Speed regulation of a compressor test bed using electrical drivers

Auteur : Hardy, Joris
Promoteur(s) : Arnst, Maarten; Van Cutsem, Thierry
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil mécanicien, à finalité spécialisée en génie mécanique
Année académique : 2017-2018
URI/URL : http://hdl.handle.net/2268.2/4593

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la “Budapest Open Access Initiative” (BOAI, 2002), l’utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s’en servir de données pour un logiciel, ou s’en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l’utilisateur s’engage à respecter les droits moraux de l’auteur, principalement le droit à l’intégrité de l’œuvre et le droit de paternité et ce dans toute utilisation que l’utilisateur entreprend. Ainsi, à titre d’exemple, lorsqu’il reproduira un document par extrait ou dans son intégralité, l’utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l’autorisation préalable et expresse des auteurs ou de leurs ayants droit.
Speed regulation of a compressor test bed using electrical drivers

Academic supervisors:
Dr. Maarten ARNST
Dr. Thierry VAN CUTSEM

Industrial supervisors:
Mr. Stéphane MALJEAN
Dr. Benoît MEYS

President of the academic jury:
Dr. Pierre DUYSINX

Academic year 2017-2018
ABSTRACT

This thesis is the result of an internship in Safran Aero Boosters. This company designs and develops components, equipment and test beds for aeronautic and spatial motors. In this context, Test cells, an activity sector of this company, is investigating compressor test beds. The first reason is a strategic reason. The market wants more compressor facilities and only few modern compressor test facilities exist. Safran Aero Boosters sees thus an opportunity to fulfill the market and to spread their activities. Besides, one need of this company is also to validate new technologies applied to the low-pressure compressor in which it is specialised, while remaining autonomous. The goal is to greatly improve the efficiency of the Research and Technology department by reducing the time elapsed between the conception and the first purchase request of new boosters technology. The behaviour of the compressor is mainly characterised by iso-speed curves. The purpose of this thesis is to simulate an iso-speed regulation of a rotating compressor test bed for whatever the mechanical load exerted by the compressor. It is mainly about the dynamic operating of electric driving machines. The first aim was initially to compare several machines and several electrical drives in order to choose the most appropriated electrical solution for this application. But as will be explained further, this choice is quite difficult to take with objectivity since all machines can suit a priori. Thus, this thesis focuses on one specific machine: the induction (=asynchronous) machine. In the end, this choice is not necessarily the best. Indeed, the most appropriate technology is not easily identifiable. One way to find it, is to study in details all a priori possible choices separately with the same physical characteristics of the mechanical load.

The structure of this thesis is composed of four chapters: an overview of the company and the thematic; the mathematical model of the asynchronous machine including a simple gearbox model; the numerical solvers to obtain the time-varying states of the system; and the proposed controls at the simulation level of practical electric supply components.

In Chapter 1, a brief presentation of the company is given. Then, the main subject is approached and the scope is defined by the company, concerning the mechanic and aerodynamic parts of the compressor. Research on the state of the art leads to several solutions explained briefly. But no specific solution stands out of the others and the induction machine is chosen for simplicity as concerns speed regulation models. And the industrial implications for the Research and Technology department are also introduced.

Chapter 2 and 3 contain tools to model and to solve the dynamics of the system compressor-gearbox-induction machine. Finally, speed regulation technologies are presented in Chapter 4. The first and most ancient technology is the rotor resistance fluctuation. The second one is called «scalar control» regulation based on voltages and frequency fluctuations at the stator of the induction machine. The last one is the «Field-Oriented Control» (FOC) technology. It basically allows to retrieve a speed control very similar to DC(Direct Current) motor and has faster response to perturbations.

To conclude, the scalar regulation offers better performance than the control via rotor resistance command. Indeed, the power losses are smaller and the dynamic is a bit better. However, the field oriented control should improve again the dynamic regulation. A future work could be to quantify the dynamic need of the compressor. That would be performed by studying the thermodynamic states of the air flow that will lead to the critical speed where the compressor should not operate. Then the simulations performed in this thesis would be performed again with a reference speed near to the critical speed.
DEDICATION AND ACKNOWLEDGEMENTS

I would like to thank Dr Thierry Van Cutsem from who I have learned a lot. My sincere gratitude goes to Dr Maarten Arnst for its interest and its availability. I really thank Frederic Neys for his moral support and the good atmosphere of the journey to work. Huge thank goes to Stephane Maljean and Benoît Meys for their encouragement and supervision inside the company. I am grateful for warm welcome of Test Cells staff. I would like to thank also my family for moral support during my internship. And finally, but not least, I would like to acknowledge all professors and administrative staff of the applied science faculty of the University of Liège for these five years of studying.
LIST OF SYMBOLS

\( e, E \)  
feed voltage

\( f \)  
frequency

\( GR \)  
Gear ratio

\( h \)  
time step

\( H \)  
Moment of inertia (in per unit, pu)

\( i \)  
Current

\( j \)  
Complex number

\( J \)  
Moment of inertia

\( k_p, k_i, k_d \)  
PID gains

\( L \)  
Inductance

\( M \)  
Mutual inductance

\( p \)  
number of pair of poles

\( p \square \)  
power related to \( \square \)

\( \mathbf{\Phi} \)  
Park matrix

\( R \)  
Resistance

\( R_e, L_e \)  
resistance, inductance of feed source

\( s \)  
slip

\( S \)  
Nominal apparent power

\( t \)  
time

\( T \)  
Torque

\( v \)  
voltage

\( W \)  
work, energy

\( X \)  
Inductance impedance

\( x, y \)  
vector of states

\( Z \)  
Impedance

\( \alpha, \beta \)  
Phases

\( \eta \)  
Gear efficiency

\( \theta \)  
Angle

\( \psi \)  
Magnetic flux

\( \omega \)  
Frequency or rotation speed
List of subscripts

- \( a, b, c \) related to Phase a, Phase b, Phase c
- \( A, B, C \) related to Phase A, Phase B, Phase C
- \( B \) related to basis value
- \( \text{comp} \) related to compressor
- \( d \) related to axis \( d \) of Park plan
- \( e \) related to electric values
- \( i \) related to initial value
- \( J \) related to Joule losses
- \( m \) related to mechanic values
- \( M \) related to magnetic values
- \( \text{mo} \) related to mechanical valve opening
- \( o \) related to axis \( o \) of Park plan
- \( P \) related to values in Park axes
- \( \text{pu} \) related to per unit values
- \( q \) related to axis \( q \) of Park plan
- \( r \) related to rotor
- \( s \) related to stator
- \( \text{sr} \) related to a stator-rotor coupling
- \( T \) related to three-phase values
- \( \text{Te} \) related to electric torque
- \( x \) related to axis \( x \)
- \( y \) related to axis \( y \)
# Table of Contents

List of Tables xiii

List of Figures xv

1 Thesis overview 1

1.1 Safran Aero Boosters ................................................. 1
1.1.1 Safran ............................................................ 1
1.1.2 Safran Aero Boosters ........................................... 2
1.1.3 Test cells ......................................................... 2
1.2 Thesis problematic .................................................. 2
1.2.1 Motor test bed .................................................. 3
1.2.2 Scope of the subject and input data ........................... 4
1.3 State of the art ...................................................... 5
1.3.1 Compressor test bed ............................................. 5
1.3.2 Synchronous machine ......................................... 6
1.3.3 Asynchronous machine ....................................... 7
1.3.4 Doubly fed asynchronous machine ............................ 8
1.3.5 Choice of an electrical drive .................................. 9
1.4 Main stakes ......................................................... 10

2 Mathematical Model of the Asynchronous Machine 11

2.1 General model ....................................................... 11
2.1.1 Relation voltage-current at each winding ...................... 12
2.1.2 Flux flowing in each winding .................................. 13
2.1.3 Voltage sources for the stator ................................ 13
2.1.4 Mechanical motion equation .................................. 14
2.2 Park transformation ................................................ 14
2.3 Model expressed in Park axes .................................... 16
2.3.1 Relation voltage-current at each winding ...................... 17
2.3.2 Flux flowing in each winding .................................. 17
2.3.3 Power balance and electric torque ............................. 18
2.4 Introduction of per unit system .................................. 20
2.4.1 Rotor motion .................................................... 21
2.4.2 Electromagnetic torque ....................................... 21
2.4.3 Flux equations and flux derivatives .......................... 22
2.4.4 Grid/supply at the stator ..................................... 22
# TABLE OF CONTENTS

2.4.5 Final system and standard parameter values of induction machine 23  
2.5 Phasor approximation 24  
2.5.1 Source voltage for a single phase 24  
2.5.2 Park transform of the three-phase balanced stator 25  
2.5.3 Stator winding equations 26  
2.6 Final system using Park and phasor approximation 27  
2.7 Induction machine at steady state 27  
2.8 Work performed in Chapter 2 30  

3 Numerical resolution 31  
3.1 Discretisation algorithm 31  
3.1.1 Initial conditions 32  
3.1.2 Method to iterate in time 33  
3.2 Simulink 35  
3.3 Simulink - script: comparison 38  
3.4 Matlab - Scilab: comparison 38  
3.5 Simulation 39  
3.5.1 Initial condition simulation 39  
3.5.2 Scale torque $T_{mo}(t)$ simulation 41  
3.5.3 Scale rotor resistance $R_r(t)$ simulation 44  
3.6 Work performed in Chapter 3 46  

4 Regulation and control methods 47  
4.1 Rotor resistance control 47  
4.1.1 Theoretical part: control algorithm 47  
4.1.2 Tuning of the PID controller 50  
4.1.3 Start-up 53  
4.1.4 Feedforward 54  
4.1.5 Coulomb friction 56  
4.1.6 Grid supply noise 58  
4.1.7 Induction machine parameter 59  
4.1.8 Compressor mechanical load parameter 61  
4.1.9 Comparison with Matlab script 62  
4.1.10 Practical implementation 63  
4.2 Scalar control 64  
4.2.1 Theoretical part: control algorithm 64  
4.2.2 Brief simulation and script comparison 66  
4.2.3 Practical implementation 66  
4.3 Field oriented control 67  
4.3.1 Electric torque re-expressed 67  
4.3.2 Introduction of the magnetising current 68  
4.3.3 Park angle computation 69  
4.3.4 Block diagram of the regulation 70  
4.4 Comparison 70  
4.5 Work performed in Chapter 4 71
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Conclusion</td>
<td>73</td>
</tr>
<tr>
<td>A</td>
<td>Appendix A: initial states computation function</td>
<td>77</td>
</tr>
<tr>
<td>B</td>
<td>Appendix B: Newton solver function</td>
<td>81</td>
</tr>
<tr>
<td>C</td>
<td>Appendix C: useful scripts</td>
<td>85</td>
</tr>
<tr>
<td>D</td>
<td>Appendix D: useful functions</td>
<td>87</td>
</tr>
<tr>
<td>E</td>
<td>Appendix E: first simulations</td>
<td>91</td>
</tr>
<tr>
<td>F</td>
<td>Appendix F: speed regulation via rotor resistance</td>
<td>105</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Comparative table of all introduced electrical machines</td>
</tr>
<tr>
<td>2.1</td>
<td>Per unit system used for the asynchronous machine</td>
</tr>
<tr>
<td>2.2</td>
<td>Per unit values on the machine base</td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison between Simulink and Matlab scripts</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparison between Matlab and Scilab</td>
</tr>
<tr>
<td>4.1</td>
<td>Default parameters for rotor resistance command during the speed regulation of the induction machine</td>
</tr>
<tr>
<td>4.2</td>
<td>Induction parameters studied</td>
</tr>
<tr>
<td>4.3</td>
<td>Table of a range of state values (at $\pm 1e^{-5}$) at 200s corresponding to the variation of the designed parameter in the range of Table 4.2</td>
</tr>
<tr>
<td>4.4</td>
<td>Default parameters for rotor resistance command during the speed regulation of the induction machine</td>
</tr>
</tbody>
</table>
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Geographic locations of the Safran group</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Architecture of the global system</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Specific areas of a test bed</td>
<td>3</td>
</tr>
<tr>
<td>1.5 Torque absorbed by the compressor depending on thermodynamic control parameters and the rotation speed</td>
<td>5</td>
</tr>
<tr>
<td>1.6 Schema of a single pair of poles, salient pole synchronous machine</td>
<td>6</td>
</tr>
<tr>
<td>1.7 Salient pole synchronous machine with 2 pairs of poles</td>
<td>6</td>
</tr>
<tr>
<td>1.8 Salient pole with 5 pairs of poles</td>
<td>6</td>
</tr>
<tr>
<td>1.9 Round rotor</td>
<td>6</td>
</tr>
<tr>
<td>1.10 Induction machine, squirrel cage principle</td>
<td>7</td>
</tr>
<tr>
<td>1.11 Wound rotor of induction machine</td>
<td>8</td>
</tr>
<tr>
<td>1.12 Application of a doubly fed induction machine</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Model of the system and dynamic equations linked to different machine parts</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Stator and rotor windings of asynchronous machine</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Inductance representation of stator and rotor windings of asynchronous machine</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Inductance representation of stator and rotor windings for the asynchronous machine</td>
<td>12</td>
</tr>
<tr>
<td>2.5 Thevenin model of power supply to the asynchronous machine</td>
<td>14</td>
</tr>
<tr>
<td>2.6 Asynchronous machine with Park axes</td>
<td>15</td>
</tr>
<tr>
<td>2.7 Inductance representation of an asynchronous machine in the Park axes</td>
<td>15</td>
</tr>
<tr>
<td>2.8 Time-varying phasor in the complex plane</td>
<td>24</td>
</tr>
<tr>
<td>2.9 Summary of all axes: winding axes, xy axes, dq axes</td>
<td>26</td>
</tr>
<tr>
<td>2.10 Influence of rotor resistance ( R_r ), stator frequency ( \omega_s ), and voltage ( V = \sqrt{E_x^2 + E_y^2} ) (all parameters in pu) on the electrical torque produced by the induction machine</td>
<td>28</td>
</tr>
<tr>
<td>2.11 Torque-speed curve of double cage induction machine</td>
<td>29</td>
</tr>
<tr>
<td>2.12 Phasor circuit of an induction machine at steady-state</td>
<td>30</td>
</tr>
<tr>
<td>3.1 Matlab function that computes ( F_I ) for a given state ( X_I )</td>
<td>33</td>
</tr>
<tr>
<td>3.2 Algorithm used to find initial steady-state conditions</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Description of the algorithm used to solve the differential-algebraic equations</td>
<td>34</td>
</tr>
<tr>
<td>3.4 Simulink inside the grid equation block</td>
<td>35</td>
</tr>
<tr>
<td>3.5 Simulink user-defined function block</td>
<td>35</td>
</tr>
<tr>
<td>3.6 Integrator block with ( x_0 ), the initial condition</td>
<td>36</td>
</tr>
<tr>
<td>3.7 Scope block to display the state during time</td>
<td>36</td>
</tr>
</tbody>
</table>
4.21 Compressor speed and torque for $e_{x,\text{var}} = 0.005^2$ ................................................. 58
4.22 Compressor speed and torque for $e_{x,\text{var}} = 0.01^2$ .................................................. 59
4.23 Compressor speed and torque for $e_{x,\text{var}} = 0.02^2$ .................................................. 59
4.24 Stator current for $e_{x,\text{var}} = 0.001^2$ ................................................................. 59
4.25 Stator current for $e_{x,\text{var}} = 0.005^2$ ................................................................. 59
4.26 Stator current for $e_{x,\text{var}} = 0.01^2$ ................................................................. 59
4.27 Stator current for $e_{x,\text{var}} = 0.02^2$ ................................................................. 59
4.28 Compressor speed around the $T_{mo}$ perturbation, $R_s$ variation ................................. 60
4.29 Compressor speed around the $T_{mo}$ perturbation, $R_e$ variation ................................. 60
4.30 Compressor speed around the $T_{mo}$ perturbation, $L_e$ variation ................................. 60
4.31 Compressor speed around the $T_{mo}$ perturbation, $L_{rr}$ variation ................................. 60
4.32 Compressor speed around the $T_{mo}$ perturbation, $L_{sr}$ variation ................................. 61
4.33 Compressor speed around the $T_{mo}$ perturbation, $L_{ss}$ variation ................................. 61
4.34 Influence of $a$ on compressor speed and compressor torque ........................................... 62
4.35 Influence of the moment of inertia of the compressor .................................................... 62
4.36 Comparison between Matlab script using the discretisation algorithm explained in Chapter 3 and the Simulink software ................................................................. 63
4.37 Speed-torque curve evolution when varying the frequency keeping a constant $\frac{e_s}{\omega_s}$ ...... 64
4.38 Speed-torque curve evolution when varying the frequency keeping a constant $\frac{e_r}{\omega_s}$ ...... 64
4.39 Block diagram of the scalar control system ..................................................................... 65
4.40 Content of the «Control System» block ......................................................................... 65
4.41 State evolution of the induction machine during the scalar regulation, comparison of Matlab script results and those given by Simulink ......................................................... 66
4.42 Slides from «Introduction to signals and systems» course of G. Drion [19] ...................... 69
4.43 Block diagram of the field oriented control of the asynchronous machine ....................... 70
4.44 Comparison of performances of the two methods: rotor resistance command or scalar control . 70
This chapter is an introduction to the thesis: the context and the research performed before the implementation of one solution. It contains a section dedicated to Safran Aero Boosters, their activity sectors and their position in the Safran group and more specifically, the products supply by Test Cells department. Next section talks about the thesis subject. It introduces the main stakes, the input data (and starting assumptions) and the scope of the thesis. Then, a state of the art is done on two plans: compressor test beds and electric driving machines. After all, the last section will expose the main implications for the company, the reasons that push them to investigate compressor test bed.

1.1 Safran Aero Boosters

1.1.1 Safran

Safran is a highly technological international group acting in three sectors: Aviation, Space, Defence. Recently, they have bought the group Zodiac Aerospace.

![Figure 1.1: Geographic locations of the Safran group](image)

As seen in Figure 1.1, Safran is present on more than 340 sites in 60 countries and employs more than 91 000 people. It is the second world leader in aeronautic equipment and the third leader in the aerospace sector.
1.1.2 Safran Aero Boosters

Safran Aero Boosters is a subsidiary of the Safran group. It represents an important technological partner for the motorists. It designs and develops components, equipment and test beds for aeronautic and spatial motors.

This company was born in the post-war (1940-1945) period at the National Factory of Herstal and initially worked on jet engines for the Belgian Air Component. In 1987, the company anciently called «FN motor» takes its independence from the national factory and enters in civil applications. «SNECMA» buys the company in 1989. The company changes his name to «Techspace Aero» in 1992 and is still nowadays well-known with that name. Quite recently, in 2005, the groups SNECMA and SAGEM merge together to form a new group called Safran. After a couple of years in 2016, all companies of the Safran group change of name and incorporate Safran in their name. The company is renamed Safran Aero Boosters. And in 2018, Safran spreads its expertise by buying the Zodiac group. Nowadays, it would not result in a change of name for the Safran companies. Zodiac companies would incorporate Safran in their names.

The activity sectors of the company are divided into four categories: the boosters, the oil systems, the test cells and the space equipment. Their main clients are GE (General Electric) Aviation, Safran Aircraft Engine and CFMI (a joint venture between Safran Aircraft Engine and GE). The booster activity consists of three products: the low pressure compressors (core product of the company), the front bearing supports and the fan disks. The oil systems offer its maintenance services and their products are the lubrication units, the oil tanks and the heat exchangers. And «Test cells» are introduced in the next subsection.

1.1.3 Test cells

Test cells is a department of the Safran Aero Boosters company. It is a world leader in engineering of ground motor test facilities. They developed expertise since more than 60 years. Their stakes are to ensure the good behaviour of the aircraft engine. The main issue is that the engine will not behave the same way on the ground as at high altitude. It is why, they add special equipment to reach for instance the same pressure condition as at high altitude. However, they did not succeed in reaching all similar conditions as at high altitude. Indeed, they did not achieve to manage the same temperature condition. There are still progress to achieve more accurately the surrounding conditions of the engine. They are specialised in key-hand project and offer a whole management of the project by realising the design, manufacture, installation and support of its products. Besides, they offer three kinds of products: Cyres(the acquisition and control systems), EoLines(specific adding equipment such as hoods, air intakes and exhaust nozzles), thrust measurement systems (thrust stands) and motor adapters. They currently are working in close collaboration with Safran Aircraft Engine on two test facilities for the LEAP aircraft engines.

1.2 Thesis problematic

The goal of this thesis is to simulate a part of a compressor test bed. The compressor has to be tested at different rotating speed to check its aerodynamic and mechanic characteristics. In order to test it, a driver is used to rotate the compressor at typical speeds of aircraft engines. The company Safran Aero Boosters wishes to investigate an electrical solution to drive the compressor in the test bed. The subject of this thesis is thus to propose a speed regulation of this compressor using electrical machine.
1.2. THESIS PROBLEMATIC

In Figure 1.2, the whole driving circuit is presented. The compressor with the air intake and exhaust are assumed well-known, i.e., torque load and speed required. In fact, no practical data are supplied as concern the torque load behaviour when inlet valve opening and outlet valve opening are modified. A particular behaviour case is chosen such as the torque would be given as a second order function of the speed with coefficients depending on valve opening/closing. A gearbox is then placed to separate the low speed side (electric machine: 1800 rpm) and the high-speed side (compressor: ≈ 10000 rpm). The model of the gearbox is not the main subject of this thesis and is just model by a simple gear ratio with an equivalent moment of inertia seen from the electrical machine. The main purpose of this thesis is the electric machine model and the control model with converters. The aim is to simulate the dynamic link between the mechanical side and the electrical side.

1.2.1 Motor test bed

At first, a brief explanation about the motor test bed is given to have the basic knowledge of test facilities. Indeed, compressor test bench will necessarily have lots of similarities with a motor test bed.

A test bed is composed of four spaces. The «Test Area» is where the motor is attached and tested. The «Control Area» is a room next to the test area. A glass allows to observe the test and; all sensors and commands are controlled with a software CYRES developed by the DACS (Data Acquisition and Control System) staff of Test Cells department. The «Preparation Area» is the room where the motor is assembled. And at last, the motor in the test area is supplied in kerosene by a hydraulic circuit linked to the «Fuel System» which stocks fuel.
CHAPTER 1. THESIS OVERVIEW

Figure 1.4: Side view of the test area: 1.Egg crate; 2.Acoustic panels; 3.Turning vanes; 4.A grid; 5.Splitters; 6.Augmenter; 7.Blast basket

In Figure 1.4, a test bed is represented with, on the left, the inlet and the exhaust on the right. The egg crate is a grid to avoid objects to be inhaled by the engine. The acoustic panels are placed at the entrance and the exhaust of the air to attenuate the noise. The turning vanes allow to reorient the flow. The grid blocks smaller object unstopped by the egg crate and helps to even out the intake flow. The splitters only serve this purpose of uniforming the flow in order to avoid chaos’s profile and pumping phenomenon on the engine.

As a matter of the compressor test bed, this part of the aircraft engine is tested alone in rotation. But now, the kerosene is no more used since the aircraft engine is missing. Another machine must be used to drive the compressor. Safran Aero Boosters wants to investigate an electric solution.

1.2.2 Scope of the subject and input data

The goal is to model the dynamics of the compressor driven by an electric machine using a gearbox with efficiency \( \eta \), assumed ideal \( \eta = 1 \) but adjustable for a further concrete application. The following data are given for the compressor:

\[
J_{comp} = 20 \text{ kgm}^2; \quad (1.1)
\]
\[
T_{comp} \in [0; 7466] \text{ Nm}; \quad (1.2)
\]
\[
GR = 1/5.67; \quad (1.3)
\]
\[
\omega_{comp} \in [925; 10206] \text{ rpm}; \quad (1.4)
\]

where \( J_{comp} \), the moment of inertia of the compressor; \( T_{comp} \), the operating range of torque load with a max acceptable torque of 9398 Nm; \( GR = \frac{\omega_{motor}}{\omega_{comp}} \) the gear ratio. As concern the electrical machine, a 10 MW motor is assumed to supply useful enough power to drive the compressor. The rotary stiffness of all mechanical components is not modelled. Yet, the damping is present in the mechanical torque exerted by the compressor. This torque is assumed as a parabolic dependent on the rotation speed:

\[
T_{comp} = \alpha \left( A^* \omega_{comp}^2 + B^* \omega_{comp} + C^* \right); \quad (1.5)
\]

with \( A^*, B^*, C^* \) constant values and \( \alpha(t) \in [0; 1] \). \( \alpha(t) \) is defined as the resulting parameter of a change in the thermodynamic circuit including the compressor (such as a change of valve opening percentage). An example of typical torque according to that assumption is presented in Figure 1.5. All information presented here is dependent on the thermodynamic circuit and the compressor characteristic which are currently
unknown by the company. They are based on feeling and experience of Safran staff. Finding more accurate characteristics is not the purpose of this thesis. The change of those parameters does a priori not impact the modelling of the driving machine. But it will result in a dynamic change and electric control parameters change.

![Compressor Speed (kRPM)
Compressor Torque (kNm)
$\alpha$: 0.4
$\alpha$: 0.55
$\alpha$: 0.7
$\alpha$: 0.85
$\alpha$: 1](image)

Figure 1.5: Torque absorbed by the compressor depending on thermodynamic control parameters and the rotation speed

The goal of the test bed is to change some thermodynamics parameters, $\alpha$, by opening or closing valves and to observe the thermodynamics effects at the same speed. The electric machine must thus be controlled to reach and keep a given speed.

As a matter of scope, this thesis is limited to some research on the state of the art, the modelling of the driving machine, the regulation and an introduction of practical implemented solutions.

### 1.3 State of the art

The state of the art in Safran Aero Boosters is quite new. No compressor test rig was ever developed by Safran Aero Boosters. This section is separated into three main parts. At first, some research on the subject of compressor test facilities is done. Then, several kinds of electric machines are studied briefly. At the end, a choice is made concerning the electric driving machine.

#### 1.3.1 Compressor test bed

Compressor test facilities are not really new. Many compressors test rigs already exist. Such installations are or were for instance, present in: the United Kingdom in Pyestock (RAE: Royal Aerospace Establishment), Germany in Koln (DLR: Deutsche Forschungsanstalt fuer Luft- und Raumfahrt), Switzerland in the LTT («Laboratoire de Thermique appliquée et de Turbomachine» i.e, thermal and turbo-machine laboratory) of the EPFL («Ecole Polytechnique Fédérale de Lausanne»), the United States at the University of Notre Dame. However only few information is given for free and literature on the subject is quite rare. It is why, Safran Aero Boosters has purchased a study of Notre-Dame University to test one Safran compressor on the compressor test rigs of Notre-Dame. They paid to access to the whole measurements during the test. Their goal is then to compare the measurements to a model established by Safran Aero Boosters that simulate the compressor test rigs. Currently three companies have a modern functional compressor facilities: AneCom, Pratt & Whitney and a Chinese company.
CHAPTER 1. THESIS OVERVIEW

1.3.2 Synchronous machine

The synchronous machine is an electromagnetic device that allows to rotate a rotor thanks to an electric three-phase feed at the stator.

![Figure 1.6: Schema of a single pair of poles, salient pole synchronous machine](image1)

![Figure 1.7: Salient pole synchronous machine with 2 pairs of poles](image2)

The rotor is either composed of permanent magnets or electromagnetic windings fed by DC (Direct Current) power supply. The stator is generally fed by a three-phase power supply. It creates a rotating magnetic field with a speed equals to the frequency of the three-phase supply. This field acts like rotating magnets which attracts the rotor. The rotor speed is the same as the stator magnetic field speed:

\[ \omega_r = \omega_s. \] (1.6)

Often, the synchronous machine can have more than one pair of poles. It has an impact on the rotor speed which diminishes:

\[ \omega_r = \frac{\omega_s}{p}; \] (1.7)

with \( p \), the number of pairs of poles.

