

POLITECNICO DI BARI
DOTTORATO DI RICERCA IN INGEGNERIA ELETTRONICA

Curriculum: Macchine ed Azionamenti Elettrici

XVII ciclo

TESI PER IL CONSEGUIMENTO DEL TITOLO

***Online Hybrid Evolutionary Algorithms
for Auto-Tuning of Electric Drives***

Candidato:
Dott. Giuseppe Leonardo Cascella

Tutor:
Prof. Luigi Salvatore
Dr. Mark Sumner

Coordinatore del Corso di Dottorato:
Prof. Francesco Torelli

Anni Accademici 2002-2005

to my mother and my brother for their outstanding support

Acknowledgements

This thesis is the result of my research work with Electrical Machines Group at the Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari.

I am deeply grateful to my supervisor, Professor Luigi Salvatore, for support, guidance and generous time without which my PhD would not have as interesting and profitable.

I would like to thank Prof. Francesco Cupertino and Prof. Silvio Stasi for the many discussions that we had and for sharing their opinion on different subjects.

I would like to express my sincere gratitude to Prof. Greg M. Asher and Dr. Mark Sumner who I worked with as visiting researcher in the Power Electronics, Machines and Control Group (PEMC) at the University of Nottingham.

Special thanks go to Chris Gerada, Prof. Cyril Spiteri-Staines, and Vincenzo Giordano, for their valuable help. It is a real pleasure to work with

vi

ACKNOWLEDGEMENTS

them.

Contents

| | |
|--|-----------|
| 1 Introduction | 7 |
| 1.1 Need for user-friendly drives | 9 |
| 1.2 Online Hybrid Evolutionary Algorithms | 10 |
| 1.3 Thesis outline | 12 |
| 2 PMSM Drive | 13 |
| 2.1 PMSM configurations | 14 |
| 2.2 PMSM model | 15 |
| 2.2.1 Stator reference frame | 17 |
| 2.2.2 Rotor reference frame | 19 |
| 2.3 Vector control of a PMSM | 21 |
| 2.4 Model-based design of the vector control | 25 |
| 2.4.1 q-axis current control loop | 25 |
| 2.4.2 Speed control loop | 28 |
| 2.4.3 d-axis current control loop | 30 |
| 2.5 Self-commissioning | 31 |
| 3 Evolutionary Algorithms | 39 |
| 3.1 Representation | 44 |
| 3.2 Search Domain | 45 |
| 3.3 Evaluation | 46 |
| 3.3.1 Training Test | 48 |
| 3.3.2 Objective Function | 49 |
| 3.3.3 Poorly Performing Solutions | 51 |
| 3.4 Evolutionary Operators | 52 |
| 3.4.1 Selection | 52 |
| 3.4.2 Recombination | 55 |
| 3.4.3 Mutation | 56 |
| 3.4.4 Reinsertion | 57 |
| 3.5 Termination | 58 |

| | |
|--|-----------|
| 4 Hybridization of EAs | 59 |
| 4.1 Nelder and Mead algorithm | 61 |
| 4.1.1 Sorting | 64 |
| 4.1.2 Centroid calculation | 64 |
| 4.1.3 Reflection | 64 |
| 4.1.4 Expansion | 67 |
| 4.1.5 Outside contraction | 68 |
| 4.1.6 Inside contraction | 68 |
| 4.1.7 Shrinking | 69 |
| 4.2 Probabilistic version | 69 |
| 4.3 Multidirectional search algorithm | 75 |
| 5 Experimental setup and results | 81 |
| 5.1 Hardware and Software Implementation | 81 |
| 5.2 Experimental Results | 84 |

List of Tables

| | |
|--|----|
| 5.1 PMSM Nameplate | 84 |
| 5.2 Values of the Control System Parameters | 87 |
| 5.3 Performance Indices Scored by x_{MC} and x_{HEA} after each speed step | 88 |

List of Figures

| | | |
|------|---|----|
| 2.1 | PMSM with different rotor configurations. | 16 |
| 2.2 | PMSM analytical model. | 17 |
| 2.3 | Block diagram of zero d-axis current control of a PMSM. | 22 |
| 2.4 | Block diagram of the anti-windup PI. | 23 |
| 2.5 | Block diagram of the q-axis current control loop. | 26 |
| 2.6 | Equivalent block diagram of q-axis current control loop after approximations. | 27 |
| 2.7 | Final equivalent block diagram of q-axis current control loop. | 27 |
| 2.8 | Block diagram of the speed control loop. | 28 |
| 2.9 | Block diagram of the simplified speed control loop. | 29 |
| 2.10 | Block diagram of the q-axis current control loop. | 30 |
| 2.11 | Block diagram of the d-axis current control loop. | 30 |
| 2.12 | Final equivalent block diagram of d-axis current control loop. | 31 |
| 2.13 | Model-based self-commissioning procedure. | 32 |
| 2.14 | Normal working after self-commissioning. | 33 |
| 2.15 | Adaptive control with parameter tracking. | 36 |
| 2.16 | Model-Reference Adaptive System (MRAS). | 37 |
| 2.17 | Final equivalent block diagram of d-axis current control loop. | 38 |
| 3.1 | Interactive evolution cycle. | 40 |
| 3.2 | Research areas of artificial intelligence. | 42 |
| 3.3 | The parameters to be optimized, encoded as chromosome. | 44 |
| 3.4 | Search domain for a PI controller optimization. | 46 |
| 3.5 | Block diagram of EAs. | 47 |
| 3.6 | The training test is a combination of speed commands (a sequence of 8 speed steps, continuous line) and load torque commands (dashed line). | 48 |
| 3.7 | j -th speed step of the training. | 50 |
| 3.8 | Fitness as function of the individual position and the selective pressure. | 54 |

| | | |
|------|---|----|
| 3.9 | Example of intermediate recombination for a two-dimension problem. | 56 |
| 3.10 | Example of mutation for a two-dimension problem. | 57 |
| 4.1 | Classification of the search techniques. | 62 |
| 4.2 | Elite-based hybrid architecture. | 63 |
| 4.3 | Some example of a simplex. | 65 |
| 4.4 | Flowchart of an iteration of Nelder and Mead's simplex method. | 66 |
| 4.5 | Operations of N-M's simplex method. | 70 |
| 4.6 | Reflection in the probabilistic simplex method. | 72 |
| 4.7 | Contraction in the probabilistic simplex method. | 73 |
| 4.8 | Operations of the concurrent simplex method. | 74 |
| 4.9 | Flowchart of the multi-dimensional simplex method. | 78 |
| 4.10 | Operations of the multi-dimensional simplex method. | 79 |
| 5.1 | Diagram of the hardware implementation. | 83 |
| 5.2 | Diagram of the software implementation. | 83 |
| 5.3 | Frontal view of the experimental setup. | 85 |
| 5.4 | Top view of the experimental setup. | 86 |
| 5.5 | Responses given by \mathbf{x}_{MC} . | 91 |
| 5.6 | Responses given by \mathbf{x}_{HEA} . | 92 |
| 5.7 | Comparison between the responses given by \mathbf{x}_{MC} and \mathbf{x}_{HEA} to the no-load speed reversal. | 93 |
| 5.8 | Comparison between the responses given by \mathbf{x}_{MC} and \mathbf{x}_{HEA} to the full-load speed reversal. | 94 |
| 5.9 | Comparison between the objective functions provided by \mathbf{x}_{MC} and \mathbf{x}_{HEA} . | 95 |
| 5.10 | Experiment to evaluate the amplitude of the speed oscillation when a 15 Hz square-wave load torque is applied. | 95 |
| 5.11 | Comparison between the amplitude frequency responses provided by \mathbf{x}_{MC} and \mathbf{x}_{HEA} . | 95 |
| 5.12 | Comparison between the responses given by \mathbf{x}_{MC} and \mathbf{x}_{HEA} to the second evaluation test. | 96 |

Chapter 1

Introduction

The electric motor drives are the key component of automation over the last twenty years. An electric motor drive is a power electronics device which feeds an electric motor so that its motion is controlled. Electrical motors are the most widespread devices for electrical-to-mechanical energy conversion, e.g., motors use the 70% of the industrial electricity consumption almost $30 \cdot 10^9$ MWh in the United States [1]. The electric drives market still remains one of the most dynamic sectors and its growth is likely to continue in the next years, according to recent studies the European turnover is expected to be worth $1.5 \cdot 10^9$ USD in 2004 and to be close to $2 \cdot 10^9$ USD by 2009 [2, 3].

