DOTTORATO DI RICERCA IN

AUTOMOTIVE PER UNA MOBILITÀ INTELLIGENTE

Ciclo 34

Settore Concorsuale: 09/E2 - INGEGNERIA DELL'ENERGIA ELETTRICA

Settore Scientifico Disciplinare: ING-IND/32 - CONVERTITORI, MACCHINE E AZIONAMENTI ELETTRICI

UNIFIED CONTROL SYSTEM FOR THREE-PHASE ELECTRIC DRIVES OPERATING IN MAGNETIC SATURATION REGION

Presentata da: Marco Bertoldi

Coordinatore Dottorato

Nicolò Cavina

Supervisore

Claudio Rossi

Co-supervisore

Mattia Ricco

Esame finale anno 2022

It doesn't matter how beautiful your theory is,
it doesn't matter how smart you are.
If it doesn't agree with experiment,
it's wrong.

R. P. Feynman

ABSTRACT

The research project aims to study and develop control techniques for a generalized three-phase and multi-phase electric drive able to efficiently manage most of the drive types available for traction application.

The generalized approach is expanded to both linear and non-linear machines in magnetic saturation region starting from experimental flux characterization and applying the general inductance definition. The algorithm is able to manage fragmented drives powered from different batteries or energy sources and will be able to ensure operability even in case of faults in parts of the system.

The algorithm was tested using model-in-the-loop in software environment and then applied on experimental test benches with collaboration of an external company.

KEYWORDS: Electric Machine, Nonlinear, Magnetic Saturation, Real-Time, FOC, Flux Weakening

CONTENTS

ABSTRACT v

TABLE OF CONTENTS vii

LIST OF FIGURES xi

1 INTRODUCTION 1

2 ALGORITHM FRAMEWORK 7

2.1 Reference Frames 7

2.2 Naming Convention 10

2.3 General Function Description 15

2.3.1 PWM Task 16

2.3.2 FOC Task 17

2.3.3 APP Task 18

2.4 Timing 19

3 UNIFIED ELECTRIC MACHINE MODEL 25

3.1 Machine Equations 26

3.1.1 Linear Machine 27

3.1.2 Magnetic Saturation 30

3.1.3 Electric Machine Magnetic Saturation Literature 31

3.1.4 Nonlinear Machine 33

3.2 Discrete Linear Model 37

3.2.1 Optimization and Sub-Interval Predictive Integration 38

3.3 Discrete Nonlinear Model 42

3.3.1 Synchronous Machine 43

3.3.2 Induction Machine 47

4 FIELD ORIENTED CONTROL 51

4.0.1 FOC Magnetic Saturation Literature 51

4.1 Operating Points 53

4.2 Linear Implementation 56

4.2.1 Flux Reference Frame and Transformations 57

4.2.2 Linear Operating Points Calculation 59

4.2.3 Flux Weakening Strategy 62

4.2.4 Variation Using Torque Derivative 68

4.3 Nonlinear Implementation 70

4.3.1 Maximum Torque per Ampere 71

4.3.2 Flux Weakening 76

4.3.3 Torque Control 82

4.4 Nonlinear Variant Implementation 82
4.4.1 Variant Flux Weakening 83
5 ALGORITHM DESCRIPTION 89
5.1 General Model Architecture 90
5.1.1 Plant 91
5.1.2 Scheduler 93
5.1.3 ASW 95
5.1.4 Function 97
5.1.5 Atomic Function Inclusion 100
5.1.6 Parameters and Calibration Values 102
5.2 MPA - Machine Parameters 103
5.3 COP PWM Coefficients 105
5.4 COF FOC Coefficients 106
5.5 IPA Inverter Parameters 107
5.6 COI Inverter Constants 108
5.7 THM Thermal Model 108
5.8 STC Switching Time Control 108
5.9 CMX Current Max 110
5.10 SVE Stator Voltage Estimator 110
5.11 ASE Angle Speed Estimation 111
5.12 CRK Clarke Transform 112
5.13 VUT Unit Vector T 113
5.14 EMS Electric Model Solver 114
5.14.1 SM implementation 115
5.14.2 IM implementation 116
5.14.3 Current error correction 118
5.15 CCR Current Correction 119
5.16 TQE Torque Estimation 119
5.17 ACR Alpha Correction 120
5.18 EMF ElectroMotive Force 121
5.19 VRG Voltage Regulator 122
5.20 ICK Inverse Clarke Transform 122
5.21 DTI Dead Time Introduction 124
5.22 SVM Space Vector Modulation 126
5.23 VSM Vs Max 128
5.24 VRS Voltage Regulator Saturation 130
5.25 SPD Set PWM Duty 131
5.26 SUP State Machine Start Up 132
5.27 FOA Field Oriented Angle 132
5.28 FOT Field Oriented Transform 133
5.29 FOS Field Oriented Speed 134
5.30 FOM Field Oriented Measures 134
5.31 CSD Current Setpoint Determination 135

5.32 FWC Flux Weakening 137
5.32.1 FWC Variant 139
5.33 IQM iQ Max 141
5.34 TMX Torque MaxMin 142
5.35 IQR IQ Reference 142
6 RESULTS 145
6.1 Discrete Model Optimization 145
6.1.1 Testing Environment 149
6.1.2 Model execution speed 150
6.1.3 Model precision 152
6.2 Linear Machine Operation 157
6.3 Flux and Inductance Characterization 164
6.4 Nonlinear Control and Flux Weakening 173
7 CONCLUSIONS 179

A PARAMETERS CONFIGURATION SCRIPT EXAMPLE 181
B REGULATOR CALIBRATION 185
B.1 Digital Regulator Implementation 185
B.2 Closed Loop System Regulators 188
B.3 General Consideration on the Calibration 194
B.4 Calibration from Load Parameters 197

ACKNOWLEDGMENTS 207
BIBLIOGRAPHY 209

LIST OF FIGURES

Figure 1.1 Example of conventional rotor types. 2

Figure 1.2 Overall general scheme of the proposed control algorithm. 4

Figure 2.1 General three-phase electric machine with *uvw* phases and rotor angle. 7

Figure 2.2 Summary and naming of the three notable reference frames used. 9

Figure 2.3 Variable and quantities short-hand naming convention. 9

Figure 2.4 Example of variable name showing the different parts. 11

Figure 2.5 Control algorithm scheme showing the task division and main outputs. 15

Figure 2.6 Schematic of the PWM task single atomic functions with execution order (from left to right) and main outputs. Dashed atomic blocks may not be necessary. 17

Figure 2.7 Schematic of the FOC task single atomic functions with execution order (from left to right) and main outputs. Dashed atomic blocks may not be necessary. 18

Figure 2.8 Schematic of possible APP task single atomic functions with execution order (from left to right) and main outputs. Dashed atomic blocks may not be necessary. 19

Figure 2.9 Scheme of the timing implementation showing the execution order around a given step k. 21

Figure 2.10 Schematic representation of a double buffer information exchange between different priority tasks. 23

Figure 3.1 Magnetization curve of common core steels, highlighting the linear and saturated region. 31

Figure 3.2 Graphic representation of equivalent and derivative inductances on a simple magnetic characteristic. 35

Figure 3.3 Example of saturated flux characteristic for IPMSM dependant on both currents. 35
Figure 3.4 Discrete time interval sub-division diagram. 39
Figure 3.5 Schematic overview of the linear Unified Model algorithm implementation. 41
Figure 3.6 Example of polynomial flux characteristic interpolation of a Synchronous Machine. 43
Figure 3.7 Schematic overview of the nonlinear integration algorithm. 46
Figure 3.8 Example of saturated flux characteristic for IM. 49
Figure 4.1 Overview of the limits and operating points for various types of linear electric machine. 53
Figure 4.2 Example of operating point difference on a IPM-SM with and without magnetic saturation. 56
Figure 4.3 Linear current and voltage limits in the current and flux reference frames. 59
Figure 4.4 Current and voltage limits, MTPA and MTPV operating point curves in the flux and current reference frames for linear machine. 62
Figure 4.5 Comparison between linear and non-linear machines operating point curves in the two reference frames. 63
Figure 4.6 Maximum Torque per Flux point at given speed. 65
Figure 4.7 Plot of the example input operating point on the ISOTcurve. 65
Figure 4.8 Maximum q axis flux at the input condition with maximum ISOTcurve. 66
Figure 4.9 Line construction of the torque approximation. 66
Figure 4.10 Translation of torque approximation to input point $\varphi_{d_{IN}}$. 66
Figure 4.11 Intersection and operating point output of the function. 68
Figure 4.12 Value of q flux that could deliver maximum torque in the new operating point. 68
Figure 4.13 Detail of the final operating point output with the geometric construction. 69
Figure 4.14 Result using derivative calculation compared to previous case. 70

Figure 4.15 Geometry construction of the MTPA point update with comparison with the old method. 74
Figure 4.16 Geometry construction of the emf limit from the applied voltages. 85
Figure 5.1 Higher level of algorithm implementation with PLANT system in feedback to the application software ASW. 91
Figure 5.2 Example of PLANT subsystem with highlighted various important signal generators for electric drive's control. 92
Figure 5.3 Schematic overview of the execution scheduling system used in the application. 94
Figure 5.4 Example of state flow event calling of the application for an input subsystem. 95
Figure 5.5 A look inside the ASW block and its division in input/function/outputs parts. 96
Figure 5.6 Example of *Input* block functions. 96
Figure 5.7 Example of *Outputs* block functions. 97
Figure 5.8 Example of *Function* block implementation in Simulink. 98
Figure 5.9 Schematic of the *Function* block architecture. 99
Figure 5.10 Practical data integrity signal routing between tasks at different priority. 100
Figure 5.11 Setup parameter masks for the rate transition and memory blocks for data integrity. 101
Figure 5.12 Atomic function first integration level: function sub-system. 101
Figure 5.13 Atomic function second integration level: bus input selection and output creation. 102
Figure 5.14 Atomic function third integration level: function-call. 102
Figure 5.15 Atomic function fourth integration level: calculations. 103
Figure 5.16 MPA function, overview. 104
Figure 5.17 MPA function, detail of nonlinear parameters calculation for SM and IM. 104
Figure 5.18 COP function, overview. 106
Figure 5.19 COF function, overview. 107
Figure 5.20 IPA function, overview. 108
Figure 5.21 COI function, overview. 109
Figure 5.22 STC function, overview. 109
Figure 5.23 CMX function, overview. 110

Figure 5.24 SVE function, overview. 111
Figure 5.25 ASE function, overview. 112
Figure 5.26 VUT function, overview. 113
Figure 5.27 EMS function, overview with highlighted the different parts. 114
Figure 5.28 EMS function, SM model. 115
Figure 5.29 EMS function, SM model integrator loop. 116
Figure 5.30 EMS function, IM model. 117
Figure 5.31 EMS function, IM model integrator loop. 117
Figure 5.32 EMS function, current error correction. 118
Figure 5.33 TQE function, overview. 120
Figure 5.34 ACR function, overview. 121
Figure 5.35 EMF function, overview. 122
Figure 5.36 VRG function, overview. 123
Figure 5.37 ICK function, overview. 124
Figure 5.38 Explanation of the dead time delay to be introduced to avoid short circuits in the inverter legs. 124
Figure 5.39 Example of the distortions created by the dead times. 125
Figure 5.40 Dead time compensation diagram showing where and how it is applied in the algorithm. 126
Figure 5.41 DTI function, overview. 126
Figure 5.42 SVM function, PWM modulation schematic. 127
Figure 5.43 SVM function, overview. 127
Figure 5.44 SVM function, visualization of preferential saturation of the voltage based on the rotating axes. 129
Figure 5.45 VSM function, overview. 129
Figure 5.46 Regulator saturation implementation showing the overall signals flow. 130
Figure 5.47 VRS function, overview. 131
Figure 5.48 SPD function, overview. 131
Figure 5.49 SUP function, overview. 132
Figure 5.50 FOA function, overview. 133
Figure 5.51 FOT function, overview. 134
Figure 5.52 FOS function, overview. 134
Figure 5.53 FOM function, overview. 135
Figure 5.54 CSD function, overview. 135
Figure 5.55 CSD function, overview. 136
Figure 5.56 FWC function, implementation overview. 137

Figure 5.57 FWC function, schematic of the most important calculations and signals. 137
Figure 5.58 FWC function, calculation of the maximum stator flux φ_{SM}. 138
Figure 5.59 FWC variant function, implementation overview. 140
Figure 5.60 FWC variant function, schematic of the most important calculations and signals. 140
Figure 5.61 FWC variant function, calculation of the maximum stator flux φ_{SM} and electromotive force. 141
Figure 5.62 IQM function, overview. 142
Figure 5.63 IQR function, overview. 143
Figure 6.1 Continuous time Simulink model: CNT. 146
Figure 6.2 Discrete time model using built in integrator: SIMULINK-DSCR. 147
Figure 6.3 Discrete time model using the algorithm under study without optimization: NAT-DSCR. 147
Figure 6.4 Discrete time model with fast matrix inversion: INV-DSCR. 148
Figure 6.5 Discrete time model with rotor reference frame integration: ROT-DSCR. 148
Figure 6.6 Discrete time model with fast rotation implementation: FAST-DSCR. 149
Figure 6.7 Execution time comparison increasing the discrete sub-interval integration number. CNT and SIMULINK-DSCR do not depend on the number of sub-intervals. Time is normalized to the CNT execution time. 152
Figure 6.8 Simulink flux machine model comparison example between the model presented: CNT, SIMULINK-DSCR, NAT-DSCR, INV-DSCR, ROT-DSCR and FAST-DSCR. 6.8a overview, 6.8b zoom to highlight differences. 153
Figure 6.9 Percentage difference between CNT and FAST-DSCR at low frequency increasing the integration sub-intervals for flux estimation in (from top to bottom) stator d-axis, stator q-axis, rotor d-axis, rotor q-axis. Stator field and rotor mechanical rotational speed equal to $6\frac{rad}{s}$. 154

Figure 6.10 Percentage difference between CNT and FAST-DSCR at high frequency increasing the integration sub-intervals for flux estimation in (from top to bottom) stator d-axis, stator q-axis, rotor d-axis, rotor q-axis. Stator field and rotor mechanical rotational speed have different values, 6200 and 5700$\frac{\mathrm{rad}}{\mathrm{s}}$ respectively. 155
Figure 6.11 Torque and speed input ramps as used for the tests in this section. 157
Figure 6.12 Operating point performance test results for EM1 (IPM-SM), shown in the current (a) and flux (b) reference frames. 159
Figure 6.13 Operating point performance test results for EM2 (SPM-SM), shown in the current (a) and flux (b) reference frames. 160
Figure 6.14 Operating point performance test results for EM3 (R-SM), shown in the current (a) and flux (b) reference frames. 161
Figure 6.15 Operating point performance test results for EM4 (WR-SM), shown in the current (a) and flux (b) reference frames. 162
Figure 6.16 Operating point performance test results for EM5 (IM), shown in the current (a) and flux (b) reference frames. 163
Figure 6.17 Flux characteristics from FEM analysis for the IPM-SM under study. Flux of d axis (a) and of q axis (b) presented. 165
Figure 6.18 Calculated derivative inductance maps for the four terms of the [L] matrix characteristic of the IPM-SM under study. 166
Figure 6.19 Calculated torque characteristic based on the flux maps of the IPM-SM under study. 167
Figure 6.20 Calculated torque characteristic contours in both current and flux reference frames for the IPM-SM under study. 167
Figure 6.21 MTPA attribute obtained via brute force in the current reference frame for the IPM-SM under study. 168
Figure 6.22 MTPA and MTPV attributes obtained via brute force in the flux reference frame for the IPM-SM under study. 168

Figure 6.23 Flux characteristics from testbench data for the SPM-SM. Flux of d axis (a) and of q axis (b) presented. 170
Figure 6.24 Calculated derivative inductance maps for the four terms of the [L] matrix characteristic of the SPM-SM. 171
Figure 6.25 Calculated torque characteristic contours in both current and flux reference frames for the SPM-SM. 171
Figure 6.26 MTPA attribute obtained via brute force in the current reference frame for the SPM-SM. 172
Figure 6.27 MTPA and MTPV attributes obtained via brute force in the flux reference frame for the SPM-SM. 172
Figure 6.28 Control algorithm output result in the current reference frame for IPM-SM with test to 1500 Nm at 8000 rpm compared with linear features and experimental optimal operating points for validation. 173
Figure 6.29 Control algorithm output result in the flux reference frame for IPM-SM with test to 1500 Nm at 8000 rpm compared with linear features and experimental optimal operating points for validation. 174
Figure 6.30 Confirmation of results by showing the same test run using a comparable linear IPM-SM in both reference frames. 174
Figure 6.31 Simulation outputs for test evaluating behaviour in the four quadrants and dynamic inputs. The plot above shows the speed and the torque reference plus output. Below the currents of the machine are presented. 176
Figure 6.32 Simulation outputs for test evaluating behaviour in the four quadrants and dynamic inputs. Overview in the current and flux planes of the operating points' trajectory. 176
Figure 6.33 Confirmation of nonlinear control results for SPM-SM in both reference frames during a torque ramp and subsequent speed ramp with comparison with comparable linear machine. 177

Figure 6.34 Confirmation of nonlinear control results for IM in both reference frames during a torque ramp and subsequent speed ramp with comparison with comparable linear operating points. 177
Figure 6.35 Comparison between the FWC variants when introducing an error in the parameters of the control. The *adj* outputs are obtained with the variant function that is less sensible to the parameter variance. 178
Figure B.1 Example of Tustin/trapezoidal approximation. 186
Figure B.2 Visualisation of a saturated regulator behaviour. 187
Figure B.3 Close-loop system scheme. 188
Figure B.4 Root loci in case of $z_{\mathtt{reg}} \leqslant p_{\mathtt{load}}$. 190
Figure B.5 Root loci in case of $z_{\mathtt{reg}} > p_{\mathtt{load}}$. 192
Figure B.6 Example of over-elongated step response for the case $z_{\mathtt{reg}} > p_{\mathtt{load}}$. 193
Figure B.7 Root position scheme for $z_{\mathtt{reg}} > p_{\mathtt{load}}$ and low K_p gain. 195
Figure B.8 Root position scheme for $z_{\mathtt{reg}} > p_{\mathtt{load}}$ and high K_p gain. 195
Figure B.9 Root position scheme for $z_{\mathtt{reg}} > p_{\mathtt{load}}$ and different K_p gains in the band producing oscillations. 196
Figure B.10 Root position scheme for $z_{\mathtt{reg}} < p_{\mathtt{load}}$, stable configuration. 197
Figure B.11 Root position changes for decreasing values of load pole. **(a)** $z_{\mathtt{reg}} > p_{\mathtt{load}}$, stable. **(b)** $z_{\mathtt{reg}} = p_{\mathtt{load}}$, limit condition. **(c)** $z_{\mathtt{reg}} < p_{\mathtt{load}}$, over-elongation. **(d)** $z_{\mathtt{reg}} = p_{\mathtt{load}}$, oscillating response. 198
Figure B.12 Example of flux linkage characteristic. **(a)** d-axis. **(b)** q-axis 199
Figure B.13 Example of interpolated differential inductance functions. **(a)** L_{dd}. **(b)** L_{dq}. **(c)** L_{qd}. **(d)** L_{qq}. 200
Figure B.14 Characteristics example for $\det\left([L]_{\mathtt{sym}}\right)$. 204

1 INTRODUCTION

The advancement in solid-state switches and computational power has revolutionised in the last 30 years all the applications of power converters thanks to highly flexible and efficient systems. For traction application, the wide spread of continuous to alternating current conversion devices, called inverters, is the key factor allowing the battery packs to be used as efficiently as possible, thus making the electric vehicles competitive against traditional internal combustion ones and more refined in their performance delivery. The inverter is the device that controls the input supply to an electric machine to produce the desired output of torque and speed.

Inverter technologies are always improving both on the hardware side with introductions of faster switches and different topologies, and on the software side with novel approaches to reduce calculation times and increase the operating range of the drive to improve the performance. The programs running on modern traction inverters are usually part of the whole vehicle software architecture and, as such, include a very extensive function library to communicate with the vehicle, generate diagnosis signals, produce infotainment data, run safely under harsh automotive conditions while also keeping an eye on the commercial side of the problem with cost reduction and flexibility to run on different systems.

The work presented focuses on the core function of an inverter, that is the generation of control signals to the inverter switches to produce the desired voltage at the electric machine's taps. The idea is to improve the existing software available in the Laboratory of Electrical Machines, Drives and Power Electronics (LEMAD) of the University of Bologna by writing it in a more flexible and modular environment and extend its operation for machines in magnetic saturation region.

A control algorithm capable to take under control all conventional three-phase AC-machines shown in Fig. 1.1 is seen as an interesting research topic for the development of unified traction drive control system. The electric machines are commonly referred as:

- IM Induction Machine;

- SM Synchronous Machine:
  - R-SM Reluctance Synchronous Machine;
  - PMAR-SM Permanent Magnet Assisted Reluctance Synchronous Machine;
  - SPM-SM Surface Permanent Magnet Synchronous Machine;
  - IPM-SM Internal Permanet Magnet Synchronous Machine;
  - WR-SM Wound Rotor Synchronous Machine.

The development of simple, reliable, and easy-to-use electric drive model is also of great interest for the realization of multibody software for vehicle dynamic simulation. The basic requirements of an electric drive control system model is the capability to calculate the output torque and the corresponding optimal supply condition (stator currents) complying with both the demanded torque and the supply constrains (current limit and voltage limit). An additional feature that can be very useful is represented by the capability to calculate the maximum torque at the actual operating point under the given supply constraints. This calculation is related to steady state condition, but is then applied to the full drive system including dynamic regulation of current and flux.

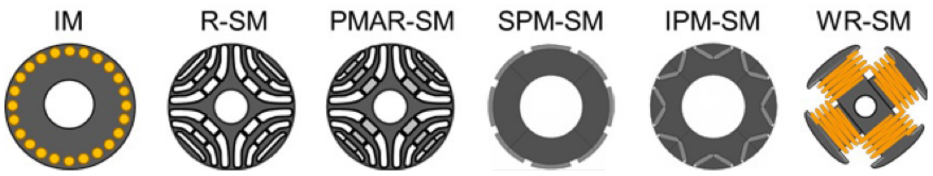


Figure 1.1: *Example of conventional rotor types.*

Optimal operating point calculation requires the correct flux estimation and orientation of the d-q reference frame. For this purpose, several solutions of unified control algorithm have been proposed in the past. More recent approaches were based on control algorithms operating in stator coordinates. These controls allow direct control of the stator flux through the direct stator voltage components and torque control through the quadrature stator current component. Stator and rotor flux are estimated by using the current model, but, unfortunately, Variable Structure Control (VSC) is required for adapting the model to different machine types. Further approaches show that the methods used for extracting the rotor flux position in a synchronous machine can be extended to IM and vice-versa.

An alternative approach is based on the exploitation of the active flux concept, which refers to a torque-producing flux in the electromagnetic torque formulas of AC machines. By using this concept, the active flux is aligned with the rotor pole in all the synchronous machines and with the rotor flux in induction machines. In this way, all salient rotor AC machines are virtually turned in non-salient rotor machines. The introduction of active flux concept yields to a more robust flux observer, especially in the case of magnetic saturation, even for saliency rotors, and is a good basis for implementing a sensorless observer of the rotor position. Once the rotor flux position is derived and the d-q reference frame orientation is found, the method for calculating the optimal operating point can be applied.

The collaboration undertaken during the doctorate with the FEV Gmbh company allowed the deployment of the algorithm on an high performance electric drive. The testbench validations and the experimental results showed how the linear machine analysis was not sufficient to extract all the performance of the motor. The subsequent development of a high performance algorithm was undertaken together with the company to produce a result optimized for DSP deployment and nonlinear efficiency.

A big focus of the development started even before writing any type fo code by creating a structure and methodology of the output. By defining the general architecture, the functionalities of the single software components and the naming convention of variables and parameters, everyone working on the algorithm is able to understand the general flow of information and the objective of the function. Additional documentation and comments inside the program was also an important consideration to make the result user-friendly and easier to calibrate and debug.

Pushed by the industry request of higher performance machines and applications, the usual machine equations that do not consider the magnetic saturation of the materials are no longer sufficient to meet the efficiency and output requirements. It was therefore necessary to account for the machine nonlinearities and generate a control that is based on the actual flux maps, ideally found experimentally, and introduces a flux control able to conjugate speed and accuracy to solve the nonlinear differential equations in a way that can be implemented on a standard DSP. The approach can also be used in multi-phase machine by increasing accordingly the d-q reference frames with the added benefit that having more degrees of freedom allows more options to increase performance and reliability.

The Unified Control Algorithm is based on the magnetic machine model that generates the machine status from the voltages applied and the current measures. Fig. 1.2 shows the broad software subdivision implementing the core control algorithm with the most important inputs and outputs. All the parts will be described in detail, but it is useful to know how the algorithm includes three main function blocks (EMS, FOC and PWM) plus a *high level control* that is not the main focus of this work.

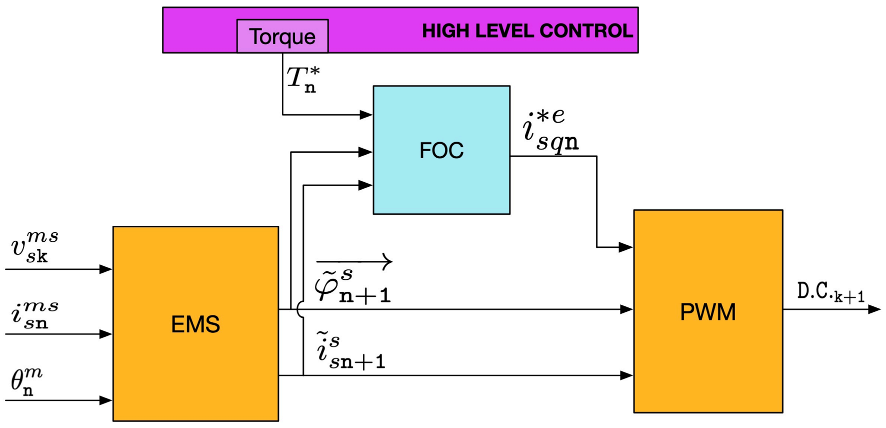


Figure 1.2: *Overall general scheme of the proposed control algorithm.*

The part of the software that implements the model of the electric motor, also known as flux estimator or EMS (Electric Model Solver), integrates the general machine equations and it has been updated for nonlinear machines. Both implementations will be explained because the simpler linear one could still be very useful and more efficient in applications where the requested performance is inside the linear region of operation.

The machine state is then used inside a group of functions collectively called Field Oriented Control or FOC that has the objective of creating the current references to satisfy the application request. In very broad terms, this part of the program takes the torque request from the driver and calculates the best operating point in the current d-q reference frame that maximizes the desired performance while respecting the limitations given by the supply voltage and maximum current available. Once again both linear and nonlinear functions will be described to cover all cases of application.

Having the machine state and the current references, the last part of the program, here called PWM (Pulse Width Modulation), is responsible to carefully generate the switching commands with the proper timing to the hardware. In fact, the inverter controls the current in the machine stator by varying the voltage applied and the voltage is changed through a technique called PWM that opens and closes the switches very quickly to average out the desired voltage value. In a fixed time period, the percentage of time that the switch is closed against when it is open is called Duty Cycle and it is the main output of the control algorithm. This part is the same for both linear and nonlinear implementations.

2 ALGORITHM FRAMEWORK

This chapter analyses the general considerations on how the program is articulated and the framework that has been decided at the beginning of the study to be applied to the whole software and description.

2.1 REFERENCE FRAMES

The general electric machine under study is the three-phase one, commonly described in its equivalent two-phase rotating reference frame for simplicity.

Given the phase convention in Fig. 2.1, the phase u is aligned to a winding of the machine and the other v and w phases with the next counterclockwise ones.

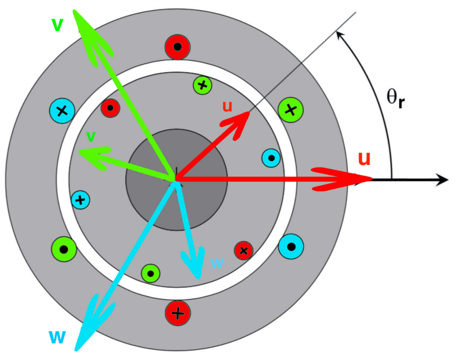


Figure 2.1: *General three-phase electric machine with uvw phases and rotor angle.*

The application of the Clarke transform allows to describe the three-phase system with an equivalent two-phase one, with direction α and β chosen for names. Given an arbitrary three-phase measure $\boldsymbol{x}_{uvw} = (x_u, x_v, x_w)$ its transformation in the equivalent system of reference $\boldsymbol{x}_{\alpha\beta}$ is given by (2.1).

$$\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} x_u \\ x_v \\ x_w \end{bmatrix} \tag{2.1}$$

Of course the inverse transformation is also useful and is operated by equation (2.2).

$$\begin{bmatrix} x_u \\ x_v \\ x_w \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} \tag{2.2}$$

The base idea for rotating electric machines is the injection of sinusoidal currents on a multi-phase arrangement to create a rotating flux that can be described by a single vector. It is therefore useful to find a reference system that rotates together with the significant values. In this way a sinusoidal value can be represented in this new reference frame by an equivalent constant value, simplifying greatly the operations. The Park transformation (2.3) and its inverse (2.4) are the mathematical way to move to and from this rotating reference frame, given the angle $\vartheta(t)$ as the time-variable angle between the two systems. The new rotating axes are given the name d and q.

$$\begin{bmatrix} x_d \\ x_q \end{bmatrix} = \begin{bmatrix} \cos(\vartheta(t)) & \sin(\vartheta(t)) \\ -\sin(\vartheta(t)) & \cos(\vartheta(t)) \end{bmatrix} \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = [T(\vartheta(t))] \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = \begin{bmatrix} \cos(\vartheta(t)) & -\sin(\vartheta(t)) \\ \sin(\vartheta(t)) & \cos(\vartheta(t)) \end{bmatrix} \begin{bmatrix} x_d \\ x_q \end{bmatrix} = [T(\vartheta(t))]^{-1} \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} \tag{2.4}$$

The choice of the angle $\vartheta(t)$ is very important and in this work the concept of active flux is used to generalise the calculation. By definition, the active flux in an electric machine is the main harmonic of the flux generating torque. For Synchronous Machines this is equivalent to say the permanent magnet flux, while for Induction Machines it is the flux generated by the rotor and it is asynchronous with respect to the mechanical rotor angle.

By the definition of active flux, three notable reference frames can be described and illustrated in Fig. 2.2:

- Stator reference frame s, aligned to the stator windings;
- Rotor reference frame r, aligned with the rotor, rotating at the mechanical speed ω_m and creating the angle ϑ_r with the stator reference frame;
- Rotor flux reference frame e, also called *virtual* aligned with the active flux.

Any quantities from different locations can therefore be expressed on multiple reference frames. Moreover, they could be

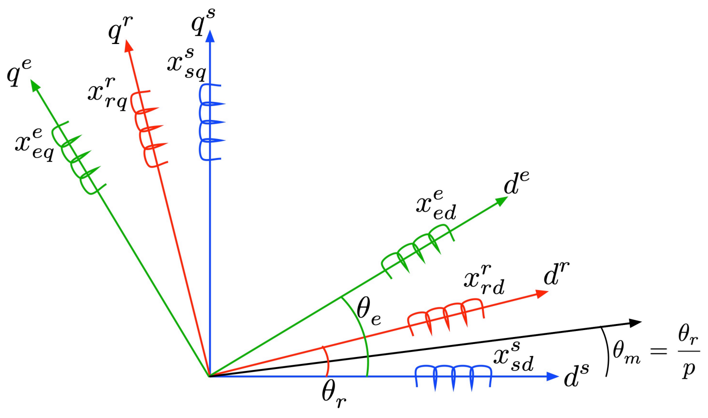


Figure 2.2: *Summary and naming of the three notable reference frames used.*

coming from different sources, like direct measures or model calculations, and even related to different time instants other than the current one. A conventional writing of all this information regarding the single quantities has been agreed and it is schematically presented in Fig. 2.3.

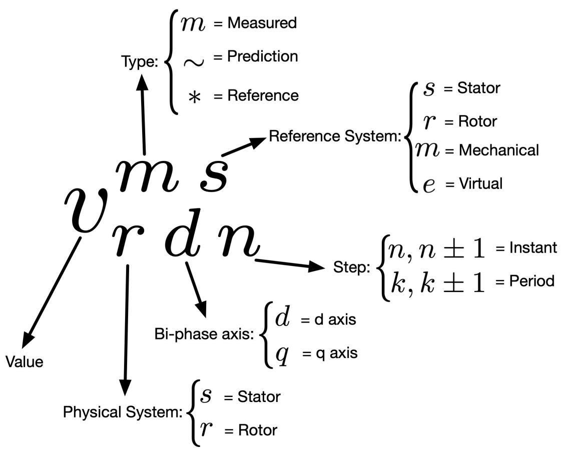


Figure 2.3: *Variable and quantities short-hand naming convention.*

Of course some reasonable simplifications can be assumed in the case of commonly used configurations. For example, when discussing the d-q reference frame it is almost always related to the rotating frame of the active flux, thus some informations about the reference can be omitted. In the same way, if the timing is not important for the arguments described, the time step information is not used and it can be assumed that the quantity is substantially the same at any point.

The most ubiquitous example is the control values for the stator currents which are universally defined in the rotor flux reference frame using the rotating system with the active flux. The proper signage for the measured case would be i_{sdn}^{me} and i_{sqn}^{me}, but it is common practice to accept i_d and i_q as alternatives.

Additionally, sometimes it is useful to introduce the concept of *natural* reference frame as the reference frame in which the quantity can be measured. For example, for rotor quantities the natural reference frame is the rotor one, while for stator quantities is the stator one.

2.2 NAMING CONVENTION

While the short-hand writing of the variables is very useful for documentation and dissemination purposes, the main goal of the work is to produce usable code that is also easily understandable. For this reason, an appropriate way of writing parameters, variables and functions' names must be selected acceptable for inline source coding editors. Moreover, the system should not only describe the quantity identification, but should also contain additional information pertaining to other important factors of its use and quality. In this way anybody looking at just the name is aware not only of what that quantity is, but also any special information related to how to use it.

Table 2.1: *List of names suffixes.*

Suffixes	
Descriptor	Name
_C	Calibration Value
_MP	Measurement Point (observation variable)
_AX,_AY	Axis values
_M	Characteristic curve
no Suffix	Input/output

For example, a variable can be enhanced by saying where it was created and its physical units. The goal is to improve the code readability and create something that is self-explanatory so that more developers can work on and maintain the same project easily.

A comprehensive naming framework was therefore agreed and codified in accordance with some general restrictions of common coding environments. The list of restrictions comprehend:

- limited use of special characters, thus agreeing on a blend *camel* and *snake* writing style;
- use of only letters "a-z", "A-Z", numbers "0-9" and the underscore character "_";
- names cannot start with a number or underscore;
- names can not be the same as reserved function names of common coding languages, especially Matlab or C code;
- maximum length is 63 characters.

The convention is then articulated in five segments, regarding variables and parameters, in this order:

1. function of origin prefix acronym, always 3 letters;
2. physical quantity;
3. physical description;
4. additional information;
5. suffix for the type of data.

Of course for functions and other type of names, the convention is more flexible. In Fig. 2.4 is proposed an example of variable name for the rotor d current showcasing how the different parts are composed. The prefix and suffix are added through underscores, while the rest of the name is in *lower camel*.

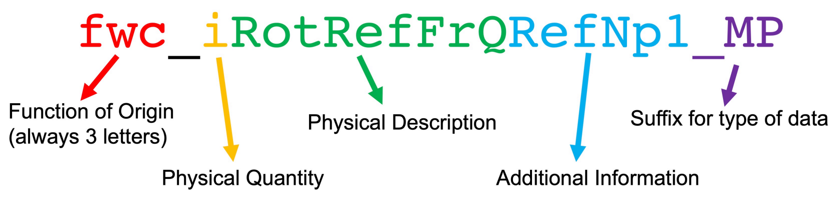


Figure 2.4: *Example of variable name showing the different parts.*

From just the name the user is aware that: this is a measurable variable (*_MP*), it comes from the Flux Weakening Control function (*fwc_*), it is a current (*i*) therefore its unit is Ampere, it is the q component in the rotor reference frame (*RotRefFrQ*) and it is the reference calculated for the next step (*RefNp1*).

Apart from the names of the origin functions that are usually specific for different projects, the other elements have been defined for any type of situation to ensure overall compatibility with the vast array of possibile software solution that can be found in the automotive environment. The following tables present the possible usable names for suffixes Tab. 2.1, physical quantities Tab. 2.3, physical descriptions Tab. 2.2 and for the additional information Tab. 2.4.

In the case of look-up tables the values are registered using the suffixes *_AX* and *_AY* for the breakpoints of the curve on the various axes, while the output measures of the table have the suffix *_M*. For example, for a 1D table of data points, the x-coordinates of the points are named with *_AX* and the y-coordinates with *_M*.

Table 2.2: *List of names possible physical descriptions.*

Physical Description	
Descriptor	Name
Rotr	Rotor
Statr	Stator
Rotrflx	Rotor flux
Wtr	Water
Amb	Ambient
U,V,W	Three phases
D	Rotating direct axis
Q	Rotating quadrature axis
Alpha,Beta	Equivalent two-phases
rotrD, Q	Rotor quantity
statrD, Q	Transformed quantity
statrAlpha, Beta	Clarke Transformed
Em	Electric Machine
Inv	Inverter
Magn	Magnets
Ele	Electrical
Mec	Mechanical
StatrRefFr	Stator reference Frame
RotrRefFr	Rotor reference Frame
RotrflxRefFr	Rotor flux reference Frame

Table 2.3: *List of physical quantities names and acronyms.*

	Physical Quantity	
Descriptor	Name	SI Unit
st	Status	–
flwrt	Flowrate	l/min
u (v)	Voltage	V
i	Current	A
trq	Torque	Nm
n	Rotational Speed	$1/min$
t	Temperature	$^\circ C$
ti	Time	s
htc	Heat Transfer Coef.	W/K
ar	Area	m^2
pwr	Power	W
b	Boolean	–
r	Ratio	–
grd	Gradient	–
p	Pressure	bar
w	Work	j
j	Inertia	$kg \cdot m^2$
agv	Angular velocity	rad/s
aga	Angular acceleration	rad/s^2
cpEl	Capacity	μF or F
diam	Diameter	m
frq	Frequency	Hz
m	Mass	kg
perc	Percentage	%
q	Charge	c (coulomb)
res	Resistance	Ω
v	Velocity	km/h or m/s
ct	Counter	–
idx	Index	–
psi	Flux Linkage	$V \cdot s$
ind	Inductance	H (henry)
no	Number	–
ag	Angle	rad
agd	Angle	° decimal degree
sin	Sine	–
cos	Cosine	–
tan	Tangent	–
pu	Per unit	–

Table 2.4: *List of possible additional information in the names.*

Additional Information	
Descriptor	Name
Min	Minimum
Max	Maximum
Flrc	Failure Reaction
K	Period-averaged quantity of current PWM cycle
Kp1	Period-averaged quantity of next PWM cycle
Km1	Period-averaged quantity of previous PWM cycle
N	Instantaneous quantity of current PWM cycle
Np1	Instantaneous quantity of next PWM cycle
Nm1	Instantaneous quantity of previous PWM cycle
Err	Error/Error Threshold
Fbk	FeedBack quantity
Ffw	FeedForward quantity
Pred	Predicted quantity
Act	Current quantity
Req	Request/target quantity
Ref	Reference quantity
Sat	Saturated quantity

2.3 GENERAL FUNCTION DESCRIPTION

It is practically universal common practice to subdivide the software in smaller functions dedicated to specific tasks that could also be re-used in different parts of the program or in other projects. Moreover, since in the LEMAD a less complex inverter control algorithm was already developed, the overall structure of the control is known with some certainty and, therefore, a function flow chart can be defined at the start. Additionally, a task sub-division into more and less critical layers is something to be researched to increase performance, because it limits the number of calculation to be executed at the faster frequency to the minimum, while moving the unnecessary to slower threads.

In general, the architecture under study makes use of 3 priority tasks that could be themselves sub-divided in additional threads. As reference, the Fig. 2.5 shows the overall flow with functions blocks and colours for the different tasks. The overall final architecture is not very far off the one presented here and proposed as the starting point of what would eventually become the control software. Of course, some modifications, renaming and additions were carried out to better suit the application and increase performance.

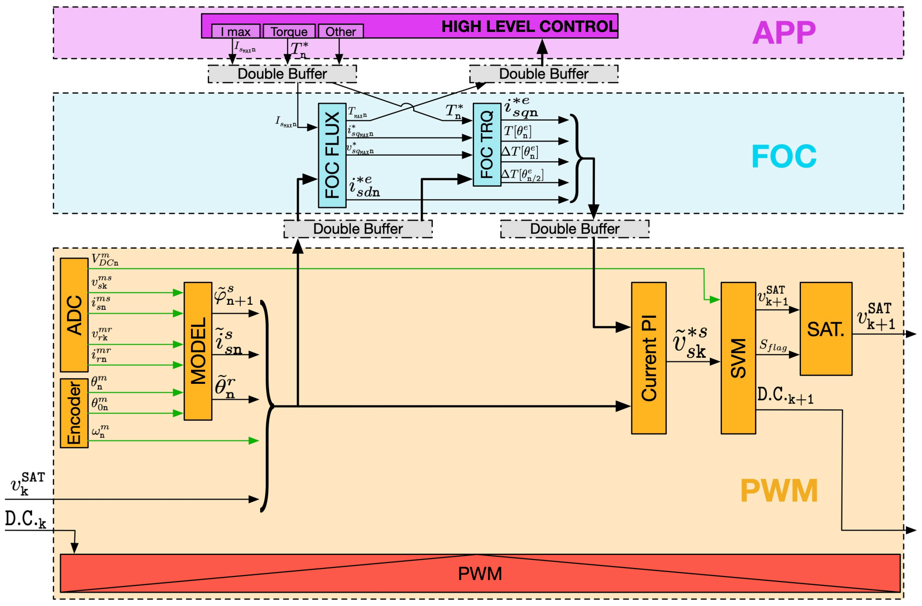


Figure 2.5: *Control algorithm scheme showing the task division and main outputs.*

2.3.1 PWM Task

To control any type of AC machine, the Pulse Width Modulation (PWM) technique is the most common solution and it is characterize by a very regular frequency of the carrier. It is therefore very important to control its output at every cycle. The highest priority task is, consequently, the one synchronous with the PWM cycle, responsible of the acquisition of the control values, such as position and currents, and in charge of defining the reference for the PWM application at every cycle. Since this task lives at the PWM frequency, it is called *PWM task* or *PWM layer* and it is represented by the orange background in Fig. 2.5.

It could be further distinguished between the proper functions in charge of actuating the PWM technique by directly commanding the solid-state switches, in red in the scheme, and the functions calculating the references based on the machine state, in orange. The application of the reference through the PWM is outside the scope of this work, mainly because it is dependent on the hardware used and the presented algorithm wants to be as general as possible.

As shown in the scheme, there are 6 main function blocks that compose the PWM task.

- *ADC* is the collection of functions that acquire current and voltage measures, including the DC bus;
- *Encoder* is the block that produces the position and speed of the machine;
- *Model* summarizes the integration of the machine equations to obtain and predict the flux and current state;
- *Current PI* are the Proportional-Integral regulators that control the current setpoint actuation;
- *SVM* collects the functions for the calculation of the Duty Cycles to operate the Space Vector Modulation;
- *Sat.* applies a saturation feedback to the regulators based on the actual voltages applied.

These general function blocks are then expanded based mainly on experience in smaller functions, each focused on a single job. These atomic functions can be considered as the building blocks of the algorithm and they are designed to be exchanged and modified without impacting the overall software, because each of them has a well defined goal and designed outputs.

The schematics overview of these functions is presented in Fig. 2.6 where they are arranged from left to right for their execution order. In fact, each column can be executed with just the previous leftmost functions completed. Each function is also presented with the main output variables, strictly needed for the next steps. The name abbreviations and their outputs are presented in Tab. 2.5, together with all the others from the different tasks.

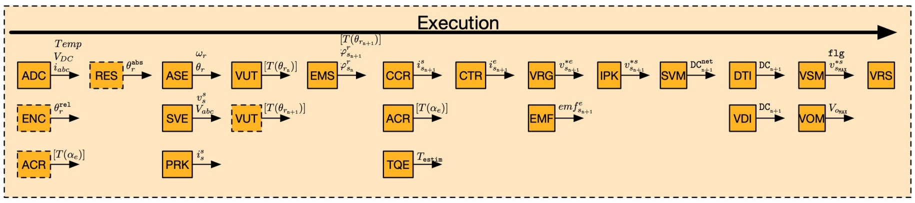


Figure 2.6: *Schematic of the PWM task single atomic functions with execution order (from left to right) and main outputs. Dashed atomic blocks may not be necessary.*

It is important to highlight again that the thread responsible to operate the switches is concurrent with the algorithm under study. Therefore, a time delay of one PWM cycle exists between the reference calculation of the PWM functions and their application by the dedicated thread. This issue will be discussed in the dedicated section 2.4.

2.3.2 FOC Task

As a feature characterizing the implemented control, the calculation of the current references based on the optimal operating point following the field orientation to the active flux can be considered with a lower priority than the PWM cycle. This means that a substantial part of heavy computations can be executed at lower frequency and still maintain correct output delivery. In this elaboration, this task is called *FOC task* or *FOC layer* as it incorporates what is usually called Field Oriented Control because it applies the Park transformation to the active flux reference frame to simplify the implementation. In Fig. 2.5 it is represented by the light blue background.

The functional blocks are just two:

- *FOC FLUX* is in charge of creating the d axis current reference and the limitations;

- *FOC TRQ* calculates the q axis current reference based on the requested torque and the limitations.

Like the PWM task, this one too is subdivided into atomic functions in charge of a single specific goal. In this case the schematic overview is shown in Fig. 2.7, with their execution order from left to right. The full name and output list is again presented in Tab. 2.5.

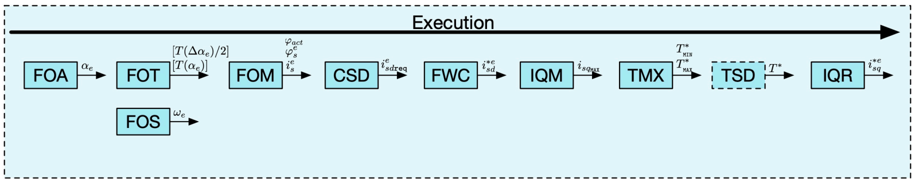


Figure 2.7: *Schematic of the FOC task single atomic functions with execution order (from left to right) and main outputs. Dashed atomic blocks may not be necessary.*

2.3.3 APP Task

This task incorporates all those functions that are not linked to the PWM frequency and that can therefore be executed at a much slower pace or even just when necessary. As the name *APP task* suggests, this layer can also be used by high level applications and controls that generate torque or speed commands, for state machines and for configuration settings.

The possible single functions can be disparate and highly depended on the final application. The implementation under study takes care of just the ones that are most useful for the proper functionality of the algorithm and could almost be considered necessary. As it can be seen in Fig. 2.5 the key feature of this layer is the definition of the torque request, but also very important is the input of the various parameters of the machine and the control.

As a possibile sub-division of the APP task, Fig. 2.8 shows the main atomic functions investigated in the development process described in this work. Some of them, like the thermal model and startup control are not part of the scope of the research and are shown just as a placeholder of important functions that were added during the deployment on real hardware for the tests. The priority and execution order is also more fluid than in the other tasks since there is no stringent timing requirement and, usually, the function are more computationally heavy.

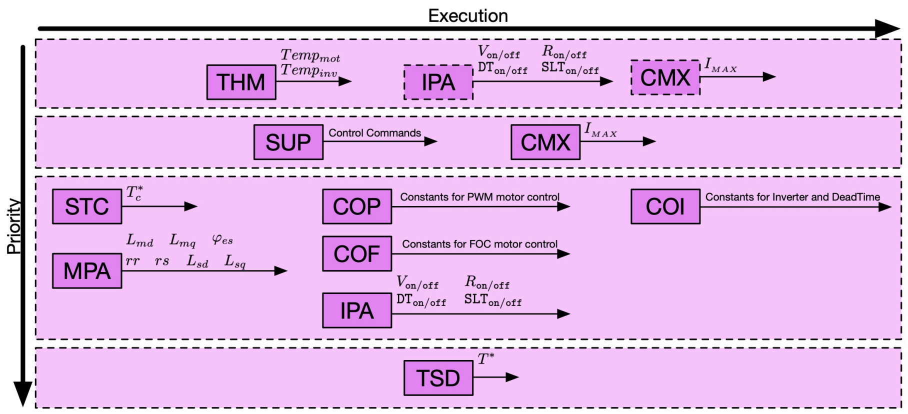

Execution
Priority
THM
$Temp_{mot}$
$Temp_{inv}$
IPA
$V_{\texttt{on/off}}$ $R_{\texttt{on/off}}$
$\text{DT}_{\texttt{on/off}}$ $\text{SLT}_{\texttt{on/off}}$
CMX
I_{MAX}
SUP
Control Commands
CMX
I_{MAX}
STC
T_c^*
COP
Constants for PWM motor control
COI
Constants for Inverter and DeadTime
MPA
L_{md} L_{mq} φ_{es}
rr rs L_{sd} L_{sq}
COF
Constants for FOC motor control
IPA
$V_{\texttt{on/off}}$ $R_{\texttt{on/off}}$
$\text{DT}_{\texttt{on/off}}$ $\text{SLT}_{\texttt{on/off}}$
TSD
T^*

Figure 2.8: *Schematic of possible APP task single atomic functions with execution order (from left to right) and main outputs. Dashed atomic blocks may not be necessary.*

The input parameters for the different subsystems have specific functions and an optimization is run by having dedicated functions that calculate constants that are useful in the faster tasks, thus improving performance by reducing the number of operations to be done at each step.

In the course of the research and development of the full algorithm for nonlinear machines, it was found that it was necessary to re-evaluate the machine parameters at every step time to ensure a satisfactory model performance because they can vary significantly. This meant that the machine parameters function and many coefficient calculations dependent of them had to be moved in the faster PWM task. This will be discussed when introducing the nonlinear control, but the scheme shown is still valid for less complex applications.

2.4 TIMING

Referring to the convention shown in Fig. 2.3, particular attention should be used to define the timing situation of this application, since the execution of the right command in the right instant is crucial for a correct response.

In fact, the correct behaviour of the system is very sensitive, for example, to the angle of the injected current in the active flux reference frame. Because the control is implemented in a discrete environment, the position of the rotor (and consequently of the rotor flux) is known only at certain instants and at high speed this position can change considerably in the step time.

Table 2.5: *Atomic function names, their abbreviation and main outputs.*

Thread	Acronym	Function Name	Output
APP	THM	Thermal Model	t_{mot}, t_{inv}
APP	CMX	Max Current	I_{MAX}
APP	STC	Switching Time Control	t_c
APP	MPA	Machine Parameters	*multiple*
APP	COP	PWM Constants	*multiple*
APP	COF	FOC Constants	*multiple*
APP	IPA	Inverter Parameters	*multiple*
APP	COI	Inverter Constants	*multiple*
APP	TSD	Torque Setpoint Determ.	trq^*
APP	SUP	Start-Up Procedure	*multiple*
FOC	FOA	Field Oriented Angle	α_e
FOC	FOT	Field Oriented Transform	$[T(\alpha_e)]$
FOC	FOS	Field Oriented Speed	ω_e
FOC	FOM	Field Oriented Measures	i_s^e, φ_s^e
FOC	CSD	Current Setpoint Determ.	$i^*_{s\,dREQ}{}^e$
FOC	FWC	Field Weakening Control	i_{sd}^{*e}
FOC	IQM	Iq Max	i^e_{sqMAX}
FOC	TMX	Torque Max	$trq_{MAX\text{-}MIN}$
FOC	IQR	Iq Reference	i_{sq}^{*e}
PWM	ADC	Analog-Digital Converters	i_{uvw}, V_{DC}
PWM	ENC	Encoder	ϑ_r^{rel}
PWM	RES	Resolver	ϑ_r^{abs}
PWM	ASE	Angle/Speed Estimation	ϑ_r, ω_r
PWM	SVE	Stator Voltage Estimation	v_s^s
PWM	CRK	Clarke Transform	i_s^s
PWM	VUT	Unit Vector T	$[T(\vartheta_{rn})]$
PWM	EMS	Electrical Model Solver	φ^r_{sn+1}
PWM	CCR	Current Correction	i^s_{sn+1}
PWM	ACR	Alpha Correction	$[T(\Delta\alpha_e)]$
PWM	TQE	Torque Estimation	trq_{estim}
PWM	CTR	Current Transformation	i^e_{sn+1}
PWM	VRG	Voltage Regulator	v^{*e}_{sn+1}
PWM	EMF	ElectroMotive Force	emf^e_{sn+1}
PWM	ICK	Inverse Clarke Transform	v^{*s}_{sn+1}
PWM	SVM	Space Vector Modulation	$D.C.^{net}_{n+1}$
PWM	DTI	Dead Time Introduction	$D.C._{n+1}$
PWM	VDI	Voltage Drop Inverter	$D.C._{n+1}$
PWM	VSM	Vs Max	v^s_{sMAX}, flg
PWM	VOM	Vo Max	V_{oMAX}
PWM	VRS	Voltage Regulator Satur.	–

A simple single delay in the angle application of the currents can therefore lead to noticeable performance reduction. Similar considerations are also valid for the current measurements and limitation implementations.

Special attention must be considered for the correct timing of the acquisition phase and the actual execution of the voltage commands. The main issue to be solved is the fact that the PWM task must run at the same time as the application of its results during the PWM switching execution. This is of course not possible because it would require all the calculations of the task to be computed in zero time. It follows that the commands are applied in the next PWM cycle, thus creating the delay.

In the same way, certain acquisition techniques make use of average measurements over the whole cycle making their use inside the same timing as problematic.

While for certain slow or low performance application these issues could be considered not important, a general algorithm must be able to minimize such errors. The scheme presented in Fig. 2.9 shows the approach used in the described implementation.

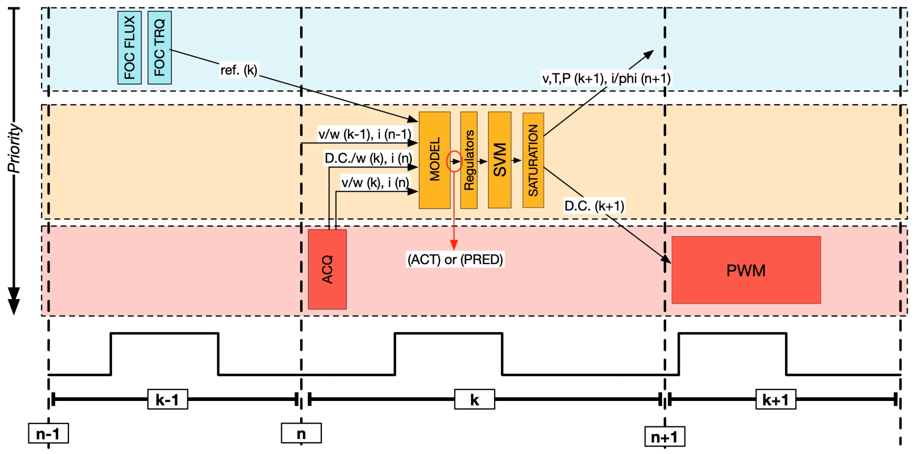


Figure 2.9: *Scheme of the timing implementation showing the execution order around a given step* k.

The first point to clarify is the distinction between the instant n and the period k. Such use has also been hinted in the shorthand naming convention of Fig. 2.3 and the available additional information to add to the code variable naming of Tab. 2.4. The difference relates to the distinction between the starting instant of the calculation step and the whole step period in between successive starting events, thus k is a period of time, while n is an instant.

This is a purely a physical difference that is in place as a reminder of quantities and measurements that can be considered constants during the PWM period, in the case of the k notation, for example voltages and speed, and those that can vary during the elapsed time thus making the value not necessarily true, for example currents and rotor position. From the operational point of view of code writing, the difference is irrelevant since there is nothing under the step size recognizable by the algorithm. From an understanding point of view, this is a good way to highlight how certain measures are made and how to use them with the appropriate approximations.

The implementation of Fig. 2.9 then shows the choices made regarding acquisition and PWM execution.

The measurements and acquisitions for each step k are made at the very beginning of the same step, stealing a little bit of time to the main function, but still ensuring synchronous timing. The other option would be to use the previous cycle $k-1$ maybe averaging or filtering multiple measurement points, but, from experience, this solution does not produce better results. In fact, the most critical measurement is usually the currents and their acquisition around the instant n is in itself a sort of filter since all the switches are usually open and the hardware state in a favorable configuration for measurements.

Regarding the switching execution, there is really no alternative other than doing it the next step and the control timing solution is to implement predictive algorithms capable of calculating the right commands not for the current cycle, but the next one. This prediction is crucial at high speeds and high dynamic response since the rotor position and currents can change a lot in a single PWM cycle.

The issues are concentrated in the PWM task since it is the most demanding and must be executed synchronously. Lower priority layers like FOC and APP are more flexible in their requirements. The only true problem is in making sure the information exchange between layers is deterministic and when the highest priority tasks interrupt the lower ones, no info is compromise by rewriting memory space. Since the correct timing of the operations with the measurements is not a trivial matter, double buffer exchange should be set up to ensure that slower threads won't have the data corrupted by newer updates, and the faster ones can safely deliver the outputs.

Fig. 2.10 shows an example of the use of a double buffer system for exchanging data between two task at different priority,

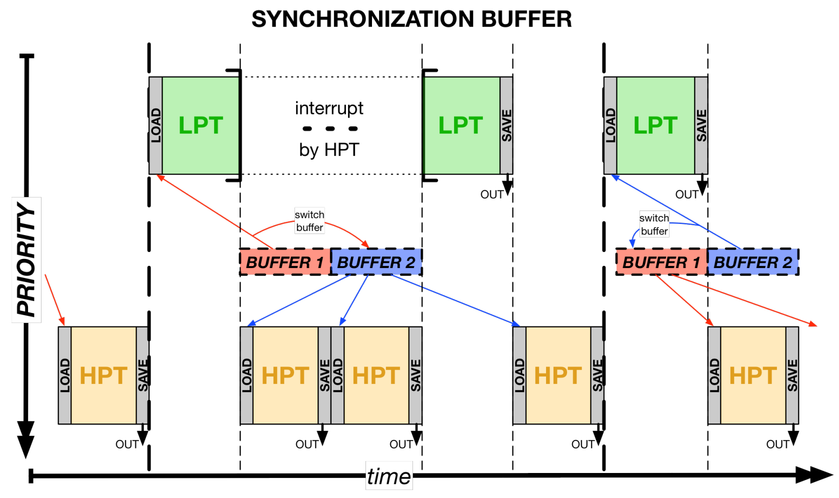


Figure 2.10: *Schematic representation of a double buffer information exchange between different priority tasks.*

meaning that the higher priority (HPT) one can always preempt a lower one (LPT), thus interrupting it. When the low priority task LPT starts, it uses a special instruction supplied by the microchip drivers to quickly and without possible interruption signals that it is using one buffer and everybody else must use the other one. The high priority task HPT can therefore write and re-write the new, released buffer without risk of data compromise, since the LPT is using the other set of data stored safely in the other buffer. When the LPT is finished and restarts, it will use the buffer last accessed by the HPT and signal to use the other one.

3 UNIFIED ELECTRIC MACHINE MODEL

This chapter is dedicated to the Electric Model Solver (EMS) part of the control algorithm. It implements the Unified Electric Machine Model developed in the LEMAD, thus using a generalized approach to find the machine fluxes and currents. The unified model is a formulation of the fundamental flux and machine equations that can describe any type of electric machine just by selecting the right parameters [1,2].

In the Field Oriented Control (FOC) strategy the rotor flux direction (active flux) must be known to operate correctly the d-q axes transformation and decouple the control scheme, thus allowing a more effective control. For Synchronous Machine the mechanical rotor angle position is enough to know the flux direction, while for Induction Machine additional information must be collected, implementing what is called a flux observer. The use of an accurate model capable of describing the status of any machine can be used as a flux observer and to calculate any values linked to the flux, like currents and torque.

The advantages of using this implementation even for Synchronous Machine are numerous and are even more important if dealing with nonlinear machines where the parameters and usual equations may no longer be valid. Some of the features include:

- it works as a flux observer for Induction Machine;
- it allows validation of the current measurements;
- it is ready for sensorless applications;
- it enables torque estimation;
- the magnetic state output can be used by higher level controls for better performance;
- it enables controls less reliant on machine parameters.

Starting from the well known machine equations, execution improvements have been studied and deployed for discrete-time solvers, like the ones on Real-Time DSPs [3,4]. The subdivision of the integration step in smaller segments is a key factor that enables high accuracy while calibrating the processor load to the application. A good approximation of the next step values (prediction) can also be easily implemented [5].

While for the linear formulation the algorithm has been written in the stator reference frame to be used as is for both IM and SM, the development of the nonlinear one required the use of rotating reference frames to make it computationally viable. This resulted in the construction of two slightly different algorithms for the two types of machines, just to keep the solution easy enough for implementation and to allow optimization for maximum performance, since the operation requirements are higher.

3.1 MACHINE EQUATIONS

The common machine model is described by two equations that express the voltage and current behaviour respectively. Since the input of the control algorithm are the voltages, the overall goal of the model is to find first the fluxes and then the currents, thus implementing the inverse equations commonly found in literature. Having found the solutions, together with the torque equation, the machine can be considered completely described in its operation state.

In the context of this work, a linear machine is one that does not present magnetic saturation of its magnetic core, which means that the inductance parameters can be considered constants and not dependent on the currents. On the other hand, a nonlinear machine accounts for the real flux behaviour of being not linearly proportional to the currents. It also implicitly accounts for cross-linkage phenomena between the axes since it is based on experimental data containing this information.

The use of the Unified Model in the EMS function for every type of machine allows the use of the most generic torque equation (3.1) in the rotor flux reference frame for three-phase machines.

$$T = \frac{3}{2} p_p \cdot (\varphi_d i_q - \varphi_q i_d) \tag{3.1}$$

Where p_p is the number of pole pairs of the motor.

This has the distinct advantage of being completely general and not dependant on the machine parameters, thus working even in magnetic saturation region.

3.1.1 Linear Machine

In this section, the derivation of the basic machine equations is done using a generalized notation, but it can be easily expanded or simplified to any type of conventional machine just by adjusting the terms definition or with the appropriate parameters. Given the three-phase machine of Fig. 2.1, the corresponding stator and rotor equations are (3.2).

$$\left.\begin{aligned} v_{sk} &= r_s i_{sk} + \frac{d\varphi_{sk}}{dt} \\ v_{rk} &= r_r i_{rk} + \frac{d\varphi_{rk}}{dt} \end{aligned}\right\} \forall \text{ phase } k = u, v, w \tag{3.2}$$

Which expand, using the Clarke transformations (2.1), into an homopolar component and a complex component, (3.3). In this work the homopolar component will be disregarded, which is an excellent approximation for machines in normal operating conditions.

$$\begin{aligned} \bar{v}_s &= r_s \bar{i}_s + \frac{d\bar{\varphi}_s}{dt} \\ \bar{v}_r &= r_r \bar{i}_r + \frac{d\bar{\varphi}_r}{dt} \end{aligned} \tag{3.3}$$

Similarly, the rotor and stator phase flux equations can be transformed into their complex form, (3.4).

$$\begin{aligned} \bar{\varphi}_s &= L_s \bar{i}_s + \bar{\varphi}_{sT} = L_s \bar{i}_s + L_m \bar{i}_r e^{j\vartheta} \\ \bar{\varphi}_r &= L_r \bar{i}_r + \bar{\varphi}_{rT} = L_r \bar{i}_r + L_m \bar{i}_s e^{-j\vartheta} \end{aligned} \tag{3.4}$$

Where L_m is the coefficient of mutual inductance and L_s and L_r are the stator and rotor auto-inductance respectively.

As said before, (3.3) and (3.4) are the general machine equations in a notation suitable for IM. However, the same description can be used for all the other types of machines with the appropriate consideration regarding the different rotor fluxes. The structure is identical, but the anisotropy introduces asymmetries in the linked fluxes.

It is easier now to express these equations in a matrix form using the d-q frame convention, through the Park transformation (2.3). Therefore, the stator and rotor voltages, for example, are written as a single column vector like in (3.5).

$$\boldsymbol{v}(t) = \begin{bmatrix} v_{sd}^{s} \\ v_{sq}^{s} \\ v_{rd}^{r} \\ v_{rq}^{r} \end{bmatrix} \tag{3.5}$$

As much as possible, the described method for linear machines tries to keep all its algorithm into the *natural* reference frame, which means that every value is given into the reference frame into which it can be measured. This is therefore true when the value has the same letter (s or r) for both the measured position and the reference frame. In this way the

In the same way as (3.5), also currents and fluxes are put in matrix form, (3.6).

$$\boldsymbol{i}(t) = \begin{bmatrix} i^s_{sd} \\ i^s_{sq} \\ i^r_{rd} \\ i^r_{rq} \end{bmatrix} \qquad \boldsymbol{\varphi}(t) = \begin{bmatrix} \varphi^s_{sd} \\ \varphi^s_{sq} \\ \varphi^r_{rd} \\ \varphi^r_{rq} \end{bmatrix} \tag{3.6}$$

Using (3.5) and (3.6), the voltage equation is written in the compact form (3.7).

$$\boldsymbol{v}(t) = [R]\boldsymbol{i}(t) + \dot{\boldsymbol{\varphi}}(t) \tag{3.7}$$

In the same way, introducing $\boldsymbol{\varphi}_e$ as the rotor excitation flux matrix, the flux equation becomes (3.8). Naturally, machines without permanent magnets or excitation windings will present this term as zero.

$$\boldsymbol{\varphi}(t) = [L]\boldsymbol{i}(t) + \boldsymbol{\varphi}_e(t) \tag{3.8}$$

With:

$$\boldsymbol{\varphi}_e(t) = \begin{bmatrix} \varphi^s_{esd} \\ 0 \\ \varphi^r_{erd} \\ 0 \end{bmatrix} \tag{3.9}$$

Other than the excitation flux matrix, any machine is described by the two parameter matrices: [R], or resistance matrix, and [L], or inductance matrix. They are defined as in (3.10) and (3.11).

$$[R] = \begin{bmatrix} r_s & 0 & 0 & 0 \\ 0 & r_s & 0 & 0 \\ 0 & 0 & r_r & 0 \\ 0 & 0 & 0 & r_r \end{bmatrix} \tag{3.10}$$

$$[L] = \begin{bmatrix} L_{sd} & 0 & L^{r\to s}_{md} & 0 \\ 0 & L_{sq} & 0 & L^{r\to s}_{mq} \\ L^{s\to r}_{md} & 0 & L_{rd} & 0 \\ 0 & L^{s\to r}_{mq} & 0 & L_{rq} \end{bmatrix} \tag{3.11}$$

The parameter matrices will have distinct differences depending on the machine types and even possible a priori simplifications based on the interpretation of the equations. For example, the inductance matrix can be constructed to consider the mutual linkage between the stator and rotor windings to be only in one privileged direction, the d axis, thus yielding to defining $L_{md}^{s\to r} \neq 0$ and $L_{mq}^{s\to r} = 0$.

