size(ia2_2));
111 for k=1:numel(Nlist)
112     N = Nlist(k);
113     delv2_idc(:,k) = N*(Sd1_2_out(:,k).*ia2_2(:,k) + ...
114         Sd2_2_out(:,k).*ib2_2(:,k) + Sd3_2_out(:,k).*ic2_2(:,k));
115     delv3_idc(:,k) = N*(Sd1_3_out(:,k).*ia2_3(:,k) + ...
116         Sd2_3_out(:,k).*ib2_3(:,k) + Sd3_3_out(:,k).*ic2_3(:,k));
117 end
118 aux_idc = delv2_idc + delv3_idc;
119 delv_pdc = zeros(size(aux_idc));
120 for k=1:numel(vdcRlist)
121     delv_vdc = vdc * vdcRlist(k);
122     delv_pdc(:,k) = aux_idc(:,k) .* delv_vdc;
123 end
124 delv_Edc = trapz(t(200001:end), delv_pdc(200001:end,:));
125
126 % Power leaving motor lines
127 [val_1, vb1_1, vc1_1] = statabcMult(vq1_1_out, vd1_1_out);
128 [val_2, vb1_2, vc1_2] = statabcMult(vq1_2_out, vd1_2_out);
129 [val_3, vb1_3, vc1_3] = statabcMult(vq1_3_out, vd1_3_out);
130 delv_pmot = (val_1.*ias_1 + vb1_1.*ibs_1 + vc1_1.*ics_1 + ...
131     val_2.*ias_2 + vb1_2.*ibs_2 + vc1_2.*ics_2 + ...
132     val_3.*ias_3 + vb1_3.*ibs_3 + vc1_3.*ics_3);
133 delv_Emot = trapz(t(200001:end), delv_pmot(200001:end,:));
134

```