This is due to a number of reasons:

- electric drives cover a wide range of power, speed and torque. They can reach 100 MW and 100000 rpm for pumps of hydro storage plants, but can also be designed for μW –applications such as micro-surgeon and robotics. The new frontier consists of nanotechnology applications: “a UC Berkeley physicist has created the first nano-scale motor - a gold rotor on a nanotube

shaft that could ride on the back of a virus,” [4];

- electric motor drives make existing electricity applications more efficient. Since 1993, in the United States, the electric motor drives have been stated one of the most promising targets for potential efficiency gain [5]. For example, it is possible to cut an estimated 10% of energy consumed by motors at idle, when no useful work is being accomplished.
- Electric drives can be adopted in extreme operating conditions such as explosive, radioactive, and underwater environments.
- The environmental impact of electric drives, in terms of noise level and waste production, is very low compared to other solutions, e.g. combustion motors.
- Power quality, electromagnetic compatibility (EMC), electromagnetic interference (EMI), and electrostatic discharge (ESD) are taken into account when designing an electric drive.
- New applications in portable tools and electric/hybrid vehicles are being created by employment of higher-power-density batteries, fuel- and solar-cells.

In conclusion, the electric motor drives are a renewed challenge not only for their manufacturers, but for the industrial sectors such as automotive, building automation, heating, robotics, aerospace, mining, constructions and biomedical, where motion control is essential.

1.1 Need for user-friendly drives

Usually manufacturers of sophisticated products run the risk that final users underutilize their products and do not achieve the expected results. It might say that such a problem is due to final user, which is not skilled enough to properly use the product. This vision was common in the past, but is anachronistic for the present market where the customer satisfaction is a high-priority objective of manufacturers. New products not only should implement advanced technology in order to provide new features, but should make available intelligent functions which help the final user to achieve best performances. For example, it is hard to imagine a camera without any autofocus function, or an operative system that does not recognize the most popular hardware.

Following such a trend, recently there has been an increased interest in smarter electric drives, i.e., more user friendly and capable of self-commissioning. Self-commissioning should avoid a typical situation, that is, the final user is forced to buy drive and motor from the same manufacturer to obtain good performances without any extra-tuning is needed. Self-commissioning consists of an automatic procedure for the tuning of controllers based on a previous parameter identification, when a motor is initially connected to the drive [6]. In this way the final user can utilize the drive with a third party motor without worrying about tuning.

The standard techniques for self-commissioning consist of specific sequences of tests to measure electrical and mechanical parameters of the motor. These identified parameters are then used for a model-based control design. By us-

ing such a technique, the control system can be tuned in a very short time. Although good enough for several applications, only a sub-optimal tuning is achieved because the issues concerning the multiple inputs, system nonlinearities and uncertainties of the model-based design remain unaddressed.

1.2 Online Hybrid Evolutionary Algorithms

A possibility could be given by the adaptive control, but the market of electric drives does not justify the expense it would be necessary to implement adaptive control in industrial drives. Such a propensity is also supported by the fact that performances of PI-based vector-controlled drives can be still improved. As the matter of fact, final users operate the standard self-commissioning, which industrial drives nowadays have. Sub-optimal performances are then improved by extra hand-calibrations that involve tedious and time-consuming tests.

In order to overcome this obstacle, self-commissioning can online optimize control system parameters, prior to the actual use of the system, which then uses these optimized controllers. This is also known in the research area of automatic control as auto-tuning or hardware-in-the-loop optimization [7]. This on-line optimization offers the advantages of the model-free design and makes self-commissioning very reliable because the controllers are experimentally evaluated.

This dissertation deals with the auto-tuning of electric drives based on Evolutionary Algorithms (EAs). The tuning of the control system implemented

in an electric drive is a multiobjective problem that involves a larger number of parameters in the presence of reasonable noise. Recently EAs have been successfully applied to this kind of optimization and have been proved to be more robust than classical techniques. EAs are search algorithms based on the mechanics of natural selection and genetics [8]. A recent and complete survey about their application in control system engineering is in [9], where the suitability of on-line EAs for robust AT is highlighted. Unfortunately during the on-line evolution of the control system, can need many tests, and the controlled process can be critically stressed by poorly performing solutions. For this reason the successful applications of on-line EAs are really limited in number.

In this dissertation, suggestions and innovative solutions are proposed to fully perform an on-line optimization without any risk for the hardware. On the one hand, a new real-time fitness implementation can halt the carrying out of an experiment, if a highly unsatisfactory solution is recognized. On the other hand, a new hybrid architecture integrates EA and simplex method in order speed up the convergence.

Parameters optimization comes out in a number of circumstances for a large class of problems. Industrial engineers recognize Matlab as one of the best program to face these problems in simulation, but still find difficulties in experimental context. From this point of view, the self-commissioning of electric drives can be considered just one application of the new online optimization proposed in this thesis. In this particular case study, following the proposed approach, manufacturers can embed this fully-automated tool into their drives without any extra-hardware. Moreover each final user of the

drive can concentrate on his own design specifications, letting the on-line GA find the most satisfactory solution, i. e. the best control system parameters for his design problem.

1.3 Thesis outline

The thesis is divided into the following chapters:

- the second chapter concerns the mathematical model of PMSMs in the stator and rotor reference frame, available variants of the vector control, their model based design, and finally their self-commissioning.
- Chapter three describes the proposed configuration of EAs for online optimization.
- In chapter four, the new hybrid architecture, which integrates EA and simplex method in order speed up the convergence, is proposed.
- In chapter five, experimental results are shown to prove the effectiveness of the proposed technique.

Chapter 2

PMSM Drive

Although the ac system of production and distribution of electrical energy were well established by the early twentieth century, only induction motors were commonly employed in the industry. PMSMs provide torque at synchronous speed, it means that their working speed depends on the frequency of the power source, e.g., with a 50 Hz-line a PMSM could work at 3000, 1500... rpm according to its number of pole pairs. For the motor startup, the rotor, besides magnets, needed a squirrel cage to produce torque at zero speed. This resulted in higher manufacturing costs discouraging the spreading of PMSMs which remained on the fringe of the scene for long time. Up to the eighties the induction motor were largely used for constant-speed applications and the dc motor dominated the market of variable-speed applications, but with advances of power electronics, new drives capable of feeding motors with variable-frequency voltages were available and changed the trend of the market. Thanks to electric drives, PMSMs could be directly used at any speed and ac motors began to replace dc ones for servo applications.

Particularly PMSMs are superseding dc motors assuring:

- higher efficiency and longer life,
- higher speed and power density,
- smaller size and better heat transfer,
- less noise and no spark,

and struggle for bigger slice of the market over induction motors assuring:

- higher efficiency and performance for applications up to 10 kW,
- higher power density and power factor,
- smaller size and better heat transfer.

Furthermore, PMSM success goes on also thanks to advances of PMs manufacturing made over last years, e.g., rare earths, such as neodymium-iron-boron, nowadays can be bought for about 20 USD per kg, replacing the older barium ferrite and Alnico. The production of better and better materials for PMs not only matches the increasing demand of classical PMSMs, i.e., radial-flux machines, but allows new solutions such as the axial-flux machines.

2.1 PMSM configurations

The radial flux PMSMs can be classified according to their rotor configuration. As shown in fig. [2.1](#) there are four different ways of mounting the magnets on the rotor. Each of them has its own characteristic.

The surface mounted magnets are simpler to be placed on the rotor, and the reluctance effects are negligible resulting in a low torque ripple. On the other hand, this solution requires high accuracy in its design to prevent magnet damaging when the rotor is placed into the stator. The airgap length

needs to be very small to increase the stator inductance and reduce the flux leakage, i.e., the part of the flux which does not cross the airgap. The surface mounted magnets are suitable for low speed applications. As shown in fig. [2.1b](#), surface mounted PMSMs can be regarded as round-rotor machines, in each direction the effective airgap consists of the actual airgap and that created by the magnets, whose permeability is almost that of the air. As a consequence

$$L_{sd} \approx L_{sq} \quad (2.1)$$

where L_{sd} and L_{sq} are the d- and q-axis inductance respectively.

Mechanical robustness of rotors with inset magnets, or even more with buried ones, makes these motors suitable for high speed applications. In this case, as shown in fig. [2.1d](#), the motor cannot be considered a round rotor machine. The effective airgap of q-axis magnetic path is only due to the actual airgap, whereas for d-axis magnetic path also the PM has to be considered. Consequently

$$L_d \leq L_q. \quad (2.2)$$

Further details about permanent-magnet synchronous motor design can be found in [\[10\]](#).