For the correct solution of the equations, all the terms must be referred to the same reference frame, conventionally the rotor one is chosen. It is therefore necessary to introduce a transformation to bring stator quantities into rotor frame and vice versa. A rotation of the rotor angle $\vartheta(t)$ is needed and, since the equations use for elements vectors, a slightly modified Park transformation is used. The matrices that operate the transformation are shown in (3.12). To simplify the writing, time dependency is omitted.

$$
\begin{aligned}
[T(\vartheta(t))] \equiv [T] &= \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
[T(\vartheta(t))]^{-1} \equiv [T]^{-1} &= \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) & 0 & 0 \\ -\sin(\vartheta) & \cos(\vartheta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{3.12}
$$

The (3.7) and (3.8) are brought into a canonical Input-State-Output form with the following choices:

- Input: voltage $\boldsymbol{v}$;
- Output: current $\boldsymbol{i}$;
- State: flux $\boldsymbol{\varphi}$

Using Park (3.12) to keep the measurements in the natural reference frame, but operating the necessary conversion to rotor frame, the complete machine equations become (3.13) and (3.14).

$$\dot{\boldsymbol{\varphi}} = \boldsymbol{v} - [R][T]^{-1}[L]^{-1}[T]\boldsymbol{\varphi} + [R][T]^{-1}[L]^{-1}\boldsymbol{\varphi}_e \tag{3.13}$$

$$\boldsymbol{i} = [T]^{-1}[L]^{-1}[T]\boldsymbol{\varphi} - [T]^{-1}[L]^{-1}\boldsymbol{\varphi}_e \tag{3.14}$$

These equations are completely general for any type of traditional linear electric machine and are expressed in the natural reference frame.

While not used directly in the control algorithm, the stator voltage expression in the rotor flux reference frame is presented here in (3.15) for Synchronous Machine since it can be a useful tool when discussing FOC operations and the main voltage regulators.

$$\begin{cases} v_d = r_s i_d + \frac{d\varphi_d}{dt} - \omega_e \varphi_q = r_s i_d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \\ v_q = r_s i_q + \frac{d\varphi_q}{dt} + \omega_e(\varphi_d + \varphi_e) = r_s i_q + L_q \frac{di_q}{dt} + \omega_e(L_d i_d + \varphi_e) \end{cases} \tag{3.15}$$

Where φ_e is the excitation flux (φ_{PM} in case of permanent magnets) and ω_e is the electrical speed of the machine.

To conclude the description, the torque expression (3.1) can be modified to include the flux definition (3.8). For linear Synchronous Machine this is particularly easy, since the rotor quantities are not present and the inductance parameter matrix include just the direct L_d and quadrature L_q values. The torque equation becomes (3.16), where the expression $\varphi_e = M_{se} i_e$ is the flux generated by the rotor and includes both the case of rotor windings and permanent magnets.

$$T = T_{rq} = \frac{3}{2} p_p \cdot (M_{se} i_e i_q + (L_d - L_q) i_q i_d) \tag{3.16}$$

3.1.2 Magnetic Saturation

The magnetic saturation occurs in ferromagnetic materials when an increase in the applied external magnetic field H does not increase the magnetization of the material further in a linear manner, as a consequence the magnetic flux density B becomes more or less constant.

The increase in H does not contribute to B as in the linear region and indeed the slope of the magnetization curve tends to decrease at high value of the field. Since the field H is directly proportional to the windings current and the flux linkage φ is linked to the current through the inductance parameter, it follows that in the linear region the approximation $\varphi = L \cdot i$ is true, but it is no longer acceptable in the saturation region.

Ferromagnetic materials are composed by domains that act like permanent magnets, they are generally randomly oriented in such a way that their magnetic fields cancel each other out producing a negligible net effect. When an external magnetizing field H is applied to the material, it aligns the domains causing their single magnetic fields to turn parallel to the external one producing a total magnetization effect which extends out from the material.

Once the domains are as aligned as the crystal structure allows them to be, there is no structural change that can be made to further increase B and therefore the magnetization remains nearly

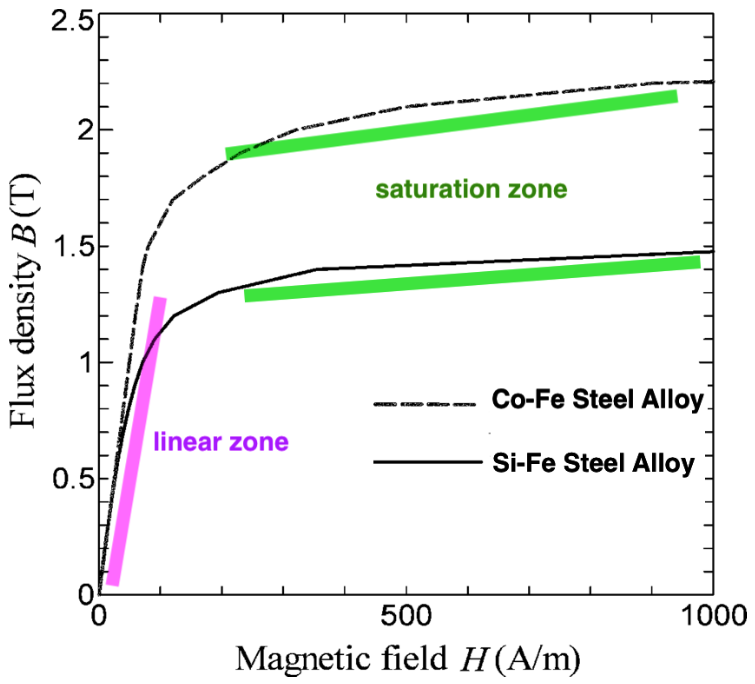


Figure 3.1: *Magnetization curve of common core steels, highlighting the linear and saturated region.*

constant. Thus, saturation puts a limit on the maximum magnetic fields achievable in ferromagnetic-core and therefore it affects the machine behaviour.

3.1.3 Electric Machine Magnetic Saturation Literature

The inclusion of the magnetic saturation in the machine equations is not a new concept, but the complexity of including time-dependent parameters has limited the successful use in a simple and effective control algorithm that could be run real-time on the application. Moreover, the use of the Unified Machine Model with the nonlinear approach valid for both IM and SM is something that surely adds to the present solutions.

Static description of the magnetic saturation phenomenons are provided in [6] and [7], but use heavy calculations that are not suited for real-time and are directed more for Finite Element Models verification. They do provide, however, the general methodology that involves finding the flux characteristic and deriving it to get the inductances required.

Many authors [8] [9] [10] [11] [12] implement nonlinear control algorithm that do not account for changes in the inductance value. These methods implement the dynamic machine response together with the magnetic model for control purposes without calculating the machine parameters. While they can provide effective control techniques, they do not provide the correct parameters that could be used by other functions like Field Oriented Control to employ highly effective algorithms, usually based on the parameters knowledge.

Others use a feedback error correction loop to adapt the model from its results either by recalculating the parameters directly [13] for controls and also studying their stability [14]. Partial error correction is carried out in [15] just on the resistance value to account for iron losses, while [16] focused only on saliency and cross-coupling without magnetic saturation. These methods have the advantage that they do not require extensive tests on the machine to find the flux equations and they can work with a wide range of variabilities, but are somewhat more complex that they need to be.

The approach that this article follows is found in other works, but the one proposed here is distinctive because of the generality, its purpose of calculating the machine parameters and its relative simplicity. For example, [17] uses the flux maps, but just to directly control the machine through minimization of energy losses without providing flux or parameter estimation. Similarly, [18] uses the method to provide just a sensorless rotor position calculation. [19] does not generalize to machines with different mutual inductances on the two axis and uses the method through high frequency injection to provide parameter estimation only when requested.

The examples in the paragraph above, moreover, are all just for synchronous machines. Examples specific just to induction machines can be found which suffer for similar problems. [20] and [21] use very complicated formulations and the use of dynamic equation for direct control of the induction machine, they can provide flux estimation, but the are not optimized for it. A decoupled-constant-parameter voltage-behind-reactance (DCPVBR) is presented for 6-phases IM in [22] that uses flux characterisation to implement saturation and cross-coupling, but lacks the simplicity and the generality of the method presented here with the model-based flux observer.

A similar application is found in [23], but it bases the algorithm on exponential approximation of the flux maps, where the present paper is independent from the interpolation used. Moreover, the former is focused on FEM verification and it is not really optimized for real-time control.

Another improvement proposed is the use of linear approximation around the selected machine status for very small changes, featured also in [24], but for spatial approximation of saliency, not saturation nonlinearities. This means that the parameter derivation from the flux maps can be executed separately from the actual control step, allowing for increase performance at high speed.

To note that the correct nonlinear parameter identification, together with a good flux and current estimation, works in synergy with a nonlinear Field Oriented (FOC) and Flux Weakening Control (FWC) to obtain optimal torque delivery performance and correct Maximum Torque per Ampere (MTPA) and Maximum Torque per Volt (MTPV) even at high saturation values. This aspect is explored in the next chapter 4.

3.1.4 Nonlinear Machine

In the magnetic saturation regime the machine fluxes cannot be considered linearly dependent on the currents, therefore the use of constant inductances is no longer valid and the expression becomes a complex nonlinear function.

The traditional approach to this problem is to use the linear model and updating the inductances using their change compared to the value at current next to zero, as proposed in [25] [26]. With this approach the parameters can be thought as differential approximations based on equivalent values and the flux equations do not change, with respect to the linear version. This solution can produce good results, but has many problems:

- the equivalent inductances may be substantially different from the real values;
- it does not account for cross-coupling;
- it does not ensure that it can work in the Field Oriented Control to produce the optimal operating curves.

Another important aspect to consider is that many types of machines do not have isotropic magnetic circuits, thus producing different magnetization behaviours on the d and q axes. Therefore, in general, the flux characteristic in an electric machine is described in the d-q reference frame by a 3D surface not proportionally dependent on the d and q currents. Such function is here expressed as $\Phi(i_d, i_q)$ and there are two of them to characterize each machine, one for the d axis (Φ_d) and one for the q axis (Φ_q)

that must be found either by experimental data or finite elements models.

The machine flux for a SM at any given current state is therefore define by finding the corresponding point on the characterization curve (3.17).

$$\begin{cases} \varphi_d = \Phi_d(i_d, i_q) \\ \varphi_q = \Phi_q(i_d, i_q) \end{cases} \tag{3.17}$$

The idea is to use only what can be directly measured from experimental data on the motor or other reliable data like FEM.

The experimental way of building these curves is carried out by injecting determined values of current in the windings at a pre-defined constant speed and by recording the value of the voltage produced. By assuming a measurement in steady state, $\frac{d\varphi}{dt} = 0$, it is quite easy to calculate the fluxes by inverting (3.15), yielding to (3.18).

$$\begin{cases} \varphi_d = \frac{-r_s i_q + v_q}{\omega_e} \\ \varphi_q = \frac{r_s i_d - v_d}{\omega_e} \end{cases} \tag{3.18}$$

It is important to note that using (3.18) the excitation flux (or the permanent magnet one depending on the machine) is not explicitly separated, but it is considered together inside the d axis flux φ_d. For the mathematical formulation of the model it is not a problem, it even helps reducing the writing complexity.

Once the curves have been obtained the inductances can be extrapolated from them.

Usually, for every point (i_o, φ_o) along the flux characteristic the inductance value is calculated simply by division $L = \frac{\varphi_o}{i_o}$. This method works great in the linear region of the magnetization since it models a proportional relation between flux and current. However, it is no longer a good approximation in high saturation regions.

The solution is therefore to use a more general inductance definition in the form of the derivative of the curve, $L = \frac{d\varphi}{di}$. This leads to the distinction between the two definitions, shown in Fig. 3.2, by calling the former method *equivalent inductance* or $L_{\mathtt{equiv}}$ and the latter *derivative inductance* or $L_{\mathtt{der}}$. Moreover, since in general the flux characteristic is function of both axes currents, the use of partial derivatives is needed. (3.19) and (3.20) express the proposed general definitions.

$$L_{\mathtt{equiv}} = \frac{\varphi_o}{i_o} \tag{3.19}$$

$$L_{\mathtt{der}} = \frac{\partial \varphi}{\partial i} = \frac{\partial \Phi(i_d(t), i_q(t))}{\partial i} \tag{3.20}$$

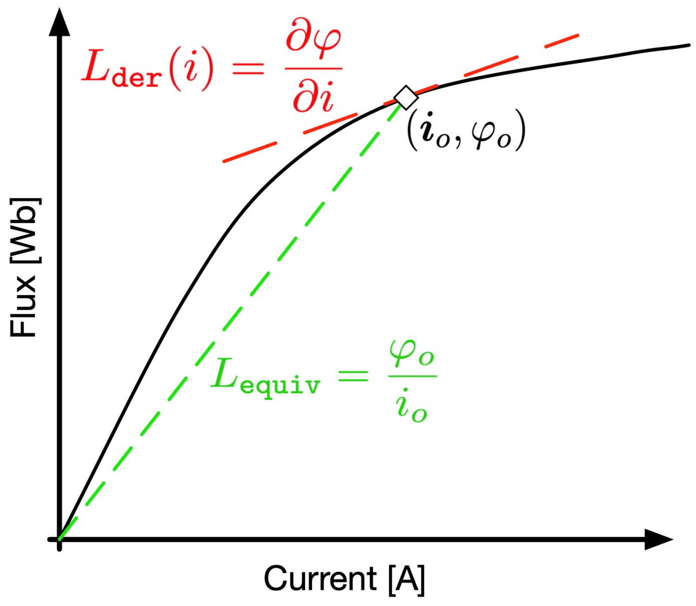

$L_{\mathtt{der}}(i) = \frac{\partial \varphi}{\partial i}$
$(\boldsymbol{i}_o, \varphi_o)$
$L_{\mathtt{equiv}} = \frac{\varphi_o}{i_o}$
Flux [Wb]
Current [A]

Figure 3.2: *Graphic representation of equivalent and derivative inductances on a simple magnetic characteristic.*

As an example of real values, Fig. 3.3 shows the two flux characteristic of an Internal Permanet Magnet Synchronous Machine.

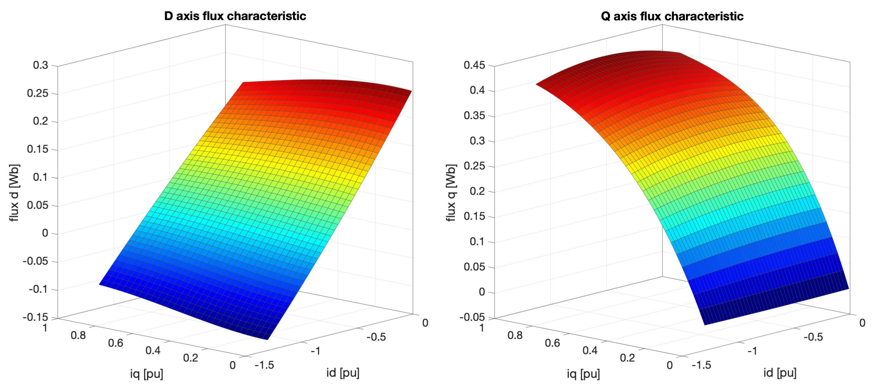


Figure 3.3: *Example of saturated flux characteristic for IPMSM dependant on both currents.*

By inserting the new inductance definitions inside the machine and flux equations, (3.3) and (3.4) respectively, it is possible to see how the previous linear approach is no longer valid because the flux can no longer be described by simply inductance times current and its derivative in time. The nonlinear formulation is a much more complex affair involving partial derivatives in current and time.

Simplifying the equations is key to obtain a final runnable code that is efficient and precise. This has been achieved by the derivative linearization around the calculation point, making use of the fact that the algorithm, in any case, is to be discretize.

To explain how this is achieved and how this helps, (3.21) re-proposes the flux characteristic (3.17) in a generalized vectorial form and highlights the function dependancies.

$$\varphi(\boldsymbol{i}(t)) = \boldsymbol{\Phi}(\boldsymbol{i}(t)) \tag{3.21}$$

If it is assumed that the operating point at a given instant is $\boldsymbol{i}_o$ that is not time-dependent, then it is possible to linearize the flux expression (3.21) around this point by taking the derivative in that point and adjusting for the linear error in a small neighborhood around it. In practice the new equation is (3.22).

$$\begin{aligned}\varphi(\boldsymbol{i}(t)) &= \left[\frac{\partial \boldsymbol{\Phi}(\boldsymbol{i}_o)}{\partial \boldsymbol{i}_o}\right](\boldsymbol{i}(t) - \boldsymbol{i}_o) + \boldsymbol{\Phi}(\boldsymbol{i}_o) = \\ &= [L_{\mathrm{der}}(\boldsymbol{i}_o)](\boldsymbol{i}(t) - \boldsymbol{i}_o) + \boldsymbol{\Phi}_o\end{aligned} \tag{3.22}$$

From now on, when talking about nonlinear machines, the inductances are all considered derivative when not explicitly said otherwise, so the matrix $[L_{\mathrm{der}}]$ will be reported as $[L]$ for simplicity.

The flux characteristic looses the time dependance and allows a simplification since the linearization takes care of the time variation. Of course the calculation instant $\boldsymbol{i}_o$ must be updated relatively frequently to ensure the validity of the linearization.

The voltage equation (3.3) is therefore rewritten as (3.23), while the previous (3.22) is already the flux one.

$$\begin{aligned}\boldsymbol{v} &= [r]\boldsymbol{i} + \frac{d\varphi(\boldsymbol{i}(t))}{dt} \\ &= [r]\boldsymbol{i} + \frac{d}{dt}\Big([L(\boldsymbol{i}_o)](\boldsymbol{i}(t) - \boldsymbol{i}_o) + \boldsymbol{\Phi}_o\Big) \\ &= [r]\boldsymbol{i} + \frac{d}{dt}\Big([L(\boldsymbol{i}_o)]\boldsymbol{i}(t)\Big) - \frac{d}{dt}([L(\boldsymbol{i}_o)]\boldsymbol{i}_o) + \frac{d}{dt}(\boldsymbol{\Phi}_o) \\ &= [r]\boldsymbol{i} + [L(\boldsymbol{i}_o)]\frac{d}{dt}(\boldsymbol{i}(t)) - 0 + 0 \\ &= [r]\boldsymbol{i} + [L(\boldsymbol{i}_o)]\frac{d\boldsymbol{i}(t)}{dt}\end{aligned} \tag{3.23}$$

The nonlinear inductances are therefore calculated as (3.24).

$$[L] = [L_{\mathrm{der}}] = [L(\boldsymbol{i}_o)] = \frac{\partial \boldsymbol{\Phi}(\boldsymbol{i}_o)}{\partial \boldsymbol{i}_o} \tag{3.24}$$

It is critical to point out how (3.23) and (3.22) are pretty much written in the same way as the linear counterparts (3.7) and (3.8).

However, because the flux maps $\boldsymbol{\Phi}$ are measured in the rotor flux reference frame, the nonlinear expressions are only valid in the rotating frame. This is the main reason why the implemented models are differentiated between SM and IM for the nonlinear approach, since for the latter type the rotor flux direction is more computationally difficult to evaluate.

3.2 DISCRETE LINEAR MODEL

Given the general machine equations in the natural reference frame (3.13) and (3.14), they need to be implemented in a discrete solver environment to be able to run on Digital Signal Processors (DSPs) devices.

Substituting the flux derivative with its discrete version of difference over time difference, the flux status φ can be calculated using recursive algorithm every sampling time. Sub-index n indicates the step. Starting from (3.13), assuming $\varphi_e = 0$ just for easier visualization and operating some grouping of the parameters, the discrete machine equation becomes (3.27).

$$\dot{\varphi} = -[R][T]^{-1}[L]^{-1}[T]\varphi + \boldsymbol{v} \tag{3.25}$$

$$\frac{\varphi_n - \varphi_{n\text{-}1}}{t_c} = -[R][T]^{-1}[L]^{-1}[T]\varphi_n + \boldsymbol{v}_n \tag{3.26}$$

$$\frac{\varphi_n - \varphi_{n\text{-}1}}{t_c} = [A_k]\varphi_n + \boldsymbol{v}_n \tag{3.27}$$

With $[A_k] = [A(\vartheta(t))] = -[R][T]^{-1}[L]^{-1}[T]$ and t_c the sample time.

Solving for φ_n, the recursive step can be evaluate in (3.28), with $[I]$ the identity matrix.

$$\begin{aligned} \varphi_n &= t_c[A_k]\varphi_n + t_c\boldsymbol{v}_n + \varphi_{n\text{-}1} \\ \varphi_n &= ([I] - t_c[A_k])^{-1}(t_c\boldsymbol{v}_n + \varphi_{n\text{-}1}) \end{aligned} \tag{3.28}$$

This algorithm allows the integration of the fluxes, but it contains a time-dependent matrix inversion that is very time consuming. Focusing on this problem, the coefficient can be reworked as shown in (3.29).

$$\begin{aligned} ([I] - t_c[A_k])^{-1} &= ([I] + t_c([R][T]^{-1}[L]^{-1}[T]))^{-1} = \\ &= ([T]^{-1}([T][T]^{-1} + t_c[R][L]^{-1})[T])^{-1} = \\ &= [T]^{-1}([I] + t_c[R][L]^{-1})^{-1}[T] = \\ &= [T]^{-1}([R][L]^{-1}([L][R]^{-1} + t_c[I]))^{-1}[T] = \\ &= [T]^{-1}([L][R]^{-1} + t_c[I])^{-1}[L][R]^{-1}[T] \end{aligned} \tag{3.29}$$

Thus, it is possible to introduce expA as (3.30) to group all machine parameters.

$$\mathtt{expA} = ([L][R]^{-1} + t_c[I])^{-1}[L][R]^{-1} = \frac{[I]}{([I] + t_c[R][L]^{-1})} \tag{3.30}$$

Consequently, the integrating algorithm becomes (3.31).

$$\varphi_n = [T]^{-1}\mathtt{expA}[T](t_c v_n + \varphi_{n\text{-}1}) \tag{3.31}$$

This approach has 2 main advantages:

- expA does not depend on the position ϑ, thus it is time invariant in case of constant switching frequency ($t_c = \text{const}$).
- expA allows the use of infinite resistance coefficients for machines with no rotor currents because the matrix $[R]$ is always used as inverse. In fact, $[R]$ is diagonal and in case of a coefficient $r = \infty$ the corresponding element in $[R]^{-1}$ is simply 0.

3.2.1 Optimization and Sub-Interval Predictive Integration

Another important optimization involves the use of an integrating method that divides the step into smaller ones (sub-intervals) for a more precise calculation without decreasing the step size. Since the integrating values are rotating vectors, the increase in precision is significant, especially at high speed. Moreover, the increase in computation complexity can be limited thanks to the appropriate simplification and reference frame selection. This method has been described in [27].

At this stage, the implementation of the predictive algorithm can also take place. In fact, (3.32) shows the recursive function at step n with all the correct timing.

$$\varphi_n = [T_n]^{-1}\mathtt{expA}[T_n](t_c v_n + \varphi_{n\text{-}1}) \tag{3.32}$$

Assuming a constant rotational speed during the step, the same identity can be rewritten to calculate the flux in the next step (n+1) using only the currently measured values (v_n) and the approximated rotational matrix of the next step ($[T_{n+1}] = [T_n]$), yielding to (3.33).

$$\varphi_{n+1} = [T_{n+1}]^{-1}\mathtt{expA}[T_{n+1}](t_c v_n + \varphi_n) \tag{3.33}$$

Introducing now the sub-division of the integration interval like in the diagram of Fig. 3.4, the rotation of interval n changes the

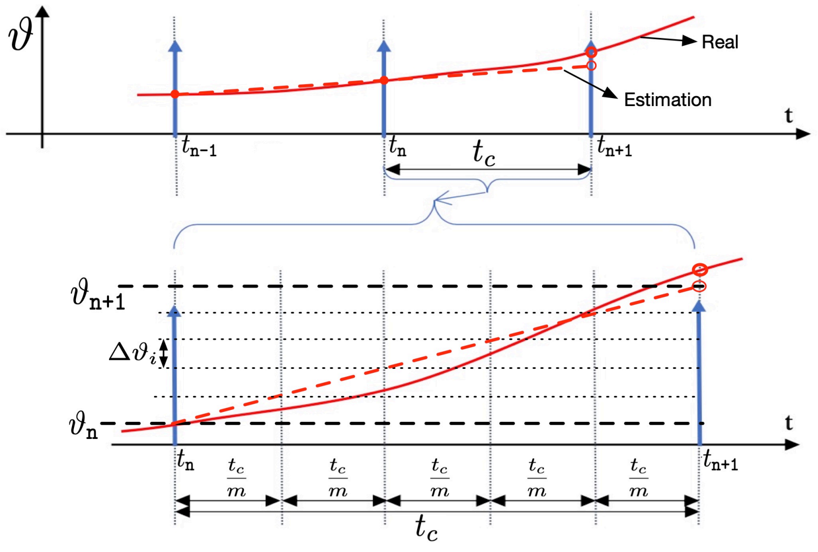


Figure 3.4: *Discrete time interval sub-division diagram.*

position of the mechanical angle from ϑ_n to ϑ_{n+1}. The rotation n is then divided in m sub-intervals. The rotation angle ϑ_i for any sub-interval $i|_{1:m}$ of interval n is considered equal to the rotation occurred in the previous step n-1.

$$\Delta\vartheta_i = \frac{\Delta\vartheta_n}{m} = \frac{\vartheta_n - \vartheta_{n\text{-}1}}{m} \tag{3.34}$$

The infinitesimal rotational matrix $[\Delta T_i]$ inside every step n is then calculated in (3.35).

$$[\Delta T_i] = [T(\Delta\vartheta_i)] = \left[T\left(\frac{\Delta\vartheta_n}{m}\right)\right] \tag{3.35}$$

Given that the rotation at step n is described by $[T_n] = [T(\vartheta_n)]$, the rotation for a generic sub-interval is $[T_i] = [T(\vartheta_i)] = [T(\vartheta_n + i \cdot \Delta\vartheta_i)]$. For the property of the rotation matrix there is also an iterative method for this calculation, given in (3.36) for both direct and inverse transformation.

$$\begin{gathered} [T_i] = [T(\vartheta_n + i \cdot \Delta\vartheta_i)] = [T_n]([\Delta T_i])^i \\ [T_i]^{-1} = [T(\vartheta_n + i \cdot \Delta\vartheta_i)]^{-1} = [T_n]^{-1}([\Delta T_i]^{-1})^i \\ \forall\, i|_{1:m} \text{ of interval } n \text{ in which } \Delta\vartheta_i \text{ is constant} \end{gathered} \tag{3.36}$$

Using (3.36), the iterative integration (3.33) becomes (3.38).

$$\varphi_{i+1} = [T_i]^{-1}\mathrm{expA}[T_i]\left(\frac{t_c}{m}v_n + \varphi_n\right) \tag{3.37}$$

$$\varphi_{i+1} = [T_n]^{-1}([\Delta T_i]^{-1})^i\mathrm{expA}[T_n]([\Delta T_i])^i\left(t_i v_n + \varphi_n\right) \tag{3.38}$$

With $\frac{t_c}{m} = t_i$ as the step time of the sub-interval.

In this implementation the iteration requires only the calculation of the power of $[\Delta T_i]$ and $[\Delta T_i]^{-1}$. The complete step is done fairly quickly and with high precision. It is worth noting that all the $[T]$ and $[\Delta T]$ matrices are almost diagonal with a very easy to calculate inverse, it is just a matter of changing the sign of the sin elements in position 12 and 21.

Additional simplifications can be applied if the reference frames are taken into consideration.

So far all the inputs, outputs and states in (3.38) are kept into their respective reference frames (their *natural* frames). This means that, since the magnetic coupling requires the terms to be into the same reference frame, the stator values are transformed into the rotor reference frame for the interaction and then transformed back after each sub-interval.

To optimize this step, the general idea is then to have the heavier calculations inside the iteration only in the rotor reference frame and transform back in the natural frame only the final results. To do this, (3.38) is presented again in (3.39) this time with the reference frames shown explicitly in the superscripts (n, s, and r for natural, stator and rotor respectively).

$$\begin{aligned}\varphi^n_{i+1} &= [T_n]^{-1}([\Delta T_i]^{-1})^i \mathtt{expA}[T_n]([\Delta T_i])^i\left(t_i v^n_n + \varphi^n_i\right)\\ &= [T_n]^{-1}([\Delta T_i]^{-1})^i \mathtt{expA}\left(t_i v^n_n [T_n]([\Delta T_i])^i + \varphi^n_i [T_n]([\Delta T_i])^i\right)\end{aligned} \tag{3.39}$$

Applying the rotations $v^r_n = v^n_n[T_n]$ and $\varphi^r_i = \varphi^n_i[T_n]$, (3.40) can be derived.

$$\varphi^n_{i+1} = [T_n]^{-1}([\Delta T_i]^{-1})^i \mathtt{expA}\left(t_i v^r_n([\Delta T_i])^i + \varphi^r_i([\Delta T_i])^i\right) \tag{3.40}$$

And then, (3.41) is derived where the voltages and fluxes are both in the rotor reference frame.

$$\varphi^n_{i+1} = [T_n]^{-1}([\Delta T_i]^{-1})^i \mathtt{expA}\left(t_i v^r_n + \varphi^r_i\right)([\Delta T_i])^i \tag{3.41}$$

Equation (3.41) is a recursive function that takes the starting voltage, transports it into the rotor reference frame, sum and multiplies the fluxes for the next sub-iteration and finally brings it all back into the natural reference frame at each iteration. This double transformation can be avoided by leaving voltage and flux in the rotor reference frame during the m iterations and transforming just the last results by using the rotational matrix calculated for the cumulative angle increment.

In this way (3.42) can be isolated from (3.41) , showing just the recursive function for every sub-interval for the rotor flux.

$$\varphi^{r}_{i+1} = \mathtt{expA}[\Delta T_i]\left(t_i v^{r}_{n} + \varphi^{r}_{i}\right) \tag{3.42}$$

In (3.42) the integral iteration uses only rotor values (constant in the period t_c) and, being recursive, the sub-interval angle $[\Delta T_i]$ does not require elevation because it will be achieved by the multiplication by itself in the next sub-interval step since the resulting φ^{r}_{i+1} will become φ^{r}_{i}.

To note how also the input voltage is rotated to account for the fact that, since in the stator reference frame it is assumed fixed and constant, in the rotor reference frame it is counter-rotating (because the frame rotates).

The final rotation angle can be calculated recursively inside the integration, or outside using (3.43) depending on which is the less demanding option for the DSP used.

$$\begin{aligned} [T_{n+1}]^{-1} &= [T_n]^{-1}([\Delta T_i]^{-1})^{m} \\ &= ([T_n][\Delta T_i]^{m})^{-1} \end{aligned} \tag{3.43}$$

Uniting (3.42) at the last step m and (3.43) yields to the last calculation (3.44) to get the final flux in the predictive n+1 step in the natural reference frame.

$$\varphi^{n}_{n+1} = [T_{n+1}]^{-1}\varphi^{r}_{m} \tag{3.44}$$

The schematic implementation of the Unified Model Algorithm here presented can be found in the Fig. 3.5

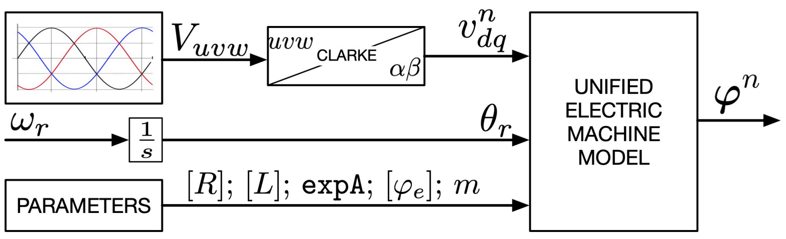


Figure 3.5: *Schematic overview of the linear Unified Model algorithm implementation.*

Finally, the excitation flux is considered in the complete implementation. Starting from the generalized expression (3.13) and using the rotor flux reference frame (3.42) where the excitation is constant, the only operation is to sub-divide its integration contribution along the calculation step by multiplying the parameters by the time interval t_i. The result is (3.45).

$$\varphi^r_{i+1} = \mathtt{expA}[\Delta T_i]\left(t_i \boldsymbol{v}^r_n + \varphi^r_i\right) + \mathtt{expA}[R][L]^{-1} t_i\, \varphi_e \tag{3.45}$$

The final implementation of any of the above functions on the real motor control would, of course, require an error correction feedback to prevent dangerous integration offsets or erroneous parameters setup and update. Such system will be discussed in a specific section together with additional optimization that simplify the matrix operations.

3.3 DISCRETE NONLINEAR MODEL

In the general case, the [L] matrix is the derivative of the flux characteristic, as seen in section 3.1.4. From an operating point of view, this characteristic must then be constructed in a way that allows its derivation. Since the curve is usually defined through a multitude of experimental points, an interpolation method is the best way to link and smooth the points. There are a few options available like spline, Akima, linear or even nearest point, but one in particular presents the most advantages.

The polynomial interpolation could be considered the best solution because:

- as long as it is first-order or higher, its derivability is ensured;
- it provides a way to apply boundary conditions that are realistic in nature, such as zero q flux when $i_q = 0$;
- it intrinsically produces a smooth surface without fast variation and it can be tailored through its order;
- it is less prone to lose control outside the experimental points boundary, allowing a certain security in the extrapolation region;
- it is somewhat less reliant on a high number of points and it is more robust to experimental noise;
- once the coefficient are found, it is extremely computationally efficient and accurate.

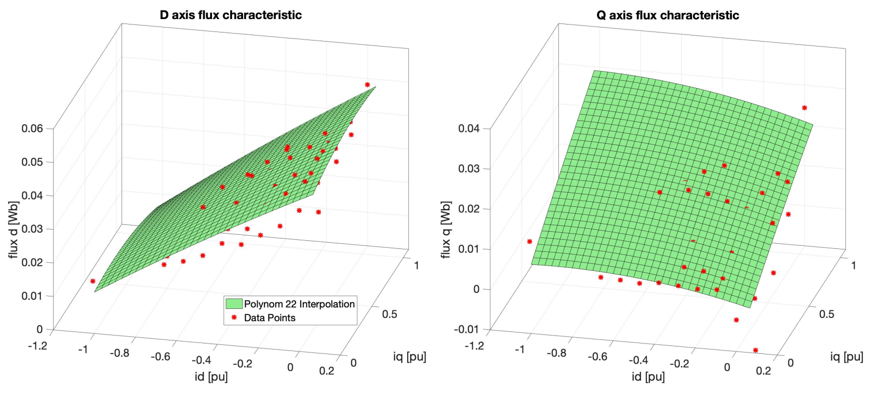


Figure 3.6: *Example of polynomial flux characteristic interpolation of a Synchronous Machine.*

In fact, the only disadvantage is that the coefficient calculation is very time consuming and it is usually best carried out offline.

For this reasons, even though a spline interpolation was briefly evaluated, the polynomial one was chosen for the tests and application presented in this work.

Fig. 3.6 shows an example of a polynomial interpolation of sparse data for a Synchronous Machine, and it is presented as two characteristics, one for the d axis and one for the q. It is interesting to see how just a 2nd order curve is able to average out possible measurement errors and provides a reasonable extrapolation.

3.3.1 Synchronous Machine

Contrary to the case of the linear machine, since the flux and inductance data is best linked to and used in the rotor reference frame, it makes sense to operate the integration steps directly in the d-q reference frame to avoid costly double transformations.

In a linear SM model and also in the model using equivalent inductances, the flux equation would simply be (3.46).

$$\begin{bmatrix} \varphi_d(i_d, i_q) \\ \varphi_q(i_d, i_q) \end{bmatrix} = \begin{bmatrix} L_d & 0 \\ 0 & L_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} \varphi_{es} \\ 0 \end{bmatrix} \tag{3.46}$$

Where the inductance matrix [L] has just the two constants L_d and L_q which are the equivalent values describing the linear relationship between currents and fluxes.

However, as anticipated in the nonlinear section 3.1.4, for Synchronous Machine the flux characteristic is different for the two axes and function of both currents. This results is the need to use partial derivatives and constructing a full inductance matrix, as presented in (3.47)

$$\begin{aligned}[L(i_{od}, i_{oq})] &= \begin{bmatrix} L_{dd}(i_{od}, i_{oq}) & L_{dq}(i_{od}, i_{oq}) \\ L_{qd}(i_{od}, i_{oq}) & L_{qq}(i_{od}, i_{oq}) \end{bmatrix} = \\ &= \begin{bmatrix} \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_d} & \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_q} \\ \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_d} & \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_q} \end{bmatrix} = [L]\end{aligned} \tag{3.47}$$

The application of the linearization simplification around a given operating point $\boldsymbol{i}_o = (i_{od}, i_{oq})$ discussed like in (3.22) is the key for the solution of the integration function and has the added benefit that in the discretization process the actual instantaneous calculating point may be a little different than the flux input.

With this the flux equation (3.22) can be expanded for Synchronous Machine into (3.48).

$$\begin{aligned}\begin{bmatrix} \varphi_d(i_d, i_q) \\ \varphi_q(i_d, i_q) \end{bmatrix} &= \begin{bmatrix} L_{dd}(i_{od}, i_{oq}) & L_{dq}(i_{od}, i_{oq}) \\ L_{qd}(i_{od}, i_{oq}) & L_{qq}(i_{od}, i_{oq}) \end{bmatrix} \begin{bmatrix} i_d - i_{od} \\ i_q - i_{oq} \end{bmatrix} + \\ &+ \begin{bmatrix} \varphi_{od}(i_{od}, i_{oq}) \\ \varphi_{oq}(i_{od}, i_{oq}) \end{bmatrix}\end{aligned} \tag{3.48}$$

The compact form (3.22) is still, obviously, true.

The currents can therefore be calculated from the fluxes by expression (3.49).

$$\boldsymbol{i} = [L]^{-1}(\boldsymbol{\varphi} - \boldsymbol{\varphi}_o) + \boldsymbol{i}_o \tag{3.49}$$

Of course, since the flux maps $\boldsymbol{\Phi}$ and the derivations are all done in the rotor flux reference frame (which for SM coincides with the rotor reference frame), the equation (3.48) is only valid in the rotor flux reference frame and all input and outputs should be in that frame.

The voltage equation (3.23) is also valid only in the rotor flux reference frame and it is used to evaluate the fluxes by means of an integrating function similar to the one for linear machines. Together with the flux equation, the solution of the system of (3.50) and (3.51) allows, as before, the calculation of the machine flux.

$$\dot{\boldsymbol{\varphi}} = -[R] \cdot \boldsymbol{i} + \boldsymbol{v} \tag{3.50}$$

$$\boldsymbol{i} = [L]^{-1}\boldsymbol{\varphi} - ([L]^{-1}\boldsymbol{\varphi}_o - \boldsymbol{i}_o) \tag{3.51}$$

For optimization purposes, the constant values $([L]^{-1}\boldsymbol{\varphi}_o - \boldsymbol{i}_o)$ are grouped together from (3.49). This way the resulting value can be calculated only once at every parameter recalculation, thus reducing the total number of operations.

Then, substituting the currents (3.51) in the electric machine equation (3.50), the model equation becomes (3.52).

$$\dot{\varphi} = -[R][L]^{-1}\varphi + [R]\left([L]^{-1}\varphi_0 - \boldsymbol{i}_0\right) + \boldsymbol{v} \tag{3.52}$$

As it is plain to see, the formulation is quite similar to the linear one (3.25) without the rotation matrices accounting for the natural reference frame and with the additional term for the linearization.

The overall model algorithm will therefore follow these steps:

1. Rotor angle measurement;
2. Currents and voltages calculation, either measurement (in the stator reference frame) or derivation from observer;
3. Current and voltage rotation in the rotor reference frame, if necessary;
4. Calculation of fluxes (interpolation) and differential inductances;
5. Update of the parameters and constants;
6. Model integration.

The rotation in the rotor reference frame at step 3 can be skipped if the currents are derived from the model integration, since the outputs of the observer are already in the reference frame required.

Moreover, thanks to the use of the linear correction of the flux equation, steps 4 and 5 can be done asynchronously and at lower frequency (within reason) with good results.

Fig. 3.7 shows the approach described in schematic form.

Since the ultimate goal is to provide a solution for real applications that use limited resources microchips and real time operation, the calculations are done in discrete-time. Therefore, as shown before in section 3.2 and assuming the same use of the predictive method, the flux derivative is the difference between steps n+1 and n, divided by the calculation step time t_c. The goal is to find the flux at the next step using the recursive algorithm. The model equation becomes (3.53).

$$\dot{\varphi} = \frac{\varphi_{n+1} - \varphi_n}{t_c} = -[R][L]^{-1}\varphi_{n+1} + [R]\left([L]^{-1}\varphi_0 - \boldsymbol{i}_0\right) + \boldsymbol{v}_n \tag{3.53}$$

Again very similarly to the linear approach described by (3.33), solving for φ_{n+1} yields to (3.54), the predictive recursive solving algorithm for the machine model.

$$\varphi_{n+1} = \frac{[I]}{([I] + t_c[R][L]^{-1})}\left(t_c\boldsymbol{v}_n + \varphi_n + t_c[R]\left([L]^{-1}\varphi_0 - \boldsymbol{i}_0\right)\right) \tag{3.54}$$

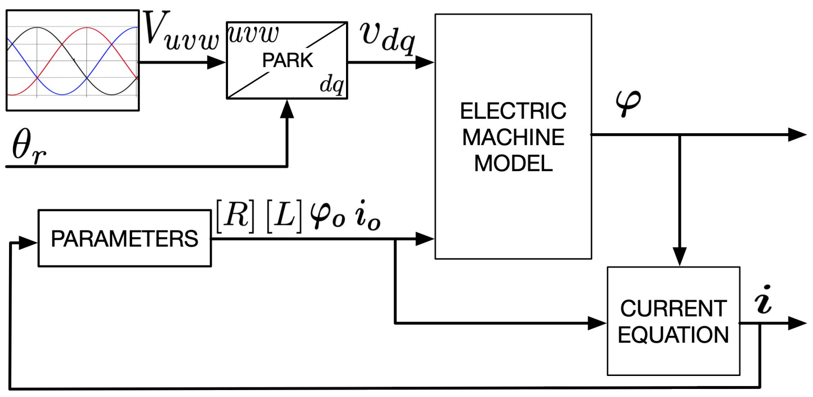


Figure 3.7: *Schematic overview of the nonlinear integration algorithm.*

Inside the integration the updated values are just the fluxes, so it is possible to calculate some constants outside the integration loop, renamed `expA` (exactly as in (3.30)) and `tcRIo` as shown in (3.55) and (4.20). Since they are only dependant on the machine parameters and the time step, this calculations can also be done outside the main integration loop, just when either the parameters or time step change.

$$\mathtt{expA} = \frac{[I]}{([I] + t_c[R][L]^{-1})} \tag{3.55}$$

$$\mathtt{tcRIo} = t_c[R]\left([L]^{-1}\varphi_o - i_o\right) \tag{3.56}$$

The interval integration algorithm (3.54) in compact form becomes (3.57), which is almost exactly similar to the linear approach in the rotor reference frame (3.42).

$$\varphi_{n+1} = \mathtt{expA} \cdot \left(t_c v_n + \varphi_n + \mathtt{tcRIo}\right) \tag{3.57}$$

Therefore, as done in the linear machine, it is possible to apply the same interval sub-division and repeat the integration for for $i = 1, ..., m$, thus producing the final implemented form (3.58).

$$\varphi_{i+1} = \mathtt{expA} \cdot \left([\Delta T_i]\left(t_c v_n + \varphi_i\right) + \mathtt{tcRIo}\right) \tag{3.58}$$

Afterwards, the rotor flux can be transformed back in the stator reference frame using the matrix rotation [T], the same used to convert the measurements in the rotor reference frame at the start of the algorithm. For predictive algorithms, the angle could be the predicted angle for the next cycle, again in the same way as the linear one (3.44).

To note how there is no explicit contribution from the permanent magnet flux, which is found in the traditional linear model. This is because it is automatically inside the d axis flux φ_d, since during the measurements for the maps their contribution is measured together.

From the fluxes, the currents are easily calculated using (3.51), always accounting for the possible difference between the instantaneous point and the actual point at which the parameters have been found. With again the constant $[L]^{-1}\boldsymbol{\varphi}_0 + \boldsymbol{i}_0$ that can be calculated separately.

Given the non-linearities of the system, it has been found that the use of the conservative flux yields better results for the prediction of the currents, since it is the only part that contributes to the current generation in the motor. The actual calculations are quite easy and just requires the use of the symmetric part of the inductance matrix $[L]_{\mathtt{sym}}$ (and its inverse $[L]^{-1}_{\mathtt{sym}}$) in place of the regular one $[L]$.

Thus the integration (3.57) and currents calculation (3.51) become (3.59) and (3.60) respectively. From them the sub-interval derivation is straight-forward.

$$\boldsymbol{\varphi}_{\mathtt{n+1}_{\mathtt{cnsv}}} = \frac{[I]}{\left([I] + t_c[R][L]^{-1}_{\mathtt{sym}}\right)}\Big(t_c\boldsymbol{v}_{\mathtt{n}} + \boldsymbol{\varphi}_{\mathtt{n}_{\mathtt{cnsv}}} + t_c[R]\big([L]^{-1}_{\mathtt{sym}}\boldsymbol{\varphi}_0 - \boldsymbol{i}_0\big)\Big) \tag{3.59}$$

$$\begin{aligned} \boldsymbol{i}_{\mathtt{n+1}} &= [L]^{-1}_{\mathtt{sym}} \cdot (\boldsymbol{\varphi}_{\mathtt{n+1}_{\mathtt{cnsv}}} - \boldsymbol{\varphi}_0) + \boldsymbol{i}_0 = \\ &= [L]^{-1}_{\mathtt{sym}}\boldsymbol{\varphi}_{\mathtt{n+1}_{\mathtt{cnsv}}} - \big([L]^{-1}_{\mathtt{sym}}\boldsymbol{\varphi}_0 + \boldsymbol{i}_0\big) \end{aligned} \tag{3.60}$$

For completion, the interval sub-division integration is also expressed in (3.61).

$$\boldsymbol{\varphi}_{\mathtt{i+1}_{\mathtt{cnsv}}} = \mathtt{expA}_{\mathtt{sym}} \cdot \Big([\Delta T_{\mathtt{i}}]\big(t_c\boldsymbol{v}_{\mathtt{n}} + \boldsymbol{\varphi}_{\mathtt{i}_{\mathtt{cnsv}}}\big) + \mathtt{tcRIo}_{\mathtt{sym}}\Big) \tag{3.61}$$

There is just an additional step to take since now all the model is in the conservative flux and it is the step to calculate the correct total flux from the conservative one by adding the asymmetric part $[L]_{\mathtt{asym}}$, as shown in (3.62).

$$\boldsymbol{\varphi}_{\mathtt{n+1}} = \boldsymbol{\varphi}_{\mathtt{n+1}_{\mathtt{cnsv}}} + [L]_{\mathtt{asym}}(\boldsymbol{i}_{\mathtt{n+1}} - \boldsymbol{i}_0) \tag{3.62}$$

3.3.2 Induction Machine

The Induction Machine (IM) is a considerably more complex machine because of the interactions between stator and rotor variable fields.

Instead of just the stator fluxes (3.17) describing completely the machine, the proper implementation would consider the IM four fluxes (φ_{sd}, φ_{sq}, φ_{rd}, φ_{rq}) all as functions of the four currents, (3.63).

$$\varphi = \Phi(i_{sd}, i_{sq}, i_{rd}, i_{rq}) \tag{3.63}$$

However, this would involve finding flux maps depending on 4 variables, only 2 of which are measurable (i_{sd}, i_{sq}). For this reason, two approximations are introduced.

The first one is that the leakage inductances are constants, so that only the mutual inductances depend on the currents, while the leakage is just linearly depended on the corresponding current. Taking as example the stator d flux, it can be written as (3.64).

$$\begin{aligned}\varphi_{sd} &= \Phi_{msd}(i_{sd}, i_{sq}, i_{rd}, i_{rq}) + \Phi_l(i_{sd}, i_{sq}, i_{rd}, i_{rq}) = \\ &= \Phi_{msd}(i_{sd}, i_{sq}, i_{rd}, i_{rq}) + l_s i_{sd}\end{aligned} \tag{3.64}$$

Where l_s is the stator leakage inductance and Φ_{msd} is the function of the mutual stator flux on the d.

The second approximation is to disregard flux cross-coupling. This would be definitely acceptable in the amp-turns reference frame, but the model integration becomes too complex for an efficient control software. The approximation still holds well also in the rotor flux reference frame thanks to the machine being isotropic. It yields to the flux being dependent only on the sum of the currents on the relative axis, based on the rotor flux reference frame. Equation (3.64) becomes (3.65), as an example for the d axis.

$$\begin{aligned}\varphi_{sd} &= \Phi_{msd}(i_{sd}, i_{sq}, i_{rd}, i_{rq}) + l_s i_{sd} = \\ &= \Phi_{msd}(i_{sd} + i_{rd}, i_{sq} + i_{rq}) + l_s i_{sd} = \\ &= \Phi_{msd}(i_{sd} + i_{rd}) + l_s i_{sd}\end{aligned} \tag{3.65}$$

Moreover, thanks to the symmetry of the induction machine, the function Φ_{msd} works both on the d and q axes using as input the total current. The function can be, therefore, a simple 2D line described just by one function that can be simply called Φ_m. An example is presented in Fig. 3.8.

In this way, the fluxes are calculated as (3.66).

$$\begin{cases}\varphi_{sd} &= \Phi_m(i_{sd} + i_{rd}) + l_s i_{sd} \\ \varphi_{sq} &= \Phi_m(i_{sq} + i_{rq}) + l_s i_{sq} \\ \varphi_{rd} &= \Phi_m(i_{sd} + i_{rd}) + l_r i_{rd} \\ \varphi_{rq} &= \Phi_m(i_{sq} + i_{rq}) + l_r i_{rq}\end{cases} \tag{3.66}$$

The rotor currents used in this way are the ones in feedback from the model outputs, similarly to the stator ones. The difference is that there is no possible error correction based on the measurements. The calculations of the model are based on the fact that the rotor is in short circuit and, therefore, the rotor voltage is 0.

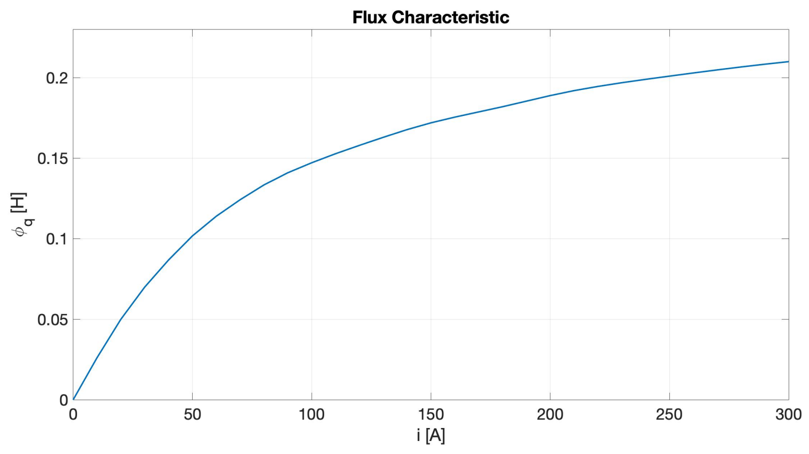


Figure 3.8: *Example of saturated flux characteristic for IM.*

Having worked out the flux connections, with the appropriate approximations, the inductance matrix is defined like in the synchronous machine (3.47) with its elements as derivatives of the flux. However, this time there is also to consider the rotor inductances and currents. Thus the previous 2x2 matrix becomes a 4x4 matrix, thankfully it is sparse and symmetric allowing for helpful optimisations in the calculations.

The L matrix is presented in (3.67).

$$[L] = \begin{bmatrix} L_{sd} & 0 & L_{msd} & 0 \\ 0 & L_{sq} & 0 & L_{msq} \\ L_{mrd} & 0 & L_{rd} & 0 \\ 0 & L_{mrq} & 0 & L_{rq} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\partial\Phi_m(i_d)}{\partial i_d} + l_s i_{sd} & 0 & \frac{\partial\Phi_m(i_d)}{\partial i_d} & 0 \\ 0 & \frac{\partial\Phi_m(i_q)}{\partial i_q} + l_s i_{sq} & 0 & \frac{\partial\Phi_m(i_q)}{\partial i_q} \\ \frac{\partial\Phi_m(i_d)}{\partial i_d} & 0 & \frac{\partial\Phi_m(i_d)}{\partial i_d} + l_r i_{rd} & 0 \\ 0 & \frac{\partial\Phi_m(i_q)}{\partial i_q} & 0 & \frac{\partial\Phi_m(i_q)}{\partial i_q} + l_r i_{rq} \end{bmatrix} \tag{3.67}$$

Like for the SM, the linearization (3.22) is used to correct the difference that may occur between the instantaneous operating point against the map point used for the inductances and fluxes calculations. The only difference is, of course, that now the L matrix is 4x4 and the current and flux vectors have 4 elements, but the equation is the same and it is still valid only in the rotor flux reference frame.

Given the very similar construction, the IM model integration equations and steps are pretty much the same as the SM with just a key difference. The flux function and the subsequents approximations are valid only in the rotor flux reference frame, therefore the integration must be carried out in this space using the rotor flux angle ϑ_a and the rotational matrix $[T_a] = [T(\vartheta_a)]$.

Therefore, a further rotation step is needed before the multiplication with the inductance matrix. Of course, afterwords, the flux must be rotated back in the rotor reference frame and the rotor flux angle calculated for the next step from the rotor flux. Since the model output is the flux state of the machine, it is easy to use its own output to calculate the flux rotation $[T_a]$ needed.

Accounting for this, the integration algorithm (3.57) becomes (3.68).

$$\varphi_{\mathtt{n+1}} = [T_a]^{-1}\mathtt{expA} \cdot \Big([T_a] \cdot (t_c \boldsymbol{v}_n + \varphi_n) + \mathtt{tcRIo}\Big) \tag{3.68}$$

The interval sub-division is once again actuated same as before and (3.69) shows the iterative step, derived in the same way as for the other occasions.

$$\varphi_{\mathtt{i+1}} = [T_a]^{-1}\mathtt{expA} \cdot \Big([T_a][\Delta T_i]\big(t_c \boldsymbol{v}_n + \varphi_{\mathtt{i}}\big) + \mathtt{tcRIo}\Big) \tag{3.69}$$

Of course, there is no distinction between symmetric and asymmetric parts of the fluxes (like in the Synchronous Machine) since the machine is isotropic. Consequently, the currents are calculated using the standard equation (3.51), just adding the rotor fluxes.

4 FIELD ORIENTED CONTROL

This chapter focuses its attention on the "FOC" part of the control algorithm of the scheme in Fig. 1.2. Its general goal is to provide the d and q current references to the regulators based on minimal use of the current to produce the requested torque while complying with the machine and supply limits.

Fig. 2.5, Fig. 2.7 and section 2.3.2 already introduced part of the structure of the functions since they provide the input/output definition and the execution timing. Those were preliminary considerations based on reasonable assumption at the beginning of the project. The final product is not very different than the one envisioned, but it has been surely expanded in functionalities and complexity, even though the overall flow is not changed substantially.

This chapter presents the mathematical explanation and derivation of the concepts used for the definition of the best operating point in terms of the d and q current references. The description of the actual algorithm implemented inside the software program will be discussed in 5 where the equations presented in this chapter are executed in code form.

Given the use of the generalized Unified Model and because the rotor currents do not directly contribute to the torque delivery, it is possible to conduct the whole FOC algorithm using just the stator reference frame, thus assuming the IM as a particular case of the general approach. In practice it means that any machine can just be described by the 2x2 inductance matrix more commonly associated to the Synchronous Machine. The only minor difference will be highlighted in the implementation chapter 5 and relates to the fact that the influence of the rotor flux on the IM stator current calculation is significant. Indeed, this interaction introduces a slower stator response and, for dynamic reasons, it is better to introduce a minor parameter variation to account for that.

4.0.1 FOC Magnetic Saturation Literature

As for the machine equations, the inclusion of the magnetic saturation in the FOC control scheme of electrical machines is not very common as of now and it is certainly never been attempted for a generalized machine by using experimental flux characterization together with linear approximation around a pivot point.

Moreover, the definition used in this work that calls a saturation approach *nonlinear*, brings some confusion since there many articles that use the word nonlinear to define the control method and not the machine equation, like [28] [29] [30] [31] for Induction Machine, and [32] [33] for Synchronous Machine.

Some approaches for IM use finite elements for the saturation definition of the inductances and explore only the MTPA region [34], while others like [35] and [36] derive and use the three-phase flux characteristic and deliver a very complex control strategy without approximation. On the other hand, [37] approximates the flux using analytical expressions and not directly a linearization around an experimental point like in the presented work. A simpler approximation is presented in [38], but again uses an analytical formulation and a power approach to be used in automotive efficiency calculations.

Regarding SM, [39] and [40] present different approaches to measure and calculate the saturated inductances for the machines, but they are not focused on the control side. [41] make use of derivative inductances, but just to account for saturation in the saliency portion of the matrix. While [42] uses a similar approach, its control is based on a minimum power solution. Like for IM, for SM [43] presented the measurements and definition of derivative inductances, but not their use in a control algorithm.

It has been found that [44] and [45] make use of derivative inductances, although using analytical models, for control applications very similar to the one presented here, but the research did not extend beyond MTPA region.

As it can be surmise, many researchers adopted the general definition of inductance as derivative of the flux to account for the magnetic saturation in search of better performance. The most advanced solutions try to use analytical characterization, but end up with very complex control algorithms which, moreover, are valid just for specific machines.

In the approach developed during this research, the use of experimental characteristics ensures the correct modelization of the states against the real machine, while the linearization method keeps the complexity at a reasonable level, especially if combined with lower execution frequency of the parameters evaluation.

4.1 OPERATING POINTS

The FOC algorithm main objective is to deliver the requested torque given the current and voltage limits derived from the machine state. It also should provide the best efficiency by using the least current. This behaviour is summarized in a coincise way by introducing loci on the d-q plane that describe these limits and machine states. Although the formulations may differ between linear and nonlinear machines, the meaning and construction of this loci is basically the same.

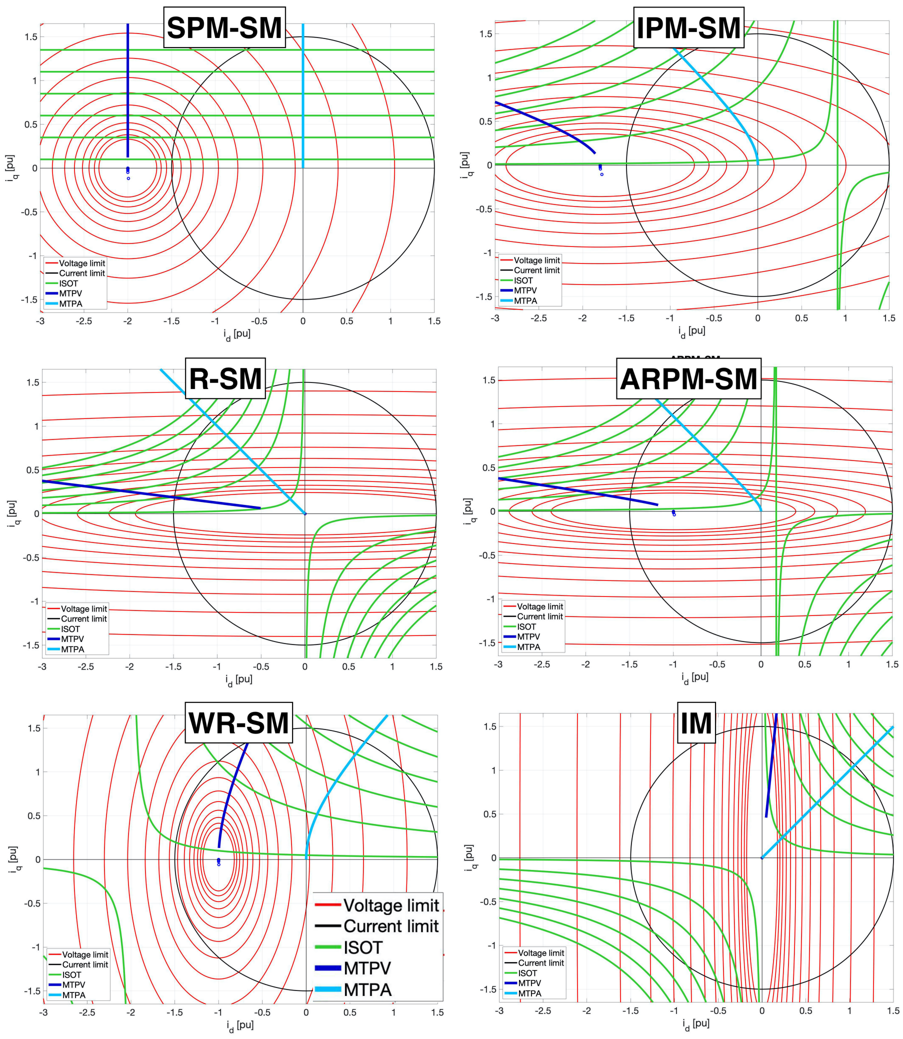


Figure 4.1: *Overview of the limits and operating points for various types of linear electric machine.*

The d-q plane commonly used is the one of the currents, onto which the current limit for linear machines is a circle of radius the maximum current magnitude I, as expressed in (4.1).

$$I^2 = i_d^2 + i_q^2 \tag{4.1}$$

On the other hand, the voltage derived from the currents makes use of the machine parameters and therefore its limit is, in the general case, an ellipse centred at the rotor excitation flux value with semi-axes proportional to the inductances, following the linear expression (4.2). This implies that the voltage limit is function of the electrical speed of the machine and thus produces a variable operating window.

$$\left(\frac{V}{\omega_e}\right)^2 = (L_d i_d + \varphi_e)^2 + (L_q i_q)^2 \tag{4.2}$$

In Fig. 4.1 these limits are shown in black and red respectively for various types of linear machines. The machine must be operated inside both of those limits.

The use of an ellipse can be considered as a significant increase in complexity of the calculation. Thankfully it is possible to lessen the execution by moving into a different reference system. In this case if the flux d-q reference frame is used, then the voltage limit becomes a circle, thus reducing the complexity. The use of the two references will be shown in parallel in 4.2 to highlight the similarities and justify its use.

For this work, it is important to define and think of the torque as equal-value curves on the d-q plane. These curves are called iso-torques or ISOT and, for example, for linear IPM-SM they are hyperboles centred at the permanent magnet flux value. The derivation of such curves starts by the torque equation (3.16). It can be manipulated to find the current of one axis from the other one and it produces this type of relation depending on the machine parameters. In Fig. 4.1 the iso-torque curves for various types of machine are presented in green.

Another important curve is the locus of the plane where a certain torque value is delivered using the minimum current. This curve is called Maximum Torque per Ampere or MTPA and it is constructed by finding the minimum radius of the current circle that intersects a given torque curve. The same principle can be applied to the voltage limit, thus drawing the Maximum Torque per Volt or MTPV curve. In Fig. 4.1, as before, this curves are shown in light blue and blue respectively.

Their formal derivation will be discussed later, but they both are consequences of finding the single intersection between the ISOT curves and the current or voltage limit respectively. Briefly, this is achieved by imposing the same derivative, as (4.3) shows for the MTPA.

$$\frac{\frac{\partial T}{\partial i_d}}{\frac{\partial T}{\partial i_q}} = \frac{\frac{\partial I}{\partial i_d}}{\frac{\partial I}{\partial i_q}} \tag{4.3}$$

Assuming a testing run with a torque ramp followed by a speed ramp, the optimal torque delivery is achieved by following the MTPA curve until the speed is high enough that the voltage limit makes it impossible to stay on that curve. At this point it is usually still possible to deliver the requested torque by moving along the ISOT curve. This is usually achieved by changing the d component of the current to weaken the machine flux. This operation is therefore called flux-weakening. It then may happen that the voltage limit becomes so stringent that the control cannot supply enough current to produce the desired torque. At this point, the algorithm must at least deliver the maximum torque possible with that voltage level, thus following the MTPV curve.

What has been discussed so far is true for linear machines, but it is not completely different from what is needed for nonlinear ones. The difference is that in the latter case the parameters are not constants and some limit formulations must be revised to ensure their validity in magnetic saturation region.

An example of such a difference is shown in Fig. 4.2 where these loci are shown at the same torque request for the same IPM-SM. The linear ones are calculated using approximated constant parameters that are derived from the flux maps chosen in the linear region of the map. The non-linear ones come from a brute-force approach by sweeping the entire flux maps and selecting the desired points. This method generates the correct curves and it has been used to prove that the more analytical method proposed can control machines in saturation.

From Fig. 4.2 it is clearly shown how the increase in q current after a certain point no longer translate in the same increase in torque because the current no longer produces an increase in flux. For this reason, higher torque must be found by increasing the d current, which, having a longer magnetic path in air, has a higher threshold for saturation.