```

135 % Power output of motors (mechanical)
136 pTm = (TL_1_out.*wm_1_out) + (TL_2_out.*wm_2_out) + (TL_3_out.*wm_3_out);
137 ETm = trapz(t(200001:end), pTm(200001:end,:));
138
139 % Motor efficiency
140 effMot = ETm./Emot;
141
142 % Full system electromechanical efficiency
143 effTm = ETm./Edc;
144
145 %% Decimate readings for display in paper
146 t_ds = downsample(t,200);
147 R_wm_1_ds = downsample(R_wm_1,200);
148 R_wm_2_ds = downsample(R_wm_2,200);
149 R_wm_3_ds = downsample(R_wm_3,200);
150 R_theta_m_1_ds = downsample(R_theta_m_1,200);
151 R_theta_m_2_ds = downsample(R_theta_m_2,200);
152 R_theta_m_3_ds = downsample(R_theta_m_3,200);
153 R_Te_1_ds = downsample(R_Te_1,200);
154 R_Te_2_ds = downsample(R_Te_2,200);
155 R_Te_3_ds = downsample(R_Te_3,200);
156 rext2_cont_ds = downsample(rext2_cont,200);
157 rext3_cont_ds = downsample(rext3_cont,200);
158 wm_1_ds = zeros(numel(t_ds),numel(Nlist));
159 wm_2_ds = wm_1_ds;
160 wm_3_ds = wm_1_ds;
161 theta_m_1_ds = wm_1_ds;
162 theta_m_2_ds = wm_1_ds;
163 theta_m_3_ds = wm_1_ds;
164 Te_1_ds = wm_1_ds;
165 Te_2_ds = wm_1_ds;
166 Te_3_ds = wm_1_ds;
167 delv2_cont_ds = wm_1_ds;
168 delv3_cont_ds = wm_1_ds;
169 for i=1:numel(Nlist)
170     wm_1_ds(:,i) = downsample(wm_1_out(:,i),200);
171     wm_2_ds(:,i) = downsample(wm_2_out(:,i),200);
172     wm_3_ds(:,i) = downsample(wm_3_out(:,i),200);
173     theta_m_1_ds(:,i) = downsample(theta_m_1_out(:,i),200);
174     theta_m_2_ds(:,i) = downsample(theta_m_2_out(:,i),200);
175     theta_m_3_ds(:,i) = downsample(theta_m_3_out(:,i),200);
176     Te_1_ds(:,i) = downsample(Te_1_out(:,i),200);
177     Te_2_ds(:,i) = downsample(Te_2_out(:,i),200);
178     Te_3_ds(:,i) = downsample(Te_3_out(:,i),200);
179     delv2_cont_ds(:,i) = downsample(delv2_cont_out(:,i),200);
180     delv3_cont_ds(:,i) = downsample(delv3_cont_out(:,i),200);
181 end
182
183 % Plot mechanical speeds (full)
184 figure
185 plot(t_ds,wm_1_ds(:,1)*30/pi)
186 hold on
187 plot(t_ds,wm_2_ds(:,1)*30/pi)
188 plot(t_ds,wm_3_ds(:,1)*30/pi)
189 plot(t_ds,R_wm_1_ds*30/pi)
190 plot(t_ds,R_wm_2_ds*30/pi)
191 plot(t_ds,R_wm_3_ds*30/pi)
192 xlabel("$t$ [s]","Interpreter","latex")
193 ylabel("$\omega_{rm}$ [rpm]","Interpreter","latex")
194 legend(["Aux.\ Converter ($N_2/N_1=5$), $\omega_{rm,1}$", ...
195     "Aux.\ Converter ($N_2/N_1=5$), $\omega_{rm,2}$", ...
196     "Aux.\ Converter ($N_2/N_1=5$), $\omega_{rm,3}$", ...
197     "Resistor, $\omega_{rm,1}$", "Resistor, $\omega_{rm,2}$", ...
198     "Resistor, $\omega_{rm,3}$"], "Interpreter", "latex", "Location", "southeast")
199 matlab2tikz('filename','fullSpd.tex','width','6in','height','2in','standalone',true)
200
201 % Plot mechanical speeds (zoomed)
202 figure
203 plot(t_ds,wm_1_ds(:,1)*30/pi)
204 hold on
205 plot(t_ds,wm_2_ds(:,1)*30/pi)
206 plot(t_ds,wm_3_ds(:,1)*30/pi)
207 plot(t_ds,R_wm_1_ds*30/pi)
208 plot(t_ds,R_wm_2_ds*30/pi)
209 plot(t_ds,R_wm_3_ds*30/pi)
210 xlim([1.9 6])
211 xlabel("$t$ [s]","Interpreter","latex")
212 ylabel("$\omega_{rm}$ [rpm]","Interpreter","latex")
213 legend(["Aux.\ Converter ($N_2/N_1=5$), $\omega_{rm,1}$", ...
214     "Aux.\ Converter ($N_2/N_1=5$), $\omega_{rm,2}$", ...
215     "Aux.\ Converter ($N_2/N_1=5$), $\omega_{rm,3}$", ...
216     "Resistor, $\omega_{rm,1}$", "Resistor, $\omega_{rm,2}$", ...

```