There exist mainly two kinds of rotor configuration for the synchronous machine: the salient pole and the round rotor. The salient pole consists of several windings mounted on a magnetic wheel where each winding stands for one pole. This rotor type has as particularities: a small length for a higher diameter and at least 2 pairs of poles or more (typically between 2 and 30 pairs) which involves rotor operating speeds between 100 RPM and 1500 RPM (with a 50 Hz feed assumed).

![Figure 1.8: Salient pole with 5 pairs of poles](image3)

![Figure 1.9: Round rotor](image4)
1.3. STATE OF THE ART

The round rotor consists of a cylindrical core with slots where windings are placed along the radial axes of the core as shown in Figure 1.9. This type of rotor is designed to rotate at a higher speed (1500 to 3000 RPM assuming a 50 Hz feed) with fewer pairs of poles (1 or 2 pairs). The axial length is greater than the salient pole and the diameter is lower.

The main advantage of the synchronous machine is the robustness in speed. Indeed, the speed is fixed independently from the mechanical load. The control is performed by the machine itself. No controller is needed to apply for a fixed speed. The disadvantages are the slip rings presence to create electromagnetic magnets in the rotor (their wear asks maintenance), the complexity of the machine start-up and the possible stall when the mechanical load is too high. Since the power needed is 10 MW, the use of permanent magnets in the rotor is no more possible as the power of that configuration is limited to a few kilowatts. The start-up can be done in several ways:

- the stator is not fed and another motor is used to reach the synchronous speed. At that speed, the motor is cut off and the stator is fed to take over;
- the rotor is short-circuited and acts as the squirrel cage of an asynchronous machine until the rotor speed approaches synchronism speed. Finally the rotor is again fed to create the electromagnetic magnets;
- the stator is fed by an electronic converter that is controlled to increase the frequency smoothly from 0 to 50 Hz.

1.3.3 Asynchronous machine

The asynchronous machine is also an electromagnetic device that rotates a rotor thanks to a three-phase supply. The main difference with the synchronous machine is that the rotor is no more fed, but short-circuited.

In Figure 1.10, the squirrel cage in the rotor is shown. This cage is submitted to a rotating magnetic field created by the stator. The cage that is composed of several conductor loops, sees a time-varying magnetic fields. By the Maxwell-Faraday law, it induces currents that create a field to oppose the variation of the field that gave birth to him. Then, an electromagnetic force is exerted since the conductor cage traversed by a current is submitted to a magnetic field: \( \vec{F} = I \vec{l} \wedge \vec{B} \); \( F \): the electromagnetic force; \( |\vec{l}| \): the current.
amplitude; $\vec{I}/|\vec{I}|$: the unit vector oriented along the current path; $\vec{B}$: the magnetic field. This machine is called asynchronous in the sense that the rotor cannot turn at the same speed as the input frequency at the stator. Indeed, if it turns at this speed, the rotor does not see a time-varying magnetic field, no current is induced and no motor torque is applied to the rotor. But what does not change between synchronous and asynchronous machine is that the magnetic fields of the stator and the rotor are still aligned.

At this stage, a parameter $s$ is defined such that the rotor current frequency is proportional to the stator frequency: $s\omega_s$. To have a feeling about what it implies physically, one can consider two extreme cases. First, the rotor is forced to rotate at the synchronous speed. It implies that no magnetic field variation is seen by the rotor and the induced current frequency is null ($s = 0$). The second case is when the rotor is blocked. The rotor sees then a time-varying magnetic field created by the stator. That induces a rotor current at exactly the same frequency as the stator supply ($s = 1$).

Seen from the stator, the rotor magnetic field frequency is given by the rotor current frequency plus the rotor speed and must be aligned to the stator magnetic field: $s\omega_s + \omega_r = \omega_s$. In power dynamics, $s$ is well-known as the slip:

$$s = \frac{\omega_s - \omega_r}{\omega_s}$$

with $s \rightarrow 0$: near the synchronous speed, $s = 1$: at the start-up.

In practice, like the synchronous machine, the induction machine can have several pairs of poles that are located at the stator. Two kinds of rotors can be implemented: the squirrel cage or the wound rotor. The squirrel cage is represented in Figure 1.10. The wound rotor (Figure 1.11) has the advantage on the squirrel cage to allow access to the rotor windings thanks to slip rings. Yet, it also comes with the drawbacks of slip rings: wear and warming.

![Figure 1.11: Wound rotor of induction machine](image)

The main advantage of the induction machine is its robustness: nearly no maintenance is required considering the squirrel cage. It is also specifically designed for dynamic application and speed regulation. Yet its efficiency is lower than the synchronous machine. It can start without starting devices but the rotor currents have a huge peak at the start-up.

### 1.3.4 Doubly fed asynchronous machine

The doubly fed asynchronous machine is an asynchronous machine with a wound rotor. The rotor is henceforth fed by a three-phase control supply and the stator is simply connected to the grid. It allows the rotor speed to vary in a relative small range 20-25%. Intuitively, it consists of forcing one frequency at the rotor. It allows to control the slip via this frequency. And the speed is regulated by controlling the slip.
In Figure 1.12, a typical application of a doubly fed induction machine is shown. It is to control the speed of a wind turbine. In practice, the wind turbine has an optimal speed where it produces a maximum of power to the grid. The wind allows the turbine to rotate and the rotor supply via AC/DC, DC/AC converters controls the speed to have the highest efficiency. In practice, the wind turbine rotates at low speed compared to the induction machine (3000 RPM - 1500 RPM depending on the number of pairs of poles). It is why, a gearbox is present and bound to the rotor. Then, the stator is directly fed by the grid. And finally, the slip rings of the wound rotor is fed and controlled by electronic power devices (converters) linked to the grid.

The main advantage of the doubly fed asynchronous machine is that it reduces the inverter and filter costs compared to the induction machine. Compared to the synchronous machine, it has a better efficiency. But slip rings are present which implies overheat, wear and maintenance. Furthermore, the speed range is limited conversely to the induction and the synchronous machine.

### 1.3.5 Choice of an electrical drive

As seen in the previous subsection, the different technologies have advantages and disadvantages. It is difficult to make an optimal choice at first sight. Here is a recapitulate table of advantages and drawbacks of all presented machines:

<table>
<thead>
<tr>
<th></th>
<th>Synchronous machine</th>
<th>Induction machine</th>
<th>Doubly fed induction machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed range</td>
<td>large</td>
<td>large</td>
<td>small</td>
</tr>
<tr>
<td>Efficiency</td>
<td>high</td>
<td>medium</td>
<td>highest</td>
</tr>
<tr>
<td>Slip rings</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Power electronic Costs</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Complexity of the model and the regulation</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Speed - torque load dependence</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1.1: Comparative table of all introduced electrical machines
As shown in Table 1.1, it is always a trade-off. One can choose the synchronous machine to have high efficiency but the slip rings ask maintenance and may lead to overheat problems while induction machine has no slip rings. But its efficiency is lower. Additionally, the induction machine has a simpler model and a simpler control system. The doubly fed induction machine seems to be more complex and attractive but still drawbacks remain such as slip rings, small speed range and high complexity. Finally, no particular machine stands out of the others and as it consists of a starting project without any experience in the domain, the complexity of the model has a huge interest. The simpler one is thus chosen to ensure the good working of the machine. In further work, the same kind of study should be performed to compare in more details all introduced electrical machines. An asynchronous machine of 10 MW is chosen with the following characteristics:

- rotor inertia: $J_r = 371 \text{ kgm}^2$;
- nominal power: 10.15 kW;
- power factor: 0.89;
- nominal voltage: $V_B = 4160 \text{ V}$;
- number of pairs of poles: $p = 2$;
- nominal frequency: $f_B = 60 \text{ Hz}$.

The nominal apparent three-phase power is equal to the nominal power divided by the power factor and is denoted $S_B$.

### 1.4 Main stakes

This work is a before-project of the study and realisation of a compressor test rig in Safran Aero Boosters. Their main stakes are to give test rigs to the Research and Technology, to remain autonomous and additionally to offer a new product which is a compressor test rig. An induction machine is envisaged driving the compressor that has to be tested.

The work consists firstly to find a model of the induction machine based on physical laws. Then, an analysis of its dynamic is asked to ensure that no mistakes have been made. To this aim, some solver tools have to be chosen. The goal is to have a first contact with the induction machine and the way it reacts to time-varying inputs such as the valve closure percentage of the compressor (linked to the rotor of the electric motor) or the rotor resistance. Finally, when the model of the induction machine is validated, the main interest of Safran is to have a simulation of speed regulation. So, some regulations must be found and the influence of the main parameters of the model must be studied. To conclude, a comparison of the different regulations is asked, as well as advice for use of this work and future work proposition.
The system is composed of an asynchronous machine, an electronic power supply or the grid to feed the machine and a mechanical load which is the compressor, the shaft and the gearbox. This chapter is divided into 7 sections. «General model» contains the general dynamic equations with very few assumptions. «Park transformation» is a transformation tool that basically allows three-phase balanced equations to be transformed in equivalent two-phase equations. «Model expressed in Park axes» is just a rearrangement of the equations using the transformation tool explained in the previous subsection. «Introduction of per unit system» deals with the equations of the machine using dimensionless quantities. Those new values are in per unit and the real physical values can be obtained by multiplying those per unit values by basis values which characterise the machine. «Phasor approximation» make the assumption that the stator supply has fast response with respect to the rates of change of rotor states and come from an assumption made in the network. «Final system using Park and phasor approximation» contains the final system of equations in per unit and with the phasor approximation assumptions. «Induction machine at steady-state» re-expresses the equations at steady-state of the induction machine. It also shows the mechanical torque produced by the electric components (electrical torque).

### 2.1 General model

The model is based on the references: [1]. To simplify the development, a single phase machine is considered but the derived equations can be extended easily to a multi-phase machine. This multi-phase machine will be deduced later in this thesis. In Figure 2.1, the rotor and the stator of the induction machine is shown as well as the link with the grid (at the stator) and with the compressor (at the rotor). The equations modelling the system are divided into 5 main parts: the applied voltages from the grid at the stator(1.), the current-voltage-flux relationship in the stator(2.), the same for the rotor(3.), the flux relationship created by both rotor and stator windings(4.) and finally the mechanical motion in rotation of the compressor (linked to the rotor via a gearbox)(5.).

![Figure 2.1: Model of the system and dynamic equations linked to different machine parts](image-url)
In Figure 2.2 and 2.3, the asynchronous machine is represented in two different ways. The first schematic shows the windings of the machine while the second schematic represents the axis of those windings modelled with the inductance symbol. The rotor angle, $\theta_r$, is the angle between one phase of the stator windings and the equivalent phase of the rotor windings.

### 2.1.1 Relation voltage-current at each winding

The Kirchoff’s law is applied on the stator and on the rotor. A motor convention is taken for those windings.

At the stator:

\[
\begin{align*}
    v_a &= R_s i_a + \frac{d\psi_a}{dt}, \\
    v_b &= R_s i_b + \frac{d\psi_b}{dt}, \\
    v_c &= R_s i_c + \frac{d\psi_c}{dt},
\end{align*}
\]

At the rotor:

\[
\begin{align*}
    0 &= R_r i_A + \frac{d\psi_A}{dt}, \\
    0 &= R_r i_B + \frac{d\psi_B}{dt}, \\
    0 &= R_r i_C + \frac{d\psi_C}{dt}.
\end{align*}
\]
As a reminder, the rotor is in short-circuit. Therefore, \( v_A, v_B \) and \( v_C \) are null. The equations binding currents to voltages at both rotor and stator windings can be rewritten in a matrix form as:

\[
\begin{align*}
v_{abc} &= R_s i_{abc} + \frac{d}{dt} \psi_{abc}; \\
0 &= R_r i_{ABC} + \frac{d}{dt} \psi_{ABC}.
\end{align*}
\] (2.1) (2.2)

### 2.1.2 Flux flowing in each winding

The flux entering in a winding is the combination of all fluxes induced by all windings. For instance, here is the flux entering the «a» winding:

\[
\psi_a = L_s i_a - M_s i_b - M_s i_c + M_{sr} \left( i_A \cos(\theta_r) + i_B \cos \left( \theta_r + \frac{2\pi}{3} \right) + i_C \cos \left( \theta_r + \frac{4\pi}{3} \right) \right).
\]

It is to note that in the literature the flux is also often written «\( \phi \)» instead of «\( \psi \)». All inductances \((L_s, M_s, L_r, M_r, M_{sr})\) are positive definite. The fluxes created by the windings \(b\) and \(c\) flow through the winding \(a\) in the opposite direction of that created by the winding \(a\). This explains their negative contribution to \(\psi_a\).

As the problem has some kinds of symmetry, when looking for \(\psi_b\), the formula remains the same except that subscripts are shifted. So the applied shift is the following «\(a, b, c\)» → «\(b, c, a\)» and «\(A, B, C\)» → «\(B, C, A\)» and it leads to \(\psi_b\) expression:

\[
\psi_b = L_s i_b - M_s i_c - M_s i_a + M_{sr} \left( i_B \cos(\theta_r) + i_C \cos \left( \theta_r + \frac{2\pi}{3} \right) + i_A \cos \left( \theta_r + \frac{4\pi}{3} \right) \right).
\]

By generalising this shift to obtain all fluxes, a matrix formula can be found to summary those equations:

\[
\begin{bmatrix}
\psi_{abc} \\
\psi_{ABC}
\end{bmatrix} = \begin{bmatrix}
L_{SS} & L_{SR}(\theta_r) \\
L_{SR}^T(\theta_r) & L_{RR}
\end{bmatrix} \begin{bmatrix}
i_{abc} \\
i_{ABC}
\end{bmatrix};
\] (2.3)

where the different matrices \(L_{SS}, L_{RR}\) and \(L_{SR}(\theta)\) is given by:

\[
L_{SS} = \begin{bmatrix}
L_s & -M_s & -M_s \\
-M_s & L_s & -M_s \\
-M_s & -M_s & L_s
\end{bmatrix}; \quad L_{RR} = \begin{bmatrix}
L_r & -M_r & -M_r \\
-M_r & L_r & -M_r \\
-M_r & -M_r & L_r
\end{bmatrix};
\]

\[
L_{SR}(\theta) = M_{sr} \begin{bmatrix}
\cos(\theta_r) & \cos \left( \theta_r + \frac{2\pi}{3} \right) & \cos \left( \theta_r + \frac{4\pi}{3} \right) \\
\cos \left( \theta_r + \frac{2\pi}{3} \right) & \cos(\theta_r) & \cos \left( \theta_r + \frac{4\pi}{3} \right) \\
\cos \left( \theta_r + \frac{4\pi}{3} \right) & \cos \left( \theta_r + \frac{2\pi}{3} \right) & \cos(\theta_r)
\end{bmatrix}.
\]

The main problem at this stage is the highly non-linear dependency of fluxes to the rotor angle. These equations are particularly difficult to solve efficiently. It is why, in the second section of this chapter, a tool, the Park transformation, is introduced to simplify these equations.

### 2.1.3 Voltage sources for the stator

The voltage applied to each phase of the stator is modelled by an equivalent Thevenin circuit for each phase.
CHAPTER 2. MATHEMATICAL MODEL OF THE ASYNCHRONOUS MACHINE

In practice, that can represent either the grid or a power device such as a frequency generator. As a motor convention was used for the stator windings, it is consistent to take a generator convention for the grid or the electronic device at which the stator is plugged. The equations resulting of this circuit are the following:

\[
\begin{align*}
    v_a &= e_a - R_e i_a - L_e \frac{d i_a}{dt} \quad \text{with} \quad e_a = \sqrt{2} E \cos(\omega_s t + \theta_e); \\
    v_b &= e_b - R_e i_b - L_e \frac{d i_b}{dt} \quad \text{with} \quad e_b = \sqrt{2} E \cos\left(\omega_s t + \theta_e - \frac{2\pi}{3}\right); \\
    v_c &= e_c - R_e i_c - L_e \frac{d i_c}{dt} \quad \text{with} \quad e_c = \sqrt{2} E \cos\left(\omega_s t + \theta_e - \frac{4\pi}{3}\right); 
\end{align*}
\]

where \( E \) is the effective voltage \(^1\) given by the grid or the electrical device plugged. \( \theta_e \) is arbitrarily chosen at this stage. But further, it will be taken in order to have a cosine with a null phase at the stator.

2.1.4 Mechanical motion equation

The rotor is rotating with a given equivalent inertia \( J \), including the induction rotor inertia, the compressor inertia and the shaft inertia. The equation is given by:

\[
J \frac{d\omega_r}{dt} = T_e - T_m(\omega_r) 
\]

\[
T_m(\omega_r) = \frac{1}{\eta GR} T_{\text{comp}}(\omega_r) 
\]

\[
J = J_r + \frac{1}{\eta GR^2} J_{\text{comp}} 
\]

where \( T_e \) is the electrical torque that must be computed as respect to electrical states, \( T_m \) is the mechanical torque applied by the compressor depending on the shaft rotation speed, \( GR \) gearbox ratio and \( \eta \) gearbox efficiency. The damping term is assumed hidden in the compressor torque.

2.2 Park transformation

The Park transformation is applied in order to go from three-phase windings to a Park plan. This transformation is performed by multiplying three-phase quantities by this matrix:

\[
\mathcal{P}(\theta_p) = \sqrt{\frac{2}{3}} \begin{bmatrix} 
\cos(\theta_p) & \cos(\theta_p - \frac{2\pi}{3}) & \cos(\theta_p + \frac{2\pi}{3}) \\
\sin(\theta_p) & \sin(\theta_p - \frac{2\pi}{3}) & \sin(\theta_p + \frac{2\pi}{3}) \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} 
\end{bmatrix} 
\]

\(^1\) also called Root Mean Square (RMS) voltage in the literature
where $\theta_\text{P}$ is the angle from «$d$» axis to «$a$» axis or «$A$» axis for respectively the three-phase stator windings and the three-phase rotor windings (Figure 2.6 and Figure 2.7). It is given by:

\[
\begin{align*}
\text{at the stator: } & \quad \theta_\text{P} = \theta_s; \\
\text{at the rotor: } & \quad \theta_\text{P} = \theta_s - \theta_r.
\end{align*}
\]

(2.11) (2.12)

This transformation is an orthogonal matrix:

\[
\mathbf{P} \mathbf{P}^T = \mathbf{I} \iff \mathbf{P}^{-1} = \mathbf{P}^T. 
\]

(2.13)

The Park transformation is used on voltages, currents and fluxes at three-phase windings to obtain the equivalent windings on «$d$», «$q$», «$o$» axes. «$d$» stands for direct axis and «$q$» stands for quadrature axis.

The interpretation of this transformation is the following: it replaces the current windings by 3 equivalent ones along «$d$», «$q$», «$o$» axes. Actually, this transformation is managed in such a way that three-phase state will only be projected on «$d$», «$q$» axes. Those windings recreate the same total magnetic field and thus, the windings along «$d$» (respectively «$q$») represents the total magnetic field projected on the «$d$» (respectively «$q$») axis. For the specific case of an induction machine, the park plan is often chosen to rotate at the stator frequency speed and thus follow the stator flux. For this application, the initial state of the park plan is when the «$q$» axis coincides with the «$a$» axis. To apply this transformation, the Park matrix multiplies a three-phase state vector such as current, flux or voltage to obtain the corresponding state in «$dqo$» axis:

\[
\mathbf{x}_\text{P} = \begin{bmatrix} x_{ds,dr} \\ x_{qs,qr} \\ x_{os,or} \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_{a,A} \\ x_{b,B} \\ x_{c,C} \end{bmatrix} = \mathbf{P} \mathbf{x}_T
\]

(2.14)

One advantage of this transformation is that if the three-phase is balanced, then, in Park plan, the state $o$ is always null. To understand better this transformation, a useful example is introduced below.
An ideal source is applied to the stator and the machine is assumed working at steady-state. The three-phase stator voltages and currents are given by:

\[
\begin{align*}
v_a(t) &= \sqrt{2}V \cos(\omega_s t + \alpha); \\
v_b(t) &= \sqrt{2}V \cos\left(\omega_s t + \alpha - \frac{2\pi}{3}\right); \\
v_c(t) &= \sqrt{2}V \cos\left(\omega_s t + \alpha + \frac{2\pi}{3}\right); \\
i_a(t) &= \sqrt{2}I \cos(\omega_s t + \beta); \\
i_b(t) &= \sqrt{2}I \cos\left(\omega_s t + \beta - \frac{2\pi}{3}\right); \\
i_c(t) &= \sqrt{2}I \cos\left(\omega_s t + \beta + \frac{2\pi}{3}\right).
\end{align*}
\]

Now if the Park transformation is applied to these voltages and currents, the following results are obtained (Park angle: \(\theta_P(t) = \theta_s^0 + \omega_s t\)):

\[
\begin{align*}
\text{i}_{ds} &= \sqrt{3}I \cos(\theta_s^0 - \beta); \\
\text{i}_{qs} &= \sqrt{3}I \sin(\theta_s^0 - \beta); \\
\text{i}_{os} &= 0; \\
\text{v}_{ds} &= \sqrt{3}V \cos(\theta_s^0 - \alpha); \\
\text{v}_{qs} &= \sqrt{3}V \sin(\theta_s^0 - \alpha); \\
\text{v}_{os} &= 0;
\end{align*}
\]

where \(\theta_s^0\) is the initial angle between the \(d\) axis and the \(a\) axis. As a reminder, a decision was taken for the initial state of the Park plan: the \(q\) axis is aligned with the \(a\) axis. That implies \(\theta_s^0 = \frac{\pi}{2}\). As expected here, the currents and the voltages in the equivalent \(d, q\) windings are constant and create a constant flux equal to the amplitude of the rotating flux induced by the three-phase stator windings. In reality, the stator generates a magnetic flux rotating at a speed \(\omega_s\). Then, considering the \(d, q\) windings, a constant flux is recreated and the Park plan is rotating at the same speed \(\omega_s\). Thus, seen from the stator, the flux created by the \(d, q\) windings has well a frequency given by \(\omega_s\).

### 2.3 Model expressed in Park axes

In this subsection, the general model is rewritten in the Park plan by replacing all three-phase states (currents, voltages, fluxes) by equivalent windings in \(\{d, q, o\}\) axes. All previous equations established in the «General model» section are transformed in the Park domain.
2.3.1 Relation voltage-current at each winding

The following development shows how to pass from equations (2.1) and (2.2) in the three-phase plan to a Park plan using formula (2.14):

\[
v_T = R_T i_T + \frac{d}{dt} \psi_T;
\]

\[
\mathcal{P}^{-1} v_p = R_T \mathcal{P}^{-1} i_p + \frac{d}{dt} \left( \mathcal{P}^{-1} \psi_p \right);
\]

\[
v_p = R_T \mathcal{P} \mathcal{P}^{-1} i_p + \mathcal{P} \frac{d}{dt} \left( \mathcal{P}^{-1} \psi_p \right) + \mathcal{P} \mathcal{P}^{-1} \frac{d}{dt} \psi_p; \tag{2.21}
\]

\[
= R_T i_p + \mathcal{P} \dot{\theta}_p \psi_p + \frac{d}{dt} \psi_p; \tag{2.22}
\]

with

\[
P = \begin{bmatrix}
0 & 1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 0 
\end{bmatrix};
\]

where the subscript \( P \) stands for Park coordinates («\( d, q, o \)» and \( T \) for the three-phase coordinates (subscripts «\( abc \)» or «\( ABC \)»). To pass from (2.21) to (2.22) equation, the derivative of the Park matrix must be computed noting that it depends on the park angular which depends itself on time. This results in a new term \( \dot{\theta}_p \) which is the speed of the rotating Park plan seen from the three-phase plan.

Now applying the formula (2.22) for the stator, the 3 following equations are obtained:

\[
v_{ds} = R_s i_{ds} + \omega_s \psi_{qs} + \frac{d \psi_{ds}}{dt} \tag{2.23}
\]

\[
v_{qs} = R_s i_{qs} - \omega_s \psi_{ds} + \frac{d \psi_{qs}}{dt} \tag{2.24}
\]

\[
v_{os} = R_s i_{os} + \frac{d \psi_{os}}{dt} \tag{2.25}
\]

Same kinds of equations is deduced for the rotor where the relative speed between the Park plan and the three-phase rotor, is \( \omega_s - \omega_r \):

\[
v_{dr} = 0 = R_r i_{dr} + (\omega_s - \omega_r) \psi_{qr} + \frac{d \psi_{dr}}{dt} \tag{2.26}
\]

\[
v_{qr} = 0 = R_r i_{qr} - (\omega_s - \omega_r) \psi_{dr} + \frac{d \psi_{qr}}{dt} \tag{2.27}
\]

\[
v_{or} = 0 = R_r i_{or} + \frac{d \psi_{or}}{dt} \tag{2.28}
\]

Those three voltages are null because they are combinations of three-phase rotor voltages which are short-circuited.

2.3.2 Flux flowing in each winding

The equation (2.3) in the Park plan becomes:

\[
\begin{bmatrix}
\mathcal{P}(\theta_s)^{-1} \psi_{dqs} \\
\mathcal{P}(\theta_s - \theta_r)^{-1} \psi_{dqr}
\end{bmatrix} =
\begin{bmatrix}
L_{SS} & L_{SR}(\theta) \\
L_{SR}^T(\theta) & L_{RR}
\end{bmatrix}
\begin{bmatrix}
\mathcal{P}(\theta_s)^{-1} i_{dqs} \\
\mathcal{P}(\theta_s - \theta_r)^{-1} i_{dqr}
\end{bmatrix};
\]

\[
\begin{bmatrix}
\psi_{dqs} \\
\psi_{dqr}
\end{bmatrix} =
\begin{bmatrix}
\mathcal{P}(\theta_s) L_{SS} \mathcal{P}(\theta_s)^{-1} & \mathcal{P}(\theta_s) L_{SR}(\theta) \mathcal{P}(\theta_s - \theta_r)^{-1} \\
\mathcal{P}(\theta_s - \theta_r) L_{SR}^T(\theta) \mathcal{P}(\theta_s)^{-1} & \mathcal{P}(\theta_s - \theta_r) L_{RR} \mathcal{P}(\theta_s - \theta_r)^{-1}
\end{bmatrix}
\begin{bmatrix}
i_{dqs} \\
i_{dqr}
\end{bmatrix};
\]

\[
\begin{bmatrix}
\psi_{dqs} \\
\psi_{dqr}
\end{bmatrix} =
\begin{bmatrix}
L_{dqs} & i_{dqs} \\
L_{dqr} & i_{dqr}
\end{bmatrix}; \tag{2.29}
\]

17
where hopefully, the new inductance matrix is constant:

$$L_{dqo} = \begin{bmatrix}
L_{ss} & L_{sr} & L_{os} & L_{sr} \\
L_{sr} & L_{ss} & L_{sr} & L_{sr} \\
L_{os} & L_{sr} & L_{rr} & L_{rr} \\
L_{sr} & L_{sr} & L_{rr} & L_{or}
\end{bmatrix}. \tag{2.30}$$

Zero entries have been left for legibility and the elements of this matrix is given below:

\begin{align*}
L_{ss} &= L_s + M_s; \tag{2.31} \\
L_{os} &= L_s - 2M_s; \tag{2.32} \\
L_{sr} &= \frac{3}{2}M_{sr}; \tag{2.33} \\
L_{rr} &= L_r + M_r; \tag{2.34} \\
L_{or} &= L_r - 2M_r. \tag{2.35}
\end{align*}

Having a constant inductance matrix was expected. Indeed, all fictive windings attached to the Park plan move at the same speed, so, the flux seen by each winding will not depend anymore on the relative positioning of the rotor as respect to the stator. Physically this relative positioning effect is hidden in the evolution of Park state values: \(\psi_{dqo}\) and \(i_{dqo}\). Its dynamic influence is mainly shown in the relation voltage-current at each winding given by equations (2.23) to (2.28).