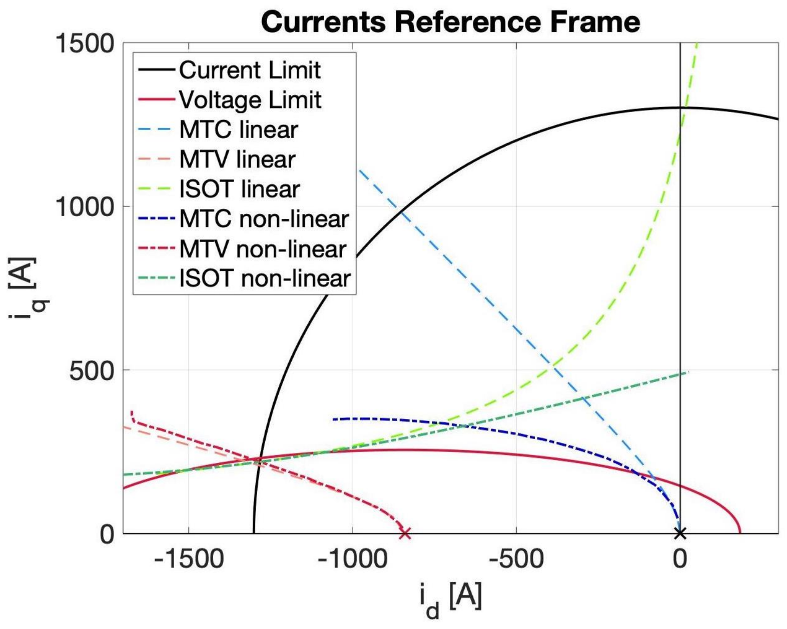


Figure 4.2: *Example of operating point difference on a IPM-SM with and without magnetic saturation.*

4.2 LINEAR IMPLEMENTATION

The overall calculation flow of the FOC block for linear machines starts from the torque request input and the generation of the d current reference corresponding to the MTPA operation point. Afterwards this value is reduced according to the appropriate flux weakening strategy based on the voltage restriction that usually requires first the use of the ISOT curve and then the MTPV curve. The current limit is applied in following functions and finally the q current reference definition can be evaluated accordingly.

The following description assumes a linear machine, which means that the inductances can be considered constants, and a negligible machine resistance.

Other than the machine parameters, the function has as main inputs:

- V_{DC}: the DC voltage;
- $v_{d_{reg}}$ and $v_{q_{reg}}$: the voltages outputs of the current regulators;
- ω_e: the electric speed;
- $\varphi^e_{sdn} = \varphi_{d_{IN}}$: the stator d flux value of the machine in the active flux reference frame, which can be called the input d flux;

- $i_{d_{\mathtt{MTPA}}}$: the d current based on the MTPA calculation done in the previous function;
- $i_{q_{\mathtt{req}}}$: the requested q current from the previous cycle.

And the most important outputs are:

- $i_{d_{\mathtt{ref}}}$: the d current reference;
- $i_{q_{\mathtt{trqMAX}}}$: the q current that would deliver the maximum torque with that d current reference without any other limit;
- $i_{q_{\mathtt{reqMAX}}}$: the maximum value that the q current can assume given the d current reference and the voltage limit;

4.2.1 Flux Reference Frame and Transformations

The algorithm makes heavy use of the flux reference frame to simplify the calculations and because the real value that this function wants to control is the machine flux.

This is not the commonly used reference frame in literature, so it can be a little disorienting at first, but the limits and operating points deriving from it are qualitatively very similar. The most important change for this implementation is that the voltage limit, which is usually an ellipse in the current reference frame, is a simple circle in the flux reference frame. Similarly, the current limit, which is a circle, in the flux reference frame is an ellipse.

Other important definition is the iso-torque curves ISOT that are hyperboles in both reference frames. MTPA and MTPV curves also maintain very similar shapes.

The equations and proofs presented in this chapter are of course completely general for any type of conventional machine, both Synchronous and Induction, unless specifically noted. However, for easier understanding, it is very convenient to have graphical representation of the operations made on an example machine, especially because of the use of different reference frames.

It is therefore defined such a example machine as a Internal Permanet Magnet Synchronous Machine with these key parameters:

- $L_d = 0.3\,[mH]$ d axis inductance;
- $L_q = 1.0\,[mH]$ q axis inductance;
- $\varphi_e = 0.23\,[Wb]$ permanent magnets flux;
- $I = 1300\,[A]$ max stator current.

Assuming steady state, it is therefore easy to write the generic transformations between flux and current reference frames by writing the generalize flux expression (3.8) to find either the fluxes or the currents with each other. (4.4) shows on the left the fluxes calculation in the current reference frame and on the right the opposite for the currents.

$$\begin{aligned} \varphi_d &= L_d i_d + \varphi_e & \qquad i_d &= \frac{\varphi_d - \varphi_e}{L_d} \\ \varphi_q &= L_q i_q & \qquad i_q &= \frac{\varphi_q}{L_q} \end{aligned} \tag{4.4}$$

It follows that the current limits in the two reference frames become (4.5).

$$I^2 = i_d^2 + i_q^2 \qquad\qquad I^2 = \left(\frac{\varphi_d - \varphi_e}{L_d}\right)^2 + \left(\frac{\varphi_q}{L_q}\right)^2 \tag{4.5}$$

It is plain to see in the left part of (4.5) that the current limit is a circle of radius I centred in the axes origin in the current reference frame. On the right side, in the flux reference frame, it is possible to derive that the current limit is an ellipse of d axis $L_d I$ and q axis $L_q I$, centred in $[\varphi_e\ , 0]$.

Similarly as the current limit, the respective equations can be written for the voltage limit, as in (4.6).

$$\left(\frac{V}{\omega_e}\right)^2 = (L_d i_d + \varphi_e)^2 + (L_q i_q)^2 \qquad\qquad \left(\frac{V}{\omega_e}\right)^2 = \varphi_d^2 + \varphi_q^2 \tag{4.6}$$

At this point, the reason to use the flux reference frame starts to become clear since the voltage limit in the this frame (right equation) is a simple circle of radius V/ω_e centered in the origin, while in the current reference frame the ellipse has a d axis value of $\frac{V}{\omega_e L_d}$, a q axis value of $\frac{V}{\omega_e L_q}$ and the center in $[-\frac{\varphi_e}{L_d}\ , 0]$.

Fig. 4.3 shows the same the limits in the two reference frames for the example machine and a relatively high speed.

As a note, the voltage is linked to the flux via the electrical speed, so that the voltage limit can also be considered as the flux limit. In this way, instead of the maximum stator voltage, the variable *stator max flux* (φ_{SM}) can be introduced, which depends on the speed and in the flux reference frame corresponds to the radius of the voltage limit circle.

$$\varphi_{SM} = \frac{V}{\omega_e} \tag{4.7}$$

With this consideration, unless specified, the terms voltage and flux can be used interchangeably in this explanation.

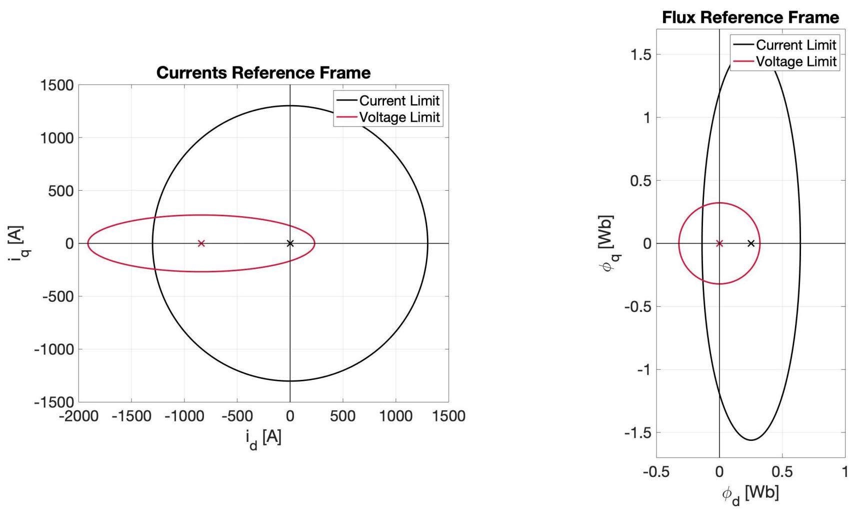


Figure 4.3: *Linear current and voltage limits in the current and flux reference frames.*

4.2.2 Linear Operating Points Calculation

In the classical FOC control implementation, the operating point of the electrical machine is decided based on three factors with the following priority:

1. produce the requested torque;
2. use the minimum current possible;
3. use the minimum voltage possible.

The collection of all the operating points that produce the requested torque in the current or flux reference frame is called the iso-torque curve, or `ISOT`. To minimize the current used, the Maximum Torque per Ampere, or `MTPA`, is derived to indicate for every current value the i_d and i_q that produce the maximum torque. Similarly, the Maximum Torque per Volt (Flux), or `MTPV`, indicates the maximum value of torque that the machine can produce with a given flux module.

The `ISOT` is found solving the generic torque equation (4.8) for a given torque value in one of the two variables substituting the flux definitions (4.4). The result in the current reference frame (4.9), assuming the machine is linear, describes a hyperbole.

$$
\begin{aligned}
T &= \frac{3}{2}p_p\Big(\varphi_d i_q - \varphi_q i_d\Big) && (4.8)\\
\frac{T}{\frac{3}{2}p_p} &= (L_d i_d + \varphi_e)i_q - L_q i_q i_d\\
K_1 &= (L_d - L_q)i_d i_q + \varphi_e i_q\\
i_q &= \frac{K_1}{(L_d - L_q)i_d + \varphi_e} && (4.9)
\end{aligned}
$$

The same can be derived in the flux reference frame, yielding to equation (4.10).

$$
\begin{aligned}
T &= \frac{3}{2}p_p\Big(\varphi_d i_q - \varphi_q i_d\Big)\\
\frac{T}{\frac{3}{2}p_p} &= \varphi_d\frac{\varphi_q}{L_q} - \varphi_q\frac{\varphi_d - \varphi_e}{L_d}\\
\frac{T}{\frac{3}{2}p_p}L_d L_q &= L_d\varphi_d\varphi_q - L_q\varphi_q(\varphi_d - \varphi_e)\\
\varphi_q &= \frac{K_2}{(L_d - L_q)\varphi_d + L_q\varphi_e} && (4.10)
\end{aligned}
$$

The MTPA curve is obtained by finding the closest point of the ISOT to the center of the current limit, which in the current reference frame is a circle. Another way of seeing it is to find the current limit so that it touches the ISOT for the given torque value in only one point, which also means that the two derivatives (gradients) at that point are the same, condition (4.11).

$$
\frac{\frac{\partial T}{\partial i_d}}{\frac{\partial T}{\partial i_q}} = \frac{\frac{\partial I}{\partial i_d}}{\frac{\partial I}{\partial i_q}} \qquad (4.11)
$$

With:

$$
\begin{aligned}
T(i_d, i_q) &= \frac{3}{2}p_p\cdot\Big((L_d - L_q)i_d i_q + \Phi_e i_q\Big)\\
I(i_d, i_q) &= \sqrt{i_d^2 + i_q^2}
\end{aligned}
$$

Deriving T and I and manipulating the generic condition (4.11), the passages yield to the definition of (4.12).

$$\frac{\frac{3}{2}p_p(L_d - L_q)i_q}{\frac{3}{2}p_p\Big((L_d - L_q)i_d + \varphi_e\Big)} = \frac{\frac{-i_d}{\sqrt{i_d^2+i_q^2}}}{\frac{-i_q}{\sqrt{i_d^2+i_q^2}}}$$

$$\frac{(L_d - L_q)i_q}{(L_d - L_q)i_d + \varphi_e} = \frac{i_d}{i_q}$$

$$(L_d - L_q)i_d^2 + \varphi_e i_d - (Ld - Lq)i_q^2 = 0 \tag{4.12}$$

Solving for i_d and applying the transformations to the flux reference frame yields to (4.13) and (4.14) which present the MTPA equations in both planes.

$$i_{d_{\mathrm{MTPA}}} = \frac{\sqrt{\varphi_e^2 + (2(L_d - L_q)i_q)^2} - \varphi_e}{2(L_d - L_q)} \tag{4.13}$$

$$\varphi_{d_{\mathrm{MTPA}}} = \frac{L_d\sqrt{\varphi_e^2 + (2(L_d - L_q)\frac{\varphi_q}{L_q})^2} - L_d\varphi_e}{2(L_d - L_q)} + \varphi_e \tag{4.14}$$

The MTPA implementation in the algorithm is executed using directly (4.13) with input the q current request $i_{q_{\mathrm{req}}}$ of the previous cycle. Later this value will be refined based on the flux weakening and limitations.

The same approach can be carried out to find the MTPVcurves, but starting from the flux reference frame since in it the voltage limit is a circle. Thus, (4.11) becomes (4.15).

$$\frac{\frac{\partial T}{\partial \varphi_d}}{\frac{\partial T}{\partial \varphi_q}} = \frac{\frac{\partial \varphi_{SM}}{\partial \varphi_d}}{\frac{\partial \varphi_{SM}}{\partial \varphi_q}} \tag{4.15}$$

With:

$$T(\varphi_d, \varphi_q) = \frac{3}{2}p_p\frac{1}{L_dL_q} \cdot \Big((L_d - L_q)\varphi_d\varphi_q + L_q\varphi_e\varphi_q\Big)$$

$$\varphi_{SM}(\varphi_d, \varphi_q) = \sqrt{\varphi_d^2 + \varphi_q^2}$$

Deriving T and φ_{SM} and manipulating the generic condition (4.15), the passages yield to the definition of (4.16).

$$\frac{\frac{3}{2}p_p\frac{1}{L_dL_q}(L_d - L_q)\varphi_q}{\frac{3}{2}p_p\frac{1}{L_dL_q}\Big((L_d - L_q)\varphi_d + L_q\varphi_e\Big)} = \frac{\frac{-\varphi_d}{\sqrt{\varphi_d^2+\varphi_q^2}}}{\frac{-\varphi_q}{\sqrt{\varphi_d^2+\varphi_q^2}}}$$

$$\frac{(L_d - L_q)\varphi_q}{(L_d - L_q)\varphi_d + L_q\varphi_e} = \frac{\varphi_d}{\varphi_q}$$

$$(L_d - L_q)\varphi_d^2 + L_q\varphi_e\varphi_d - (Ld - Lq)\varphi_q^2 = 0 \tag{4.16}$$

As before for the MTPA, solving for φ_d and applying the transformation in the current reference frame produces (4.18) and (4.17), the MTPV equations in both reference frames.

$$i_{d_{MTPV}} = \frac{\sqrt{(L_q\varphi_e)^2 + (2(L_d - L_q)L_q i_q)^2} - L_q\varphi_e}{2L_d(L_d - L_q)} - \frac{\varphi_e}{L_d} \tag{4.17}$$

$$\varphi_{d_{MTPV}} = \frac{\sqrt{(L_q\varphi_e)^2 + (2(L_d - L_q)\varphi_q)^2} - L_q\varphi_e}{2(L_d - L_q)} \tag{4.18}$$

Using the limits of Fig. 4.3, the operating point curves described are shown together in Fig. 4.4 for a torque request of 1850 Nm and zoomed around the III quadrant in both reference frames.

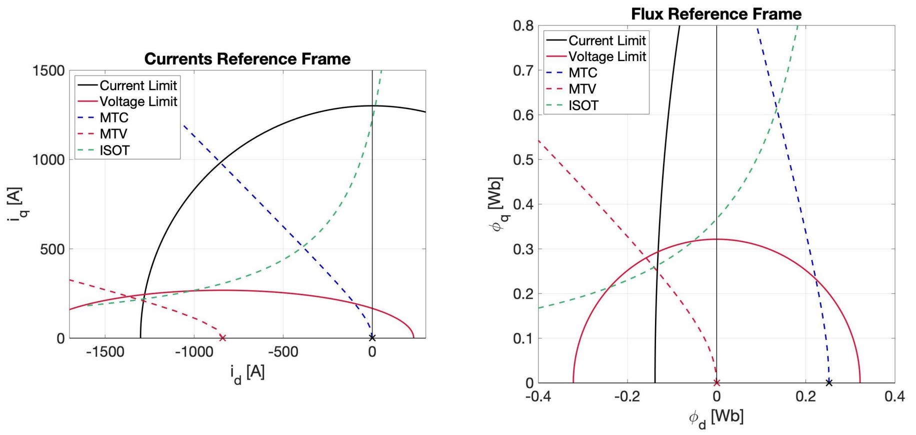


Figure 4.4: *Current and voltage limits, MTPA and MTPV operating point curves in the flux and current reference frames for linear machine.*

As a quick preview, Fig. 4.5 shows how the operating point curves change introducing flux saturation with comparison to the same case with linear machine. The magnetic saturation mainly impacts the q axis, so that the machine is less capable to produce flux in that direction. It therefore needs to use more d flux to produce the same torque as the linear machine.

4.2.3 Flux Weakening Strategy

From the operating curves point of view, the flux weakening region of the machine operation is the interaction between the voltage limit and the iso-torque curve, at least until there is such intersection. Finding the optimal operating point is equivalent in the two reference frames, thanks to the reasons nd the transformations (4.4) described in the previous section. However, dealing

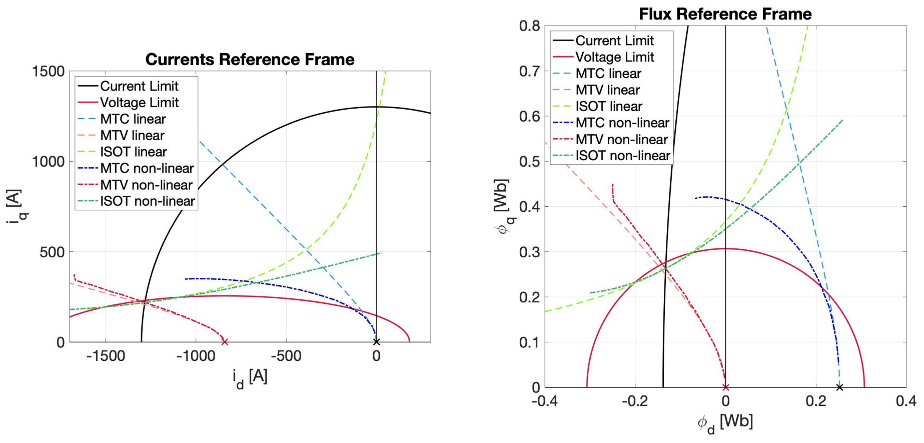


Figure 4.5: *Comparison between linear and non-linear machines operating point curves in the two reference frames.*

with a circle voltage limit instead of an ellipse is much simpler and, in the case of limited calculation resources and speed focus of high performance systems, the development of the flux weakening in the flux reference frame brings advantages in both aspects.

The goal of the flux weakening function is to find the intersection between the voltage limit and the iso-torque knowing the current state of the machine. Since the two curves are both 2^{nd} order equations, the solver system would be a 4^{th} order equation. Solving such problems can be usually done in two ways: direct solution and numerical approximation. Both solution present pros and cons, but they are both very heavy in terms of calculations and for this reason an alternative solution is proposed to approximate the solution.

The idea is to use a linear approximation of the ISOT curve around the operating point, so that the intersection of a line with a circle is a much simpler system that can be solve analytically with few calculations.

To find the best line approximating the torque, the first solution proposed involves the use of the maximum MTPV point for the current speed and the maximum torque point as the two points through which the line is constructed and then translated to the actual operating point. The second solution is by using the derivative in that point, and it is presented in the next section 4.2.4.

Following is the step-by step description of the algorithm. The figures show the results in both reference frames, but all the calculations are, as in the implemented function, just in the flux reference.

First of all, the DC voltage, the speed and the regulators voltage outputs ($v_{d_{\text{reg}}}$ and $v_{q_{\text{reg}}}$) are used to find the voltage limit, which means the radius φ_{SM}. Compared to the previously mentioned equation (4.7), the regulator's outputs are used to better estimate the actual machine flux by using the regulators to account for the machine resistance and the dynamic of the system. After a low-pass filter for more robustness, their module is subtracted to the stator maximum voltage and the result divided by the electrical speed to obtain the maximum stator flux φ_{SM}, see (4.19).

$$\varphi_{\text{SM}} = \frac{V - \sqrt{(v_{d_{\text{reg}}}|_{\text{lowpass}})^2 + (v_{q_{\text{reg}}}|_{\text{lowpass}})^2}}{\omega_e} \tag{4.19}$$

Then a couple of recurring constants are calculated from the machine parameters in equations (4.20) and (4.21). This is done to avoid multiple execution of the same values.

$$K = \frac{L_q \varphi_e}{L_d - L_q} \tag{4.20}$$

$$K_4 = \frac{K}{4} = \frac{L_q \varphi_e}{4(L_d - L_q)} \tag{4.21}$$

Knowing the voltage limit and the MTPV it is possible to find the intersection, thus obtaining the maximum torque point for the given flux φ_{SM}. This point, defined as ($\varphi_{d0_{\text{MTPV}}}$, $\varphi_{q0_{\text{MTPV}}}$), is calculated with (4.22) and is plotted in Fig. 4.6. For this example, the speed chosen is 3000 rpm.

$$\begin{aligned} \varphi_{d0_{\text{MTPV}}} &= -\sqrt{\left(\frac{\varphi_{\text{SM}}}{\sqrt{2}}\right)^2 + K_4^2} - K_4 \\ \varphi_{q0_{\text{MTPV}}} &= \sqrt{\varphi_{\text{SM}}^2 - \varphi_{d0_{\text{MTPV}}}^2} \end{aligned} \tag{4.22}$$

At the same time, it is possible to plot the input flux operating point which lays on the ISOT and has coordinates ($\varphi_{d_{\text{IN}}}$, $\varphi_{q_{\text{req}}}$). The q component comes from the previous cycle q current request $i_{q_{\text{req}}}$, transformed in flux request. The use of the request instead of the actual q axis input flux is done to slightly anticipate the torque request and have a more stable working point, but the difference is negligible in the contest of this explanation. Fig. 4.7 shows the input point, chosen for this example at $\varphi_{d_{\text{IN}}} = 0.02\,[Wb]$.

Knowing the input point and the max torque per flux point, it is possible to find the maximum q flux to obtain the maximum torque $\varphi_{q_{\text{MAX}}}$, since the ISOT curve passing through the max flux point is the maximum value of the torque available for the given

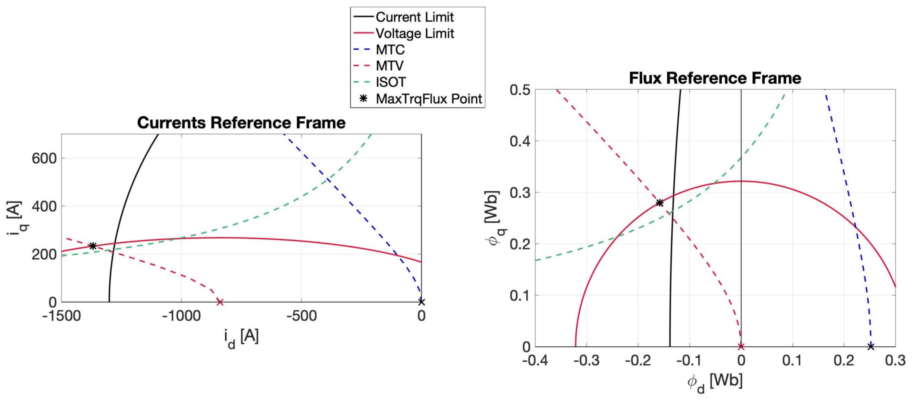


Figure 4.6: *Maximum Torque per Flux point at given speed.*

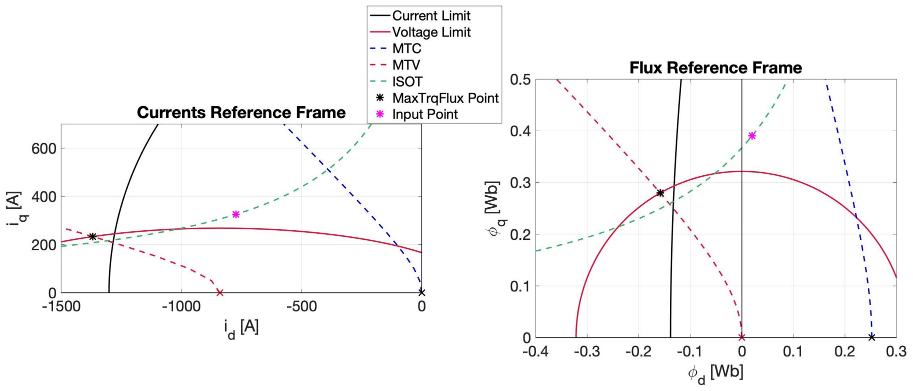


Figure 4.7: *Plot of the example input operating point on the* `ISOT`*curve.*

speed. In Fig. 4.8 both the `ISOT` curve and the maximum point are shown, even if for the algorithm only the point is required through equation (4.23).

$$\varphi_{q_{\mathtt{MAX}}} = \frac{\varphi_{q0_{\mathtt{MTPV}}}(\varphi_{d0_{\mathtt{MTPV}}} + K)}{\varphi_{d_{\mathtt{IN}}} + K} \tag{4.23}$$

At this point, the line passing through ($\varphi_{d_{\mathtt{IN}}}$, $\varphi_{q_{\mathtt{MAX}}}$) and ($\varphi_{d0_{\mathtt{MTPV}}}$, $\varphi_{q0_{\mathtt{MTPV}}}$) is defined to use as torque approximation. In the function, only the slope is necessary for the next steps because it will be translated. It is found using the classical formulation $m = \frac{\Delta y}{\Delta x}$. Moreover, the division, which is a very resource consuming operation on a DSP, can be avoided at this time and the numerator and denominator expressed separately, like in (4.24). For plotting purposes, the complete line is presented in Fig. 4.9.

$$\begin{aligned} \mathtt{num} &= \varphi_{q_{\mathtt{MAX}}} - \varphi_{q0_{\mathtt{MTPV}}} \\ \mathtt{den} &= \varphi_{d_{\mathtt{IN}}} - \varphi_{d0_{\mathtt{MTPV}}} \end{aligned} \tag{4.24}$$

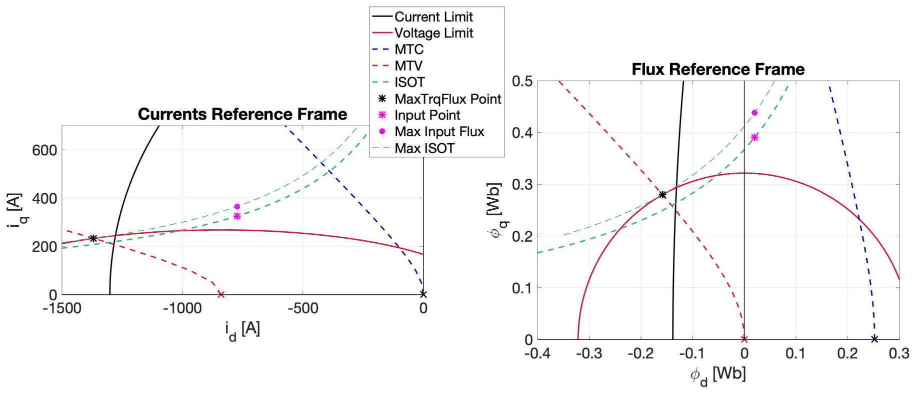


Figure 4.8: *Maximum* q *axis flux at the input condition with maximum* ISOT*curve.*

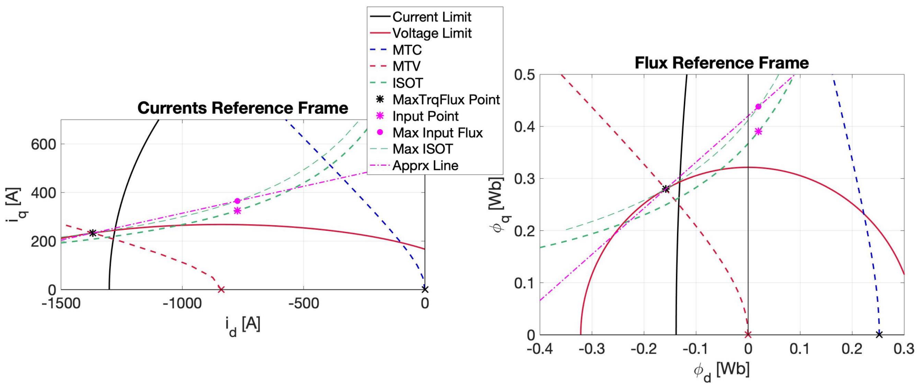


Figure 4.9: *Line construction of the torque approximation.*

The next step is to take the line found and use it at the input point $\varphi_{d_{IN}}$ to find the intersection with the voltage limit. Fig. 4.10 shows the translation of such line.

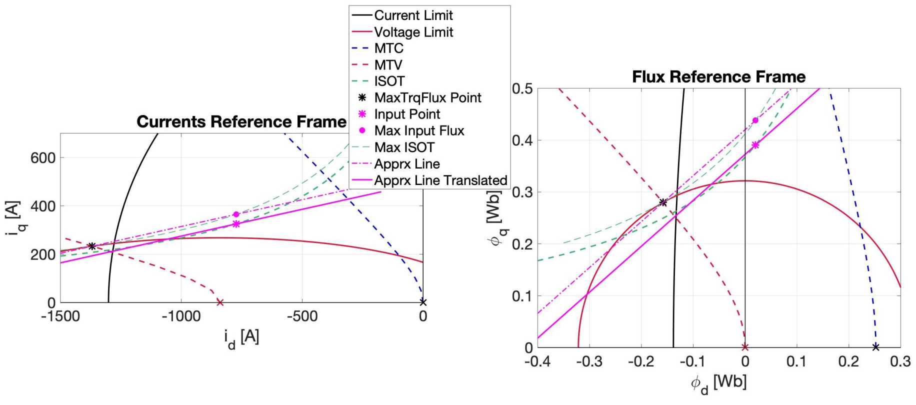


Figure 4.10: *Translation of torque approximation to input point* $\varphi_{d_{IN}}$.

The simple system in (4.25) between a line and a circle is used to solve for the intersection.

$$\begin{cases} r^2 = x^2 + y^2 \\ y = mx + q \end{cases} \tag{4.25}$$

And it leads to (4.26).

$$\begin{aligned} (m^2+1)x^2 + 2mqx + q^2 - r^2 = 0 \\ x = \frac{-mq + \sqrt{(m^2+1)r^2 - q^2}}{m^2+1} \end{aligned} \tag{4.26}$$

The positive square root is always selected because the point wanted is the right-most one on the circunference for every possible case. Substituting the placeholders with the actual values

$$\begin{aligned} r &= \varphi_{\mathtt{SM}} \\ q &= \varphi_{q_{\mathtt{req}}} - m\varphi_{d_{\mathtt{IN}}} \end{aligned}$$

it is possible (4.27) to find the d axis value of the intersection $\varphi_{d_{\mathtt{ref}}}$, which is the flux value to impose the machine for flux weakening.

$$\varphi_{d_{\mathtt{ref}}} = \frac{\varphi_{d_{\mathtt{IN}}} - m\varphi_{q_{\mathtt{req}}} + \sqrt{m^2\varphi_{\mathtt{SM}}^2(m^2+1) - m^2(\varphi_{d_{\mathtt{IN}}} - m\varphi_{q_{\mathtt{req}}})^2}}{m^2+1} \tag{4.27}$$

At this point the algorithm has found the optimal operating point and the function ends with a regulator that produces the correct d current reference ($i_{d_{\mathtt{ref}}}$) from the error of the input d flux and the reference flux. The regulator output is saturated to the value of the MTPA current calculated in the previous function. In this way the control uses the MTPA value until the voltage limit allows it.

The function does calculate explicitly the second coordinate of the intersection point using the modified voltage limit equation (4.28) to calculate the actual maximum q flux reference available $\varphi_{q_{\mathtt{refMAX}}}$. Fig. 4.11 shows the output operation point reference for the control produced by the flux weakening algorithm here described.

$$\varphi_{q_{\mathtt{refMAX}}} = \sqrt{\varphi_{\mathtt{SM}}^2 - \varphi_{d_{\mathtt{ref}}}^2} \tag{4.28}$$

Additionally, this algorithm can also calculate the maximum torque (4.29) that the machine could deliver without the torque request, limited only by the voltage. This is the same as (4.23), but using the new d flux output, $\varphi_{d_{\mathtt{ref}}}$. Fig. 4.12 presents this last point.

$$\varphi_{q_{\mathtt{trqMAX}}} = \frac{(\varphi_{q0_{\mathtt{MTPV}}}(\varphi_{d0_{\mathtt{MTPV}}} + K)}{\varphi_{d_{\mathtt{ref}}} + K} \tag{4.29}$$

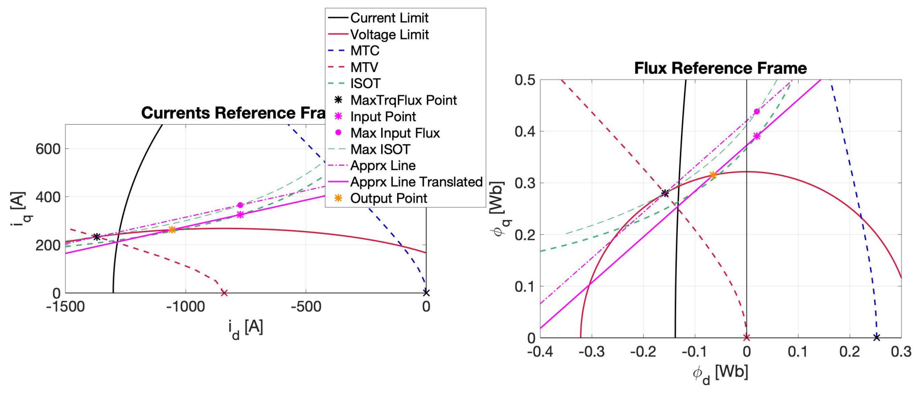


Figure 4.11: *Intersection and operating point output of the function.*

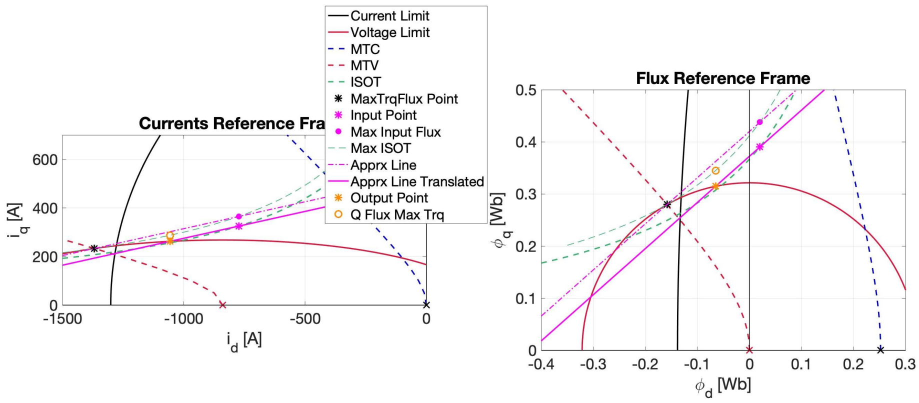


Figure 4.12: *Value of* q *flux that could deliver maximum torque in the new operating point.*

Both $\varphi_{q_{\mathtt{refMAX}}}$ and $\varphi_{q_{\mathtt{trqMAX}}}$ are then transformed into the current values $i_{q_{\mathtt{reqMAX}}}$ and $i_{q_{\mathtt{trqMAX}}}$ respectively, which are the other two outputs of the function. The former is used to limit the q current reference derived from the torque request and the latter is used to calculate it. In fact, given the d current, the q current reference is proportional to the maximum torque request.

Lastly, Fig. 4.13 shows a zoomed version of the final result.

4.2.4 Variation Using Torque Derivative

The other approach that can be used is to use another line to approximate the torque hyperbole. Given the input operating point ($\varphi_{d_{\mathtt{IN}}}$, $\varphi_{q_{\mathtt{req}}}$) that lays on the torque curve, the tangent through it is the line used for the intersection.

This method requires fewer steps since, given the voltage circle and the input point, there is no need to calculate additional points.

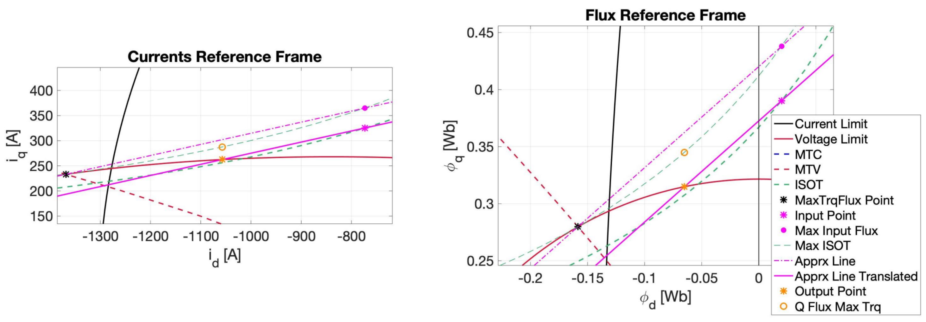


Figure 4.13: *Detail of the final operating point output with the geometric construction.*

From the torque equation (4.8) in the current reference frame, the torque function in the flux can be obtained as in (4.30) with highlighted some small simplification to group the constants.

$$\begin{aligned}
T &= \frac{3}{2}p_p\left(\varphi_d i_q - \varphi_q i_d\right) \\
\frac{2T}{3p_p} &= \left(\varphi_d\frac{\varphi_q}{L_q} - \varphi_q\frac{\varphi_d-\varphi_e}{L_d}\right) \\
\frac{2T}{3p_p}L_dL_q &= L_d\varphi_d\varphi_q - L_q\varphi_d\varphi_q + L_q\varphi_e\varphi_q \\
K_T &= (L_d - L_q)\varphi_d\varphi_q + L_q\varphi_e\varphi_q \\
f_T(\varphi_d,\varphi_q) &= (L_d - L_q)\varphi_d\varphi_q + L_q\varphi_e\varphi_q - K_T \qquad (4.30)
\end{aligned}$$

From (4.30), the generalized equation for the slope of the tangent (derivative) given a point on the curve is the one given in (4.31) starting from the torque function f_T previously found.

$$m(\varphi_{d0},\varphi_{q0}) = \frac{-\frac{\partial f_T}{\partial \varphi_d}}{\frac{\partial f_T}{\partial \varphi_q}} = \frac{-(L_d - Lq)\varphi_{q0}}{(L_d - L_q)\varphi_{d0} + \varphi_e L_q} \qquad (4.31)$$

In the case under study, given the input point, the slope (4.31) becomes (4.32);

$$m = \frac{-(L_d - Lq)\varphi_{q_{req}}}{(L_d - L_q)\varphi_{d_{IN}} + \varphi_e L_q} \qquad (4.32)$$

No translation is required as for the case described before and the intersection between this line and the voltage circle is the same as before to find the reference d flux, (4.27). From here, however, it is a little more complicated to calculate the maximum torque available without re-calculating the intersection between circle and MTPV.

Fig. 4.14 shows the result of the use of the tangent compared to the previous case.

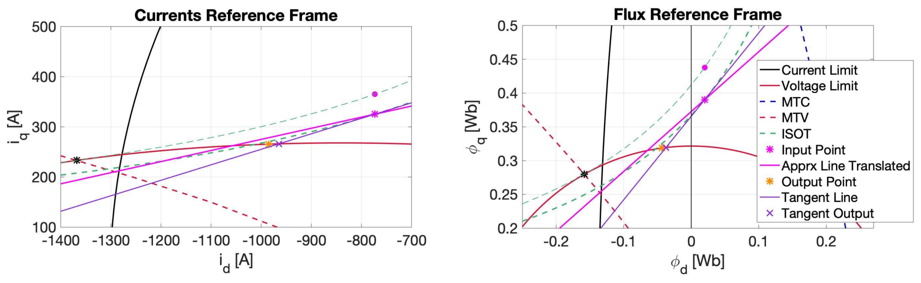


Figure 4.14: *Result using derivative calculation compared to previous case.*

4.3 NONLINEAR IMPLEMENTATION

As previously mentioned, the general idea behind the solution of the reference problem in the FOC for nonlinear machines is quite similar to the linear one. In a way, the curves that are involved are still of an high enough order that direct solution of intersection is not feasible with regular DSPs, so approximations have to take place. Additionally, though, the equations are more complex since the inductance matrix is now a full matrix and the values are no longer constants.

The use of the flux reference frame is still a key point of the solution described, but the transformation is no longer simply (4.4), but the generalized flux equations with linearization approximation (3.48) must be used. Its expanded form is again presented in (4.33) as a system of equations.

$$\begin{cases} \varphi_d = L_{dd}(i_d - i_{od}) + L_{dq}(i_q - i_{oq}) + \varphi_{od} \\ \varphi_q = L_{qd}(i_d - i_{od}) + L_{qq}(i_q - i_{oq}) + \varphi_{oq} \end{cases} \tag{4.33}$$

As a reminder, (i_{od}, i_{oq}) is a operating point closely related to the actual machine state from which a linearization of the flux characteristic (and consequently the inductances) is evaluated to remove the time dependance of the machine parameters. It follows that inductances and state fluxes are functions of this point, but not the time.

The overall algorithm will therefore use the same approach as the linear one:

- calculate the MTPA;
- use the actual machine state to evaluate the intersection between voltage limit and the iso-torque curve using the derivative method;

- apply a PI regulator to the reference d flux to obtain the d current reference;
- derive the q reference based on the torque request;
- apply voltage and current limits.

A major difference, on the other hand, is that it has not been found a way to estimate the maximum available torque. This is because the ISOT is not a simple hyperbole anymore, but a complex function that should be solved in its entirety to predict its behaviour. This is no easy task and in the main algorithm is not necessary because it is just needed its derivative in a point.

With regards to the Induction Machine, since it is treated just as a special case of the generic formulation, the parameters to use are the stator ones accounting for the equivalent rotor circuit. The matrix is the same as the SM, where the values are (4.34).

$$\begin{cases} L_{dd} &= L_{sd} \\ L_{dq} &= 0 \\ L_{qd} &= 0 \\ L_{qq} &= \sigma_q L_{sq} \end{cases} \tag{4.34}$$

With $\sigma_q = 1 - \frac{L_m^2}{L_{sq}L_{rq}}$.

Of course, given the different starting parameters, all the flux weakening constants that can be calculated outside the actual function are different from the SM ones, but the equations are the same.

The only small difference introduced is in the calculation of the q current given the torque request and the d current in the function that follows the torque request. Even here, the equation used for the SM would have worked, but oscillations might have occurred because in the induction machine the stator flux does not have the same dynamics as the stator currents due to the inertia of the rotor flux. The change will be described in the relative paragraph, but, in general terms, new parameters are found to keep the rotor flux less variable.

4.3.1 Maximum Torque per Ampere

The first step of the FOC control is to provide the current references to obtain the torque request at the maximum efficiency possible, thus staying on the MTPA curve. Ideally, for every torque value there is a univocal point on the MTPA, thus it could be possible to determine the references of d and q current just from the torque value. However, this method would require the direct solution of quartic equations, which is a heavy task to run in real time.

The algorithm proposed, on the other hand, relays to the base that the new operating point can be derived from the old one, thus enabling the use of approximations because the step is usually very small, reducing the error. The obvious advantage is the avoidance of the quartic solution.

The overall idea is then:

- start from the old operating point;
- move it to the next torque request;
- approximate the `ISOT` in a adequate way (derivative);
- intersect the approximate `ISOT` with the current limit;
- get the new operating point.

As written before, the general torque equation presented in 3.1 as (3.1) is still valid and re-proposed here below (4.35).

$$T = \frac{3}{2} p_p (\varphi_d i_q - \varphi_q i_d) \tag{4.35}$$

The torque is, therefore, a function of both fluxes and currents. Collecting the constants to $T_{\mathtt{norm}} = \frac{2T}{3p_p}$, it is possible to use the flux equation (4.33) to obtain the torque as function of just the currents, as shown in (4.36).

$$\begin{aligned} T_{\mathtt{norm}}(\boldsymbol{i}) = & - L_{qd} i_d^2 + (L_{dd} - L_{qq}) i_d i_q + \\ & + (L_{qq} i_{oq} - \varphi_{oq} + L_{qd} i_{od}) i_d + \\ & + L_{dq} i_q^2 - (L_{dd} i_{od} - \varphi_{od} + L_{dq} i_{oq}) i_q \end{aligned} \tag{4.36}$$

Just as an exercise, it is possible, in the same way, to calculate the torque as function of the fluxes, as shown in (4.37)

$$\begin{aligned} T_{\mathtt{norm}}(\boldsymbol{\varphi}) = & \frac{-L_{qd}}{L_{dd}L_{qq} - L_{dq}L_{qd}} \varphi_d^2 + \frac{L_{dd} - L_{qq}}{L_{dd}L_{qq} - L_{dq}L_{qd}} \varphi_d \varphi_q + \\ & + \left(i_{oq} - \frac{L_{dd}\varphi_{oq} + L_{qd}\varphi_{od}}{L_{dd}L_{qq} - L_{dq}L_{qd}} \right) \varphi_d + \\ & + \frac{L_{dq}}{L_{dd}L_{qq} - L_{dq}L_{qd}} \varphi_q^2 - \\ & - \left(i_{od} - \frac{L_{qq}\varphi_{od} + L_{dq}\varphi_{oq}}{L_{dd}L_{qq} - L_{dq}L_{qd}} \right) \varphi_q \end{aligned} \tag{4.37}$$

The condition used for the Maximum Torque per Ampere in the linear machine (4.11) is the same here, as the torque curve and the current circle must intersect in only one point. The partial derivatives of (4.36) are (4.38) and (4.39) that are also renamed `NumT` and `DenT` for clarity in the next steps.

$$\begin{aligned}\frac{\partial}{\partial i_d}T_{norm} = \texttt{NumT} =& 2L_{qd}i_d - (L_{dd} - L_{qq})i_q + \\ &+ (\varphi_{oq} - L_{qd}i_{od} - L_{qq}i_{oq})\end{aligned} \tag{4.38}$$

$$\begin{aligned}\frac{\partial}{\partial i_q}T_{norm} = \texttt{DenT} =& 2L_{dq}i_q - (L_{dd} - L_{qq})i_d + \\ &+ (\varphi_{od} - L_{dq}i_{oq} - L_{dd}i_{od})\end{aligned} \tag{4.39}$$

The `MTPA` algorithm has as input the requested torque and the previous i_d set point. Given the previous operating point, the new cycle will most likely introduce a new torque set-point, thus the need to re-calculate the optimal `MTPA` point. The new torque request is, in theory, moving the operating point only on the q axis. As a consequence, after the new q current is calculated, the new d request is found from that.

Tests were carried with an algorithm that used directly the `MTPA` equation to find the new i_d, thus solving the quadratic (4.36) (using a square root and a division) and it worked fine for a very large number of cases. However, it did not produce great results when the torque request presents very high change rates. In this case, the method tends to create overshoots and vibrations because it takes a while to converge on the `MTPA` curve from a point far away.

To improve this behaviour, a new algorithm is presented that makes the convergence faster and has the added bonus of not requiring the square root calculation (the division is still present).

Referring to Fig. 4.15, where the torque request has a torque step of $100\,\mathrm{Nm}$, the previous operating point `P1` together with the new torque request `T2` yields to the new request of i_q which is point `P2`. This point is derived from the `ISOT` and equation (4.36) solved for i_q, which is a linearized version also used and explained later in (4.43). The old algorithm would have then used the `MTPA` formulation solved for i_d using this point `P2` to get operating point `P4`. While not wrong, it can be seen that in case of step requests there is an overshoot (if the step is positive) that in the next step will be used as the starting point. The next point will be undershot, and so forth until convergence.

The new algorithm speeds up the process by approximating the `ISOT` curve with its tangent (instead of an horizontal line) and calculating the MTPA of this line, obtaining point P3. It still does not produce the perfect point with just one step, but the convergence is faster especially at high power and it has the added bonus of avoiding the square root. A for-loop has been added to tune the accuracy and efficiency of the finding with great results already with just a couple of iterations.

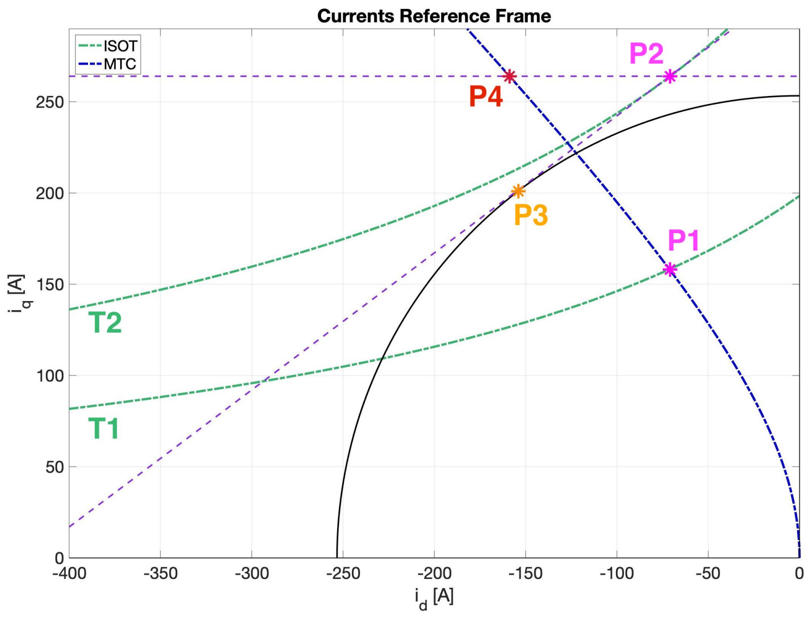


Figure 4.15: *Geometry construction of the MTPA point update with comparison with the old method.*

Starting from the requested torque and the last requested current set point (P2), the new d current request is found as the solution of the system (4.40) between the line passing through P2 and its perpendicular passing through the current circle center (0, 0), the latter being the definition of circle tangent to a line.

$$\begin{cases} i_q = m \cdot i_d + q \\ i_q = -\frac{1}{m} i_d \end{cases} \tag{4.40}$$

From here, given $m = \frac{\mathtt{NumT}}{\mathtt{DenT}}$ and $q = i_q - m \cdot i_d$, the d current value of P2 is derived as (4.41).

$$\begin{aligned}
m \cdot i_{d_{req}} + q &= -\frac{1}{m} i_{d_{req}} \\
(1+m^2) \cdot i_{d_{req}} + mq &= 0 \\
i_{d_{req}} &= \frac{-mq}{1+m^2} \\
i_{d_{req}} &= \frac{m^2 \cdot i_d - m \cdot i_q}{1+m^2} \\
i_{d_{req}} &= \frac{\mathtt{DenT}^2}{\mathtt{NumT}^2 + \mathtt{DenT}^2} \left(\frac{\mathtt{NumT}^2}{\mathtt{DenT}^2} i_d - \frac{\mathtt{NumT}}{\mathtt{DenT}} i_q \right) \\
i_{d_{req}} &= \frac{\mathtt{NumT} \cdot \left(\mathtt{NumT} \cdot i_d - \mathtt{DenT} \cdot i_q \right)}{\mathtt{NumT}^2 + \mathtt{DenT}^2}
\end{aligned} \tag{4.41}$$

Having found the d current, the q one can be found using the torque expression (4.36). To solve for i_q would require the calculation of the quadratic equation, but a simplified solution can be used with very reasonable approximation and much lower operation cost. In fact the torque hyperbole can be linearized around the searched point and the current is found solving (4.42).

$$T^*_{\mathtt{norm}}(\boldsymbol{i}) = \Big(\varphi_{od} - L_{dd} i_{od} + (L_{dd} - L_{qq}) i_d\Big) i_q + (L_{qq} i_{oq} - \varphi_{oq}) i_d \tag{4.42}$$

It follows that $i_{q_{req}}$ is found as (4.43), obviously taking care not to divide by zero.

$$i_{q_{req}} = \frac{(\varphi_{oq} - L_{qq} i_{oq}) i_{d_{req}} + T^*_{\mathtt{norm}}}{\varphi_{od} - L_{dd} i_{od} + (L_{dd} - L_{qq}) i_{d_{req}}} \tag{4.43}$$

This q current request formula (4.43) is the only major difference between IM and SM current reference determination. Although (4.43) is still valid even for IM, the particular dynamic response of this machine has been found to possibly produce oscillation of the torque request in particular cases (at high dynamic requests). This phenomenon was found to be started by the fact that the stator flux is influenced by the rotor flux and the latter has a lower dynamic that delays the stator flux from the stator current measurement.

The solution found requires just the changing of the parameters to simulate more consistent (inside the time step) flux values with respect to the current measured. Specifically, the straight inductances are substituted by sigma inductances $\sigma_d L_{sd}$ and $\sigma_q L_{sq}$, thus including the rotor effect and yielding to (4.44).

$$
\begin{aligned}
i_{q_{req}} &= \frac{(\varphi_{oq} - \sigma_q L_{sq} i_{oq}) i_{d_{req}} + T^*_{\mathtt{norm}}}{\varphi_{od} - \sigma_d L_{sd} i_{od} + (\sigma_d L_{sd} - \sigma_q L_{sq}) i_{d_{req}}} = \\
&= \frac{(\mathtt{Fts}_q) i_{d_{req}} + T^*_{\mathtt{norm}}}{\mathtt{Fts}_d + (\sigma_d L_{sd} - \sigma_q L_{sq}) i_{d_{req}}}
\end{aligned}
\tag{4.44}
$$

With:

$$
\begin{aligned}
\mathtt{Fts}_d &= \varphi_{od} - \sigma_d L_{sd} \cdot i_{od} \\
\mathtt{Fts}_q &= \varphi_{oq} - \sigma_q L_{sq} \cdot i_{oq}
\end{aligned}
\tag{4.45}
$$

A small optimization is achieved by using directly the feedback $i_{d_{req}}$ coming from the flux weakening function to calculate the $i_{q_{req}}$ when the machine is in flux weakening region. This is not strictly necessary, but it improves accuracy and might reduce calculation time if implemented carefully, since it bypasses the for-loop.

Also in this function, the possible saturation of i_d to a minimum value (in absolute terms) can be inserted. This is to ensure a minimum flux in the machine that is a required feature in induction machines for a proper performance, but can be useful in synchronous machines, too. In fact, by forgoing some efficiency, keeping the d current constant, the response of the motor can be improved since the torque will be dependent only on the q current, which has a lower time constant.

4.3.2 Flux Weakening

In the flux weakening region the performance of the machine is limited by the voltage. This means that the optimal operating point must be found on the voltage limit which, in the current reference frame, is an ellipse. As explained, to simplify the calculations, it is useful to use the flux reference frame instead of the current one. This way the voltage limit is a circle and its intersections are easier to implement.

Moreover, voltage and flux are linearly dependent through the electrical speed, so that the voltage limit can be interpreted as a flux limit and the usual voltage limit equation (4.6) can be written as (4.46).

$$
\left(\frac{V}{\omega_e}\right)^2 = \varphi^2_{\mathrm{SM}} = \varphi^2_d + \varphi^2_q \tag{4.46}
$$

Working in the flux reference frame and using the flux as the voltage limit allows to directly generate and control the d flux value as the variable implementing the flux weakening. Thanks to a PI regulator and the estimation of the machine flux, the d current reference can be obtained.

The overall algorithm is divided in these steps, similarly to the linear implementation:

- Obtain the fluxes and verify the signs;
- Approximate the `ISOT` at the requested state by calculating its slope;
- Find the intersection with the voltage (flux) limit;
- Calculate the `MTPV` and limit the flux accordingly, generating the d flux reference;
- Use a PI regulator to produce the appropriate d current;
- Calculate the limitations for later functions.

As for linear machines, the voltage limit value is function of the DC voltage and speed, yielding to the calculation of the *stator maximum flux* (φ_{SM}) used in the algorithm, as shown before in (4.7). Thanks to this dependency, the terms voltage and flux can be used interchangeably in this section.

Given that the flux estimation could be not accurate since it comes from the model integration, a feedback correction is carried out by modifying the maximum flux trying to limit this effect. The feedback comes from the main current regulator outputs. In fact, these regulators already correct the voltage outputs accounting for the resistance drop and the flux miscalculations. Since only the later effect is to be considered, the regulators outputs are passed through a lowpass filter with the machine time constant $\frac{L_{dd}}{r_s}$ which removes the contribution of the resistance.

This outputs a correction voltage vector that, subtracted from the V_{DC} gives a better value of the maximum flux available in that instant. The simple algorithm just uses the magnitude of the vector, but, although very simple, the problem may get worse if the estimated flux is overshot. In fact, the magnitude (without direction) subtraction on a vector produces a greater limitation. The solution is, of course, to use a vectorial subtraction which, on the other hand, requires more complex calculations, including trigonometric functions.

The maximization of the flux using the vector is carried out with these steps:

- apply the lowpass filters on both the d and q axes current regulators outputs ($v_{d_{reg}}$ and $v_{q_{reg}}$);
- find the direction of the flux;
- rotate the regulator voltages vector along the flux, which is also the direction of the DC voltage;
- subtract from the DC voltage the perpendicular component;
- sum (or subtract if the speed is negative) the parallel component.

Contrary to the linear machine, in this case the fluxes and currents on the two axes are not independent, which means that the flux on one axis is function of the currents on both and the same can be said for the currents. This link is described by the generic nonlinear flux equation (3.22) and it can be expanded as in the following (4.47) to highlight the fact.

$$\begin{aligned}
\varphi_d &= L_{dd}(i_d - i_{od}) + L_{dq}(i_q - i_{oq}) + \varphi_{od} \\
\varphi_q &= L_{qd}(i_d - i_{od}) + L_{qq}(i_q - i_{oq}) + \varphi_{oq} \\
i_d &= L^{-1}_{dd}(\varphi_d - \varphi_{od}) + L^{-1}_{dq}(\varphi_q - \varphi_{oq}) + i_{od} \\
i_q &= L^{-1}_{qd}(\varphi_d - \varphi_{od}) + L^{-1}_{qq}(\varphi_q - \varphi_{oq}) + i_{oq}
\end{aligned} \tag{4.47}$$

Keeping in mind that L^{-1}_{dd} is short-hand for the matrix element of the inverse inductance matrix $[L^{-1}]$ and not just the reciprocal of the element of the direct matrix.

As input of the function both the requested currents coming from the MTPA function are needed. The d axis one is used as saturation in the PI regulator to limit the i_d to the correct value during slow speed operations. The q axis request is used because it lays on the ISOT curve and gives the flux requested $\varphi_{q_{req}}$ to obtain the desired torque before flux weakening, by equation (4.47).

$$\varphi_{q_{req}} = L_{qd}(i_{d_{ref}} - i_{od}) + L_{qq}(i_{q_{ref}} - i_{oq}) + \varphi_{oq} \tag{4.48}$$

At this point, using the d flux input ($\varphi_{d_{IN}}$) coming from the flux estimator and the q flux request ($\varphi_{q_{req}}$) the correct ISOT is determined by the operating point on it. The intersection between this curve and the voltage limit gives the optimal operating point for the next step. As explained before, the limit is a circle and the ISOT is a non standard hyperbole, therefore, the intersection equation is a quartic function that is not easily solved on low cost DSPs.

As it was done for the MTPA calculation, the linear approximation around the working point of the torque curve is used to simplify the operation with reasonable error. The linearization is done using the derivative of the ISOT. This means that the intersection is now between a line and a circle, which is much easier to solve.

The derivative of the torque is found from the torque expression as function of the flux (4.37) and the slope m of the line passing through the operating point is function of the two partial derivatives, as in (4.49)

$$m = -\frac{\frac{\partial T_{norm}}{\partial \varphi_d}}{\frac{\partial T_{norm}}{\partial \varphi_q}} \tag{4.49}$$

The numerator and denominator can be kept separate to avoid one division that will be done in the intersection step together with the other division required, thus optimizing the overall function. In the slope function, therefore, only these two values are calculated like (4.50) and (4.51).

$$\begin{aligned} \mathtt{NumF} = -\frac{\partial T_{norm}}{\partial \varphi_d} &= 2L_{qd}\varphi_{d_{IN}} - (L_{dd} - L_{qq})\varphi_{q_{req}} + \\ &+ \Big(L_{qd}\varphi_{od} - L_{dd}\varphi_{oq} + (L_{dd}L_{qq} - L_{dq}L_{qd})i_{oq}\Big) \end{aligned} \tag{4.50}$$

$$\begin{aligned} \mathtt{DenF} = \frac{\partial T_{norm}}{\partial \varphi_q} &= 2L_{dq}\varphi_{q_{req}} - (L_{dd} - L_{qq})\varphi_{d_{IN}} + \\ &+ \Big(-L_{dq}\varphi_{oq} + L_{qq}\varphi_{od} - (L_{dd}L_{qq} - L_{dq}L_{qd})i_{od}\Big) \end{aligned} \tag{4.51}$$

As in other equations, some constants can be calculated at a lower frequency than the main cycle time, like the big parenthesis that depend only on the calculated points on the flux maps. Of course, these constants cannot be calculated offline like in the linear machine, but they need to be re-evaluated in real time. However, the intrinsic linearization of the equations ensures that the algorithm performs a good approximation around the map point, thus not requiring a flux re-calculation at every cycle.

Having the slope of the torque in the requested point as the derivative in that point, the intersection with the voltage circle is easily obtained using the usual system (4.52).

$$\begin{cases} \varphi_{SM}^2 = \varphi_d^2 + \varphi_q^2 \\ \varphi_q = m \cdot \varphi_d + q \end{cases} \tag{4.52}$$

That leads to the general form (4.53) of the d flux reference based on the ISOT intersection.

$$\varphi_{d_{TRQ}} = \frac{-mq + \sqrt{(m^2+1)\varphi_{SM}^2 - q^2}}{m^2+1} \tag{4.53}$$

With q defined as (4.54) from (4.52) at the calculation point.

$$q = \varphi_{q_{req}} - m \cdot \varphi_{d_{IN}} \tag{4.54}$$

In the intersection function, the equation (4.53) is used while adding some control cases in the event that the root is negative, the division is by zero or the numerator is close to the minimum value of the floating point number.

The last step to provide the correct d flux reference is the imposition of the MTPV limit at higher speed. As the geometric construction suggests, the MTPV curve provides a lower limit to the d flux. It is therefore implemented as a lower saturation at the value of the MTPV at the current operating point. The MTPV limit is found by solving for the single point of intersection between the torque (as function of flux) and the voltage limit, in the same way the MTPA equation was with the current limit.

The MTPV as function of the fluxes is expressed in (4.55).

$$\begin{aligned} \mathtt{MTPV}(\varphi_d, \varphi_q) = {} & (L_{dd} - L_{qq})\varphi_d^2 + \\ & + \Big(2(L_{dq} + L_{qd})\varphi_q + (L_{qq}\varphi_{od} - L_{dq}\varphi_{oq} - \\ & - (L_{dd}L_{qq} - L_{dq}L_{qd})i_{od})\Big)\varphi_d + \\ & + (L_{dd} - L_{qq})\varphi_q^2 + \Big(L_{dd}\varphi_{oq} - L_{qd}\varphi_{od} - \\ & - (L_{dd}L_{qq} - L_{dq}L_{qd})i_{oq}\Big)\varphi_q \end{aligned} \tag{4.55}$$

That can also be written as (4.56), grouping the constants that can be calculated in a separate function at a lower frequency.

$$\begin{aligned} \mathtt{MTPV}(\varphi_d, \varphi_q) = {} & (L_{dd} - L_{qq})\varphi_d^2 + \\ & + \Big(2(L_{dq} + L_{qd})\varphi_q + \nabla_{Tq}\Big)\varphi_d + \\ & + \Big(\nabla_{Td} - (L_{dd} - L_{qq})\varphi_q\Big)\varphi_q \end{aligned} \tag{4.56}$$

Where:

$$\nabla_{Td} = L_{dd}(\varphi_{oq} - L_{qq}i_{oq}) - L_{qd}(\varphi_{od} - L_{dq}i_{oq}) \tag{4.57}$$
$$\nabla_{Tq} = L_{qq}(\varphi_{od} - L_{dd}i_{od}) - L_{dq}(\varphi_{oq} - L_{qd}i_{od}) \tag{4.58}$$

The limit value of φ_d is therefore the solution of the quadratic equation, which requires a check to make sure that the square root is real. Given the generic quadratic solution, the determinant $b^2 - 4ac$ must be positive. When it is not, a decision must be made to ensure a calculation of a value that makes sense. The easier way would be to saturate the lower value to zero, but the solution still is dependent on the input q flux (plus the currents and parameters) without any sort of feedback correction.

Although the lower saturation would still give acceptable behaviours, a better way has been found by re-calculating the q flux from the determinant using the input parameters to find the correct value that produces a d flux value consistent with the desired behaviour. The q flux is the solution of the quadratic equation of the determinant (4.59).

$$\begin{aligned} \mathtt{det}_{\mathtt{MTPV}}(\varphi_q) = &\Big(4(L_{dd}-L_{qq})^2+2(L_{dq}+L_{qd})^2\Big)\varphi_q^2+ \\ &+\Big(2(L_{dq}+L_{qd})\cdot\nabla_{Tq}-4(L_{dd}-L_{qq})\cdot\nabla_{Td}\Big)\varphi_q+\nabla_{Tq}^2 \end{aligned} \tag{4.59}$$

The solution of the MTPV quadratic (4.56) in the case of negative determinant is therefore just the general writing $\frac{b}{2a}$, where b is calculated using the φ_q from the solution of (4.59), yielding to (4.60).

$$\varphi_{d_{\mathtt{MTV}}} = \frac{2(L_{dq}+L_{qd})\varphi_q+\nabla_{Tq}}{2(L_{dd}-L_{qq})} \tag{4.60}$$

Finally, the minimum flux value between the ISOT intersection and the MTPV solution is the flux reference $\varphi_{d_{\mathtt{ref}}}$ (4.61) to be used against the input d flux coming from the previous cycle into the PI regulator.

$$\varphi_{d_{\mathtt{ref}}} = \mathrm{MIN}(\varphi_{d_{\mathtt{TRQ}}}, \varphi_{d_{\mathtt{MTV}}}) \tag{4.61}$$

Consequently, the output of the regulator is the d current reference $i_{d_{\mathtt{ref}}}$.

The other important output of the flux weakening function is the maximum q current allowed by the voltage limit, also called the *flux weakening maximum* q *current*, $\varphi_{q_{\mathtt{MAX}}}$. From the actual d flux and the maximum flux allowed, it is possible to obtain the maximum q flux with (4.62), which therefore accounts for the voltage limit.

$$\varphi_{q_{\mathtt{MAX}}} = \sqrt{\varphi_{\mathtt{SM}}^2-\varphi_{d_{\mathtt{IN}}}^2} \tag{4.62}$$

Given the d current reference, the maximum q current is found using the generic flux link equations, presented in (4.63) in matrix form.

$$\boldsymbol{i} = [L]^{-1}(\boldsymbol{\varphi}-\boldsymbol{\varphi}_0)+\boldsymbol{i}_0 \tag{4.63}$$

With some passages, the maximum deliverable q current is found using (4.64).

$$\begin{aligned} i_{q_{\mathtt{fwcMAX}}} &= L_{qd}^{-1}(\varphi_d-\varphi_{0d})+L_{qq}^{-1}(\varphi_q-\varphi_{0q})+i_{0q} \\ &= L_{qq}^{-1}(\varphi_{q_{\mathtt{MAX}}}-(\varphi_{0q}-L_{qd}i_{0d}+L_{qd}i_{d_{\mathtt{ref}}}))+i_{0q} \end{aligned} \tag{4.64}$$

4.3.3 Torque Control

After the definition of the d current reference, some checks are carried out to ensure that the operating point is inside the voltage and current limits and then the q current reference can be calculated.