```

217     "Resistor,  $\omega_{\text{rm},3}$ "],"Interpreter","latex","Location","southeast")
218 matlab2tikz('filename','zoomSpd.tex','width','6in','height','2in','standalone',true)
219
220 % Plot electromagnetic torques
221 figure
222 plot(t_ds,Te_1_ds(:,1)*30/pi)
223 hold on
224 plot(t_ds,R_Te_1_ds*30/pi)
225 plot(t_ds,Te_2_ds(:,1)*30/pi)
226 plot(t_ds,R_Te_2_ds*30/pi)
227 plot(t_ds,Te_3_ds(:,1)*30/pi)
228 plot(t_ds,R_Te_3_ds*30/pi)
229 xlim([0.5 6])
230 xlabel("$t$ [s]","Interpreter","latex")
231 ylabel("$T_e$ [N\cdot m]","Interpreter","latex")
232 legend(["Aux.\ Converter ($N_2/N_1=5$)", $T_{e,1}$","Resistor, $T_{e,1}$",...
233         "Aux.\ Converter ($N_2/N_1=5$)", $T_{e,2}$","Resistor, $T_{e,2}$",...
234         "Aux.\ Converter ($N_2/N_1=5$)", $T_{e,3}$","Resistor, $T_{e,3}$"],...
235        "Interpreter","latex","Location","southeast","NumColumns",2)
236 matlab2tikz('filename','Te.tex','width','6in','height','2in','standalone',true)
237
238 % Plot position differences on motor 2
239 figure
240 plot(t_ds,(theta_m_2_ds-theta_m_1_ds)*180/pi)
241 hold on
242 plot(t_ds,(R_theta_m_2_ds-R_theta_m_1_ds)*180/pi)
243 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
244         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus","Resistor"],...
245        "Interpreter","latex")
246 xlabel("$t$ [s]","Interpreter","latex")
247 ylabel("$\Delta\theta_{\text{rm},2}$ [deg]","Interpreter","latex")
248 xlim([2 6])
249 % title("Position Synchronization for 20% Torque Differential")
250 matlab2tikz('filename','theta2.tex','width','3in','height','2in','standalone',true)
251
252 % Plot position differences on motor 3
253 figure
254 plot(t_ds,(theta_m_3_ds-theta_m_1_ds)*180/pi)
255 hold on
256 plot(t_ds,(R_theta_m_3_ds-R_theta_m_1_ds)*180/pi)
257 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
258         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus","Resistor"],...
259        "Interpreter","latex")
260 xlabel("$t$ [s]","Interpreter","latex")
261 ylabel("$\Delta\theta_{\text{rm},3}$ [deg]","Interpreter","latex")
262 xlim([2 6])
263 % title("Position Synchronization for 30% Torque Differential")
264 matlab2tikz('filename','theta3.tex','width','3in','height','2in','standalone',true)
265
266 % Plot speed differences on motor 2
267 figure
268 plot(t_ds,(wm_2_ds-wm_1_ds)*30/pi)
269 hold on
270 plot(t_ds,(R_wm_2_ds-R_wm_1_ds)*30/pi)
271 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
272         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus","Resistor"],...
273        "Interpreter","latex")
274 xlabel("$t$ [s]","Interpreter","latex")
275 ylabel("$\Delta\omega_{\text{rm},2}$ [rpm]","Interpreter","latex")
276 xlim([2 6])
277 % title("Speed Synchronization for 20% Torque Differential")
278
279 % Plot speed differences on motor 3
280 figure
281 plot(t_ds,(wm_3_ds-wm_1_ds)*30/pi)
282 hold on
283 plot(t_ds,(R_wm_3_ds-R_wm_1_ds)*30/pi)
284 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
285         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus","Resistor"],...
286        "Interpreter","latex")
287 xlabel("$t$ [s]","Interpreter","latex")
288 ylabel("$\Delta\omega_{\text{rm},3}$ [rpm]","Interpreter","latex")
289 xlim([2 6])
290 % title("Speed Synchronization for 30% Torque Differential")
291
292 % Plot commands for motor 2
293 figure
294 plot(t_ds,delv2_cont_ds/sqrt(2))
295 ylabel("$\Delta V_2^{\prime}$ [V]","Interpreter","latex")
296 ylim([0,20])
297 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
298         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus"],...

```