2.2 PMSM model

Fig. [2.2](#) illustrates the scheme of a permanent-magnet synchronous motor which the machine analytical model is based on. For sake of clarity, it refers to a machine with 3-phase stator windings and one pole pair, but the equations

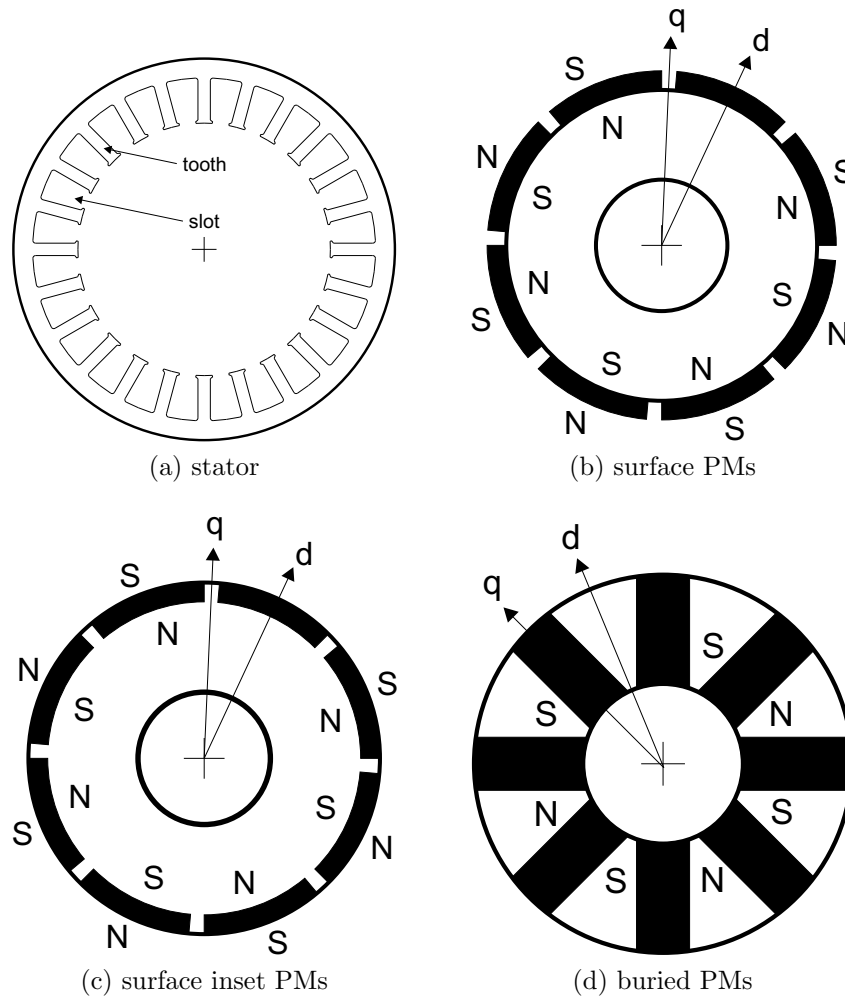


Figure 2.1: PMSM with different rotor configurations.

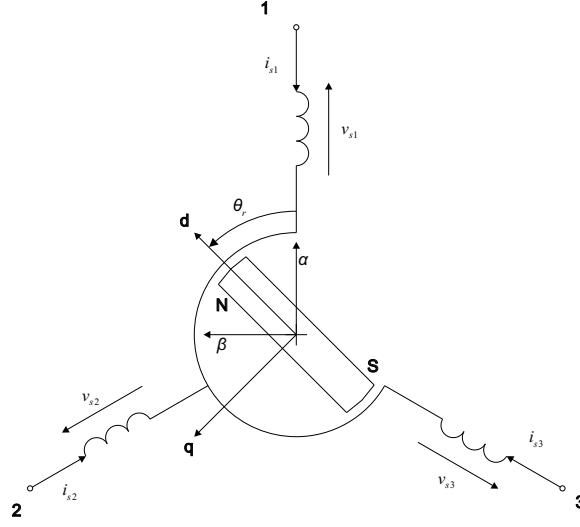


Figure 2.2: PMSM analytical model.

reported in this section are valid for any PMSM. For a exhaustive treatment of the machine analytical models please refer to [11].

2.2.1 Stator reference frame

Variables which refer to the stator reference frame will be indicated by subscript 123. Adopting a matrix form, one gets

$$\mathbf{v}_{s,123} = \mathbf{R}_s \mathbf{i}_{s,123} + \frac{d\boldsymbol{\Psi}_{s,123}}{dt} \quad (2.3)$$

where the vectors of stator voltages, stator currents, and flux linkages are respectively:

$$\mathbf{v}_{s,123} = \begin{bmatrix} v_{s1} & v_{s2} & v_{s3} \end{bmatrix}^T \quad (2.4)$$

$$\mathbf{i}_{s,123} = \begin{bmatrix} i_{s1} & i_{s2} & i_{s3} \end{bmatrix}^T \quad (2.5)$$

$$\Psi_{s,123} = \begin{bmatrix} \Psi_{s1} & \Psi_{s2} & \Psi_{s3} \end{bmatrix}^T. \quad (2.6)$$

The vector of flux linkages, equation (2.6), can be also written as function of the currents

$$\Psi_{s,123} = (\mathbf{L}_{ls} + \mathbf{L}_m + \mathbf{L}_{\Delta m}) \mathbf{i}_{s,123} + \Psi_{PM,123} \quad (2.7)$$

where

$$\mathbf{L}_{ls} = \begin{bmatrix} L_{ls} & 0 & 0 \\ 0 & L_{ls} & 0 \\ 0 & 0 & L_{ls} \end{bmatrix} \quad (2.8)$$

$$\mathbf{L}_m = \begin{bmatrix} L_m & -\frac{1}{2}L_m & -\frac{1}{2}L_m \\ -\frac{1}{2}L_m & L_m & -\frac{1}{2}L_m \\ -\frac{1}{2}L_m & -\frac{1}{2}L_m & L_m \end{bmatrix} \quad (2.9)$$

$$\mathbf{L}_{\Delta m} = L_{\Delta m} \begin{bmatrix} \cos 2\theta_r & \cos(2\theta_r - \frac{2}{3}\pi) & \cos(2\theta_r + \frac{2}{3}\pi) \\ \cos(2\theta_r - \frac{2}{3}\pi) & \cos(2\theta_r - \frac{4}{3}\pi) & \cos 2\theta_r \\ \cos(2\theta_r + \frac{2}{3}\pi) & \cos 2\theta_r & \cos(2\theta_r + \frac{4}{3}\pi) \end{bmatrix} \quad (2.10)$$

$$\Psi_{PM,123} = \Psi_{PM} \begin{bmatrix} \cos \theta_r & \cos(\theta_r - \frac{2}{3}\pi) & \cos(\theta_r - \frac{4}{3}\pi) \end{bmatrix} \quad (2.11)$$

are the matrices of stator leakage and magnetization inductances and the vector of the flux linkages created by permanent magnets, respectively; and

$$L_m = \frac{1}{3} \left(\frac{N_s^2}{R_{md}} + \frac{N_s^2}{R_{mq}} \right) \quad (2.12)$$

$$L_{\Delta m} = \frac{1}{3} \left(\frac{N_s^2}{R_{md}} - \frac{N_s^2}{R_{mq}} \right) \quad (2.13)$$

where N_s is the turns number of the stator windings, R_{md} and R_{mq} are d- and q-axis reluctances, respectively.

For round-rotor machines the above equations get simpler because

$$R_{md} = R_{mq} \Rightarrow L_{\Delta m} = 0, \quad (2.14)$$

and the electromagnetic torque is

$$T_{em} = N_p \Psi_{PM} \left(i_{s1} \sin \theta_r + i_{s2} \sin \left(\theta_r - \frac{2}{3}\pi \right) + i_{s3} \sin \left(\theta_r - \frac{4}{3}\pi \right) \right) \quad (2.15)$$

2.2.2 Rotor reference frame

Variables which refer to the rotor reference frame will be indicated by subscript dq . The zero-component equation is omitted because is trivial. Applying Park's transformation to stator reference equations one gets

$$\mathbf{v}_{s,dq} = R_s \mathbf{i}_{s,dq} + \omega_r \mathbf{J} \Psi_{s,dq} + \frac{d\Psi_{s,dq}}{dt} \quad (2.16)$$

where the stator resistance is R_s and the vectors of stator voltages, stator currents, and flux linkages are respectively:

$$\mathbf{v}_{dq} = \begin{bmatrix} v_{sd} & v_{sq} \end{bmatrix}^T, \quad (2.17)$$

$$\mathbf{i}_{dq} = \begin{bmatrix} i_{sd} & i_{sq} \end{bmatrix}^T, \quad (2.18)$$

$$\Psi_{dq} = \begin{bmatrix} \Psi_{sd} & \Psi_{sq} \end{bmatrix}^T, \quad (2.19)$$

and the matrix \mathbf{J} , which replaces the j operator when the complex notation is used, is defined as follows

$$\mathbf{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \quad (2.20)$$

The vector of flux linkages, eq. (2.19), can be also written as function of the currents

$$\Psi_{s,dq} = \mathbf{L}_{s,dq} \mathbf{i}_{s,dq} + \Psi_{PM,dq} \quad (2.21)$$

where

$$\Psi_{PM,dq} = \begin{bmatrix} \Psi_{PM} \\ 0 \end{bmatrix} \quad (2.22)$$

$$\mathbf{L}_{s,dq} = \begin{bmatrix} L_{sd} & 0 \\ 0 & L_{sq} \end{bmatrix}. \quad (2.23)$$