Given the current limit and the $i_{d_{ref}}$, the current saturation is found using the usual current circle limit (4.65).

$$i_{q_{reqMAX}} = \sqrt{I^2 - i_{d_{ref}}^2} \tag{4.65}$$

The value found is then further limited to the voltage limit calculated in the flux weakening (4.64) as $i_{q_{fwcMAX}}$. This generates the maximum q current that lays inside both current and voltage limits, which is, as shown in (4.66), the minimum of the two. The limiting values are passed through a low pass filter which is very important for the algorithm stability.

$$i_{q_{MAX}} = \mathrm{MIN}\left(i_{q_{reqMAX}}, i_{q_{fwcMAX}}\right) \tag{4.66}$$

The q current output calculated before in the MTPA part of the algorithm (4.43), or (4.44) for IM, after the sign correction, is limited to $i_{q_{MAX}}$ thus yielding to the optimal $i_{q_{ref}}$.

4.4 NONLINEAR VARIANT IMPLEMENTATION

During the testing phase of the real motor it was noticeable that the control was inclined to weaken the flux slightly earlier than expected. The behaviour has been investigated and the major reason behind it is believed to be the measurements uncertainties and approximations of the flux characterization.

In fact, the presented formulation of the FOC heavily relies on the machine parameters to operate. If these are not know with a relatively high precision, then the generated operating point is not optimal and the control has no real way to correct itself. This reliance is also true regarding the machine model used to generate the flux state, itself used in the flux weakening.

As a possible solution, a revised FOC algorithm has been proposed which uses less parameters and more direct measures, when possible.

The general idea is to not use the fluxes produced by the machine model and exploit the regulators outputs as a sort of direct measurement of the flux state. One of the main error sources has been found to be the calculation of the maximum q current $i_{q_{\mathtt{reqMAX}}}$ from the reference d current applied $i_{d_{\mathtt{ref}}}$ shown in (4.64) because of the use of the inverse inductance, the maximum d flux $\varphi_{q_{\mathtt{MAX}}}$ of (4.62) and because there is no feedback or control of any kind.

Trying to provide $i_{q_{\mathtt{reqMAX}}}$ from more robust inputs involved the use of an additional regulator on the q axis. This is not an ideal solution firstly because it introduces a dynamic that must be accounted for and secondly because the regulator must be tuned to produce acceptable references.

Due to limited testing time, a proper testbench validation has not been carried out and, although the control is runnable and stable, it was not possible to asses if an appreciable improvement has been achieved on a real machine.

In simulation, as it will be presented in the results section 6, the new solution did not perform better than the original one if the machine parameters were assumed correct, thus highlighting how the new regulator introduces a difficult tuning variable. However, if an error in the flux characteristics was introduced, the new algorithm was able to significantly perform better.

The MTPA and torque control parts of the algorithm are exactly the same as the previously described algorithm for nonlinear machines, so the explanation will focus just on the flux weakening part and its differences.

4.4.1 Variant Flux Weakening

Since the machine flux state is not know at this stage because it was chosen to ignore the model, the first step is the generation of requested fluxes values from other sources. Both the d and q axis general flux equations (4.47) are applied using as input currents the q request coming from the MTPA function and the d request generated in the previous cycle. Very similar to the other implementation (4.48), but this time both requested fluxes are obtained.

The obtained point is therefore the desired flux that should satisfy the torque request, thus being on the ISOT. Given this definition, the next d flux reference can be found through the ISOT intersection with the voltage circle exactly as previously described with the ISOT linearization (4.49) and intersection (4.53).

At this stage the MTPV is already verified because the initial flux has been generated from a valid point because of how the current feedback operates.

The next, more important step, is the calculation of the actual machines fluxes to be compared to the request to generate the currents references through the regulators.

These values come from the values of the applied voltages $v_{d_{\mathtt{ref}}}$ and $v_{q_{\mathtt{ref}}}$. In fact, they can be considered as measurements since the inverter is made to apply these reference voltages and they include inside information about the true state of the machine. It is therefore possible to extract the flux values and their limits.

Assuming the general voltage equations in the rotating reference frame (3.15) and assumed the steady state $\frac{\varphi}{dt} = 0$, the contributions can be divided between the resistance times current term, also called voltage drop or Δv, and the flux multiplied by the speed, which is called ElectroMotive Force or `emf`. (4.67) shows such a case and it is general for any machine type.

$$\begin{cases} v_d = r_d i_d - \omega_e \varphi_q = r_d i_d + \mathtt{emf}_d \\ v_q = r_q i_q + \omega_e \varphi_d = r_q i_q + \mathtt{emf}_q \end{cases} \tag{4.67}$$

It is plain to see the relationship between `emf` and the fluxes. The idea is to use this link to produce flux values not directly obtained through the machine parameters.

The algorithm then uses (4.67) to calculate the `emf` like in (4.68).

$$\begin{cases} \mathtt{emf}_d = vd - r_d i_d \\ \mathtt{emf}_q = vq - r_q i_q \end{cases} \tag{4.68}$$

The fluxes are consequently calculated directly by dividing by the electrical speed (4.68) to give (4.69) keeping in mind how the `emf` are and the fluxes are defined switching the axes d and q.

$$\begin{cases} \varphi_d = \frac{1}{\omega_e}\mathtt{emf}_q = \frac{1}{\omega_e}(vq - r_q i_q) \\ \varphi_q = \frac{1}{\omega_e}\mathtt{emf}_d = \frac{1}{\omega_e}(vd - r_d i_d) \end{cases} \tag{4.69}$$

In this way it is also possible to define the maximum `emf` module available based on the maximum stator voltage V and the contribution of the resistances $\Delta \boldsymbol{v}_r$. Simply put, it is the maximum voltage minus the magnitude of the voltage drop. The only complication is that the values under investigation are vectors, so the actual limit construction must be done using the proper vectorial form to ensure that the voltage drop is accounted in the right way. In fact, there are case, like during regeneration, in which the resistance helps the flux production and the limit is actually higher than the maximum voltage value.

To keep the formulations easy enough, an approximation is introduced by assuming the same `emf` direction and, thus, the geometric construction of the $\mathtt{emf}_{\mathtt{lim}}$ is summarized in Fig. 4.16. Where the voltage applied by the inverter is named $\boldsymbol{v}_{\mathtt{out}}$ and the result obtained is the $\mathtt{emf}_{\mathtt{lim}}$ magnitude.

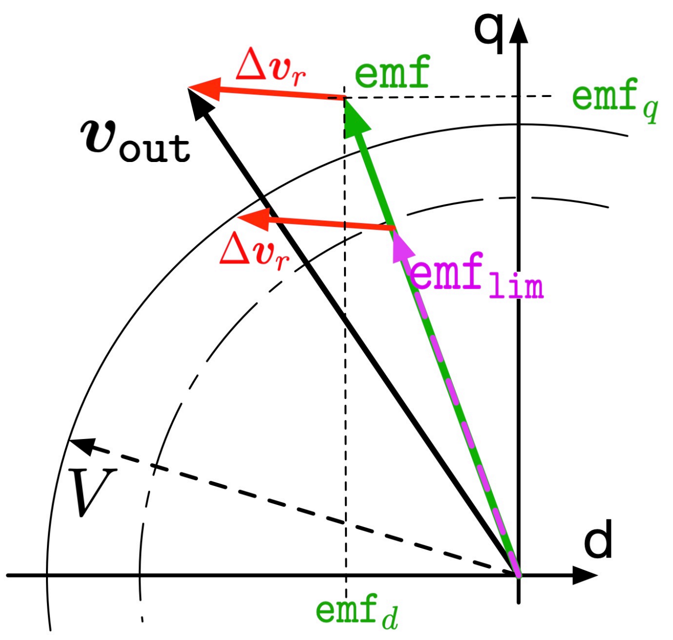

q
d
emf
emf$_q$
emf$_d$
$\Delta \boldsymbol{v}_r$
$\Delta \boldsymbol{v}_r$
$\boldsymbol{v}_{\mathtt{out}}$
emf$_{\mathtt{lim}}$
V

Figure 4.16: *Geometry construction of the* `emf` *limit from the applied voltages.*

Knowing the currents either from the measurements or the model prediction and the resistance parameter, the voltage drop vector is easily found with the definition (4.70).

$$\begin{cases} \Delta v_{rd} = r_d i_d \\ \Delta v_{rq} = r_q i_q \end{cases} \tag{4.70}$$

The $\Delta \boldsymbol{v}_r$ is subtracted vectorially from the voltage applied $\boldsymbol{v}_{\text{out}}$ to obtain the EMF applied `emf`, by (4.68).

Then the `emf` limit must be found by finding the $\mathtt{emf}_{\mathtt{lim}}$ that makes true the vectorial sum lesser than the limit, as expressed in (4.71).

$$V^2 = (\mathtt{emf}_{\mathtt{limd}} + \Delta v_{rd})^2 + (\mathtt{emf}_{\mathtt{limq}} + \Delta v_{rq})^2 \tag{4.71}$$

Since the single condition has two variables, to solve it it is necessary to introduce another equation. As explained, the approximation used is that the `emf` vectors, the applied one and the limit one have the same direction. This means that the slope of the vectors is the same and this allows to link the d axis component to the q axis one, as shown in (4.72).

$$\frac{\mathtt{emf}_q}{\mathtt{emf}_d} = \frac{\mathtt{emf}_{\mathtt{limq}}}{\mathtt{emf}_{\mathtt{limd}}} \quad \rightarrow \quad \mathtt{emf}_{\mathtt{limq}} = \frac{\mathtt{emf}_q}{\mathtt{emf}_d} \mathtt{emf}_{\mathtt{limd}} \tag{4.72}$$

The two conditions (4.71) and (4.72) are combined and the solution of the quadratic relationship is carried out to find the magnitude of the $\mathtt{emf}_{\mathtt{lim}}$. Some passages are shown before the solution is derived in (4.73), of course selecting only the positive root.

$$\begin{aligned}
V^2 &= \mathtt{emf}^2_{\mathtt{limd}} + 2\mathtt{emf}_{\mathtt{limd}}\Delta v_{rd} + \Delta v^2_{rd} + \mathtt{emf}^2_{\mathtt{limq}} + 2\mathtt{emf}_{\mathtt{limq}}\Delta v_{rq} + \Delta v^2_{rq} \\
0 &= \mathtt{emf}^2_{\mathtt{limd}} + \frac{\mathtt{emf}^2_{\mathtt{q}}}{\mathtt{emf}^2_{\mathtt{d}}}\mathtt{emf}^2_{\mathtt{limd}} + 2(\Delta v_{rd} + \frac{\mathtt{emf}_{\mathtt{q}}}{\mathtt{emf}_{\mathtt{d}}}\Delta v_{rq})\mathtt{emf}_{\mathtt{limd}} + (\Delta v^2 - V^2) \\
0 &= (\mathtt{emf}^2_{\mathtt{d}} + \mathtt{emf}^2_{\mathtt{q}})\mathtt{emf}^2_{\mathtt{lim}} + 2(\mathtt{emf}_{\mathtt{d}}\Delta v_{rq} + \mathtt{emf}_{\mathtt{q}}\Delta v_{rq})\mathtt{emf}_{\mathtt{lim}} + (\Delta v^2 - V^2) \\
0 &= (\mathtt{emf}^2)\mathtt{emf}^2_{\mathtt{lim}} + 2(b)\mathtt{emf}_{\mathtt{lim}} + (\Delta v^2 - V^2) \\
\mathtt{emf}_{\mathtt{lim}} &= \frac{-(b) + \sqrt{(b)^2 - \mathtt{emf}^2(\Delta v^2 - V^2)}}{\mathtt{emf}^2}
\end{aligned} \tag{4.73}$$

The $\mathtt{emf}$ and their maximum value are then used to calculate the maximum allowed q flux and, consequently the maximum q current using a PI regulator.

Given the module of the $\mathtt{emf}$ as $\mathtt{emf}^2 = \mathtt{emf}^2_{\mathtt{d}} + \mathtt{emf}^2_{\mathtt{q}}$ and the maximum module of the $\mathtt{emf}$ as $\mathtt{emf}_{\mathtt{lim}}$, then the maximum allowed $\mathtt{emf}_{\mathtt{d}}$ would be (4.74).

$$\mathtt{emf}_{\mathtt{d_{MAX}}} = \sqrt{\mathtt{emf}^2_{\mathtt{lim}} - \mathtt{emf}_{\mathtt{q}}} \tag{4.74}$$

The way of creating a reference for the max q current is then a simple matter because the reference $\mathtt{emf}_{\mathtt{d_{MAX}}}$ can be compared to the actual value $\mathtt{emf}_{\mathtt{d}}$ obtained in (4.68). If they are both divided by the electrical speed, then the comparison is directly between two fluxes and the regulator can operate on a flux error to generate the q current reference.

As an additional feature, this regulator can be saturated to the maximum available q current $i_{q_{MAX}} = \sqrt{I^2 - i^2_{d_{ref}}}$ exactly in the same way the d flux regulator is saturated to the MTPA value.

The application of both the voltage and current limits is thus applied as a consequence of the implementation used. For the sake of changing as few functions as possible, later saturations in place for the torque control are still implemented in the deployed algorithm given that they add an additional safety measure without appreciable performance degradation.

Given the critical nature of the flux regulators in this implementation, effort has been spent to simplify the tuning and providing an efficient regulation. The solution was the definition of variable coefficients of the regulators based on the machine parameters. The goal is to have as tunable just per-unit values that relate to the actual k_p and k_i through the inductances. Given that for the nonlinear implementation these values change with the operation point, it is possible to modify the response of the system during the operation, improving the performance.

5 ALGORITHM DESCRIPTION

The section presents a more detailed account of how the analytical algorithm is implemented inside the development environment focusing on the adaptations made to the equations presented in the previous chapters to be correctly implemented.

The software suite adopted is Matlab Simulink because it is already in use both LEMAD and collaborating company and, thanks to its advanced features to automatically generate robust C code, it is almost an industry standard for the Model-In-the-Loop (MiL) and Hardware-In-the-Loop (HiL) development process.

In fact, Simulink was created to model complex time-dependent systems using a user-friendly graphical interface in which signals and variables are lines and functions can be grouped inside blocks for easy management. The solver engine is optimized for continuous-time systems to accurately model real-time applications, but it has also discrete-time solvers to simulate a control algorithm running on DSPs.

This is quite ideal for MiL development since it allows to run both the control and the simulation of the system under study in the same environment. This development method is very useful and widely spread in the industry sector because it reduces greatly the need of a physical system to develop the first implementations and allows faster improvements and verifications.

The automatic code generation tool of Matlab then safely and efficiently converts the software already partially validated to be deployed on the target hardware, ideally leaving to direct manual coding just the low level software functions specific to the DSP used. This feature is also advantageous to more easily switch hardware, since the Simulink generated code can be considered general and deployable to numerous different target with few menu configurations.

The first part of this section will describe the overall architecture of the software model without going much into specifics and afterwards every function will be presented and the most crucial ones will be discussed in detail since they represent the main objective of the research and the most novel interpretation of the control algorithm for nonlinear machines.

5.1 GENERAL MODEL ARCHITECTURE

To accomodate the various requirements that the model must fulfill that include MiL development, HiL deployment and code generation, a main system subdivision must be followed to isolate the functions that must be removed or changed for the different objectives. For example the implementation of the model of the real object under testing is grouped in a single block called *PLANT* and it will not be used for code generation.

The first distinction to actuate is the one between the main software application under testing and the model of the system simulating the real world. The former is called *ASW* while the latter is the *PLANT*. In this way there is a quick and effective way to separate all that will be deployed on the DSP running in discrete time and what is added just to make the simulation realistic during development and that can run in continuous time.

The outputs of the application are acting on the plant model that is reacting accordingly and produces feedback measurement points to enable realistic control behaviours, as much as the plant accuracy is capable.

Another responsibility of the *PLANT* block is the implementation of the interrupt signals to simulate the internal triggering clock of the microprocessor. In fact, when deployed, the functions and tasks will be triggered to be executed by timing flags called *interrupts* generated either by events or recursive scheduling. The interrupts of course have different priorities and the management of these triggers is a fundamental requirement of the basic software specific of the hardware.

An example of recursive scheduling is the *APP Task* that can be effectively executed every dozen milliseconds without waiting for any particular situation. As a different example, a task in charge of reading a communication signal is executed just when an incoming message is detected and in turn it can create an interrupt signal to trigger one or multiple functions that need the message to operate.

The *PLANT* then outputs triggering signals simulating the interrupts and there is a specific subsystem called *Scheduler* at the same level that implements the scheduler, which is the function in charge of deciding the order of operation of all the other functions and pretty much actuates the discrete time mode in the Simulink model.

The last general system is the *Diagnostic - Plots* one which, as the names suggests, is used to collect all the tools to visualize and debug the software. It usually is a long series of Scope blocks that group signal coming from different parts of the system to trace the behaviour, but it also can contain log elements for further elaborations or record.

This overall distinction is very useful to easily separate the various parts of the model which are used in different parts of the development process. the separation between *ASW* and *PLANT* is particularly emphasised because the final product will not run anything inside the latter subsystem.

The overview of these main blocks is presented in Fig. 5.1 where the feedback signals are highlighted in green.

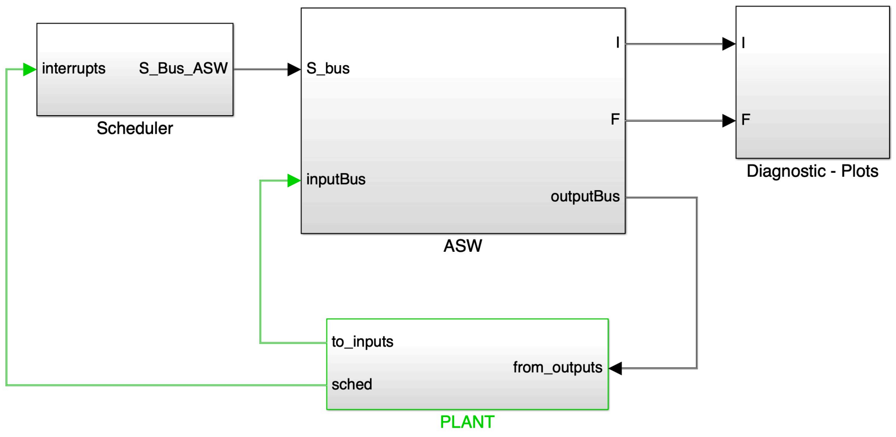


Figure 5.1: *Higher level of algorithm implementation with PLANT system in feedback to the application software ASW.*

All the block's inputs and outputs are managed with buses thus collecting all the information in a single structure that can be routed easily around the diagram. Tab. 5.1 provides the name and brief description of the main buses.

The core functions that operate the electric drive control algorithm are inside the *ASW* block, in an additional layer to make it independent of the hardware used. Because the focus of the research is just in the actual control and not the infrastructure to run it, only a brief explanation will be given in the next pages about the general considerations implemented to allow proper code generation and functionality of the end product.

5.1.1 Plant

In the plant subsystem is collected all that is not directly related to the software that will be deployed on the application. This includes the models of the real system or control functions and

Table 5.1: *List and description of main signal buses.*

Name	Description
inputBus	Measurement points and user inputs to the system, from Plant of real hardware
S_bus	Collection of all the scheduling triggers to the ASW
outputBus	All outputs of the algorithm that relate to the execution of the software objective
I	Relay of the actual input signals after the pre-processing
F	Full list of outputs of the single atomic functions
sched	Collection of the interrupts timing the scheduling

the outside signals that the user might generate. Moreover it has the simulation of the hardware interrupts that are used as timing signals to the scheduler of the application.

This subsystem can become one of the more complex used because its goal is to try to replicate the real conditions of the real-life device, the measuring sensors and the variable user inputs. For the electric machine control application the main features of this block are the generation of the the torque request, the speed calculation and the model of the electric drive simulating the behaviour of the device to control. Many other expressions can be added to make the plant as real as possible.

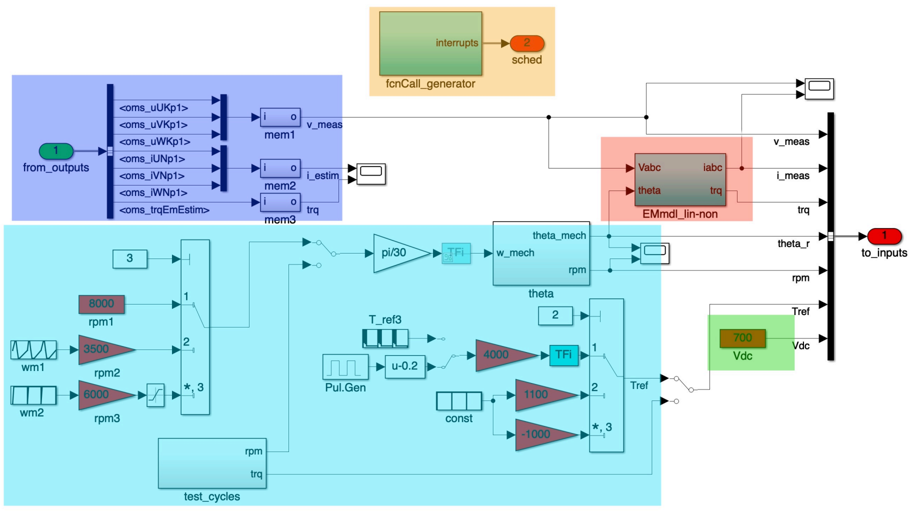


Figure 5.2: *Example of PLANT subsystem with highlighted various important signal generators for electric drive's control.*

In Fig. 5.2 is presented an example of such system for rapid development that includes the essential outputs for a good simulation of the real system. The shaded areas delimit the sections of code that are dedicated to:

BLUE acquisition of the main outputs of the application software that will be applied to the real system, in this case the phase voltages. Currents and torque estimation are also included for comparison.

ORANGE generation of the interrupt signals;

RED continuous time electric drive model that takes the voltages and provides the best approximation of the machine currents as see from the sensors;

GREEN simulation of the input DC voltage;

LIGHT BLUE generation of the torque request and speed measurement, either by manual tables or through pre-defined test cycles. The speed is also integrated to provide the mechanical rotor angle of the machine.

As explained, this is the bare minimum for electric drives and it does not include more advanced features like proper sensor acquisition simulation, hardware basic software, communication delays, user online configuration commands and DC voltage source models like, for example, the battery pack.

5.1.2 Scheduler

The objective of this system is to call in the correct order the atomic functions of the application software.

Traditionally Simulink coding automatically defines the execution order of the calculations based on the blocks requirements. For more complex models, however, the use of an explicit user-defined scheduler controlling the overall atomic function execution order is warmly suggested to increase the compatibility and flexibility of the end product.

The method adopted makes use of the *function-call* signal type of Simulink. This is similar to traditional subsystem triggers, but acts more like calling functions in C code or similar, hence the name. As is understandable, the main advantage is the similarity with the system used in final code generated, the thing that will be deployed on the hardware, thus reducing possible compatibility issues. Another advantage is the possibility of calling more than once the same function without having to create a new instance in the diagram, like in line code style.

The generation of the function-call signals in Simulink is best actuated through the state flow blocks, which in turn allow easy definition of the order of events. Fig. 5.3 illustrates the schematic logic behind the scheduler adopted for this application with the task distinction between *APP*, *FOC* and *PWM* tasks as described in section 2.3.

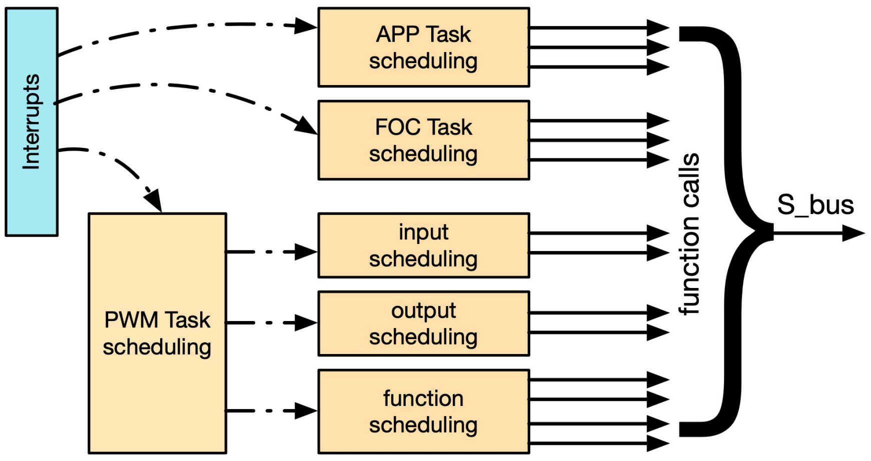
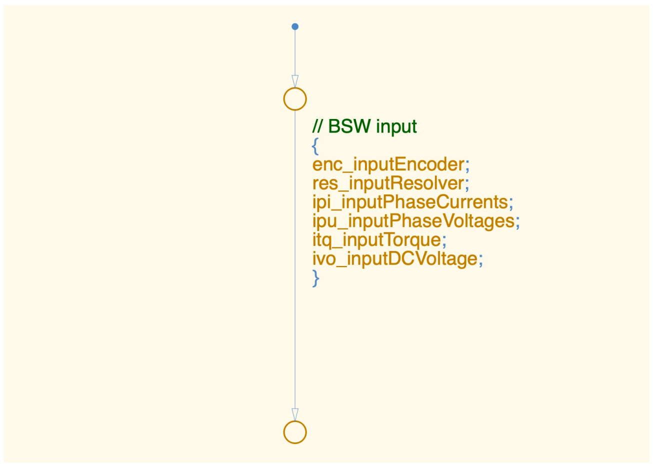


Figure 5.3: *Schematic overview of the execution scheduling system used in the application.*

Fig. 5.4, instead, shows how the order and event calling is operatively implemented inside the state machine block of the application. The names are the actual names of the function-call signals generated and they have the same name of the function they trigger for easy recognition. The yellow color of the words indicates that those are events defined in the explorer section of the diagram and are interpreted by Simulink as function-call outputs. Of course, the execution order starts from the top.

In summary, every time an interrupt happens, it triggers the execution of a state machine specific to that timing that in turn generates, in order, the specific function calls to the desired subsystems, or to other state machines to sub-divide the logical steps. All the outputs are then grouped in the same bus to be distributed wherever is needed.



```
// BSW input
{
enc_inputEncoder;
res_inputResolver;
ipi_inputPhaseCurrents;
ipu_inputPhaseVoltages;
itq_inputTorque;
ivo_inputDCVoltage;
}
```

Figure 5.4: *Example of state flow event calling of the application for an input subsystem.*

5.1.3 ASW

This block is the collection of all the atomic functions of the application software under development. Some structural grouping is made for logic purposes, but it is important to remember that the execution is decided by the scheduler and, in theory, every function could be called at any time. This means that every atomic function should be considered as a separate entity, almost independent.

Of course, especially in complex applications, not always it is possible to generalize too much, and the functions will have a very specific order of execution to work properly. However, the naming convention and preliminary output definitions discussed in 2.2 and 2.3 have been defined to enable this independent working method. Meaning that, thanks to this conventions, the single functions can be modified, improved or substituted with variants with very limited knowledge of the entire system, while still being effective and, more importantly, reducing issues.

The subdivision of this system is in three parts (*Input*, *Function* and *Outputs*), as shown in Fig. 5.5 to reflect the logical order and functionalities of the types of functions collected.

Predictably, the *Input* block subdivision groups all those functions that acquire signals from the outside, being them sensors or through communication protocols like CAN-bus. The objective is to condition the different signals in a standardised manner, ensuring that following calculation are provided with the valid signals in the requested range and type.

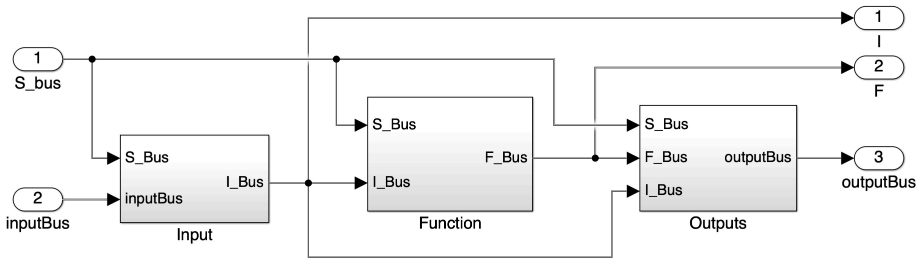


Figure 5.5: *A look inside the ASW block and its division in input/-function/outputs parts.*

An example of the minimum required input functions for the electric drive control is shown in Fig. 5.6 and involves the functions explained in Tab. 5.2

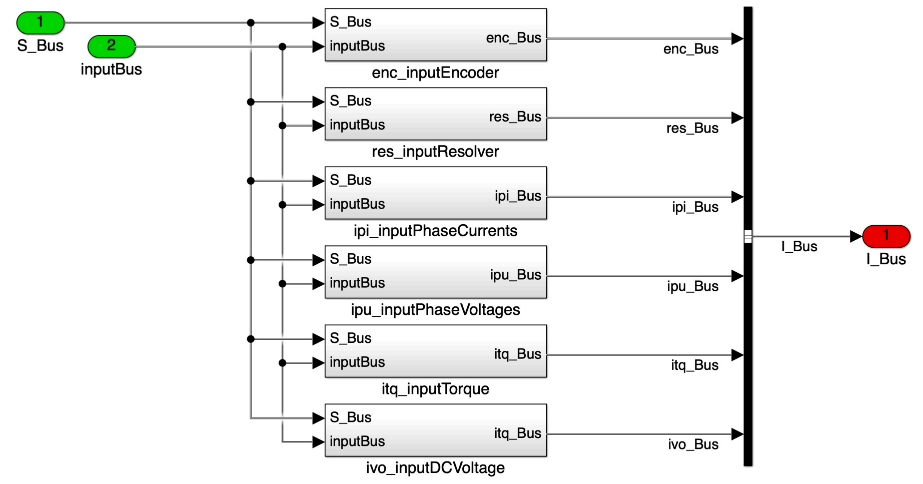


Figure 5.6: *Example of* Input *block functions.*

Table 5.2: *List of possible input functions for inverter control applications.*

Abbr.	Name	Description
`enc`	`inputEncoder`	Encoder position angle sensor
`res`	`inputResolver`	Resolver position angle sensor
`ipi`	`inputPhaseCurrents`	Phase currents measurements
`ipu`	`inputPhaseVoltages`	Phase voltages measurements
`itq`	`inputTorque`	User torque request
`ivo`	`inputDCVoltage`	DC bus voltage measure

On the other end of the execution order, the *Outputs* block collects a number of functions that prepare the output signals of the application to be used by the hardware device or communication protocols. The example in Fig. 5.7 is again related to the inverter control under study. The only required outputs are the Duty Cycles of the PWM driver, that are processed in the `opd_outputPwmDuty` block. The other two are simpler collections of measurements and statuses that may be used for debugging or diagnostic purposes.

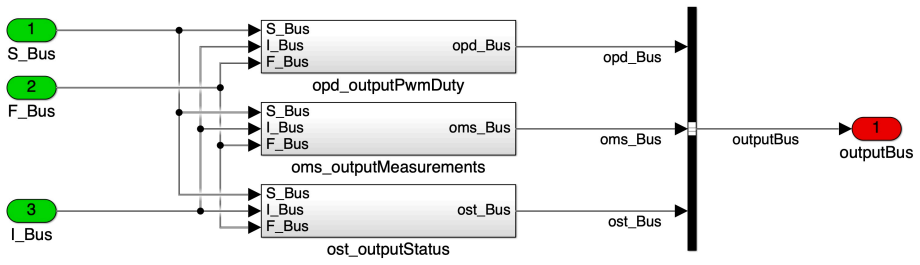


Figure 5.7: *Example of* Outputs *block functions.*

Key feature of this subdivision is that these two conditioning and validation layers are the parts that are specific for the use of the control algorithm. Which means that if the testing method or destination hardware is changed, then only these functions have to be modified to accomodate the new setup, leaving unaltered the core functions and features.

A very important example is their use in the MiL compared to the DSP deployment. In the first case these functions are pretty much empty, containing just saturations and datatype matching, because the inputs of the system are generated by Simulink functions in the same programming environment, therefore already good for the application. In the second case, instead, most inputs are generated by low-level software or driver components specific to the DSP through dedicated function calls and their management must be carried out to provide the signals to the core functions given their requirements.

The development of such blocks is outside of the scope of this research.

5.1.4 Function

The core functions that actuate the desired application are all contained in the *Function* subsystem and they are to be considered completely independent of the hardware implementation, thanks to the conditioning layers before and after, as explained in the previous section.

Inside here, the list of functions preliminarily defined in Tab. 2.5 is implemented and the brief overview of Fig. 5.8 shows the high number of atomic functions needed and developed. With few exeptions, the list and names were maintained, as was the subdivision in three priority tasks described in section 2.3 and illustrated in Fig. 2.5.

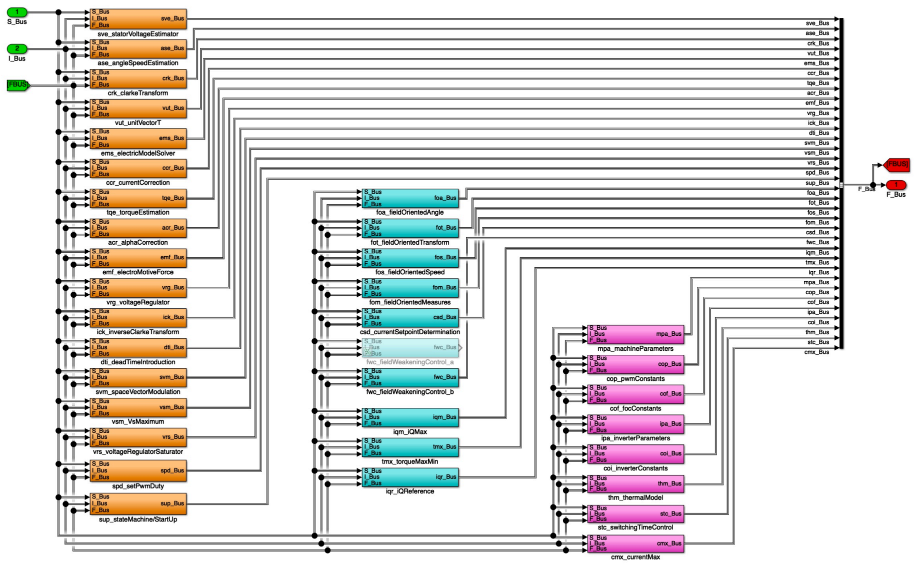


Figure 5.8: *Example of* Function *block implementation in Simulink.*

The tasks are grouped in columns and drawn with matching colours for easier recognition. Moreover, an overall sense of the execution order is kept by putting the first functions to be run at the top of the columns and the others underneath in order. However, as described in the scheduler section 5.1.2, the true order of execution is not decided at this level, but in the specific scheduler system. It is in any case advisable to keep a general order to improve debugging.

To better understand the logic behind the functions and signals use, Fig. 5.9 presents a simplified schematic view of the system.

It is clear to see that there are only 3 signal structures in the whole program:

S_BUS contains all the functions-call signals coming from the scheduler, thus defining the timing of the functions;

I_BUS comes from the *Input* sub-system and contains the outside inputs;

F_BUS collects all the individual function outputs of this system, feeds them back to all the included functions and delivers the the signals to the outside output systems.

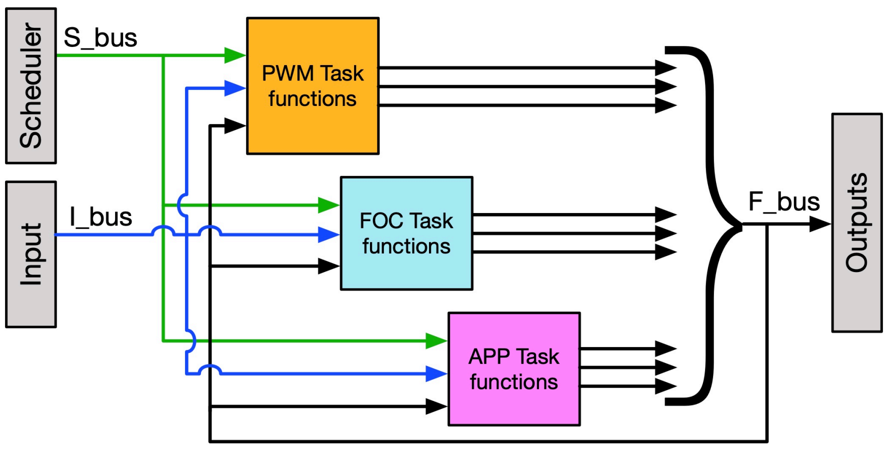


Figure 5.9: *Schematic of the* Function *block architecture.*

It is very important to note that the use of the F_bus in feedback to all the functions effectively provides every block with all the information created in the application and, consequently, renders useless the normal execution order detection used by Simulink because everything is in a giant loop with not distinct starting point.

The last remark may seem like a huge problem, but the use of the scheduler together with the function-call method bypasses completely the need of the Simulink engine to order the operations (at this level). It additionally has the advantage, as discussed previously, to mimic the normal scheduling approach used in C code, thus producing more efficient code. It also does not require a delay on the feedback loop.

In this sense, the F_bus can be though of like a global C structure that can be accessed at any time by any function and updated with the scheduler timing.

In the implemented scheme of Fig. 5.8 the feedback signal is not immediately obvious because, for visual clarity reasons, the connecting line has been substituted with a goto/from routing method.

It is interesting to point out that it is not forbidden by the architecture to include one whole task inside a single function. However, this is not a clear improvement, especially increasing the system complexity and working as a team, or even when talking about portability solutions. Indeed, as described, the method allows much flexibility in developing the various algorithm pieces at different speeds and precisions, maybe using older, more reliable functions while setting up the new parts or quickly redefining the whole structure with just different scheduling without rewriting the whole program.

Buffering issues between tasks

Given that the `F_bus` is to be considered like a global structure that can be accessed by task at different priorities, the issue of preemption and data integrity has to be addressed, as previewed in section 2.4.

As a small recap, it must be avoided that lower priority tasks that are interrupted have their own data corrupted by the new results. It has been explained that the solution is the implementation of a double buffer communication layer between the tasks.

The practical way to implement this in the architecture described (that uses function-call) is by using a combination of rate transition and memory blocks when a low priority task (with slower timing) is using a variable coming from a higher priority one (faster). The contrary, from faster to lower priority, does not require any particular considerations, but a memory block could provide an easy way to initialize the signal until the slower task is run the first time.

The schematic summary is illustrated in Fig. 5.10 and the mask parameters to be used are shown in Fig. 5.11.

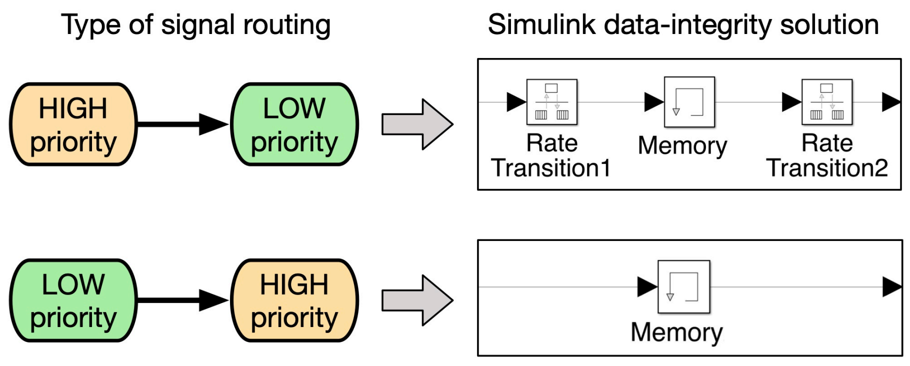
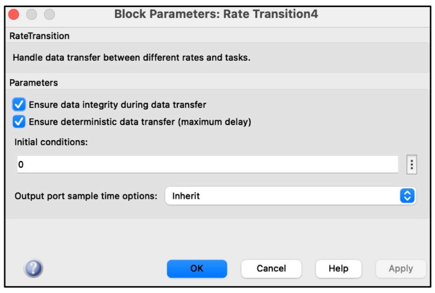


Figure 5.10: *Practical data integrity signal routing between tasks at different priority.*

5.1.5 Atomic Function Inclusion

At this stage, thanks to the previous sections, it should be clear the overall information flow of the program. It is therefore important to know how to insert properly a function in such an articulated architecture.



Block Parameters: Rate Transition4

RateTransition

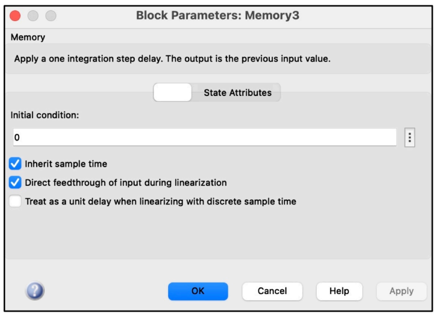
Handle data transfer between different rates and tasks.

Parameters

- [✓] Ensure data integrity during data transfer
- [✓] Ensure deterministic data transfer (maximum delay)

Initial conditions: 0

Output port sample time options: Inherit



Block Parameters: Memory3

Memory

Apply a one integration step delay. The output is the previous input value.

Initial condition: 0

- [✓] Inherit sample time
- [✓] Direct feedthrough of input during linearization
- [ ] Treat as a unit delay when linearizing with discrete sample time

Figure 5.11: *Setup parameter masks for the rate transition and memory blocks for data integrity.*

As an example, the simple function implementing the Clarke transformation `crk_clarkeTransform` is used to show the various layers that are needed in the Simulink scheme for a proper inclusion inside the general algorithm.

The first layer is, of course, the block at the *Function* subsystem level, Fig. 5.12. This level is identical for every function and it just provides all the three input buses to the subsequent layer and outputs the a single bus with all the outgoing signals.

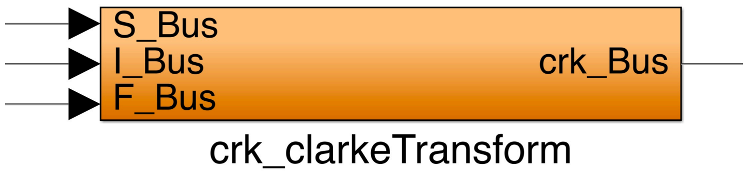


Figure 5.12: *Atomic function first integration level: function subsystem.*

The next level shown in Fig. 5.13 is used to extract from the input buses the requested signals needed for the function execution and to create the output bus with the proper name, which is the function name's abbreviation. If one bus is not used, then it still enters this layer and it is just terminated as shown in the example for the `I_bus`.

Then there is a level dedicated to the function-call implementation, Fig. 5.14. This separation can be useful if the function is implemented through direct C code, also called an S-function, since it requires a more rigid input-output definition.

Finally, Fig. 5.15, inside the last layer the proper calculations executing the goal of the function are carried out and inside here the rules are more flexible since there should not be any outside communications other than the approved ones defined by the previous levels. To note how at this level the model can be

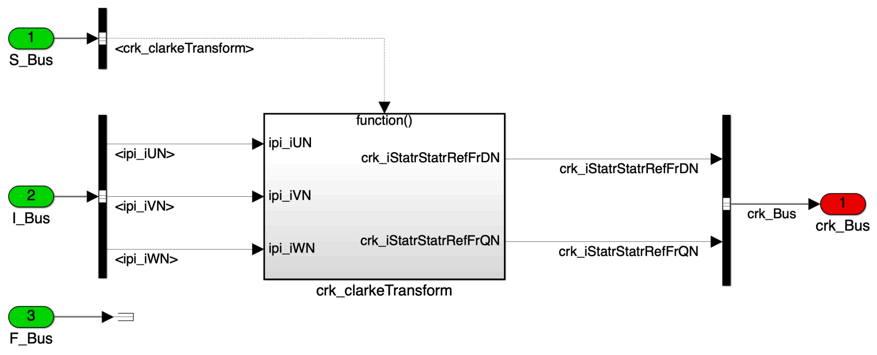


Figure 5.13: *Atomic function second integration level: bus input selection and output creation.*

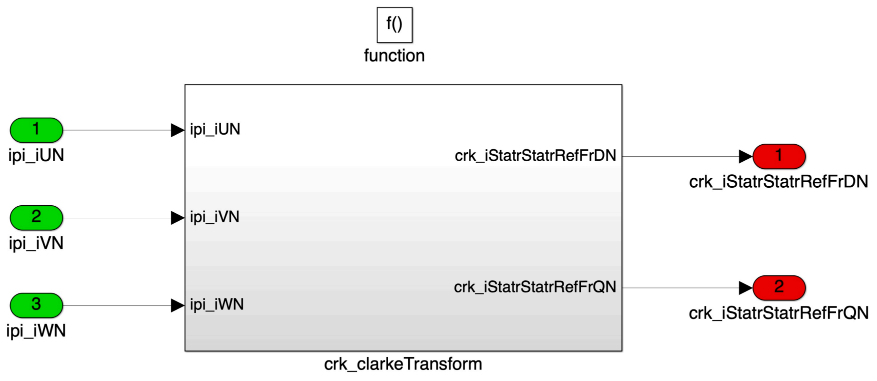


Figure 5.14: *Atomic function third integration level: function-call.*

intended in every aspect as a basic Simulink scheme in discrete time.

In the example there is a simple block implementing the transformation and a switch selection to choose different kind of input manipulations if there are current sensors on all three phases or just two.

5.1.6 Parameters and Calibration Values

The various parameters used throughout the model are inserted by simply using them in the parameter field of the blocks. Their definition is done by using one or multiple Matlab configuration scripts that load in the base workspace the constants with the proper name and value.

This is standard for code generation, since the coder will interpret the numbers in the workspace as tunable parameters and create dedicated C structures easily accessible.

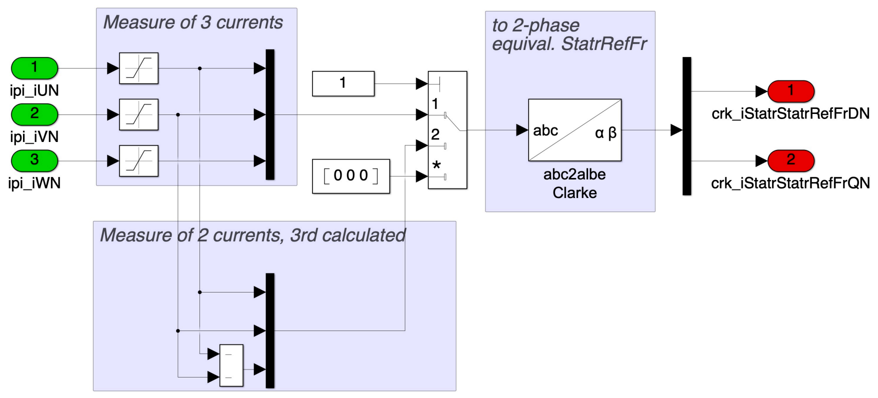


Figure 5.15: *Atomic function fourth integration level: calculations.*

As said in the naming section 2.2, all of these parameters are considered calibration values, thus requiring a "_C" suffix.

If necessary, simple elaborations and logic can be carried out in the script with the condition that at the end of the script run only actual calibration values are present in the workspace and not temporary values not intended to go on the hardware application.

Appendix A reports one example of parameter configuration script used during the software implementation showing also the internal subdivision for the different functions.

5.2 MPA – MACHINE PARAMETERS

One of the most important function is surely the one that calculates and collects all the machine parameters like pole pairs, machine type, resistances and inductances. its inner working are shown in Fig. 5.16.

This is a function that has been heavily improved during the development process from linear machine to nonlinear, while maintaining the compatibility. Indeed, in the linear machine uses only constant values of equivalent inductance or, in special cases, they can be described using relatively simple lookup tables.

For nonlinear machines, on the other hand, the calculations involve the choice of the flux interpolation method and the partial derivative of the result, plus the output of the point related to the calculated values which is key for the linear approximation at the base of the equations, starting from (3.22).

For the model under testing, the method chosen is the polynomial interpolation for its faster execution time and the lack of online adjustments requirements of the application. In this case,

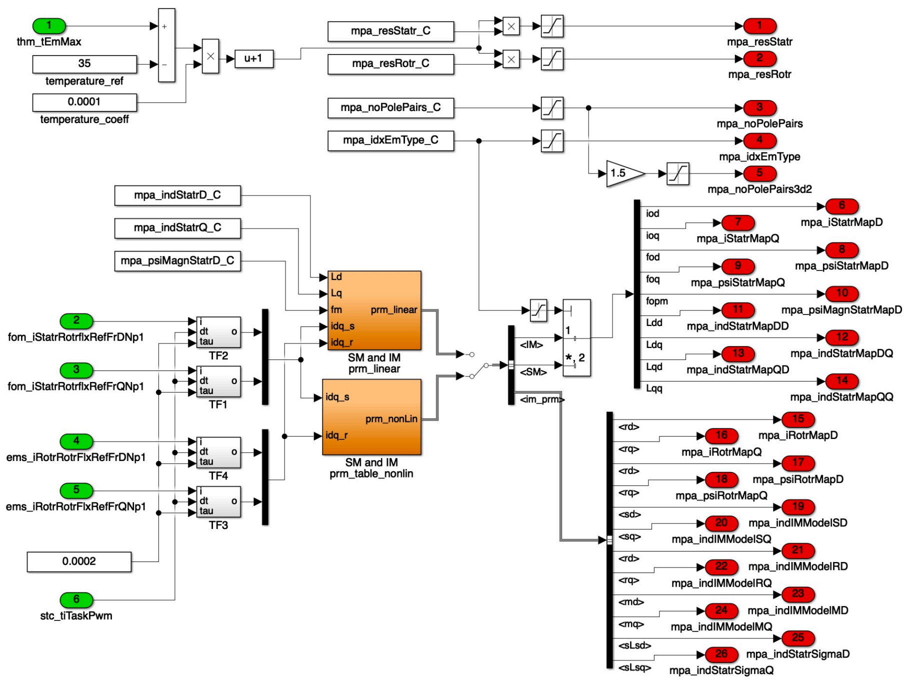


Figure 5.16: *MPA function, overview.*

then, the polynomial coefficients are loaded into a Matlab function presented in Fig. 5.17, because it slightly simplifies the derivation and modification of the parameters in the development phase.

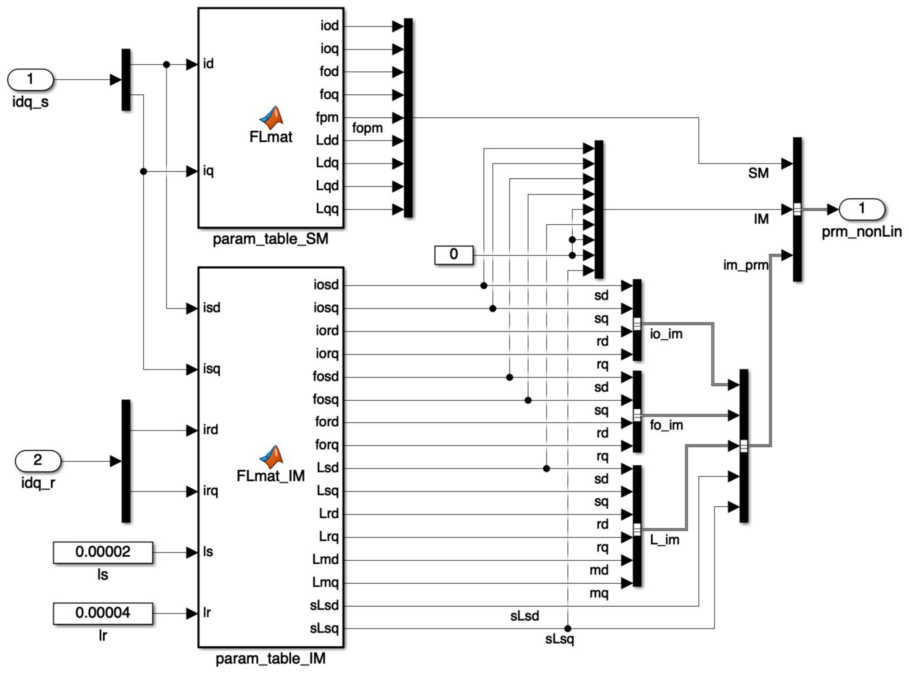


Figure 5.17: *MPA function, detail of nonlinear parameters calculation for SM and IM.*

To point out the visible difference between SM and IM, with the latter one requiring a lot more outputs to be processed. This of course is due to the fact that the nonlinear implementation distinguishes the machine model between the two types of motors, in contrast to the general Unified Machine Algorithm, to optimize the calculations.

Therefore, while the IM uses the full resistance and inductance matrix, including stator and rotor values, the SM model is streamlined by using just the stator portion, the only one that influence the machine state. This means that the IM model makes use of 2 rotor currents, 2 rotor fluxes and 6 inductances more than the other type, accounting for the symmetries of the inductance matrix (3.67).

As an additional optimization, the system does use the same outputs for the stator values, which are also the most relevant for the FOC task. In fact, the FOC is independent of the machine type, apart for a small detail described in section 4.3.1, and the coefficients of the matrix identifying the motor are already appropriately calculated in this block to be compatible in both cases.

5.3 COP PWM COEFFICIENTS

This is a function that takes the machine parameters and some others and calculates useful constants for the PWM task, so that the execution of the main task is faster. Of course this is not so true for the nonlinear approach because it requires the frequent machine parameter re-calculation, but thanks to the linearization approximation it is still true that the function in this block can be run a bit slower.

The overview of the calculations carried out inside the system is presente din Fig. 5.18.

It pretty much consist of the definition of the various inductance matrices used by the electric model like the inverse one, the symmetric, the inverse of the symmetric, the asymmetric and the special case for the inverse of the IM matrix.

Plus it calculates the parts of the current equation that use only parameters, expressed by (3.49).

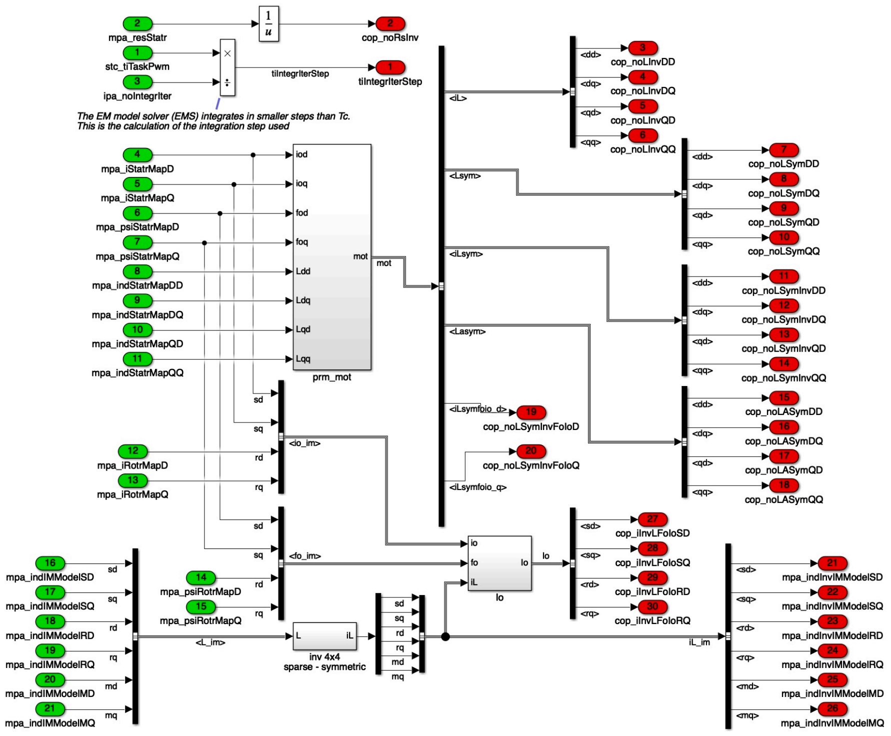


Figure 5.18: *COP function, overview.*

5.4 COF FOC COEFFICIENTS

In the same way as COP, this function groups many various coefficient calculations starting from the machine parameters that are going to be use in the FOC task to try to limit the number of operations inside the higher priority threads.

They are so many that for visualization purposes is was better to compact them all inside a subsystem and only show in Fig. 5.19 the inputs and outputs.

Any part of the equations described in section 4 that do not contain variables is instead pre-calculated inside here, like:

- $L_{dd} - L_{qq}$ and $L_{qq} - L_{dd}$;
- $2L_{dq}$ and $2L_{qd}$;
- $L_{dd}L_{qq} + L_{dq}L_{dq}$;
- $\varphi_{od} - L_{dd}i_{od} - L_{dq}i_{oq}$ and $\varphi_{oq} - L_{qq}i_{oq} - L_{qd}i_{od}$.

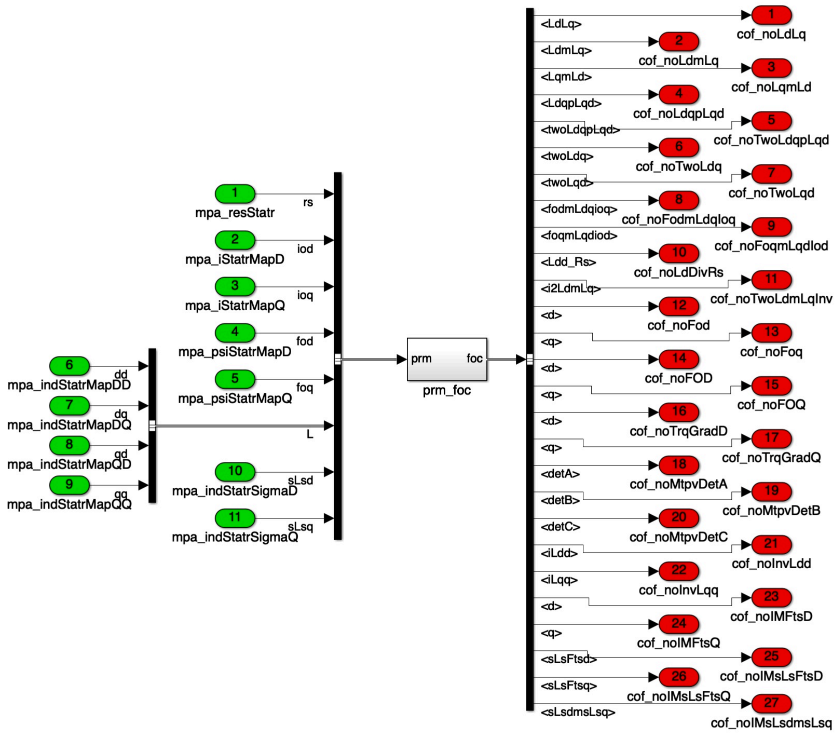


Figure 5.19: *COF function, overview.*

5.5 IPA INVERTER PARAMETERS

In this block are introduced all those parameters that are related to the inverter device and its proper control. It is also used to input other parameters not directly related to the machine ones.

For example, some of the values to define can be seen in Fig. 5.20 and are:

- switching Dead Times for the hardware and for the software compensation;
- current threshold to enable dead time sign switch during operation;
- number of iterative integration cycles of the EM model solver;
- the static voltage reserve used in the flux weakening;
- the coefficients for the voltage PI regulators.

The example also shows the predisposition to accept the temperature of the inverter and a possible way to consider it for tweaking some values to improve the behaviour.

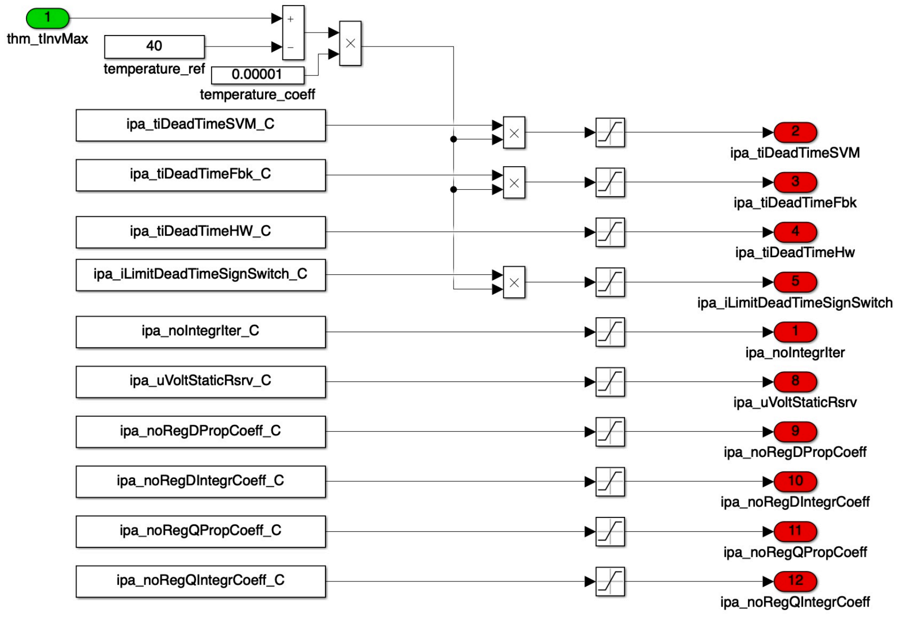


Figure 5.20: *IPA function, overview.*

5.6 COI INVERTER CONSTANTS

Like COP and COF, this system is used to elaborate and prepare useful constants that are used inside the inverter control part of the algorithm. In the example case, Fig. 5.21, it boils down to converting the dead time expressed in seconds into the required per unit values and to pre-process the regulators coefficients accordingly to the implementation used, see Appendix B.

5.7 THM THERMAL MODEL

This is just a placeholder for a high level function simulating the thermal behaviour of the electric drive that is implemented by the collaborating company. This would provide the temperature measurements and estimations to be used to update model parameters like the stator resistance and improve performance. Moreover, it would enable the actuation of safety derating and limitations.

5.8 STC SWITCHING TIME CONTROL

This function is responsible to define the task frequency configuration. In the example case of Fig. 5.22 the timing of the three

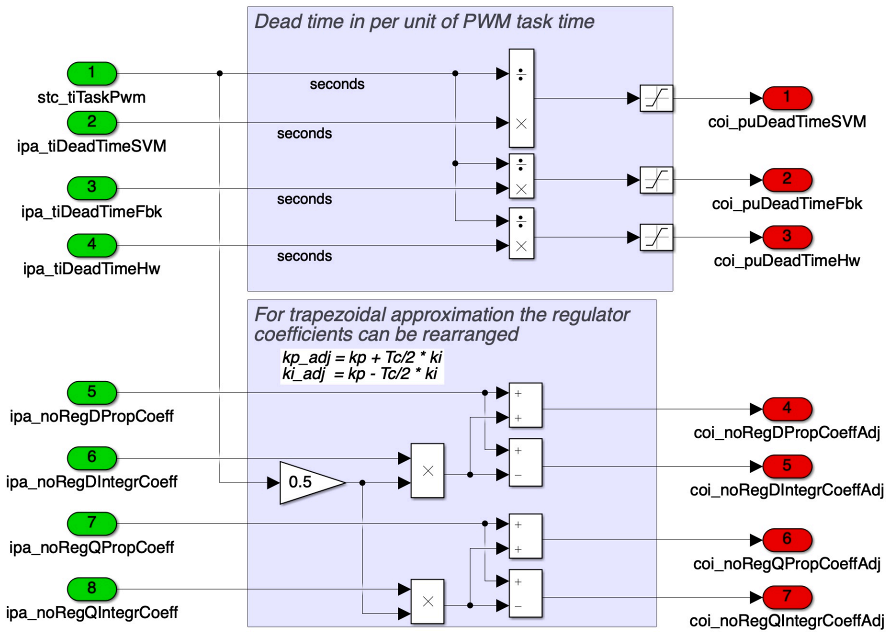


Figure 5.21: *COI function, overview.*

priority tasks are constants and therefore the block just verifies the proper ranges and derives the frequency from the period.

However, especially the PWM task, the basic software interrupts are the ones responsible in the application to generate the timing signals. This means that controls must be implemented inside this system to either align the tasks to the interrupts or command the basic software to apply the desired timing. This holds especially true in applications with variable frequency.

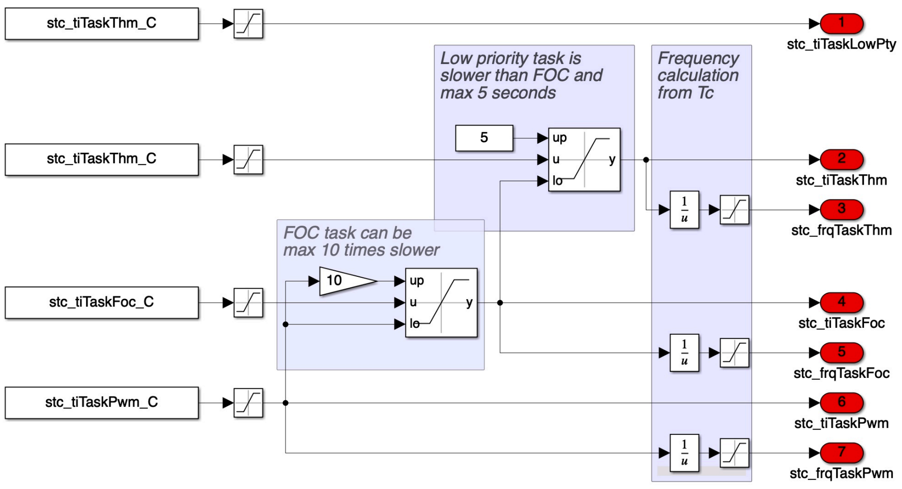


Figure 5.22: *STC function, overview.*

5.9 CMX CURRENT MAX

In this function the maximum stator current value allowed is provided. This value has its own input path because it could be used to play an important role in the derating or energy management algorithm by limiting the current used by the machine and, consequently, the power. For this reason it could be moved to higher priority tasks, if needed.

Fig. 5.23 shows the simple version used in the example, with a the calculation of the maximum magnitude from the RMS value and the predisposition to variation based on the temperature.