```

299     "Interpreter","latex","Location","southeast")
300 xlabel("$t$ [s]","Interpreter","latex")
301 xlim([2 6])
302 % title("Commands for 20% Torque Differential")
303 % matlab2tikz('filename','Vcommand2.tex','width','3in','height','2in','standalone',true)
304
305 figure
306 plot(t_ds,rect2_cont_ds)
307 ylabel("$r_{e,2}^* [\Omega]","Interpreter","latex")
308 xlabel("$t$ [s]","Interpreter","latex")
309 xlim([2 6])
310 % matlab2tikz('filename','Rcommand2.tex','width','3in','height','2in','standalone',true)
311
312 % Plot commands for motor 3
313 figure
314 plot(t_ds,delv3_cont_ds/sqrt(2))
315 ylabel("$\Delta V_{3}^{\prime} [V]","Interpreter","latex")
316 ylim([0,35])
317 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
318         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus"],...
319        "Interpreter","latex","Location","southeast")
320 xlabel("$t$ [s]","Interpreter","latex")
321 xlim([2 6])
322 % title("Commands for 30% Torque Differential")
323 % matlab2tikz('filename','Vcommand3.tex','width','3in','height','2in','standalone',true)
324
325 figure
326 plot(t_ds,rect3_cont_ds)
327 ylabel("$r_{e,3}^* [\Omega]","Interpreter","latex")
328 xlabel("$t$ [s]","Interpreter","latex")
329 xlim([2 6])
330 % matlab2tikz('filename','Rcommand3.tex','width','3in','height','2in','standalone',true)
331
332 [delv2_vqe,delv2_vde] = qd2qdMult(delv2_vq_out,delv2_vd_out,theta_e_out);
333 [delv3_vqe,delv3_vde] = qd2qdMult(delv3_vq_out,delv3_vd_out,theta_e_out);
334 v2_norm = sqrt(delv2_vqe.^2+delv2_vde.^2);
335 v3_norm = sqrt(delv3_vqe.^2+delv3_vde.^2);
336 for i = 1:numel(Nlist)
337     v2_norm(:,i) = v2_norm(:,i)*Nlist(i);
338     v3_norm(:,i) = v3_norm(:,i)*Nlist(i);
339 end
340
341 % Generate block averages
342 t_fa = zeros(359,1);
343 rect2_fa = zeros(359,1);
344 rect3_fa = zeros(359,1);
345 delv2_fa = zeros(359,4);
346 delv3_fa = zeros(359,4);
347 R_pdc_fa = zeros(359,1);
348 pdc_fa = zeros(359,4);
349 R_pmot_fa = zeros(359,1);
350 pmot_fa = zeros(359,4);
351 R_ploss_fa = zeros(359,1);
352 delv_pmot_fa = zeros(359,4);
353 R_pTm_fa = zeros(359,1);
354 pTm_fa = zeros(359,4);
355 for i=1:359
356     t_fa(i) = t((i-1)*1667+1);
357     rect2_fa(i) = mean(rect2_pwm((i-1)*1667+1:i*1667+1));
358     rect3_fa(i) = mean(rect3_pwm((i-1)*1667+1:i*1667+1));
359     delv2_fa(i,:) = mean(v2_norm((i-1)*1667+1:i*1667+1,:),1);
360     delv3_fa(i,:) = mean(v3_norm((i-1)*1667+1:i*1667+1,:),1);
361     R_pdc_fa(i) = mean(R_pdc((i-1)*1667+1:i*1667+1));
362     pdc_fa(i,:) = mean(pdc((i-1)*1667+1:i*1667+1,:),1);
363     R_pmot_fa(i) = mean(R_pmot((i-1)*1667+1:i*1667+1));
364     pmot_fa(i,:) = mean(pmot((i-1)*1667+1:i*1667+1,:),1);
365     R_ploss_fa(i) = mean(R_ploss((i-1)*1667+1:i*1667+1));
366     delv_pmot_fa(i,:) = mean(delv_pmot((i-1)*1667+1:i*1667+1,:),1);
367     R_pTm_fa(i) = mean(R_pTm((i-1)*1667+1:i*1667+1));
368     pTm_fa(i,:) = mean(pTm((i-1)*1667+1:i*1667+1,:),1);
369 end
370
371 % Plot actual values for motor 2
372 figure
373 plot(t_fa,delv2_fa/sqrt(2))
374 ylabel("$\overline{\Delta V}_{2}^{\prime} [V]","Interpreter","latex")
375 legend(["Aux.\ Converter ($N_2/N_1=5$)","$N_2/N_1=3$",...
376         "$N_2/N_1=2$, full bus","$N_2/N_1=2$, halved bus"],...
377        "Interpreter","latex","Location","southeast")
378 xlabel("$t$ [s]","Interpreter","latex")
379 xlim([2 6])
380 % title("Fast-Average Voltage/Resistance for 20% Torque Differential")

```