Combining equations (2.16), (2.22), and (2.23), one obtains

$$\mathbf{v}_{s,dq} = R_s \mathbf{i}_{s,dq} + \omega_r \mathbf{J} \mathbf{L}_{s,dq} \mathbf{i}_{s,dq} + \mathbf{L}_{s,dq} \frac{d\mathbf{i}_{s,dq}}{dt} + \omega_r \mathbf{J} \Psi_{PM,dq}, \quad (2.24)$$

that can be also rewritten for sake of clarity as

$$\mathbf{v}_{s,dq} = \begin{bmatrix} R_s & -\omega_r L_{sq} \\ \omega_r L_{sd} & R_s \end{bmatrix} \mathbf{i}_{s,dq} + \begin{bmatrix} L_{sd} & 0 \\ 0 & L_{sq} \end{bmatrix} \frac{d\mathbf{i}_{s,dq}}{dt} + \omega_r \begin{bmatrix} 0 \\ \Psi_{PM} \end{bmatrix}. \quad (2.25)$$

The electromagnetic torque is given by

$$T_{em} = \underbrace{\frac{3}{2}N_p(L_{sd} - L_{sq})i_{sd}i_{sq}}_{\text{reluctance torque}} + \underbrace{\frac{3}{2}N_p\Psi_{PM}i_{sq}}_{\text{magnet torque}}, \quad (2.26)$$

where the reluctance torque assumes more importance for PMSMs with inset and buried magnets. For round-rotor machines the reluctance torque is negligible, see equation (2.1), and the electromagnetic torque formula can be further simplified as

$$T_{em} = \frac{3}{2}N_p\Psi_{PM}i_{sq}. \quad (2.27)$$

2.3 Vector control of a PMSM

The principle behind the vector control is to separately control the components of the stator current vector in the rotor reference frame in order to control the flux and torque of the motor. In this way the PMSM can be seen as a separately-excited dc motor in which the armature and field currents are equivalent to the d- and q-axis stator currents. From equation (2.26), there is evidence that a given value of torque can be obtained by suitably regulating both d- and q-axis components of the stator current vector. It means that the torque has one degree of freedom, i.e., a further criterion can be used to share the stator current between i_{sd} and i_{sq} in order to produce a given value of torque.

There are different criterion to implement the vector control. The first is the zero d-axis current (ZDAC) control, which will be described in this section. It is one of the most widely used in the industry and forces the torque to

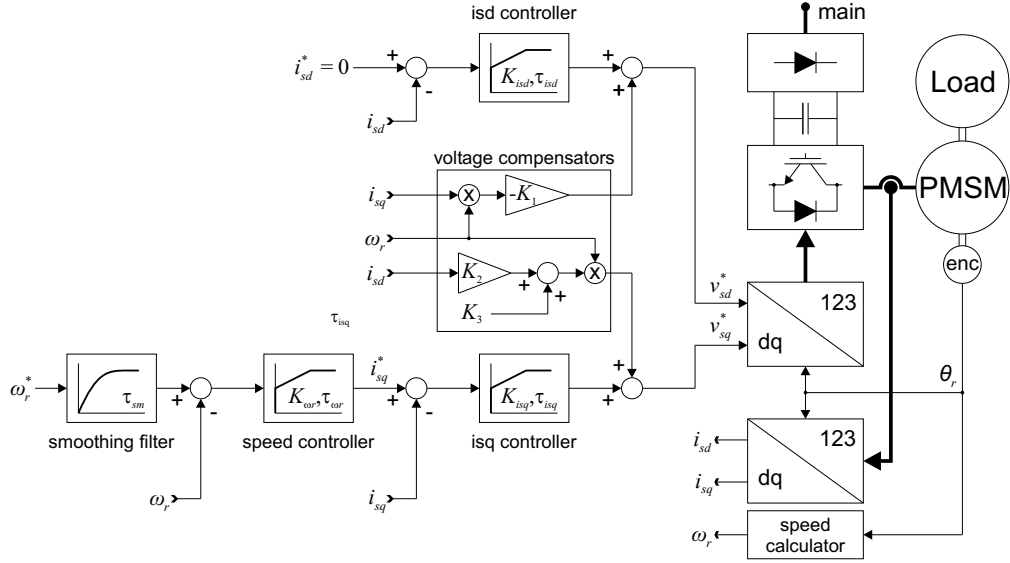


Figure 2.3: Block diagram of zero d-axis current control of a PMSM.

be proportional to current magnitude for the PMSM [12, 13]. For applications in which efficiency is essential, the maximum torque per unit current and maximum efficiency variants has to be considered. The former produces maximum torque per unit current by minimizing copper losses [14, 15], instead of the latter which minimizes net losses [16, 17]. The unity power factor control allows a better utilization of the inverter [12, 15]. The constant mutual flux linkages control [12, 15] maintains the magnet flux linkages value in the core to avoid saturation.

As it will be described later, a ZDAC controlled surface mounted PMSM has been used in the experimental setup and its block diagram is illustrated in fig. 2.3. It has to be noted that for this type of motors, the ZDAC control is equivalent to the maximum torque per unit current control, equation (2.27).

The PMSM is fed by a voltage source inverter (VSI). The voltage reference

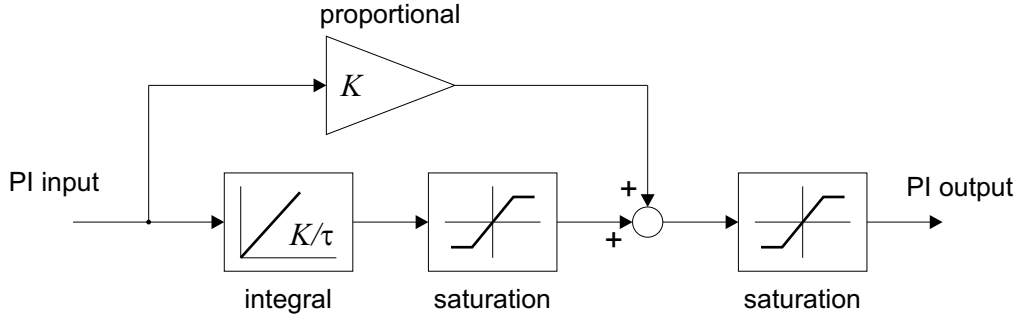


Figure 2.4: Block diagram of the anti-windup PI.

in the stator coordinates $\mathbf{v}_{s,123}^*$ is obtained as follows

$$\mathbf{v}_{s,123}^* = \underbrace{\begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}}_{(123) \leftarrow (\alpha\beta)} \underbrace{\begin{bmatrix} \cos \theta_r & -\sin \theta_r \\ \sin \theta_r & \cos \theta_r \end{bmatrix}}_{(\alpha\beta) \leftarrow (dq)} \mathbf{v}_{s,dq}^* \quad (2.28)$$

Both the d- and q-axis voltage references, v_{sd}^* and v_{sq}^* , are provided partly by current controllers, partly by voltage compensators. The d- and q-axis current controllers are two anti-windup PIs.

The anti-windup PI has been implemented as linear PI with two saturations on the integrator and sum outcomes. The block diagram is illustrated in fig. 2.4, and when it is working in the linear zone, saturations can be not considered, and the transfer function is

$$K \frac{1 + \tau s}{\tau s} \quad (2.29)$$

where K and τ are the proportional gain and the time constant of the PI.

As regards the d- and q-axis current controllers the two transfer functions

are respectively:

$$K_{isd} \frac{1 + \tau_{isd}s}{\tau_{isd}s} \quad (2.30)$$

and

$$K_{isq} \frac{1 + \tau_{isq}s}{\tau_{isq}s}. \quad (2.31)$$

The d- and q-axis voltage compensators, whose feedforward actions are:

$$- \omega_r i_{sq} K_1 \quad (2.32)$$

and

$$\omega_r (i_{sd} K_2 + K_3) \quad (2.33)$$

respectively, improve the performance of the current control loops by reducing the influence of the cross coupling terms in (2.25).

The reference of the q- axis current is provided by the speed controller, an anti-windup PI with transfer function

$$K_{\omega r} \frac{1 + \tau_{\omega r}s}{\tau_{\omega r}s} \quad (2.34)$$

The speed reference is prefiltered before being passed to the speed loop, to reduce the overshoot and settling associated with step demands, whilst retaining a fast disturbance rejection. This is achieved using a first-order low-pass filter (FOLPF), namely smoothing filter

$$\frac{1}{1 + \tau_{sm}s} \quad (2.35)$$

whose time constant has to be tuned according to the speed loop bandwidth.

2.4 Model-based design of the vector control

In this section will be described a way to design the vector control according to suggestions given by literature [15, 18, 19] and supposing the knowledge of motor and hardware parameters. For this reason this kind of design relies on the accuracy with which motor and hardware models are known and belongs to model-based design class. The design of the vector control consists of tuning of the ten parameters above mentioned: K_{isd} , τ_{isd} , K_{isq} , τ_{isq} , $K_{\omega r}$, $\tau_{\omega r}$, τ_{sm} , K_1 , K_2 and K_3 .