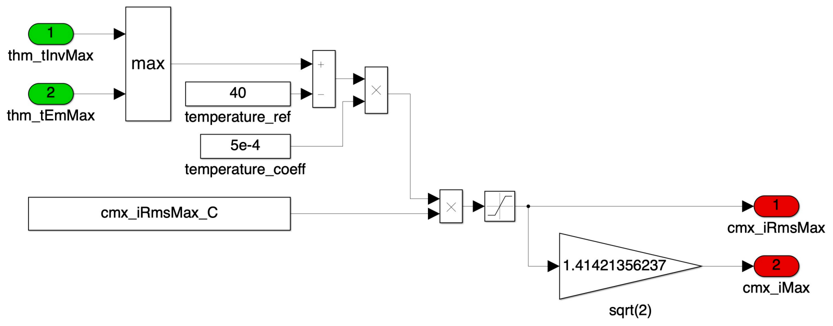


Figure 5.23: *CMX function, overview.*

5.10 SVE STATOR VOLTAGE ESTIMATOR

This function is responsible to acquire the voltage measurements and apply the Clarke transformation to obtain the two-phase equivalent system in the stator reference frame.

The transform equations are the ones presented in the previous chapter, (2.1).

In Fig. 5.24 showing the implemented example, it is shown that there might be different ways to obtain the voltages and they are part of the features of the model, since there could be better or more efficient ways to provide these variables based on the application and the software already accounts for some of them. In particular, the three option provided are:

- directly with measurements of the phase voltages via sensors or specific basic software functions;
- using the output phase voltage references of the previous cycle before the SVM block, meaning before a possible saturation and the application of the dead time corrections.

- using the already transformed values in the requested frame calculated in the VSM function that account for the dead time and the SVM saturation.

It is interesting to point out in this simpler example the timing of the variables entering the function and their name. In fact, the values coming from the previous cycle of the program are labelled with `Kp1` which, for the convention explained in section 2.2, means they refer to the next calculation step, or, in other words, are the predicted values. However, the source blocks generating thess inputs are executed in the previous sampling time with respect to the execution of this function. This means that, while for the generating function these are really predictive, their use as inputs in this function makes them the values for the current step in the context.

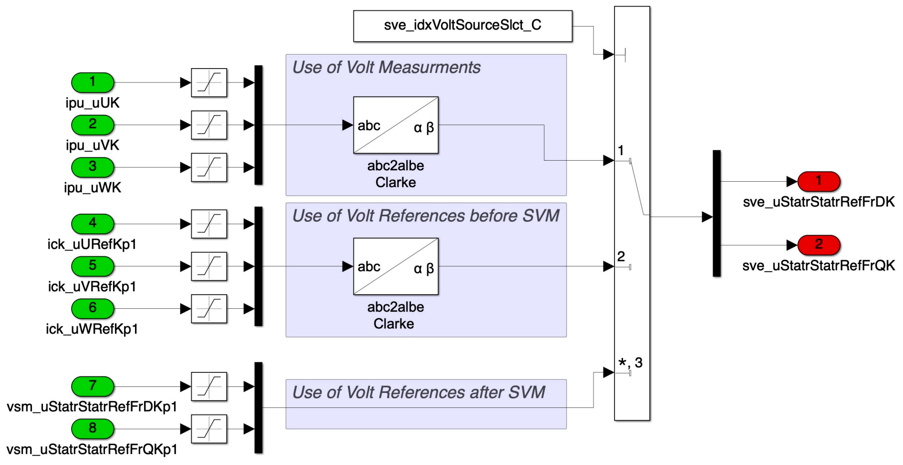


Figure 5.24: *SVE function, overview.*

5.11 ASE ANGLE SPEED ESTIMATION

This system provides the rotor angle position of the electric machine and is is also used for the speed calculation in different units.

Fig. 5.25 helps to visualize the various blocks of calculation carried out. First there is a selection between the resolver or encoder acquisition sensor depending on the application. Then the angle offset is applied to align the read angle with the permanent magnets flux, if the machine is Synchronous. This very important step allows the control to be certain that the d-q reference frame is correctly oriented with the existing flux and, therefore, the FOC algorithm can work properly as designed. The angle is kept in a finite range, which was chosen as $[0 - 2\pi]$, with the use of *mod* blocks.

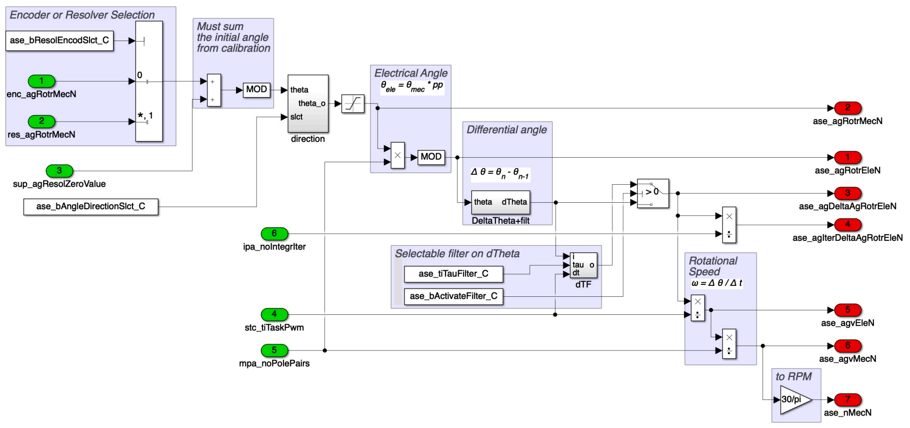


Figure 5.25: *ASE function, overview.*

Then there is the possibility of changing the rotation direction and this outputs the proper mechanical angle of the rotor. The next step is the division by the pole pairs to get the electrical angle, still in the same range of allowed values.

Knowing the position, applying a delay and subtracting the values it is possible to calculate the angle difference between time steps which is a value used to predict the next rotation value, assuming constant speed.

Dividing the delta position by the time period, the speed is calculated and expressed in electrical speed, mechanical or rpm.

It is worth reminding that the good knowledge of the rotor position is key for the correct operation of the control system in high performance applications.

5.12 CRK CLARKE TRANSFORM

This function was used as an example in previous sections and it was shown in Fig. 5.15.

Like SVE applies the Clarke transformation to the voltages, this one applies them to the phase current measurements. The only worthy comment is the optional selection of deriving the third current from the other two in case of the application requires it.

5.13 VUT UNIT VECTOR T

To apply the Park transformations and its inverse required by the control, the matrix operators [T] and $[T]^{-1}$ described in (2.3) and (2.4) must be defined. This matrices can also be interpreted as vectorial directions in the d-q reference frame, expressed through a unit vector described by the matrix coefficients.

Critically, the two matrices needed contain only two important values, the sine and cosine of the angle they refer to, meaning that it is not necessary to carry around the 8 terms to apply the transformations. By implementing them in a simplified way, only the two numbers are required.

Therefore, this function applies the trigonometric functions sine and cosine to the various angles needed. Different ways of approximating are provided as options depending on the hardware and environment used because this type of calculations are considered quite heavy and different application may have different optimized solutions to the problem.

The three angle transformations considered are the ones needed mainly in the Electric Machine model and are:

- the measured rotor electric angle;
- the predicted rotor electric angle assuming constant speed;
- the angle variation at each integrating step of the electric model solver.

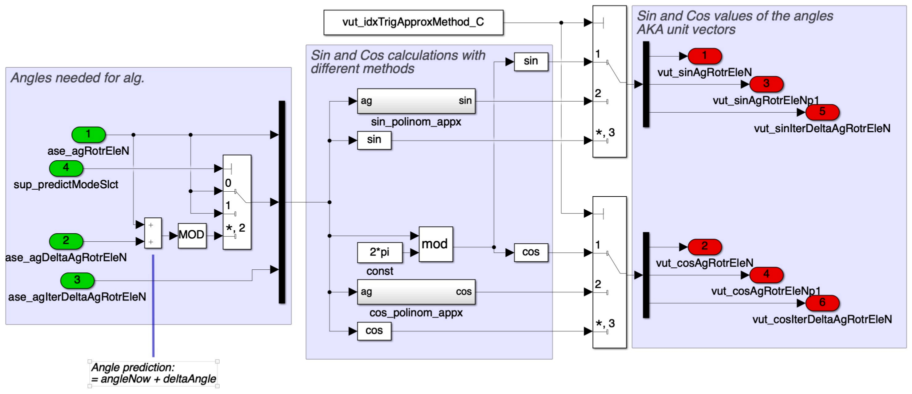


Figure 5.26: *VUT function, overview.*

5.14 EMS ELECTRIC MODEL SOLVER

This system incorporates the Unified Electric Machine Model and the integration solver to obtain and predict the machine fluxes and currents from the stator voltages. The detailed explanation of the model and also its discrete-time implementation is found in the dedicated chapter 3. In this section, instead, the more practical approach is discussed with some considerations that are given as already thoroughly elaborated in the main chapter.

In Fig. 5.27 the function implementation is shown in the Simulink environment and some parts that will be explored are highlighted for quick reference.

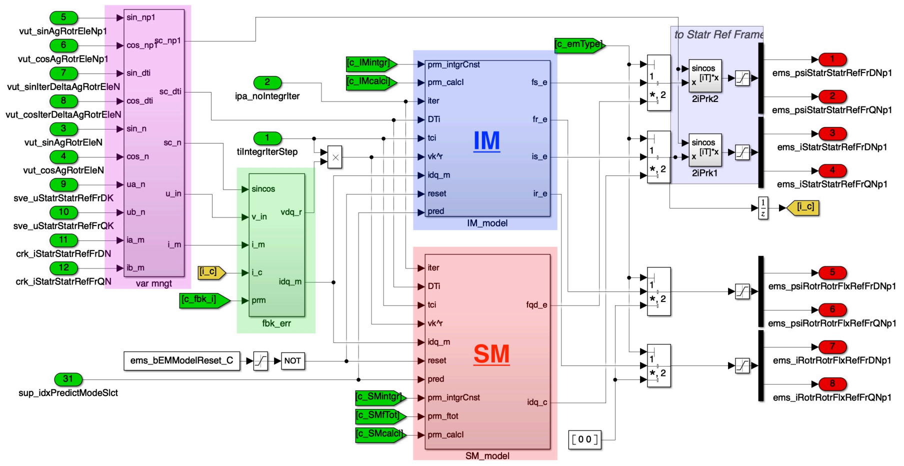


Figure 5.27: *EMS function, overview with highlighted the different parts.*

Contrary to the linear implementation, for nonlinear machine it has been necessary to divide some calculation between IM and SM, mainly to unload the SM part of many useless calculations. Also, it is worth noting that the linear approach is almost identical to the SM one, albeit with simpler and non-variable constants and the fact that the nonlinear integration is carried out directly in the rotor flux reference frame.

The first part worth pointing out has been highlighted in purple and, together with a similar one that has not been shown for space reasons, is just a block that collects some of the input variables (and the parameters for the unseen one) in easier to manipulate vectors.

The next parts are going to be discussed further in the following sections and they are:

RED the Synchronous Machine model implementation;

BLUE the Induction Machine model implementation;

GREEN the current feedback error correction.

The outputs of the functions are the same in every case and they are the 4 machine fluxes (stator and rotor) and 4 currents (stator and rotor) in their natural reference frames. Of course, for the SM, the rotor outputs are simply given as constant 0.

5.14.1 SM implementation

The function for the Synchronous Machine model is implemented as Fig. 5.28 and it revolves around the integration of the machine equation as explained in section 3.3.1.

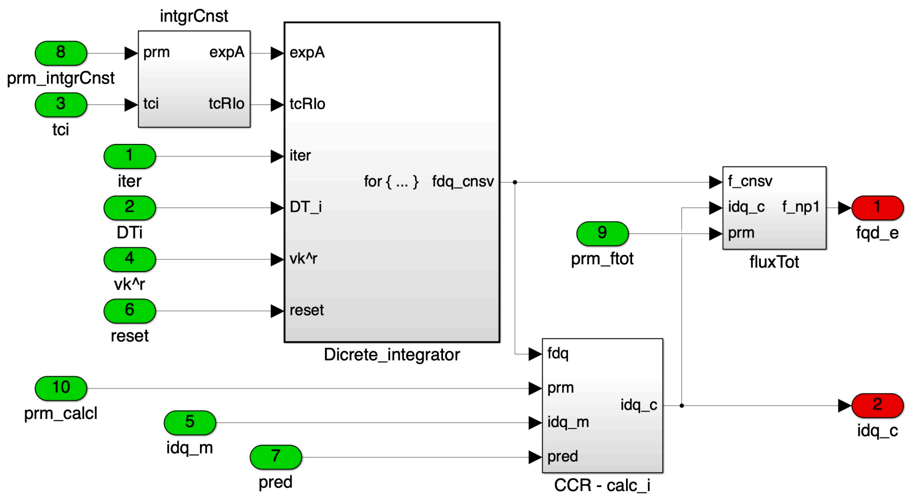


Figure 5.28: *EMS function, SM model.*

As stated, the fluxes are derived by integrating the voltages via a recursive discrete integrator which has been put in an iterative *for* loop. This allows the function to more precisely approximate the results while, simultaneously, not complicating the algorithm. The accuracy derives from the repetition of simple iterations and it has the added bonus of giving a tuning freedom to match the application.

The block first calculates the constants `expA` and `tcRIo` that do not change in the iteration step and then runs the integrator loop for the determined number of times.

The loop is shown in Fig. 5.29 and it applies the described equation (3.61) recursively, re-proposed underneath here for clarity.

$$\varphi_{i+1_{cnsv}} = \mathtt{expA}_{sym} \cdot \left([\Delta T_i] \left(t_c v_n + \varphi_{i_{cnsv}} \right) + \mathtt{tcRIo}_{sym} \right)$$

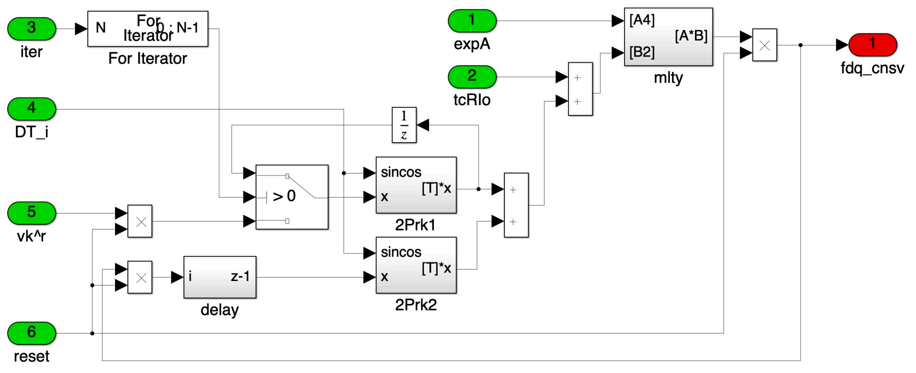


Figure 5.29: *EMS function, SM model integrator loop.*

Because it has been found that flux field is better approximated as non-conservative, the integration is carried out using the conservative part of the fluxes (via the symmetric inductance matrix) from which the currents can be calculated using the conservative nonlinear equation (3.60). The total fluxes are then obtained from (3.62) using the known currents.

This last step caused a small issue since the current calculation in the linear machine was carried out in its own atomic function CCR. In order to keep all the fluxes elaboration inside the same function, it has been decided to instead move the CCR inside this function. The idea was to keep simplify the model and reduce the number of functions and feedbacks.

Critically, the CCR function also operates part of the prediction algorithm selection. In fact, it has the possibility to pass through the measured values of the currents, which are by definition related to the current time step, or the ones calculated from the predicted fluxes, thus being referred to the next cycle. This option selection has to be coordinated with other parts of the control to ensure angle and measurements synchronization.

5.14.2 IM implementation

Apart from the addition of the rotor values, the overall implementation of the Induction Machine model shown in Fig. 5.30 is remarkably similar. They both have a pre-calculation of the integration constants, the *for* loop actuating the iterative integration and the last part for the evaluation of the currents.

It comes without saying that the constants are specific for the IM.

The first major visual change is after the CCR function, again moved inside this function for the reasons explained for the SM,

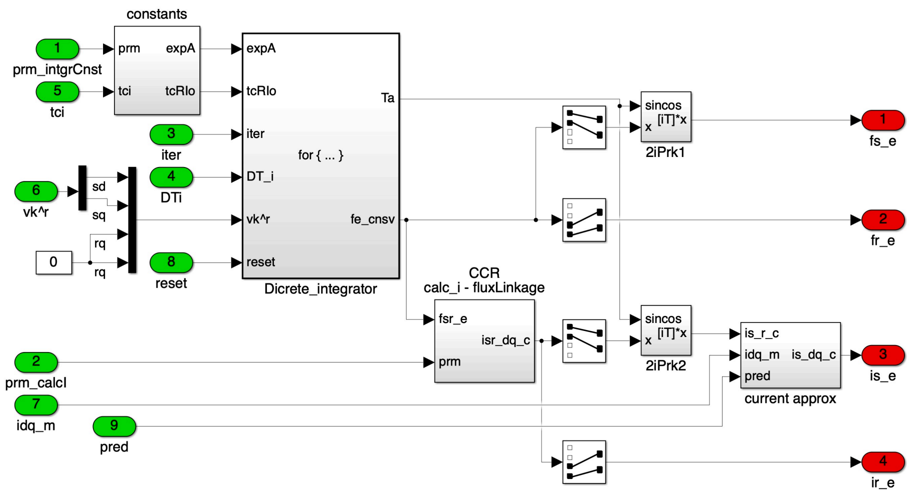


Figure 5.30: *EMS function, IM model.*

where the function seems to be split in two. This is actually true because the whole integration procedure is done in the rotor flux reference frame, as explained in the dedicated section 3.3.2, but the measurements of the currents are in the rotor one, as are the requirements for the outputs of the system. It is, therefore, necessary to first calculate the currents in the virtual reference frame where the equations are valid and then rotate back fluxes and currents before the signal's exit to the rotor one.

As for the SM, the predictive selection option for the currents is operated inside here, with the same warnings.

Following the explanation in the dedicated section, the implementation integration step for IM (3.69) is shown in Fig. 5.31.

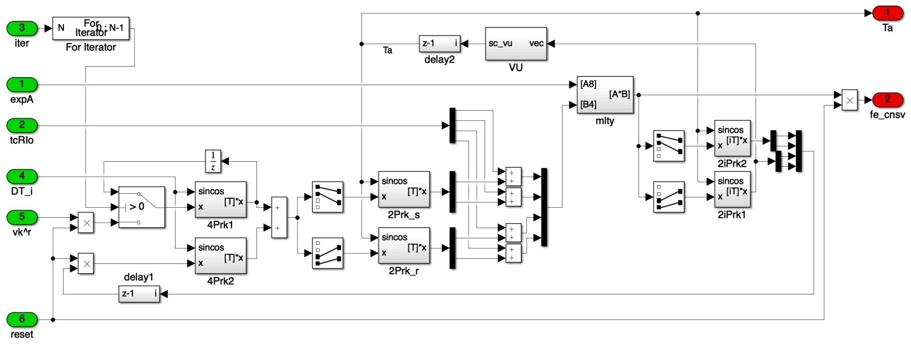


Figure 5.31: *EMS function, IM model integrator loop.*

As the equation suggests once again, the operations are similar between the machines, ignoring the addition of the rotor values. The major difference, obviously, is the fact that the output rotor fluxes are used to calculate their own direction angle to know the rotor flux reference frame rotation. The angle is then fed back through its sine/cosine to operate the Park transformation required and also gave as output to do the inverse at the end of the integration process.

5.14.3 Current error correction

This function, at the start of the EMS system, is used to make the system more precise and robust by evaluating the difference between the measured and calculated value of the currents and operating a feedback error correction to the input of the integration process. This way the error is actively controlled to be next to zero, thus improving the performance.

The implementation is shown in Fig. 5.32 and it looks complicated just because it features different options.

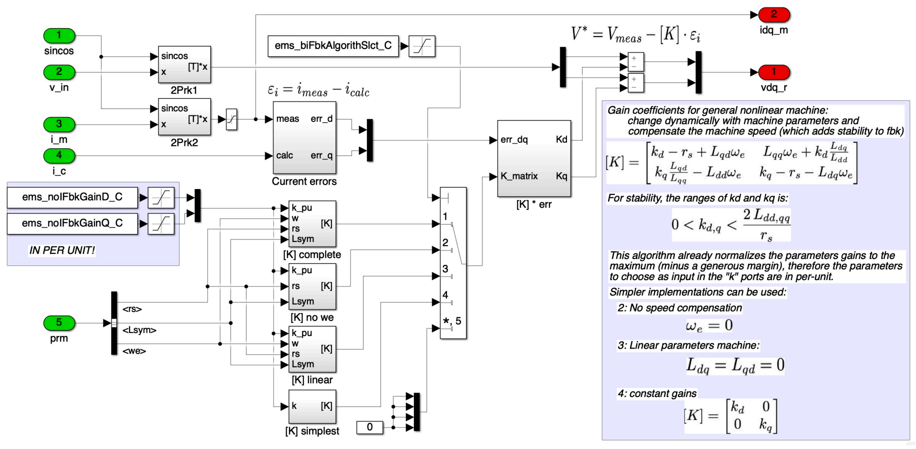

sincos
v_in
i_m
i_c
prm
2Prk1
2Prk2
ems_biFbkAlgorithSlct_C
$\varepsilon_i = i_{meas} - i_{calc}$
Current errors
$V^* = V_{meas} - [K] \cdot \varepsilon_i$
idq_m
vdq_r
err_dq
K_matrix
[K] * err
ems_noIFbkGainD_C
ems_noIFbkGainQ_C
IN PER UNIT!
[K] complete
[K] no we
[K] linear
[K] simplest
Gain coefficients for general nonlinear machine: change dynamically with machine parameters and compensate the machine speed (which adds stability to fbk)
For stability, the ranges of kd and kq is:
This algorithm already normalizes the parameters gains to the maximum (minus a generous margin), therefore the parameters to choose as input in the "k" ports are in per-unit.
Simpler implementations can be used:
2: No speed compensation
3: Linear parameters machine:
4: constant gains

Figure 5.32: *EMS function, current error correction.*

The integration inputs to be corrected are the voltages and their correction is based on the error of the currents, all expressed in the rotor reference frame. This type of feedback is a variation of the Luenberger state observer.

The main output $\boldsymbol{v}^*$ is then defined with (5.1) in vector form.

$$\boldsymbol{v}^* = \boldsymbol{v}_{\text{in}} - [\text{K}]\boldsymbol{\varepsilon}_{\text{i}} \tag{5.1}$$

Where $\boldsymbol{\varepsilon}_{\text{i}}$ is the vector of the differences between the measured and calculated currents on the two axes, as expressed in (5.2).

$$\boldsymbol{\varepsilon}_{\text{i}} = \begin{cases} \varepsilon_{\text{id}} = \text{i}_{\text{d}_{\text{meas}}} - \text{i}_{\text{d}_{\text{calc}}} \\ \varepsilon_{\text{iq}} = \text{i}_{\text{q}_{\text{meas}}} - \text{i}_{\text{q}_{\text{calc}}} \end{cases} \tag{5.2}$$

The definition of the correction coefficient matrix [K] is expressed in its general form as (5.3) and it is carried out considering the variable nonlinear machine parameters and compensates for the speed to add stability and robustness.

$$[K] = \begin{bmatrix} k_d - r_s + L_{qd}\omega_e & L_{qq}\omega_e + k_d\frac{L_{dq}}{L_{dd}} \\ k_q\frac{L_{qd}}{L_{qq}} - L_{dd}\omega_e & k_q - r_s - L_{dq}\omega_e \end{bmatrix} \tag{5.3}$$

The calibration of the gain matrix is done through the parameters k_d and k_q and they present a stability range presented in (5.4), thus making tuning a little bit easier.

$$\begin{cases} 0 < k_d < \frac{2L_{dd}}{r_s} \\ 0 < k_q < \frac{2L_{qq}}{r_s} \end{cases} \tag{5.4}$$

As mentioned, this function presented various options to change the gain matrix type to suit different applications that may require less error correction because the knowledge of the parameters is better, the dynamic response is less important or the machine is operated only in the linear region.

The four options are:

1. full [K] matrix;
2. no speed correction: $\omega_e = 0$;
3. machine with linear parameters: $L_{dq} = L_{qd} = 0$;
4. constant gains:
$$[K] = \begin{bmatrix} k_d & 0 \\ 0 & k_q \end{bmatrix}.$$

5.15 CCR CURRENT CORRECTION

This function in the nonlinear implementation is kept empty because it was necessary to move it inside the EMS model.

For linear machines, this is the place where the current equation (3.14) described in 3.1.1 is applied using the machine parameters and the fluxes calculated by the Electric Model solver.

5.16 TQE TORQUE ESTIMATION

This function, Fig. 5.33, is used to produce a torque estimation from the state of the machine derived from the fluxes of the EMS function and the currents of the CCR one.

The equation applied is the general one (3.1) that works for both linear and nonlinear machines described in section 3.1.

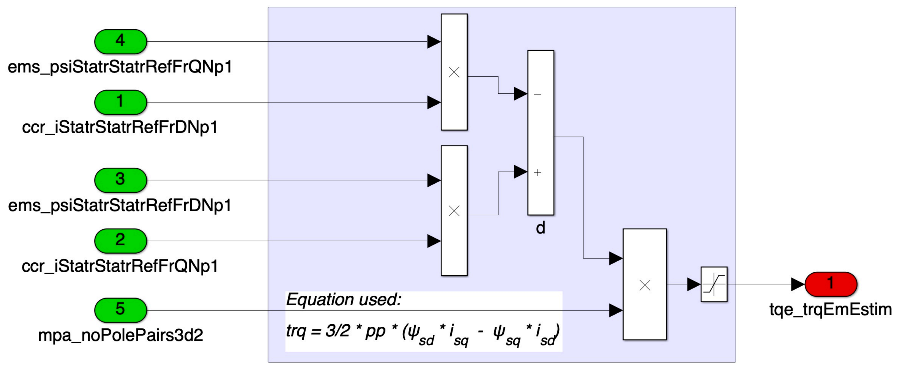


Figure 5.33: *TQE function, overview.*

5.17 ACR ALPHA CORRECTION

In summary, this function applies the Park transformation to the machine's currents to be confronted with the requested ones from the FOC at the input of the voltage PI regulators. Fig .5.34 shows the implementation.

The equation applied is the same as the one described in the general description of the algorithm which is (2.3), but it is not explicitly use in the matrix form to optimize the execution. In fact, in this way, only the sine and cosine values of the rotation angle are needed, and not all the four terms of the matrix. The outputs are, of course, identical.

The angle of orientation of the rotor flux is the one coming from the FOC task. This is because the algorithm architecture decided upon is constructed to maintain the electric machine model in the natural reference frame as much as possible. It implies that, in the PWM task until this point in the execution order, the stator currents either measured or estimated are still in the two-phase equivalent stator reference frame.

The rotor flux and its direction is instead arguably needed only in the FOC task, so there is where it is calculated and its position is then distributed from there, specifically function FOA, 5.27.

For nonlinear machines, this distinction is no longer completely valid mainly because the flux characteristic is reasonably found only in the d-q reference frame, meaning that, as explained in the dedicated section 3.3, the model is now using the rotating frame, too. However, it has been decided to keep the linear output framework by operating the inverse transform inside the machine model solver. The main reason behind the decision was that, at this stage, a small execution inefficiency by using a double transformation was deemed not critical for the development and it allows to keep perfect compatibility with the linear case which is still relevant for many applications.

Of course the voltage regulators must operate in the rotating d-q reference frame, so this function operates the Park transformation of the stator current vector.

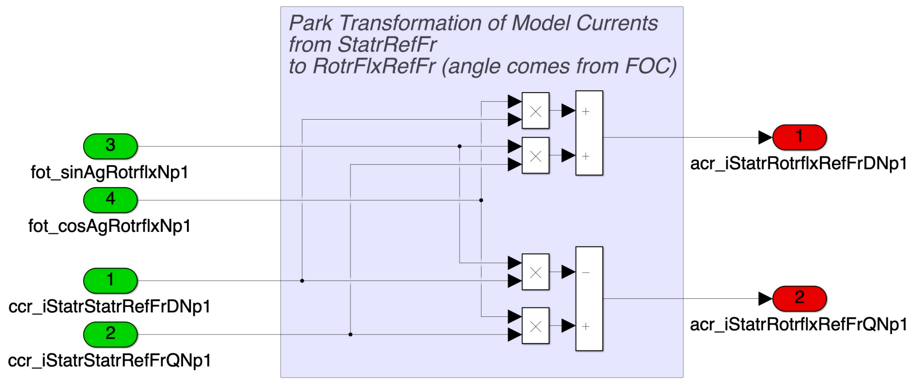

Park Transformation of Model Currents
from StatrRefFr
to RotrFlxRefFr (angle comes from FOC)
3
fot_sinAgRotrflxNp1
4
fot_cosAgRotrflxNp1
1
ccr_iStatrStatrRefFrDNp1
2
ccr_iStatrStatrRefFrQNp1
1
acr_iStatrRotrflxRefFrDNp1
2
acr_iStatrRotrflxRefFrQNp1

Figure 5.34: *ACR function, overview.*

5.18 EMF ELECTROMOTIVE FORCE

This function operates the back electromotive force compensation of the voltage regulators, also known as decoupling, and it is presented in Fig. 5.35.

The voltage applied by the control is generated in feedback by the regulators based on the voltage equation in the rotor flux reference frame (3.15) which, as also discussed in the FOC section 4.4, is composed of three terms: the voltage drop on the resistances, the term related to the derivative of the fluxes and the electromotive force `emf`.

It is common practice, then, to separately calculate the `emf` portion of the voltage to be applied to compensate the regulators and making them work more efficiently since, in this way, they have to regulate a smaller part of the voltage. Another reason to do this is that, looking at the equation, it can be seen how the `emf` terms are cross-coupled to the control axes. Therefore, by compensating for it, the two axes as controlled are completely separated or decoupled.

The equation implemented is a small variation to the FOC one and directly obtained from the general one (3.15), yielding to (5.5).

$$\begin{cases} \mathtt{emf}_d = -\omega_e \varphi_q \\ \mathtt{emf}_q = \omega_e \varphi_d \end{cases} \tag{5.5}$$

Where the fluxes are coming from the electric model are are, therefore, still in the stator reference frame, so they have to be rotated in the rotor flux frame with the use of a Park transformation.

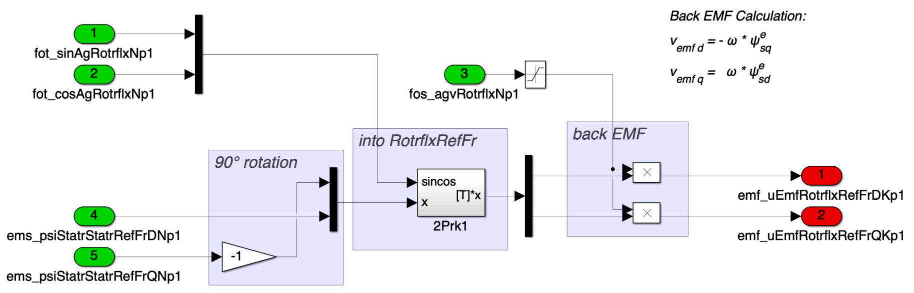


Figure 5.35: *EMF function, overview.*

5.19 VRG VOLTAGE REGULATOR

In Fig. 5.36 is shown the implementation of the main voltage regulators in the algorithm under scrutiny.

In here, the machine currents in the rotor flux reference frame, either measured or estimated, are compared (subtracted from) to the reference ones generated by the FOC task which define the target machine state that would deliver the desired performance inside the safety limits.

The error is then fed to PI regulators to generate the voltage references on the d and q axes.

The regulators are implemented using the recursive form (5.6) and are saturated by re-calculation of the state from the final outputs using function VRS, which means after the SVM operates its corrections. This method automatically applies an anti-windup feature since the state is never allowed to overshoot. For further information, Appendix B comprehensively explains the discrete regulators implementation, their calibration and possible issues.

$$v_n = \left(K_p + \frac{t_c K_i}{2}\right)\varepsilon_n - \left(K_p - \frac{t_c K_i}{2}\right)\varepsilon_{n-1} + v_{n-1} \tag{5.6}$$

5.20 ICK INVERSE CLARKE TRANSFORM

After the voltage regulators generate the voltage reference in the d-q reference frame, this function applies the inverse Park and

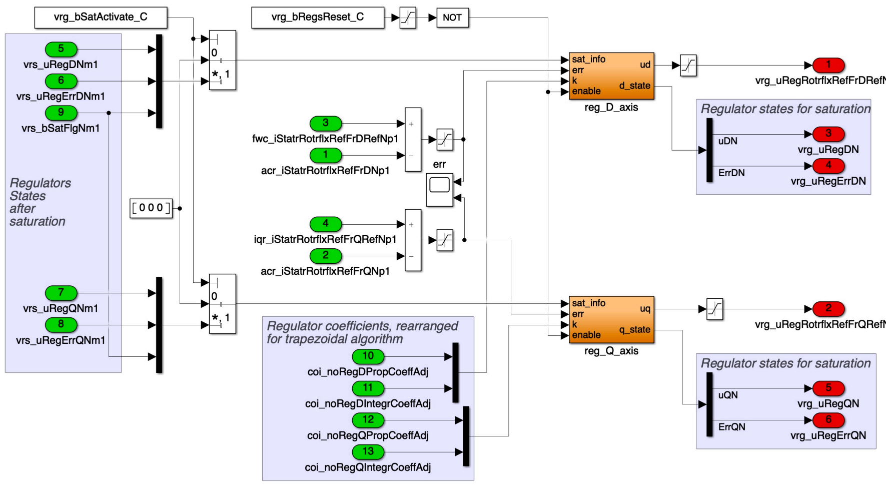


Figure 5.36: *VRG function, overview.*

Clarke transforms to them to obtain the references in the real three-phase system. Fig. 5.37 shows the example implementation.

This is also the system where the back electromotive force compensation is carried out by adding the `emf` values, calculated in the *EMF* function, to the regulators outputs, thus obtaining the total voltage to be applied.

Another key aspect in this block is the choice of the rotation angle to be applied in the Park transformation. In fact, two aspects must be considered:

1. this voltages will be applied to the machine in the next PWM cycle;

2. the inverter is able to apply only an average voltage along the PWM period while the machine is still rotating.

The first consideration is related to the possibility of using a predictive algorithm to generate directly the voltage references for the next cycle. It means that there is a selector in this block to choose if this predictive algorithm is active, inactive or a half way point where there is no prediction of the voltage values, but the angle extra rotation is applied anyway. These operating modes must be coordinated with the model integration and the FOC task.

The second consideration implies that the voltage references calculated are technically valid only at the start of the PWM cycle where the angle is correct. However, during the time step the rotor flux keeps rotating and the voltages get increasingly more out of phase. The solution to try to solve this error is to rotate the transformation angle of an extra half of the delta rotation, thus approximately centering the voltage applied to half-way of the PWM cycle.

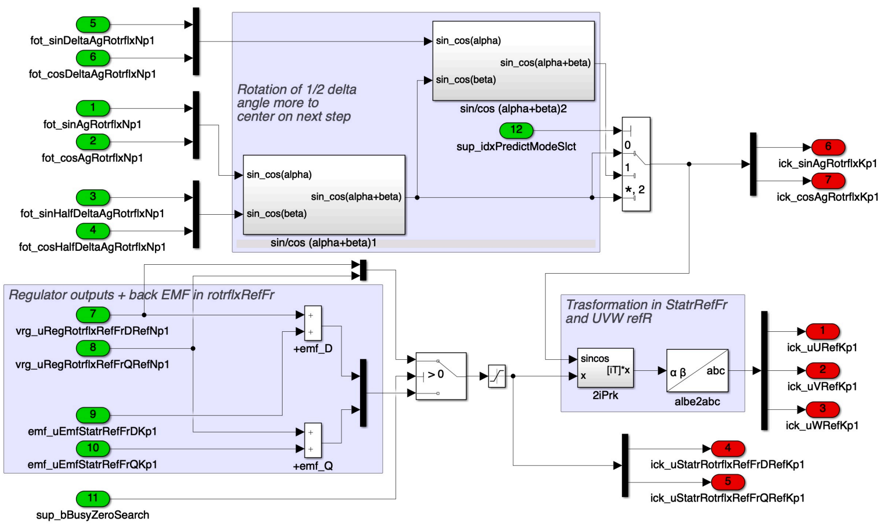


Figure 5.37: *ICK function, overview.*

5.21 DTI DEAD TIME INTRODUCTION

The dead time is a delay that has to be introduced in all inverter legs between the turn off of one switch and the turn on of the other to avoid short circuit issues. This is best described through the Fig. 5.38 where it is obvious that this delay implies that the real voltages applied are different than the ones requested which might cause problems.

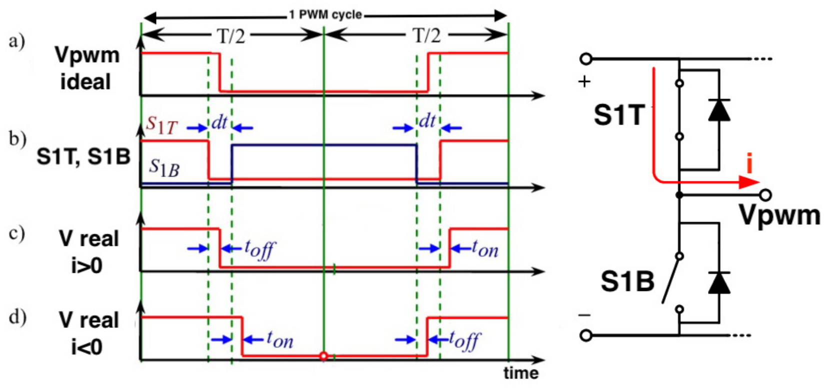


Figure 5.38: *Explanation of the dead time delay to be introduced to avoid short circuits in the inverter legs.*

At the control level the dead time represents a virtual decrease or increase of the Duty Cycle depending on the current direction. In fact, if the current is positive the dead time leads to a reduction of the real voltage applied, while for negative current it means an increase of the desired voltage. This, in turn, creates distortions in the voltages and currents outputs, shown in Fig. 5.39.

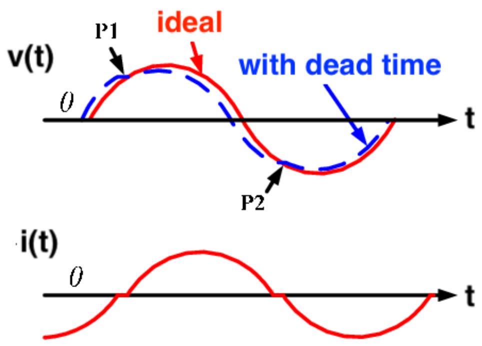


Figure 5.39: *Example of the distortions created by the dead times.*

Luckily, the dead time is a known inverter parameter as are the current directions, so the control can compensate this distortion by summing the dead time value to the phase duty cycle if the current on that phase is positive or subtracting it if negative. As summarized in (5.7) where j is the phase.

$$\begin{cases} D.C._{j} = D.C._{j} + \mathtt{dt} & \text{if} \quad i_j > 0 \\ D.C._{j} = D.C._{j} - \mathtt{dt} & \text{if} \quad i_j < 0 \end{cases} \tag{5.7}$$

To the inverter hardware, then, the voltage references (duty cycles) will arrive slightly different from the desired ones because, once the hardware operates the switches it must insert the dead times and the actual applied voltages will become the correct ones.

The saturations to be implemented in the SVM function to ensure the voltage stays inside the allowed space must, therefore, be done after the compensation. However, to know the true voltage applied for the next cycle this compensation must be removed from the actual duty cycles that went to the inverter. In the end, the schematic view of the compensation is shown in Fig. 5.40.

The function DTI (Dead Time Introduction), implemented as in Fig. 5.41, calculates the values in per unit with respect to the current directions of the compensations to be applied by the two subsequent functions.

Extra care is applied when the current is close to 0 because in that region the true direction cannot be assumed correct for noise and measurement uncertainty reasons. The switch between

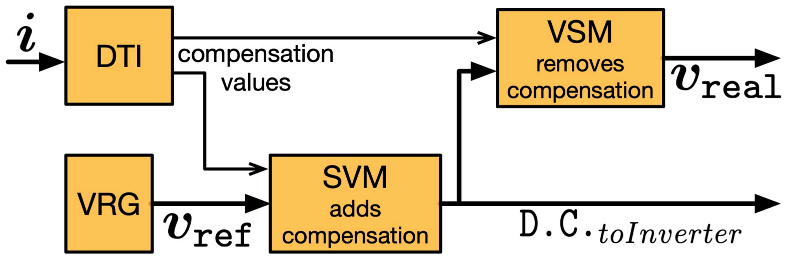


Figure 5.40: *Dead time compensation diagram showing where and how it is applied in the algorithm.*

positive and negative compensation is therefore carried out using a simple lookup table that reduces the compensation amount gradually to zero the closer the current to zero.

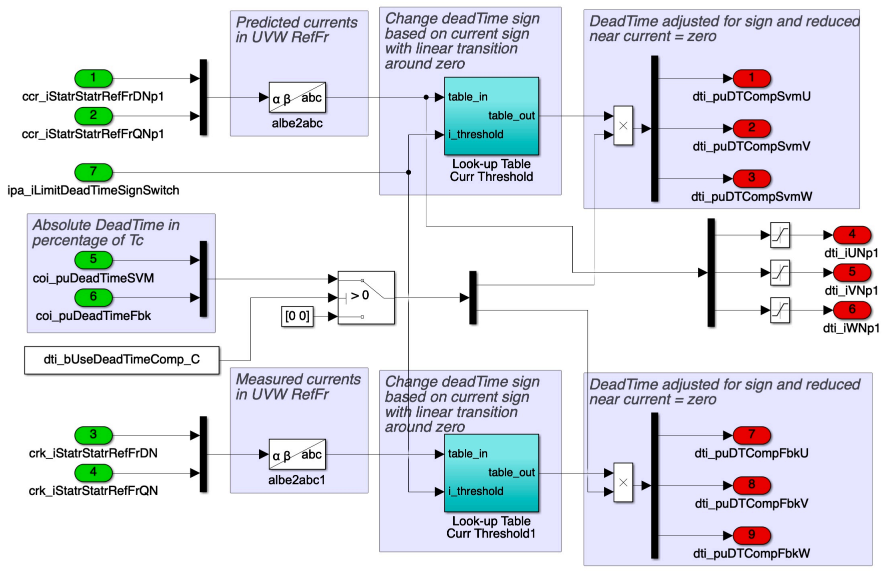

Predicted currents in UVW RefFr
Change deadTime sign based on current sign with linear transition around zero
DeadTime adjusted for sign and reduced near current = zero
1 ccr_iStatrStatrRefFrDNp1
2 ccr_iStatrStatrRefFrQNp1
7 ipa_iLimitDeadTimeSignSwitch
albe2abc
table_in
table_out
i_threshold
Look-up Table Curr Threshold
1 dti_puDTCompSvmU
2 dti_puDTCompSvmV
3 dti_puDTCompSvmW
Absolute DeadTime in percentage of Tc
5 coi_puDeadTimeSVM
6 coi_puDeadTimeFbk
[0 0]
> 0
dti_bUseDeadTimeComp_C
4 dti_iUNp1
5 dti_iVNp1
6 dti_iWNp1
Measured currents in UVW RefFr
3 crk_iStatrStatrRefFrDN
4 crk_iStatrStatrRefFrQN
albe2abc1
Look-up Table Curr Threshold1
7 dti_puDTCompFbkU
8 dti_puDTCompFbkV
9 dti_puDTCompFbkW

Figure 5.41: *DTI function, overview.*

5.22 SVM SPACE VECTOR MODULATION

This function transforms the voltage references in the duty cycles to be applied by the inverter.

It does this using a modulation technique known as SVPWM (Space Vector Pulse Width Modulation), but is has other features. Fig. 5.43 presents the implementation under study.

The modulation is simply obtained by dividing the phase voltages by the DC bus value. This automatically produces per unit

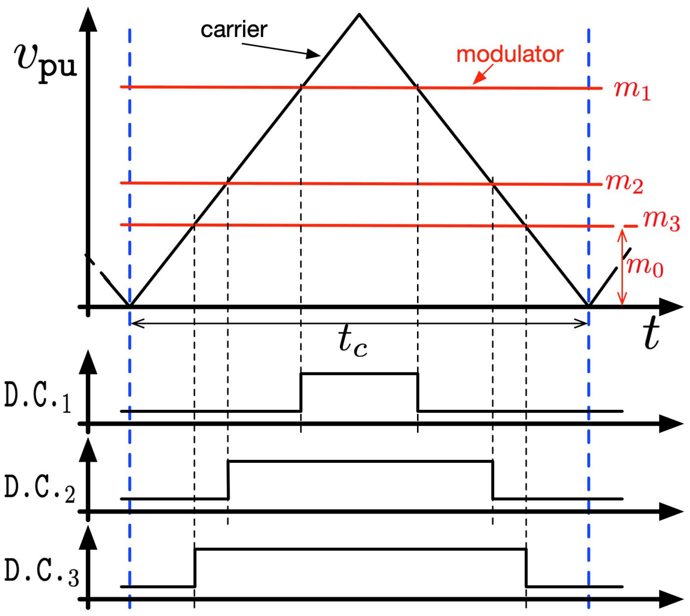


Figure 5.42: *SVM function, PWM modulation schematic.*

signals representing the modulation ratios m, as schematized in Fig. 5.42. In reality these are values in the range $[-1, 1]$, but their relative distance is the important information.

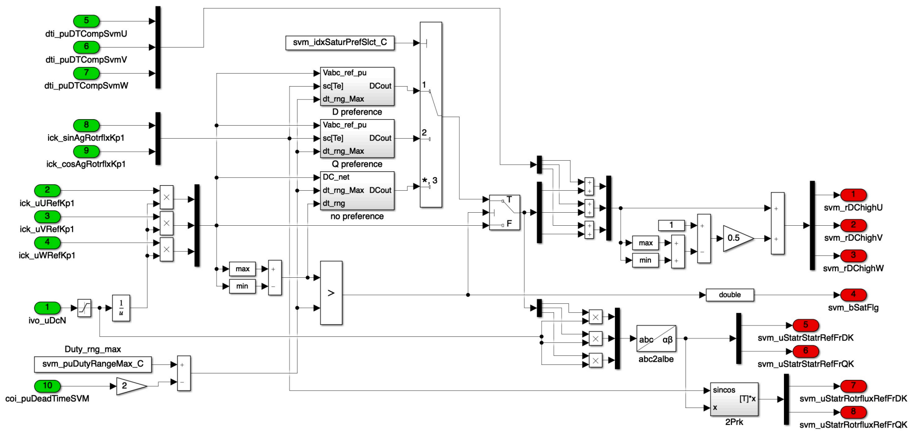


Figure 5.43: *SVM function, overview.*

To convert the modulators in duty cycles it is sufficient to rescale them in the range $[0, 1]$. The general approach has a degree of freedom, presented as m_0 in the figure, that allows the choice of various methods. The one adopted for this example is called *7 segments* or *SVPWM* and it is carried out by centering the three modulator in the middle of the range which also means that the distance between the lower bound to the lower modulator is equal to the distance between the upper bound and the higher modulator.

This is usually regarded as one of the best solutions to reduce the current ripple, but it requires 6 distinct commutation events, increasing the losses. Another technique is the so called *5 segments* where the highest or lowest modulator is kept at the boundary, thus keeping it from commuting and potentially reduce the losses.

In this function the dead times are also compensated to ensure that the real voltages applied by the hardware correspond to the reference ones. To do this, the dead time correction values are added (with sign) to the duty cycles and the reasoning is explained better in the DTI section 5.21.

Another important feature is the saturation of the voltages to the hexagonal limit in the three-phase reference system. The first step is to compare the difference between the maximum and minimum modulator against the maximum range allowed, minus the dead time. If it is bigger, then it means that the reference voltage cannot be applied and must be reduced.

In the implemented function, this reduction can be executed with different types of priorities. Referring to Fig. 5.44 the voltage request (blue) can be saturated inside the limit expressed by the hexagon in the three-phase reference system without any axis preference, keeping the same direction (purple). Otherwise, the reduction can try to reduce less the d axis voltage (green) or the q axis (red) giving *preference* to one over the other.

The practical advantages of one over the other of these systems is very minor and could even promote issues in particular cases. Moreover, the calculation complexities introduced make the saturation without axis preference the best option in all practical cases.

5.23 VSM VS MAX

The name of this function is a little misleading, but it was not changed from the original branding. It does not really calculate the maximum voltage, but it elaborates the duty cycles applied to the machine to obtain the actual voltages applied accounting for the dead time.

As described extensively in the DTI description 5.21, the dead time introduces a distortion in the applied voltages that can be compensated, but then the SVM could saturate the actual output. Therefore, to know the real applied values, this block starts from

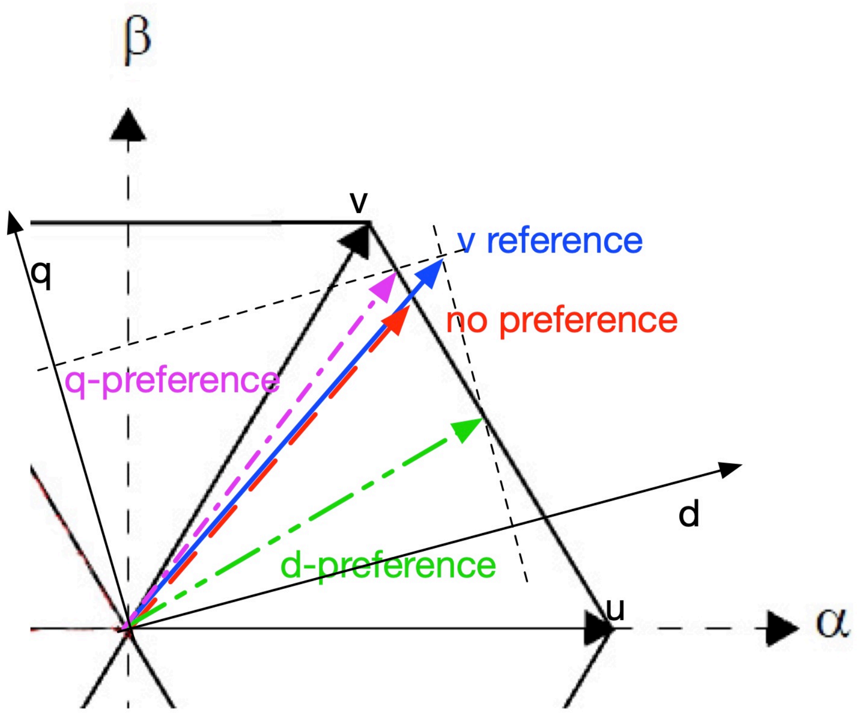


Figure 5.44: *SVM function, visualization of preferential saturation of the voltage based on the rotating axes.*

the SVM duty cycles and calculates back the voltages subtracting the dead time compensation.

Fig. 5.45 shows also how there is an option to compensate for the predictive nature of the algorithm, if necessary. The outputs are then delivered in both three-phase and two-phase reference systems.

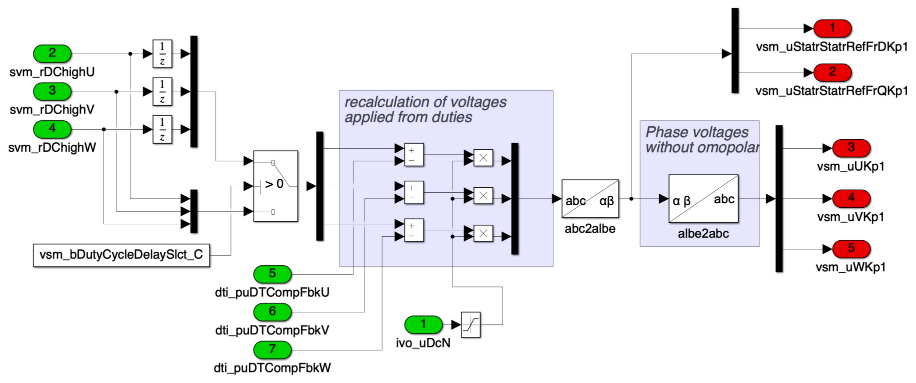


Figure 5.45: *VSM function, overview.*

5.24 VRS VOLTAGE REGULATOR SATURATION

This function operates the voltage regulators saturation and anti-windup by re-calculating the correct regulator state after the possible saturation of the voltages carried out by the SVM function.

Fig. 5.46 illustrates the overall algorithm implemented and how the outputs of this block are actuated in the next cycle, thus losing a step-time in their name.

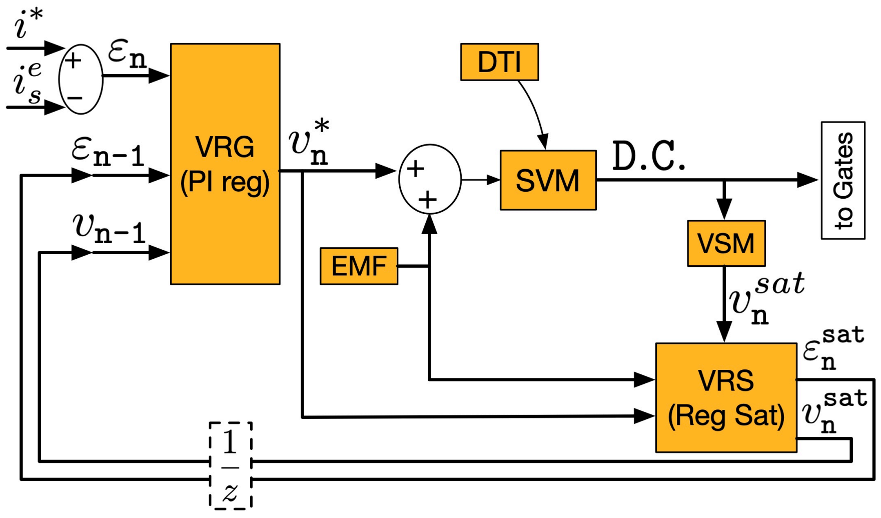


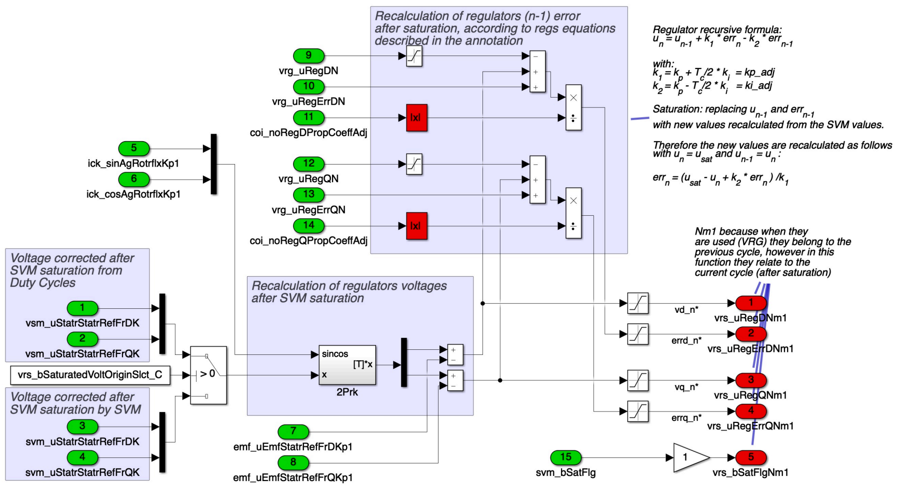
Figure 5.46: *Regulator saturation implementation showing the overall signals flow.*

In practice this block takes the real voltages going to the inverter after the SVM has applied its limits and recalculates the regulators error based on the new saturated values, using (5.8) and shown in the Fig. 5.47.

$$\varepsilon_n^{sat} = \frac{v_n^{sat} - v_{n-1} + \left(K_p - \frac{t_c K_i}{2}\right)\varepsilon_{n-1}}{\left(K_p + \frac{t_c K_i}{2}\right)} \tag{5.8}$$

To do this, of course, it needs to re-transform the stator voltages in the rotor flux reference frame and it must subtract the `emf` compensation to obtain the proper voltage exiting the regulators.

The new error and voltage value will then be used in the regulator implementation as starting states, thus limiting the error and ensuring that no windup can occur. For further information, Appendix B explores the specific discrete implementation and calibration of the regulators.



Recalculation of regulators (n-1) error after saturation, according to regs equations described in the annotation
Regulator recursive formula: $u_n = u_{n-1} + k_1 * err_n - k_2 * err_{n-1}$
with: $k_1 = k_p + T_c/2 * k_i$ = kp_adj; $k_2 = k_p - T_c/2 * k_i$ = ki_adj
Saturation: replacing u_{n-1} and err_{n-1} with new values recalculated from the SVM values.
Therefore the new values are recalculated as follows with $u_n = u_{sat}$ and $u_{n-1} = u_n$:
$err_n = (u_{sat} - u_n + k_2 * err_n) / k_1$
Nm1 because when they are used (VRG) they belong to the previous cycle, however in this function they relate to the current cycle (after saturation)
Voltage corrected after SVM saturation from Duty Cycles
Voltage corrected after SVM saturation by SVM
Recalculation of regulators voltages after SVM saturation
vrg_uRegDN
vrg_uRegErrDN
coi_noRegDPropCoeffAdj
vrg_uRegQN
vrg_uRegErrQN
coi_noRegQPropCoeffAdj
ick_sinAgRotrflxKp1
ick_cosAgRotrflxKp1
vsm_uStatrStatrRefFrDK
vsm_uStatrStatrRefFrQK
vrs_bSaturatedVoltOriginSlct_C
svm_uStatrStatrRefFrDK
svm_uStatrStatrRefFrQK
emf_uEmfStatrRefFrDKp1
emf_uEmfStatrRefFrQKp1
2Prk
svm_bSatFlg
vrs_uRegDNm1
vrs_uRegErrDNm1
vrs_uRegQNm1
vrs_uRegErrQNm1
vrs_bSatFlgNm1

Figure 5.47: *VRS function, overview.*

5.25 SPD SET PWM DUTY

This function, shown in Fig. 5.48, is used to tailor the duty cycle output to the hardware requirements. For example, in this case, both duty cycles for the high and low switch are calculated with the relative "hardware" implementation of the dead time.

Some application may even require the actual commutation instants (off to on and on to off) to be calculated and this would be the place to put such functions.

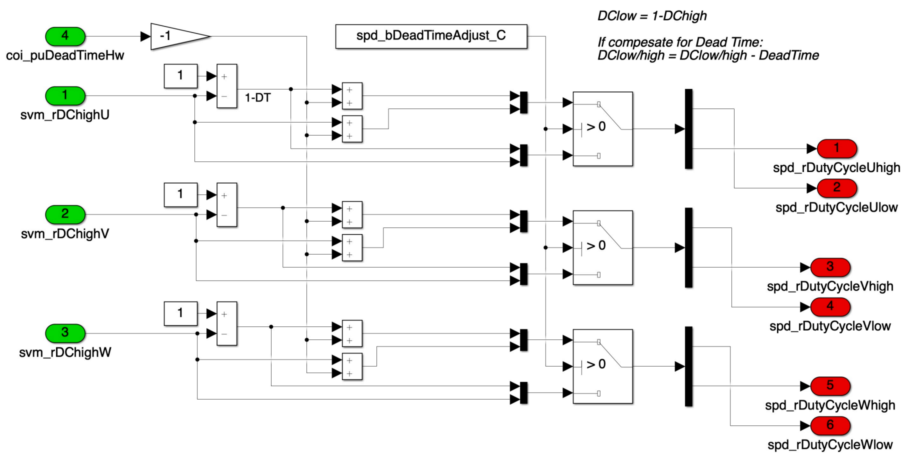


Figure 5.48: *SPD function, overview.*

5.26 SUP STATE MACHINE START UP

This is another block that partially operates as a placeholder for more specific application's requirements. It would contain the state machines that control the modes and behaviours of the algorithm, triggering particular states and taking care of the proper start up of the system.

In the case of Fig. 5.49, it is presented a possible state machine that, given a user input, activates the resolver zero procedure, automatically saves the value and exits the state.

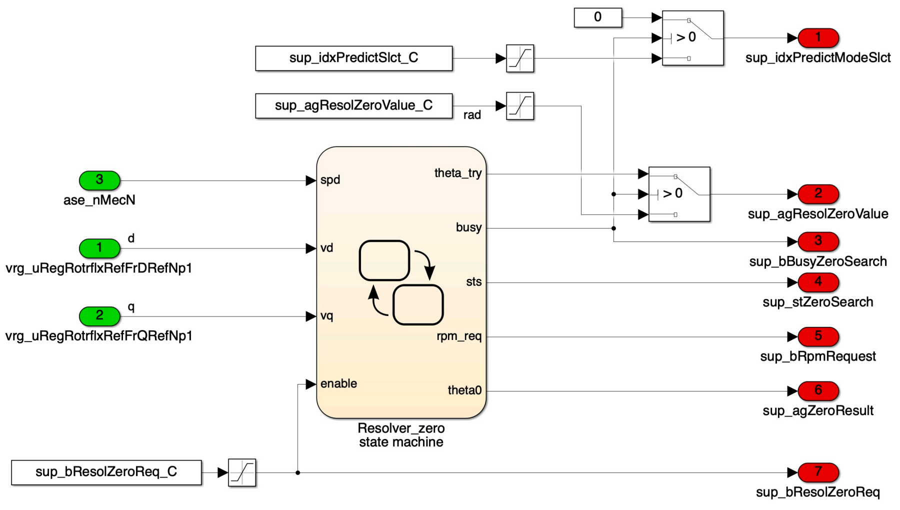


Figure 5.49: *SUP function, overview.*

5.27 FOA FIELD ORIENTED ANGLE

This is the function that calculates the rotor flux angle and starts the Field Oriented Control.

Of course, for Synchronous Machine the thing is quite easy because it coincides with the electrical rotor angle. For this, as seen in Fig. 5.50, in the block there is a selector between SM and IM machines.

For Induction Machines, the equation that provides the angle makes use of the fluxes and currents available from the Electric Machine model, also known as flux observer for this reason, and it is (5.9).

$$\vartheta^e = \tan^{-1}\left(\frac{\varphi^s_{sq} - i^s_{sq} \cdot \sigma L_{sq}}{\varphi^s_{sd} - i^s_{sd} \cdot \sigma L_{sq}}\right) \tag{5.9}$$

The second important output of this block is the delta angle: the difference between the current angle and the previous measurement. This is of course tied with the speed of rotation and it is also used inside here to operate the predictive algorithm assuming constant speed. This last feature must be accompanied by other adjustments in the machine model and the inverse transformation of the voltages.

As a reminder, the calibration alignment of the angle acquisition with the mechanical rotor angle is not necessary in here. It is implemented in the ASE function and the values coming to this one are all already accounting for it.

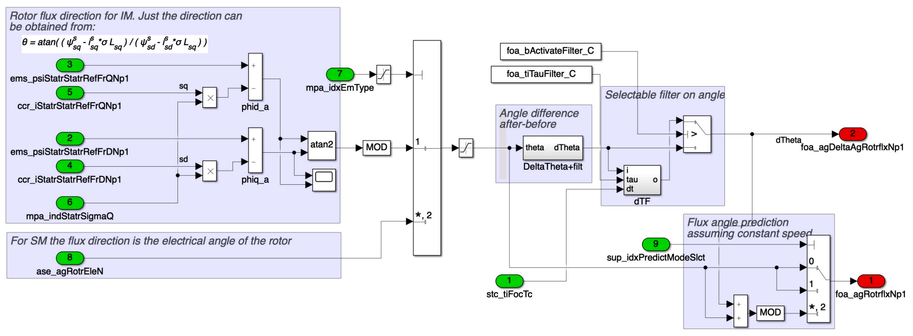

Rotor flux direction for IM. Just the direction can be obtained from:
$\theta = atan((\psi^s_{sq} - i^s_{sq} * \sigma L_{sq}) / (\psi^s_{sd} - i^s_{sd} * \sigma L_{sq}))$
3
ems_psiStatrStatrRefFrQNp1
5
ccr_iStatrStatrRefFrQNp1
sq
phid_a
2
ems_psiStatrStatrRefFrDNp1
4
ccr_iStatrStatrRefFrDNp1
sd
phiq_a
6
mpa_indStatrSigmaQ
atan2
MOD
7
mpa_idxEmType
1
*, 2
For SM the flux direction is the electrical angle of the rotor
8
ase_agRotrEleN
foa_bActivateFilter_C
foa_tiTauFilter_C
Angle difference after-before
theta dTheta
DeltaTheta+filt
Selectable filter on angle
i
tau
dt
o
dTF
dTheta
2
foa_agDeltaAgRotrflxNp1
1
stc_tiFocTc
Flux angle prediction assuming constant speed
9
sup_idxPredictModeSlct
0
1
*, 2
MOD
1
foa_agRotrflxNp1

Figure 5.50: *FOA function, overview.*

5.28 FOT FIELD ORIENTED TRANSFORM

The function is the same as the VUT one, but for the FOC task and the rotor flux angle. The goal is to construct the rotation matrix to enable the Park transformation, but again the optimization requires the calculation of just the sine and cosine of the angle, and not the full matrix.

As it can be seen in Fig. 5.51 there are multiple angles important for the calculations. The first is the rotor flux angle, predicted or not depending on the selection via the configuration file. then there is the half of the variation of the angle in the time step, which is useful to operate the inverse Park transformation on the output voltage to center the request in the middle of the rotation to average out the errors, as explained in section 5.20. The third is the delta variation of the angle in the sampling period.

Like in the VUT function, there are implemented several different trigonometric approximation to be decided based on the application. There is also a selectable option to use only 2 sin/cos and deriving the 3rd without trigonometry.