### 2.3.3 Power balance and electric torque

A way to find the torque expression is to establish a power balance. The total power balance is separated into 2 parts: one at the stator and one at the rotor.

At the stator, the input power is given by the power entering the stator. This power is partially dissipated by Joule losses \(p_{js}\). The remaining components are passed to the rotor \(p_{s\rightarrow r}\). Besides a magnetic energy \(W_{Ms}\) is needed to create the flux but the windings return the consumed energy in such a way that on average, it does not consume or produce any power. The power balance at the stator is expressed below:

$$p_T(t) = p_{js} + \frac{dW_{Ms}}{dt} + p_{s\rightarrow r}. \tag{2.36}$$

The stator power is given by:

$$p_T(t) = v_a^T i_a;$$

$$p_T(t) = v_b^T i_b;$$

$$p_T(t) = v_c^T i_c;$$

$$p_T(t) = v_{ds}^T g_P g_P^T i_{ds};$$

$$p_T(t) = v_{qs}^T g_P g_P^T i_{qs};$$

$$p_T(t) = v_{os}^T g_P g_P^T i_{os};$$

$$p_T(t) = v_{ds} i_{ds} + v_{qs} i_{qs} + v_{os} i_{os}.$$
From the above equation, all terms of equation (2.36) can be identified:

\[ P_{fs} = R_s \left( i_{ds}^2 + i_{qs}^2 + i_{os}^2 \right) \]  (2.37)

\[ \frac{dW_{Ms}}{dt} = i_{ds} \frac{d\psi_{ds}}{dt} + i_{qs} \frac{d\psi_{qs}}{dt} + i_{os} \frac{d\psi_{os}}{dt} \]  (2.38)

\[ P_{s \rightarrow r} = \omega_s \left( \psi_{qs} i_{ds} - \psi_{ds} i_{qs} \right) \]  (2.39)

At this stage, it is important to note that the transferred power is both of mechanical and electrical nature. Indeed, the electrical nature is obvious. If the stator is fed by an ideal source, the equivalent stator windings currents on \( d \) and \( q \) axis are constant such as proven in a previous subsection («Park transformation»). But the transferred power is still able to vary in time since the fluxes \( \psi_{ds} \) and \( \psi_{qs} \) depend on \( i_{dr} \) and \( i_{qr} \) which are themselves related to the rotor mechanical motion via \( \omega_r \).

At the rotor, the transferred power from the stator is equal to the derivative of kinetic energy, the mechanical power absorbed by the compressor, the Joule losses and the derivative of the magnetic energy in rotor windings:

\[ P_{s \rightarrow r} = \frac{dW_c}{dt} + P_m + P_{fr} + \frac{dW_{Mr}}{dt}. \]  (2.40)

Noting that voltages \( v_{dr}, v_{qr} \) and \( v_{or} \) are null, the following equation is assumed:

\[ v_{dr} i_{dr} + v_{qr} i_{qr} + v_{or} i_{or} = 0. \]

Replacing voltages thanks to equations (2.26),(2.27) and (2.28), it follows that:

\[ R_r(i_{dr}^2 + i_{qr}^2 + i_{or}^2) + i_{dr} \frac{\psi_{dr}}{dt} + i_{qr} \frac{\psi_{qr}}{dt} + i_{or} \frac{\psi_{or}}{dt} + (\omega_s - \omega_r)(\psi_{qr} i_{dr} - \psi_{dr} i_{qr}) = 0. \]

Some terms of equation (2.40) are identified:

\[ P_{fr} = R_r(i_{dr}^2 + i_{qr}^2 + i_{or}^2); \]  (2.41)

\[ \frac{dW_{Mr}}{dt} = i_{dr} \frac{\psi_{dr}}{dt} + i_{qr} \frac{\psi_{qr}}{dt} + i_{or} \frac{\psi_{or}}{dt}; \]  (2.42)

\[ \frac{dW_{Mr}}{dt} = -(\omega_s - \omega_r)(\psi_{qr} i_{dr} - \psi_{dr} i_{qr}). \]  (2.43)

The equations (2.39) and (2.43) are injected in equation (2.40):

\[ \omega_s \left( \psi_{qs} i_{ds} - \psi_{ds} i_{qs} \right) = \frac{dW_c}{dt} + P_m - (\omega_s - \omega_r)(\psi_{qr} i_{dr} - \psi_{dr} i_{qr}). \]  (2.44)

The last part concerns the mechanical part. By multiplying the mechanical motion equation by the rotational speed, it leads to:

\[ \int \frac{d\omega}{dt} \omega = T_e \omega - T_m(\omega)\omega; \]  (2.45)

where \( \omega = \frac{\omega_p}{p} \) with \( p \), the number of pairs of poles. The equations (2.44) and (2.45) represent the same power balance:

\[ \frac{dW_c}{dt} = \int \frac{d\omega}{dt} \omega; \]  (2.46)

\[ P_m = T_m(\omega)\omega; \]  (2.47)

\[ T_e \omega = (\omega_s - \omega_r)(\psi_{qr} i_{dr} - \psi_{dr} i_{qr}) + \omega_s \left( \psi_{qs} i_{ds} - \psi_{ds} i_{qs} \right). \]  (2.48)
Applying the flux equations (2.29) and (2.30), it can be deduced that:

\[
\psi_{qs}i_{ds} - \psi_{ds}i_{qs} = (L_{ss}i_{qs} + L_{sr}i_{qr})i_{ds} - (L_{ss}i_{ds} + L_{sr}i_{dr})i_{qs};
\]
\[
= L_{sr}(i_{qr}i_{ds} - i_{dr}i_{qs}); \quad (2.49)
\]

\[
\psi_{qr}i_{dr} - \psi_{dr}i_{qr} = (L_{rr}i_{qr} + L_{sr}i_{qs})i_{dr} - (L_{rr}i_{dr} + L_{sr}i_{ds})i_{qr};
\]
\[
= L_{sr}(i_{qs}i_{dr} - i_{ds}i_{qr}) = - (\psi_{qs}i_{ds} - \psi_{ds}i_{qs}). \quad (2.50)
\]

Using the relation (2.50), the electrical torque (2.48) becomes:

\[
T_e = \frac{\omega_r}{\omega} (\psi_{qs}i_{ds} - \psi_{ds}i_{qs});
\]
\[
= p (\psi_{qs}i_{ds} - \psi_{ds}i_{qs}); \quad (2.51)
\]
\[
= p (\psi_{dr}i_{qr} - \psi_{qr}i_{dr}). \quad (2.52)
\]

2.4 Introduction of per unit system

In order to have some general data for the induction machine parameters (resistances, inductances), a basis system is established to transform the dynamic equations into quasi-adimentionnalized equations. The only dimension that remains is the time. Indeed in practice, devices with similar design have close parameter values in per unit whatever the power supply. So in reality, those parameters mainly depend on the power.

<table>
<thead>
<tr>
<th></th>
<th>in the single phase circuit equivalent to stator windings</th>
<th>in each of the ( d, q ) windings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>( t_B = \frac{1}{\omega_s} = \frac{1}{\omega_B} )</td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>( S_B ): nominal apparent three-phase power</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>( V_B ): nominal (RMS) phase-neutral</td>
<td>( \sqrt{3} V_B )</td>
</tr>
<tr>
<td>Impedance</td>
<td>( I_B = \frac{S_B}{3V_B} )</td>
<td>( \sqrt{3} I_B )</td>
</tr>
<tr>
<td>Flux</td>
<td>( V_B t_B )</td>
<td>( \sqrt{3} V_B t_B )</td>
</tr>
<tr>
<td>Rotor speed</td>
<td>( \omega_{mB} = \frac{\omega_B}{p} )</td>
<td></td>
</tr>
<tr>
<td>Torque</td>
<td>( T_B = \frac{S_B}{\omega_{mB}} )</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Per unit system used for the asynchronous machine

Now, the equivalent dynamic equations in per unit system will be derived. First, the motion equation of the rotor will be changed in per unit. Then, the electromagnetic torque will also be deduced. At the next sub-subsection, fluxes as respect to currents and derivative of fluxes will be given in per unit. Afterwards, the voltage source equation will be bound to the equivalent Park states. And finally, the whole system obtained in per unit is set with tables of standard per unit values used for the resistances and inductances.
2.4. INTRODUCTION OF PER UNIT SYSTEM

2.4.1 Rotor motion

As a reminder, the rotor motion is given by the dynamic equation:

\[ J \frac{d\omega_m}{dt} = T_e - T_m. \]  

(2.53)

The base torque is set as:

\[ T_B = \frac{S_B}{\omega_{mB}}; \]  

(2.54)

with \( \omega_{mB} = \frac{\omega_s}{p} = \frac{\omega_B}{p}. \)  

(2.55)

By dividing the equation (2.53) by the base torque \( T_B \), it becomes:

\[ \frac{J \omega_{mB} \omega_B}{p S_B} \frac{d\omega}{dt} = \frac{T_e}{p}, \text{pu} - \frac{T_m}{p}, \text{pu}; \]

noting that \( \omega = p \omega_m \), the rate of phase change at the rotor. \( \omega \) is rewritten in term of per unit and basis parameters: \( \omega = \omega_B \omega_{r,pu} \). And \( \omega_{r,pu} \) is thus no more the rotor speed but the rate of phase change in per unit system. Here is the development that leads to a per unit equation:

\[ \frac{J \omega_{mB} \omega_B}{p S_B} \frac{d\omega_{r,pu}}{dt} = \frac{T_e}{p}, \text{pu} - \frac{T_m}{p}, \text{pu}; \]

\[ \frac{J \omega_{mB}^2}{S_B} \frac{d\omega_{r,pu}}{dt} = \frac{T_e}{p}, \text{pu} - \frac{T_m}{p}, \text{pu}; \]

\[ 2H \frac{d\omega_{r,pu}}{dt} = \frac{T_e}{p}, \text{pu} - \frac{T_m}{p}, \text{pu}; \]

\[ T_{m,pu} = T_{me,pu} \left( A \omega_{r,pu}^2 + B \omega_{r,pu} + C \right) \]  

(2.56)

(2.57)

The parameter \( H \) is given in second and it represents the ratio between the kinetic energy of rotating masses at synchronous speed and the apparent nominal power:

\[ H = \frac{1}{2} \frac{J \omega_{mB}^2}{S_B}. \]  

(2.58)

An important note is that the equation (2.56) does not depend on \( p \), the number of pairs of poles.

2.4.2 Electromagnetic torque

As a reminder the electromagnetic torque is given by:

\[ T_e = p(\psi_{dr,iqr} - \psi_{qriqr}) \]  

(2.59)

Now, dividing by the basis torque \( T_B \), it becomes:

\[ T_{e,pu} = \frac{T_e}{T_B} = \frac{\omega_{mB}}{S_B} p(\psi_{dr,iqr} - \psi_{qriqr}) \]  

(2.60)

Noting that \( S_B = 3V_B I_B \) as set in the table 2.1, the equation is simplified:

\[ T_{e,pu} = \frac{\psi_{dr}}{\sqrt{3}V_B I_B} \frac{i_{qriqriqr}}{\sqrt{3}V_B I_B} - \frac{\psi_{qriqr}}{\sqrt{3}V_B I_B} \frac{i_{dr}}{\sqrt{3}V_B I_B} \]

\[ = \psi_{dr,pu} i_{qriqr} - \psi_{qriqr} i_{dr,pu} \]  

(2.61)

An important note is that the equation (2.61) does not depend on \( p \), the number of pairs of poles.
2.4.3 Flux equations and flux derivatives

For the fluxes derivatives, the equations (2.23) to (2.28) becomes in per unit:

\[
\begin{align*}
\psi_{ds,pu} &= R_s \psi_{ds,pu} + \frac{d\psi_{ds,pu}}{dt} t_B; \\
\psi_{qs,pu} &= R_s \psi_{qs,pu} - \omega_s \psi_{ds,pu} + \frac{d\psi_{qs,pu}}{dt} t_B; \\
\psi_{os,pu} &= R_s \psi_{os,pu} + \frac{d\psi_{os,pu}}{dt} t_B; \quad (2.62) \\
\psi_{dqo,pu} &= L_{dqo,pu} \begin{bmatrix} i_{dqo,s,pu} \\ i_{dqo,r,pu} \end{bmatrix}; \quad (2.68)
\end{align*}
\]

And finally, the same transformation to a per unit system can be applied to equation (2.29):

\[
\begin{align*}
\psi_{dqo,pu} &= L_{dqo,pu} \begin{bmatrix} i_{dqo,s,pu} \\ i_{dqo,r,pu} \end{bmatrix}; \quad (2.68)
\end{align*}
\]

knowing that the inductance basis is \( L_B = Z_B t_B \).

2.4.4 Grid/supply at the stator

The voltage sources at the stator is given in per unit:

\[
\begin{align*}
\psi_{ds,pu} &= R_s \psi_{ds,pu} + \frac{d\psi_{ds,pu}}{dt} t_B; \\
\psi_{qs,pu} &= R_s \psi_{qs,pu} - \omega_s \psi_{ds,pu} + \frac{d\psi_{qs,pu}}{dt} t_B; \\
\psi_{os,pu} &= R_s \psi_{os,pu} + \frac{d\psi_{os,pu}}{dt} t_B; \quad (2.62) \\
\psi_{dqo,pu} &= L_{dqo,pu} \begin{bmatrix} i_{dqo,s,pu} \\ i_{dqo,r,pu} \end{bmatrix}; \quad (2.68)
\end{align*}
\]

And the equivalent values in the park plan in per unit are:

\[
\begin{align*}
\begin{bmatrix} v_{ds,pu} \\ v_{qs,pu} \\ v_{os,pu} \end{bmatrix} &= \frac{1}{\sqrt{3}} \mathcal{P}(\theta^0_s + \omega_s t) \begin{bmatrix} v_{a,pu} \\ v_{b,pu} \\ v_{c,pu} \end{bmatrix}; \quad (2.69) \\
\begin{bmatrix} i_{ds,pu} \\ i_{qs,pu} \\ i_{os,pu} \end{bmatrix} &= \frac{1}{\sqrt{3}} \mathcal{P}(\theta^0_s + \omega_s t) \begin{bmatrix} i_{a,pu} \\ i_{b,pu} \\ i_{c,pu} \end{bmatrix}. \quad (2.70)
\end{align*}
\]
2.4.5 Final system and standard parameter values of induction machine

The final system that models the dynamic of the machine, is:

\[
v_a = e_a - R_e i_a - L_e \frac{di_a}{dt} t_B \quad \text{with} \quad e_a = \sqrt{2} \cos(\omega_s t + \theta_e); \quad (2.71)
\]

\[
v_b = e_b - R_e i_b - L_e \frac{di_b}{dt} t_B \quad \text{with} \quad e_b = \sqrt{2} \cos(\omega_s t + \theta_e - \frac{2\pi}{3}); \quad (2.72)
\]

\[
v_c = e_c - R_e i_c - L_e \frac{di_c}{dt} t_B \quad \text{with} \quad e_c = \sqrt{2} \cos(\omega_s t + \theta_e - \frac{4\pi}{3}); \quad (2.73)
\]

\[
\begin{bmatrix}
    v_{ds} \\
    v_{qs} \\
    v_{os}
\end{bmatrix} = \frac{1}{\sqrt{3}} \mathcal{P}(\theta_s^0 + \omega_s t) \begin{bmatrix}
    v_a \\
    v_b \\
    v_c
\end{bmatrix}; \quad (2.74)
\]

\[
\begin{bmatrix}
    i_{ds} \\
    i_{qs} \\
    i_{os}
\end{bmatrix} = \frac{1}{\sqrt{3}} \mathcal{P}(\theta_s^0 + \omega_s t) \begin{bmatrix}
    i_a \\
    i_b \\
    i_c
\end{bmatrix}; \quad (2.75)
\]

\[
v_{ds} = R_s i_{ds} + \omega_s \psi_{qs} + \frac{d\psi_{ds}}{dt} t_B; \quad (2.76)
\]

\[
v_{qs} = R_s i_{qs} - \omega_s \psi_{ds} + \frac{d\psi_{qs}}{dt} t_B; \quad (2.77)
\]

\[
v_{os} = R_s i_{os} + \frac{d\psi_{os}}{dt} t_B; \quad (2.78)
\]

\[
0 = R_s i_{dr} + (\omega_s - \omega_r) \psi_{qr} + \frac{d\psi_{dr}}{dt} t_B; \quad (2.79)
\]

\[
0 = R_s i_{qr} - (\omega_s - \omega_r) \psi_{dr} + \frac{d\psi_{qr}}{dt} t_B; \quad (2.80)
\]

\[
0 = R_s i_{or} + \frac{d\psi_{or}}{dt} t_B \quad (2.81)
\]

where everything is in per unit (the «pu» subscript is given up to be more legible). Typical values for the induction machine is shown in the table 2.2.

| \(R_s\) | 0.01 - 0.12 pu |
| \(L_{ss} - L_{sr}\) | 0.07 - 0.15 pu |
| \(L_{sr}\) | 1.8 - 3.8 pu |

| \(R_r\) | 0.01 - 0.13 pu |
| \(L_{rr} - L_{sr}\) | 0.06 - 0.18 pu |

Table 2.2: Per unit values on the machine base

Concerning \(X_e, R_e\), in practice, they are very small (\(\approx 1 \times 10^{-3}\)). And for the per unit inertia \(H\) expressed in second, its value is generally between 0.4 and 2 s. In practice, it is better to evaluate the moment of inertia \(J\).
and to compute $H$ by the formula:

$$H = \frac{1}{2} J \omega_m R_S.$$  

2.5 Phasor approximation

The phasor (or quasi-sinusoidal) approximation is a model simplification that comes from the grid. It consists to say that the input voltages and currents are of the following forms:

$$v(t) = \sqrt{2} V(t) \cos(\omega_s t + \alpha(t));$$
$$i(t) = \sqrt{2} I(t) \cos(\omega_s t + \beta(t));$$

where intuitively the real signal is approximated by a sine wave with time-varying amplitude and time-varying phase (resulting in a frequency change of the signal). In transmission lines of networks, this assumption is very common. And there is no reason not to apply it since the power supply follows this same assumption. It is widely used for power system dynamic simulators. It also consists of neglecting the "transformer voltages" at the stator $\frac{d\psi_d}{dt}$ and $\frac{d\psi_q}{dt}$. The three-phase balance is also assumed.

2.5.1 Source voltage for a single phase

Each single phase $a$, $b$ and $c$ of the stator takes of the form:

$$v(t) = \sqrt{2} V(t) \cos(\omega_s t + \alpha(t)) = \sqrt{2} \Re \left( \tilde{V}(t) e^{j\omega_s t} \right)$$
$$i(t) = \sqrt{2} I(t) \cos(\omega_s t + \beta(t)) = \sqrt{2} \Re \left( \tilde{I}(t) e^{j\omega_s t} \right)$$

where $\tilde{V}(t)$ and $\tilde{I}(t)$ are a complex vector rotating at $\omega_s$ speed and $j = \sqrt{-1}$, the complex number. Those complex vectors are called phasors and are definite as:

$$\tilde{V}(t) = V(t) e^{j\alpha(t)} = V_x(t) + j V_y(t);$$
$$\tilde{I}(t) = I(t) e^{j\beta(t)} = I_x(t) + j I_y(t).$$

Figure 2.8: Time-varying phasor in the complex plane

In figure 2.8, the $x$, $y$ axes rotate at a constant speed $\omega_s$. This figure illustrates the voltage $v(t)$ which is the projection of the phasor $\tilde{V}(t) e^{j\omega_s t}$ on the real axis $\Re$. In this way, if $\tilde{V}(t)$ does not depend on time, the...
signal \( v(t) \) retrieved is just a cosine with constant frequency and amplitude since the maximum positive is reached at \( t = 0 \) (or in general when \( \omega_s t = 2\pi \)). Indeed, for this particular case, \( \hat{V} e^{j\omega_s t} \) vector and \( \mathcal{R} \) axis is superposed. Now considering a time-varying amplitude, the signal can have a variable amplitude in time. The time-varying phase introduce a variable frequency of the signal, it allows to model shrinking and stretching of the cosine signal in time.

Starting from the equation,
\[
v_a = e_a - R_ei_a - L_e \frac{di_a}{dt},
\]
the phasor approximation can be applied to \( v_a \) and \( i_a \):
\[
\sqrt{2} \mathcal{R}[\hat{V}(t)e^{j\omega_s t}] = \sqrt{2} \mathcal{R}[\hat{E}e^{j\omega_s t} - R_e \sqrt{2} \mathcal{R}[\hat{I}(t)e^{j\omega_s t}] - L_e \frac{d}{dt}[\sqrt{2} \mathcal{R}[\hat{I}(t)e^{j\omega_s t}]]
\]
where the phasor \( \hat{E}e^{j\omega_s t} = E e^{j\theta_j} e^{j\omega_s t} \) is the phasor related to \( e_a \) in equation (2.4). At this stage, the previous equation is passed into the complex plane:
\[
\hat{V}(t)e^{j\omega_s t} = \hat{E}e^{j\omega_s t} - R_e \hat{I}(t)e^{j\omega_s t} - L_e \frac{d}{dt}[\hat{I}(t)e^{j\omega_s t}].
\]
Using the chain rule, it becomes:
\[
\hat{V}(t)e^{j\omega_s t} = \hat{E}e^{j\omega_s t} - R_e \hat{I}(t)e^{j\omega_s t} - L_e \frac{d}{dt}[\hat{I}(t)e^{j\omega_s t}] - L_e \hat{I}(t)j\omega_s e^{j\omega_s t}.
\]
As fast response to the network is considered, the derivative term \( \frac{d}{dt}[\hat{I}(t)] \) is neglected. Besides, dividing by \( e^{j\omega_s t} \), the final equation is:
\[
\hat{V}(t) = \hat{E} - R_e \hat{I}(t) - j\omega_s L_e \hat{I}(t);
\]
which is equivalent to:
\[
\begin{align*}
\text{Real part:} & \quad V_x = E_x - R_e I_x + \omega_s L_e I_y; \\
\text{Imaginary part:} & \quad V_y = E_y - R_e I_y - \omega_s L_e I_x.
\end{align*}
\]

### 2.5.2 Park transform of the three-phase balanced stator

Using the example of a three-phase balanced at constant amplitude and phase (see «Park transformation»), the park transformation with time-varying amplitude and phase can be obtained:
\[
\begin{bmatrix}
v_{ds,pu} \\
v_{qs,pu} \\
v_{os,pu}
\end{bmatrix} = \frac{1}{\sqrt{3} V_B} \mathcal{R}(\theta_s^0 + \omega_s t)
\begin{bmatrix}
v_a \\
v_b \\
v_c
\end{bmatrix}
= \frac{1}{\sqrt{3} V_B}
\begin{bmatrix}
\sqrt{3} V(t) \cos(\theta_s^0 - \alpha(t)) \\
\sqrt{3} V(t) \sin(\theta_s^0 - \alpha(t)) \\
0
\end{bmatrix};
\]
\[
= \begin{bmatrix}
V_{pu}(t) \cos(\theta_s^0 - \alpha(t)) \\
V_{pu}(t) \sin(\theta_s^0 - \alpha(t)) \\
0
\end{bmatrix}.
\]
Finally, noting that \( v_a = \mathcal{R}(V(t)e^{j\alpha(t)} e^{j\omega_s t}) \) and \( \theta_s^0 = \frac{\pi}{3} \) (to have the q axis initially along the a axis), the xy axes are aligned with the \( qd \) axes. Indeed they rotate at the same speed even if they are located in a different plane (the \( dq \) axes are in the real 2D plane while xy axes are in the complex plane).
The equation (2.97) can be rewritten in per unit as:

\[ v_{ds} = V(t) \cos\left(\frac{\pi}{2} - \alpha(t)\right) = V(t) \sin(\alpha(t)) = \mathcal{J}(V(t)e^{j\alpha(t)}) = V_y(t) \]  
(2.98)

\[ v_{qs} = V(t) \sin\left(\frac{\pi}{2} - \alpha(t)\right) = V(t) \cos(\alpha(t)) = \mathcal{R}(V(t)e^{j\alpha(t)}) = V_x(t) \]  
(2.99)

The same observation can be deduced for the current:

\[ i_{ds} = I(t) \cos(\frac{\pi}{2} - \beta(t)) = V(t) \sin(\beta(t)) = \mathcal{J}(I(t)e^{j\beta(t)}) = I_y(t) \]  
(2.100)

\[ i_{qs} = I(t) \sin(\frac{\pi}{2} - \beta(t)) = V(t) \cos(\beta(t)) = \mathcal{R}(I(t)e^{j\beta(t)}) = I_x(t) \]  
(2.101)

The main advantage of a balanced three-phase assumption is that the three-phase supply is just model by its time-varying magnitude and phase. To retrieve the real currents and voltages of another phase in the stator windings, a simple phase shift of 120° is sufficient.

### 2.5.3 Stator winding equations

The stator equations projected to the Park plane are as a reminder:

\[ v_{ds} = R_s i_{ds} + \omega_s \psi_{qs} + \frac{d\psi_{ds}}{dt} t_B; \]  
(2.102)

\[ v_{qs} = R_s i_{qs} - \omega_s \psi_{ds} + \frac{d\psi_{qs}}{dt} t_B; \]  
(2.103)

where \( \frac{d\psi_s}{dt} \) are negligible. Thus, those equations become:

\[ v_{ds} = R_s i_{ds} + \omega_s \psi_{qs}; \]  
(2.104)

\[ v_{qs} = R_s i_{qs} - \omega_s \psi_{ds}. \]  
(2.105)
2.6 Final system using Park and phasor approximation

The final system using Park and phasor approximation is the following.

The source voltage equations are:

\[ v_{qs} = E_x - R_e i_{qs} + \omega_s L_e i_{ds}; \quad (2.106) \]
\[ v_{ds} = E_y - R_e i_{ds} - \omega_s L_e i_{qs}. \quad (2.107) \]

The voltage equations at the stator and the rotor Park windings are:

\[ v_{ds} = R_s i_{ds} + \omega_s \psi_{qs}; \quad (2.108) \]
\[ v_{qs} = R_s i_{qs} - \omega_s \psi_{ds}; \quad (2.109) \]
\[ 0 = R_r i_{dr} + (\omega_s - \omega_r) \psi_{qr} + \frac{d\psi_{dr}}{dt} i_{ds}; \quad (2.110) \]
\[ 0 = R_r i_{qr} - (\omega_s - \omega_r) \psi_{dr} + \frac{d\psi_{qr}}{dt} i_{ds}; \quad (2.111) \]

The flux equations are:

\[ \psi_{ds} = L_{ss} i_{ds} + L_{sr} i_{dr}; \quad (2.112) \]
\[ \psi_{qs} = L_{ss} i_{qs} + L_{sr} i_{qr}; \quad (2.113) \]
\[ \psi_{dr} = L_{sr} i_{ds} + L_{rr} i_{dr}; \quad (2.114) \]
\[ \psi_{qr} = L_{sr} i_{qs} + L_{rr} i_{qr}. \quad (2.115) \]

The rotor motion equation is:

\[ 2H \frac{d\omega_r}{dt} = T_e - T_m; \quad (2.116) \]
\[ T_e = \psi_{dr} i_{qr} - \psi_{qr} i_{dr}; \quad (2.117) \]
\[ T_m = T_m0 (A\omega_r^2 + B\omega_r + C). \quad (2.118) \]

2.7 Induction machine at steady state

By solving the system of equations (equation (2.106) to (2.118)) at steady-state, the electrical torque can be deduced. As a reminder, «steady-state» means that all time derivative terms are null. Since 13 equations are in presence, a maximum of 13 unknowns are admitted. Those unknowns are \( \psi_{dr}, \psi_{qr}, \omega_r, v_{ds}, v_{qs}, i_{ds}, i_{qs}, i_{dr}, i_{qr}, \psi_{ds}, \psi_{qs}, T_e, T_m \). The torque at steady-state is thus given by:

\[ N_T e = L_{ss}^2 R_f (E_x^2 + E_y^2); \quad (2.119) \]
\[ A_T e = (L_{rr} L_{ss} - L_{sr}^2) + L_{ss} \omega_s L_e)^2 + L_{rr}^2 (R_e + R_s)^2; \quad (2.120) \]
\[ B_T e = 2L_{sr}^2 R_f (R_s + R_e); \quad (2.121) \]
\[ C_T e = R_f^2 ((R_e + R_s)^2 + (\omega_s L_{ss} + \omega_s L_e)^2); \quad (2.122) \]
\[ T_e = \frac{N_T e (\omega_s - \omega_r)}{(\omega_s - \omega_r)(A_T e (\omega_s - \omega_r) + B_T e \omega_s) + C_T e}. \quad (2.123) \]

With this formula, influence of stator frequency, voltage and rotor resistance can be observed. Those parameters can be changed in order to regulate the speed.
In figure 2.10, increasing the rotor resistance implies a spread of the function along the rotor phase speed $\omega_r$. That is very useful to start the motor. Indeed, assuming a constant mechanical torque of 1 pu, if $R_r = 0.05$, the equilibrium is located at the intersection of $T_m$ and $T_e$. At the start, the mechanical torque is greater than the electrical motor torque and so, the motor cannot start. When increasing $R_r$ above 0.15, the electrical motor torque is greater at the start ($\omega_r = 0$) than the mechanical torque and the rotor accelerates until $T_e = T_m$. This functioning point is located at a higher speed than the speed $\omega_{r,\text{max}}$ corresponding to the maximum electrical torque.