Initially, last three parameters can be set as:

$$K_1 = L_{sq}, \quad (2.36)$$

$$K_2 = L_{sd}, \quad (2.37)$$

and

$$K_3 = \Psi_{PM} \quad (2.38)$$

to completely remove cross coupling actions in (2.25).

2.4.1 q-axis current control loop

Thanks to feedforward actions which compensate cross coupling terms, q-axis current control loop gets simpler as illustrated in fig. 2.5. The control

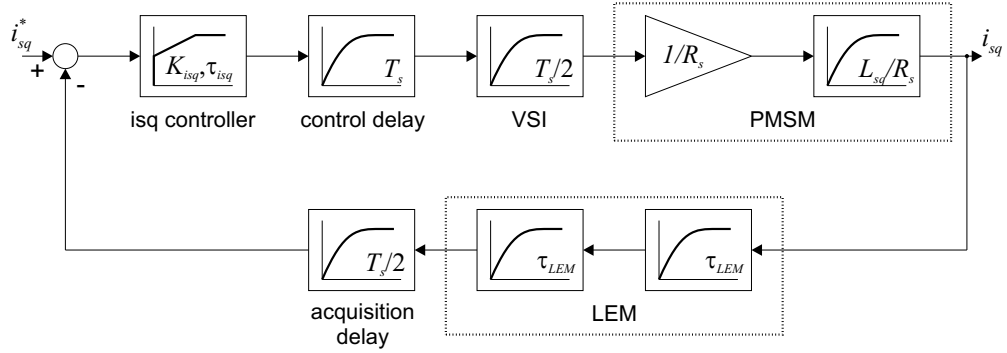


Figure 2.5: Block diagram of the q-axis current control loop.

algorithm is implemented with a microprocessor board which works with a sample time T_s . It means that after a clock edge, the microprocessor takes one sample time to produce the control action for the inverter. Such a delay is taken into account by means of the FOLPF that follows the controller. Next there is a further FOLPF modelling the delay due to the inverter, and then the motor. Regarding the current feedback, LEM component is modelled as a double FOLPF with time constant τ_{LEM} . Finally the last delay is due to the A/D converter.

In order to make simpler block diagram with unitary feedback, let us approximate the several little delays in the loop as one FOLPF with time constant

$$\tau_{\Sigma i} = \frac{T_s}{2} + \frac{T_s}{2} + T_s + 2\tau_{LEM}, \quad (2.39)$$

and the two zeros due to feedback as an initial FOLPF with one negative time constant

$$(1 + 2\tau_{LEM}s) \left(1 + \frac{T_s}{2}s\right) \approx \frac{1}{1 - \left(\frac{T_s}{2} + 2\tau_{LEM}\right)s} \quad (2.40)$$

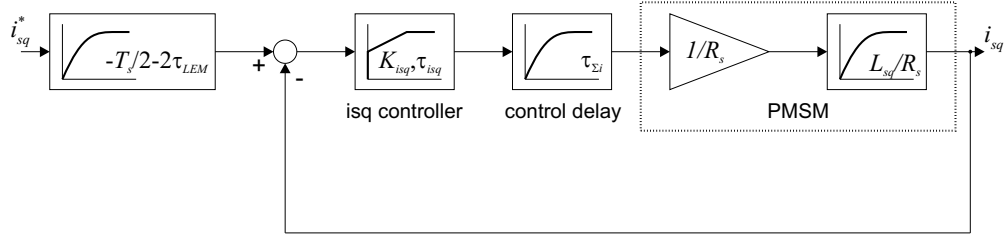


Figure 2.6: Equivalent block diagram of q-axis current control loop after approximations.

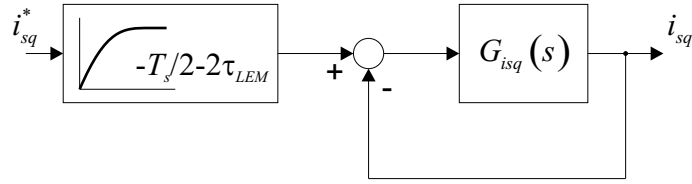


Figure 2.7: Final equivalent block diagram of q-axis current control loop.

as shown in fig. 2.6. Applying the Absolute Value Optimum (AVO) criterion, PI constants can be calculated as follows:

$$K_{isq} = \frac{R_s \tau_{isq}}{2T_{\Sigma i}} \quad (2.41)$$

and

$$\tau_{isq} = \frac{L_{sq}}{R_s} \quad (2.42)$$

and the q-axis current loop results in the diagram in fig. 2.7 where

$$G_{isq}(s) = \frac{1}{2\tau_{\Sigma i} s (1 + s\tau_{\Sigma i})}. \quad (2.43)$$

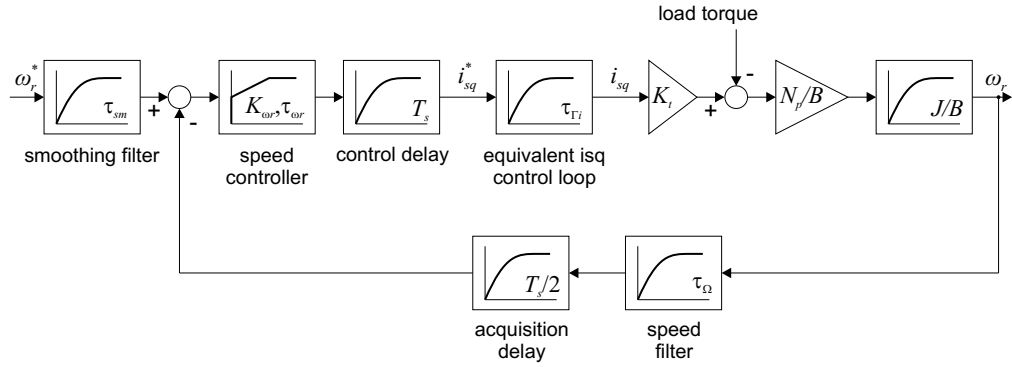


Figure 2.8: Block diagram of the speed control loop.

2.4.2 Speed control loop

In fig. 2.8, the equivalent block diagram of the speed control loop is shown. It has to be noted that the constant gain, $K_t = \frac{3}{2}N_p\Psi_{PM}$, is used to get the electromagnetic torque from i_{sq} , see equation (2.27). The whole current control loop has been modelled as a FOLPF with time constant:

$$\tau_{\Gamma i} = 2\tau_{\Sigma i} - \frac{T_s}{2} - 2\tau_{LEM}. \quad (2.44)$$

Also in this case the idea is to get a simpler equivalent block diagram as shown in fig. 2.9, by making following approximations. The several little delays in the loop are modelled as one FOLPF with time constant

$$\tau_{\Sigma\omega} = \frac{T_s}{2} + \tau_{\Omega} + T_s + \tau_{\Gamma i}. \quad (2.45)$$

The viscous friction is assumed to be negligible

$$B \approx 0 \quad (2.46)$$

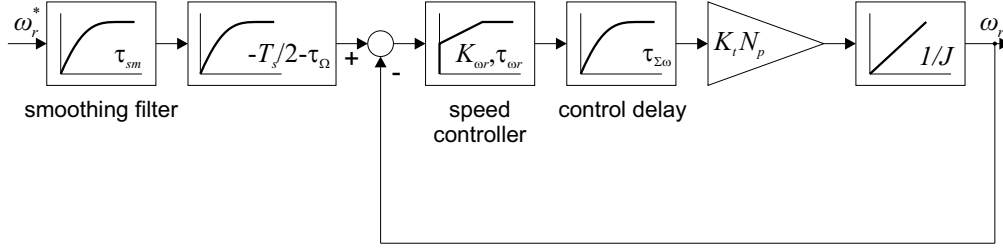


Figure 2.9: Block diagram of the simplified speed control loop.

and

$$\left(1 + \frac{T_s}{2}s\right) (1 + \tau_\Omega s) \approx \frac{1}{1 - \left(\frac{T_s}{2} + \tau_\Omega\right)s}. \quad (2.47)$$

Applying the Symmetrical Optimum (SO) criterion, PI and smoothing filter constants can be calculated as follows:

$$K_{\omega r} = \frac{J}{K_t N_p 2\tau_{\Sigma\omega}}, \quad (2.48)$$

$$\tau_{\omega r} = 4\tau_{\Sigma\omega}, \quad (2.49)$$

and

$$\tau_{sm} = 4.8\tau_{\Sigma\omega}. \quad (2.50)$$

In this way the whole speed control loop can be view as in fig. ?? where

$$G_\omega(s) = \frac{1 + 4T_{\Sigma\omega}s}{8T_{\Sigma\omega}^2 s^2 (1 + T_{\Sigma\omega}s)}. \quad (2.51)$$

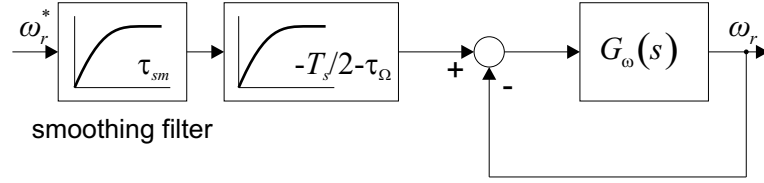


Figure 2.10: Block diagram of the q-axis current control loop.