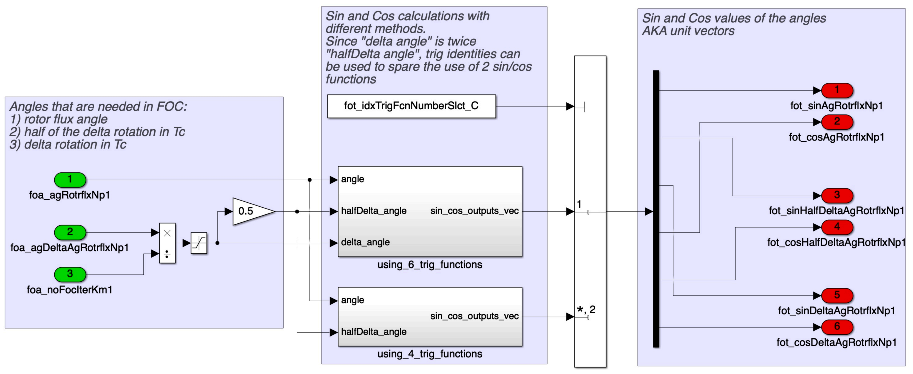

Angles that are needed in FOC:
1) rotor flux angle
2) half of the delta rotation in Tc
3) delta rotation in Tc
foa_agRotrflxNp1
foa_agDeltaAgRotrflxNp1
foa_noFocIterKm1
0.5
Sin and Cos calculations with different methods.
Since "delta angle" is twice "halfDelta angle", trig identities can be used to spare the use of 2 sin/cos functions
fot_idxTrigFcnNumberSlct_C
angle
halfDelta_angle
delta_angle
sin_cos_outputs_vec
using_6_trig_functions
using_4_trig_functions
Sin and Cos values of the angles AKA unit vectors
fot_sinAgRotrflxNp1
fot_cosAgRotrflxNp1
fot_sinHalfDeltaAgRotrflxNp1
fot_cosHalfDeltaAgRotrflxNp1
fot_sinDeltaAgRotrflxNp1
fot_cosDeltaAgRotrflxNp1

Figure 5.51: *FOT function, overview.*

5.29 FOS FIELD ORIENTED SPEED

This block outputs the rotor flux speed by dividing the delta angle calculated in FOA by the execution time of the FOC task. There is also a selectable low pass filter to reduce the noise, as shown in Fig. 5.52

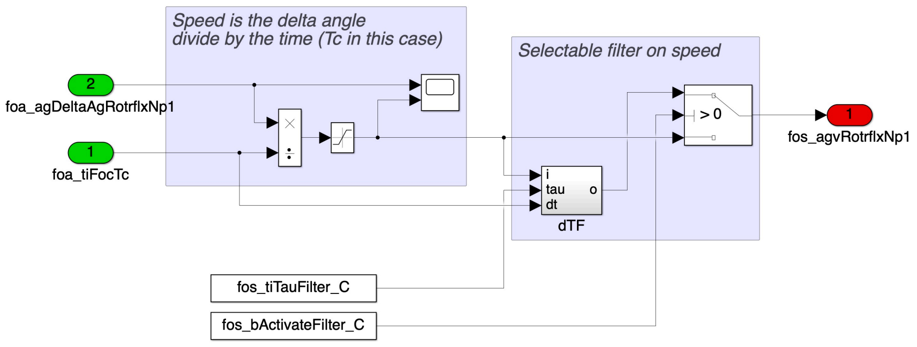


Figure 5.52: *FOS function, overview.*

5.30 FOM FIELD ORIENTED MEASURES

In this function the Park transformation is applied to the fluxes and currents coming from the Electric Machine model using the rotor flux angle, thus moving the machine state in the rotor flux reference frame, ready for the optimal operating point calculation.

The Fig. 5.53 illustrates its inner tasks.

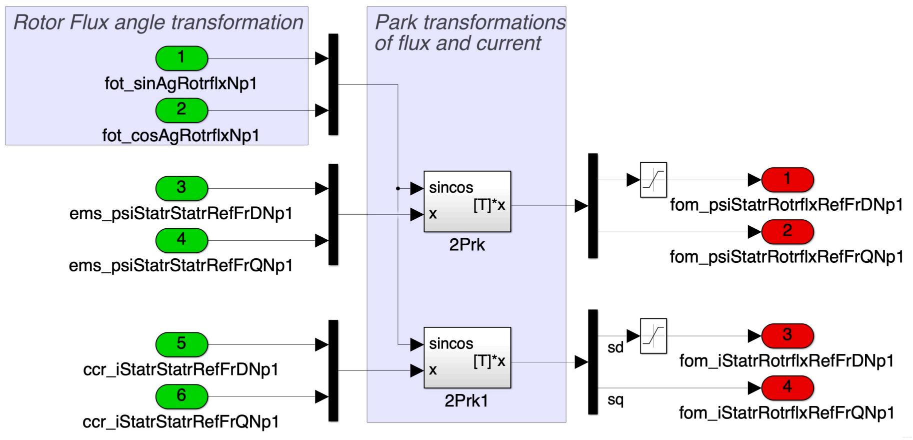
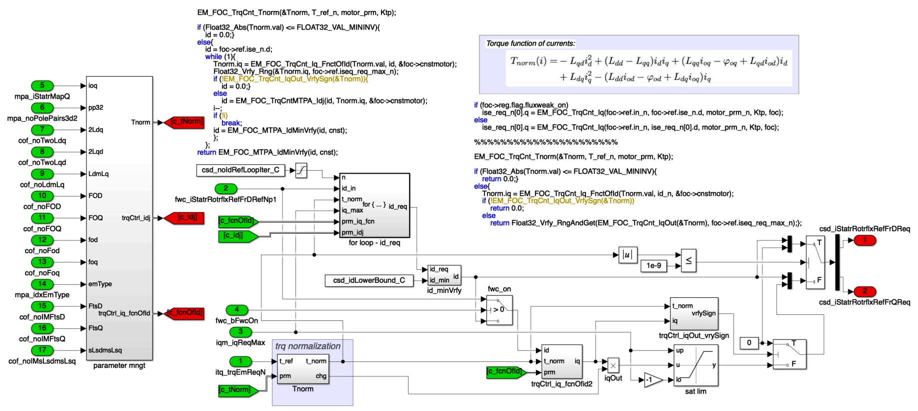

Rotor Flux angle transformation
1
fot_sinAgRotrflxNp1
2
fot_cosAgRotrflxNp1
Park transformations of flux and current
3
ems_psiStatrStatrRefFrDNp1
4
ems_psiStatrStatrRefFrQNp1
5
ccr_iStatrStatrRefFrDNp1
6
ccr_iStatrStatrRefFrQNp1
sincos
x
[T]*x
2Prk
sincos
x
[T]*x
2Prk1
1
fom_psiStatrRotrflxRefFrDNp1
2
fom_psiStatrRotrflxRefFrQNp1
sd
sq
3
fom_iStatrRotrflxRefFrDNp1
4
fom_iStatrRotrflxRefFrQNp1

Figure 5.53: *FOM function, overview.*

5.31 CSD CURRENT SETPOINT DETERMINATION

The system is in charge of producing the d and q current request based on the MTPA operating point. Given a torque request, it is able to output the optimal operating point for the given machine that produces that torque with minimum current, without caring about limitations.

The execution of this is implemented like in Fig. 5.54 and its principal operations are exposed in the section 4.3.1.



```
EM_FOC_TrqCnt_Tnorm(&Tnorm, T_ref_n, motor_prm, Ktp);

if (Float32_Abs(Tnorm.val) <= FLOAT32_VAL_MININV){
  id = 0.0;}
else{
  id = foc->ref.ise_n.d;
  while (1){
    Tnorm.iq = EM_FOC_TrqCnt_Iq_FnctOfId(Tnorm.val, id, &foc->cnstmotor);
    Float32_Vrfy_Rng(&Tnorm.iq, foc->ref.iseq_req_max_n);
    if (!EM_FOC_TrqCnt_IqOut_VrfySgn(&Tnorm)){
      id = 0.0;}
    else
      id = EM_FOC_TrqCntMTPA_Idj(id, Tnorm.iq, &foc->cnstmotor);
    i--;
    if (!i)
      break;
    id = EM_FOC_MTPA_IdMinVrfy(id, cnst);
    };
  };
return EM_FOC_MTPA_IdMinVrfy(id, cnst);
```

Torque function of currents:

$$T_{norm}(i) = -L_{qd}i_d^2 + (L_{dd} - L_{qq})i_d i_q + (L_{qq}i_{oq} - \varphi_{oq} + L_{qd}i_{od})i_d \\ + L_{dq}i_q^2 - (L_{dd}i_{od} - \varphi_{od} + L_{dq}i_{oq})i_q$$

```
if (foc->reg.flag.fluxweak_on)
  ise_req_n[0].q = EM_FOC_TrqCnt_Iq(foc->ref.in_n, foc->ref.ise_n.d, motor_prm_n, Ktp, foc);
else
  ise_req_n[0].q = EM_FOC_TrqCnt_Iq(foc->ref.in_n, ise_req_n[0].d, motor_prm_n, Ktp, foc);

%%%%%%%%%%%%%%%%%%%%%%%%%

EM_FOC_TrqCnt_Tnorm(&Tnorm, T_ref_n, motor_prm, Ktp);

if (Float32_Abs(Tnorm.val) <= FLOAT32_VAL_MININV){
  return 0.0;}
else{
  Tnorm.iq = EM_FOC_TrqCnt_Iq_FnctOfId(Tnorm.val, id_n, &foc->cnstmotor);
  if (!EM_FOC_TrqCnt_IqOut_VrfySgn(&Tnorm))
    return 0.0;
  else
    return Float32_Vrfy_RngAndGet(EM_FOC_TrqCnt_IqOut(&Tnorm), foc->ref.iseq_req_max_n);};
```

Figure 5.54: *CSD function, overview.*

As a further clarification, Fig. 5.55 focuses a little more closely on the action and highlights some of the passages to be discussed. To be pointed out that the leftmost block in Fig 5.54 is omitted from view and it is just a block used to group the various parameters in more compact structures.

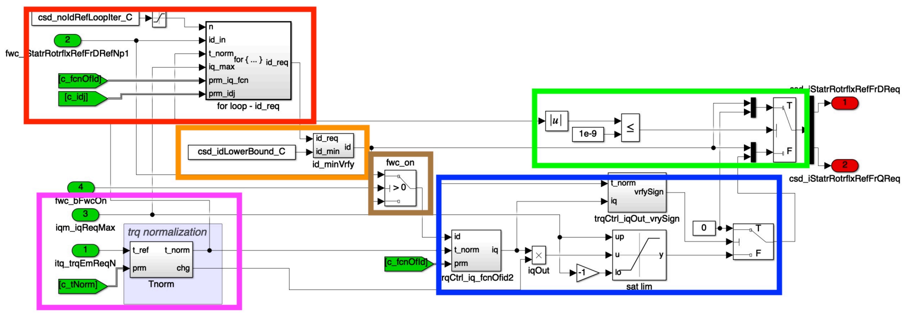


Figure 5.55: *CSD function, overview.*

Given the colour code in the figure, the parts are divided as follows:

GREEN forces the q current request to 0 if the torque request is zero.

MAGENTA applies $T_{norm} = \frac{2T}{3p_p}$ to streamline the equations.

ORANGE imposes a lower bound saturation to the d current request. This is useful especially for IM to keep the machine flux up and improving the responsiveness.

BROWN is activated if the algorithm is in flux weakening and it forces the d current request to the reference coming from the FWC block. In this way the algorithm is mostly deactivated when it is not needed.

BLUE outputs the q request from the torque and d current, the latter one after the possible saturations. To do this it applies the linearized torque equation solved for i_q expressed as (4.43).

ORANGE calculates the i_d request by starting from the torque request and the state of the previous cycle in the form of the i_d reference from FWC. For the reasons explained in section 4.3.1, this procedure is done recursively to improve performance and it consist of approximating a q request and from there adjusting the d one by applying the MTPA definition: intersecting the current circle with the linearized ISOTin one point to obtain the minimum current to produce that value of torque. The key equations are (4.41) and (4.43). Additionally, in this part of the FOC there is the only slight distinguish between IM and SM which amounts to a small redefinition of the IM d inductance to account for the slower dynamic response of the axis.

5.32 FWC FLUX WEAKENING

The FWC function here presented is responsible for the generation of the d current reference in the rotor flux reference frame in accordance with the voltage and current limits. The theoretical considerations that are exploited in the implementation shown in Fig. 5.56 have been presented in the relative chapter 4, in particular the section about the nonlinear flux weakening implementation 4.3.2. This part is instead more focused on the practical description.

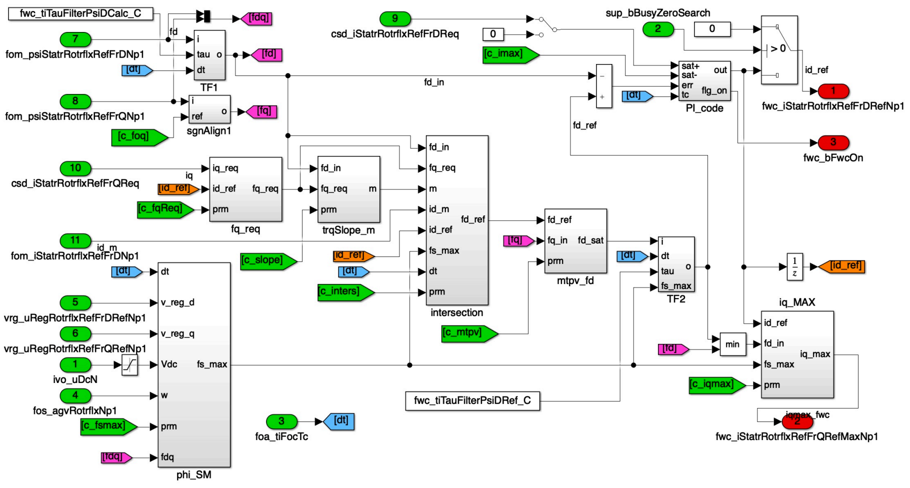

fwc_tiTauFilterPsiDCalc_C
fom_psiStatrRotrflxRefFrDNp1
fom_psiStatrRotrflxRefFrQNp1
csd_iStatrRotrflxRefFrQReq
fom_iStatrRotrflxRefFrDNp1
vrg_uRegRotrflxRefFrDRefNp1
vrg_uRegRotrflxRefFrQRefNp1
ivo_uDcN
fos_agvRotrflxNp1
TF1
sgnAlign1
fq_req
trqSlope_m
intersection
mtpv_fd
TF2
phi_SM
csd_iStatrRotrflxRefFrDReq
sup_bBusyZeroSearch
PI_code
fwc_iStatrRotrflxRefFrDRefNp1
fwc_bFwcOn
iq_MAX
fwc_iStatrRotrflxRefFrQRefMaxNp1
fwc_tiTauFilterPsiDRef_C
foa_tiFocTc
min

Figure 5.56: *FWC function, implementation overview.*

Given the complexity of the system, in Fig. 5.57 is presented a simplified schematic showing only the most important signals, how they relate to each other and how the outputs are generated.

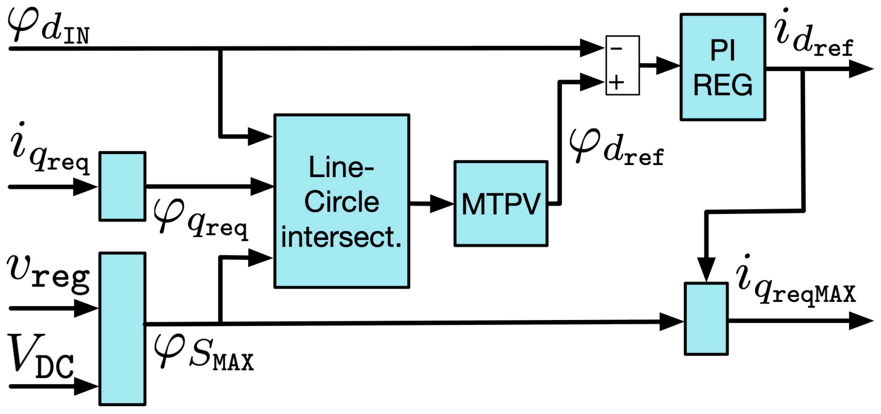

$\varphi d_{\mathtt{IN}}$
$i_{q_{\mathtt{req}}}$
$v_{\mathtt{reg}}$
$V_{\mathtt{DC}}$
$\varphi q_{\mathtt{req}}$
$\varphi S_{\mathtt{MAX}}$
Line-Circle intersect.
MTPV
$\varphi d_{\mathtt{ref}}$
−
+
PI REG
$i_{d_{\mathtt{ref}}}$
$i_{q_{\mathtt{reqMAX}}}$

Figure 5.57: *FWC function, schematic of the most important calculations and signals.*

The first steps to take are the definition of the flux limit and the operating point in the flux reference system, since the feature of this implementation is the use of this reference system to simply the voltage limit as a less complex circle by using a different reference frame.

The maximum flux limit is defined as (4.7), but there are some additional considerations to be applied and Fig. 5.58 is shown to help explain.

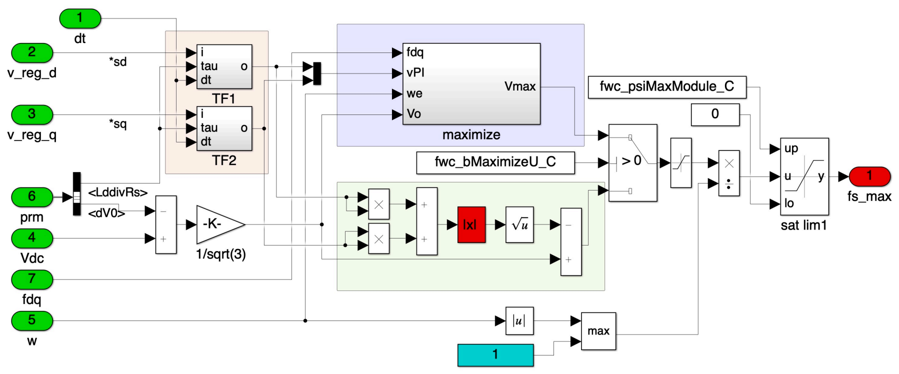


Figure 5.58: *FWC function, calculation of the maximum stator flux* φ_{SM}.

The first note is that the inverter controls the machines by changing the voltage and if the change cannot be executed because the limit is already applied, then it is possible to introduce errors or, worst, loose control of the machine. To avoid this situation, a margin is kept below the limit to allow the control to move the voltage request in all directions. From the maximum voltage V is therefore subtracted a static margin through parameter dV0.

Then, as also explained in the EMF section 5.18, part of the voltage applied by the inverter is used not to contrast the machine flux (electromotive force), but to compensate the windings resistances' voltage drops Δv_r. This is therefore an additional margin that has to be given to the control for a safe and efficient implementation.

The proposed way, shaded in orange in the figure, to know the drops is by using the regulators' outputs, which compensate this mechanism plus the derivative of the flux according to (3.15), and pass them through a low pass filter with the same time constant as the machine's one. The obtained voltages can be considered just the compensation for the machine's resistance, meaning that by subtracting this term from to the maximum voltage, the left over part is the one that influences the fluxes.

The algorithm has the selectable choice to do this subtraction either looking just at the magnitudes of the values, green shade, or using a proper vector subtraction, blue shade.

Of course the vector solution produces more accurate results, especially considering that the directions of the values could work out to be advantageous to the flux limit, thus increasing it. However, the selection has been given because requires a division, a square root and more multiplication with respect to the simple one.

In fact, the direction of the Δv_r must be found by dividing the two sides by the hypotenuse, giving the sine and cosine values of the transformation to be applied to the total voltage limit. The oriented limit is then reduced by the drop vector and then found the magnitude again.

The other starting point of the FWC is the definition of the starting input flux point from which to derive the next reference. This is achieved for the q part by finding out the flux using the flux-current relation (4.47) on the q request current from the CSD function, thus also giving an information on the `MTPA` requested torque. The d flux input $\varphi_{d_{\mathtt{IN}}}$ is the one coming from the machine model.

Having the starting point and the limit, the function then approximates the `ISOT` with the derivative in the given point and applies the line-circle intersection that, when the flux weakening is active, moves the d reference by simultaneously complying with the voltage limit and delivering the requested performance.

This d flux reference is then saturated to the `MTPV` value, found starting from the flux state defined by the reference d flux and the input q flux from the machine model.

A PI regulator is then in charge to reduce the error between reference and input d flux to produce the current reference.

The last part is the calculation of the maximum q current available $i_{q_{\mathtt{fwcMAX}}}$ in the new state by applying (4.64) with just a small check to ensure that it is not negative.

5.32.1 FWC Variant

As explained in section 4.4, the application may require the limitation of the use of the machine parameters to be more robust and efficient. This variant tries to solve this problem with the added complexity of an additional regulator on the q axis. Fig. 5.59 shows the variant as implemented.

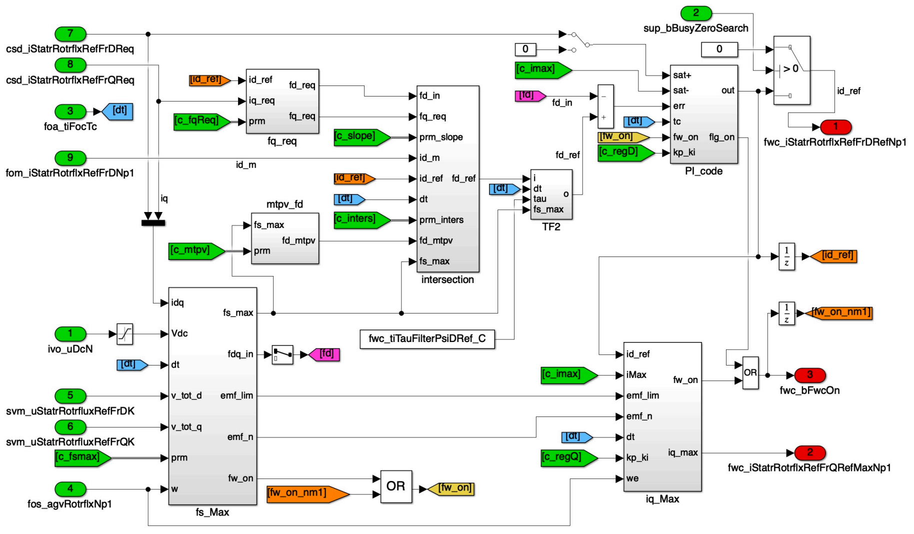
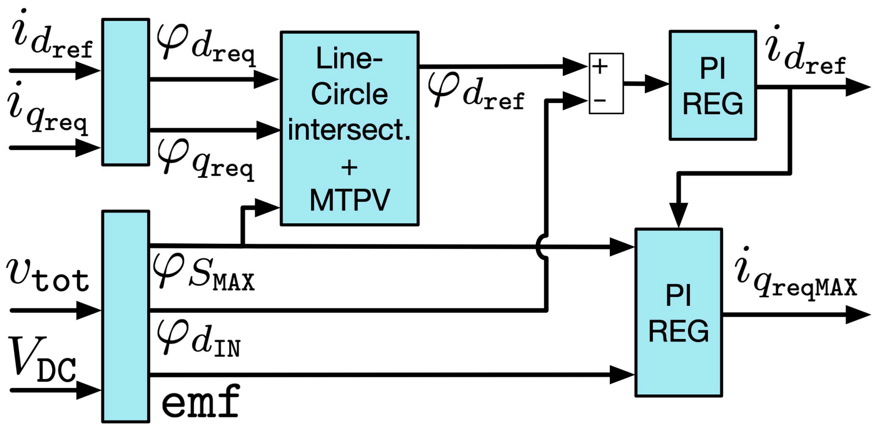


Figure 5.59: *FWC variant function, implementation overview.*

Given the complexity, as for the main solution, Fig. 5.60 is exposed to clarify and compare the algorithm.



$i_{d_{\texttt{ref}}}$
$i_{q_{\texttt{req}}}$
$\varphi_{d_{\texttt{req}}}$
$\varphi_{q_{\texttt{req}}}$
Line-Circle intersect. + MTPV
$\varphi_{d_{\texttt{ref}}}$
PI REG
$i_{d_{\texttt{ref}}}$
$v_{\texttt{tot}}$
$V_{\texttt{DC}}$
$\varphi_{S_{\texttt{MAX}}}$
$\varphi_{d_{\texttt{IN}}}$
emf
PI REG
$i_{q_{\texttt{reqMAX}}}$

Figure 5.60: *FWC variant function, schematic of the most important calculations and signals.*

The intersection between the ISOT and the voltage limit is carried out exactly in the same way, but the d value of starting operating flux point is no longer the $\varphi_{d_{\texttt{IN}}}$ from the machine model flux, but it is derived from the old d current reference.

The second major difference is in the *fs_max* function which sees the addition of the calculation of the back electromotive force values to approximate the flux state of the machine without the use of parameters.

The function then takes the total voltages outputs going to the inverter to calculate the back emf vector state as (4.69), its magnitude and its maximum limit with the approximation (4.73). The emf values divided by the speed are then converted into fluxes. The d flux derived in this manner is then used as the input value to be compared to the reference in the d axis PI regulator.

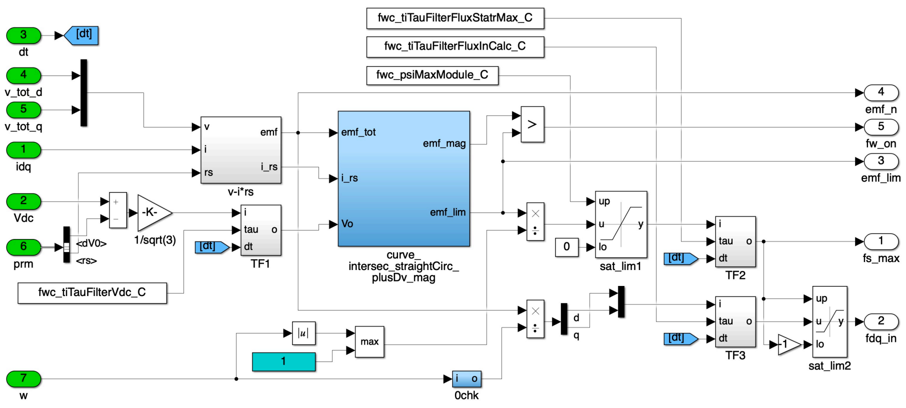


Figure 5.61: *FWC variant function, calculation of the maximum stator flux φ_{SM} and electromotive force.*

The last major subsystem is the one calculating the maximum q current which, since it tries to avoid the use of the inductance values, is implemented with the use of a PI regulator. The fluxes to compare are both derived from the emf and the value saturated to the current limit to maintain a realistic output complaining with the limits. At both the regulator inputs are applied low pass filter to make the algorithm more robust and controllable.

5.33 IQM IQ MAX

This function finds the maximum available q axis current based on the i_d reference produced by the flux weakening system. It accounts for the current limit and the voltage limit.

The current limit is ensured by the upper part of the implementation shown in Fig. 5.62 where the i_d reference is subtracted to the maximum magnitude of the current circle and produces the maximum request.

The result is then used to limit the value of the maximum i_q reference obtained in the flux weakening from the application of the voltage limit. The end result is therefore the maximum available q current that lays inside the operational zone of the machine.

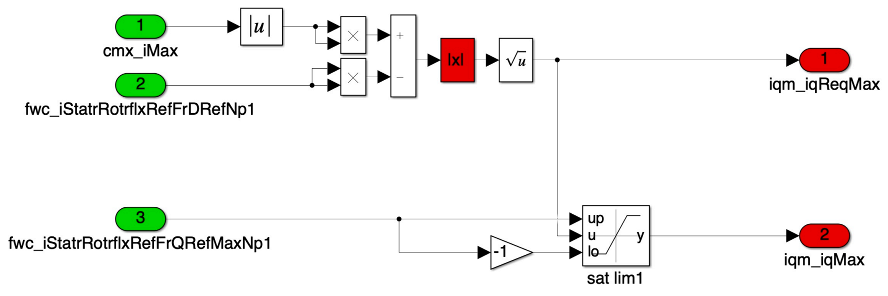


Figure 5.62: *IQM function, overview.*

5.34 TMX TORQUE MAXMIN

This function would be used to estimate the maximum torque available in the given machine state.

For the linear machine this would be an easy operation given the knowledge of the machine fluxes and the d current reference from the flux weakening. The general linear torque equation (3.16) would be used in reverse with the i_q given by the IQM function, thus being the maximum available.

This is however no longer true for the nonlinear machine, because the machine state has been derived through a linearization of the machine parameters around the instantaneous operating point. This means that all the equations are valid (with very reasonable approximation) only close to that point. It follows that it is not possible to estimate with confidence the behaviour of the machine in an arbitrarily distant point without having to, at minimum, re-calculate all the machine parameters or, worst, applying some complex recursive procedures.

5.35 IQR IQ REFERENCE

In this function, presented in Fig. 5.63, the q axis reference current is generated by taking the q request from the MTPA calculation, which satisfy the torque request, and applying to it a saturation to the maximum available q value from the dedicated IQM function.

Both signals are used with low pass filter to ensure smooth torque delivery.

Additionally there is also an actuation of the resolver zero procedure that ensures a q current reference of 0 during its execution.

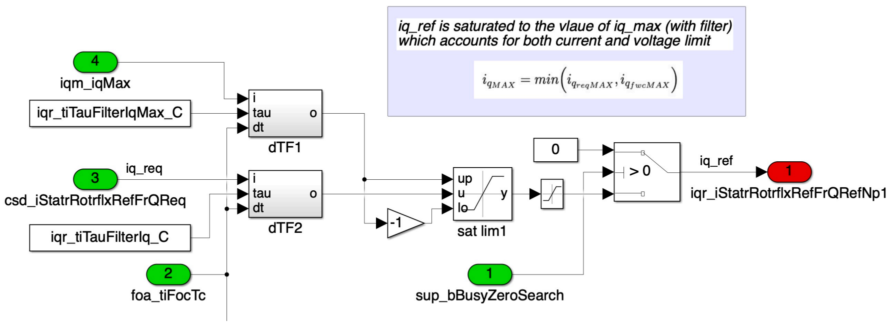

iq_ref is saturated to the vlaue of iq_max (with filter) which accounts for both current and voltage limit
$i_{qMAX} = min\left(i_{q_{reqMAX}}, i_{q_{fweMAX}}\right)$
4
iqm_iqMax
iqr_tiTauFilterIqMax_C
i
tau
dt
o
dTF1
3
iq_req
csd_iStatrRotrflxRefFrQReq
i
tau
dt
o
dTF2
iqr_tiTauFilterIq_C
2
foa_tiFocTc
-1
up
u
lo
y
sat lim1
0
> 0
iq_ref
1
iqr_iStatrRotrflxRefFrQRefNp1
1
sup_bBusyZeroSearch

Figure 5.63: *IQR function, overview.*

6 RESULTS

This part is dedicated to the presentation of the results obtained during the development of the control software under study.

Other than the validation results from the final nonlinear algorithm, the milestones accomplished along the way are also presented that have been kept throughout the development with minor modification because they are still valid in essence for the new approach or they can be considered for simpler implementations in other applications.

The tests shown have all been carried out in the Simulink MiL environment exposed in the previous chapters. This testing environment has been as proven an effective way to verify the algorithm performance under many aspects by the experience of the LEMAD. However, experimental validation is always needed at least as a final step to work out specific issues, especially regarding signals noise and hardware non-ideality.

For the purposes of this work, the experimental testbench validation was carried out by the collaborating company on a confidential project. The data is therefore not presented in this thesis, but they were first of all crucial to push an efficient nonlinear implementation and, second, to validate the algorithm outputs, verifying also the MiL environment.

6.1 DISCRETE MODEL OPTIMIZATION

This section focuses on the validation of the Unified Electric Machine Model discrete-time solver through the interval sub-division and general optimizations at the core of the EMS function, presented in section 3.2.1.

The derivation of that final discrete-time algorithm (3.45) made use of various intermediate steps that are just equivalent formulations of the same solving recursive function for the integration of the continuous-time model equation (3.13).

As explained, Simulink is able to implement continuous-time solvers as well as similar solutions for discrete-time, making an hypothetical direct implementation of the starting model equations a possibility. The optimization carried out, however, was demonstrated to have advantages both in execution speed and accuracy compared to the standard Simulink solution.

For a complete analysis, it has been decided to compare various steps along the optimization against the continuous and discrete Simulink implementations. All formulations have been run with the same inputs.

In particular, the tested models are hereby presented.

CNT: continuous model

This function implements directly (3.13) and the scheme in Fig. 6.1. This model is the best optimized solution for the modelling environment since Simulink is developed specifically for this type of solver. It does not run real-time, thus making it extremely accurate. This implementation has therefore been chosen as the base reference for precision for the other implementations. Its execution speed has been selected to normalize the results of the other methods.

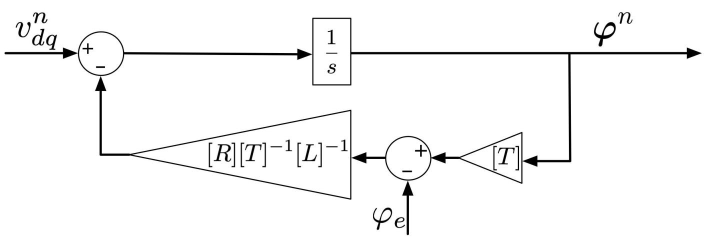


Figure 6.1: *Continuous time Simulink model: CNT.*

SIMULINK-DSCR: discrete model with Simulink integrator block

This function is exactly the same as the CNT continuous model, but it uses the Simulink built-in *discrete integrator* block with forward Euler method that replicates in discrete form the Laplace notation, Fig. 6.2.

NAT-DSCR: discrete model with sub-interval integration in natural reference frame

The function NAT-DSCR shown in Fig. 6.3 implements (3.39) using a recursive integration method and a *for loop* to do the sum of the sub-intervals. It therefore uses all the values in their natural reference frames and multiplies the old rotational matrix with the sub-interval step rotation.

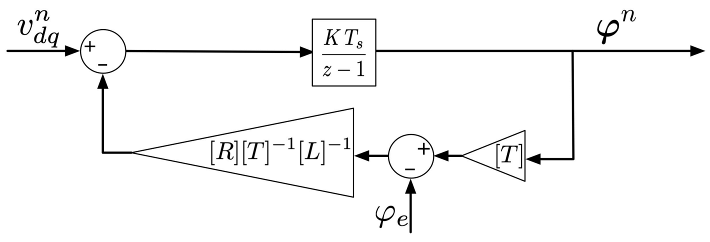


Figure 6.2: *Discrete time model using built in integrator: SIMULINK-DSCR.*

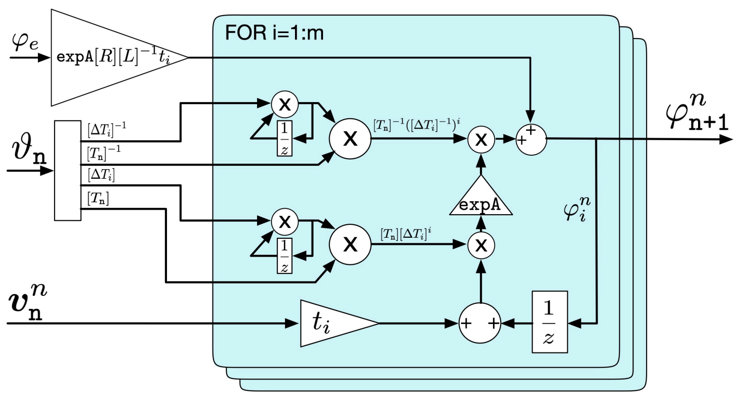


Figure 6.3: *Discrete time model using the algorithm under study without optimization: NAT-DSCR.*

INV-DSCR: discrete model with fast matrix inversion

The function INV-DSCR shown in Fig. 6.4 implements (3.39) using an approach very similar to NAT-DSCR. The difference is in the calculation of any inverse rotation matrix $[T_n]^{-1}$ or $[\Delta T_i]^{-1}$. Taking advantage of the fact that the inversion of $[T_n]$ is the same matrix with just two sign changes, there is a significant simplification by not using a separate calculation of the inverse matrices at every iteration, but just multiplicating for -1 the two elements in position 12 and 21 of (2.3).

Another small improvement is that the calculation of $\frac{t_c}{m}v^n$ is moved outside the *for loop*.

ROT-DSCR: discrete model with sub-interval integration in rotor reference frame

The function ROT-DSCR, shown in Fig. 6.5, implements (3.45). It represents the sub-integrations done in rotor reference frame and then converts back to the natural frame just the results. This is done using the final rotational matrix $[T_{n+1}]^{-1}$ directly calculated

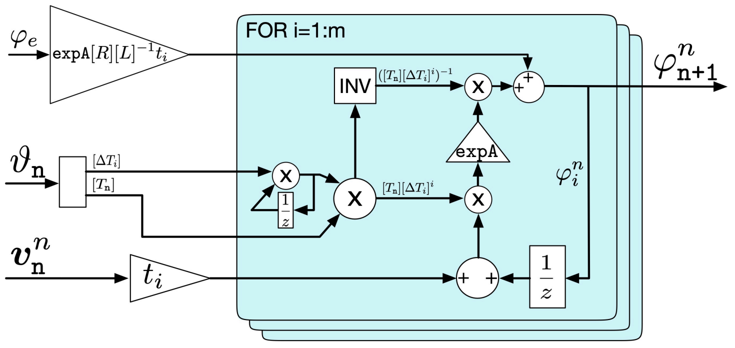


Figure 6.4: *Discrete time model with fast matrix inversion: INV-DSCR.*

as (3.43) for the rotation ϑ_{n+1} outside of the iterative loop. With this approach the *for loop* is very much simplified with just a sum and a matrix multiplication between three elements.

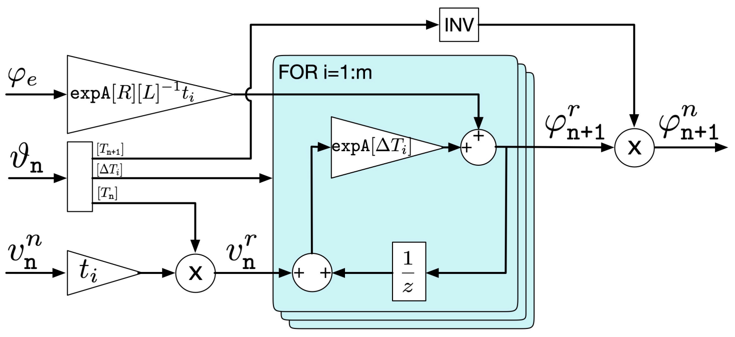


Figure 6.5: *Discrete time model with rotor reference frame integration: ROT-DSCR.*

FAST-DSCR: discrete model with fast rotation transformation

The function FAST-DSCR, shown in Fig. 6.6, implements (3.45) and improves on the previous one by employing a custom matrix multiplication with fewer operations. So far all the matrix multiplications were conducted with the general matrix multiplication rules that use all the elements of the matrices involved. However, matrices $[T_n]$ and $[\Delta T_i]$ can be considered sparse and their use can be simplified considering only the minimum elements involved. In particular, only the upper left corner of (2.3) (regarding the stator values) actually operates a rotation with sine and cosine, while the rest of the elements are either 0 or 1, not requiring any mathematical operation.The rotation matrix multiplication is therefore replaced by a much faster custom function that involves only the minimum number of operations indicated in Fig. 6.6 with dotted squares.

The blocks that in the previous models generated the full rotational matrices are here substituted by similar ones that just output the sine and cosine values.

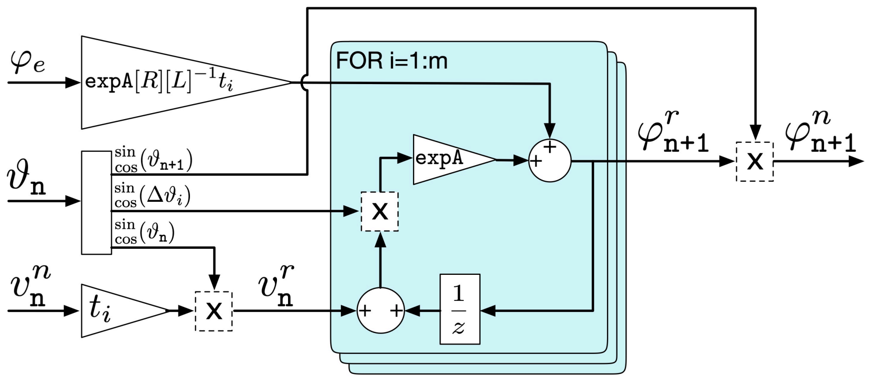

φ_e
$\mathtt{expA}[R][L]^{-1}t_i$
FOR i=1:m
ϑ_n
$^{\sin}_{\cos}(\vartheta_{n+1})$
$^{\sin}_{\cos}(\Delta\vartheta_i)$
$^{\sin}_{\cos}(\vartheta_n)$
expA
φ^r_{n+1}
φ^n_{n+1}
X
v^n_n
t_i
v^r_n
$\frac{1}{z}$

Figure 6.6: *Discrete time model with fast rotation implementation: FAST-DSCR.*

6.1.1 Testing Environment

To compare performance and accuracy, the different algorithms described above are implemented in the same manner. The comparison is between the different types of integration models that, starting from the voltage and rotation applied to the machine, outputs the fluxes. For this reason the input side of the model compared is the same, with the generation of an identical voltage source and speed reference.

All of the models have the same parameters, both for the machine simulated and for the solver and inputs. To highlight the possible final application, the machine chosen is an high performance induction machine for electric vehicles, up to 250 kW of power and 15000 rpm maximum nominal speed, with its main data summarized in Table 6.1. An IM has been chosen, mainly, because the formulation implies the use of the full 4x4 matrix implementation, without simplifications that could be used in SM to improve calculation performances, making it a more complex and severe test-bench.

To asses how good the discrete approximations presented are to produce the correct machine fluxes, the first comparison is done feeding the model with a symmetric three-phase nominal voltage of 360 V, constant frequency of $6\frac{\mathrm{rad}}{\mathrm{s}}$ and the rotor rotating at the same constant speed. The relative low speed and equal value between stator and rotor frequency was chosen to immediately qualitatively give a sense of how good is the implementation since, in a IM, if the speeds are equal, the rotor fluxes are constant. The other testing condition is carried out at an electric speed of $6000\frac{\mathrm{rad}}{\mathrm{s}}$ to prove both the improvements using increasing sub-intervals integration steps and the algorithm capabilities at high frequency.

Given that the machine equations are known and the continuous implementation is the best approximation possible for a model, its output has been chosen as the true one against which the other method are compared. As a remainder, this method cannot be practically implemented in real time DSP, because of the DSP architecture and asynchronous calculation requirements of the method. Moreover, Simulink is optimized to run this type of models, thus producing better performances than any possible discrete-time implementation.

The machine parameters and all the constants needed are calculated by a separate subsystem that takes the user inputs (resistance, inductance, number of iterations, pole pairs and magnet flux) and constructs the required matrices. In particular, it calculates matrix `expA` using (3.30). This division of operation wants to simulate the fact that in the real program some constants can be calculated in different parts of the algorithm to be used by more that one subsystem and that can also run much slower that the main control functions.

Table 6.1 summarises both the simulation parameters and the machine ones. The entire model is run with a variable-step solver and the discrete algorithms are inserted into a triggered subsystem that produces the requested discrete sample timing.

All the tests are performed by supplying the motor with a standard sinusoidal three-phase voltage with known (variable) frequency. Clarke transformation is operated and the rotor speed is set as a constant.

6.1.2 Model execution speed

Execution speed testing are carried out using the parameters in Table 6.1 and varying the number of sub-interval integrations. Of course the CNT and SIMULINK-DSCR models do not have this variable and their execution time remains constant, apart for a slight improvement between the first and second run, the reason of which is to be found in the Simulink solver engine.

Table 6.1: *Simulink general simulation and machine parameters for every implementation used.*

Description	Value	Unit
Solver	ode45	-
Max-Min Step Size	*auto*	-
Simulation Time	5	s
Discrete Sample Time	0.125	ms
Discrete Sample Frequency	8000	Hz
Number of Sub-intervals	*variable*	-
Nominal Voltage	360	V
Nominal Current	230	A
Max Speed	15000	rpm
Pole Pairs	4	-
Stator Resistance	3.4	mΩ
Rotor Resistance	1.3	mΩ
Stator Inductance	0.16	mH
Rotor Inductance	0.16	mH
Mutual Inductance	0.143	mH
Magnet Flux	*none*	-

Results are given in Fig. 6.7 normalized with respect to the CNT execution time and the times are found taking the average total running time over 10 simulations. For single sub-interval integration every proposed algorithm is faster than the normal discretization of the SIMULINK-DSCR. Of course, as it will be shown later, the accuracy is not as good. For all the proposed algorithms, then, more than a couple of subdivisions are required in practice to obtain a reasonable accuracy and the calculation manipulation can increase significantly the performances.

The NAT-DSCR implementation of Fig. 6.3 uses full transformation matrices, standard inversions and two matrix power elevations all inside the for loop, thus the calculation time increases steeply at the increase of steps. Already at 2 subdivisions the execution time is more than the discrete model SIMULINK-DSCR. The INV-DSCR implementation of Fig. 6.4 that uses a fast inversion technique permits to calculate only one power elevation, therefore the operations inside the loop are reduced significantly and this model is already much faster. Moreover, since the optimization is focused inside the loop, the execution time is much less dependent on the number of sub-intervals.

By following the description of the solvers, the tendency is to move calculations outside of the for loop, thus it can be seen how all the models are quite close at lower steps, while the difference gets bigger increasing the number of sub-intervals. This is what happens in ROT-DSCR of Fig. 6.5 where the *for loop* calculations are substantially simplified.

The best solution described, FAST-DSCR of Fig. 6.6, is still more time efficient than the SIMULINK-DSCR integration process with 15 integration intervals and it is done by having only one matrix multiplication, two sums, one delay and 3 custom simplified matrix transformations.

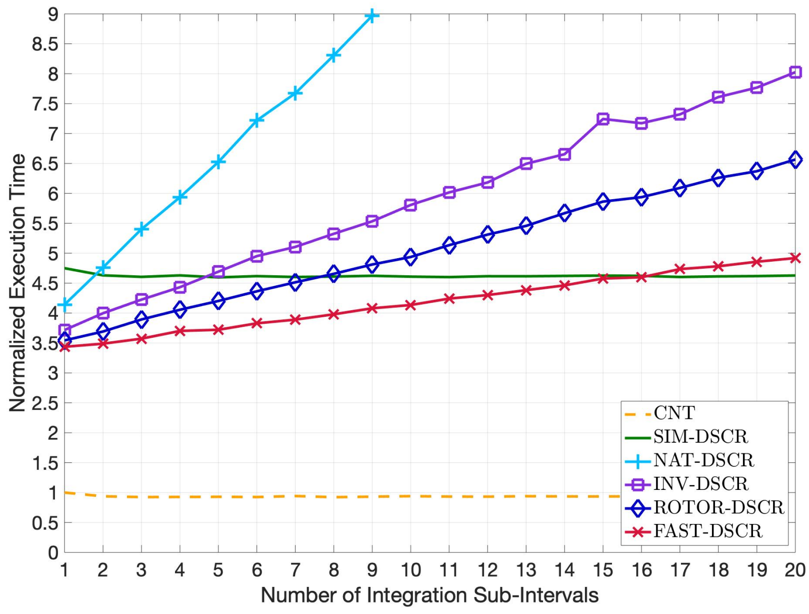
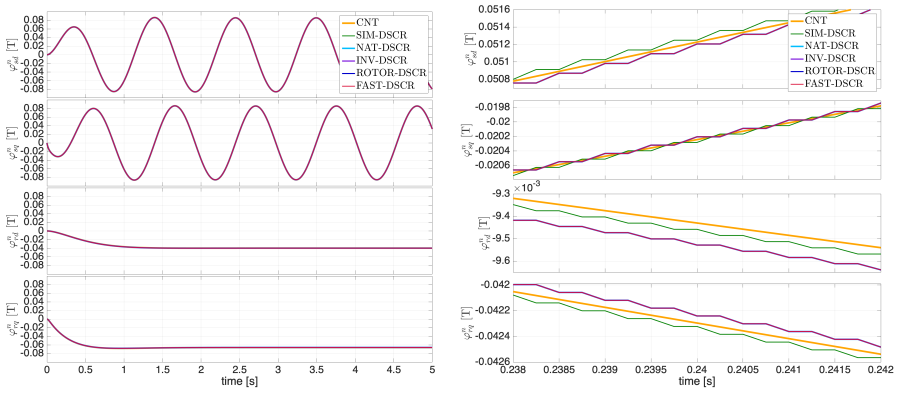


Figure 6.7: *Execution time comparison increasing the discrete sub-interval integration number. CNT and SIMULINK-DSCR do not depend on the number of sub-intervals. Time is normalized to the CNT execution time.*

6.1.3 Model precision

This section shows how the different models proposed produce, in fact, very good approximations of the machine flux both at slow and high speeds. The CNT model (Continuous) is used again as the reference output and the others are compared to it. Fig. 6.8 presents, as an example, the outputs of all the models for the machine with rotor turning at the same angular velocity of $6\frac{rad}{s}$ as the voltage input: the continuous time (CNT), the discrete time with Simulink integrator (SIMULINK-DSCR), the discrete time in natural reference frame (NAT-DSCR), with fast inverse transformation (INV-DSCR), in rotor reference frame (ROT-DSCR) and with fast transformation (FAST-DSCR). The last 4 are all conducted with just 1 sub-interval integration. Overall, it is easy to see how all the models correctly solve the flux equation and give macroscopically equal results.

Fig. 6.8b shows a magnified view of the difference between the CNT solution and the proposed methods. These errors may impact the whole control system since they are predictions used by almost all other functions of the motor control algorithm afterwards and their relative error increases with the flux frequency, thus making it much more problematic at high speeds. As stated before, in the real control system a current error feedback is necessary to compensate for the small integration errors, but it is not described in this paper.



(a) (b)

Figure 6.8: *Simulink flux machine model comparison example between the model presented: CNT, SIMULINK-DSCR, NAT-DSCR, INV-DSCR, ROT-DSCR and FAST-DSCR. 6.8a overview, 6.8b zoom to highlight differences.*

Detailed description and error evaluation is going to be analyzed only for the FAST-DSCR algorithm, given that it has been verified in section 6.1.2 that it is the simplest and fastest solver among those proposed. Moreover, since comparing a continuous signal with a discrete one would produce a discontinuous error value, the difference is shown with a very short mean filter (on 2 data points) to better visualize the results.

Since the proposed solution is also a prediction of the flux at the next iteration step φ_{n+1}, the error is calculated between the current algorithm iteration value and the value at the next step of the CNT model, considered as the true approximation. This is carried out also for the SIMULINK-DSCR which is not predictive, thus its error is intrinsically larger. However, this decision reflects the ultimate use of the algorithms proposed, which is the best flux estimator for high performance motor control that must implement predictive calculations.

Fig. 6.9 shows the difference between the CNT and FAST-DSCR algorithms as percentage of the maximum value of flux. The error of the SIMULINK-DSCR algorithm is added as a reference. The four plots are for the four fluxes estimated by the model which are the stator and rotor flux in the d and q axes. In this example the rotational speed of the stator field and the electrical angular speed of the rotor are the same and equal to $6\frac{\mathrm{rad}}{\mathrm{s}}$, therefore slow compared to common values.

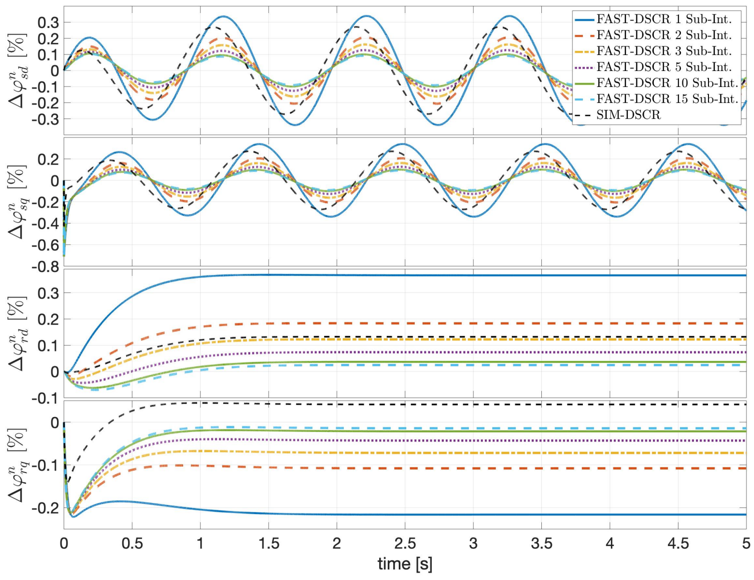


Figure 6.9: *Percentage difference between CNT and FAST-DSCR at low frequency increasing the integration sub-intervals for flux estimation in (from top to bottom) stator d-axis, stator q-axis, rotor d-axis, rotor q-axis. Stator field and rotor mechanical rotational speed equal to $6\frac{\mathrm{rad}}{\mathrm{s}}$.*

The proposed solver is expected to have better accuracy than SIMULINK-DSCR the faster the motor rotation. Therefore Fig. 6.10 shows the error percentage when the stator field rotates at $6200\frac{\mathrm{rad}}{\mathrm{s}}$ and the rotor has a mechanical angular velocity of $5700\frac{\mathrm{rad}}{\mathrm{s}}$, modelling a condition of 8.1% slip close to the nominal maximum mechanical speed of 15000 rpm. At this high speed, a clear improvement of the model precision can be seen over the standard integration of SIMULINK-DSCR.

To quantify the exact level of improvement, Table 6.2 shows the mean error squared of the percentage error for the different fluxes and integration methods at both low and high speed. Although

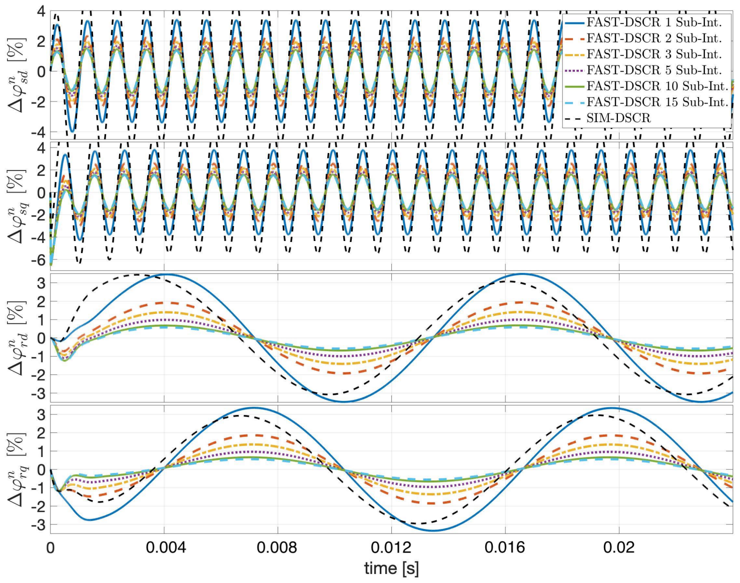


Figure 6.10: *Percentage difference between CNT and FAST-DSCR at high frequency increasing the integration sub-intervals for flux estimation in (from top to bottom) stator* d*-axis, stator* q*-axis, rotor* d*-axis, rotor* q*-axis. Stator field and rotor mechanical rotational speed have different values,* 6200 *and* 5700 $\frac{\mathrm{rad}}{\mathrm{s}}$ *respectively.*

the error is always calculated with respect to the CNT solver, in the table the proposed FAST-DSCR method with just 1 integration sub-interval is taken as reference to calculate the percentage improvement of the other instances. The reason behind this choice is to highlight how many sub-intervals are needed for similar accuracy and how much better accuracy can be achieved with the FAST-DSCR compared to the SIMULINK-DSCR.

As it can be seen in Fig. 6.10 and Table 6.2, just 1 sub-interval is not enough to justify the speed improvement of FAST-DSCR. Although it performs a little bit better at high speed, the results for the stator flux might be influenced more by the predictive nature rather than true precision. However, increasing the number of integration sub-intervals, a consistent increase in precision is confirmed both in respect to the single interval and to the standard integrator SIMULINK-DSCR.

Precision improvements are also relatively less significant above 5 sub-intervals, improving around 20% from 2 to 5 intervals and only around 10% or less from 5 to 15. In cases where the improvements are comparable with the standard integration SIMULINK-DSCR, namely at low speeds, there is still the advantage of the proposed solver in terms of execution speed. At high speed,

Table 6.2: *Mean error squared results of low and high frequency simulations. Base error is the one of the 1 sub-interval model.*

	Low Frequency Example $6\frac{rad}{s}$		**High Frequency Example** $6000\frac{rad}{s}$	
Algorithm	Mean Err. Sq.	Variation	Mean Err. Sq.	Variation
		Stator d-axis - φ^n_{sd}		
SIM-DSCR	34.2e-7	−37.1%	120.0e-5	+113.6%
FAST 1 int.	54.3e-7	0%	57.2e-5	0%
FAST 2 int.	20.2e-7	−62.8%	26.3e-5	−53.9%
FAST 3 int.	12.5e-7	−76.9%	18.7e-5	−67.3%
FAST 5 int.	7.8e-7	−85.7%	13.5e-5	−76.4%
FAST 10 int.	4.9e-7	−90.8%	10.2e-5	−82.2%
FAST 15 int.	4.2e-7	−92.3%	9.2e-5	−84.0%
		Stator q-axis - φ^n_{sq}		
SIM-DSCR	33.9e-7	−37.0%	150.0e-5	+113.3%
FAST 1 int.	53.9e-7	0%	72.3e-5	0%
FAST 2 int.	20.8e-7	−61.4%	33.6e-5	−53.5%
FAST 3 int.	13.4e-7	−75.1%	24.0e-5	−66.8%
FAST 5 int.	8.8e-7	−83.7%	17.5e-5	−75.8%
FAST 10 int.	6.1e-7	−88.7%	13.4e-5	−81.5%
FAST 15 int.	5.3e-7	−90.1%	12.1e-5	−83.3%
		Rotor d-axis - φ^n_{rd}		
SIM-DSCR	14.9e-7	−87.7%	48.5e-5	−19.9%
FAST 1 int.	121.8e-7	0%	60.6e-5	0%
FAST 2 int.	29.2e-7	−76.0%	18.9e-5	−69.3%
FAST 3 int.	12.6e-7	−89.7%	10.0e-5	−83.6%
FAST 5 int.	4.4e-7	−96.4%	5.0e-5	−91.8%
FAST 10 int.	1.3e-7	−98.9%	2.4e-5	−96.0%
FAST 15 int.	0.8e-7	−99.3%	1.8e-5	−97.1%
		Rotor q-axis - φ^n_{rq}		
SIM-DSCR	1.9e-7	−95.7%	42.3e-5	−22.8%
FAST 1 int.	44.6e-7	0%	54.8e-5	0%
FAST 2 int.	12.7e-7	−71.4%	16.8e-5	−69.3%
FAST 3 int.	6.7e-7	−84.9%	9.0e-5	−83.6%
FAST 5 int.	3.6e-7	−92.0%	4.5e-5	−91.8%
FAST 10 int.	2.1e-7	−95.1%	2.1e-5	−96.1%
FAST 15 int.	1.9e-7	−95.7%	1.6e-5	−97.2%

however, the FAST-DSCR solver performs much better both in precision and in calculation speed.

Considering both factors, it is verified that the use of the FAST-DSCR solver with a minimum of 5 sub-intervals and a maximum of 10 yields the best compromise between execution speed and precision for most applications.

6.2 LINEAR MACHINE OPERATION

Here are reported the test outputs for some linear machines to prove the validity of the control algorithm.

All the runs were conducted with the same inputs types to showcase the capability of following the optimal operating points and respecting the currents and voltage limits. As shown in Fig. 6.11, the torque request is first demanded with a fairly steep rate at no rotor speed to make the machine run on the MTPA.

After the torque request as reached its constant value, the speed is then increased at a constant rate. The control will therefore follow the flux weakening strategy while also taking care of the limitations.

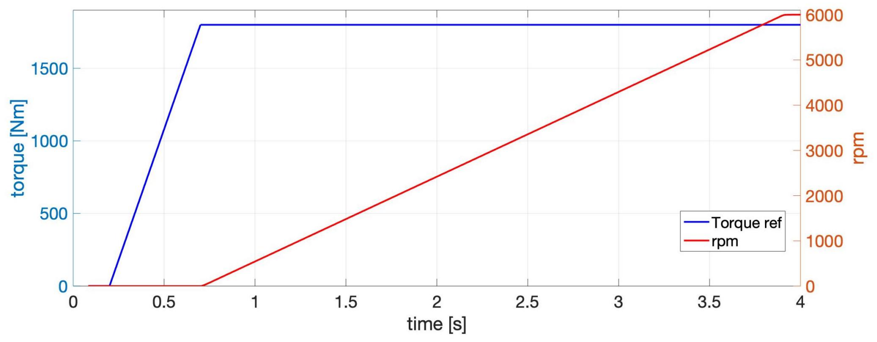


Figure 6.11: *Torque and speed input ramps as used for the tests in this section.*

The machines presented were chosen without any particular application in mind. Thanks to the Unified model approach the few parameters shown in Tab. 6.3 are all that is needed to change motors. The final speed value for all tests is 8000 rpm and the switching frequency 8000 Hz.

First, EM1 is an Internal Permanet Magnet Synchronous Machine and the control correctly follows the optimal path during the operation, Fig. 6.12. The ISOT, MTPA and MTPV curves on the plots are all derived from the analytical expressions. The magenta lines are the plotted values of the control outputs. The resulting currents are shown in the d-q current reference frame, Fig. 6.12a, and the fluxes in the corresponding flux one, Fig. 6.12b.

Machine EM2 is a Surface Permanent Magnet Synchronous Machine and in the same way the optimal path is respected in its entirety, Fig. 6.13, showing both current and flux reference planes.

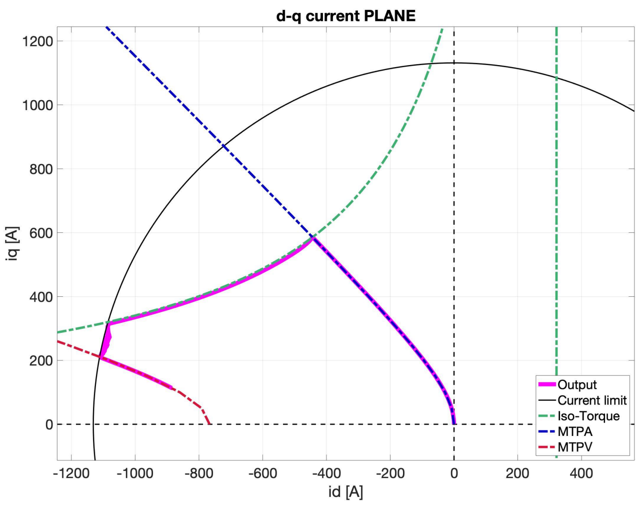
Good results are also found in Fig. 6.14 for the EM3 which is a Reluctance Synchronous Machine, meaning that there are no permanent magnets to provide an excitation flux.

Table 6.3: *List of the parameters used for the linear Electric Machines (EM) test proposed.*

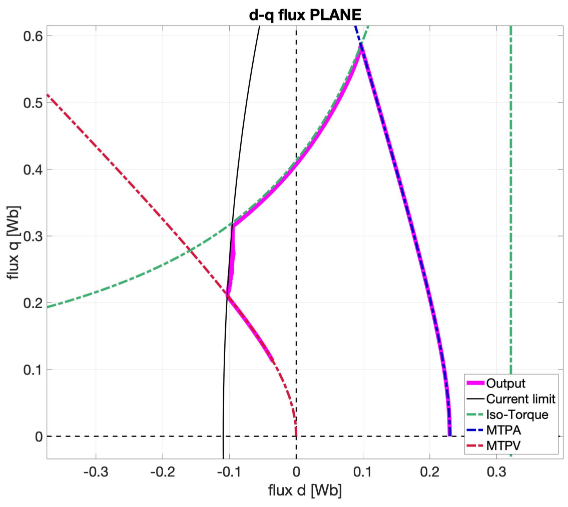
Param.	Units	EM1	EM2	EM3	EM4	EM5
		IPM-SM	SPM-SM	R-SM	WR-SM	IM
L_d	mH	0.3	0.32	0.17	0.4	2
L_q	mH	1.0	0.32	0.24	0.2	0.11
φ_e	Wb	0.23	0.2	0	0.03	0
I_{MAX}	A	800	660	600	600	200
Trq_{refMAX}	Nm	1900	1000	200	500	340
V_{DC}	V	700	700	700	700	220
p_p	–	4	4	4	4	4
r_s	mΩ	3.9	3.9	3.9	3.9	10

Fig. 6.15 proposed the results obtained for a Wound Rotor Synchronous Machine, EM4, where a small constant excitation current is simulated to provide an utilization example. During MTPV operations there is a noticeable offset in the control results, probably given by a non-perfect tuning of the FWC regulator, which has been kept the same in all tests for the sake of limiting the differences and showing the robustness of the Unified approach.

Finally, Fig. 6.16 proposes the results for EM5, the Induction Machine. Here there might seem to be some problems since the flux outside the MTPA looks to have a substantial offset and the current presents some delays. However, it has to be taken into account that the IM has much higher time constants and the regulator constants have not been changed. Delays in the torque response are to be expected with such high dynamic load request.