In practice, a second squirrel cage is often added with high resistance to improve the start-up of induction machine. Sometimes, this cage is not present but the machine, itself, can serve this purpose since the mechanical tree links to the rotor. That phenomenon is illustrated in figure 2.11. The point «0.» is an unstable equilibrium state. Indeed, assuming the only presence of the inner cage (with small resistance), if the rotor is a bit accelerated, then, the motor torque is greater than the mechanical load and it accelerates along the blue curve until reaching point «2.». Similarly, if a small perturbation slows down the rotor from point «0.», it also leads to leave the equilibrium point to decelerate the rotor since mechanical load is greater than motor torque. The points «1.», «2.» and «3.» are stable points: if a small perturbation occurs, the state retrieved after a certain amount of time, is the same equilibrium point. Using only the inner cage (with small resistance), the motor cannot start: at $\omega_r = 0$, the motor torque is lower than the mechanical load. And using only the outer cage, the electrical machine is able to start but it reaches a rotor phase speed quite far from the stator frequency $\omega_s = 0.8$ pu. Having 2 cages allow both the start-up and a tighter approach of the stator frequency (equilibrium point «3.»).
2.7. INDUCTION MACHINE AT STEADY STATE

When looking at the frequency influence and the voltage influence on the electrical motor torque (figure 2.10), it is obvious that the best way to change significantly the rotor speed is to use a time-varying stator frequency. Increasing the frequency allows to shift the operating area\(^2\) to higher rotor phase speed. Unfortunately, increasing the frequency also implies a decrease of the maximum torque. If the maximum torque becomes lower than the mechanical load, the rotor speed will decrease until it stops. It is why, it is interesting to increase the stator voltage to keep a constant maximum torque. The maximum torque and the corresponding rotor phase speed is given by:

\[
T_{e,\max} = \frac{N_T e}{2\sqrt{A_T e C_T e + B_T e \omega_s}}; \quad (2.124)
\]

\[
\omega_{e,T_{\max}} = \omega_s - \frac{C_T e}{A_T e}; \quad (2.125)
\]

where \(T_{e,\max}\) is independent of the rotor resistance \(R_r\) and \(\omega_{e,T_{\max}}\), independent of stator voltages.

Besides, the equations (2.106) to (2.115) can be interpreted as a phasor circuit:

\[
v_{qs} = E_x - R_e i_{qs} + \omega_s L_e i_{ds}; \quad (2.126)
\]

\[
v_{ds} = E_y - R_e i_{ds} - \omega_s L_e i_{qs}; \quad (2.127)
\]

\[
v_{qs} = R_s i_{qs} - \omega_s (L_{ss} i_{ds} + L_{sr} i_{dr}); \quad (2.128)
\]

\[
v_{ds} = R_s i_{ds} + \omega_s (L_{ss} i_{qs} + L_{sr} i_{qr}); \quad (2.129)
\]

\[
0 = \frac{\omega_s}{\omega_s - \omega_r} R_r i_{qr} - \omega_s (L_{sr} i_{ds} + L_{rr} i_{dr}); \quad (2.130)
\]

\[
0 = \frac{\omega_s}{\omega_s - \omega_r} R_r i_{dr} + \omega_s (L_{sr} i_{qs} + L_{rr} i_{qr}). \quad (2.131)
\]
In phasor, it gives:

\[
\bar{V}_s = \bar{E} - R_e \bar{I}_s - j \omega_s L_e \bar{I}_s; \\
\bar{V}_s = R_s \bar{I}_s + j \omega_s (L_{ss} \bar{I}_s + L_{sr} \bar{I}_r); \\
0 = \frac{\omega_s}{\omega_s - \omega_r} R_t \bar{I}_r + j \omega_s (L_{sr} \bar{I}_r + L_{rr} \bar{I}_r);
\]

which can be modelled as a phasor circuit such as shown in Figure 2.12.

![Figure 2.12: Phasor circuit of an induction machine at steady-state.](image)

### 2.8 Work performed in Chapter 2

The model of the induction machine is finally established. Even, its steady-state relationships are obtained and in adequacy with the literature. It remains yet to find a solver to solve those equations. Then, to verify the solver, one will introduce a scale variation of the inputs to see if the operating steady-points of the machine are well those expected by the steady-state relationships. Afterwards, speed regulations will be proposed and performed. A sensitivity analysis will also be performed. To conclude, the comparison between both regulations will be done, future work will be introduced and advices about this work will be given.
In this chapter, two ways to solve the dynamic system are introduced and compared. A comparison between 2 softwares (Scilab and Matlab) is also performed. The system is a differential-algebraic system of equations (DAEs). It has the following form:

\[
\frac{dx(t)}{dt} = f(x(t), y(t), t); \quad (3.1)
\]

\[
0 = g(x(t), y(t), t). \quad (3.2)
\]

And in Matlab, itself, two ways exist to solve the system. The first way to solve it, is to discrete the equation using the trapezoidal and the Newton methods on a Matlab script. A second way is to use the simulink software in Matlab, that interpret the logic of block diagram into c language. The main disadvantage of simulink is that a Matlab simulink license is needed and its cost is very expensive for companies.

### 3.1 Discretisation algorithm

The equation (3.1) corresponds to this system:

\[
\frac{d\psi_{dr}}{dt} = (-\omega_B) \left[ R_r i_{dr} + (\omega_s - \omega_r) \psi_{qr} \right]; \quad (3.3)
\]

\[
\frac{d\psi_{qr}}{dt} = (-\omega_B) \left[ R_r i_{qr} - (\omega_s - \omega_r) \psi_{dr} \right]; \quad (3.4)
\]

\[
\frac{d\omega_r}{dt} = \left( \frac{1}{2H} \right) \left[ \psi_{dr} i_{qr} - \psi_{qr} i_{dr} - T_{mo} (A\omega_r^2 + B\omega_r + C) \right]; \quad (3.5)
\]

and the equation (3.2) represents the remaining equations:

\[
0 = -v_{qs} + E_x - R_e i_{qs} + \omega_s L_e i_{ds}; \quad (3.6)
\]

\[
0 = -v_{ds} + E_y - R_e i_{ds} - \omega_s L_e i_{qs}; \quad (3.7)
\]

\[
0 = -v_{ds} + R_s i_{ds} + \omega_s \psi_{qs}; \quad (3.8)
\]
0 = −v_{qs} + R_s i_{qs} − \omega_s \psi_{ds}; \quad (3.9)
0 = −\psi_{ds} + L_{ss} i_{ds} + L_{sr} i_{dr}; \quad (3.10)
0 = −\psi_{qs} + L_{ss} i_{qs} + L_{sr} i_{qr}; \quad (3.11)
0 = −\psi_{ds} + L_{sr} i_{ds} + L_{rr} i_{dr}; \quad (3.12)
0 = −\psi_{qr} + L_{sr} i_{qs} + L_{rr} i_{qr}. \quad (3.13)

The time-varying unknowns are:

\[
x = \begin{bmatrix} \psi_{dr} & \psi_{qr} & \omega_r \end{bmatrix}^T; \quad (3.14)
\]

\[
y = \begin{bmatrix} v_{ds} & v_{qs} & i_{ds} & i_{qs} & i_{dr} & i_{qr} & \psi_{ds} & \psi_{qs} \end{bmatrix}. \quad (3.15)
\]

To be complete, a time-varying parameter vector containing all characteristics of the induction machine is defined:

\[
u(t) = [R_s \quad R_r \quad R_e \quad L_{ss} \quad L_{sr} \quad L_{rr} \quad L_e \quad \omega_s \quad H \quad T_{mo} \quad A \quad B \quad C \quad e_x \quad e_y \quad \omega_B]. \quad (3.16)
\]

The following of this section is composed of 2 parts. The first part concerns the initial conditions while the second part talks about the dynamic system resolution.

### 3.1.1 Initial conditions

As explained previously, if the Torque-speed curve has a low motor torque at \( \omega_r = 0 \), the motor cannot start. But it may be interesting to look at the behaviour of the machine at steady-state when a perturbation occurs. For this purpose, it is useful to solve the non-linear system at steady-state using the Newton method and a guess state vector \( x_{\text{guess}} \) and \( y_{\text{guess}} \). The phase voltage is fixed to have a zero phase at the initial condition for \( \bar{V}_s \). It implies thus:

\[
v_{qs} = 1; \quad (3.17)
\]

\[
v_{ds} = 0. \quad (3.18)
\]

The equations (3.6) and (3.7) are given up and the time derivatives are replaced by zero. It remains 9 equations with 9 unknowns, \( X_I \), and 15 parameters, \( u_I \):

\[
X_I = \begin{bmatrix} \psi_{dr} & \psi_{qr} & \omega_r & i_{ds} & i_{qs} & i_{dr} & i_{qr} & \psi_{ds} & \psi_{qs} \end{bmatrix}; \quad (3.19)
\]

\[
u_I = [R_s \quad R_r \quad R_e \quad L_{ss} \quad L_{sr} \quad L_{rr} \quad L_e \quad \omega_s \quad H \quad T_{mo} \quad A \quad B \quad C \quad v_{ds} \quad v_{qs}]. \quad (3.20)
\]

The system is of the form:

\[
F_I(X_I) = 0; \quad (3.21)
\]
and is detailed in the Matlab code in figure 3.1.

![Image](https://example.com/image.png)

Figure 3.1: Matlab function that computes $F_I$ for a given state $X_I$.

The Newton method to solve a non-linear system is given by the following formula that is repeated until convergence:

$$X_{I,k+1} = X_{I,k} - J^{-1}(X_{I,k}) F_I(X_{I,k});$$  
(3.22)

where $J$ is the jacobian matrix. The element $\langle i, j \rangle$ of the jacobian matrix is:

$$[J(X_{I,k})]_{ij} = \frac{\partial F_i(X_{I,k})}{\partial x_j}$$
(3.23)

As the inverse of a matrix is not easy to compute, a LU decomposition is performed to solve the system:

$$J(X_{I,k}) \Delta_{k+1,k} = -F_I(X_{I,k});$$
(3.24)

$$X_{I,k+1} = X_{I,k} + \Delta_{k+1,k}. $$
(3.25)

In Matlab, the LU decomposition is used like this: $\Delta_{k+1,k} = J(X_{I,k}) \backslash \{-F_I(X_{I,k})\}$. The program iterates until the absolute value $F_I(X_{I,k})$ is lower than a tolerance value. This value has been set to $1e-15$. At the end, the grid voltage can be computed to complement the parameter vector $u(t)$:

$$e_x = v_{qs} + R_e i_{qs} - \omega_s L_e i_{ds};$$
(3.26)

$$e_y = v_{ds} + R_e i_{ds} + \omega_s L_e i_{qs};$$
(3.27)

and furthermore, the frequency basis, $\omega_B$, is set to 60 Hz.

### 3.1.2 Method to iterate in time

To obtain the dynamic evolution of the system, the trapezoidal rule is applied to the first 3 equations and the last 8 serves to complement the number of equations necessary to find the 11 unknowns at time $t+1$.
with a time step $h$:

$$\dot{x} = f(x, y, u) \Rightarrow x_{t+1} = x_t + \frac{h}{2} \left( f(x_{t+1}, y_{t+1}, u_{t+1}) + f(x_t, y_t, u_t) \right); \quad (3.28)$$

$$0 = g(x, y, u) \Rightarrow 0 = g(x_{t+1}, y_{t+1}, u_{t+1}). \quad (3.29)$$

---

**Figure 3.3:** Description of the algorithm used to solve the differential-algebraic equations.

Finally, the system is rewritten as 11 equations without any time-derivatives and the Newton method
can be used to solve the system at each time step:

\[
\begin{align*}
0_{1 \times 3} &= f(x_{t+h}, y_{t+h}, u_{t+h}) - \frac{2}{h} x_t + \frac{7}{h^2} x_t + f(x_t, y_t, u_t) \\
0_{1 \times 8} &= g(x_{t+h}, y_{t+h}, u_{t+h})
\end{align*}
\]

\[\begin{bmatrix}
0_{1 \times 3} \\
0_{1 \times 8}
\end{bmatrix} = \begin{bmatrix}
F(x_{t+h}, y_{t+h}, u_{t+h})
\end{bmatrix}
\] (3.30)

with \(x_{t+h}\) and \(y_{t+h}\), the 11 unknowns computed. The whole algorithm to solve the system is presented in Figure 3.3. \(T_{Sim}\) stands for the simulation time, \(\Delta t+h,t\) stands for the states increment and \(\frac{DF}{DX}\) stands for the Jacobian matrix of \(F\) with respect to \([x_{t+h} \ y_{t+h}]\).

### 3.2 Simulink

Simulink is a software of system modelling developed by MathWorks. It is based on graphical block diagram established by the user to model the dynamics of a physical system. This graphical language is then compiled and solve by Matlab internal routines.

This way to program is very intuitive and allows a good understanding of the model. In Figure 3.4, a subsystem is shown. Simulink allows to create subsystems to keep legible global system. The corresponding global system that uses this block is illustrated in Figure 3.11.

**Grid Equation**

**Electrical torque**

![Figure 3.4: Simulink inside the grid equation block](image)

![Figure 3.5: Simulink user-defined function block](image)

In this global system, several subsystem blocks are present. The first one «Stator Voltage with Thevenin grid» models the grid equations (equations (3.6) and (3.7)). The second one «Flux at Stator windings» models the stator windings voltage-flux relationship given by equation (3.8) and (3.9). The «Inverse Flux equation» block describes the current-flux relationship given by equations (3.10) to (3.13). Another kind of block is the Matlab function block (figure 3.5). This block allows to compute outputs thanks to inputs of the block using a Matlab function script. The integrator block is also used to make a loop system where \(x_0\) is the initial value of the state. The scope block displays the state evolution during time. The mux block allows to merge several states for instance to display them on the same scope. The from-file block imports a previously saved timeseries variable.
Concerning solvers, Simulink can work in two different modes: the continuous mode or the discrete mode. The discrete mode is especially used for embedded applications. And it is why, it is used to simulate the induction machine driving the compressor. In Figure 3.10, a block diagram coming from Mathworks is presented illustrating the different solver available for the two specific modes. It also speaks about variable time step or fixed time steps. Basically, variable time step is based on line search methods that allow to vary the time step during the simulation in order to gain computation time. For the discrete mode, three solvers exist: Backward Euler, Forward Euler and Trapezoidal. The Forward Euler is an explicit method while the two others are implicit. The advantage of implicit methods is that it is more stable than the explicit methods. An explicit method computes the next states of the system thanks to a function that only depends on the current states:

$$x(t + h) = F(x(t)); ~ (3.31)$$

while an implicit method must solve an equation to find the next states:

$$x(t + h) = G(x(t), x(t+h)). ~ (3.32)$$

These equations can be non-linear and root-finding algorithms (like the Newton's method) are used to solve them.

Here is the three methods:

- **Forward Euler:**
  $$x(t + h) = x(t) + hf(x(t))$$

- **Backward Euler:**
  $$x(t + h) = x(t) + hf(x(t+h))$$

- **Trapezoidal:**
  $$x(t + h) = x(t) + \frac{h}{2} \left( f(x(t+h)) + f(x(t)) \right)$$

It is quite evident that the implicit methods cost more computation time than the forward Euler.
Asynchronous machine

Open-loop Torque scale function

Figure 3.11: Global system example to illustrate Simulink.
3.3 Simulink - script: comparison

A comparison is performed in Table 3.1. Simulink is a block diagram interpreter while the Matlab script consists of a time loop simulation. Even if Simulink is an easier way to understand what is performed by the simulation, it is also more difficult to deal with. Indeed, debug can really become a nightmare in Simulink while in Matlab script, it is easier. That mainly comes from the fact that for the Matlab script each instruction is performed line by line until errors occur while Simulink block diagram is interpreted and translate in c language, it is henceforth difficult to access to the states of the system to identify the possible mistakes. Besides, if one wants to perform certain task instead of other depending on a condition, it is really easy to implement it in a script file while in Simulink, it is more difficult (using switch and logic circuits). Finally, as concerned results display, it is quite easily done in Simulink with the «Scope block». However, as soon as a graphic showing one state with respect to another state is wanted, Simulink has poor block settings in «XY graph»: no change in title or axis label is feasible.

<table>
<thead>
<tr>
<th></th>
<th>Simulink</th>
<th>Matlab script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphical presentation and understanding</td>
<td>+ easy understandable by peers</td>
<td>− a time loop more difficult to understand at first sight</td>
</tr>
<tr>
<td>Error analyse and debug</td>
<td>− difficult to understand from where errors come</td>
<td>+ easier debug by displaying values during the simulation</td>
</tr>
<tr>
<td>Adding a condition loop</td>
<td>− more difficult as it impacts the whole block diagram system</td>
<td>+ a simple if loop</td>
</tr>
<tr>
<td>Result display</td>
<td>+ simply add a scope block but need to rerun the simulation if the scope block was not there</td>
<td>− need a lot of instruction to have a nice graphic (plot,xlabel,ylabell,...)</td>
</tr>
<tr>
<td>State-state graphic</td>
<td>− displaying one state with respect to another is possible but the graphic axes are not easily labelled</td>
<td>+ the graphics keep the same configuration and nice display can be performed</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison between Simulink and Matlab scripts

This comparison is subjective and is mainly based on my small personal experience of 2 years on the Simulink tool and an experience of 5 years on Matlab scripts.

3.4 Matlab - Scilab: comparison

In Table 3.2, a brief comparison of two software is presented: Scilab and Matlab.
### Table 3.2: Comparison between Matlab and Scilab

<table>
<thead>
<tr>
<th>Matlab</th>
<th>Scilab</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High price;</td>
<td>• Open and free;</td>
</tr>
<tr>
<td>• Quality ensured;</td>
<td>• Medium quality;</td>
</tr>
<tr>
<td>• Constancy of the quality.</td>
<td>• No constancy of the quality, at all.</td>
</tr>
</tbody>
</table>

Matlab is a numerical computing software that provides a powerful development environment for scientific applications and engineering. It is sufficiently efficient for the application targeted. But MathWorks has the monopoly of this kind of user-friendly programming software and uses it to ask expensive licence to companies which use their software. Scilab is a free open-source software that is basically a free copy of Matlab with fewer features. The programming language is really similar and it also has a tool called «xcos» that is the equivalent of Simulink on Matlab. So the advantage of this software is purely the cost. However, it has a big problem. The constancy of the quality is not ensured at all. For instance, taking the Scilab software version 6.0.0, «xcos» software can create file equivalent to Simulink model. But after saving, closing the created model and re-opening it, the content of the file has disappeared and the run of the model is not enable (display an error) while before the saving, the model ran without errors. The latest version of Scilab is 6.0.1 which seems to run better. Indeed these bugs were solved. Nevertheless using a software that has such a variant quality is not viable. It is why the work is performed in Matlab.

### 3.5 Simulation

This section serves to understand the dynamic behaviour of the induction machine. The good programming of the equations is confirmed by the comparison between Simulink and the discretisation algorithm using trapezoidal rule and Newton’s method, and by the good behaviour of the machine to external variations. Several simulations are proposed here in order to validate that the model behaves as expected when encountering some variations of physical quantities such as:

- the load torque $T_m$ varying via the parameter $T_{mo}(t)$;
- the rotor resistance $R_r(t)$;
- the supply voltage.

All simulations use a time step $h = 0.01\text{s}$ which is sufficiently small to ensure accuracy. Besides, a way to find the initial conditions of the system at a given speed is also analysed. That method serves to initialise the system at a steady-state before performing the variation of the previously evoked physical quantities.

#### 3.5.1 Initial condition simulation

For the initial condition (IC), the interest of this algorithm is to find the electrical states (currents, voltages, magnetic fluxes) for a given rotor speed. For this test, the guess vector has been changed several times and the initial conditions ($\omega_r; T_e = \psi_{d1} i_{qr} - \psi_{qr} i_{d1}$) is observed. The initial guess vector is given by:

$$X_{guess} = \omega_{r,guess}^T 1 \times 9.$$  \hspace{1cm} (3.33)
The parameter vector used is:

\[ u_I = [R_s, R_f, R_e, L_{ss}, L_{sf}, L_{lr}, L_e, \omega_s, H, T_m, A, B, C, v_{ds}, v_{qs}]; \]

\[ u_I = [0.05, 0.02, 0.001, 2.09, 2, 2.15, 0.001, 1, 1.5, 0.6, 0.5, 1.5, 0.2, 0, 1]. \]

The algorithm to find steady-state initial condition is presented in Figure 3.2. At this stage, 4 values were tested for \( \omega_{r,guess} \): 0.3, 0.4, 0.5, 0.8 pu.

In figures 3.12 to 3.15, the steady-state electrical torque and the mechanical load torque is shown. The yellow circle is the initial guess while the purple one stands for the initial steady-state condition found. As matter of steady-state, it means that the initial condition found must necessarily be located on an intersection of the \( T_e \) curve with the \( T_m \) curve. It is seen that depending on the guess \( \omega_{r,guess} \), different operating points are reached. In Figure (3.15), the purple circle is not visible. It is located at \( \omega_r = -2.9533 \) with \( T_e = T_m = 0.0787 \). Indeed, mathematically, it is possible since the mechanical load is a parabola that intersects with 4 points the electrical torque curve. Yet, physically, there is no interest in this operating point.

Figure 3.12: Initial conditions for \( \omega_{r,guess} = 0.8 \)

Figure 3.13: Initial conditions for \( \omega_{r,guess} = 0.5 \)

Figure 3.14: Initial conditions for \( \omega_{r,guess} = 0.4 \)

Figure 3.15: Initial conditions for \( \omega_{r,guess} = 0.3 \)
After testing lots of guess values (figure 3.16), four different speeds $\omega_r$ are available: -2.953, 0.312, 0.711, 0.9616 pu. Yet, for a $\omega_{r,\text{guess}}$ in the range [0.15;0.3], the initial speed retrieved is far from the guess: -2.953. To conclude this test, the initial condition speed does not have an intuitive well-defined relationship with the guess. And the guess must not necessarily be near to the wanted initial condition. In further simulations, if the initial condition must be evaluated, it is thus mandatory to check the initial conditions retrieved to see if it is well the wanted initial speed.

### 3.5.2 Scale torque $T_{mo}(t)$ simulation

In this subsection, the induction machine is placed in steady-state condition at its operating speed (0.9543 pu: the intersection between load torque curve and electric torque curve) thanks to the initial conditions computation introduced in the previous subsection. Then, the mechanical load exerted by the
compressor is changed by opening or closing valves:

\[ T_m = T_{mo} \left( A\omega^2 + B\omega + C \right) \]  \hspace{1cm} (3.34)

the closing corresponds to an increase of \( T_{mo} \). Physically, the closing of the valves leads to an increase in pressure and thus more friction on the compressor which means the braking torque increases. The simulation here has no physical meaning in particular. The braking torque applied is much bigger than the operating range of the compressor torque. So, the practical braking torque will be much smaller than the one which is imposed at this stage in the model. It is done to see if the model follows the theoretic torque-speed curve of the induction machine well. Here is the applied variation to the model:

Figure 3.18: Block diagram of the scale torque \( T_{mo} \) simulation

The parameter vector used is:

\[ u_I = [R_s \; R_r \; R_e \; L_{ss} \; L_{sr} \; L_{ee} \; \omega_s \; H \; T_{mo} \; A \; B \; C \; v_{ds} \; v_{qs}] ; \]
\[ u_I = [0.05 \; 0.03 \; 0.001 \; 2.09 \; 2 \; 2.15 \; 0.001 \; 1 \; 1.5 \; T_{mo}(t) \; 2 \; 0 \; 0 \; 0 \; 1] ; \]
\[ f_B = 60 \text{ Hz}. \]
3.5. SIMULATION

Figure 3.21: Current in per unit

![Figure 3.21: Current in per unit](image1)

Figure 3.22: Torque speed curve

![Figure 3.22: Torque speed curve](image2)

In Figure 3.22, the mechanical torque curves are drawn in dashed lines for the different valve closure percentage ($T_{mo}$) and the mechanical load torque passes from one curve to the others according to the current value of $T_{mo}$. The full blue curve is the theoretic curve of the electric torque apply by the induction machine. This curve does not change during the simulation. And finally, the full red curve represents the electric torque applied during the simulation. Initially, the full red curve starts at a speed of 0.9543 at the intersection between dashed yellow and blue curves. After 75s, $T_{mo}$ changes from 0.6 to 1.1 pu. After around 5 seconds, the machine reaches a new steady-state that is a speed of 0.83 pu (intersection between the blue and the dashed purple curves) by following the electric torque curve (full blue line). After 150s, $T_{mo}$ becomes equal to 1.6 pu and the state goes to a new steady state which is the intersection between the dashed green and blue curves. At last, after 225s, $T_{mo}$ becomes equal to 2.1 pu and the state goes to a new steady state which is the intersection between the dashed light blue and blue curves. To conclude about this figure, the dynamic curve (full red line) follows well the steady-state theoretic curve and the stable steady-points given by the intersection between the theoretic electric torque and the theoretic mechanical torque, are well steady points for the dynamic torque. The model behaves as expected when the load torque is modified.

High peaks for the mechanical torque in Figure 3.20 comes from a modelling error and a numeric phenomenon. The first one is the fact that the opening/closing of valves is not continuous while in practice it should be continuous. At the moment where $T_{mo}$ changes, the rotor speed is the same as before change but the mechanical torque is suddenly increased (high peaks). After a certain elapsed time, the rotor speed has decreased and consequently, the mechanical torque also. The numeric phenomenon is that the overshoot decreases a bit when the time step decreases but it does not disappear completely (no figure is displayed here not to overload the thesis). Big variations cannot be reproduced without numerical errors for a time step of 0.01s.