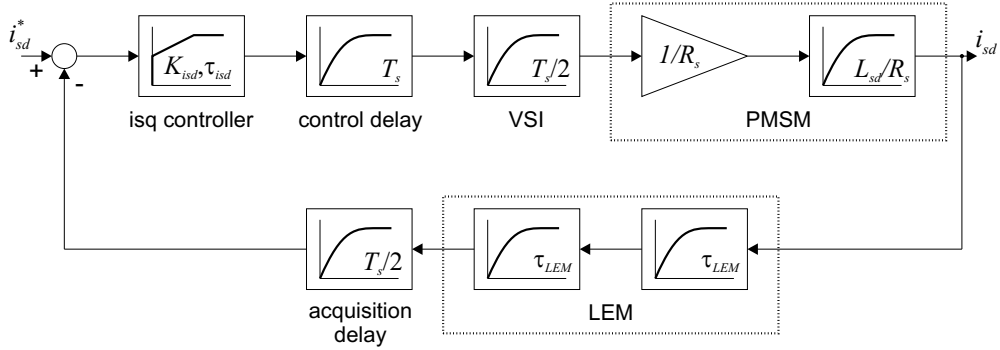


Figure 2.11: Block diagram of the d-axis current control loop.

2.4.3 d-axis current control loop

As regards the design of the d-axis current controller, one can follow the same procedure adopted for the q-axis current control loop. This is because the equivalent scheme of d-axis current control loop is the same than that of the q-axis one, except for the time constant of the motor which is L_{sd}/R_s instead of L_{sq}/R_s . Obviously for round-rotor machine, d- and q-axis current control loops would be identical as obtained by matching figures [2.11](#) and [2.5](#).

As described in subsection [2.4.1](#), making similar approximations and applying the Absolute Value Optimum (AVO) criterion, PI constants can be calculated as follows:

$$K_{isq} = \frac{R_s \tau_{isd}}{2T_{\Sigma i}} \quad (2.52)$$

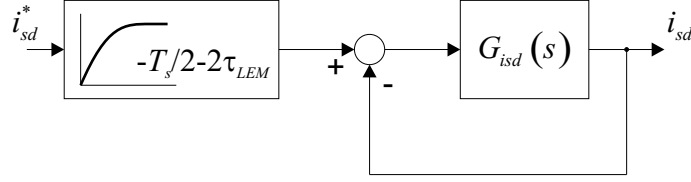


Figure 2.12: Final equivalent block diagram of d-axis current control loop.

and

$$\tau_{isd} = \frac{L_{sd}}{R_s} \quad (2.53)$$

where

$$\tau_{\Sigma i} = \frac{T_s}{2} + \frac{T_s}{2} + T_s + 2\tau_{LEM}. \quad (2.54)$$

The d-axis current loop results in the diagram in fig. [2.12](#) where

$$G_{isd}(s) = \frac{1}{2\tau_{\Sigma i}s(1 + s\tau_{\Sigma i})}. \quad (2.55)$$

2.5 Self-commissioning

As highlighted in the previous section, the fact that one can design the vector control by applying some simple formula is an unquestionable advantage of the model-based design. On the other hand, the higher the accuracy of values of motor and hardware parameters is, the better the design results. Consequently, a prior parameters identification becomes crucial. Furthermore, such a procedure could be ideally automatic and available on drives. Starting from such an idea, a considerable part of the research has been recently dedicated to the so-called self-commissioning. Referring to [\[6\]](#) for a more precise def-

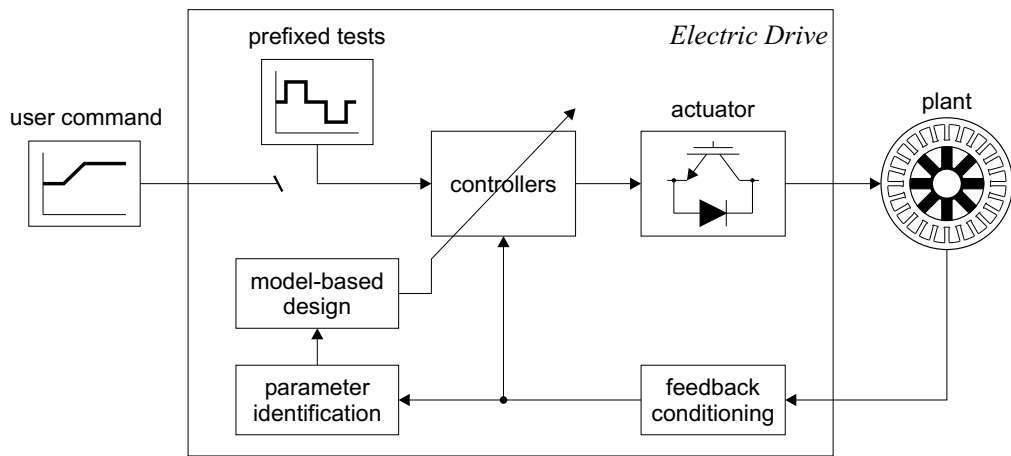


Figure 2.13: Model-based self-commissioning procedure.

initiation, the self-commissioning consists of an automatic procedure for the tuning of controllers based on a previous parameter identification, when a motor is initially connected to the drive, figures 2.13 and 2.14. For industrial use, it has to be implemented without any extra-hardware and with only the knowledge of machine ratings (from motor nameplate). In this way the final user can utilize the drive with a third-party motor without any hand-tuning. According to literature related to the research area of automatic control, this subject is known as auto-tuning [7].

In this section, main contributions given by researchers will be dealt to describe the state of the art of self-commissioning of electric drives. Although for commercial reasons, most papers expressly deal with self-commissioning for induction motor, they are very interesting also for permanent-magnet synchronous motors because same concepts can be quickly readapted.

Initially self-commissioning was proposed by Schierling in [20]. The author describes a specific sequence of tests at standstill in order to measure electrical parameters of an induction motor. For the determination of the total

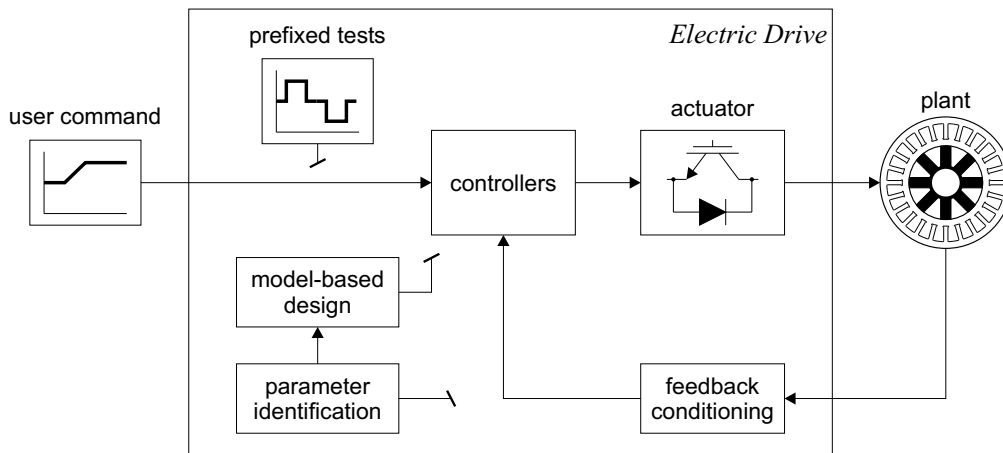


Figure 2.14: Normal working after self-commissioning.

leakage inductance, two d-axis voltage steps are applied and the stator current response is analyzed. These steps must be applied for a time much shorter than the rotor time constant. Then the rotor and stator resistances and rotor time constant are determined by impressing different values of d-axis current. Unfortunately this paper is poor of details, for instance it leaves in doubt about the length of the voltage steps. Such a length depends on the rotor time constant which is not yet identified. Moreover the identification of mechanical parameters is only touched on.

Self-commissioning was dealt with more details some year later in [21] and then in [22]. The former, after a concise survey on vector control techniques at that time available for industrial drive, shows how to implement an automatic procedure to identify both electrical and mechanical parameters in the following order:

- stator resistance is determined from the ratio between the dc voltage applied and the dc current measured at standstill,
- stator transient time constant is obtained from the time and magnitude of

a current spike due to an applied voltage spike at standstill,

- current controllers are set,
- rotor time constant is determined by the analyzing the current response when the inverter is turned off after that the flux is built up,
- rotor magnetizing current is obtained from the no-load current when the motor is operated with the v/f open-loop control,
- flux controller is set,
- mechanical time constant is calculated from the acceleration time when the motor is operated from zero speed to a prefixed speed value by applying a constant torque.