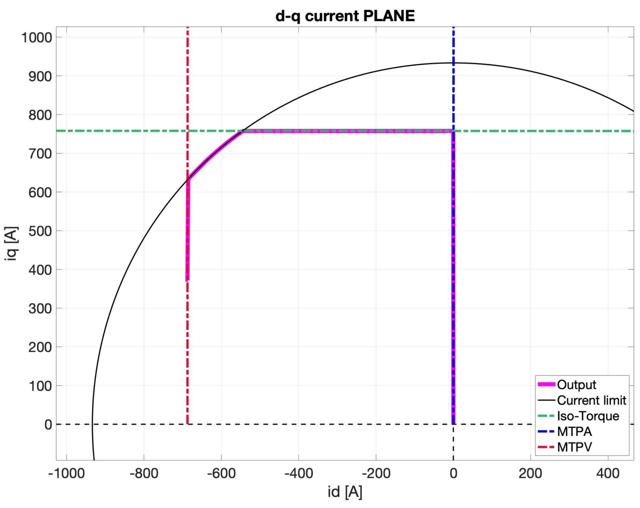


(a)

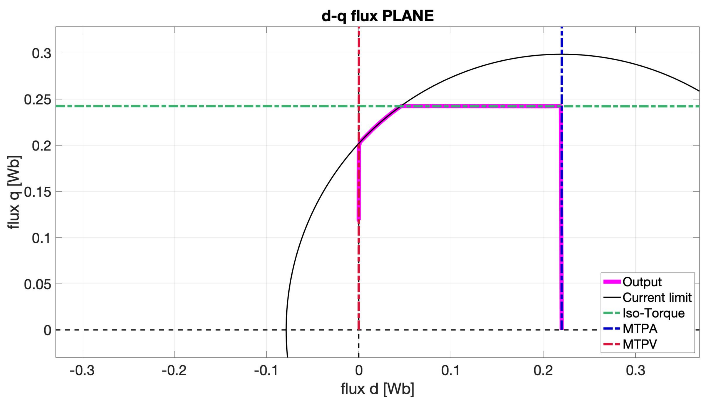


(b)

Figure 6.12: *Operating point performance test results for* EM1 *(IPM-SM), shown in the current (a) and flux (b) reference frames.*

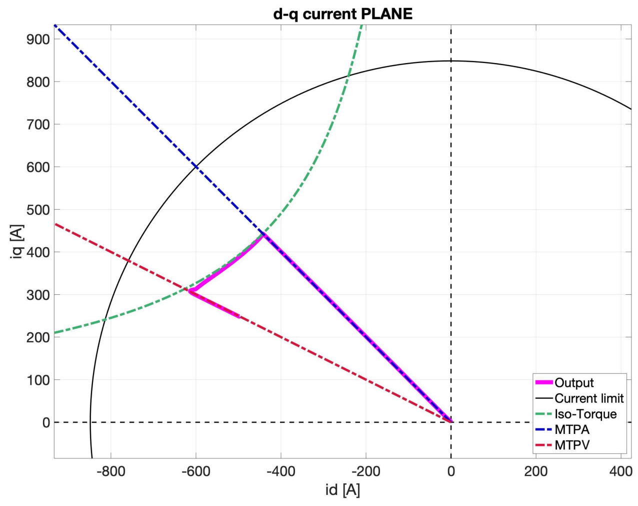


(a)

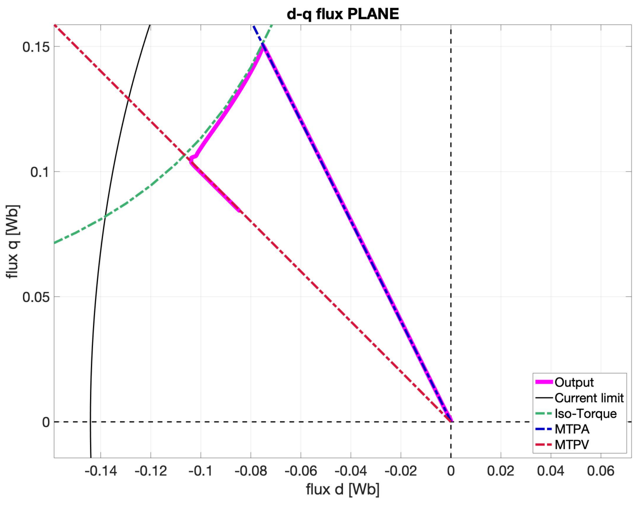


(b)

Figure 6.13: *Operating point performance test results for* EM2 *(SPM-SM), shown in the current (a) and flux (b) reference frames.*

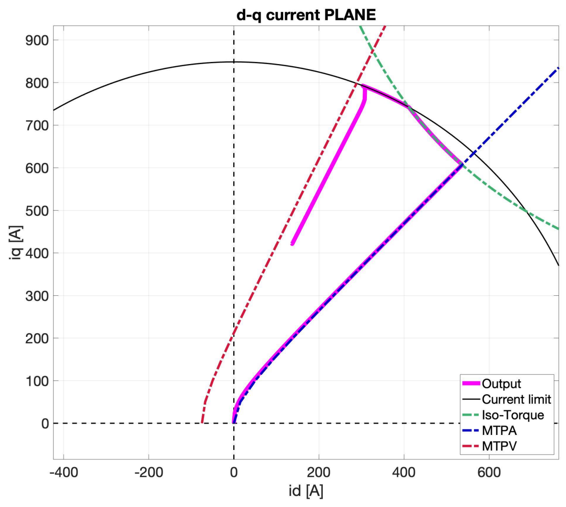


(a)

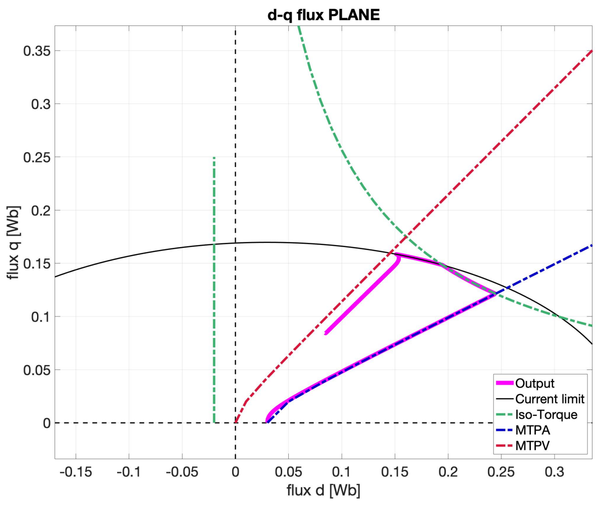


(b)

Figure 6.14: *Operating point performance test results for* EM3 *(R-SM), shown in the current (a) and flux (b) reference frames.*

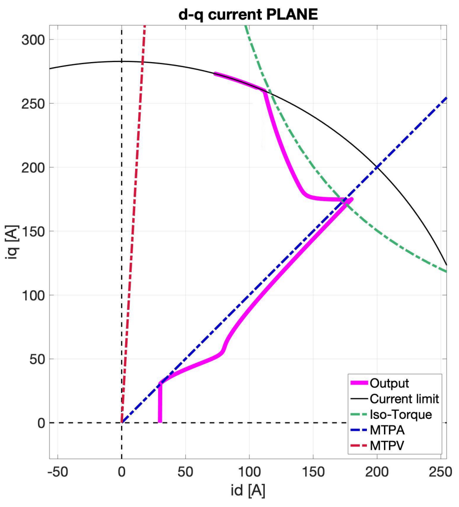


(a)

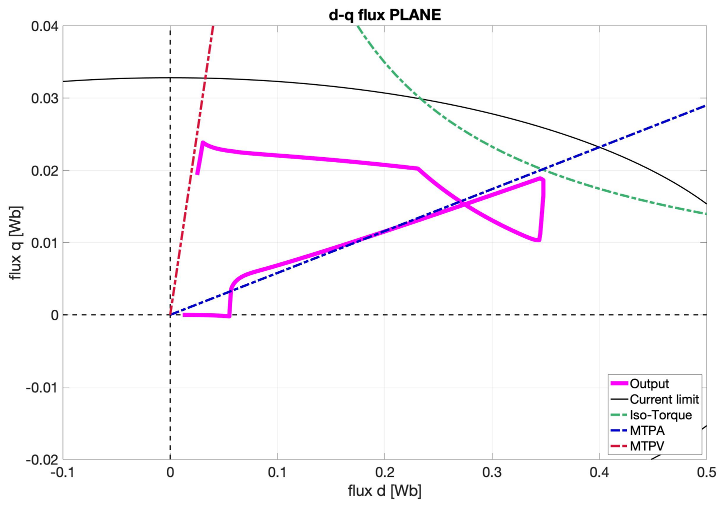


(b)

Figure 6.15: *Operating point performance test results for* EM4 *(WR-SM), shown in the current (a) and flux (b) reference frames.*



(a)



(b)

Figure 6.16: *Operating point performance test results for* EM5 *(IM), shown in the current (a) and flux (b) reference frames.*

6.3 FLUX AND INDUCTANCE CHARACTERIZATION

The nature of the magnetic saturation makes it very difficult to realistically produce analytical behaviours for testing. For this reason, the bulk of the development work for nonlinear control was carried out on a particular electric machine for which the required data was available from, in this case, FEM analysis and experiment confirmation at the testbench.

This method was almost ideal because the computer simulation produced a very detailed characteristic from which the conversion to the parameters needed for the control was relatively easy.

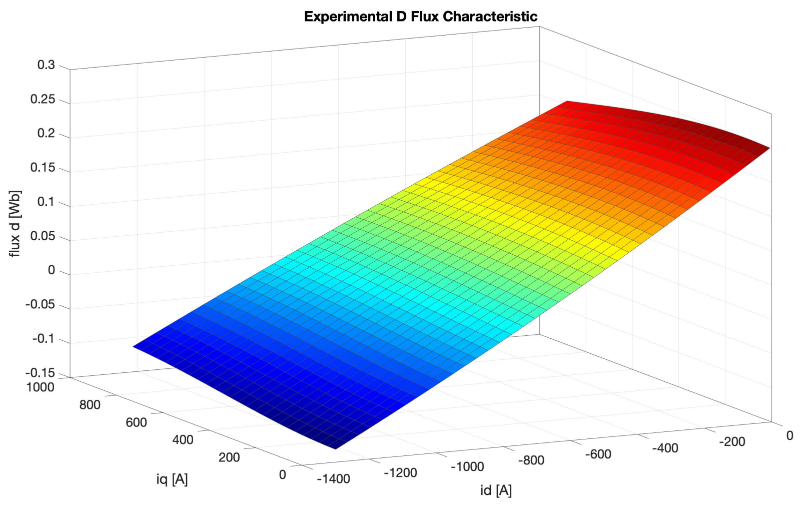
In this section some of this elaboration is presented to provide an explanation for some of the results and to showcase an example of nonlinear machine with both saturation and cross coupling magnetic links. Moreover, it was possible to obtain access to another type of high performance machine and partially carry out the characterization tests. These are also presented afterwards as an example of real machine data.

The first machine was an Internal Permanet Magnet Synchronous Machine with over 900 [A] of nominal current at 700 [V]. As mentioned, the magnetic Finite Elements Model was carried out to provide the machine characterization. The initial output of this elaboration was the equivalent inductances, but the flux maps were also provided. These proved to be the starting point to construct the needed behaviours through the calculation of the derivative inductances.

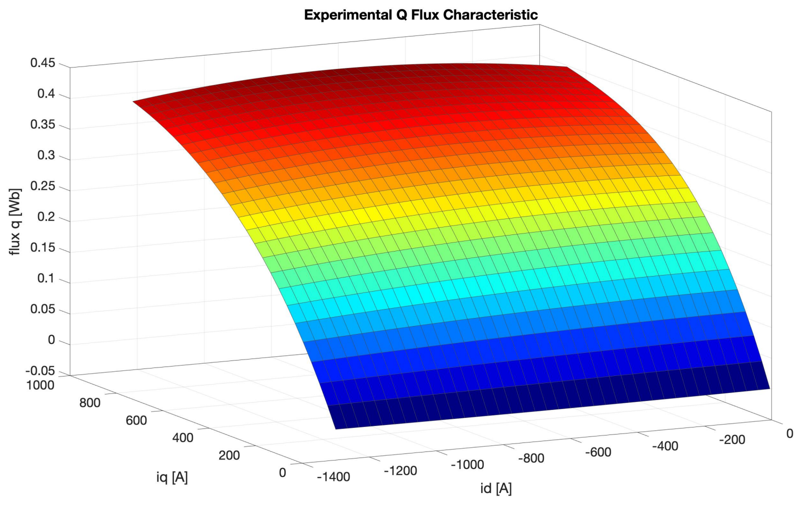
Therefore, Fig. 6.17 provides the 3D visualization of the machine fluxes as provided, with a very fine mesh of datapoints linearly interpolated.

In this case, the polynomial interpolation was chosen to approximate the curves and to obtain a continuous and derivable mathematical function which is the requisite for the inductances definition.

For this machine, a 3rd order d axis and 4th order q axis polynomial was deemed the best fit to either curve, rendering explicit the dependence of both fluxes from the two currents, as expected. The form for both axes, then, resembles (6.1), where the coefficients are intentionally not disclosed.



(a)



(b)

Figure 6.17: *Flux characteristics from FEM analysis for the IPM-SM under study. Flux of* d *axis (a) and of* q *axis (b) presented.*

$$\begin{aligned}\varphi = & k_{00} + k_{10} i_d + k_{01} i_q + k_{20} i_d^2 + k_{11} i_d i_q + k_{02} i_q^2 + k_{30} i_d^3 + \\ & + k_{21} id^2 i_q + k_{12} i_d i_q^2 + k_{03} i_q^3 + k_{31} i_d^3 i_q + k_{22} i_d^2 i_q^2 + \\ & + k_{13} i_d i_q^3 + k_{04} i_q^4\end{aligned} \tag{6.1}$$

Thanks to the polynomial form, the inductances are then easily produced by operating the partial derivatives on the two currents. For example, (6.2) shows the case of derivative in i_d for the polynomial in question.

$$\begin{aligned}\frac{\partial\varphi}{\partial i_d} = & k_{10} + 2k_{20}i_d + k_{11}i_q + 3k_{30}i_d^2 + 2k_{21}i_d i_q + k_{12}i_q^2 + \\ & + 3k_{31}i_d^2 i_q + 2k_{22}i_d i_q^2 + k_{13}i_q^3\end{aligned} \tag{6.2}$$

The four curves describing the four terms of the derivative inductance matrix that define the machine are thus presented in Fig. 6.18.

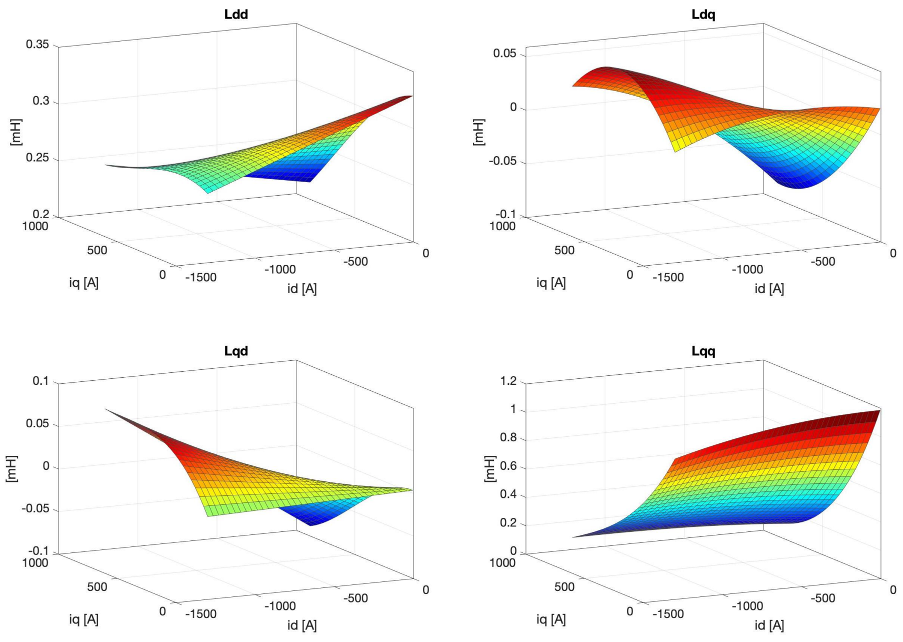


Figure 6.18: *Calculated derivative inductance maps for the four terms of the* [L] *matrix characteristic of the IPM-SM under study.*

At this point, it is also possible to derive the torque characteristic by applying the general formula (3.1) for every point currents point. The 3D representation of this is given in Fig. 6.19.

By knowing the torque behaviour, the optimal operating points given a certain current magnitude can be found by means of trying all the possible solutions. In fact, Fig. 6.20 shows the 3D torque contours which are in practice the ISOT used by the control algorithm. The operation can be done in both current and flux reference frames, which is very convenient for finding both MTPA and MTPV. It is possible to see directly how the curves are not exactly hyperbole with the same asymptote, but they are some other form of more complex hyperbole.

By brute force method the MTPA is consequently found as shown in Fig. 6.21 with even two types of interpolating methods, a linear and a cubic one.

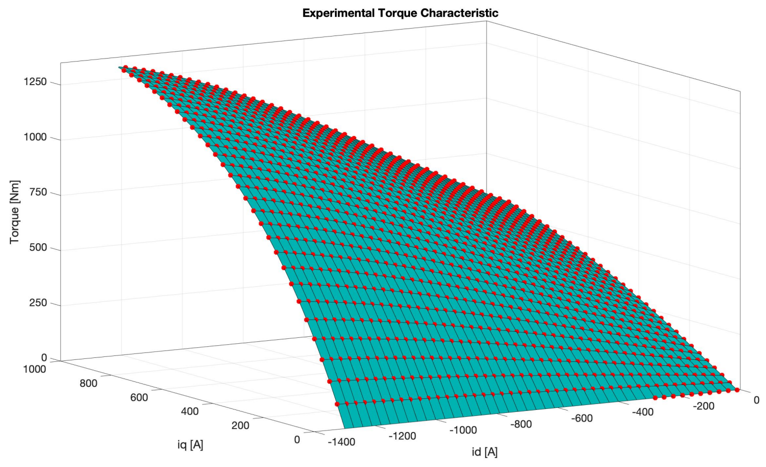


Figure 6.19: *Calculated torque characteristic based on the flux maps of the IPM-SM under study.*

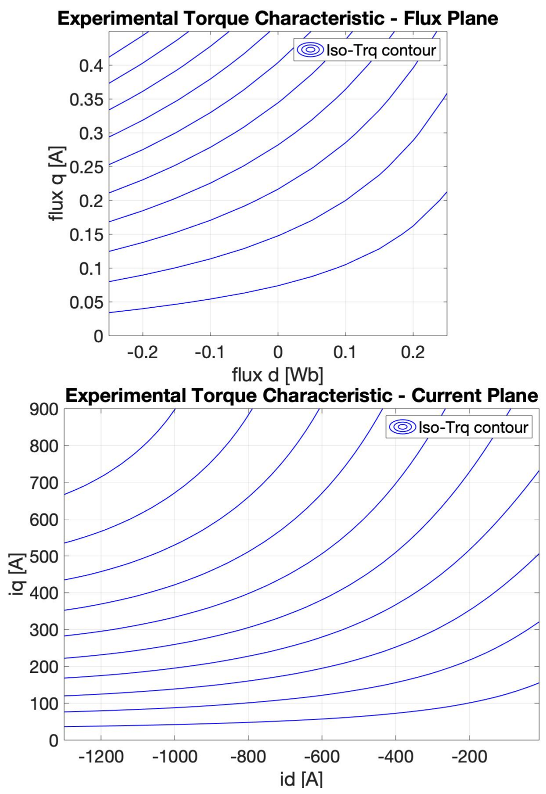


Figure 6.20: *Calculated torque characteristic contours in both current and flux reference frames for the IPM-SM under study.*

The same reasoning can be carried out in the flux reference frame to find the MTPV characteristic of the machine, as presented in Fig. 6.22 togehter with also the MTPA transformed from the current plane.

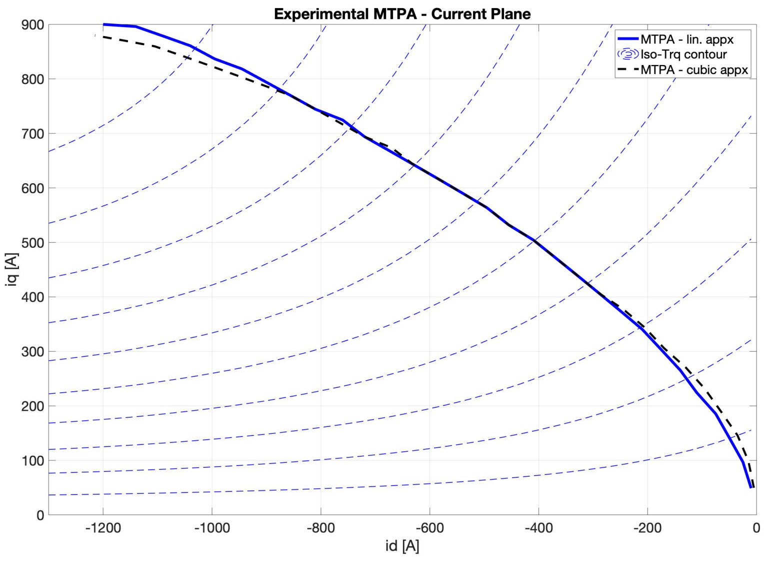


Figure 6.21: *MTPA attribute obtained via brute force in the current reference frame for the IPM-SM under study.*

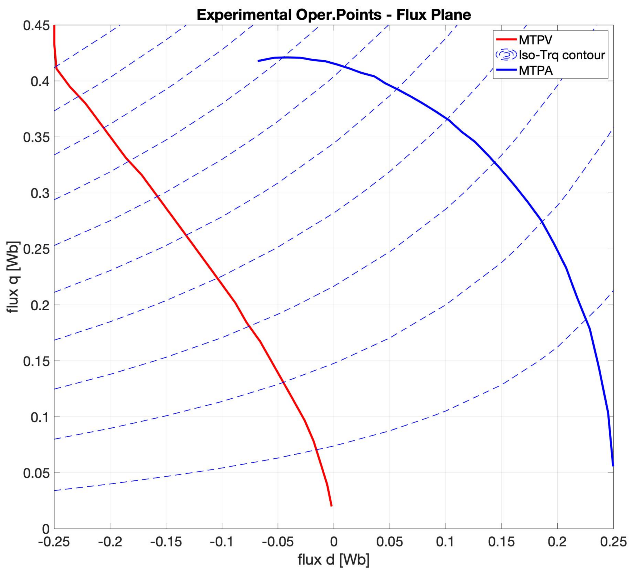


Figure 6.22: *MTPA and MTPV attributes obtained via brute force in the flux reference frame for the IPM-SM under study.*

As it can be seen, these curves are derived exclusively through purely geometric considerations without really employing any equations linking the values. They can, therefore, be considered at the same level as experimental results, thus providing a verification benchmark for the control algorithm. Isolating the desired curves and comparing them to the software outputs allowed the proper feedback correction during the development phase and provides now an intuitive performance comparison.

The other type of machine that was made available for a short time to conduct experimental characterization was an Surface Permanent Magnet Synchronous Machine.

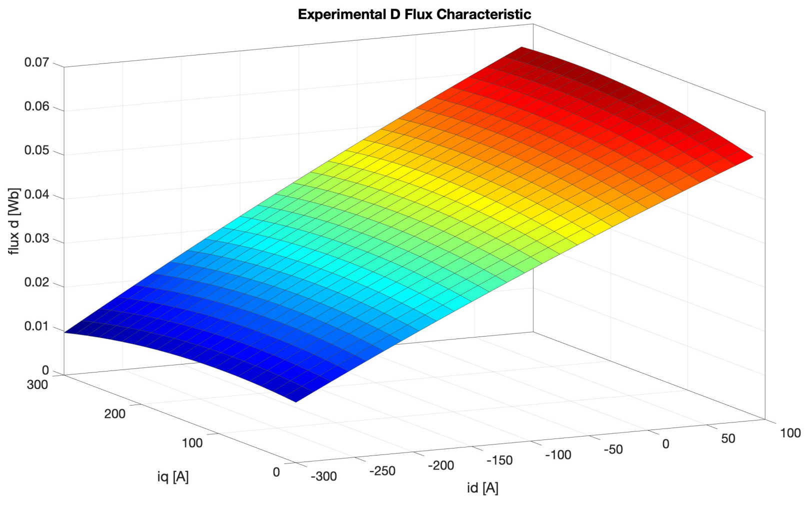
Instead of having the FEM analysis, the starting point of the characterization were testbench runs with the goal of collecting the data needed to operate the flux calculation through (3.18). Given the limited time, the datapoints were just a couple of dozen, but this is a testament to the capability of the polynomial interpolation to effectively smooth and extrapolate from little information.

Like for the other machine, the same calculation were carried out to provide experimental feedback and validation. Fig. 6.23 presents the interpolated data which, in this case, was carried out using a 2^{nd} order d axis and 2^{nd} order q axis polynomial.

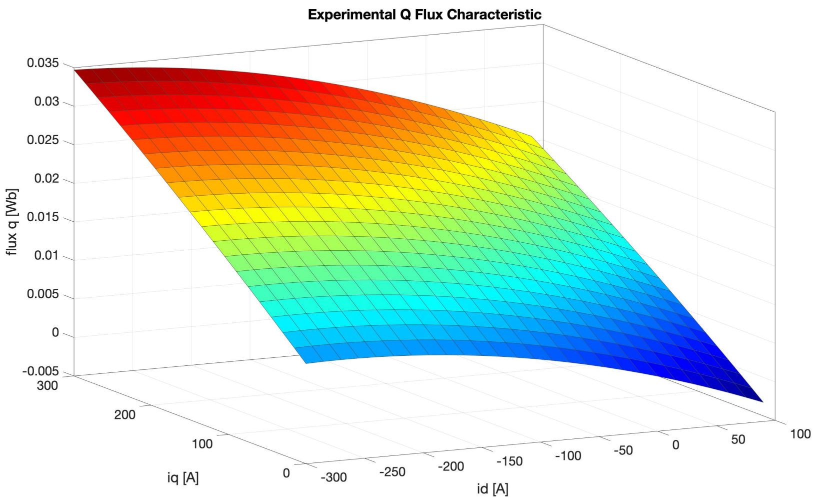
The derivatives were then calculated to obtain the inductances, Fig. 6.24. Given the lower order polynomial, the values are represented by planes in the 3D space. It is somewhat possible to notice that the machine has different values of L_{dd} and L_{qq} even if this type of motor is usually considered isotropic. This shows how the real applications can often differ substantially from the ideal case.

Same as for the IPM-SM, the torque can also be found easily and the contours of the 3D characteristic are presented in Fig. 6.25 in both current and flux reference frames.

Again using the brute force method the `MTPA` and `MTPV` curves are found by calculating for every torque value the least current or flux needed, respectively, to deliver it. Fig. 6.26 and Fig. 6.27 show the results of both features.



(a)



(b)

Figure 6.23: *Flux characteristics from testbench data for the SPMSM. Flux of* d *axis (a) and of* q *axis (b) presented.*

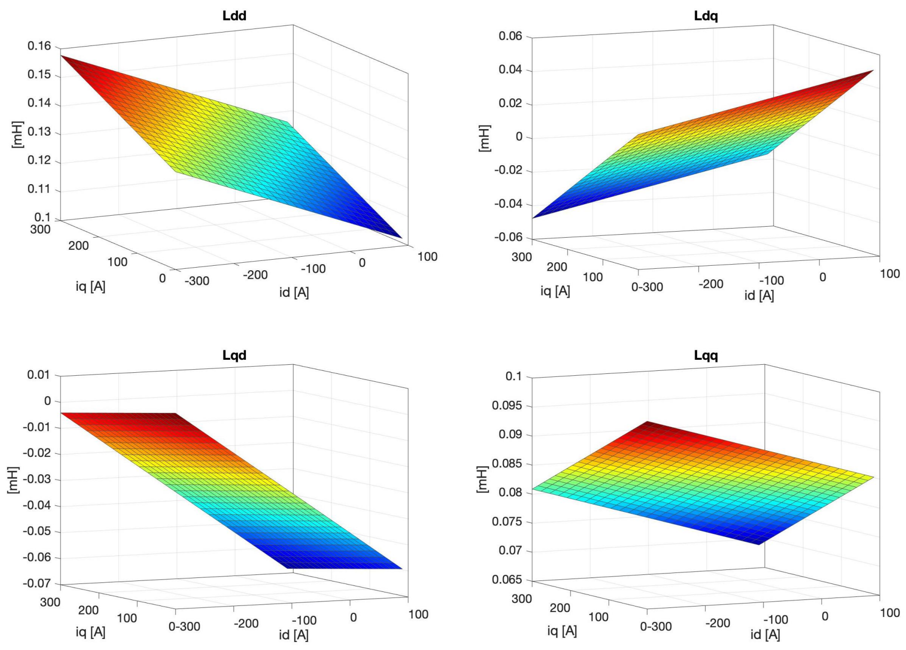


Figure 6.24: *Calculated derivative inductance maps for the four terms of the* [L] *matrix characteristic of the SPM-SM.*

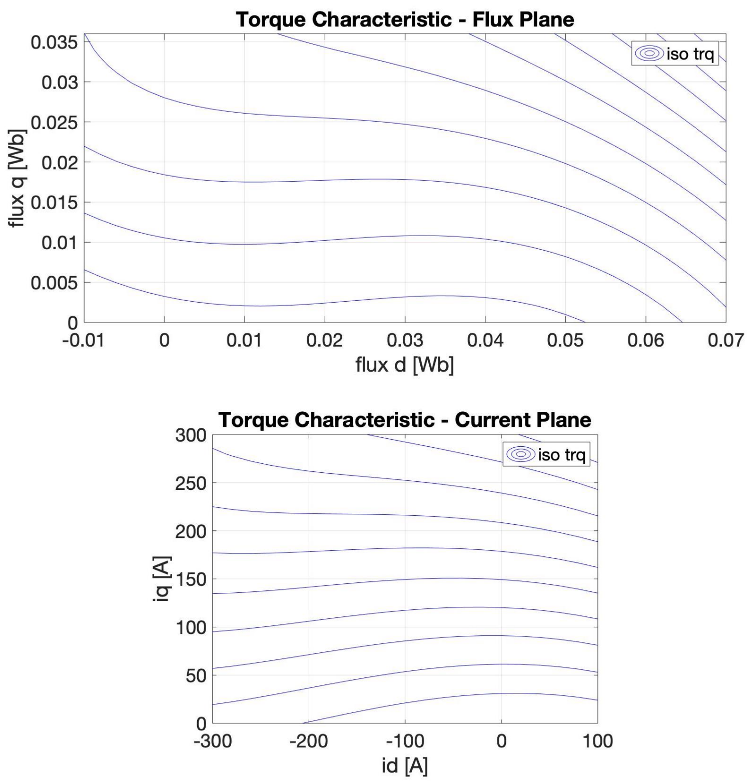


Figure 6.25: *Calculated torque characteristic contours in both current and flux reference frames for the SPM-SM.*

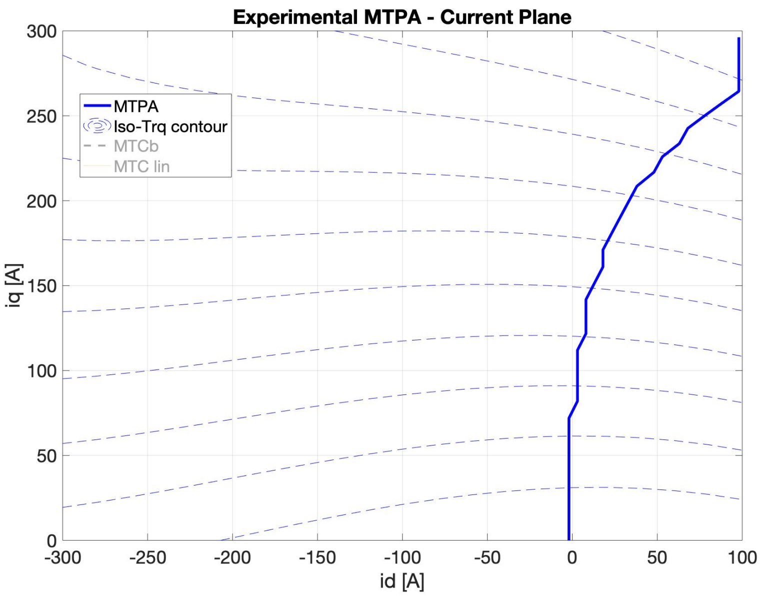


Figure 6.26: *MTPA attribute obtained via brute force in the current reference frame for the SPM-SM.*

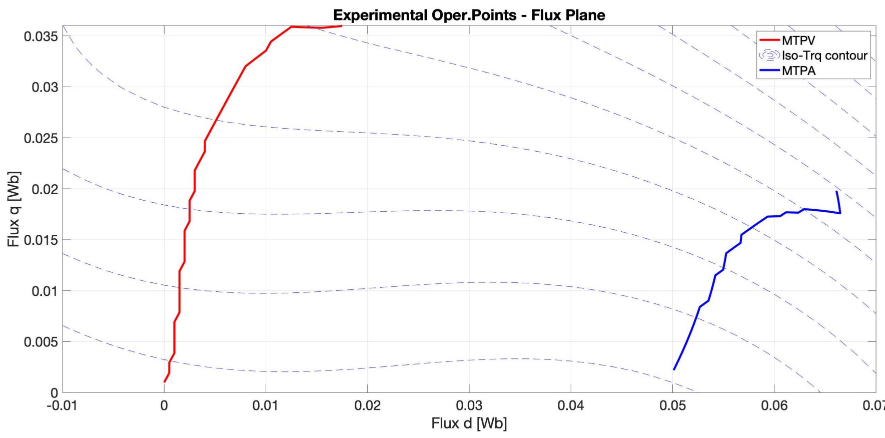


Figure 6.27: *MTPA and MTPV attributes obtained via brute force in the flux reference frame for the SPM-SM.*

6.4 NONLINEAR CONTROL AND FLUX WEAKENING

Finally, all of the characterizations, development and calibrations can come together to deliver a control algorithm that is capable of operating the electric machine in the magnetic saturation region analytically from the flux characteristics.

Unfortunately, as explained in the previous section, the experimental data to validate such results is limited. Moreover, the exact machine parameters and outputs are intentionally not the real ones in the simulations provided, although they have been modified in a way as not to introduce any comparability issues.

The figure that sums up the achievements is Fig. 6.28 where the IPM-SM is controlled during a torque ramp to 1500 Nm at zero speed, followed by a speed ramp to 8000 rpm, like in Fig. 6.11, to showcase the entire range of operation of the machine.

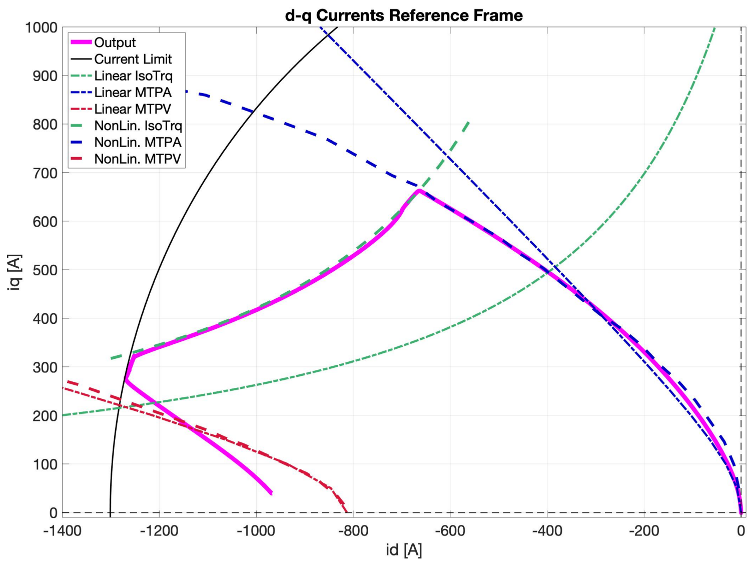


Figure 6.28: *Control algorithm output result in the current reference frame for IPM-SM with test to* 1500 Nm *at* 8000 rpm *compared with linear features and experimental optimal operating points for validation.*

The results are also provided in the flux reference frame in Fig. 6.29.

Both images show the linear operating curves as an additional confirmation of the substantial difference between the two types of approach and the potential loss of performance by not actuating the proper nonlinear control. As a reminder, Fig. 6.30 presents

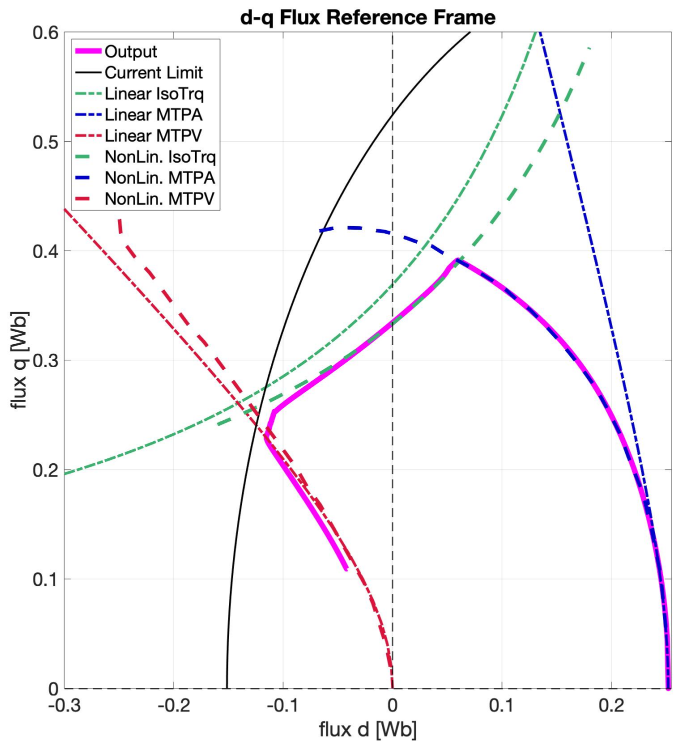


Figure 6.29: *Control algorithm output result in the flux reference frame for IPM-SM with test to* 1500 Nm *at* 8000 rpm *compared with linear features and experimental optimal operating points for validation.*

the same identical test with a comparable linear machine just to visually confirm the results.

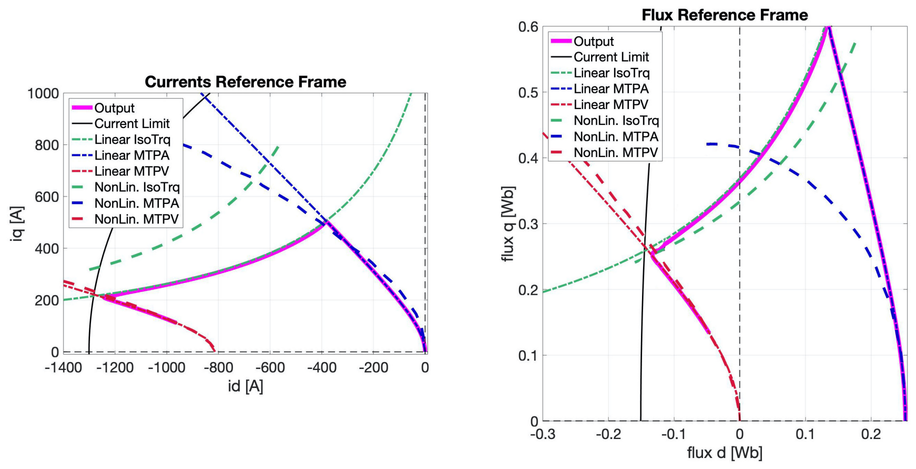


Figure 6.30: *Confirmation of results by showing the same test run using a comparable linear IPM-SM in both reference frames.*

It is clear to see that the control is able to provide the performance expected and follow the optimal curves and applying the correct limitations. There is some offset that emerges at high speed, especially visible in the MTPV region in the current reference frame. Given that those operating points are found by the flux weakening function operating int he flux reference frame and because the error is much smaller in that plane, it is possible that the parameters approximations are not precise enough in that particular part of the operation.

It is also important to keep in mind that the test is carried out with a non-indifferent dynamic component since the speed ramp is above 2000 $\frac{\text{rpm}}{\text{s}}$ and the deep flux weakening state is particularly difficult for stability reasons.

As a note, it can be seen how the flux state of the machine is not very different with or without the magnetic saturation, while the currents are much higher for the nonlinear one. This is a confirmation that more current is necessary to get the same flux.

To confirm the functionality of the control over a wider range of operation conditions, the next test was conducted requesting power outputs on all four of the quadrants. The torque is varied in steps both positive and negative, while the speed presents a positive ramp followed by one in the other direction, down to negative values.

Fig. 6.31 shows the torque and speed inputs and also the torque output estimate of the model. The second part also shows the machine's currents on both axes together with their references generated by the FOC. The behaviour of the control is in line with a perfect satisfaction of the requests in all the situations.

As additional information, Fig. 6.32 presentes the current and flux outputs on the respective d-q reference frames. The torque transitions are, as expected, dynamically taxing and in the regards the trajectories do not always follow the ideal curves.

Other machines have also been modelled with some approximations given that the real saturation interaction between the axes is not known a priori. The nonlinear results of the ramp tests designed to show the general behaviour are presented for a SPM-SM in Fig. 6.33 and in Fig. 6.34 for a IM, once again with the graphical comparison with the linear operating points for comparable machines.

In all cases, even if the full range is not obtained, it is clear to see that the nonlinear approach is sound and delivers the performance in accordance with the experimental data, although

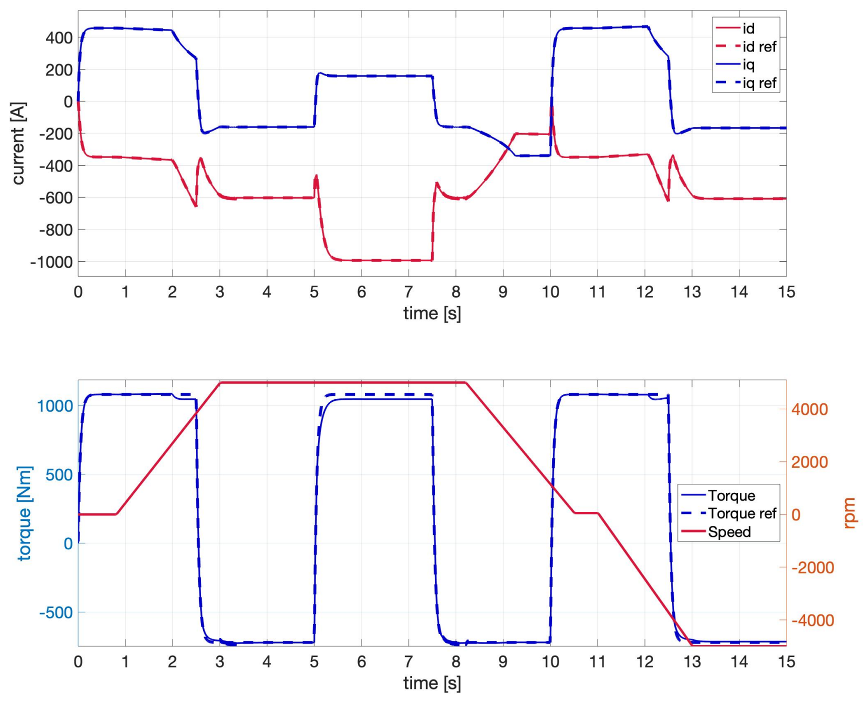


Figure 6.31: *Simulation outputs for test evaluating behaviour in the four quadrants and dynamic inputs. The plot above shows the speed and the torque reference plus output. Below the currents of the machine are presented.*

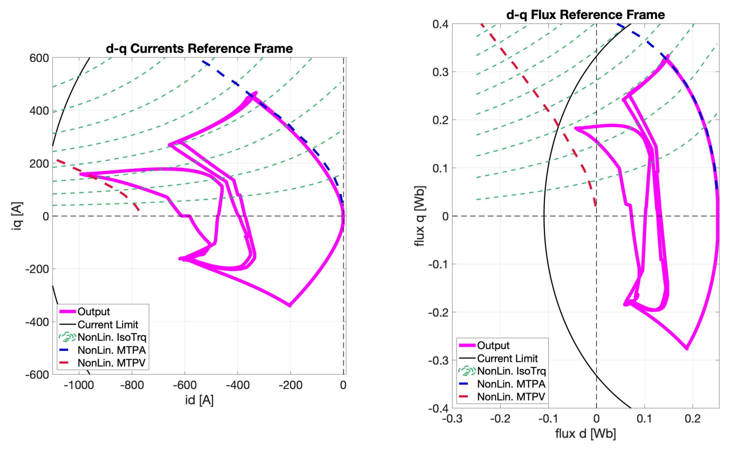


Figure 6.32: *Simulation outputs for test evaluating behaviour in the four quadrants and dynamic inputs. Overview in the current and flux planes of the operating points' trajectory.*

with some discrepancies to be attributed mainly to imperfect parameter calibration.

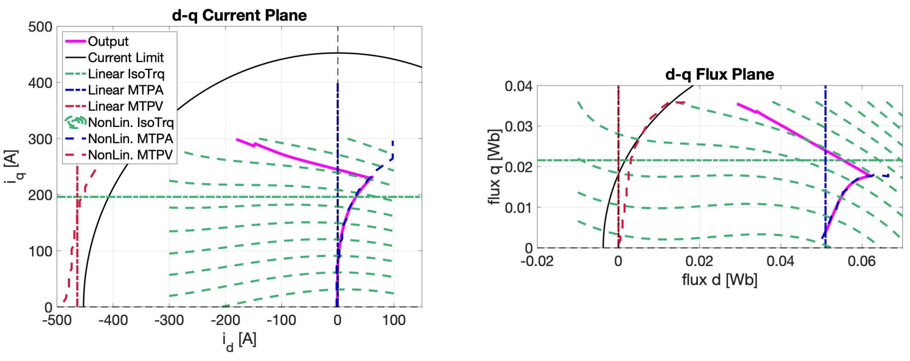


Figure 6.33: *Confirmation of nonlinear control results for SPM-SM in both reference frames during a torque ramp and subsequent speed ramp with comparison with comparable linear machine.*

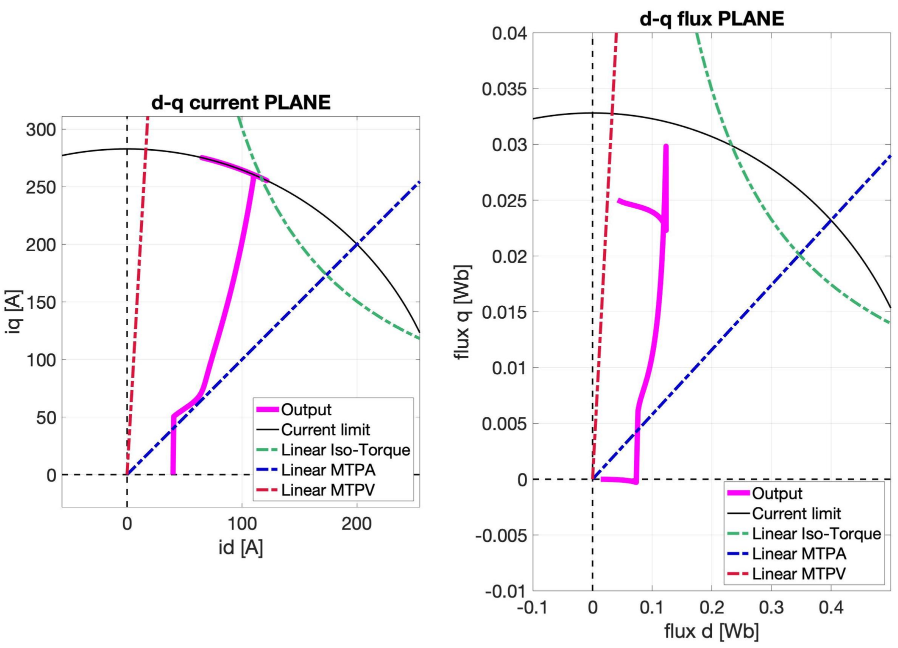


Figure 6.34: *Confirmation of nonlinear control results for IM in both reference frames during a torque ramp and subsequent speed ramp with comparison with comparable linear operating points.*

The last verification test presented has been carried out to show the behaviour of the FWC function variant that was developed to try to provide a control algorithm less reliant on the machine parameters. This could be especially helpful if preliminary test to characterize the device are not possible or very limited. As discussed in the dedicated section 4.4, however, this implementation requires an additional regulator.

The test consist in running the same torque and speed ramp first with the control having the same exact loaded characteristics as the ones used in the plant parameters and a second run where the control d inductance is multiplied by a factor before being used. This implies that the control has an error in the inductance matrix. For the example provided the factor was chosen to be fairly high at 1.2, giving a 20% increased Ld.

The results are provided in Fig. 6.35 which summarize both the variants of the FWC functions, tested in the same way. The dotted lines are the runs with the parameter error and it is clear that the variant is capable to be closer to the real machine.

It is also possible to see that the original version is more stable in deep flux weakening operation and it is still to be considered the better solution if the parameters are known with reasonable confidence.

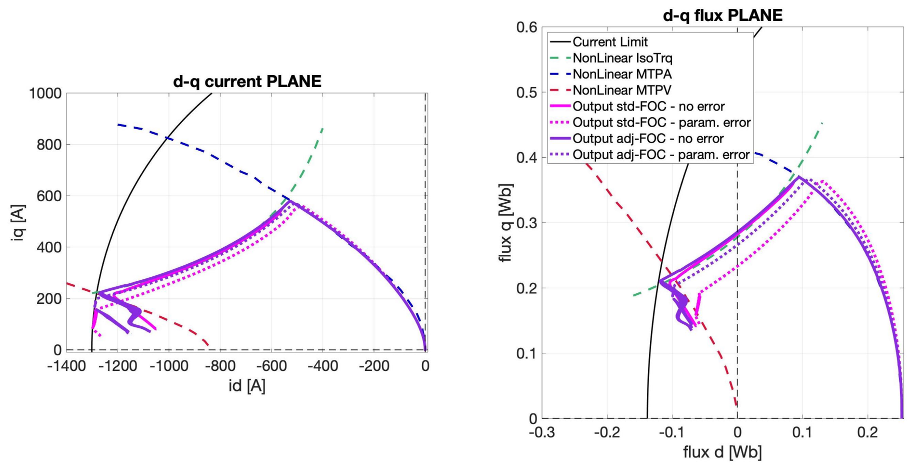


Figure 6.35: *Comparison between the FWC variants when introducing an error in the parameters of the control. The* adj *outputs are obtained with the variant function that is less sensible to the parameter variance.*

7 CONCLUSIONS

This work had the main objective to produce a Unified Machine Model control algorithm for the use on three-phase machines operating in magnetic saturation region and the results indicate that it was successful. Special care was given to construct a robust and flexible software development architecture specifically to allow teamwork and efficient automatic code generation for embedded devices.

The process started from the definition of the general flow of information among the various tasks of the function, to the subdivision into atomic functions with their specific goal determined by their outputs. The conventional naming of the variables and parameters was also established to ensure each name would carry sufficient information to be self-explanatory, thus helping the teamwork development.

The tasks were separated to allow maximum execution speed by keeping the highest priority task as light as possible.

The single functions were then developed for the linear machine. In particular the electric machine model was implemented utilising the Unified Machine Model approach which is completely general for any type of traditional machine such as IM, SMPM-SM, IPM-SM, ARPM-SM, R-SM and WR-SM. Such algorithm provides the ability of changing just a few parameters to model any machine and calculate its complete state in both flux and current space. This information can be then used by the other parts of the control to improve the behaviour of the system and provide useful estimations, like the torque.

The machine model has been implemented through an iterative integration routine that combines the simplicity of a forward Euler scheme with high accuracy thanks to the interval sub-division in smaller steps. This method has been validated against other standard ones in the simulation environment highlighting its advantages and it has been noted how it also offers the possibility to tune its performance to the application, if needed.

The other focus of the research was devoted to the Field Oriented Control and the generation of the best operating point condition accounting for the current and voltage limit. The key aspect at the center of the algorithm was the approximation of the torque curve with a linear function, thus rendering the search for the minimum a much simpler line-circle intersection, rather than a quartic equation. Moreover, the use of the flux reference frame was exploited to consider the voltage limit as a simple circle. In this way the application of the limit could be carried out using the same principles as the current one with the same high level of execution efficiency.

All the other parts of the control were also developed with the highest standards and the full algorithm was tested thanks to the collaboration with the FEV Gmbh company on their experimental testbench.

In order to develop a robust control system which is able to control a nonlinear machine, a new linearization method, starting from the experimental magnetic characterization was incorporated in the algorithm.

The machine inductance parameters were then found through the general definition of derivative of the flux and incorporated in the model with the use of a linear approximation around a measured operating point to enable the solution of the equations.

The unified implementation was partially sacrificed in the name of a more efficient SM model, but the FOC part is still universal and based on the same considerations as the linear machine, just with more complex equations.

The new nonlinear control has been validated for selected machines in the MiL environment and at the testbench, proving that the concept is sound and efficient in delivering the desired performance out of machines operating in the saturation region. The real-world tests also provided many improvement points to be added like noise reduction, limited operating modes, acquisition conditioning, selectable options for application adjustment and improved safety measures.

The final result, therefore, has been experimentally proven effective and a novel solution to the control in the magnetic saturation region.

Future development would see this control be verified at the testbench with even more types of electric machines. Its modular architecture is also well suited for is use as a based point for the research of specific improvements in just some parts of the algorithm like, for example, resonant regulators or sensorless operation. Another big area of development would be the extension of the presented ideas for multi-phase machines by increasing the number of state spaces, defining FOC goals for the higher harmonics and implementing advanced safety features.

A PARAMETERS CONFIGURATION SCRIPT EXAMPLE

In this appendix is presented an example of the script where all the parameters for the application software are configured.

```
%%%                     Universal Inverter Software                     %%%
%%% Parameters Configuration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
 %
 % Description : Loading configuration parameters for Lemad/FEV Universal
 %               Inverter Software
 %
 %     Version : 30.2 (03/09/21)
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clearvars

%/*************************************************************************/%
%%% Input: MPA Machine Parameters:
% Select Machine Type:
%   1 = IM
%   2 = SM-PMSM
%   3 = I-PMSM / AR-PMSM
%   4 = R-SM
%   5 = WR-SM
mpa_idxEmType_C   = 2;
mpa_noPolePairs_C = 4;          %[-] Pole Pairs

if mpa_idxEmType_C == 1
    mpa_resStatr_C = 0.01;      %[Ohm] Rs
    mpa_resRotr_C  = 0.006;     %[Ohm] Rr
else
    mpa_resStatr_C = 0.007;    %[Ohm] Rs
    mpa_resRotr_C  = inf;       %[Ohm] Rr
end
mpa_indStatrD_C     = 0.0007;  %[H] Ld
mpa_indStatrQ_C     = 0.001;   %[H] Lq
mpa_psiMagnStatrD_C = 0.14;    %[Wb] F_extS

% for IM
mpa_inStatrDispersion_C = 0.00002; %[H] stator leakage inductance for IM
mpa_inRotrDispersion_C  = 0.00004; %[H] rotor leakage inductance for IM
mpa_tiTauFilterMapCurrents_C = 0.0002; %[s] low pass filter map input current

%/*************************************************************************/%
%%% Input: STC Switching Time Control:
stc_tiTaskPwm_C  = 0.0002;    %[s]
stc_tiTaskFoc_C  = 0.0004;    %[s] not considered for now
stc_tiTaskThm_C  = 0.1;       %[s]

%/*************************************************************************/%
%%% Input: IPA Inverter Parameters:
```

```
ipa_noIntegrIter_C          = 5;   % integration sub-intervals range:[2-10]
ipa_uVoltStaticRsrv_C       = 20;  %[V] 70V is for FWC_B, for A around 10V

%%%--------------------
% Regulator calibration parameters depending on machine parameters:
if mpa_idxEmType_C == 1
    % IM
    sLsd = 5.9e-5;  % sigma*Lsd
    sLsq = 6e-5;    % sigma*Lsq
    Ld = (1 + (mpa_resRotr_C/mpa_resStatr_C)) * sLsd;
    Lq = sLsq;
    kkd_z = 0.4; kkd_p = 0.1;
    kkq_z = 0.4; kkq_p = 0.1;
else
    % SM
    Ld = mpa_indStatrD_C;
    Lq = mpa_indStatrQ_C;
    kkd_z = 0.6; kkd_p = 0.1;
    kkq_z = 0.6; kkq_p = 0.1;
end
%%%--------------------

% D axis
p_loadD = mpa_resStatr_C / Ld;         % load pole: r/L
z_regD  = kkd_z * p_loadD;             % reg zero -> decided based on load pole
p_fastD = kkd_p * (1/stc_tiTaskPwm_C); % fast pole << tc

ipa_noRegDPropCoeff_C = ((mpa_resStatr_C*p_fastD)+(Ld*p_fastD^2))/...
                        (p_fastD+z_regD);                    %kp
ipa_noRegDIntegrCoeff_C = z_regD * ipa_noRegDPropCoeff_C;     %ki

% Q axis
p_loadQ = mpa_resStatr_C / Lq;         % load pole: r/L
z_regQ  = kkq_z * p_loadQ;             % reg zero -> decided based on load pole
p_fastQ = kkq_p * (1/stc_tiTaskPwm_C); % fast pole << tc

ipa_noRegQPropCoeff_C = ((mpa_resStatr_C*p_fastQ)+(Lq*p_fastQ^2))/...
                        (p_fastQ+z_regQ);                    %kp
ipa_noRegQIntegrCoeff_C = z_regQ * ipa_noRegQPropCoeff_C;     %ki

clearvars p_loadD z_regD p_fastD p_loadQ z_regQ p_fastQ kkd_z kkd_p kkq_z kkq_p;

ipa_tiDeadTimeHW_C       = 2e-6;  %[s] Dead Time used for "hardware"
ipa_tiDeadTimeSVM_C      = 2e-6;  %[s] Dead Time used for SVM compensation
ipa_tiDeadTimeFbk_C      = 2e-6;  %[s] Dead Time used for FeedBack compensation
ipa_iLimitDeadTimeSignSwitch_C = 20; %[A] Current threshold to decrease DT @ 0A

%/*************************************************************************/%
%%% Input: CMX Current Max:
if mpa_idxEmType_C == 1
    cmx_iRmsMax_C   = 300;  %[A]
else
    cmx_iRmsMax_C   = 520;  %[A]
end

%/*************************************************************************/%
%%% Input: ASE Angle Speed Estimation:
ase_bResolEncodSlct_C     = 1;      %[] 1=resolver   0=encoder
ase_noIntegrWrapTime_C    = 50000;  %[] change if PWM tc different from FOC tc
ase_noIntegrWrapCycle_C   = 50;     %[] change if PWM tc different from FOC tc
ase_bAngleDirectionSlct_C = 0;      %[] Angle direction selection [0 or 1]
ase_bActivateFilter_C     = 1;      % filter select
ase_tiTauFilter_C         = 0.001;  %[s] time constant speed filter

%/*************************************************************************/%
%%% Input: SUP Input Resolver:
```

```
sup_idxPredictSlct_C   = 2; % Prediction Mode: 0=no pred, 1=Hybrid, 2=predictive
sup_bResolZeroReq_C    = 0; % if it is =1, then the zero procedure is activated
sup_agResolZeroValue_C = 0; %[rad] correction angle, to set after zeroing

%/*********************************************************************/%
%%% Input: EMS electric Model Solver:
ems_bEMModelReset_C = 0;   % reset of the Electric Machine Model integration
ems_noIFbkGainD_C   = 0.8; %[pu] iD fbk gain, in pu, stability range implied
ems_noIFbkGainQ_C   = 0.8; %[pu] iQ fbk gain, in pu, stability range implied
ems_biFbkAlgorithSlct_C = 1; % Current feedback algorithm selection. 1=Complete,
                             % 2=no speed comp., 3=lin machine, 4=const, 5=no fbk
ems_bCurrentCalcSlct_C = 3;  % current prediction mode:

%/*********************************************************************/%
%%% Input: CCR Current Correction:
ccr_bCurrentCalcSlct_C  = 0;   % 1=current prediction is delta calculation

%/*********************************************************************/%
%%% Input: SVE Stator Voltage Estimator:
sve_idxVoltSourceSlct_C = 2;  % 1=measurements, 2=from DutyCycles, 3=from Vref

%/*********************************************************************/%
%%% Input: VUT Unit Vector T:
vut_idxTrigApproxMethod_C = 3;   % 1=CORDIC, 2=polinomial, 3=no approx

%/*********************************************************************/%
%%% Input: VRG Voltage Regulators:
vrg_bRegsReset_C   = 0;   % regulators reset. 1=reset
vrg_bSatActivate_C = 1;   % 1=saturation on the regs is active

%/*********************************************************************/%
%%% Input: DTI Dead Time Introduction:
dti_bUseDeadTimeComp_C = 0;  % 1= activate the compensation

%/*********************************************************************/%
%%% Input: VSM Vs Max:
vsm_bDutyCycleDelaySlct_C = 0; % 1=unit delay on the input duty cycle

%/*********************************************************************/%
%%% Input: VRS Voltage Regulator Saturator:
vrs_bSaturatedVoltOriginSlct_C = 0; % 1=use volt sat by duty cycle, =0 from SVM

%/*********************************************************************/%
%%% Input: SVM Space Vector Modulation:
svm_idxSaturPrefSlct_C = 3; % 3=sat with no axis preference, 1=d pref, 2=q pref
svm_puDutyRangeMax_C   = 1; % max pu range of duty cycle [0-1] 1=all cycle range

%/*********************************************************************/%
%%% Input: FOA Field Oriented Angle:
foa_bActivateFilter_C = 1;       % 1=low pass filter on DeltaAgRotrflx active
foa_tiTauFilter_C     = 0.001;   %[s] low pass filter time constant

%/*********************************************************************/%
%%% Input: FOS Field Oriented Speed:
fos_bActivateFilter_C = 1;       % 1=low pass filter on RotrFlx speed active
fos_tiTauFilter_C     = 0.005;   %[s] low pass filter time constant

%/*********************************************************************/%
%%% Input: FOT Field Oriented Transform:
fot_idxTrigApproxMethod_C  = 3;  % 1=CORDIC, 2=polinomial, 3=no approx
fot_idxTrigFcnNumberSlct_C = 1;  % 1=use 6 trig functions, 2=use 4 trig func.

%/*********************************************************************/%
%%% Input: CSD Current Setpoint Determination:
csd_noIdRefLoopIter_C  = 3;   %[] number of iteration for calc of id MTPA req
if mpa_idxEmType_C == 1
```

```
    % IM
    csd_idLowerBound_C = 30; %[A] min current for IM, max current for SM
else
    % SM
    csd_idLowerBound_C = 0;  %[A] min current for IM, max current for SM
end

%/**************************************************************************/%
%%% Input: FWC Field Weakening Control:
fwc_noFluxRegDPropCoeff_C    = 2000;  %[] kp for D-flux regulator
fwc_noFluxRegDIntegrCoeff_C  = 20000; %[] ki for D-flux regulator
fwc_psiMaxModule_C           = 4;     %[Wb] max flux module
fwc_tiTauFilterPsiDCalc_C    = 0.001; %[s] low pass filter on D Flux Calc
fwc_tiIsedFbkFilter_C = 0.001; %[s] filter on delta current for current fbk
fwc_isedMeasFdbk_C    = 0;     %[]  gain on current fdk

% just for version A
fwc_tiTauFilterPsiDRef_C  = 0.005; %[s] low pass filter t-const on D Flux Ref
fwc_bMaximizeU_C    = 1; % 1=vector calc for maxFlux, 0=faster, less accurate

% just for version B
fwc_noFluxRegQPropCoeff_C     = 2000;   %[] kp for Q-flux regulator
fwc_noFluxRegQIntegrCoeff_C   = 20000;  %[] ki for Q-flux regulator
fwc_tiTauFilterVdc_C          = 0.0002; %[s] lowpass filter DC Voltage
fwc_tiTauFilterFluxStatrMax_C = 0.0005; %[s] lowpass filter Max Stator Flux
fwc_tiTauFilterFluxInCalc_C   = 0.0008; %[s] lowpass filter Flux input calc
fwc_tiTauFilterIQMax_C        = 0.001;  %[s] lowpass filter Max Iq calc
fwc_tiTauFilterIQLim_C        = 0.001;  %[s] lowpass filter Iq Lim for IqMax
fwc_tiTauFilterFemMax_C       = 0.001;  %[s] lowpass filter EMF Max for IqMax

fwc_idxSlctRegCoeffVar_C       = 0;    % 1= use variable reg. coefficients
fwc_noVarPropCoeffRegFluxD_C   = 0.15; %[] coeff for variable kp for D-flux
fwc_noVarIntegrCoeffRegFluxD_C = 50;   %[] coeff for variable ki for D-flux
fwc_noVarPropCoeffRegFluxQ_C   = 0.5;  %[] coeff for variable kp for Q-flux
fwc_noVarIntegrCoeffRegFluxQ_C = 40;   %[] coeff for variable ki for Q-flux

%/**************************************************************************/%
%%% Input: IQR Iq reference:
iqr_tiTauFilterIqMax_C = 0.004; %[s] lowpass filter iq Max
iqr_tiTauFilterIq_C    = 0.004; %[s] lowpass filter iq ref

%/**************************************************************************/%
%%% Input: SPD Set PWM Duty:
spd_bDeadTimeAdjust_C = 1; % 1=output duties adjusted to consider the DT
```

B REGULATOR CALIBRATION

B.1 DIGITAL REGULATOR IMPLEMENTATION

The general transfer function of a PI regulator is (B.1).

$$u = \frac{K_p s + K_i}{s} \varepsilon \tag{B.1}$$

Given that a generic transfer function must be expressed as a ratio between polynomials, the form $u = K_p \varepsilon + \frac{k_i}{s} \varepsilon$ is not correct. It can then be implemented in a digital system with the following recursive expression (B.2).

$$u_n = \left(K_p + \frac{t_c K_i}{2}\right) \varepsilon_n - \left(K_p - \frac{t_c K_i}{2}\right) \varepsilon_{n\text{-}1} + u_{n\text{-}1} \tag{B.2}$$

In the application the values not dependent on variable can be calculated separately, improving performance. Specifically:

$$k_p = \left(K_p + \frac{t_c K_i}{2}\right) k_i = \left(K_p - \frac{t_c K_i}{2}\right)$$

Equation (B.2) is obtained using the Tustin approximation $s \approx \frac{2}{t_c} \frac{1-z^{-1}}{1+z^{-1}}$ following the steps (B.3), which approximates the integral of a generic quantity with the area of the trapezoid subtended in a sampling interval and it is also known as trapezoidal method. See Fig. B.1.

$$\begin{aligned}
u &= K_p \varepsilon + \frac{K_i}{s} \varepsilon \\
s \cdot u &= K_p \cdot s \cdot \varepsilon + K_i \varepsilon \\
\frac{2}{t_c} \frac{1-z^{-1}}{1+z^{-1}} u &= K_p \frac{2}{t_c} \frac{1-z^{-1}}{1+z^{-1}} \varepsilon + K_i \varepsilon \\
(1-z^{-1}) u &= K_p (1-z^{-1}) \varepsilon + \frac{t_c}{2} K_i (1+z^{-1}) \varepsilon \\
u_n - u_{n\text{-}1} &= K_p (\varepsilon_n - \varepsilon_{n\text{-}1}) + \frac{t_c}{2} K_i (\varepsilon_n + \varepsilon_{n\text{-}1}) \\
u_n &= \left(K_p + \frac{t_c K_i}{2}\right) \varepsilon_n - \left(K_p - \frac{t_c K_i}{2}\right) \varepsilon_{n\text{-}1} + u_{n\text{-}1}
\end{aligned} \tag{B.3}$$

The presence of the two s in the transfer function involves the presence of two state variables in the digital function, for example $\varepsilon_{n\text{-}1}$ and $u_{n\text{-}1}$.