The left part of the theoretic electric torque is the stall of the machine. When the mechanical torque increases due to $T_{mo}$, it results in a faster decreased of the rotor speed. In practice, the currents of the stator and the rotor increase with the load and above a limit around 1.5 pu, the currents are considered too much for the machine. Thenceforth, protective mechanisms are set on to stop the machine. So in practice, the machine will be chosen to remain in the right part of the theoretic curve. Hopefully with a 10 MW induction machine, the torque basis value of the per unit system is much greater than the torque operating range exerted by the compressor. The basis values are:
CHAPTER 3. NUMERICAL RESOLUTION

\[ S_B = 11.4 \ \text{MVAR}; \quad Z_B = 4.5 \ \Omega; \]
\[ V_B = 4160 \ \text{V}; \quad H = 1.69 \ \text{s}; \]
\[ f_B = 60 \ \text{Hz}; \quad T_B = 60.5 \ \text{kNm}; \]
\[ I_B = 913.8 \ \text{A}; \quad \omega_{m B} = 1800 \ \text{RPM}. \]

The torque and speed range in per unit corresponding to the scope section in chapter 1, are:

\[
\begin{align*}
&[0; 7466] \ \text{Nm} \quad \frac{1}{(T_BGR\eta)} \quad [0; 0.6997] \ \text{pu}; \quad (3.35) \\
&[925; 10206] \ \text{RPM} \quad \frac{GR/\omega_{m B}}{[0.0906; 1]} \ \text{pu}. \quad (3.36)
\end{align*}
\]

As shown in Figure 3.22, the maximum theoretic torque is about 1.53 pu. As given by equation 3.35, the dynamic torque will not reach the stall for this configuration of the machine (for the given \( R_r, R_s, L_s \) and all machine characteristics).

### 3.5.3 Scale rotor resistance \( R_r(t) \) simulation

The induction machine is now started from null states and the rotor resistance is changed during the simulation. In practice it is performed by plugging variable resistance to the wound rotor. As explained in Figure 2.10, increasing the rotor resistance spreads the electric torque curve keeping the same maximum torque. It leads to start the machine at the right part of the curve and to avoid starting in the stall area.

![Figure 3.23: Block diagram of the scale rotor resistance \( R_r \) simulation](image)

The parameter vector used is:

\[
\begin{align*}
u_I &= [R_s \quad R_r \quad R_e \quad L_{ss} \quad L_{sr} \quad L_{rr} \quad L_e \quad \omega_s \quad H \quad T_{mo} \quad A \quad B \quad C \quad v_{ds} \quad v_{qs}]; \\
u_I &= [0.05 \quad R_r(t) \quad 0.001 \quad 2.09 \quad 2 \quad 2.15 \quad 0.001 \quad 1 \quad 1.5 \quad 0.6 \quad 2 \quad 0 \quad 0 \quad 0 \quad 1]; \\
f_B &= 60 \ \text{Hz}.
\end{align*}
\]

The rotor resistance \( R_r(t) \) varies as a uniform scale function with those values in per unit: 10; 1; 0.5; 0.1; 0.03.
3.5. SIMULATION

In Figure 3.27, the dynamic torque-speed point evolves at each transition of the rotor resistance: the operating point passes from one theoretic electric torque curve to another. It goes with an overshoot of currents (in stator and rotor) and so torque (as torque is linked to currents). These overshoot is higher when it occurs at high speed. At high speed, small oscillations appear near 180s and 240s but they are quickly damped. The speed is low sensitive to the rotor resistance for large rotor resistance while it is highly sensitive at high speed and small rotor resistance.

Figure 3.24: Rotor speed in per unit

Figure 3.25: Mechanical and electrical torque in per unit

Figure 3.26: Current in per unit

Figure 3.27: Torque speed curve
One can also observe some oscillations before following the theoretic electric curve until intersection with the mechanical load torque curve. Nevertheless the oscillations at 60s and 120s are numeric and not physical. This can be observed by repeating the computation with a smaller time step $h = 0.001s$. It can be seen in Figure 3.28 and 3.29 where the electric torque is drawn during time for respectively $h = 0.01s$ and $h = 0.001s$. The large overshoot and the oscillations were thus uniquely numeric.

![Figure 3.30: Current in per unit](image)

![Figure 3.31: Torque speed curve](image)

However, it remains some overshoots of smaller peaks in the currents and torque. Overshoots are damped faster at high speed that at low speed. These overshoots are due to the fact that the rotor resistance changes too fast (which is not realistic). Besides, as expected, the currents increase when the rotor resistance decreases since the load torque increases.

### 3.6 Work performed in Chapter 3

The chapter 3 contains all information about solvers that are used and the first simulations that helps to validate the model of the induction machine. Now that the model is validated, the speed regulation of the machine will be discussed since the compressor has to be tested in iso-speed. To this aim, some regulations will be studied and a sensitivity analyses will be performed. To conclude, the three methods will be compared, future work will be proposed and advices in the use of this work will be given.
Regulation and control methods are developed in this chapter. They are used to control the speed of the asynchronous machine. Three methods are described. The first one is the rotor resistance control using a wound rotor. It consists of connecting variable resistance to the slip rings of the rotor to control the speed. The second one is a voltage and frequency control to vary the speed or to maintain a fixed speed with a time-varying mechanical load. It is better known as «scalar control». The last one is a field-oriented control («vector control»). It brings the control method very similar to that of the DC machine where a current is tuned to control the electromagnetic torque applied. For both methods (except the field-oriented control), three parts are given: an algorithm part which explains how the regulation is performed using the previously established model of the induction machine; a simulation part of the model and a practical part containing what kinds of devices are required to perform this regulation. Finally, the three methods are compared on both sides: algorithm, sensitivity and ease of implementation.

4.1 Rotor resistance control

4.1.1 Theoretical part: control algorithm

As explained for the Figure 2.10, increasing the rotor resistance results in a displacement of the intersection between the mechanical load torque curve and the electric torque curve to the left. Id est, the steady-state rotor speed decreases with an increase of resistance.

In Figure 4.1, the Simulink block-diagram of the regulation via rotor resistance is presented. The «Asynchronous Machine» block models the machine according to Chapter 2. The inputs of this block are the following: \( T_{mo}(t) \), the coefficient present in the mechanical load torque to represent the influence of thermodynamic inputs (opening/closing of valves); \( e_x \) and \( e_y \), the three-phase feed voltages projected on \( d, q \) axes; \( \omega_s(ws) \), the applied stator frequency; and \( R_r \), the rotor resistance.

Considering the thermodynamic parameter \( T_{mo} \), it is set initially at \( T_{mo_i} \) and after \( t = 150s \), it changes suddenly to \( T_{mo_p} \). Since it is not realistic to change in such way the thermodynamic parameter, a rate limiter is set to 0.05/s in absolute value which means that one can pass from 0.5 to 1.0 in 10s.

Then, the controller implemented is a PID controller. More sophisticated control algorithms exist but they are more complex and need more information about the system. Since the system is not yet built and that the project is at study level, there is no interest to complicate the control algorithm.
Asynchronous machine

PID control resistance

Figure 4.1: Block-diagram of the Simulink resistance control of an induction machine

The PID block takes an error $e$ as input and comes out a result $u$:

$$u(t) = k_p e(t) + k_i \int_0^t e(t) dt + k_d \frac{de(t)}{dt}.$$  \hspace{1cm} (4.1)

One advantage of the PID control is its simplicity. Furthermore, it does not depend on the system on which it is implemented. It does not need a model of the induction machine to control it. Basically, the controller tries to decrease the error to 0. As a brief reminder of the PID control, the three terms are quickly introduced:

- $k_p$: the proportional gain. Assuming $k_d = k_i = 0$ (i.e, only a proportional gain), when the reference speed is reached, the output is null, no correction applied to the input of the system. With the induction machine, the rotor resistance is null (or limited by the saturation block), it implies an acceleration, the error becomes positive and the rotor resistance increases (with $k_p > 0$) which implies a deceleration. Finally, the rotor speed oscillates around the reference speed without stabilising.

- $k_i$: the integrator gain. This parameter introduces memory (if $k_i \neq 0$) in the control. It allows to reach a null error with constant output control different from 0. At null error ($e = 0$), the proportional contribution is null but the integrator contribution is not and, furthermore, integrator contribution remains constant.

- $k_d$: the derivative gain. It allows to predict the error, to converge faster to the reference speed and to be more stable to noise in the system.

As concern the start-up, it can be performed by the controller using a moving reference:

$$\omega_{\text{ref}}(t) = \begin{cases} \frac{\omega_{\text{ref.end}}}{1 - \exp(-1/\tau)} \left(1 - e^{-\frac{t}{\tau_{\text{start}}}}\right) & \text{if } t < T_{\text{start}} \\ \omega_{\text{ref.end}} & \text{otherwise} \end{cases}$$  \hspace{1cm} (4.2)

where $\omega_{\text{ref.end}}$, the final reference to reach; $T_{\text{start}}$, the elapsed time before setting the final reference to the control system; $\tau$, a start parameter that makes the curve more abrupt when tending to zero. At the start-up, the electric torque must also exceed a static torque to move the compressor. And thus numerically, the mechanical torque is equal to the electric torque until the electric torque exceeds the static mechanical torque:

$$T_{\text{m,coulomb}}(t) = \begin{cases} T_e & \text{if } T_e < T_{\text{statique}} \text{ et } \omega_r = 0; \\ T_m & \text{otherwise.} \end{cases}$$  \hspace{1cm} (4.3)
A feedforward can also be set up to improve the control of the induction machine. It consists to add to the output of the PID, the rotor resistance computed from the feedforward. The feedforward block computes the rotor resistance that leads to reach the reference speed at steady-state by solving:

\[ T_e(R_r, \omega_{\text{ref}}) = T_m(\omega_{\text{ref}}) \]  

for \( R_r \). It is rewritten as following:

\[ \frac{N T_e R_r (\omega_s - \omega_{\text{ref}})}{(\omega_s - \omega_{\text{ref}})(A T_e (\omega_s - \omega_{\text{ref}}) + B T_e R_r \omega_s) + C T_e R_r^2} = T_{\text{mo}} \left( A \omega_{\text{ref}}^2 + B \omega_{\text{ref}} + C \right); \]

with

\[ N T_e = L_s^2 \left( E_x^2 + E_y^2 \right); \]
\[ A T_e = \left( L_{rr} L_{ss} - L_{sr}^2 + L_{rr} \omega_s L_e \right)^2 + L_{rr}^2 \left( R_e + R_s \right)^2; \]
\[ B T_e = 2 L_{sr}^2 \left( R_s + R_e \right); \]
\[ C T_e = \left( \left( R_e + R_s \right)^2 + \left( \omega_s L_{ss} + \omega_s L_e \right)^2 \right). \]

It can be rewritten as a second order function with respect to \( R_r \):

\[ a R_r^2 + b R_r + c = 0; \]

with

\[ a = C T_e; \]
\[ b = (\omega_s - \omega_{\text{ref}})B T_e \omega_s - N T_e (\omega_s - \omega_{\text{ref}}) \]
\[ \frac{1}{T_{\text{mo}}(A \omega_{\text{ref}}^2 + B \omega_{\text{ref}} + C)}; \]
\[ c = (\omega_s - \omega_{\text{ref}})^2 A T_e. \]

Then, if no roots are found, the feedforward block returns the current rotor resistance. If roots are found, the feedforward block returns the higher roots as the new rotor resistance. The reason why the higher is chosen is that the higher resistance ensured that the intersection between the load torque curve and the electric torque curve is on the right part (id est, at higher speed than the speed corresponding to the max torque, remaining in the motor mode) of the theoretic electric torque curve.

In the following subsections, several parameters of the model are varied in order to emphasise their impacts on the speed regulation of the induction machine:

- the PID controller \( k_p, k_i, k_d \);
- the start-up \( \tau \);
- the coulomb friction \( T_{\text{statique}} \);
- the feedforward;
- the noise in the grid supply \( e_s, u_s \);
- the induction parameter \( u(t) \) (Eq.(3.16));
- the load characteristics \( H, T_{\text{mo}}, A, B, C \).
For the following subsections, here are the default parameters chosen for the model:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_s$</td>
<td>0.05</td>
<td>$R_r$</td>
<td>25</td>
</tr>
<tr>
<td>$\omega_s$</td>
<td>1</td>
<td>$L_{ss}$</td>
<td>2.09</td>
</tr>
<tr>
<td>$L_{sr}$</td>
<td>2</td>
<td>$L_{rr}$</td>
<td>2.15</td>
</tr>
<tr>
<td>$R_e$</td>
<td>0.001</td>
<td>$L_e$</td>
<td>0.001</td>
</tr>
<tr>
<td>$e_x$</td>
<td>1</td>
<td>$e_y$</td>
<td>0</td>
</tr>
<tr>
<td>$T_{mo,i}$</td>
<td>0.5</td>
<td>$T_{mo,p}$</td>
<td>0.9</td>
</tr>
<tr>
<td>$A$</td>
<td>0.6297</td>
<td>$B$</td>
<td>0.07</td>
</tr>
<tr>
<td>$C$</td>
<td>0</td>
<td>$T_{statique}$</td>
<td>0</td>
</tr>
<tr>
<td>$T_{start}$</td>
<td>50s</td>
<td>$\tau$</td>
<td>0.4</td>
</tr>
<tr>
<td>$k_p$</td>
<td>150</td>
<td>$k_i$</td>
<td>60</td>
</tr>
<tr>
<td>$k_d$</td>
<td>0</td>
<td>$\omega_{ref}$</td>
<td>8000 RPM</td>
</tr>
<tr>
<td>$h$</td>
<td>1e-2</td>
<td>$feedforward$</td>
<td>0</td>
</tr>
<tr>
<td>Coulomb</td>
<td>0</td>
<td>$t_{perturb}$</td>
<td>150s</td>
</tr>
</tbody>
</table>

Table 4.1: Default parameters for rotor resistance command during the speed regulation of the induction machine

The corresponding Matlab script and Simulink model is attached in the Appendix F.

### 4.1.2 Tuning of the PID controller

The PID controller is tuned by choosing all gains back to back. The first varying parameter is the proportional gain $k_p$ keeping $k_i = k_d = 0$:

$$k_p = \{1, 30, 90, 150\}.$$  \hspace{1cm} (4.10)

As shown in Figure 4.3, the reference speed that should be reached is 8000 RPM and the start-up reference speed increases smoothly until 50s. Increasing the proportional gain decreases the static error. But even with a $k_p = 150$, a static error remains which implies that an integrator gain is needed. In Figure 4.2, the stator current increases when the proportional gain increases, which is logic. Indeed, the torque must increase to reach higher speed and consequently the current, since the current allows to create an electromagnetic torque.

![Figure 4.2: Stator current](image-url)
The second varying parameter is the integrator gain $k_i$ keeping $k_p = 150$ and $k_d = 0$: $k_i = \{-1, 0, 10, 30, 90\}$. It allows to remove the static error as shown in Figure 4.4. Besides, it appears that a negative integrator gain leads to a divergence. The same effect can be observed for a negative $k_p$. It is physically logic. When the error is positive, the speed is too high and the rotor resistance must be increased which means add a term proportional to the absolute value of $k_i$. It means $k_i$ and $k_p$ must be positive.

Figure 4.3: Compressor speed and compressor motor torque

Figure 4.4: Compressor speed and compressor motor torque
In Figure 4.5, the effect of the changing closure valve is visible around 150s. $T_{mo}$ changes from 0.5 to 0.9 at a limited rate of 0.05 per second. Thus at 158s, the final value for $T_{mo}$ is reached. Increasing $k_i$ allows to decrease the undershoot and to retrieve the reference speed faster. With a sufficiently high integrator gain, an overshoot in the torque appears. It is also the case for the current in the stator. To avoid overshoot in the torque, an integrator gain of 60 is chosen.
Finally, the derivative gain is varied: \( k_d = \{0, 1, 7\} \) with \( k_p = 150 \) and \( k_i = 60 \). The influence of the derivative gain is very poor as illustrated in Figure 4.7 (even after zooming, Figure 4.8). Indeed, all curves are superposed and quasi-identical. Furthermore, for \( k_d \geq 8 \), the simulation diverges. Consequently, there is no interest to implement a derivative gain and the control applied is a PI regulation \( (k_d = 0) \).

To conclude, the PID parameters are \( k_p = 150, k_i = 60, k_d = 0 \). For further simulations via rotor resistance command, those parameters are kept.

### 4.1.3 Start-up

The start-up is effectuated thanks to a time-varying reference, equation (4.2), applied to the PI controller. The \( \tau \) parameter is changed to allow faster start-up. The reference speeds for each \( \tau \) is represented in dashed grey line.

![Figure 4.9: Rotor speed and rotor resistance during time, start-up](image)

![Figure 4.10: Stator current and vertical zoom, start-up](image)
The more $\tau$ tends to 0, the more the machine starts quickly (Figure 4.9). But for $\tau$ too small, the reference speed changes too fast. The error with respect to the reference speed is higher. The output of the PI controller decreases and a rotor resistance lower than 0.01 is wanted. But that cannot be reached in practice (there is saturation, the lowest possible rotor resistance is the one of the induction machine itself).

The actuated rotor resistance is higher than the one given by the PI controller. The speed increases slower than the one that should be observed (if negative rotor resistance was allowed). It results in an area between the reference speed and the dynamic speed higher than expected and consequently, the integrator part $k_i \int_0^t e(t) \, dt$ is higher. Since the steady-state output of the PI (null error) depends only on the integrator contribution, the integrator contribution keeps growing during the saturation. When the error is now of the opposite sign, the rotor resistance remains unchanged because of the large negative integrator contribution. A large overshoot occurs to compensate the integrator contribution in such a way that the area between reference and dynamic curve from 12s to 50s is approximately the same as the area from 0s to 12s. This phenomena is called windup effect.

Many anti-windup systems exist to attenuate this overshoot (back calculation method). The basic idea of such a system is to change the output of the controller by adding a term to the error of the integrator contribution. This term is proportional to the difference between the actuated rotor resistance and the output of PI. Thus, here, the term to add is positive while the error is negative. The new error to integrate is thus higher and results in a lower integrator contribution and also a lower overshoot. But as the start-up is not really the matter of this application, no anti-windup has been set up.

In Figure 4.10, the stator current has an overshoot that increases when $\tau$ tends to 0. The current for $\tau = 0.01$ is unfeasible by the machine. It leads in practice to the use of protective mechanisms and the machine stops.

To conclude, the parameter $\tau$ is chosen to avoid saturation: $\tau = 0.4$.

4.1.4 Feedforward

A feedforward is added to the regulation (Figure 4.11). It impacts the PID controller and has the goal to improve the reaction to a perturbation (valve closure change). The function script is available in Appendix ??.

The feedforward is triggered when $\text{feedforward}$ is set to 1.

![Figure 4.11: Block diagram of the regulation via rotor resistance command using a feedforward](image)

To better observe the impact of the feedforward, $\text{feedforward}$ is varied between 0 and 1. The resulting speed as well as the electric torque at the perturbation around 150s is shown in Figure 4.12. When
Feedforward increases, the undershoot diminishes from 0.1% to 6e-5%. This constitutes a great improvement.

Figure 4.12: Compressor speed and motor torque, feedforward

The more feedforward is large the more the torque starts to increase at a higher rate. It results to decrease the torque overshoot. This explanation can be enforced by the behaviour of the rotor resistance that decreases faster for a larger feedforward (Figure 4.13). It is quite logic since the output of the PI is no more the rotor resistance but a smaller value that represents the increment or decrement of the steady-state rotor resistance. At the end the output of the PI controller is zero.

Figure 4.13: Actuated rotor resistance, feedforward
As a reminder, the torque overshoot happens at 158s while the valve closure starts to change at 150s, because the valve stops moving at 158s due to a speed limiter. The feedforward introduces a small undershoot in the actuated rotor resistance. Figure 4.14 represents the stator current in per unit at the end of $T_{mo}$ variation. It behaves similarly to the motor torque. Indeed, the overshoot decreases with the feedforward.

Yet, the feedforward has some implications that must be highlighted. It needs to have a measurement of the mechanical braking torque (in an explicit or implicit way).

4.1.5 Coulomb friction

In the «Asynchronous Machine» block, a block is added after the mechanical torque to implement the coulomb friction (remark: *feedforward* is kept equal to 1). This block is seen in Figure 4.15 and its content is shown in Figure 4.16.
In Figure 4.16, the unit delay blocks (UD1, UD2) are initialised to a true value. So, at the start-up ($T_e \leq T_{statique}$), the logic operator blocks (LO1, LO2) see true inputs and their outputs are true. The logic value coming in the «Switch» block is higher than 0 and the top input is connected to the output. The parameter $\text{coulomb}\_\text{static}$ allows to activate the coulomb friction phenomenon. When $\text{coulomb}\_\text{static}$ is set to zero, there is no coulomb friction in the model, unlike in the case where it is set to 1. As soon as the electric torque is higher than the static torque, the output of LO2 is zero. At the next time step, the output of LO1 is zero will remain zero which means output of LO2 equal to zero. The output of the «Switch» is connected to the bottom input and the mechanical torque apply to the model is the one depending on the rotor speed (equation (2.118)).

![Figure 4.17: Compressor speed and motor torque in RPM and kNm](image1)

![Figure 4.18: Zoom of the compressor speed and motor torque to see the details of start-up](image2)

The static torque shown in Figure 4.17 and 4.18, is expressed in per unit. The corresponding value in kNm are the following:

$$T_{statique} = [0.1493, 1.4932, 1.7918] \text{ kNm};$$

$$= [0.013993, 0.13993, 0.16792] T_R GR\eta.$$  \hspace{1cm} (4.11)

The impact of the coulomb friction is a larger increase in motor torque at start-up. This phenomenon is logic since the speed error is larger. The actuated rotor resistance is lower, the current increase faster and consequently the torque. As for speed, it is maintained to zero until the motor torque exceeds the static torque. For the yellow curve, the speed is zero until $t = 2.23s$ since the motor torque applied to the compressor is lower than 1.7918 kNm.

This Coulomb friction phenomenon complicates the model and it may lead to an error in Simulink. The software may not be able to solve the system. The solution to overcome it is to reduce the time step $h$ to 0.001 instead of 0.01 (for instance).
4.1.6 Grid supply noise

The grid supply is not perfect and some noises can appear in electric signals. These noises are modelled in Figure 4.19 as Gaussian random variable with zero mean and a specified variance. The electric quantities affected by the noise are the supply voltage $e_x$, $e_y$ and the stator frequency $\omega_s$ (corresponding to $w_s$ in Figure 4.19).

The supply voltage $e_x$ is influenced by noises of different standard deviations: $\sqrt{e_{x,\text{var}}} = \{0.1, 0.5, 1, 2\}$ % of 1 pu. The results of the simulation are displayed from Figure 4.20 to Figure 4.27. One observes a large influence of the noise applied to the voltage. Indeed, at only 2% for the standard deviation, the torque varies in a range of 800 Nm around its mean which is really bad for mechanical properties. Furthermore, changing the PID coefficients does not change anything. The standard deviation noise must be restricted to 0.5% of its nominal voltage. The supply voltage $e_y$ can also be noisy and same effects are observed. The feed frequency noise also leads to the propagation of the noise to the electric torque applied. At the end, the stator frequency noise must also be limited to a standard deviation of maximum 0.5%.

![Figure 4.19: Block diagram of a speed regulation via rotor resistance actuation, study of grid noises](image)
4.1.7 Induction machine parameter

The induction machine parameters are changed to study their influences on the machine. They are varied one after the other according to Table 4.2.
There is to note that when \( L_{sr} \) changes, the parameters \( L_{ss} \) and \( L_{rr} \) are also impacted keeping respectively \( L_{ss} - L_{sr} \) and \( L_{rr} - L_{sr} \) constants. The figures 4.28 to 4.33 represents the compressor speed and the influences of the induction parameters. Concerning the feedforward, it is performed using a fixed model of the induction machine: \( R_s = 0.05, L_{sr} = 2, L_{ss} = 2.09, L_{rr} = 2.15, R_e = 1e^{-3}, L_e = 1e^{-3} \). As expected the feedforward looses efficiency when the current induction machine parameters move away from those of the model.

The variations of the stator resistance and the cable parasitic resistance have few influenced on the speed dynamic response of the machine. Errors on the model (for those parameters) does not lead to high differences at a mechanic level (torque and speed). However, it is not the same for the inductances. It can lead to an overshoot or undershoot of 0.4 RPM.
4.1. Rotor Resistance Control

Finally, the other states are given in a table (Table 4.3) to avoid an overload of figures. So, the stator resistance mainly influences the rotor current and the rotor flux. The mutual inductance has large influence on the stator current and the rotor flux. The stator self inductance has an impact on the rotor current and the rotor flux.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Stator current</th>
<th>Rotor current</th>
<th>Stator flux</th>
<th>Rotor flux</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_s$</td>
<td>[0.6538, 0.6542]</td>
<td>[<strong>0.4198, 0.4417</strong>]</td>
<td>[0.9951, 0.9469]</td>
<td>[<strong>0.947, 0.9</strong>]</td>
</tr>
<tr>
<td>$R_e$</td>
<td>[0.6538, 0.6538]</td>
<td>[0.4269, 0.4308]</td>
<td>[0.9789, 0.9702]</td>
<td>[0.9313, 0.9228]</td>
</tr>
<tr>
<td>$L_e$</td>
<td>[0.6538, 0.6538]</td>
<td>[0.4269, 0.4315]</td>
<td>[0.9789, 0.9687]</td>
<td>[0.9314, 0.9214]</td>
</tr>
<tr>
<td>$L_{ss} - L_{sr}$</td>
<td>[0.6538, 0.6542]</td>
<td>[<strong>0.4226, 0.4412</strong>]</td>
<td>[0.9784, 0.9784]</td>
<td>[<strong>0.9409, 0.9012</strong>]</td>
</tr>
<tr>
<td>$L_{rr} - L_{sr}$</td>
<td>[0.6406, 0.6583]</td>
<td>[0.4255, 0.4278]</td>
<td>[0.9785, 0.9784]</td>
<td>[0.9343, 0.9293]</td>
</tr>
<tr>
<td>$L_{sr}$</td>
<td>[<strong>0.6936, 0.5015</strong>]</td>
<td>[0.4292, 0.418]</td>
<td>[0.9783, 0.9789]</td>
<td>[<strong>0.9262, 0.9511</strong>]</td>
</tr>
</tbody>
</table>

Table 4.3: Table of a range of state values (at ±1e-5) at 200s corresponding to the variation of the designed parameter in the range of Table 4.2

4.1.8 Compressor Mechanical Load Parameter

The mechanical load influence can be studied. Two parameters are chosen to be varied: $J_{comp}$ and $a$. The parameter $a$ is given in percentage in the following way:

$$T_m = T_{mo}(A\omega_r^2 + B\omega_r + C); \quad (4.13)$$

$$A = a \frac{7466}{T_B GR\eta}; \quad (4.14)$$

$$B = (1-a) \frac{7466}{T_B GR\eta}. \quad (4.15)$$

Concerning the compressor moment of inertia, those are the values that are set: [15, 20, 25, 30]kgm$^2$.

In Figure 4.34, the mechanical torque load diminishes when $a$ increases. Indeed, $\omega_r^2$ is below $\omega_r$ since $\omega_r$ is comprised in [0; 1] in per unit. When $a$ is equal to 100%, the squared term has a higher weight and thus, the torque decreases. Concerning the compressor speed, there is nearly no impacts on it between two different values of $a$. In Figure 4.35, the moment of inertia is changed. The regulation is hardly affected around 150-158s. This is due to the fact that the load torque does not depend on the moment of inertia (in this model) and the electric torque is also independent from the moment of inertia. Henceforth, the speed is only given by the PID and the feedforward that act to be closed to the reference speed. However, the torque at start-up of the machine is higher for a high moment of inertia. Since the model asks the machine to start
in a same elapsed time, the acceleration must be the same and consequently when the moment of inertia increases, the needed start torque increases.

![Figure 4.34: Influence of \( a \) on compressor speed and compressor torque](image)

![Figure 4.35: Influence of the moment of inertia of the compressor](image)

### 4.1.9 Comparison with Matlab script

In this subsection, the results of the simulink software and the matlab script are shown in Figure 4.36. As expected, the same evolution on the states of the machine and the compressor are obtained. Furthermore, looking at the torque-speed diagram, the dynamic curve follows nearly a vertical path to go from \( T_{mo} = 0.5 \) to
$T_{iso} = 0.9$. So the path between the two steady points is quasi-iso-speed. The parameters for the simulations are exactly those written in Table 4.1. Indeed, the feedforward has not been implemented in the Matlab script but it can easily be copied and pasted from Simulink.