It has to be emphasized that this procedure carries out the self-commissioning of a 25 kVA induction motor drive in 60 s.

Also in [22], electrical parameters are measured with tests at standstill, whereas mechanical parameters are identified by open loop runup and run-down tests; but it is underlined the importance of recursive algorithm and multiple tests to achieve a good level of identification accuracy. Moreover a significant contribution was in extra closed-loop tests to fine tune the rotor and torque time constants under varying operating conditions. The authors describe how the drive can work well in one operating condition, but that changed conditions lead to problems. Obviously, no problem arises when the drive works over a small operating condition range in which the commissioning has been performed. To enlarge this range, special care has to be taken in regards of the rotor time constant, rather than PI gains, which needs to be updated according to condition changing.

Following this idea many researchers focused mostly on more accurate and

quicker procedures for rotor time constant identification over a wide range of operating condition [23, 24, 25]. A simple auto-tuning strategy for the rotor time constant was proposed in [26]. The principle is simple and effective: three samples of the speed profile generated by a torque command are recursively used for the on-line correction. The further and very interesting step was to develop and generalize such a auto-tuning strategy to two-degree-of-freedom controllers, such as PIs, and apply to self-commissioning [27]. Unfortunately description in [27] lacks details. As regards PMSMs, an effective solution for self-commissioning has been proposed in [28], where the PI design is based on the parameter measurement using adaptive identification. In conclusion, after an initial success, the model-based self-commissioning of PI-based vector control schemes has unveiled its limitations. Such limitations are partly inherited by the simple control structure and partly by the model-based design. Relative to the control structure, it is a cascade control with linear regulators, such as PIs, and a feedforward action, whereas the motor is a nonlinear system. About the model-based design, see section 2.4, many approximations are made and issues concerning the multiple inputs, system nonlinearities and uncertainties still remain unaddressed. Although good enough for several applications, such a self-commissioning leads only to a sub-optimal tuning.

Possible solution are given by the nonlinear, adaptive, and intelligent control [7, 29, 30, 31]. Nonlinear control developed techniques, for instance, based on fuzzy logic [32] or sliding-mode control [33], to improve the robustness of the control scheme to parameter variations. Also the more advanced adaptive control was applied to electric drives in several ways. The simplest is

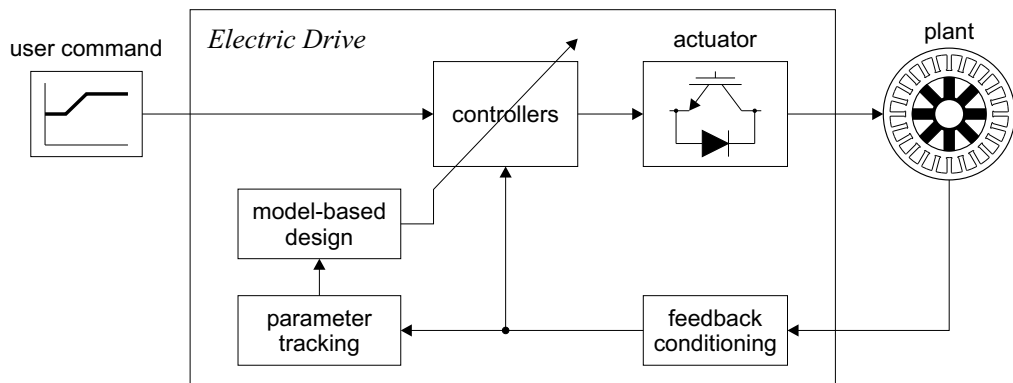


Figure 2.15: Adaptive control with parameter tracking.

illustrated in fig. 2.15, see [22, 23]. During the normal working of the drive, a recursive algorithm, namely parameter tracking, keeps updated the parameter estimate. This is used to reapply, at each sample time, the design rules for the controller tuning. The accuracy of parameter values is maintained regardless operating conditions, but this meets only one of the the problems related to model-based design letting the others unaddressed. In fig. 2.16 is shown the Model-Reference Adaptive System (MRAS), [34]. The reference model produces the output with which the plant should ideally respond to the command. The mismatch between reference and actual output drives the parameter adjustment. Realizing the outer adaptation loop, MRAS is very powerful and flexible because fully bypasses problems related to model-based design. Unfortunately, as often happens, powerfulness and flexibility involve some risk. For instance, adaptation gain is not simple to set and stability region is hard to be found. Finally implementation is harder because of its complexity.

Before attempting to replace a well-known PI control with a sophisticate adaptive control, one should investigate whether it has been made the most

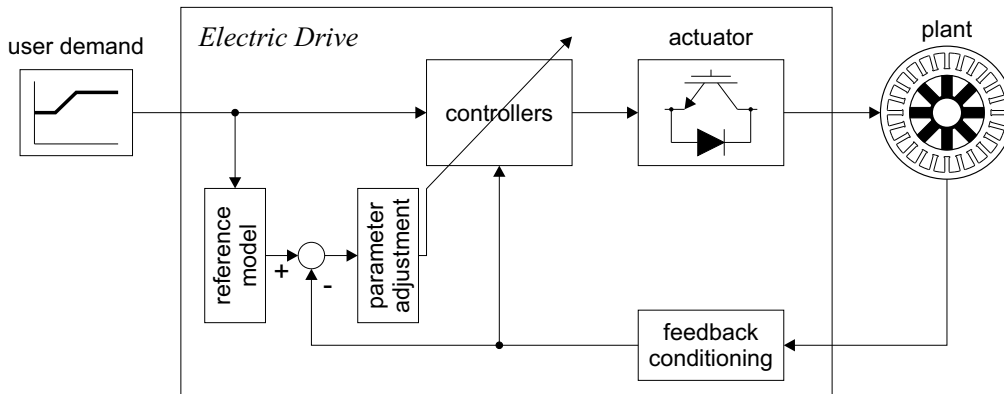


Figure 2.16: Model-Reference Adaptive System (MRAS).

of the simpler PI control [35]. About particular applications, such as missile guidance, adaptive control has been necessary. On the contrary, the market of electric drives did not justify the expense it would be necessary to implement adaptive control in industrial drives. Such a propensity is also supported by the fact that performances of PI-based vector-controlled drives can be still improved. As the matter of fact, final users operate the standard self-commissioning, which industrial drives nowadays have. Sub-optimal performances are then improved by extra hand-calibrations that involve tedious and time-consuming tests.

Considering that almost all industrial drives are PI-based vector-controlled, and that human time is more expensive than machine time, in this dissertation a new self-commissioning scheme is proposed in fig. 2.17. From a comparison with the standard, see fig. 2.13, it has to be noted that the controller auto-tuning is driven by the performance evaluation. As for adaptive control, it completely avoids model-based issues. Further, either prefixed tests or user tests can be used during auto-tuning. The core of this scheme

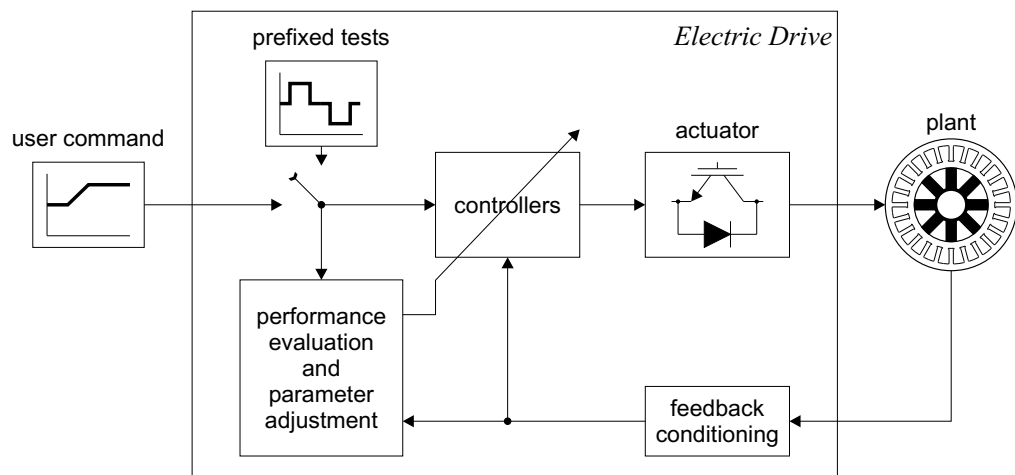


Figure 2.17: Final equivalent block diagram of d-axis current control loop.

is in the mechanism for parameter adjustment. In the next chapters the algorithms which it is based on will be fully detailed.