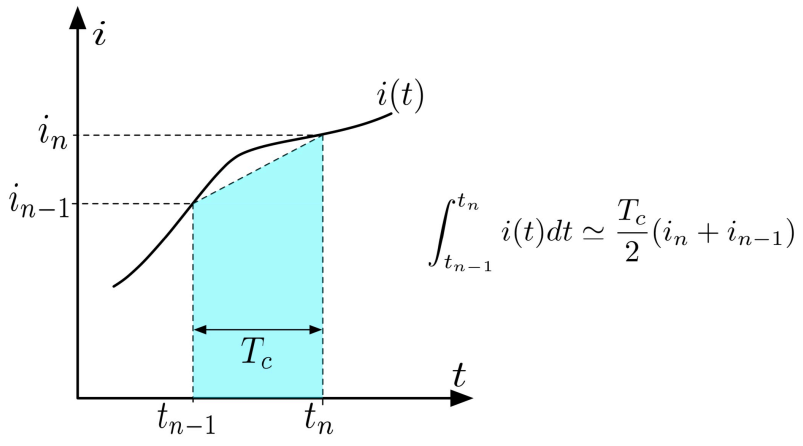

i
$i(t)$
i_n
i_{n-1}
$\int_{t_{n-1}}^{t_n} i(t)dt \simeq \frac{T_c}{2}(i_n + i_{n-1})$
T_c
t
t_{n-1}
t_n

Figure B.1: *Example of Tustin/trapezoidal approximation.*

The controller can then be implemented through the following steps:

1. calculation of the error at the current instant ε_n;
2. recalculation of the regulator output
$$u_n = \left(K_p + \frac{t_c K_i}{2}\right)\varepsilon_n - \left(K_p - \frac{t_c K_i}{2}\right)\varepsilon_{n\text{-}1} + u_{n\text{-}1};$$
3. storage of the state variables for the following cycle $\varepsilon_{n\text{-}1} = \varepsilon_n$ and $u_{n\text{-}1} = u_n$.

As an added advantage, this type of implementation solves the problem of the regulator saturation. As a matter of fact, when the output exceeds the allowed limit u_{lim} all that is required to saturate it is placing

$$u_n = u_{\text{lim}}.$$

It is, however, necessary to recalculate the state variables in order to make them consistent with the corrected (i.e. limited) output of the regulator. Since $\varepsilon_{n\text{-}1}$ and $u_{n\text{-}1}$ are state variables, which means they will be used in the next cycle, and having set $u_n = u_{\text{lim}}$, the steps in (B.4) are necessary.

$$\begin{aligned}
\varepsilon_n^{\text{temp}} &= \frac{u_{\text{lim}} + \left(K_p - \frac{t_c K_i}{2}\right)\varepsilon_{n\text{-}1} - u_{n\text{-}1}}{\left(K_p + \frac{t_c K_i}{2}\right)} \\
\varepsilon_{n\text{-}1} &= \varepsilon_n = \varepsilon_n^{\text{temp}} \\
u_{n\text{-}1} &= u_n = u_{\text{lim}}
\end{aligned} \tag{B.4}$$

Where $\varepsilon_n^{\text{temp}}$ is a temporary working variable.

The error recalculation for the next cycle has the following meaning:

during the current sampling time the actual error had not been correctly defined by the difference between the reference and the measurement, but rather by the exact quantity that the regulator would bring to the saturation.

This meaning is demonstrated by (B.5).

$$\begin{aligned} u_n &= u_{lim} \\ u_{lim} &= \left(K_p + \frac{t_c K_i}{2}\right)\varepsilon_n - \left(K_p - \frac{t_c K_i}{2}\right)\varepsilon_{n-1} + u_{n-1} \\ \varepsilon_n &= \frac{u_{lim} + \left(K_p - \frac{t_c K_i}{2}\right)\varepsilon_{n-1} - u_{n-1}}{\left(K_p + \frac{t_c K_i}{2}\right)} \end{aligned} \tag{B.5}$$

Looking at it another way, when the saturation occurs the error $\varepsilon_n = i_n^{ref} - i_n^{meas}$ is recalculated so that the new *virtual* error ε_n^{virt} creates a *virtual* reference $i_n^{ref,virt} = i_n^{meas} + \varepsilon_n^{virt}$ that brings the regulator exactly to saturation. The concept is visualised in Fig. B.2.

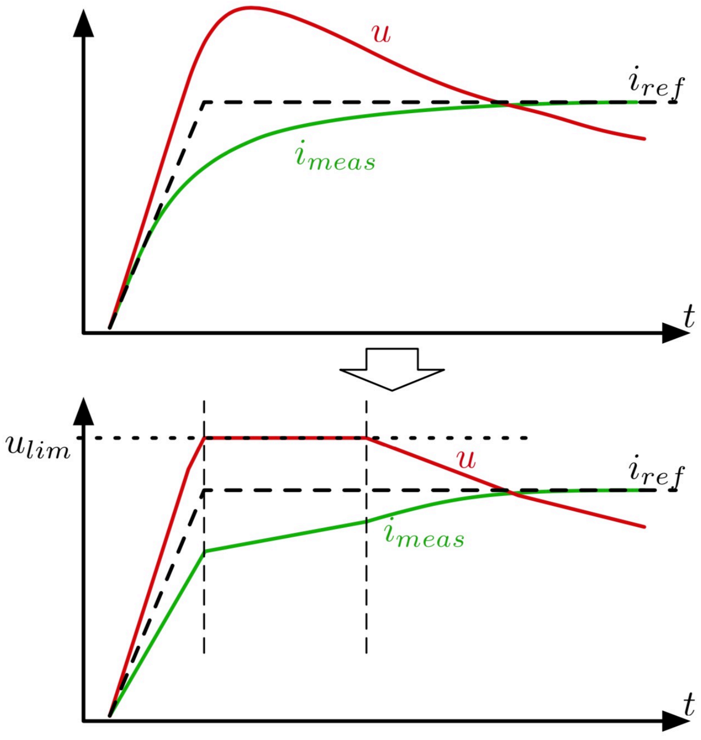


Figure B.2: *Visualisation of a saturated regulator behaviour.*

B.2 CLOSED LOOP SYSTEM REGULATORS

In the event that the regulator has to manage a first-order system (or a good approximation of it as, for example, the electric motors typically are), the transfer function of the load is expressed in (B.6) in continuous time.

$$\mathrm{TF}_{\mathrm{load}} = \frac{1}{Ls + r} \tag{B.6}$$

The system is then controlled by the regulator, whose response depends on the calibration of its parameters K_p and K_i. To study the whole system it is therefore necessary to introduce the transfer function (B.7) of a PI regulator.

$$\mathrm{TF}_{\mathrm{reg}} = \frac{K_p \cdot s + K_i}{s} = K_p \frac{s + \frac{K_i}{K_p}}{s} = K_p \frac{s + z_{\mathrm{reg}}}{s} \tag{B.7}$$

It is easily spotted how the regulator introduces a zero, equal to $z_{\mathrm{reg}} = \frac{K_i}{K_p}$, that defines the position of the root places in the close-loop current regulation. To note also that the pole $p_{\mathrm{load}} = \frac{r}{L}$ of the ohmic-inductive load represents the equivalent circuit to be regulated.

The close-loop configuration representing the system is described by the scheme in Fig. B.3 and, consequently, it can be studied with its complete transfer function, introduced in (B.8).

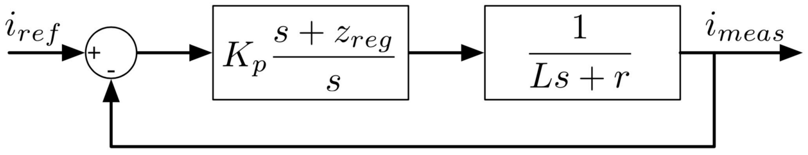


Figure B.3: *Close-loop system scheme.*

$$\begin{aligned}
\mathrm{TF} &= \frac{\mathrm{TF}_{\mathrm{load}} \cdot \mathrm{TF}_{\mathrm{reg}}}{1 + \mathrm{TF}_{\mathrm{load}} \cdot \mathrm{TF}_{\mathrm{reg}}} = \frac{\frac{K_p \cdot s + K_i}{s} \cdot \frac{1}{Ls+r}}{1 + \frac{K_p \cdot s + K_i}{s} \cdot \frac{1}{Ls+r}} = \\
&= \frac{K_p \cdot s + K_i}{L \cdot s^2 + (K_p + r)s + K_i} = K_p \frac{s + z_{\mathrm{reg}}}{L \cdot s^2 + (K_p + r)s + K_i} = \\
&= \frac{K_p}{L} \cdot \frac{s + z_{\mathrm{reg}}}{s^2 + \frac{K_p + r}{L}s + \frac{K_i}{L}}
\end{aligned} \tag{B.8}$$

The new equation can thus be studied to establish the system behaviour.

The first step is the verification that the close-loop poles do not have imaginary parts. To do this, (B.8) can be rewritten to highlight the roots as (B.9).

$$TF = \frac{K_p}{L} \cdot \frac{s + z_{reg}}{(s+p_1)\cdot(s+p_2)} \tag{B.9}$$

Where p_1 and p_2 are real roots poles of the denominator polynomial equation (B.10).

$$s^2 + \frac{K_p + r}{L}s + \frac{K_i}{L} = 0 \tag{B.10}$$

For which the relationship (B.11) is true.

$$p_2 \geqslant p_1 \tag{B.11}$$

Since, from the general solution:

$$s^2 + \frac{K_p + r}{L}s + \frac{K_i}{L} = 0$$

$$p_{1,2} = \begin{cases} \frac{(K_p+r) - \sqrt{(K_p+r)^2 - 4K_iL}}{2L} \\ \frac{(K_p+r) + \sqrt{(K_p+r)^2 - 4K_iL}}{2L} \end{cases} \Rightarrow \quad p_2 \geqslant p_1$$

Given the transfer function form (B.12)

$$TF = \frac{A_1}{\tau_1 s + 1} + \frac{A_2}{\tau_2 s + 1} = \frac{(A_1\tau_2 + A_2\tau_1)s + (A_1 + A_2)}{\tau_1 \cdot \tau_2 \cdot s^2 + (\tau_1 + \tau_2)s + 1} \tag{B.12}$$

And the equivalent form:

$$\begin{aligned} TF &= K_p \frac{s + z_{reg}}{L \cdot s^2 + (K_p + r)s + K_i} = \\ &= \frac{K_p}{K_i} \frac{s + \frac{K_i}{K_p}}{\frac{L}{K_i}s^2 + \frac{K_p+r}{K_i}s + 1} = \frac{\frac{K_p}{K_i}s + 1}{\frac{L}{K_i}s^2 + \frac{K_p+r}{K_i}s + 1} \end{aligned}$$

The following is derived:

$$\begin{cases} (A_1\tau_2 + A_2\tau_1) = \frac{K_p}{K_i} \\ (A_1 + A_2) = 1 \end{cases} \Rightarrow \begin{cases} A_1 = \frac{\tau_1 - \frac{1}{z_{reg}}}{\tau_1 - \tau_2} \\ A_2 = \frac{\frac{1}{z_{reg}} - \tau_2}{\tau_1 - \tau_2} \end{cases} \tag{B.13}$$

Therefore, given (B.13), the form (B.14) for the TF can be written.

$$TF = \frac{A_1}{\tau_1 s + 1} + \frac{A_2}{\tau_2 s + 1} \quad \text{where} \begin{cases} \tau_1 = \frac{1}{p_1} \\ \tau_2 = \frac{1}{p_2} \end{cases} \tag{B.14}$$

In particular, it can be verified that the response will depend on the relative position between the pole of the load (p_{load}) and the zero of the regulator (z_{reg}) and between the zero of the regulator (z_{reg}) and the regulator gain. It is therefore possible to draw the following conclusion that it is possible to distinguish different behaviours depending on this relationships.

Regulator zero lesser than load pole

If the zero of the regulator has a lower value than the load pole ($z_{reg} \leqslant p_{load}$), then the transfer function's poles are given by (B.15). In this case the poles never have imaginary parts.

$$p_{1,2} = \begin{cases} \frac{(K_p+r)-\sqrt{(K_p+r)^2-4K_iL}}{2L} \\ \frac{(K_p+r)+\sqrt{(K_p+r)^2-4K_iL}}{2L} \end{cases} \quad \text{with} \quad \frac{r}{L} \geqslant \frac{K_i}{K_p} \tag{B.15}$$

Which means that:

$$p_2 \geqslant p_{load} \geqslant z_{reg} \geqslant p_1 \quad \Rightarrow \quad \tau_1 \geqslant \frac{1}{z_{reg}} \geqslant \frac{1}{p_{load}} \geqslant \tau_2$$

Which means that the numerators are (B.16)

$$\begin{cases} A_1 = \frac{\tau_1 - \frac{1}{z_{reg}}}{\tau_1 - \tau_2} \in [0,1] \\ A_2 = \frac{\frac{1}{z_{reg}} - \tau_2}{\tau_1 - \tau_2} \in [0,1] \end{cases} \tag{B.16}$$

This all mean that the transfer function properties are (B.17) with a root representation like Fig. B.4 and, in this case, the poles never have imaginary parts.

$$\begin{cases} p_2 \geqslant p_{load} \geqslant p_1 \\ A_1, A_2 \in [0,1] \quad , \quad A_1 + A_2 = 1 \end{cases} \tag{B.17}$$

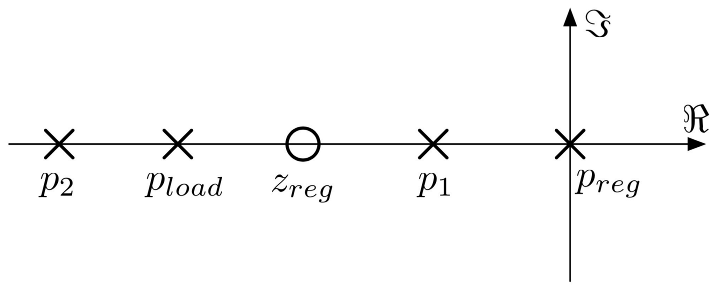


Figure B.4: *Root loci in case of* $z_{reg} \leqslant p_{load}$.

The answer to the unitary impulse is (B.18).

$$i(t) = A_1 \cdot e^{-\frac{t}{\tau_1}} + A_2 \cdot e^{-\frac{t}{\tau_2}} \tag{B.18}$$

It follows that the dynamic response is limited by the term described by the "slow" pole p_1, i.e. by:

$$TF_{p_1} = \frac{A_1}{\tau_1 s + 1}$$

It is easy to verify that the regulator gain acts by balancing the terms A_1 and A_2 in the sense that, as this gain increases, the coefficient A_2 keeps getting larger than A_1, thus yielding to (B.19).

$$K_p \to \infty \quad \Rightarrow \quad \begin{cases} A_1 \to 0 \\ A_2 \to 1 \end{cases} \tag{B.19}$$

Moreover, always increasing the regulator gain value, the "fast" pole p_2 tends to assume higher values, consequently increasing the dynamic response of the corresponding part of the transfer function:

$$TF_{p_1} = \frac{A_2}{\tau_2 s + 1}$$

It follows that, to improve the dynamic performance of the control, it is necessary to act on the gain of the regulator using values that, compatibly with the "noise" of measurement, are as high as possible.

In any case, this solution is advisable in those machines in which the time constants of the equivalent representations of the stator winding are sufficiently low and therefore it is not necessary to "force" the dynamics of the system.

Regulator zero greater than load pole

If the regulator zero is greater than the load pole ($z_{\mathtt{reg}} > p_{\mathtt{load}}$) it means that the system dynamics are being "forced", e.g. the regulator is trying to make the close-loop system assume a dynamic response that tends to exceed that of the load to be regulated.

In this case, it is necessary to limit the regulator gain to appropriate values in order to make sure that the close-loop transfer function poles do not have imaginary solutions, i.e. to make sure that (B.10) has two real roots.

If this is satisfied, then the total transfer function takes a form (B.20) similar to the previous one (B.14).

$$TF = \frac{A_1}{\tau_1 s + 1} + \frac{A_2}{\tau_2 s + 1} \tag{B.20}$$

The schematic representation of this case is shown in Fig. B.5. The presence of two non-imaginary poles is needed to prevent oscillating close-chain responses. Furthermore, the "overcoming" of the load dynamics to be controlled is ensured by the fact that $p_2 \geqslant p_1 \geqslant p_{\mathtt{load}}$.

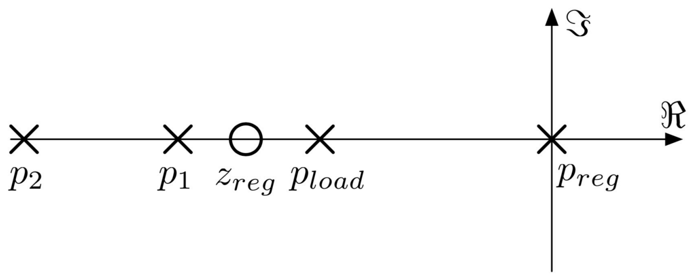


Figure B.5: *Root loci in case of* $z_{reg} > p_{load}$.

Deriving as before the transfer function properties starting from (B.21) and given the case under study (B.22), the transfer function has poles defined by (B.23).

$$p_{1,2} = \begin{cases} \frac{(K_p+r)-\sqrt{(K_p+r)^2-4K_iL}}{2L} \\ \frac{(K_p+r)+\sqrt{(K_p+r)^2-4K_iL}}{2L} \end{cases} \quad \text{with} \quad \frac{K_i}{K_p} > \frac{r}{L} \tag{B.21}$$

$$p_2 > P_1 > z_{reg} > p_{load} \quad \Rightarrow \quad \frac{1}{p_{load}} > \frac{1}{z_{reg}} > \tau_1 > \tau_2 \tag{B.22}$$

$$\begin{cases} A_1 = \frac{\tau_1 - \frac{1}{z_{reg}}}{\tau_1 - \tau_2} < 0 \\ A_2 = \frac{\frac{1}{z_{reg}} - \tau_2}{\tau_1 - \tau_2} > 1 \end{cases} \tag{B.23}$$

It is plain to see how this configuration has a big limitation in the form (B.24).

$$\begin{cases} A_1 \leqslant 0 \\ A_2 \geqslant 1 \\ A_1 + A_2 = 1 \end{cases} \tag{B.24}$$

This means that this solution introduces a sort of over-elongation in the response, like it is shown for a step response in Fig. B.6.

Since it is preferable to avoid this type of behaviour, it is necessary to adopt a precaution to correct it. The solution chosen is to pre-filter the reference using a low-pass filter with a time constant τ equal to the inverse of the controller zero $z_{reg} = \frac{K_i}{K_p}$. In this way the total transfer function becomes (B.25).

$$\begin{aligned} TF &= \frac{1}{\tau \cdot s + 1} \cdot \frac{K_p}{L} \cdot \frac{s + z_{reg}}{(s+p_1)\cdot(s+p_2)} = \\ &= \frac{1}{\frac{1}{z_{reg}}} \cdot \frac{\frac{K_p}{L} z_{reg}\left(\frac{1}{z_{reg}s+1}\right)}{(s+p_1)\cdot(s+p_2)} = \\ &= \frac{\frac{K_p}{L} z_{reg}}{(s+p_1)\cdot(s+p_2)} \end{aligned} \tag{B.25}$$

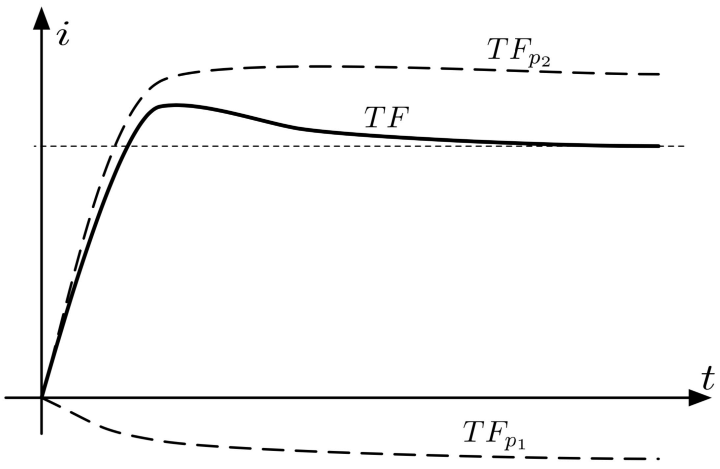


Figure B.6: *Example of over-elongated step response for the case* $z_{reg} > p_{load}$.

This new transfer function, which determines a behaviour without over-elongations, can be described by the usual form (B.20) and from that it can be derived how the two partial transfer function (TF_{p_1} and TF_{p_2}) are characterised by a higher dynamic than the load. It is therefore possible to demonstrate that (B.26) like in the first case (B.17).

$$\begin{cases} A_1, A_2 \in [0, 1] \\ A_1 + A_2 = 1 \end{cases} \tag{B.26}$$

Gain influence

As gain K_p of the regulator increases, the fast pole p_2 tends to increase, too. In digital systems it is necessary to pay attention to the fact that it should not assume values so high as to get close to the sampling frequency of the digital system, which means that:

$$p_2 << \frac{1}{t_c} \tag{B.27}$$

Usually an order of magnitude is sufficient. Extreme settings require simulation and/or experimental testings.

B.3 GENERAL CONSIDERATION ON THE CALIBRATION

The behaviour of the close-loop system depends on the coefficients of the regulator and the values they assume with respect to the load to be regulated.

In the previous section it has been shown how the regulator gain influences the transfer function response and it can be summarised in (B.28).

$$K_p \to \infty \quad \Rightarrow \quad \begin{cases} A_1 \to 0 \\ A_2 \to 1 \\ p_1 \to z_{\mathtt{reg}} = \frac{K_i}{K_p} \\ p_2 \to \infty \end{cases} \tag{B.28}$$

It follows that the regulator K_p gain can be increased to improve the dynamic performance, since the slow pole p_1 tends to have an ever smaller influence ($A_1 \to 0$) while the fast pole assumes ever higher values. Unfortunately, it is not possible to increase the value arbitrarily:

- It is necessary to guarantee the stability of the system which, in reality, is a mixture of a discrete time system (PI regulator) and a continuous time system (load to be regulated). The analytical description in this regard is rather complex and in any case requires approximations. A factor of 10 between the fast pole and the sampling frequency is generally sufficient to guarantee good behaviour.
- High gain values can amplify the measurement noise, in this case the quality of the measurement system, naturally, comes into play.
- High gain values can excite high-order components that have been neglected in the discussion which assumed a simple first-order load.

Particular attention must be paid when the regulator zero is faster than the load pole, (B.29).

$$\frac{K_i}{K_p} = z_{\mathtt{reg}} > p_{\mathtt{load}} = \frac{r}{L} \tag{B.29}$$

This situation means that the close-loop poles can have an imaginary part other than zero. In particular, it can be verified that there is a band of values of the gain K_p which leads to an oscillating response. In practice, the gain must be set either "high" or "low".

The case with a "low" setting may be inconvenient as it would lead to have a rather poor dynamic response of the system. Fig. B.7 schematizes the position of the poles and zeros with $z_{reg} > p_{load}$ and reduced gain.

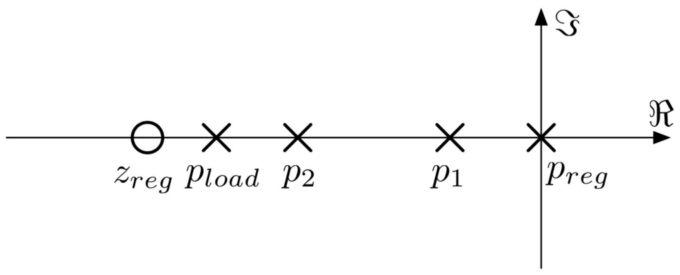


Figure B.7: *Root position scheme for $z_{reg} > p_{load}$ and low K_p gain.*

Fig. B.8, on the other hand, presents the system with a high gain value.

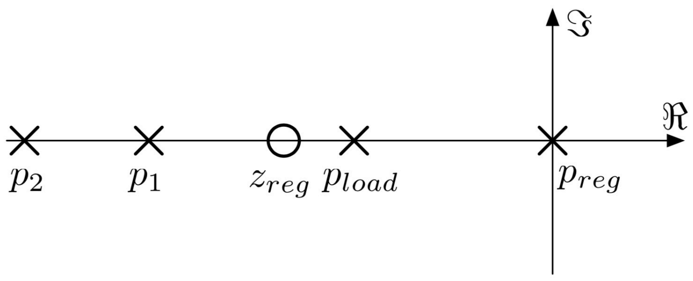


Figure B.8: *Root position scheme for $z_{reg} > p_{load}$ and high K_p gain.*

Finally, Fig. B.9 shows the same system with different K_p gains in he forbidden band, leading to imaginary poles and oscillations.

An approach in parameter calibration can be to set the fast pole p_2 based on the desired dynamic performance, while, of course, keeping in mind the constrain on t_c, and then to set the regulator zero so that it satisfy $z_{reg} < p_{load}$ with a certain safety margin, (B.30).

$$z_{reg} < p_{load} \quad \Rightarrow \quad \frac{K_i}{K_p} < \frac{r}{L} \quad \Rightarrow \quad \frac{K_i}{K_p} = K_{i_{PU}} \cdot \frac{r}{L} \quad \text{with } K_{i_{PU}} \in [0, 1] \tag{B.30}$$

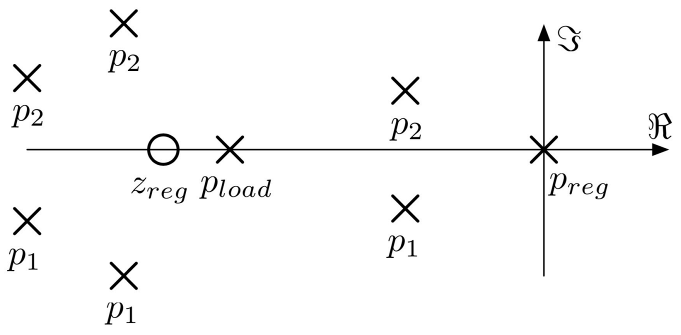


Figure B.9: *Root position scheme for $z_{reg} > p_{load}$ and different K_p gains in the band producing oscillations.*

The following formulas (B.31) will then be used to calculate the coefficients K_p and K_i.

$$\begin{cases} p_2 = \frac{(K_p+r)+\sqrt{(K_p+r)^2-4K_iL}}{2L} \\ \frac{K_i}{K_p} = K_{i_{PU}} \cdot \frac{r}{L} \quad \text{with } K_{i_{PU}} \in [0,1] \end{cases} \tag{B.31}$$

It is therefore possible, with (B.32), to check if the slow pole p_1 has a low influence ($A_1 \to 0$) as it should be for better performance.

$$\begin{cases} \tau_1 = \frac{1}{p_1} = \frac{2L}{(K_p+r)+\sqrt{(K_p+r)^2-4K_iL}} \\ \tau_2 = \frac{1}{p_2} \\ A_1 = \frac{\tau_1 - \frac{1}{z_{reg}}}{\tau_1-\tau_2} < 0 \end{cases} \tag{B.32}$$

If the coefficient A_1 is too high, the following solutions are possibile:

- Increase (if possibile) the fast pole, by increasing the gain;
- Reduce the safety margin of the regulator zero position by bringing it closer to the load pole;
- Force the dynamics of the system by placing $z_{reg} > p_{load}$. However, it is necessary to accurately calibrate the gain in order to avoid an oscillating type of response and to introduce the "pre-filtering" of the reference to reduce over-elongation of the response.

B.4 CALIBRATION FROM LOAD PARAMETERS

The main problem for the selection of the load parameters for the PI voltage regulators is the positioning of the regulator zero in reference to the pole of the load.

Generally, good results can be obtained by setting the zero to be $z_{reg} < p_{load}$, calibrating its position depending on the load parameters. Something like Fig. B.10 should be the desired configuration.

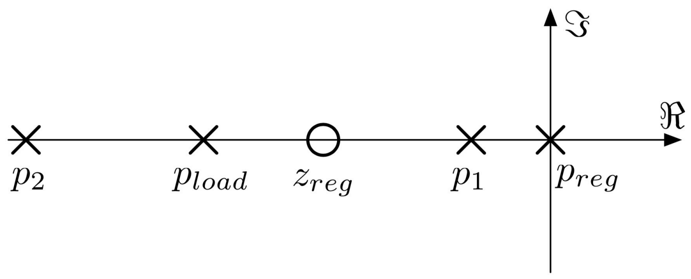


Figure B.10: *Root position scheme for $z_{reg} < p_{load}$, stable configuration.*

However, the load parameters can change during operation due to:

- r for temperature changes;
- L for different magnetic saturation that the motor can assume as a function of the currents.

The position of the load pole can, therefore, vary with respect to the "reference" position, which is the one with which the the regulator zero has been set.

The worst case scenario happens when the load pole p_{load} decreases toward zero, causing also other two poles of the close-loop (p_1 and p_2) to change position. The behaviour is shown in Fig. B.11 for decreasing values of the load pole.

A limit condition is reached when $z_{reg} = p_{load}$, Fig. B.11b. In this case the coefficient A_1 becomes zero and, consequently, the influence of the slow pole is zero.

A further decrease of the load pole, Fig. B.11c, leads to an overshoot response ($A_1 < 0$) and the requirement of a pre-filer on the input to compensate it.

If the pole value keeps decreasing, it would reach the situation with the two close-loop pole coinciding and the creation of imaginary roots, Fig. B.11d, thus producing an oscillating response.

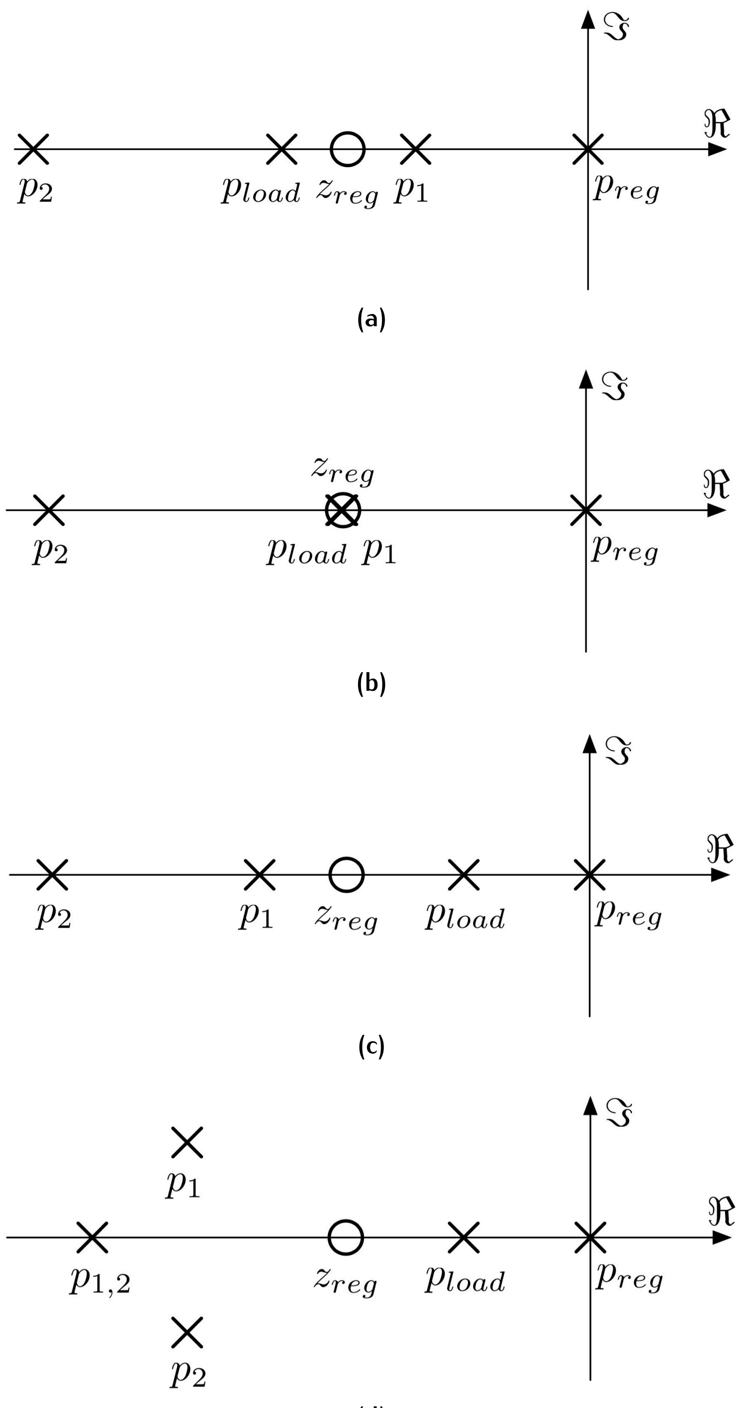


Figure B.11: *Root position changes for decreasing values of load pole.* ***(a)*** $z_{reg} > p_{load}$, *stable.* ***(b)*** $z_{reg} = p_{load}$, *limit condition.* ***(c)*** $z_{reg} < p_{load}$, *over-elongation.* ***(d)*** $z_{reg} = p_{load}$, *oscillating response.*

In the absence of methods to update the load parameters based on the actual motor operating conditions, the only solution for the safe calibration of the regulators is to consider the worst case conditions, keeping in mind that $p_{\mathtt{load}} = \frac{r}{L}$, which are:

- low r, considering the motor cold;
- high L, considering the motor far from magnetic saturation.

Choice of L *value*

Though experimental tests or numerical simulations it is possible to obtain flux mappings Φ according to the motor currents describing the variation of the flux linkage φ. The Φ is therefore the experimental function describing φ, as seen in (B.33) as an example for synchronous motors.

$$\begin{cases} \varphi_d = \Phi_d(i_d, i_q) \\ \varphi_q = \Phi_q(i_d, i_q) \end{cases} \tag{B.33}$$

Fig. B.12 shows examples of this functions for data relating to the test motor.

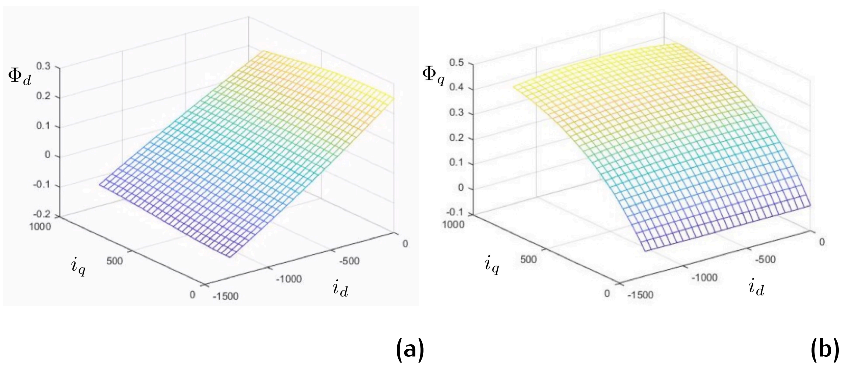


Figure B.12: *Example of flux linkage characteristic.* ***(a)*** *d-axis.* ***(b)*** *q-axis*

As presented in the main body, the solution adopted is the linearization of the flux relation around an operating point "o", here again shown as (B.34).

$$\begin{bmatrix} \varphi_d(t) \\ \varphi_q(t) \end{bmatrix} = \begin{bmatrix} \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_{od}} & \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_{oq}} \\ \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_{od}} & \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_{oq}} \end{bmatrix} \cdot \begin{bmatrix} i_d(t) - i_{od} \\ i_q(t) - i_{oq} \end{bmatrix} + \begin{bmatrix} \Phi_d(i_{od}, i_{oq}) \\ \Phi_q(i_{od}, i_{oq}) \end{bmatrix} \tag{B.34}$$

The numerical solution of the mathematical model of the motor would take place through a continuous updating of the operation point (i_{od}, i_{oq}).

Focusing the attention on the inductance matrix, it can be seen how the new formulation (B.35) relates to the classical one in (B.36). The terms L_{dq} and L_{qd} consider the eventual magnetic cross-coupling between the d and q axis circuits.

$$\begin{aligned}[L(i_{od}, i_{oq})] &= \begin{bmatrix} L_{dd}(i_{od}, i_{oq}) & L_{dq}(i_{od}, i_{oq}) \\ L_{qd}(i_{od}, i_{oq}) & L_{qq}(i_{od}, i_{oq}) \end{bmatrix} = \\ &= \begin{bmatrix} \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_{od}} & \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_{oq}} \\ \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_{od}} & \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_{oq}} \end{bmatrix} \end{aligned} \tag{B.35}$$

$$[L] = \begin{bmatrix} L_d & L_{dq} \\ L_{qd} & L_q \end{bmatrix} \tag{B.36}$$

For the use of the experimental Φ functions starting from punctual measurements, an interpolation is needed that is also differentiable. While in the main body the polynomial interpolation is presented, here is brought as an example a possible alternative obtained using SP-line Third Order Catmull-Rom Method for the differential inductances that constitutes the L matrix, Fig. B.13.

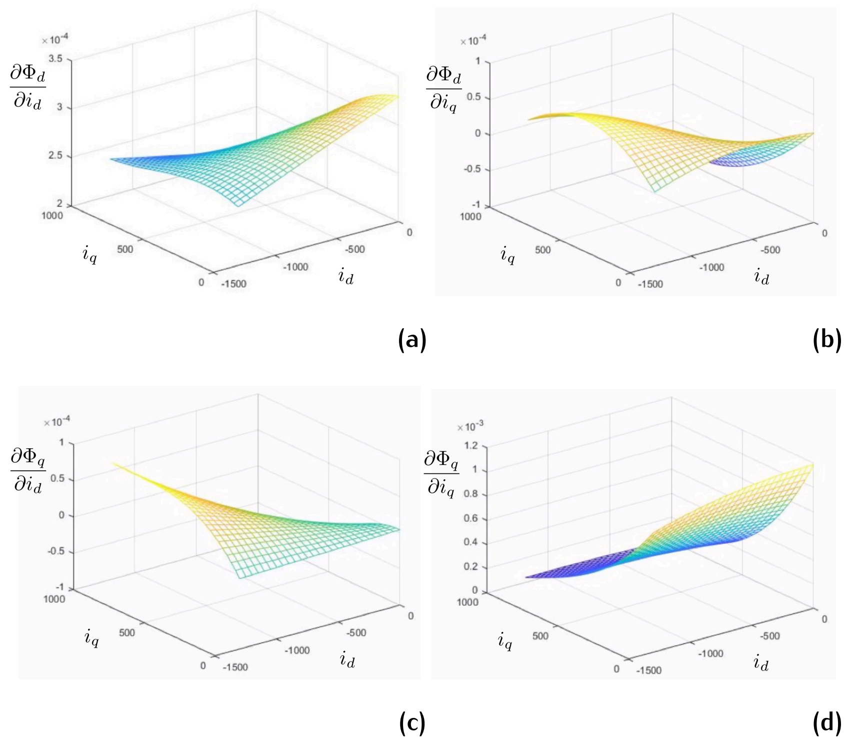


Figure B.13: *Example of interpolated differential inductance functions.* ***(a)*** L_{dd}. ***(b)*** L_{dq}. ***(c)*** L_{qd}. ***(d)*** L_{qq}.

In general, in systems that do not consider magnetic saturation, a sort of equivalent inductance matrix is defined like (B.37).

Moreover, a single working point (i_{od}, i_{oq}) is used and it is chosen considering a particularly significant motor operating condition, for example the nominal operation or one under heavy overload.

$$[L] = \begin{bmatrix} L_d & 0 \\ 0 & L_q \end{bmatrix} = \begin{bmatrix} \frac{\partial \Phi_d(i_{od}, i_{oq})}{\partial i_{od}} & 0 \\ 0 & \frac{\partial \Phi_q(i_{od}, i_{oq})}{\partial i_{oq}} \end{bmatrix} \tag{B.37}$$

In this case it is difficult to consider the magnetic cross-coupling. A possible solution would be to introduce it like in (B.38).

$$[L] = \begin{bmatrix} L_{dd} & L_{dq} \\ L_{qd} & L_{qq} \end{bmatrix} = \begin{bmatrix} \frac{\Phi_d(i_{od}, 0)}{i_{od}} & \frac{\Phi_d(i_{od}, i_{oq}) - \Phi_d(i_{od}, 0)}{i_{oq}} \\ \frac{\Phi_q(i_{od}, i_{oq}) - \Phi_q(0, i_{oq})}{i_{od}} & \frac{\Phi_q(0, i_{oq})}{i_{oq}} \end{bmatrix} \tag{B.38}$$

Fluxes as a conservative field

Some It is often considered that fluxes in function of the currents represent a conservative field, i.e. that there is a potential function $e(i_d, i_q)$ such as (B.39) and (B.40). In this case the matrix of differential inductances would be symmetric.

$$\begin{bmatrix} \varphi_d(i_d, i_q) \\ \varphi_q(i_d, i_q) \end{bmatrix} = \begin{bmatrix} \frac{\partial e(i_d, i_q)}{\partial i_d} \\ \frac{\partial e(i_d, i_q)}{\partial i_q} \end{bmatrix} \tag{B.39}$$

$$\begin{aligned} [L(i_d, i_q)] &= \begin{bmatrix} L_{dd}(i_d, i_q) & L_{dq}(i_d, i_q) \\ L_{qd}(i_d, i_q) & L_{qq}(i_d, i_q) \end{bmatrix} = \\ &= \begin{bmatrix} \frac{\partial \varphi_d(i_d, i_q)}{\partial i_d} & \frac{\partial \varphi_d(i_d, i_q)}{\partial i_q} \\ \frac{\partial \varphi_q(i_d, i_q)}{\partial i_d} & \frac{\partial \varphi_q(i_d, i_q)}{\partial i_q} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 e(i_d, i_q)}{\partial i_d \partial i_d} & \frac{\partial^2 e(i_d, i_q)}{\partial i_d \partial i_q} \\ \frac{\partial^2 e(i_d, i_q)}{\partial i_q \partial i_d} & \frac{\partial^2 e(i_d, i_q)}{\partial i_q \partial i_q} \end{bmatrix} \end{aligned} \tag{B.40}$$

If no interaction between stator and rotor is taken into account as a hypothesis, then it is possible to demonstrate that the fluxes are a conservative field. However, such simplification is very hard to justify in the case of electric motors, thus making this solution not the best one for accurate control. Moreover, data from the actual test motor shows that the fluxes do not have this property.

Even in the event that the fluxes are a conservative field (B.39), the interpolation of the flow maps would still be a problem. In fact, it would have to be done using a methodology that includes the constrain (B.41) to be usable. Such methodology is not discussed, as it has been shown how the conservative field hypothesis is not suitable for the electric machine.

$$\begin{aligned} L_{dq}(i_d,i_q) = \frac{\partial^2 e(i_d,i_q)}{\partial i_d \partial i_q} = \frac{\partial^2 e(i_d,i_q)}{\partial i_q \partial i_d} = L_{qd}(i_d,i_q) \;\Rightarrow \\ \Rightarrow \frac{\partial \varphi_d(i_d,i_q)}{\partial i_q} = \frac{\partial \varphi_q(i_d,i_q)}{\partial i_d} \end{aligned} \tag{B.41}$$

Stability considerations

A simplified study of the stability of a synchronous machine with its rotor stopped is carried out below to demonstrate the stability of the overall system.

The Lyapunov method is used, since, given (B.42), its stability is determined by evaluating the behaviour of (B.43).

$$\begin{cases} v_d = r \cdot i_d + \frac{d\varphi_d}{dt} \\ v_q = r \cdot i_q + \frac{d\varphi_q}{dt} \end{cases} \quad \Rightarrow \quad \begin{bmatrix} \frac{d\varphi_d}{dt} \\ \frac{d\varphi_q}{dt} \end{bmatrix} = -r \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} v_d \\ v_q \end{bmatrix} \tag{B.42}$$

$$\begin{bmatrix} \frac{d\varphi_d}{dt} \\ \frac{d\varphi_q}{dt} \end{bmatrix} = -r \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{B.43}$$

Using the differential approach described in (B.35), it follows that the flux derivatives become (B.44).

$$\begin{bmatrix} \frac{d\varphi_d}{dt} \\ \frac{d\varphi_q}{dt} \end{bmatrix} = \begin{bmatrix} L_{dd}(i_d,i_q) & L_{dq}(i_d,i_q) \\ L_{qd}(i_d,i_q) & L_{qq}(i_d,i_q) \end{bmatrix} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} \tag{B.44}$$

Substituting (B.44) into (B.43) it yields to (B.46).

$$\begin{bmatrix} L_{dd}(i_d,i_q) & L_{dq}(i_d,i_q) \\ L_{qd}(i_d,i_q) & L_{qq}(i_d,i_q) \end{bmatrix} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} = -r \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{B.45}$$

$$\begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} = -r \begin{bmatrix} i_d \\ i_q \end{bmatrix} \cdot \begin{bmatrix} L_{dd}(i_d,i_q) & L_{dq}(i_d,i_q) \\ L_{qd}(i_d,i_q) & L_{qq}(i_d,i_q) \end{bmatrix}^{-1} \tag{B.46}$$

The Lyapunov stability condition requires the Lyapunov function V to be determined negative. (B.47) introduces such function and its derivative (B.48) is used to study its behaviour.

$$V(i_d,i_q) = \frac{1}{2}\left(i_d^2 + i_q^2\right) \quad \Rightarrow \quad V(i_d,i_q) \geqslant 0, \quad \forall\, i_d, i_q \tag{B.47}$$

$$\begin{aligned} \frac{V(i_d,i_q)}{dt} &= \frac{\partial V}{\partial i_d} \cdot \frac{di_d}{dt} + \frac{\partial V}{\partial i_q} \cdot \frac{di_q}{dt} = i_d \cdot \frac{di_d}{dt} + i_q \cdot \frac{di_q}{dt} = \\ &= \begin{bmatrix} i_d & i_q \end{bmatrix} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} \end{aligned} \tag{B.48}$$

In conclusion, combining Lyapunov stability condition (B.48) and the machine equation (B.46), equation (B.49) representing the time derivative of the Lyapunov function can be studied.

$$\begin{aligned}\frac{V(i_d, i_q)}{dt} &= -r \begin{bmatrix} i_d & i_q \end{bmatrix} \cdot \begin{bmatrix} L_{dd}(i_d, i_q) & L_{dq}(i_d, i_q) \\ L_{qd}(i_d, i_q) & L_{qq}(i_d, i_q) \end{bmatrix}^{-1} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} = \\ &= -r \begin{bmatrix} i_d & i_q \end{bmatrix} \cdot [L]^{-1} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix}\end{aligned} \tag{B.49}$$

To verify the stability, the time derivative (B.49) must be defined negative. This involves making the symmetric part of $[L]^{-1}$ to be positive defined, which is equivalent to the definition of positive of its inverse, that is [L].

Going back to the machine under study, it is, therefore, demonstrated how the stability of the motor is achieved with the positive definition of the symmetric part $[L]_{sym}$ of the differential inductance matrix. This condition is shown in (B.50) starting from (B.49), keeping in mind that all the inductances are dependent on (i_d, i_q), but some are not shown for space reasons.

$$\begin{aligned}\begin{bmatrix} i_d & i_q \end{bmatrix} \cdot [L]_{sym} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} &> 0, \quad \forall i_d, i_q \\ \begin{bmatrix} i_d & i_q \end{bmatrix} \begin{bmatrix} L_{dd}(i_d, i_q) & \frac{L_{dq}-L_{qd}}{2} \\ \frac{L_{qd}-L_{dq}}{2} & L_{qq}(i_d, i_q) \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} &> 0, \quad \forall i_d, i_q\end{aligned} \tag{B.50}$$

For a square matrix (B.50) is equivalent to (B.51).

$$\begin{cases} L_{dd}(i_d, i_q) \text{ or } L_{qq}(i_d, i_q) & > 0 \\ \det\left([L]_{sym}\right) & > 0 \end{cases} \tag{B.51}$$

Fig. B.14 gives an example of the trend of $\det\left([L]_{sym}\right)$ given the parameters used in this chapter.

Rotating regulators

Placing the regulators, as it is common, in a rotating reference frame (FOC) and assuming a sufficient compensation of the electromagnetic forces yields to the classic electric circuit equations (B.52) in the two axes.

$$\begin{cases} v_d = r \cdot i_d + \frac{d\varphi_d}{dt} \\ v_q = r \cdot i_q + \frac{d\varphi_q}{dt} \end{cases} \tag{B.52}$$

Given that the the derivative of the fluxes can also be written as (B.44) using the differential approach (B.35), the machine

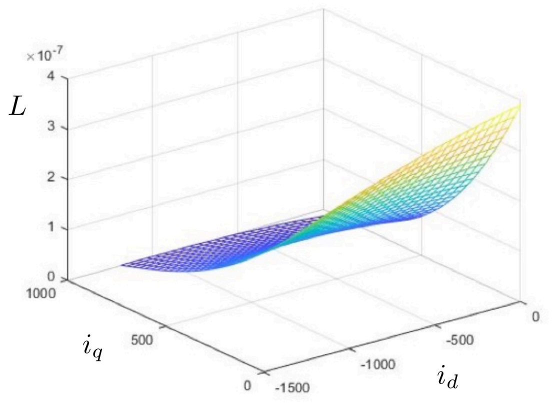


Figure B.14: *Characteristics example for* $\det([L]_{sym})$.

equations become (B.53) and they are what the regulator has to control.

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = r \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} L_{dd}(i_d, i_q) & L_{dq}(i_d, i_q) \\ L_{qd}(i_d, i_q) & L_{qq}(i_d, i_q) \end{bmatrix} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} \tag{B.53}$$

Given this new approach, it can be seen how the system dynamics are influenced by the differential inductances described by (B.35) rather than the quantities defined as $L = \frac{\varphi}{i}$.

Another point very important to notice is the fact that the two magnetic circuits interact with each other due to the magneti cross-coupling. Experimental test have highlighted this feature. For example, following strong variations in the q-axis current such as to strongly vary the saturation degree of the machine, evident perturbations of the d-axis current are registered.

A possible solution would involve the possibility to decompose the inductance matrix [L] as a product of the matrices of eigenvalues [D] (which is diagonal) and of eigenvectors [V]. Moreover, since [L] is symmetric, its decomposition allows to use $[V]^{-1} = [V]^T$, so that (B.54) can be written.

$$[L] = [V][D][V]^{-1} = [V] \cdot [D] \cdot [V]^T \tag{B.54}$$

The decomposition brings to the re-formulation of the machine equations like in (B.55).

$$\begin{aligned} \begin{bmatrix} v_d \\ v_q \end{bmatrix} &= r \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} + [L] \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} \\ \begin{bmatrix} v_d \\ v_q \end{bmatrix} &= r \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} + [V] \cdot [D] \cdot [V]^{-1} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} \\ [V]^{-1} \cdot \begin{bmatrix} v_d \\ v_q \end{bmatrix} &= r \cdot [V]^{-1} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} + [D] \cdot [V]^{-1} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} \end{aligned} \tag{B.55}$$

It could be hypothesised that, in a reference system c rotated through the matrices of the eigenvalues [V], the influence of the magnetic cross-coupling disappears and therefore, if there is a good confidence of the inductances and an interpolation method that does not involve discontinuity, it is convenient to implement the current regulators in this system, (B.56).

$$\begin{bmatrix} x_d \\ x_q \end{bmatrix}^c = [V]^{-1} \cdot \begin{bmatrix} x_d \\ x_q \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} v_d \\ v_q \end{bmatrix}^c = r \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix}^c + \begin{bmatrix} L_d^c & 0 \\ 0 & L_q^c \end{bmatrix} \cdot \begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix}^c \tag{B.56}$$

To derive the voltage references in the stator system it will therefore be necessary to perform a double rotation, the first to give the voltages in the field orientation system (FOC), the second to express them in the stator system.

Currently, no simulations or tests have been conducted to validate such a methodology to compensate the magnetic cross-coupling. So, it is still to be seen if it is a viable solution or not.

ACKNOWLEDGMENTS

Il percorso che mi ha portato a questo importante traguardo è stato lungo e pieno di avvenimenti. Le persone con cui ho condiviso questo periodo e che mi hanno aiutato sono tante e la lista riportata qua sotto è sicuramente incompleta, ma li ringrazio tutti.

Grazie innanzitutto al Prof. Claudio Rossi mio supervisore e riferimento. La fiducia che ha riposto in me durante questi anni e i progetti ambiziosi ma ripaganti che mi ha proposto mi hanno fatto crescere come ricercatore e come persona.

Grazie anche al mio correlatore Alessio Pilati senza cui il progetto di tesi non sarebbe certamente altrettanto completo, la sua esperienza nei controlli motore e del lato software sono stati vitali in molti ambiti di sviluppo del programma. Ringrazio anche FEV Italia per avermi permesso di portare avanti il mio percorso di studi dandomi l'opportunità di collaborare in un ambiente di altissimo livello. Grazie in particolare a Michele e Andrea, miei contatti più frequenti.

Quindi ringrazio tutto il laboratorio LEMAD dove ho trascorso la maggior parte del tempo e che ha riempito i miei momenti di difficoltà e felicità con tanta professionalità e senso dell'umorismo. Grazie agli amici e colleghi Davide, Matteo, Carlo, Gabriele, Michele, Luca V., Oliviero, Johnnidel, Albino, Luca B. e Paolo. Non avervi visto quotidianamente durante il periodo di lockdown è stata decisamente la parte più difficile degli ultimi anni.

Ringrazio le persone che ho conosciuto tramite Onda Solare, che mi hanno portato in giro per il mondo e che mi hanno fatto provare emozioni assolutamente uniche, tra cui la perdita di sonno. Voglio solo sottolineare Mauro, Ruggero, Morena, Luigi, Marco S., Marco F., Hermes, Dondi, Stefano, Francesco, Giangiacomo, Giacinto, Filippo, Andrea, Valentina e tutti gli altri troppo numerosi per elencarli.

Ancora una volta voglio ringraziare ViaC: sapete cosa avete fatto.

Infine ricordo i parenti e la mia famiglia che non ho visto quanto averi voluto durante questo periodo, ma mi sono sempre stati vicini. Sono loro il motivo per cui ho potuto realizzare tutto quello che ho fatto e ringrazio tantissimo per il continuo supporto.

Last but not least, I would like to thank all the people that I met during my travels and in particular my supervisors at FEV company in Aachen where I spent part of my doctorate. Thanks in particular to Anna, Herold, Julian and Ohrun for welcoming me into that new environment and supporting me during the long development phase.

BIBLIOGRAPHY

[1] G. Pellegrino, R. I. Bojoi, P. Guglielmi, "Unified Direct-Flux Vector Control for AC Motor Drives", *IEEE Transactions on Industry Applications*, Vol. 47, No. 5, Sept.-Oct.2011.

[2] C. Rossi, A. Pilati, D. Casadei, "Unified model and Field Oriented Control Algorithm for Three-Phase AC Machines", *15th European Conference on Power Electronics and Applications (EPE)*, Sept. 2-6, 2013, Lille, France.

[3] A. Samat, P. Saedin, A. I. Tajudin, N. Adni, "The Implementation of Field Oriented Control for PMSM Drive Based on TMS320F2808 DSP Controller", *IEEE International Conference on Control System, Computing and Engineering*, Nov. 23-25, 2012, Penang, Malaysia.

[4] X. Guo-kai, S. Peng, Z. Xiu-chun, "Research on Discrete Model Reference Adaptive Control System of Electric Vehicle", *International Joint Conference (SICE-ICASE)*, Oct. 18-21, 2006, Bexco, Korea.

[5] P. Cortes, J. Rodriguez, *Predictive Control of Power Converters and Electric Drives*, Wiley, 2012.

[6] Zhang Jian, Wen Xuhui, Wang Youlong and Li Wenshan, "Modeling and analysis of nonlinear interior permanent magnet synchronous motors considering saturation and cross-magnetization effects," *2016 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*, Busan, 2016, pp. 611-615, doi: 10.1109/ITEC-AP.2016.7513025.

[7] Ying Yan, Jianguo Zhu, Haiwei Lu, Youguang Guo and Shuhong Wang, "Study of A PMSM Model Incorporating Structural and Saturation Saliencies," *2005 International Conference on Power Electronics and Drives Systems*, Kuala Lumpur, 2005, pp. 575-580, doi: 10.1109/PEDS.2005.1619752.

[8] Q. Xie, C. Mu, R. Jia, G. Wu, N. Yang and C. Wu, "Nonlinear dynamic analysis for a permanent magnet synchronous motor model and its application in magnetic flux optimization," *2018 Chinese Control And Decision Conference (CCDC)*, Shenyang, 2018, pp. 1698-1703, doi: 10.1109/CCDC.2018.8407401.

[9] Li Tao, Wang Jiuhe, Tang Yi and Wu Lei, "A new nonlinear control stragety for PMSM," *2010 International Conference On Computer Design and Applications*, Qinhuangdao, 2010, pp. V3-186-V3-189, doi: 10.1109/ICCDA.2010.5541001.

[10] P. V. Medagam and J. Yang, "Robust Speed Control of PMSM with Nonlinear Position Observer," *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*, Harbin, China, 2019, pp. 1-4, doi: 10.1109/ICEMS.2019.8922313.

[11] T. D. Do, H. H. Choi and J. Jung, "θ-D Approximation Technique for Nonlinear Optimal Speed Control Design of Surface-Mounted PMSM Drives," in *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 4, pp. 1822-1831, Aug. 2015, doi: 10.1109/TMECH.2014.2356138.

[12] R. Bhattarai, N. Gurung and S. Kamalasadan, "Discrete real-time state observer based feedback control methodology for Doubly Fed Induction machine," *2016 IEEE International Conference on Power Electronics*, Drives and Energy Systems (PEDES), Trivandrum, 2016, pp. 1-6, doi: 10.1109/PEDES.2016.7914534.

[13] G. M. Schoonhoven and M. N. Uddin, "Wide speed range operation of PMSM drives using nonlinear flux control techniques," *8th International Conference on Electrical and Computer Engineering*, Dhaka, 2014, pp. 603-606, doi: 10.1109/ICECE.2014.7026857.

[14] M. Vilathgamuwa, M. A. Rahman and K. J. Tseng, "Nonlinear control of interior permanent magnet synchronous motor," *Conference Record of the 2000 IEEE Industry Applications Conference. Thirty-Fifth IAS Annual Meeting and World Conference on Industrial Applications of Electrical Energy (Cat. No.00CH37129)*, Rome, Italy, 2000, pp. 1115-1120 vol.2, doi: 10.1109/IAS.2000.881971.

[15] M. N. Uddin and Sang Woo Nam, "Real-time performance of a nonlinear controller based IM drive," *2008 Canadian Conference on Electrical and Computer Engineering*, Niagara Falls, ON, 2008, pp. 000141-000144, doi: 10.1109/CCECE.2008.4564511.

[16] Z. E. Idrissi, H. E. Fadil and F. Giri, "Nonlinear control of salient-pole PMSM for electric vehicles traction," *2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)*, Marrakech, 2018, pp. 231-236, doi: 10.1109/MELCON.2018.8379099.

[17] H. Ge, B. Bilgin and A. Emadi, "Global loss minimization control of PMSM considering cross-coupling and saturation," *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Montreal, QC, 2015, pp. 6139-6144, doi: 10.1109/ECCE.2015.7310520.

[18] A. K. Jebai, F. Malrait, P. Martin and P. Rouchon, "Sensorless position estimation of Permanent-Magnet Synchronous Motors using a nonlinear magnetic saturation model," *2012 XXth International Conference on Electrical Machines*, Marseille, 2012, pp. 2245-2251, doi: 10.1109/ICElMach.2012.6350194.

[19] P. V. Medagam and J. Yang, "Robust Speed Control of PMSM with Nonlinear Position Observer," *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*, Harbin, China, 2019, pp. 1-4, doi: 10.1109/ICEMS.2019.8922313.

[20] A. Accetta, F. Alonge, M. Cirrincione, M. Pucci and A. Sferlazza, "Feedback linearizing control of induction motor considering magnetic saturation effects," *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Montreal, QC, 2015, pp. 4463-4470, doi: 10.1109/ECCE.2015.7310290.

[21] F. Alonge, M. Cirrincione, M. Pucci and A. Sferlazza, "A nonlinear observer for rotor flux estimation considering magnetic saturation effects in induction motor drives," *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Montreal, QC, 2015, pp. 2892-2898, doi: 10.1109/ECCE.2015.7310065.

[22] N. Amiri, S. Ebrahimi, J. Jatskevich and H. W. Dommel, "Saturable and Decoupled Constant-Parameter VBR Model for Six-Phase Synchronous Machines in State-Variable Simulation Programs," in *IEEE Transactions on Energy Conversion*, vol. 34, no. 4, pp. 1868-1880, Dec. 2019, doi: 10.1109/TEC.2019.2907268.

[23] Z. Li and H. Li, "MTPA control of PMSM system considering saturation and cross-coupling," *2012 15th International Conference on Electrical Machines and Systems (ICEMS)*, Sapporo, 2012, pp. 1-5.

[24] Y. Yan, J. Zhu, Y. Guo and H. Lu, "Modeling and Simulation of Direct Torque Controlled PMSM Drive System Incorporating Structural and Saturation Saliencies," emphConference Record of the 2006 IEEE Industry Applications Conference Forty-First IAS Annual Meeting, Tampa, FL, 2006, pp. 76-83, doi: 10.1109/IAS.2006.256487.

[25] J. A. Walker, D. G. Dorrell and C. Cossar, "Flux-linkage calculation in permanent-magnet motors using the frozen permeabilities method," in *IEEE Transactions on Magnetics*, vol. 41, no. 10, pp. 3946-3948, Oct. 2005, doi: 10.1109/TMAG.2005.854973.

[26] A. Sun, J. Li and R. Qu, "Inductance calculation in variable-flux flux-intensifying permanent magnet synchronous machines using improved frozen permeability method," *2015 IEEE International Magnetics Conference (INTERMAG)*, Beijing, 2015, pp. 1-1, doi: 10.1109/INTMAG.2015.7157110.

[27] C. Rossi, A. Pilati, M. Bertoldi "A Novel High Performance Discrete Flux Integrator for Control Algorithms of Fast Rotating AC Machines," *Applied Sciences. 2021*; 11(5):2150. https://doi.org/10.3390/app11052150

[28] Y. B. Salem and L. Sbita, "A Nonlinear State Feedback Control for Induction Motors," *2006 IEEE International Conference on Industrial Technology*, Mumbai, 2006, pp. 2067-2072, doi: 10.1109/ICIT.2006.372519.

[29] H. K. Khalil, E. G. Strangas and S. Jurkovic, "Speed Observer and Reduced Nonlinear Model for Sensorless Control of Induction Motors," in *IEEE Transactions on Control Systems Technology*, vol. 17, no. 2, pp. 327-339, March 2009, doi: 10.1109/TCST.2008.2000977.

[30] L. Abdi, N. Rostami, P. Bagheri and S. Tohidi, "Nonlinear model predictive control of permanent magnet linear synchronous motor," *2017 8th Power Electronics, Drive Systems & Technologies Conference (PEDSTC)*, Mashhad, 2017, pp. 448-453, doi: 10.1109/PEDSTC.2017.7910367.

[31] A. Devanshu, M. Singh and N. Kumar, "An Improved Nonlinear Flux Observer Based Sensorless FOC IM Drive With Adaptive Predictive Current Control," in *IEEE Transactions on Power Electronics*, vol. 35, no. 1, pp. 652-666, Jan. 2020, doi: 10.1109/TPEL.2019.2912265.

[32] Li Tao, Wang Jiuhe, Tang Yi and Wu Lei, "A new nonlinear control strategy for PMSM," *2010 International Conference On Computer Design and Applications*, Qinhuangdao, 2010, pp. V3-186-V3-189, doi: 10.1109/ICCDA.2010.5541001.

[33] D. Semenov, Q. An, L. Sun and G. Mirzaeva, "Nonlinear speed control for five-phase PMSM in Electric Vehicles,"

2016 Australasian Universities Power Engineering Conference (AUPEC), Brisbane, QLD, 2016, pp. 1-5, doi: 10.1109/AUPEC.2016.7749357.

[34] L. Alberti, N. Bianchi and S. Bolognani, "Field oriented control of induction motor: A direct analysis using finite element," *2008 34th Annual Conference of IEEE Industrial Electronics*, Orlando, FL, 2008, pp. 1206-1209, doi: 10.1109/IECON.2008.4758126.

[35] P. J. Coussens, A. P. Van den Bossche and J. A. Melkebeek, "Magnetizing current control strategies for nonlinear indirect field oriented control," *IAS '95. Conference Record of the 1995 IEEE Industry Applications Conference Thirtieth IAS Annual Meeting*, Orlando, FL, USA, 1995, pp. 538-545 vol.1, doi: 10.1109/IAS.1995.530346.

[36] M. Pucci, "State-space space-vector model of the induction motor including magnetic saturation and iron losses," *2017 IEEE Energy Conversion Congress and Exposition (ECCE)*, Cincinnati, OH, 2017, pp. 2404-2411, doi: 10.1109/ECCE.2017.8096464.

[37] A. Accetta, F. Alonge, M. Cirrincione, M. Pucci and A. Sferlazza, "Feedback linearizing control of induction motor considering magnetic saturation effects," *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Montreal, QC, 2015, pp. 4463-4470, doi: 10.1109/ECCE.2015.7310290.

[38] O. Wallscheid, M. Meyer and J. Böcker, "An open-loop operation strategy for induction motors considering iron losses and saturation effects in automotive applications," *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*, Sydney, NSW, 2015, pp. 981-985, doi: 10.1109/PEDS.2015.7203404.

[39] Y. Wang, J. Zhu and Y. Guo, "A Comprehensive Analytical Mathematic Model for Permanent-Magnet Synchronous Machines Incorporating Structural and Saturation Saliencies," in *IEEE Transactions on Magnetics*, vol. 46, no. 12, pp. 4081-4091, Dec. 2010, doi: 10.1109/TMAG.2010.2071392.

[40] F. Mink, N. Kubasiak, B. Ritter and A. Binder, "Parametric model and identification of PMSM considering the influence of magnetic saturation," *2012 13th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, Brasov, 2012, pp. 444-452, doi: 10.1109/OPTIM.2012.6231768.

[41] M. Seilmeier, S. Ebersberger and B. Piepenbreier, "PMSM model for sensorless control considering saturation induced secondary saliencies," *2013 IEEE International Symposium on Sensorless Control for Electrical Drives and Predictive Control of Electrical Drives and Power Electronics (SLED/PRECEDE)*, Munich, 2013, pp. 1-8, doi: 10.1109/SLED-PRECEDE.2013.6684519.

[42] H. Ge, B. Bilgin and A. Emadi, "Global loss minimization control of PMSM considering cross-coupling and saturation," *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Montreal, QC, 2015, pp. 6139-6144, doi: 10.1109/ECCE.2015.7310520.

[43] Zhang Jian, Wen Xuhui, Wang Youlong and Li Wenshan, "Modeling and analysis of nonlinear interior permanent magnet synchronous motors considering saturation and cross-magnetization effects," *2016 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*, Busan, 2016, pp. 611-615, doi: 10.1109/ITEC-AP.2016.7513025.

[44] Yan Ying, Zhu Jianguo, Guo Youguang and Jin Jianxun, "Numerical simulation of a PMSM model considering saturation saliency for initial rotor position estimation," *2008 27th Chinese Control Conference*, Kunming, 2008, pp. 114-118, doi: 10.1109/CHICC.2008.4604955.

[45] Z. Li and H. Li, "MTPA control of PMSM system considering saturation and cross-coupling," *2012 15th International Conference on Electrical Machines and Systems (ICEMS)*, Sapporo, 2012, pp. 1-5.