![Torque](image1)

![Torque - speed](image2)

![Rotor resistance](image3)

![Rotor speed](image4)

Figure 4.36: Comparison between Matlab script using the discretisation algorithm explained in Chapter 3 and the Simulink software.

### 4.1.10 Practical implementation

To implement this regulation, here is a list of the main components needed:

- an induction machine of 10 MW with a quite accurate model;
- a wound rotor to be able to connect resistances;
- a torque-meter;
- an electronic converter AC-AC (example, matrix converter);
- a filter between the grid and the converter;
- a PLC (Programable Logic Controller) to control the induction machine;
- stator current measurements in the three phases;
- a tunable variable resistance.
### 4.2 Scalar control

#### 4.2.1 Theoretical part: control algorithm

The scalar control of the induction machine is used to reduce the power losses. Indeed, the previous use of a rotor resistance command took place with a fixed stator voltage and frequency. Besides, having a high rotor resistance (which was the case at the start-up), implied lots of power losses. What is expected by using a scalar control is to reduce the losses and at the same time, to decrease the temperature by removing the slip rings of the rotor (choosing a squirrel cage instead of wound rotor). At this stage, changing the frequency allows to displace the theoretic torque-speed curve of the machine in a horizontal way as described in Figure 2.10. However, increasing the frequency implies a decrease of the maximum electric torque what may lead to problems such as stall or an unbearable overshoot of current in windings. It is why the stator voltage is changed to keep a fixed maximum torque.

![Figure 4.37: Speed-torque curve evolution when varying the frequency keeping a constant $\frac{e_x}{\omega_s}$](image1)

![Figure 4.38: Speed-torque curve evolution when varying the frequency keeping a constant $\frac{e_x^2}{\omega_s}$](image2)

In most articles of literature, the ratio $\frac{e_x}{\omega_s}$ is kept constant to have a constant maximum torque. However, with the tool (expression of the steady electric torque) already given in Chapter 2, it is possible to draw the evolution of this curve using a ratio $\frac{e_x^2}{\omega_s}$ constant when varying the stator frequency. But conversely to what was expected, the maximum torque is not really the same for the different curves drawn in Figure 4.37. This observation is also found in reference [16]. Yet, asking to keep a ratio $\frac{e_x^2}{\omega_s}$ constant leads to better results since the maximum torque is nearly the same for each curve (Figure 4.38). Actually, in literature, more assumptions are done to use «scalar control» such as a small slip hypothesis, a $\bar{\bar{E}} = \bar{V}_s$ hypothesis and considering $R_s \bar{I}_s$ as negligible compared to $\bar{V}_s$. Finally, since the induction machine is already modelled in detail, it is used to improve the scalar regulation. The following is a reminder of the expression for the maximum electric torque:

\[
N_{Te} = L_{sr}^2 R_r (E_x^2 + E_y^2); \tag{4.16}
\]

\[
A_{Te} = (L_{rr} L_{ss} - L_{sr}^2 + L_{rr} \omega_s L_{ee})^2 + L_{rr}^2 (R_e + R_s)^2; \tag{4.17}
\]

\[
B_{Te} = 2L_{ss}^2 R_r (R_e + R_s); \tag{4.18}
\]

\[
C_{Te} = R_r^2 ((R_e + R_s)^2 + (\omega_s L_{ss} + \omega_s L_{ee})^2); \tag{4.19}
\]

\[
T_{e,\text{max}} = \frac{N_{Te}}{2\sqrt{A_{Te}C_{Te} + B_{Te}\omega_s}}. \tag{4.20}
\]

At this stage, the maximum electric torque that is chosen, is the one given by nominal characteristics of induction machine (i.e, $\omega_s = 1$ and $e_x = 1$). Then, the stator voltage can be computed as a function of the...
4.2. SCALAR CONTROL

The block diagram in Figure 4.39 shows the modelling of the system where the induction machine is modelled by the same block as for the regulation with rotor resistance command. The content of the controller block is shown in Figure 4.40. It consists to pass the speed error in a PID that returns the stator frequency which passes itself in another block that computes the voltage according to the equation 4.24.

![Figure 4.39: Block diagram of the scalar control system](image_url)

![Figure 4.40: Content of the "Control System" block](image_url)

The saturation blocks are set to have 0 as the lower limit and 1 for the upper limit. It is true that under certain conditions the motor can reach overspeed which means that the speed can be higher than the nominal speed. However, the stator voltage cannot exceed the nominal voltage and the theoretic electric torque curve starts to decrease the maximum torque with increase of the stator frequency. It may also lead to unbearable current in the rotor and in the stator. To conclude, it is better to avoid overspeed. Here are the parameters used for scalar control:

\[
A_{ex} = (L_{rT} L_{ss} - L_{sr}^2 + L_{rT} \omega_s L_e)^2 + L_{sr}^2 (R_e + R_s)^2; \quad (4.21)
\]

\[
B_{ex} = 2 \times L_{sr}^2 (R_s + R_e); \quad (4.22)
\]

\[
C_{ex} = (R_e + R_s)^2 + (\omega_s L_{ss} + \omega_s L_e)^2; \quad (4.23)
\]

\[
\omega_s^2 = \frac{T_{e,max} 2\sqrt{A_{ex} C_{ex} + B_{ex} \omega_s}}{L_{sr}^2}. \quad (4.24)
\]
### Table 4.4: Default parameters for rotor resistance command during the speed regulation of the induction machine

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_s$</td>
<td>0.05</td>
<td>$R_r$</td>
<td>0.05</td>
</tr>
<tr>
<td>$\omega_{s,i}$</td>
<td>0</td>
<td>$L_{ss}$</td>
<td>2.09</td>
</tr>
<tr>
<td>$L_{sr}$</td>
<td>2</td>
<td>$L_{tr}$</td>
<td>2.15</td>
</tr>
<tr>
<td>$R_e$</td>
<td>0.001</td>
<td>$L_e$</td>
<td>0.001</td>
</tr>
<tr>
<td>$e_{s,i}$</td>
<td>0</td>
<td>$e_{r,i}$</td>
<td>0</td>
</tr>
<tr>
<td>$T_{mo,i}$</td>
<td>0.5</td>
<td>$T_{mo,p}$</td>
<td>0.9</td>
</tr>
<tr>
<td>$A$</td>
<td>0.6297</td>
<td>$B$</td>
<td>0.07</td>
</tr>
<tr>
<td>$T_{start}$</td>
<td>50s</td>
<td>$T_{statique}$</td>
<td>0</td>
</tr>
<tr>
<td>$k_p$</td>
<td>2</td>
<td>$k_i$</td>
<td>2</td>
</tr>
<tr>
<td>$k_d$</td>
<td>0</td>
<td>$\omega_{ref}$</td>
<td>8000 RPM</td>
</tr>
<tr>
<td>$h$</td>
<td>1e-2s</td>
<td>$t_{perturb}$</td>
<td>150s</td>
</tr>
</tbody>
</table>

#### 4.2.2 Brief simulation and script comparison

The simulation tests can follow the same methodology as the rotor resistance regulation. One can vary one more time all parameters. However, it has not been done to avoid redundancy in results. The PID is tuned in the same way as the previous regulation, i.e., by changing successively $k_p$ and $k_i$ (the derivative term is given up). Finally, the same simulation is performed by a Matlab script using the same induction machine model.

![Figure 4.41: State evolution of the induction machine during the scalar regulation, comparison of Matlab script results and those given by Simulink](image)

#### 4.2.3 Practical implementation

The instruments needed to control the machine is nearly the same as the regulation with rotor resistance command. The only difference is that no torque measurement is needed. The regulation is not only a PI
controller independent of the machine model. It also consists of stator voltage supply computations as a function of the stator frequency (output of the PI controller) and depending on the machine parameters (resistances, inductances). It is why a good measurement of the induction machine parameter is required. Identification techniques may be interesting to find indirectly the induction machine parameters.

4.3 Field oriented control

In this section, the field oriented control is introduced in a theoretical way. No simulation has been performed for this kind of regulation. This is an introduction to the realisation of this type of controller. It consists in measuring the stator currents (the rotor speed also) and to project those quantities on a specific Park plan built for the controller. This Park plan is defined as aligned with the magnetic flux in such a way that:

\[
\psi_{qr} = 0; \quad (4.25)
\]

\[
\psi_{dr} \neq 0. \quad (4.26)
\]

Conversely to the Chapter 2, where the Park plan was defined as a rotating plan at a fixed speed \( \omega_s \), here, it rotates at varying speed.

4.3.1 Electric torque re-expressed

In this subsection, the motor torque is rewritten taking into account the equations (4.25) and (4.25). As a reminder, here is the electric torque expression:

\[
T_e = \psi_{dr} i_{qr} - \psi_{qr} i_{dr}. \quad (4.27)
\]

The rotor currents, \( i_{dr}, i_{qr} \), can be computed as functions of \( \psi_{dr}, \psi_{qr}, i_{ds}, i_{qs} \):

\[
\psi_{dr} = L_{sr} i_{ds} + L_{rr} i_{dr} \Rightarrow i_{dr} = \frac{\psi_{dr}}{L_{rr}} - \frac{L_{sr} i_{ds}}{L_{rr}}; \quad (4.28)
\]

\[
\psi_{qr} = L_{sr} i_{qs} + L_{rr} i_{qr} \Rightarrow i_{qr} = \frac{\psi_{qr}}{L_{rr}} - \frac{L_{sr} i_{qs}}{L_{rr}}. \quad (4.29)
\]

By injecting it in the electric torque (4.27), it becomes:

\[
T_e = \frac{L_{sr}}{L_{rr}} \left[ \psi_{qr} i_{ds} - i_{qs} \psi_{dr} \right]; \quad (4.30)
\]

and since \( \psi_{qr} \) is equal to 0, the electric torque retrieved is very similar to the one of a DC machine \( T_e = k_1 \psi i_r \) with \( \psi = k_2 i_f \), where \( k_1 \) and \( k_2 \) are constants depending on the machine parameters:

\[
T_e = -\frac{L_{sr}}{L_{rr}} i_{qs} \psi_{dr}. \quad (4.31)
\]

The strategy to regulate the speed is to maintain a fixed flux \( \psi_{dr} \) at its maximum. Intuitively, it is like asking to have the most powerful magnet to rotate the rotor. The rotor can be assimilated as a magnetic dipole in the presence of a magnetic field (created by the windings of the stator). The couple is then given by the following formula:

\[
T_e = M \wedge B; \quad (4.32)
\]

where the magnetic moment \( M \) is assimilated to an expression of the magnetising current and the magnetic field to the current \( i_{qs} \). However, \( \psi_{dr} \) must be below its saturation value to avoid power losses. Indeed, at
saturation, the currents will increase dramatically. Afterwards, the torque is controlled by regulating the current $i_{qs}$.

Finally, the torque is indirectly measured, since the current $i_{qs}$ is obtained by Park transformation of three-phase stator currents measured. Concerning the $\psi_{dr}$, it is computed in the following subsection.

### 4.3.2 Introduction of the magnetising current

The magnetising current $i_{mr}$ is the current that "magnetises" the rotor. It is defined as:

$$\psi_{dr} = L_{sr} i_{mr}. \quad (4.33)$$

Henceforth, the flux $\psi_{dr}$ is indirectly measured by computing the magnetising current. Since $\psi_{dr} = L_{sr} i_{ds} + L_{rr} i_{dr}$, this current can be re-expressed as a function of $i_{dr}$ and $i_{ds}$:

$$i_{mr} = i_{ds} + \frac{L_{rr}}{L_{sr}} i_{dr}. \quad (4.34)$$

The current $i_{dr}$ can thus be re-expressed as a function of $i_{mr}$ and $i_{ds}$:

$$i_{dr} = \frac{L_{sr}}{L_{rr}} (i_{mr} - i_{ds}). \quad (4.35)$$

The magnetising current can be computed thanks to the stator current measurements. To this purpose, the time derivative of the flux $\psi_{dr}$ is rewritten:

$$t_B \frac{d\psi_{dr}}{dt} = - (R_r i_{dr} + (\omega_s - \omega_r) \psi_{qr}); \quad (4.36)$$

$$= -R_r i_{dr}. \quad (4.37)$$

Now by injecting $\psi_{dr}$ and $i_{dr}$ from equations (4.33) and (4.35), the magnetising current is expressed in an ordinary differential equation depending on the measurement $i_{ds}$:

$$T_{mr} \frac{di_{mr}}{dt} + i_{mr} = i_{ds}; \quad (4.38)$$

$$T_{mr} = \frac{t_B L_{rr}}{R_r} \text{ in second.} \quad (4.39)$$

So the current $i_{mr}$ is obtained by inserting $i_{ds}$ as input of an order lag transfer function. To obtain its discrete transfer function (z-transform), the trapezoidal method is applied to the derivative:

$$i_{mr}(t + h) = i_{mr}(t) + \frac{h}{2 T_{mr}} (i_{ds}(t) - i_{mr}(t) + i_{ds}(t + h) - i_{mr}(t + h));$$

$$i_{mr}(t + h) \left(1 + \frac{h}{2 T_{mr}} \right) + \left(\frac{h}{2 T_{mr}} - 1 \right) i_{mr}(t) = \frac{h}{2 T_{mr}} (i_{ds}(t + h) + i_{ds}(t)). \quad (4.40)$$

Considering discrete indices, it leads to:

$$i_{mr}[n] \left(1 + \frac{h}{2 T_{mr}} \right) + \left(\frac{h}{2 T_{mr}} - 1 \right) i_{mr}[n - 1] = \frac{h}{2 T_{mr}} (i_{ds}[n] + i_{ds}[n - 1]); \quad (4.41)$$

where $[n]$ is the unknown state and $[n - 1]$ is the previous state. This equation is a small LTI system and the transfer function linking $i_{ds}$ to $i_{mr}$ can easily be computed. The method is illustrated in Figure 4.42.
4.3. FIELD ORIENTED CONTROL

Transfer function of LTI systems (discrete case)

Let’s consider the discrete LTI system described by the difference equation

\[ \sum_{k=0}^{M} a_k y[n - k] = \sum_{k=0}^{M} b_k u[n - k] \]

the Z-transform gives

\[ \sum_{k=0}^{M} a_k z^{-k} Y(z) = \sum_{k=0}^{M} b_k z^{-k} U(z) \]

The transfer function is therefore given by

\[ H(z) = \frac{Y(z)}{U(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{M} a_k z^{-k}} = \frac{N(z)}{D(z)} \]

Figure 4.42: Slides from «Introduction to signals and systems» course of G. Drion [19]

The transfer function to compute \( i_{mr} \) with \( i_{ds} \) is given by:

\[ H(z) = \frac{\frac{h}{T_{mr}} \left( 1 + z^{-1} \right)}{\left( 1 + \frac{h}{2T_{mr}} \right) + \left( \frac{h}{2T_{mr}} - 1 \right) z^{-1}}. \] (4.42)

4.3.3 Park angle computation

Finally, the rotation speed of the Park plan must be computed in order to apply the Park transformation to the measurements. At this aim, first, one can demonstrate that the rotor quadrature current \( i_{qr} \) is directly proportional to the stator quadrature current \( i_{qs} \):

\[ \psi_{qr} = 0 = L_{sr} i_{qs} + L_{rr} i_{qr}; \]
\[ i_{qr} = -\frac{L_{sr}}{L_{rr}} i_{qs}. \] (4.43)

Then using the equation containing the derivative of rotor flux, the Park plan speed \( \omega_s \) can be isolated:

\[ 0 = R_r i_{qr} - (\omega_s - \omega_r) \psi_{dr}; \]
\[ \omega_s = \omega_r + \frac{R_r i_{qr}}{\psi_{dr}}. \] (4.44)

The last thing to do is to inject equation (4.43) in the previous one:

\[ \omega_s = \omega_r - \frac{R_r \frac{L_{sr}}{L_{rr}} i_{qs}}{L_{sr} i_{mr}}; \]
\[ \omega_s \omega_B = \omega_r \omega_B - \frac{i_{qs}}{t_B \frac{L_{mr}}{R_r} i_{mr}}; \]
\[ \omega_s \omega_B = \omega_r \omega_B - \frac{i_{qs}}{T_{mr} i_{mr}}. \] (4.45)

The rotation speed of the Park plan is obtained since \( i_{qs} \) is given by Park transformation of the stator current measurements and the rotor speed \( \omega_r \) is measured thanks to an encoder.
4.3.4 Block diagram of the regulation

The block diagram of the regulation is shown in Figure 4.43. The encoder measure the rotor speed. The Field weakening is a block that gives as output a constant magnetising current but when the rotor speed overshoots the nominal speed, this magnetising current will decrease. Indeed, the stator voltage is maximum (nominal voltages) and the only way to increase the torque (to accelerate) is to reduce the magnetising current. That explanation is quite intuitive and not really theoretic. For more information about Field Weakening, it is advised to consult reference [2].

![Block diagram of the field oriented control of the asynchronous machine](image)

**Figure 4.43:** Block diagram of the field oriented control of the asynchronous machine

4.4 Comparison

![Comparison of performances of the two methods: rotor resistance command or scalar control](image)

**Figure 4.44:** Comparison of performances of the two methods: rotor resistance command or scalar control.
4.5. WORK PERFORMED IN CHAPTER 4

In Figure 4.44, the results of the first two methods introduced are displayed. "resistance" stands for the speed regulation with rotor resistance command while "scalar" stands for the scalar control. As a reminder, the command of the scalar regulation is effectuated on the stator frequency and additionally on the stator voltage to keep constant maximum electric torque. For equity, the feedforward is not used when using the rotor resistance command. As observed, the scalar control is clearly the favourite. The undershoot in speed is lower, the efficiency is higher and the joule losses are divided by more than two. The currents are very similar. The overshoot of current at the start is higher but it should not lead to issues since it is still below 1pu.

4.5 Work performed in Chapter 4

This chapter is the end of the work. An electric machine has been chosen to drive the compressor at a constant speed. This machine was modelled thanks to physical laws. Solver tools have been created to simulate the machine in operation. The model was validated by several simulation test: valve closure $T_{mo}$ scale function, rotor resistance scale function. Three regulations of speed were introduced in this chapter to keep the compressor at a constant speed whatever the aerodynamic conditions in which the compressor is. There were analysed theoretically and two of them were implemented in both Simulink and Matlab script (to allow an easy further translation in free programming languages such as c, c++, scilab). The regulations were verified by comparing results from Simulink and results from Matlab. All main stakes are finally full-filled.
To conclude this work, a brief summary is realised. The need was to find a way to control the speed of a compressor and to perform the simulations. To characterise a compressor, aerodynamic states are observed in iso-speed changing valve openings. Since the compressor exerts a braking torque depending on the valve openings, the electric motor driving the compressor should be regulated to maintain a fixed speed. Many motors could suit for this application and one of them has been chosen: the induction machine. A model was established thanks to literature and courses. This model was finally included in a closed-loop system that regulates its speed. Three methods were proposed: rotor resistance command, scalar control and field oriented control. Two were implemented and compared. It has been concluded that the scalar control has better performance (for the dynamic and for the power losses).

The project contains still many points that would be interesting to develop. The first idea would be to perform the same simulations using a synchronous machine. It would allow to highlight the advantages and the disadvantages of this kind of machine compared to the induction one. Another interesting idea is to simulate the field oriented control that should be able to improve again control. The importance of converters were also a bit neglected in this work. At that range of power, it is important to well design the AC-AC converter. At this aim, some references are added: [10], [12]. Finally, the operating speed plays an important role in the control of the machine. The control may be problematic near the nominal speed since all states of the machine will change. Currents may reach unbearable values due to a small overshoot.

Concerning the aerodynamic part, a future work would be to change the mechanical torque, previously set as a hypothesis. The idea is to find the mechanical torque based on a simulation of the thermodynamic states of the compressor in operation and to incorporate it in the induction machine regulation. Henceforth, the system would model all main components of a compressor test bed: the thermodynamic circuit of air flow, the mechanic behaviour of the compressor, the electric machine that is controlled to rotate the compressor at a fixed speed and the gearbox efficiency.


This appendix contains the useful functions to compute the initial steady-states of an induction machine.

```matlab
% Function [IC,ex,ey]=Initial_condition(x_guess,parameter_I)
% returns the initial steady-states for an asynchronous machine.
% IC correspond to this: [psidr psiqr wr ids iqs idr iqr psids psiqs]
% The inputs:
% x_guess(1X9): a guess for the initial condition.
% parameter_I=[Rs Rr Re Lss Lsr Lrr wN H Tmo A B C vds vqs]

function [IC,ex,ey]=Initial_condition(x_guess,parameter_I)

% Check vector size
assert(length(x_guess)==9,'The guess vector must contain 9 values')
assert(length(parameter_I)==15,'The parameter of machine must contain 15 values')

% Parameter
% x_guess(1X9): a guess for the initial condition.
% x_guess:
% x=x_guess;

% Set Tolerance and number max of iteration
Tolerance=1e-15;
Nbr_max_iteration=2000;
iteration=0;

% Compute F(x_k): all equation
equation=equ_evaluation(x,parameter_I);

% While max(abs(equation))>Tolerance
while max(abs(equation))>Tolerance

% Compute Jacobian f'(x_k)
Matrix=MA_jacobian(x,parameter_I);

% Solve system: x_{k+1}-x_k
xk_xk1=Matrix\(-equation');

% Iterate x_{k+1}=x_k+(x_{k+1} - x_k)
x=x+xk_xk1';

% Compute f(x_k)
end
```
equation=equ_evaluation(x,parameter_I);%1X10
iteration=iteration+1;
if iteration>=Nbr_max_iteration
fprintf('Max number of iteration reached: %d
',Nbr_max_iteration);
fprintf('Current absolute error: [%3.3e %3.3e %3.3e %3.3e %3.3e %3.3e %3.3e %3.3e %3.3e]
',equation)
fprintf('Please set higher tolerance: %3.3e or try another guess\n',Tolerance)
IC=zeros(1,9);
ex=NaN;
ey=NaN;
return
end
end

%return the initial condition
IC=x;

%parameter_I=[Rs Rr Re Lss Lsr Lrr Le wn H Tmo A B C vds vqs]
[Rs,Rr,Re,Lss,Lsr,Lrr,Le,wn,H,Tmo,A,B,C,vds,vqs]=vec2el(parameter_I);
x=[psidr psiqr wr ids iqs idr iqr psids psiqs]=vec2el(x);

%Compute ex ey
ex=vqs+Re*iqs-wN*Le*ids;
ey=vds+Re*ids+wN*Le*iqs;

end
equ_evaluation.m:

%Function equation=equ_evaluation(x,parameter_I):
% Evaluate the equations of the system to solve at states given in x
% using the parameter of induction machine: parameter_I.
% The inputs:
x=[psidr psiqr wr ids iqs idr iqr psids psiqs]
% parameter_I=[Rs Rr Re Lss Lsr Lrr Le wn H Tmo A B C vds vqs]
function equation=equ_evaluation(x,parameter_I)

%parameter_I=[Rs Rr Re Lss Lsr Lrr Le wn H Tmo A B C vds vqs]
[Rs,Rr,Re,Lss,Lsr,Lrr,Le,wn,H,Tmo,A,B,C,vds,vqs]=vec2el(parameter_I);
x=[psidr psiqr wr ids iqs idr iqr psids psiqs]=vec2el(x);
equation=zeros(1,9);

%equation tension rotor:
equation(1)=Rr*idr+(wN-wr)*psiqr;
equation(2)=Rr*iqr-(wN-wr)*psidr;
%equation of motion:
equation(3)=psidr*iqr-psiqr*idr-Tmo*(A*wr^2+B*wr+C);
%equation tension stator:
equation(4)=-vds+Rs*ids+wN*psiqs;
equation(5)=-vqs+Rs*iqs-wN*psids;
%equation inductance stator and rotor: (6) -> (9)
equation(6)=-psids+Lss*ids+Lsr*idr;
equation(7)=-psiqs+Lss*iqs+Lsr*iqr;
equation(8) = -psidr + Lsr*ids + Lrr*idr;
equation(9) = -psiqr + Lsr*iqs + Lrr*iqr;
end

MA_jacobian.m:

% Function Matrix = MA_jacobian(x, parameter_I)
% Evaluate the Jacobian matrix of the system given the current states x
% and the parameters of the induction machine

% Inputs:
% x = [psidr psiqr wr ids iqs idr iqr psids psiqs]
% parameter_I = [Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]
function Matrix = MA_jacobian(x, parameter_I)

% parameter_I = [Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]
[Rs, Rr, Re, Lss, Lsr, Lrr, Le, wN, H, Tmo, A, B, C, vds, vqs] = vec2el(parameter_I);

% x = [psidr psiqr wr ids iqs idr iqr psids psiqs]
[psidr, psiqr, wr, ids, iqs, idr, iqr, psids, psiqs] = vec2el(x);

% Jacobian computed thanks to "Jacobian_sys"
Matrix = [0 wN - wr -psiqr 0 0 Rr 0 0 0;...
wr - wN 0 psidr 0 0 0 Rr 0 0 0;...
-iqr -idr -Tmo*(B + 2*A*wr) 0 0 -psiqr psidr 0 0 0;...
0 0 0 Rs 0 0 0 0 wN;...
0 0 0 0 Rs 0 0 0 -wN 0;...
0 0 0 0 Lss 0 Lsr 0 -1 0;...
0 0 0 0 Lss 0 Lsr 0 0 0;...
-1 0 0 Lsr 0 Lrr 0 0 0;...
0 -1 0 0 Lsr 0 Lrr 0 0 0;]
end

Jacobian_sys.m:
syms Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs;
syms psidr psiqr wr ids iqs idr iqr psids psiqs;
variable = [psidr psiqr wr ids iqs idr iqr psids psiqs];

% equation tension rotor:
equation(1) = Rr*idr + (wN-wr)*psiqr;
equation(2) = Rr*iqr - (wN-wr)*psidr;

% equation of motion:
equation(3) = psidr*iqr - psiqr*idr - Tmo*(A*wr^2 + B*wr + C);

% equation tension stator:
equation(4) = -vds + Rs*ids + wN*psiqs;
equation(5) = -vqs + Rs*iqs - wN*psids;

% equation inductance stator and rotor: (6) -> (9)
equation(6) = -psids + Lss*ids + Lsr*idr;
equation(7) = -psiqs + Lss*iqs + Lsr*iqr;
equation(8) = -psidr + Lsr*ids + Lrr*idr;
equation(9) = -psiqr + Lsr*iqs + Lrr*iqr;

jacobian(equation, variable)
This appendix contains the useful functions to compute the next states of the induction machine.