```

381 matlab2tikz('filename','Vactual2.tex','width','3in','height','2in','standalone',true)
382
383 figure
384 plot(t_fa,rect2_fa)
385 ylabel("$\bar{r}_{e,2}$ [$\Omega$]", "Interpreter", "latex")
386 xlabel("$t$ [s]", "Interpreter", "latex")
387 xlim([2 6])
388 matlab2tikz('filename','Ractual2.tex','width','3in','height','2in','standalone',true)
389
390 % Plot actual values for motor 3
391 figure
392 plot(t_fa,delv3_fa/sqrt(2))
393 ylabel("$\overline{\Delta V}_{3\prime}$ [V]", "Interpreter", "latex")
394 legend(["Aux.\ Converter ($N_2/N_1=5$)", "$N_2/N_1=3$", ...
395         "$N_2/N_1=2$, full bus", "$N_2/N_1=2$, halved bus"], ...
396        "Interpreter", "latex", "Location", "southeast")
397 xlabel("$t$ [s]", "Interpreter", "latex")
398 xlim([2 6])
399 % title("Fast-Average Voltage/Resistance for 30% Torque Differential")
400 matlab2tikz('filename','Vactual3.tex','width','3in','height','2in','standalone',true)
401
402 figure
403 plot(t_fa,rect3_fa)
404 ylabel("$\bar{r}_{e,3}$ [$\Omega$]", "Interpreter", "latex")
405 xlabel("$t$ [s]", "Interpreter", "latex")
406 xlim([2 6])
407 matlab2tikz('filename','Ractual3.tex','width','3in','height','2in','standalone',true)
408
409 % Plot main converter input power
410 figure
411 plot(t_fa,pcd_fa/1000)
412 hold on
413 plot(t_fa,R_pcd_fa/1000)
414 ylabel("$\bar{p}_{in}$ [kW]", "Interpreter", "latex")
415 legend(["Aux.\ Converter ($N_2/N_1=5$)", "$N_2/N_1=3$", ...
416         "$N_2/N_1=2$, full bus", "$N_2/N_1=2$, halved bus", "Resistor"], ...
417        "Interpreter", "latex", "Location", "southeast")
418 xlabel("$t$ [s]", "Interpreter", "latex")
419 xlim([2 6])
420 % title("Fast-Average Input Power to Main Converter")
421 matlab2tikz('filename','pin.tex','width','6in','height','2in','standalone',true)
422
423 % Plot motor input power
424 figure
425 plot(t_fa,pmot_fa/1000)
426 hold on
427 plot(t_fa,R_pmot_fa/1000)
428 ylabel("$\bar{p}_{mot}$ [kW]", "Interpreter", "latex")
429 legend(["Aux.\ Converter ($N_2/N_1=5$)", "$N_2/N_1=3$", ...
430         "$N_2/N_1=2$, full bus", "$N_2/N_1=2$, halved bus", "Resistor"], ...
431        "Interpreter", "latex", "Location", "southeast")
432 xlabel("$t$ [s]", "Interpreter", "latex")
433 xlim([2 6])
434 % title("Fast-Average Power Flow to Motors")
435 matlab2tikz('filename','pmot.tex','width','6in','height','2in','standalone',true)
436
437 % Plot power removed from line (loss)
438 figure
439 plot(t_fa,delv_pmot_fa/1000)
440 hold on
441 plot(t_fa,R_ploss_fa/1000)
442 ylabel("$\bar{p}_{loss}$ [kW]", "Interpreter", "latex")
443 legend(["Aux.\ Converter ($N_2/N_1=5$)", "$N_2/N_1=3$", ...
444         "$N_2/N_1=2$, full bus", "$N_2/N_1=2$, halved bus", "Resistor"], ...
445        "Interpreter", "latex", "Location", "southeast")
446 xlabel("$t$ [s]", "Interpreter", "latex")
447 xlim([2 6])
448 % title("Fast-Average Power Removed from Lines")
449 matlab2tikz('filename','ploss.tex','width','6in','height','2in','standalone',true)
450
451 % Plot mech. torque power
452 figure
453 plot(t_fa,pTm_fa/1000)
454 hold on
455 plot(t_fa,R_pTm_fa/1000)
456 ylabel("$\bar{p}_{Tm}$ [kW]", "Interpreter", "latex")
457 legend(["Aux.\ Converter ($N_2/N_1=5$)", "$N_2/N_1=3$", ...
458         "$N_2/N_1=2$, full bus", "$N_2/N_1=2$, halved bus", "Resistor"], ...
459        "Interpreter", "latex", "Location", "southeast")
460 xlabel("$t$ [s]", "Interpreter", "latex")
461 xlim([2 6])
462 % title("Fast-Average Mechanical Torque Power")

```

```
463 matlab2tikz('filename','ptm.tex','width','6in','height','2in','standalone',true)
464
465 %% Functions for handling multiple runs at once
466 function [x_a, x_b, x_c] = statabcMult(x_q, x_d)
467
468 x_a = x_q;
469 x_b = x_q.*-0.5 - x_d.*sqrt(3)/2;
470 x_c = x_q.*-0.5 + x_d.*sqrt(3)/2;
471 end
472
473 function [y_q, y_d] = qd2qdMult(x_q, x_d, theta)
474
475 y_q = x_q.*cos(theta) - x_d.*sin(theta);
476 y_d = x_q.*sin(theta) + x_d.*cos(theta);
477 end
```