**Newton_solve_NL_system.m:**

```matlab
function [x_t1,y_t1]=Newton_solve_NL_system(x_t0,y_t0,h,parameter)

% Tolerance
Tolerance=1e-12;

% Initialize to the previous state:
xi=x_t0;
xd_t0=Newton_f_evaluation(x_t0,y_t0,parameter);
yi=y_t0;

% Initialize the increment vector
Delta=ones(9,1);
iteration=0;
Nbr_max_iteration=2000;

while max(abs(Delta))>Tolerance
    DF=Newton_Jacobian_evaluation(xi,yi,h,parameter);
    F=Newton_system_evaluation(xi,yi,x_t0,xd_t0,h,parameter);
    Delta=DF'*(-F);
    xy_i=[x_i y_i]+Delta';
    x_i=xy_i(1:3);
    y_i=xy_i(4:11);
    iteration=iteration+1;
end

if iteration>Nbr_max_iteration
    fprintf('Max number of iteration reached: %d\n',Nbr_max_iteration);
    fprintf('Current absolute error: [%.3e %.3e %.3e %.3e %.3e %.3e %.3e %.3e %.3e %.3e %.3e]\n',Delta);
    fprintf('Please set higher tolerance: %.3e or try another guess\n',Tolerance)
    x_t1=NaN*ones(1,3);
    y_t1=NaN*ones(1,8);
end
```
APPENDIX B. APPENDIX B: NEWTON SOLVER FUNCTION

Newton_f_evaluation.m:

1 %Function f=Newton_f_evaluation(x,y,parameter)
2 % Evaluate the equations f of the system (See Ch3)
3 %
4 %Inputs:
5 % x: the current states [psidr,psiqr,wr]
6 % y: the current states [vds,vqs,ids,iqs,idr,iqr,psids,psiqs]
7 % parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)
8 function f=Newton_f_evaluation(x,y,parameter)
9 %x_prec do not change during convergence BUT x,y change
10 [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB]=vec2el(parameter);
11 %Assignment of time dependent parameters:
12 [psidr,psiqr,wr]=vec2el(x);
13 [vds,vqs,ids,iqs,idr,iqr,psids,psiqs]=vec2el(y);
14 %the 3 eq: xdot=f(x,y,u)
15 f(1)=-wB*(Rr*idr+(wN-wr)*psiqr);
16 f(2)=-wB*(Rr*iqr-(wN-wr)*psidr);
17 f(3)=(1/(2*H))*(psidr*iqr-psiqr*idr-Tmo*(A*wr^2+B*wr+C));
18 end

Newton_g_evaluation.m:

1 %Function g=Newton_g_evaluation(x,y,parameter)
2 % Evaluate the equations g of the system (See Ch3)
3 %
4 %Inputs:
5 % x: the current states [psidr,psiqr,wr]
6 % y: the current states [vds,vqs,ids,iqs,idr,iqr,psids,psiqs]
7 % parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)
8 function g=Newton_g_evaluation(x,y,parameter)
9 %x_prec do not change during convergence BUT x,y change
10 [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB]=vec2el(parameter);
11 %Assignment of time dependent parameters:
12 [psidr,psiqr,wr]=vec2el(x);
13 [vds,vqs,ids,iqs,idr,iqr,psids,psiqs]=vec2el(y);
14 %the 12 eq: 0=g(x,y,u)
15 g(1)=-vqs+ex-Re*iqs+wN*Le*ids;
16 g(2)=-vds+ey-Re*ids-wN*Le*iqs;
17 g(3)=-vds+Rs*ids+wN*psiqs;
18 g(4)=-vqs+Rs*iqs-wN*psids;
19 g(5)=psids+Lss+ids+Lsr*idr;
20 g(6)=-psiqs+Lss+iqs+Lsr*iqr;
21 g(7)=-psidr+Lsr+ids+Lrr*iqr;
22 g(8)=-psiqr+Lsr+iqs+Lrr*iqr;
23 end
Newton_Jacobian_evaluation.m:

```matlab
function DF=Newton_Jacobian_evaluation(x_i,y_i,h,parameter)
% Evaluate the jacobian matrix of the non linear system to solve(See Ch3)
% Inputs:
% x_i: the current states [psidr,psiqr,wr]
% y_i: the current states [vds,vqs,ids,iqs,idr,iqr,psids,psiqs]
% parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)
function DF=Newton_Jacobian_evaluation(x_i,y_i,h,parameter)

%parameter
[Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB]=vec2el(parameter);

%Assignment of time dependent parameters:
[psidr,psiqr,wr]=vec2el(x_i);
[vds,vqs,ids,iqs,idr,iqr,psids,psiqs]=vec2el(y_i);

%Jacobian initialisation
DF=[ -2/h, -wB *(wN - wr), psiqr *wB, 0, 0, 0, 0, -Rr *wB, 0, 0, 0;
     wB*(wN - wr), -2/h, -psidr *wB, 0, 0, 0, 0, 0, -Rr *wB, 0, 0;
     iqr/(2*H), -idr/(2*H), - 2/h - (Tmo*(B + 2*A*wr))/(2*H), 0, 0, 0, 0, -psiqr/(2 *H), psidr/(2*H), 0, 0;
     0, 0, 0, 0, -1, Le *wN, -Re, 0, 0, 0;
     0, 0, 0, -1, 0, -Re, -Le*Wn, 0, 0, 0;
     0, 0, 0, 0, 0, -1, Rs, 0, 0, 0;
     0, 0, 0, 0, 0, 0, -1, Rs, 0, 0, 0;
     0, 0, 0, 0, 0, 0, 0, Lss, 0, Lr, 0;
     0, 0, 0, 0, 0, 0, 0, Lsr, 0, Lr, 0;
     -1, 0, 0, 0, 0, 0, 0, Lsr, 0, Lr, 0];
end
```

Newton_system_evaluation.m:

```matlab
function F=Newton_system_evaluation(x_i,y_i,x_prec,xd_prec,h,parameter)
% Evaluate the equations of the whole system F
% Inputs:
% x_i: the iterated states [psidr,psiqr,wr]
% y_i: the iterated states [vds,vqs,ids,iqs,idr,iqr,psids,psiqs]
% x_prec: the previous states [psidr,psiqr,wr]
% xd_prec: the derivative of the previous states [psidr,psiqr,wr]
% h: time step
% parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)
function F=Newton_system_evaluation(x_i,y_i,x_prec,xd_prec,h,parameter)

%ATTENTION: x_prec, xd_prec, u_suiv do not change during convergence BUT x,y change
%Evaluate the 3 eq: xdot=f(x,y,u) at iteration i of convergence
f=Newton_f_evaluation(x_i,y_i,parameter);

%the 3 first eq: f{ij(t+1) - 2/h * x{i,t+1} + c(t) =0
%with c(t)=xd_prec+c/h*x_prec
F(1:3)=f-2/h*x_i+xd_prec+2/h*x_prec;
F(4:11)=Newton_g_evaluation(x_i,y_i,parameter);
end
```

Jacobian_sys.m:
clear all;
syms Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C h wB ex ey;
syms psidr psiqr wr vds vqs ids iqs idr iqr psids psiqs;
variable=[psidr psiqr wr vds vqs ids iqs idr iqr psids psiqs];
f(1)=-wB*(Rr*idr+(wN-wr)*psiqr)-2/h*psidr;
f(2)=-wB*(Rr*iqr-(wN-wr)*psidr)-2/h*psiqr;
f(3)=(1/(2*H))*(psidr*iqr-psiqr*idr-Tmo*(A*wr^2+B*wr+C))-2/h*wr;
g(1)=-vqs+ex-Re*iqs+wN*Le*ids;
g(2)=-vds+ey-Re*ids-wN*Le*iqs;
g(3)=-vds+Rs*ids+wN*psiqs;
g(4)=-vqs+Rs*iqs-wN*psids;
g(5)=-psids+Lss*ids+Lsr*ids;
g(6)=-psiqs+Lss*iqs+Lsr*iqr;
g(7)=-psidr+Lsr*ids+Lrr*idr;
g(8)=-psiqr+Lsr*iqs+Lrr*iqr;
jacobian([f g],variable)
This appendix contains the useful scripts to establish tools for simulink simulations.

**Algebraic_loop_suppresion.m:**

```matlab
clc;clear all;close all;
syms vds vqs ids iqr psi_ds psi_qs real;
syms psi_dr psi_qr real;
syms Re wN Le Rs Lss Lsr Lrr real;
syms ex ey real;

variable=[vds vqs ids iqr psi_ds psi_qs];

eq(1)=vqs+Re*iqs-wN*Le*ids;
eq(2)=vds+Re*ids+wN*Le*iqs;
eq(3)=-vds+Rs*ids+wN*psi_qs;
eq(4)=-vqs+Rs*iqs-wN*psi_ds;
eq(5)=-psi_ds+Lss*ids+Lsr*idr;
eq(6)=-psi_qs+Lss*iqs+Lsr*iqr;
eq(7)=Lsr*ids+Lrr*idr;
eq(8)=Lsr*iqs+Lrr*iqr;

b(1)=ex;
b(2)=ey;
b(7)=psi_dr;
b(8)=psi_qr;
b

matrixA=jacobian(eq,variable);

end

matrixA\b'
```

**torque_max_speed_max_verification.m:**

```matlab
clear all;
syms ws N A B C wr real;
Torque=N*(ws-wr)./(ws-wr).^2*(A*(ws-wr)+B+ws*C)
DerivTorque=diff(Torque,'wr')
wrmax=solve(DerivTorque,'wr')
```
APPENDIX C: USEFUL SCRIPTS

7    subs(Torque, 'wr', wrmax(1))
8    subs(Torque, 'wr', wrmax(2))
This appendix contains some useful functions to simulate the regulation of the induction machine.

**E\text{\_torque\_max\_constant.m}**:  

```matlab
% Function E=E\text{\_torque\_max\_constant}(Tmax,parameter)
% Evaluate the feed voltage to keep the electric max torque equal to Tmax
% it depends on ws that is given in the vector: parameter (u(t) in the thesis)

% Inputs:
% Tmax: the maximum torque that should be kept
% parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,\omega N,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)

function E=E\text{\_torque\_max\_constant}(Tmax,parameter)

[Rs,Rr,Re,Lss,Lsr,Lrr,Le,\omega N,H,Tmo,A,B,C,ex,ey,wB]=vec2el(parameter);

A=(Lrr*Lss-Lsr^2+Lrr*\omega N*Le)^2+Lrr^2*(Re+Rs)^2;
B=2*Lsr^2*(Rs+Re);
C=(Re+Rs)^2+(\omega N*Lss+\omega N*Le)^2;

E=sqrt(Tmax*(2*(A*C)^0.5+B*\omega N)/(Lsr^2));
if E<0
    E=0;
end
```

**echelon_uniform.m**:  

```matlab
% Function Echel=echelon_uniform(Value,time)
% Returns a scale function uniform with the values contained in the vector Value
% and realise a uniform scale function during a time: time(end).

% Inputs:
% Value: vector containing the different step of the scale function
% time: time vector, containing the time simulation

function Echel=echelon_uniform(Value,time)

nbr_value=length(Value);

for t=1:nbr_value-1
    if E<0
        E=0;
    end
```

---

87
APPENDIX D: USEFUL FUNCTIONS

debut=(t-1)*nbr_constant+1;
fini=t*nbr_constant;
Echel(debut:fin)=Value(t)*ones(1,nbr_constant);
end
t=t+1;
debut=(t-1)*nbr_constant+1;
Echel(debut:end)=Value(t)*ones(1,nbr_time-nbr_constant*(nbr_value-1));

speed_torqueMax.m:

%Function [wr_Tmax,Tmax]=speed_torqueMax(parameter)
% Evaluate the maximum theoretic electric torque and the corresponding
% speed of the rotor knowing the parameters of the induction machine
%Inputs:
% parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)
function [wr_Tmax,Tmax]=speed_torqueMax(parameter)

[Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB]=vec2el(parameter);
N=Lsr^2*Rr*(ex^2+ey^2);
A=(Lrr*Lss-Lsr^2+Lrr*wN*Le)^2+Lrr^2*(Re+Rs)^2;
B=2*Lsr^2*Rr*(Rs+Re);
C=Rr^2*((Re+Rs)^2+(wN*Lss+wN*Le)^2);
wr_Tmax=wN-(C/A)^0.5;
Tmax=N/(2*(A*C)^0.5+B*wN);
end

Torque_eval.m:

%Function [wr, Torque]=Torque_eval(parameter)
% Evaluate the electric torque-speed curve knowing the parameters of the
% induction machine
%Inputs:
% parameter: [Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB] (u in Ch3)
function [wr, Torque]=Torque_eval(parameter)

wr=linspace(-0.5,1.5,2000);

[Rs,Rr,Re,Lss,Lsr,Lrr,Le,wN,H,Tmo,A,B,C,ex,ey,wB]=vec2el(parameter);
N=Lsr^2*Rr*(ex^2+ey^2);
A=(Lrr*Lss-Lsr^2+Lrr*wN*Le)^2+Lrr^2*(Re+Rs)^2;
B=2*Lsr^2*Rr*(Rs+Re);
C=Rr^2*((Re+Rs)^2+(wN*Lss+wN*Le)^2);
Torque=N*(wN-wr)./((wN-wr).*(A*(wN-wr)+B*wN)+C);
end

vec2el.m:

%Function varargout=vec2el(vector)
% return variable by variable the elements of the vector
%Input:
% vector: a vector/matrix where each returned element correspond to one
% element of the vector(one column if it is a matrix)
function varargout=vec2el(vector)
    L=size(vector,2);
    assert(nargout==L,'Vector is not same length as number of output');
    for i=1:L
        varargout{i}=vector(:,i);
    end
end
This appendix contains the first simulations performed in Chapter 3.

**Essai_initialisation_1.m:**

```matlab
%% Clear workspace,...
clear all;close all;

%% Set default figure parameters
screensize = get( groot, 'Screensize' );
pourcentage=[30 50];
Position=([(100-pourcentage)/100.*screensize(3:4)/2 screensize(3:4).*pourcentage/100];
set(groot,'defaultFigurePosition',Position)
set(groot,'defaultLineLinewidth',2)
set(groot,'defaultAxesTickLabelInterpreter','latex'); set(groot,'defaultLegendInterpreter','latex');
set(groot,'defaultAxesFontSizeMultiplier',1.4)
set(groot,'defaultAxesTitleFontSizeMultiplier',1.7)
set(groot,'defaultAxesFontSize',16)
set(groot,'defaultTextFontSize',16)
set(0,'defaultfigurecolor',[1 1 1])

%% Asynchronous Machine: parameters
%parameter_I=[Rs Rr Rs Lss Lsr Lrr Le wN H Tmo A B C vds vqs]
%Assign values to those variables in per unit system
Rs=0.05;
Rr=0.02;
wN=1;
Lss=2+0.09;
Lsr=2;
Lrr=2+0.15;
Re=1e-3;
Le=-1e-3;
vqs=1;
vds=0;
H=1;
Tmo=0.6;
A=0.5;
```
B=1.5;
C=0.2;
textbox=['$T_m=' num2str(Tmo) '\omega_r^2+\omega_r+\omega_m']$; 

wB=50; %Hz

parameter_I=[Rs Rs Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs];

%% Guess vector
%Unknown variables:
%x=[psidr psiqr wr ids iqs idr iqr psids psiqs]

%Choose a suitable guess vector for the Newton method
wr_guess=0.5;
if wr_guess<0.85
x_guess=0.01*ones(1,9);
else
x_guess=1*ones(1,9);
end

x_guess(3)=wr_guess;

%% Compute initial condition
[IC,ex,ey]=Initial_condition(x_guess,parameter_I);
%IC=[psidr psiqr wr ids iqs idr iqr psids psiqs]

fprintf('ex=%.8f
',ex);
fprintf('ey=%.8f
',ey);

parameter=[Rs Rs Re Lss Lsr Lrr Le wN H Tmo A B C ex ey wB];

%% Check the speed value at steady-state
[wr, Torque]=Torque_eval(parameter);
figure;
plot(wr,Torque); hold on
plot(x_guess(3),Tmo*(A*x_guess(3)^2+B*x_guess(3)+C), 'o', 'Markersize', 10)
plot(IC(3),Tmo*(A*IC(3)^2+B*IC(3)+C), 'o', 'Markersize', 10)

set(leg1,'position',[0.1716 0.3355 0.3852 0.1842]);
set(tex1,'position',[0.0706 -1.9930 0]);
set(tex2,'position',[0.0740 -2.4116 0]);
%% Asynchronous Machine: parameters

%parameter_I=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]

%% Guess vector

% Unknown variables:
% x=[psidr psiqr wr ids iqs idr iqr psids psiqs]

wr_guess_val=0:0.001:1;

for j=1:length(wr_guess_val)
    x_guess=wr_guess_val(j);
    % Choose a suitable guess vector for the Newton method
    x_guess=0.01*ones(1,9);
    x_guess(3)=wr_guess;
    x_guess=wr_guess*ones(1,9);

    %% Compute initial condition
    [IC,ex,ey]=Initial_condition(x_guess,parameter_I);
    %IC=[psidr psiqr wr ids iqs idr iqr psids psiqs]
    wr_val(j)=IC(3);
    psids(j)=IC(8);
end

figure

% plot(wr_guess_val,wr_val-mean(wr_val)); hold on
% plot(wr_guess_val,50*(psids-mean(psids))-1)

plot(wr_guess_val,wr_val)
APPENDIX E. APPENDIX E: FIRST SIMULATIONS

71 xlabel('$\omega_{r,guess}$')
72 ylabel('$\omega_r$ (from IC) ')
73
74 fprintf('ex=%.8f
',ex);
75 fprintf('ey=%.8f
',ey);
76
77 parameter=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C ex ey wB];
78
79 % Check the speed value at steady-state
80 [wr, Torque]=Torque_eval(parameter);
81 figure;
82 plot(wr,Torque); hold on
83 plot(wr,Tmo*(A*wr.^2+B*wr+C))
84 plot(x_guess(3),Tmo*(A*x_guess(3)^2+B*x_guess(3)+C),'o','Markersize',10)
85 plot(IC(3),Tmo*(A*IC(3)^2+B*IC(3)+C),'o','Markersize',10)
86 Tmo*(A*IC(3)^2+B*IC(3)+C)
87 IC(3)
88 xlim([0 1.5])
89 leg1=legend('Theoric $T_e$ curve','Mechanical load $T_m$','Initial guess','Final IC');
90 xlabel('$\omega_r$')
91 ylabel('Torque (pu)')
92
93 set(leg1,'position',[0.1716 0.3355 0.3852 0.1842]);
94 set(tex1,'position',[0.0706 -1.9930 0]);
95 set(tex2,'position',[0.0740 -2.4116 0]);

simulation_scale_resistance.m:

1 %%% Clear workspace,...
2 clc;clear all;close all;
3
4 %%% Set default figure parameters
5 screensize = get (groot, 'Screensize' ) ;
6 pourcentage=80;
7 Position=[(100-pourcentage)/100*screensize(3:4)/2 screensize(3:4)*pourcentage/100];
8 set (groot, 'defaultFigurePosition',Position)
9 set (groot, 'defaultLineLinewidth',2)
10 set (groot, 'defaultAxesTickLabelInterpreter','latex'); set (groot,'defaultLegendInterpreter','latex');
11 set (groot, 'defaultAxesFontSizeMultiplie',1.4)
12 set (groot, 'defaultAxesTitleFontSizeMultiplier',1.7)
13 set (groot, 'defaultTextInterpreter','latex');
14 set (groot, 'defaultAxesFontSize',16)
15 set (0,'defaultfigurecolor',[1 1 1])
16
17 %%% Add folders to the search path
18 addpath(genpath('Initial Conditions'))
19 addpath(genpath('Newton time iteration functions'))
20 addpath(genpath('Useful function'))
21 addpath(genpath('Display graph'))
22
23 %%% Set the per unit Bases
24 P_nominale=10150*1e3; %W
25 SB=P_nominale/0.89;
26 VB=4160; %V
27 fB=60; %Hz
28 tB=1/(2*pi*fB); %s
29 wB=1/tB;
30 IB=SB/(3*VB);
31 ZB=3*VB^2/2SB;
Asynchronous Machine: Initial parameters [Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]

%parameter_I=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]
%Assign values to those variables in per unit system

Rs=0.05;
Rr=10;
wN=1;
Lss=2+0.09;
Lsr=2;
Lrr=2+0.15;
Re=1e-3;
Le=1e-3;
vqs=1;
vds=0;
H=1.5;
Tmo=0.6;
A=2;
B=0;
C=0;

parameter_I=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs];

%% Dynamic Parameter and check initial
parameter=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C 1 0 wB];

%Check the speed value at steady-state, there must be an intersection
[wrth, Torque]=Torque_eval(parameter);
figure;
plot(wrth,Torque); hold on
plot(wrth,Tmo*(A*wrth.^2+B*wrth+C))

%Look if there exist a stationary point
Intersection=Torque-Tmo*(A*wrth.^2+B*wrth+C);
assert(~(isempty(Intersection(Intersection>0)) || isempty(Intersection(Intersection<0))));

%% Time discretisation
Tsim=300;
h=1e-2;
time=0:h:Tsim;
nbr_time=length(time);

%% Rr(t)
Rr_val=[Rr 1 0.5 0.1 0.03];
Rr_t=echelon_uniform(Rr_val,time);
figure;
plot(time,Rr_t)

%% Time loop
X=zeros(nbr_time,3);
Y=zeros(nbr_time,8);

for t=1:1:nbr_time-1
    parameter(2)=Rr_t(t);
    if min(abs((0:10:100)-100*(t-1)/(nbr_time-1)))<(100*0.5/(Tsim/h))
        fprintf('Avancement: %.3f%% \n',100*(t-1)/(nbr_time));
    end
    x_t0=X(t,:);
    y_t0=Y(t,:);
    [x_t1,y_t1]=Newton_solve_NL_system(x_t0,y_t0,h,parameter);
APPENDIX E. APPENDIX E: FIRST SIMULATIONS

95 \[ X(t+1,:) = x_{t1}; \]
96 \[ Y(t+1,:) = y_{t1}; \]
97 end
toc
88
89 \[ \text{Display} \]
90 \[ [\text{psdr } \psi_{qr} \ w_r] = \text{vec2el}(X); \]
91 \[ [\text{vds } \psi_{q} \ i_d \ i_q \ \text{psdr } \psi_{iqs}] = \text{vec2el}(Y); \]
92 figure
93 subplot(2,3,1)
94 plot(time, \( T_m \) \( \ast \) \( (A \ast w_r \ast \text{\textasciitilde}2 \ast B \ast w_r \ast C) \')); hold on
95 plot_{Te}(X,Y,time)
96 str_leg={\'\$T_m$ imposed', '$T_e$ retrieved'};
97 legend(str_leg, 'Fontsize', 15)
98 subplot(2,3,2)
99 \[ \text{for } i=2:length(R_r_val) \]
100 \[ \text{parameter}(2) = R_r_val(i); \]
101 \[ [w_r, aTorque] = \text{Torque_eval}(\text{parameter}); \]
102 \[ \text{plot}(w_r, aTorque, '--'); \]
103 \[ \text{end} \]
104 \[ p(1) = \text{plot}_{Tewr}(X,Y,time); \]
105 \[ p(3) = \text{plot}(w_r, \text{Torque}, \text{\textasciitilde}2 \ast B \ast w_r \ast C), ': '); \]
106 \[ \text{str_leg} = \text{\'Theoric Steady-state', \'Dynamic evolution', \'Mechanical torque'}; \]
107 \[ \text{legend}(p, \text{str_leg}) \]
108 subplot(2,3,3)
109 \[ \text{plot}_{is}(X,Y,time) \]
110 subplot(2,3,4)
111 \[ \text{plot}(time, R_r_t) \]
112 \[ \text{xlabel}('Time(s)') \]
113 \[ \text{ylabel}('R_{r}$ (pu)') \]
114 \[ \text{title}('Rotor resistance') \]
115 subplot(2,3,5)
116 \[ \text{plot}_{wr}(X,Y,time) \]
117 subplot(2,3,6)
118 \[ \text{plot}_{rendement}(X,Y, R_r_t', R_s, time) \]
119 \[ \text{set}(\text{groot}, 'defaultFigurePosition', [584 253 557 489]) \]
120 figure
121 plot(time, \( T_m \) \( \ast \) \( (A \ast w_r \ast \text{\textasciitilde}2 \ast B \ast w_r \ast C) \')); hold on
122 plot_{Te}(X,Y,time)
123 str_leg={\'\$T_m$ imposed', '$T_e$ retrieved'};
124 legend(str_leg, 'Fontsize', 15)
125 title('')
126 figure
127 \[ \text{for } i=2:length(R_r_val) \]
128 \[ \text{parameter}(2) = R_r_val(i); \]
129 \[ [w_r, aTorque] = \text{Torque_eval}(\text{parameter}); \]
130 \[ \text{plot}(w_r, aTorque, '--'); \]
131 \[ \text{end} \]
132 \[ p(2) = \text{plot}_{Tewr}(X,Y,time); \]
133 \[ p(3) = \text{plot}(w_r, \text{Torque}, \text{\textasciitilde}2 \ast B \ast w_r \ast C), ': '); \]
134 \[ \text{str_leg} = \text{\'Theoric Steady-state', \'Dynamic evolution', \'Mechanical torque'}; \]
135 \[ \text{legend}(p, \text{str_leg}) \]
136 title('')
%% Clear workspace,...
clc;clear all;close all;

%% Set default figure parameters
screensize = get( groot, 'ScreenSize' );
pourcentage=80;
Position=[(100-pourcentage)/100*screensize(3:4)/2 screensize(3:4)*pourcentage/100];
set(groot, 'defaultFigurePosition',Position)
set(groot, 'defaultFigurePosition',Position)
set(groot, 'defaultLineLinewidth',2)
set(groot, 'defaultAxesTickLabelInterpreter','latex'); set(groot, 'defaultLegendInterpreter','latex');
set(groot, 'defaultAxesLabelFontSizeMultiplier',1.4)
set(groot, 'defaultAxesFontSizeMultiplier',1.7)
set(groot, 'defaultAxesTitleFontSizeMultiplier',1.7)
set(groot, 'defaultTextInterpreter','latex');
set(groot, 'defaultAxesFontSize',16)
set(0, 'defaultfigurecolor', [1 1 1])

%% Add folders to the search path
addpath(genpath('Initial Conditions'))
addpath(genpath('Newton time iteration functions'))
addpath(genpath('Useful function'))
addpath(genpath('Display graph'))

%% Set the per unit Bases
P_nominal=10150*1e3; %W
SB=P_nominal/0.89;
VB=4160;%V
fB=60;%Hz
tB=1/(2*pi*fB); %s
wB=1/tB;
IB=SB/(3*VB);
ZB=3*VB^2/2*SB;

%% Asynchronous Machine: Initial parameters [Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]
$parameter_I=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs]
%Assign values to those variables in per unit system
Rs=0.05;
Rr=0.03;
wN=1;
Lss=2+0.09;
Lsr=2;
Lrr=2+0.15;
Re=1e-3;
Le=1e-3;
vqs=1;
vds=0;
H=1.5;
Tmo=0.6;
A=2;
B=0;
C=0;

parameter_I=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C vds vqs];

%% Dynamic Parameter and check initial
parameter=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C 1 0 wB];

%Check the speed value at steady-state, there must be an intersection
[wrth, Torque]=Torque_eval(parameter);
figure;
plot(wrth,Torque); hold on
plot(wrth,Tmo*(A*wrth.^2+B*wrth+C))

%Look if there exist a stationnary point
Intersection=Torque-Tmo*(A*wrth.^2+B*wrth+C);
assert(~(isempty(Intersection(Intersection>0)) || isempty(Intersection(Intersection<0))))

%% Initial conditions: x_guess=[psidr psiqr wr ids iqs idr iqr psids psiqs]
%Unknown variables:
%x_guess=[psidr psiqr wr ids iqs idr iqr psids psiqs]

%Choose a suitable guess vector for the Newton method
wr_guess=1;
x_guess=0.1*ones(1,9);
x_guess(3)=wr_guess;

%Compute initial condition fixing vds vqs
[IC,ex,ey]=Initial_condition(x_guess,parameter_I);
if isnan(ex)
    return
end

%x=[psidr psiqr wr]
%y=[vds vqs ids iqs idr iqr psids psiqs]
x_0=IC(1:3);
y_0=[vds vqs IC(4:end)];

parameter=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C ex ey wB];

%% Time discretisation
Tsim=300;
h=1e-2;
time=0:h:Tsim;
nbr_time=length(time);

%% Tmo(t)
T_t=echelon_uniform(Tmo+[0 0.5 1 1.5],time);
figure;
plot(time,T_t)
%% Time loop
X=zeros(nbr_time,3);
Y=zeros(nbr_time,8);
X(1,:)=x_0;
Y(1,:)=y_0;
for t=1:1:nbr_time-1
parameter(10)=T_t(t);
if min(abs((0:10:100)-100*(t-1)/(nbr_time-1)))<(100*0.5/(Tsim/h))
fprintf('Avancement: %.3f%%\n',100*(t-1)/(nbr_time));
end
x_t0=X(t,:);
y_t0=Y(t,:);
[x_t1,y_t1]=Newton_solve_NL_system(x_t0,y_t0,h,parameter);
X(t+1,:)=x_t1;
Y(t+1,:)=y_t1;
end

%% Display
[psidr psiqr wr]=vec2el(X);
[vds vqs ids iqs idr iqr psids psiqs]=vec2el(Y);
figure
subplot(2,3,1)
plot(time,T_t.*(A*wr.^2+B*wr+C)');hold on
plot_Te(X,Y,time)
str_leg={'$T_m$ imposed','$T_e$ retrieved'};
legend(str_leg,'Fontsize',15)

subplot(2,3,2)
plot(wrth,Torque);hold on
plot_Tewr(X,Y,time)
plot(wrth,T_t(1)*(A*wrth.^2+B*wrth+C),':')
plot(wrth,T_t(end)*(A*wrth.^2+B*wrth+C),':')
legend('Theoric Steady-state','Dynamic evolution','$Tm$')

subplot(2,3,3)
plot_is(X,Y,time)

subplot(2,3,4)
plot(time,T_t.*(A*wr.^2+B*wr+C)');hold on
plot_Te(X,Y,time)
xlabel('Time(s)')
ylabel('T_m\ (m)$')
title('$T_m=T_{mo}(A\omega_r^2+B\omega_r+C)\$')

subplot(2,3,5)
plot_wr(X,Y,time)

subplot(2,3,6)
plot_rendement(X,Y,Rr,Rs,time)

set(groot,'defaultFigurePosition',[584 253 557 489])
figure
plot(time,T_t.*(A*wr.^2+B*wr+C)');hold on
plot_Te(X,Y,time)
str_leg={'$T_m$ imposed','$T_e$ retrieved'};
legend(str_leg,'Fontsize',15)
title('')
APPENDIX E. APPENDIX E: FIRST SIMULATIONS

Torque_graph.m:

```matlab
%% Clear workspace,...
clear all;close all;

%% Set default figure parameters
screensize = get(groot,'Screensize');
pourcentage=80;
Position=[(100-pourcentage)/100*screensize(3:4)/2 screensize(3:4)*pourcentage/100];
set(groot,'defaultFigurePosition',Position)
set(groot,'defaultLineLinewidth',2)
set(groot,'defaultAxesTickLabelInterpreter','latex'); set(groot,'defaultLegendInterpreter','latex');
set(groot,'defaultTextInterpreter','latex');
set(groot,'defaultAxesFontSize',16)
set(0,'defaultfigurecolor',[1 1 1])

%% Parameter of inductive machine
Rs=0.05;
Rr=0.9;
%w_ref=1;  %omega ref
Lss=2+0.09;
Lsr=2;
Lrr=2+0.15;
wN=1;
H=1.5;
Tmo=0.8;
A=2;
B=0;
C=0;
Re=0.1e-3;
```
Le=0.1e-3;
ex=1.0;
ey=0;
wB=50;  #Hz

parameter=[Rs Rr Re Lss Lsr Lrr Le wN H Tmo A B C ex ey wB];  #theta_e=0

%S Define the several resistances, voltages and frequencies
Rr_vec=[0.05:0.05:0.25];
w_ref_vec=[0.4:0.1:0.8];
ex_vec=[0.4:0.1:0.8];

%S Display
figure
[w, Tor]=Torque_eval(parameter);
plot(w,Tor);hold on
[w, Tor]=Torque_eval(parameter);
plot(w,Tor)
[a,b]=speed_torqueMax(parameter)
figure(2)
subplot(1,3,1)
for i=1:length(Rr_vec)
    parameter(2)=Rr_vec(i);
    [w, Tor]=Torque_eval(parameter);
    plot(w,Tor)
    [max_pos(i),max_T(i)]=speed_torqueMax(parameter);
    if i==1
        hold on
    end
    str_legend{i}=['Rr: ' num2str(Rr_vec(i))];
end
str_legend{i+1}='Axes';
plot([-1 2],[0 0],'k--');
plot([0 0],[0 -6],'k--');
legend(str_legend)
figure(3)
subplot(2,3,1)
plot(Rr_vec,max_pos);
ylabel('\omega_{r,max}')
xlabel('R_r')
ylim([-6 4])
figure(3)
subplot(2,3,4)
plot(Rr_vec,max_T);
xlabel('R_r')
ylabel('T_{e,max}')
clear max_pos max_T
figure(2)
subplot(1,3,2)
for i=1:length(w_ref_vec)
    parameter(8)=w_ref_vec(i);
    parameter(14)=(1/1)*(w_ref_vec(i))^0.5;
    [w, Tor]=Torque_eval(parameter);
```matlab
[max_pos(i), max_T(i)] = speed_torqueMax(parameter);
plot(w, Tor)
if i == 1
    hold on
end
str_legend{i} = ['$\omega_s$': num2str(w_ref_vec(i))];
end
str_legend{i+1} = 'Axes';
plot([-1 2], [0 0], 'k--');
plot([0 0], [-6 4], 'k--');
legend(str_legend)
parameter(8) = w_ref_vec(end);
parameter(14) = w_ref_vec(end);
xlabel('$\omega_r$');
ylabel('$T_e$');
title('Variable frequency');
ylim([-6 4]);
figure(3)
subplot(2, 3, 2)
plot(w_ref_vec, max_pos);

subplot(2, 3, 5)
plot(w_ref_vec, max_T);
xlabel('$\omega_{ref}$');
figure(2)
subplot(1, 3, 3)
for i = 1:length(ex_vec)
    parameter(14) = ex_vec(i);
[w, Tor] = Torque_eval(parameter);
[max_pos(i), max_T(i)] = speed_torqueMax(parameter);
plot(w, Tor)
if i == 1
    hold on
end
str_legend{i} = ['V: ' num2str(ex_vec(i))];
end
str_legend{i+1} = 'Axes';
plot([-1 2], [0 0], 'k--');
plot([0 0], [-6 4], 'k--');
legend(str_legend)
xlabel('$\omega_r$');
ylabel('$T_e$');
title('Variable voltage');
ylim([-6 4]);
figure(3)
subplot(2, 3, 6)
plot(w_ref_vec, max_T);
xlabel('$V=\sqrt{ex^2+ey^2}$');
figure(4)
[w, Tor] = Torque_eval(parameter);
plot(w, Tor); hold on;
```
\begin{verbatim}
parameter(2)=Rr_vec(end);
[w2, Tor2]=Torque_eval(parameter);
plot(w,Tor2);
plot(w,Tor+Tor2);
wr=linspace(-1,1.5,100);
A=0.5;
B=-0.1;
C=1;
plot(wr,Tmo*(A*wr.^2+B*wr+C),'-.');
plot([-1 2],[0 0],'k--');
plot([0 0],[-3 2],'k--');
legend('Inner cage','Outer cage','Doubly cage','Mechanical torque','Axes')
xlabel('$\omega_r$')
ylabel('$T_e$')
\end{verbatim}
APPENDIX F: SPEED REGULATION VIA ROTOR RESISTANCE

It is composed of the Simulink model (runned thanks to a Matlab script) and the Matlab script. The simulink block diagram is not printed but it is given in attached file.

Simulation_Rr_control.m:

```matlab
%% Clear workspace,...
clear all;close all;

%% Set default figure parameters
screenSize = get( gRoot, 'ScreenSize' );
pourcentage=80;
Position=([(100-pourcentage)/100*screenSize(3:4)/2 screenSize(3:4)*pourcentage/100]);
set(groot,'defaultFigurePosition',Position)
set(groot,'defaultLineLinewidth',2)
set(groot,'defaultAxesTickLabelInterpreter','latex'); set(groot,'defaultLegendInterpreter','latex');
set(groot,'defaultAxesLabelFontSizeMultiplier',1.4)
set(groot,'defaultAxesTitleFontSizeMultiplier',1.7)
set(groot,'defaultTextInterpreter','latex');
set(groot,'defaultAxesFontSize',16)
set(0,'defaultfigurecolor',[1 1 1])

%% Add folders to the search path
addpath(genpath('Initial Conditions'))
addpath(genpath('Newton time iteration functions'))
addpath(genpath('Useful function'))
addpath(genpath('Display graph'))

%% Physical parameter known
GR=1/5.67;    % Gear ratio
teta=1;       % Gear efficiency
Inertia_motor=371;    % kgm^2
Inertia_compressor=20;  % kgm^2
Inertia_equivalent=Inertia_motor+(1/(teta*GR^2))*Inertia_compressor;

%% Set the per unit Bases
P_nominal=10150*1e3;    %W
PF=0.89;                 % power factor
SB=P_nominal/PF;         % nominal apparent power
VB=4160;                % V
fB=60;                 % Hz
```
APPENDIX F: SPEED REGULATION VIA ROTOR RESISTANCE

\[ t_B = \frac{1}{2\pi f_B}; \quad \text{Time basis: } s \]

\[ w_B = \frac{1}{t_B}; \quad \text{Pulse basis: } s^{-1} \]

\[ I_B = \frac{SB}{3+VB}; \quad \text{Current basis: } A \]

\[ Z_B = \frac{3+VB^2}{SB}; \quad \text{Resistance basis: } \Omega \]

\[ \text{poles\_pair} = 2; \quad \text{nbr of pair of poles} \]

\[ \omega_m = \frac{w_B}{\text{poles\_pair}}; \quad \text{Mechanic speed basis: rad/s} \]

\[ H = 0.5 \cdot \text{Inertia\_equivalent} \cdot w_B^2 \cdot \frac{SB}{2}; \quad \text{Inertia in pu system: } s \]

\[ T_{\text{torque}} = SB \cdot w_B; \quad \text{Torque basis: } Nm \]

**Asynchronous Machine: Initial parameters**

\[ R_s = 0.05; \]

\[ R_r = 25; \]

\[ w_N = 1; \]

\[ L_{ss} = 2 + 0.09; \]

\[ L_{sr} = 2; \]

\[ L_{rr} = 2 + 0.15; \]

\[ R_e = 1 \cdot 10^{-3}; \]

\[ L_e = 1 \cdot 10^{-3}; \]

\[ T_{m0\_i} = 0.5; \]

\[ T_{m0\_p} = 0.9; \]

\[ A = 0.9 \cdot 7466 / (T_{\text{torque}} \cdot GR \cdot \eta); \quad \text{90\% of max torque} \]

\[ B = 0.1 \cdot 7466 / (T_{\text{torque}} \cdot GR \cdot \eta); \quad \text{10\% of max torque} \]

\[ C = 0; \]

**Set Electrical Parameters and check the presence of a steady point**

\[ \{R_s, R_r, R_e, L_{ss}, L_{sr}, L_{rr}, L_e, w_N, H, T_{m0\_i}, A, B, C, w_B\}; \]

\( \text{assert} \left( \text{isempty} \left( \text{Intersection} \left( \text{Intersection} > 0 \right) \right) \right) \] || isempty(Intersection(Intersection<0)))

**Time discretisation**

\[ T_{sim} = 200; \]

\[ h = 1\cdot10^{-2}; \]

\[ \text{time} = 0:h:T_{sim}; \]

\[ \text{nbr\_time} = \text{length}(\text{time}); \]

\[ t_{perturb} = 150; \quad \text{time at which a perturbation occurs (valve change)} \]

**Wreference(t)**

\( w_{rp} = 8000; \quad \text{reference speed to reach in [900 10206] RPM} \]

\( w_{ref\_end} = w_{rp} \cdot GR \cdot \frac{2\pi}{60} / w_B; \quad \text{reference speed in pu} \]

\[ T_{start} = 50; \quad \text{Time to start the induction machine: } s \]

\[ \tau_u = 0.4 \cdot T_{start}; \quad \text{Gain: } \rightarrow 0 \Rightarrow \text{start quicker} \]

\[ A_R = -\frac{w_{ref\_end}}{1 - \exp(-T_{start}/\tau_u)}; \]

\[ w_{ref\_time} = A_R \cdot (\exp(-t/\tau_u) - 1); \]

**Time loop**

**Initialize the states to zeros**

\[ X = \text{zeros}(\text{nbr\_time}, 3); \quad \{psidr, psiqr, wr\} \]

\[ Y = \text{zeros}(\text{nbr\_time}, 8); \quad \{vds, vqs, ids, iqs, idr, iqr, psids, psiqs\} \]

\[ e_i = \text{zeros}(\text{nbr\_time}, 1); \quad e_i = \text{int}(0, t, "w(t) - wref(t)") \]
ed=zeros(nbr_time,1); \( \% \text{ed} = \left( \frac{d}{dt} \right) \left[ wr(t) - wref(t) \right] \)
ep=zeros(nbr_time,1); \( \% \text{ep} = \left[ wr(t) - wref(t) \right] \)
kp=150; \( \% \text{proportional gain} \)
ki=60; \( \% \text{integrator gain} \)
kd=0; \( \% \text{derivative gain: 0 => PI controller} \)
ei=Rr/ki*ones(nbr_time,1); \( \% \text{initialize PID} \)

DRrDT_lim=10; \( \% \text{Rate limiter on actuation: max 150pu/s for Rr} \)
T_t(1)=Tmo_i; \( \% \text{Register valve closure Tmo + initial value} \)
Rr_t(1)=Rr; \( \% \text{Register rotor resistance + initial value} \)
DTmoDT_lim=0.05; \( \% \text{max increase of 0.1 in 2s} \)

\% Loop time
for t=1:1:nbr_time-1
    \% Display the advancement in the simulation
    if min(abs([0:10:100]-100*(t-1)/(nbr_time-1)))<(100*0.5/(Tsim/h))
        fprintf('Avancement: %.3f%%
',100*(t-1)/(nbr_time));
    end

    \% Continuous increased of ex (\text{\texttt{-}parameter(14)})
    if time(t)<1
        parameter(14)=time(t); \( \% \text{ex= time if time<1s} \)
    else
        parameter(14)=1; \( \% \text{ex=1 otherwise} \)
    end

    \% Asset the reference speed to follow: wreference
    wreference=wref_time(t); \( \% \text{wreference = wref(t)} \)
    if time(t)>=Tstart
        wreference=wref_end; \( \% \text{wreference = wref_end if time > Tstart} \)
    end
    wref_time(t)=wreference; \( \% \text{register wreference} \)

    \% Register theoretic electric torque curve before the perturbation
    if time(t)==t_perturb
        [wrth1, Torque1]=Torque_eval(parameter);
    end

    \% Change Tmo (\text{\texttt{-}parameter(10)}) at \text{t_perturb=150s}
    if time(t)>=t_perturb
        if parameter(10)<Tmo_p
            if (Tmo_p-parameter(10))/h>DTmoDT_lim \( \% \text{apply a rate limiter} \)
                parameter(10)=parameter(10)...+h*DTmoDT_lim*sign(Tmo_p-parameter(10));
            else
                parameter(10)=Tmo_p;
            end
        end
    end

    \% Solve induction machine system of equations
    x_t0=X(t,:);
y_t0=Y(t,:);
[x_t1,y_t1]=Newton_solve_NL_system(x_t0,y_t0,h,parameter);
X(t+1,:)=x_t1;
Y(t+1,:)=y_t1;

\% PID controller using \( kd=0 \) => PI controller
if time(t)>=Tstart
    ei(t+1)=ei(t)+h/2*(-2*wreference+X(t,3)+X(t+1,3));
else
    ei(t+1)=ei(t)+h/2*(-wref_time(t+1)-wref_time(t)+X(t,3)+X(t+1,3));

end %ei computation
ep(t+1)=-wreference+X(t+1,3); %ep computation
ed(t+1)=2/h*(-X(t,3)+X(t+1,3))-ed(t); %ed computation

Rr_pre=parameter(2); %Previous Rr
parameter(2)=kp*ep(t+1)+ki*ei(t+1)+kd*ed(t+1); %PI controller->Rr

if parameter(2)<0.01
fprintf('SATURATED\n');
parameter(2)=0.01;
end

%Rate Limiter of Rr to 150pu/s
DRDT=(parameter(2)-Rr_pre)/h;
if abs(DRDT)>DRrDT_lim
fprintf('RATE LIMITED\n');
parameter(2)=Rr_pre+h*DRrDT_lim*sign(DRDT);
end

%Register Tmo and Rr
T_t(t+1)=parameter(10);
Rr_t(t+1)=parameter(2);
end

wref_time(t+1)=wreference;

%Register final theoric electric torque curve
[wrth2, Torque2]=Torque_eval(parameter);

%% Display
[psidr,psiqr,wr]=vec2el(X);
[vds,vqs,ids,iqs,idr,iqr,psids,psiqs]=vec2el(Y);
figure
subplot(2,3,1)
plot(time,T_t.*(A*wr.^2+B*wr+C)');hold on
plot_Te(X,Y,time)
str_leg={'$T_m$ imposed','$T_e$ retrieved'};
legend(str_leg,'Fontsize',15)

subplot(2,3,2)
p(1)=plot(wrth1,Torque1,'--');hold on
plot(wrth2,Torque2,'--');
p(2)=plot_Tewr(X,Y,time);
p(3)=plot(wrth,Tmo_i*(A*wrth.^2+B*wrth+C),':');
plot(wrth,Tmo_p*(A*wrth.^2+B*wrth+C),':');
str_leg={'Theoric Steady-state','Dynamic evolution','Mechanical torque'};
legend(p,str_leg)

subplot(2,3,3)
plot_is(X,Y,time)

subplot(2,3,4)
plot(time,Rr_t)
xlabel('Time(s)')
ylabel('R_{r}(\Omega)')
title('Rotor resistance')

subplot(2,3,5)
plot_wr(X,Y,time)

subplot(2,3,6)
plot_rendement(X,Y,Rr_t',Rs,time)

%% figure
```

subplot(2,3,1)
plot(time,1e-3*TorqueB*eta*GR*T_t.*(A*wr.^2+B*wr+C));hold on
Te=(psidr.*iqr-psiqr.*idr);
plot(time,1e-3*TorqueB*eta*GR*Te)
xlabel('Time(s)')
ylabel('Torque(kNm)')
title('Torque')

str_leg={'$T_m$ imposed', '$T_e$ retrieved'};
legend(str_leg, 'Fontsize',15)

subplot(2,3,2)
p(1)=plot(1e-3*60/(2*pi)*wmB/GR*wrth,1e-3*TorqueB*eta*GR*Torque, '--');hold on
plot(1e-3*60/(2*pi)*wmB/GR*wrth1,1e-3*TorqueB*eta*GR*Torque1, '--');
plot(1e-3*60/(2*pi)*wmB/GR*wrth2,1e-3*TorqueB*eta*GR*Torque2, '--');
p(2)=plot(1e-3*60/(2*pi)*wmB/GR*wr,1e-3*TorqueB*eta*GR*Te);
xlabel('Speed (kRPM)')
ylabel('Torque(kNm)')
title('Torque - speed')

str_leg={'Theoric Steady-state', 'Dynamic evolution', 'Mechanical torque'};
legend(p,str_leg)

subplot(2,3,3)
plot(time,(3)^0.5*IB*(ids.^2+iqs.^2).^0.5); hold on
plot(time,(3)^0.5*IB*(idr.^2+iqr.^2).^0.5);
xlabel('Time(s)')
ylabel('Current(A)')
legend('$I_s=\sqrt{i_{ds}^2+i_{qs}^2}$', '$I_r=\sqrt{i_{dr}^2+i_{qr}^2}$')

subplot(2,3,4)
plot(time,ZB*Rr_t);
xlabel('Time(s)')
ylabel('$R_{r}$')

subplot(2,3,5)
[psidr psiqr wr]=vec2el(X);
[vec2el(Y)]
title('Rotor resistance')

subplot(2,3,6)
plot_rendement(X,Y,Rr_t',Rs,time)
figure
subplot(1,2,1)
plot(time,Rr_t*ZB) \text{Hz}
xlabel('Time')
ylabel('Rotor resistance(ohm)')

subplot(1,2,2)
ph=Rs.*(idr.'*2+iqr.'*2)+Rs.*(ids.'*2+iqs.'*2);
Te=(psidr.*iqr-psiqr.*idr);
plot=time,6*SB*ph);hold on
```
# APPENDIX F: SPEED REGULATION VIA ROTOR RESISTANCE

```matlab
plot(time, 1e-6*SB*Te.*wr)
plot(time, 1e-6*SB*ptot)
xlabel('Time(s)')
ylabel('Power($MW$)')
```

**simulink_initial_resistance.m**:

```matlab
%% Reset all
clear all; close all; clc;

%% Set default figure parameters
screensize = get(groot, 'Screensize');
pourcentage = 80;
Position = [(100 - pourcentage) / 100 * screensize(3:4) / 2 screensize(3:4)...
            * pourcentage / 100];
set(groot, 'defaultFigurePosition', Position)
set(groot, 'defaultLineLinewidth', 2)
set(groot, 'defaultAxesTickLabelInterpreter', 'latex')
set(groot, 'defaultLegendInterpreter', 'latex');
set(groot, 'defaultAxesFontSize', 16)
set(0, 'defaultfigurecolor', [1 1 1])

%% Physical parameter known from the compressor
GR = 1 / 5.67; % Gear ratio
eta = 1; % Gear efficiency
Inertia_motor = 371; % kgm^2
Inertia_compressor = 20; % kgm^2
Inertia_equivalent = Inertia_motor + (1 / (eta * GR^2)) * Inertia_compressor;

%% Set the per unit Bases
P_nominal = 10150 * 1e3; % nominal power in W
PF = 0.89; % power factor
VB = 4160; % nominal voltage in V and voltage basis
fB = 60; % nominal frequency in Hz and frequency basis
poles_pair = 2; % nbr of pair of poles

SB = P_nominal / PF; % nominal apparent power and power basis

%% Asynchronous Machine: Initial parameters
[Rs Rr Lss Lsr Lrr Le ws H Tmo A B C vds vqs]

assign values to those variables in per unit system
Rs = 0.05;
ws = 1;
Lss = 2 + 0.09;
Lsr = 2;
Lrr = 2 + 0.15;
Re = 1e-3;
Le = 1e-3;
```

110
%H already given;
%Torque load parameters
Tmo_i=0.5;
Tmo_p=0.9;
t_perturb=150;  %time when Tmo is changed
coulomb_static=0;  %enable coulomb friction: 1-on; 0-off
A=0.9*7466/(TorqueB*GR*eta);
B=0.1*7466/(TorqueB*GR*eta);
C=0;
Tstatique=0*7466/(TorqueB*GR*eta*Tmo_i);  %Static torque if coulomb-on

%% wref(t): start-up
T_Start=50;  %Time to start the machine
Tau_gain=0.4;  %the smaller the abrupter exponential signal
wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time

%% PID parameter
Rrin=25;  %the initial rotor resistance
kp=150;  %proportional gain
ki=60;  %integrator gain
kd=0;  %derivative gain
wrpm=8000;  %reference speed to reach in [900 10206] RPM
wreference=wrpm*GR*2*pi/60/wmB;  %reference speed in pu
feedforward=0;  %feedforward: 1-on; 0-off

%% Time step
h=1e-2;  %in s, the minimum time step used by the model

%% Set the model workspace of simulink model: file Resistance_Control.slx
Name='Resistance_Control';
load_system(Name);

%% PID parameter
Rrin=25;  %the initial rotor resistance
kp=150;  %proportional gain
ki=60;  %integrator gain
kd=0;  %derivative gain

%% Simulation
async_vec=[10 90 150];  %start-up variation
$Lsr_vec=[1.8 2.5 3.5 3.8];
for i=1:length(async_vec)
    hws.assignin('feedforward',0)
    hws.assignin('coulomb_static',0)  %Set parameter kp in the simulink model
    hws.assignin('kp',async_vec(i));
end
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
$h=1e-2;  %in s, the minimum time step used by the model
$hws.assignin('Rrin',25);
$hws.assignin('T_start',50);  %Time to start the machine
$hws.assignin('tau_gain',0.4);  %the smaller the abrupter exponential signal
$hws.assignin('wref(t)=ARr*(exp(-time/Tau_gain)-1) with ARr -> a constant
and time -> time
$hws.assignin('kp',async_vec(i));
$hws.assignin('ki',0);
$hws.assignin('kd',0);
simOut = sim(Name,...
    'StopTime', '200', ...'FixedStep', num2str(h), ... 
    'ZeroCross', 'on', ... 
    'SaveTime', 'on', 'TimeSaveName', 'tout', ... 
    'SaveState', 'on', 'StateSaveName', 'xoutNew', ... 
    'SaveOutput', 'on', 'OutputSaveName', 'youtNew', ... 
    'SignalLogging', 'on', 'SignalLoggingName', 'logout');

%Get back teh results;
Controls = get(simOut, 'Controls');    % [ep, ei, ed, wref, Rr_feed]
Inputs = get(simOut, 'Inputs');         % [Tmo, ex, ey, ws, Rr]
Powers = get(simOut, 'Powers');         % [ptot, pjoule]
% [psidr psiqr psids psiqs idr iqr ids iqs vds vqs wr]
States = get(simOut, 'States');
Torques = get(simOut, 'Torques');       % [Tm Te]

figure(1)
subplot(2,1,1) % Display breaking torque(t) for load
plot(Torques.time, Torques.data(:,1));hold on
xlabel('Time(s)')
ylabel('Mechanical torque')

subplot(2,1,2) % Display motor torque(t) for electric
plot(Torques.time, Torques.data(:,2));hold on
xlabel('Time(s)')
ylabel('Electric torque')

figure(2)
subplot(2,1,1) % Display rotor speed
plot(States.time, States.data(:,11));hold on
xlabel('Time(s)')
ylabel('Rotor speed')

subplot(2,1,2) % Display rotor resistance
plot(Inputs.time, Inputs.data(:,5));hold on
xlabel('Time(s)')
ylabel('Rotor resistance(pu)')

figure(3)
subplot(2,1,1) % Display compressor speed in RPM
plot(States.time, States.data(:,11)*wmB/GR*60/(2*pi));hold on
xlabel('Time(s)')
ylabel('Compressor speed(RPM)')

subplot(2,1,2) % Display compressor motor torque in kNm
plot(Torques.time, Torques.data(:,2)*TorqueB*GR/1000);hold on
xlabel('Time(s)')
ylabel('Electric torque(kNm)')

figure(4)
subplot(2,1,1) % Display compressor speed in RPM
plot(States.time, States.data(:,11)*wmB/GR*60/(2*pi));hold on
xlabel('Time(s)')
ylabel('Compressor speed(RPM)')

figure(5)
% Display rotor speed
plot(States.time, (States.data(:,1).^2+States.data(:,2).^2).^0.5);hold on
xlabel('Time(s)')
ylabel('$\psi_r=\sqrt{\psi_{dr}^2+\psi_{qr}^2}$')

112
subplot(2,1,1)
plot(States.time,(States.data(:,3).^2+States.data(:,4).^2).^0.5);hold on
xlabel('Time(s)')
ylabel('$\psi_s=\sqrt{\psi_{ds}^2+\psi_{qs}^2}$')

figure(6)
plot(States.time,(States.data(:,9).^2+States.data(:,10).^2).^0.5);hold on
xlabel('Time(s)')
ylabel('$V_s=\sqrt{v_{ds}^2+v_{qs}^2}$')

fprintf('Step: %d/%d
',i,length(async_vec))

str_leg{i}=['$k_p$: ' num2str(async_vec(i))];
end

%Display legend
figure(1)
subplot(2,1,1)
legend(str_leg)
subplot(2,1,2)
legend(str_leg)

figure(2)
subplot(2,1,1)
str_leg=str_leg;
str_leg{end+1}='reference';
plot(States.time,Controls.data(:,4),('--', 'color',[0.3 0.3 0.3]));
legend(str_leg)
subplot(2,1,2)
legend(str_leg)

figure(4)
subplot(2,1,1)
plot(States.time,Controls.data(:,4)*wmB/GR*60/(2*pi),('--'));
str_leg=str_leg;
str_leg{end+1}='reference';
legend(str_leg)
subplot(2,1,2)
legend(str_leg)

figure(5)
subplot(2,1,1)
legend(str_leg)
subplot(2,1,2)
legend(str_leg)