Chapter 3

Evolutionary Algorithms

Evolutionary algorithms (EAs) are inspired by concepts of evolution and adaptation discovered and studied by Charles Darwin in the nineteenth century. According to Darwin, the natural selection is the main mechanism which evolution is based on.

A population can survive in a given environment if enough individuals are able to reproduce. The principle of natural selection states that the higher the capability of adaptation of an individual, the higher the probability of reproduction. It means that the better individuals are likely to transfer their genetic inheritance to offspring. Hence, throughout the evolution, new populations inherit better and better genetic material from previous ones. The process that governs sharing and transferring of genetic material from individuals to their offspring is called sexual reproduction. Furthermore, offspring's genes can be low-probabilistically altered by means of genetic mutation, also known as asexual reproduction. Such a process helps in maintaining a diverse population, i.e., a population where individuals with different genes

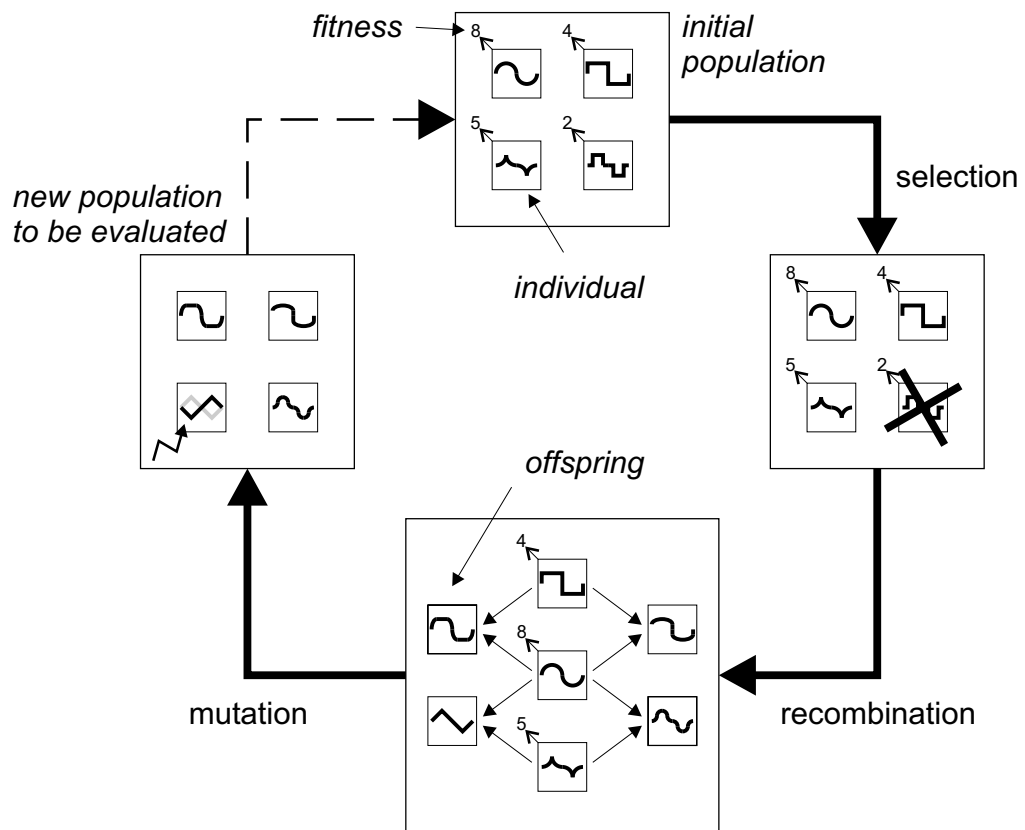


Figure 3.1: Interactive evolution cycle.

coexist. In this way natural selection has a wide-ranging genetic inheritance to operate on.

Figure 3.1 illustrates the iterative cycle above described which EAs are based on. An initial population is evaluated, it means that each individual receives a positive number indicating how capable of adaptation that individual is. Such a number is also known as fitness. According to their fitness, a part of the population is selected to reproduce. After genetic mutation, offspring create a new population ready to be evaluated. Thus the iterative cycle can restart.

Studies on the artificial intelligence began between the fifties and sixties.

This research area produced three of the most popular subjects in the history of computer science: neural networks, fuzzy logic, and evolutionary algorithms (EAs) also known as evolutionary computation (EC). The latter includes genetic algorithms (GAs), evolutionary strategies (ESs), genetic programming, evolutionary programming, and learning classifier systems, as shown in fig. 3.2. The concise classification of these areas needs to be handled carefully, because since the beginning all methods have been often used to contaminate each other.

GAs, initially proposed in [36], were made popularized by Holland in [37, 38]. A complete description of the implementation of natural mechanisms onto a computer was presented in [8, 39]. These highlight two key points of GAs, that are, an individual is binary-encoded into a bit-string representing the genes, and one of the most important resources of evolution consists of sexual reproduction, namely crossover. Furthermore, the Schema Theorem was proposed to force a GA in a theoretical framework. In the nineties, the Schema Theorem was subjected to criticism [40, 41], but received also favourable feedback by [42, 43]. However, controversial studies for a theoretical formalization did not play down the role of GAs in engineering optimization problems. Indeed, a further advance in this direction was made implementing their real-coded version [44, 45, 46]. The real coding, typical of ESs, is one of the features that makes GAs very attractive for optimization problems.

ESs were expressly developed to cope with parameter optimization [47]. The native real coding allows to associate directly each parameter to one gene. Such strategies rely on mutation and selection, rather than recombination, to

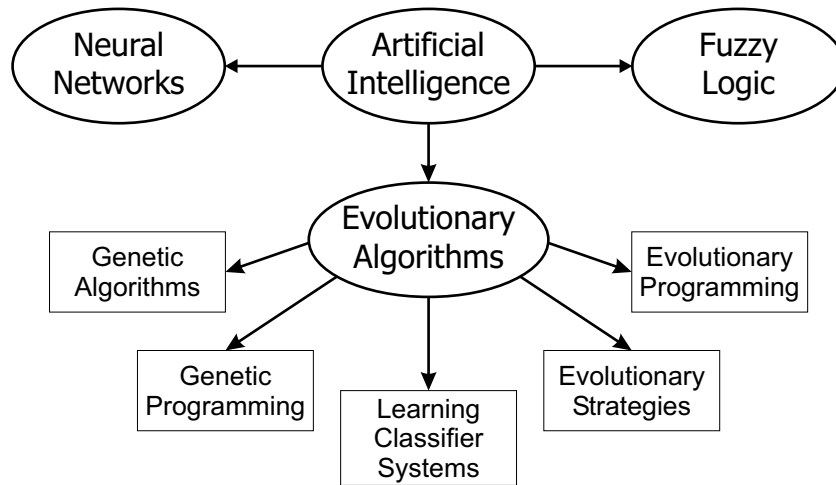


Figure 3.2: Research areas of artificial intelligence.

evolve a good solution. Last key point is the self-adaptation, for instance the mutation ratio changes throughout evolution according the evolution trend. Genetic programming can be view as a GA that operates on function and instruction sets, instead of binary-coded parameters, to optimize the code of a program [48]. Also learning classifier systems can be thought as GAs, but simple rules, such as *if <condition> else <action>*, are encoded as genes [49]. Regarding evolutionary programming, it was developed independently from ESs and is virtually not related to any coding, but real differences are hard to be singled out [50].

As aforementioned, EAs, specially GAs and ESs, have been successfully used for optimizations. Parameter optimization consists of searching of the best set of parameters which solves a given problem. The domain of feasible solutions is called search space. The goodness of a solution at solving the given problem is numerically evaluated by a fitness function. The main advantages provided by EAs can be summarized as follows:

- EAs do not get trapped into local minima of the fitness function because of their stochastic nature and because they do not use much local information.
- EAs work on a population of possible solutions which maintains diversity throughout evolution. This allows a wide exploration of the search space.
- EAs can cope with ill-behaved fitness functions in terms of multi-modality, discontinuity, and noise.

For these reasons EAs have an exploration ability which includes them in the class of global search methods. On the other hand, their robustness is obtained at the expense of a slow convergence rate that leads to high computational costs. Such a drawback can be partly solved customizing the configuration of EAs according to the given problem. This is because the choice of representation, search domain, evaluation, and evolutionary operators is problem-oriented. Considering the suggestions of literature on both EAs and electric drives, several configurations have been tested and the most satisfactory one will be described in this chapter.

As regards electric drives, this approach has been used in [51, 52, 53, 54]. The first of these papers has proposed the tuning of the PI speed controller of a brushless dc motor drive evaluating the response to a speed reference step through a very simple fitness function. In [53, 54] the potential of on-line GAs has been better exploited. In fact two controllers, for the current and speed loops of a dc motor drive, have been optimized. The evolution process in parallel tunes the controller parameters and chooses the controller structures cascading elementary controllers. In [52] a fuzzy logic controller for a dc motor drive has been genetically designed and compared with a PID controller under variable load conditions.

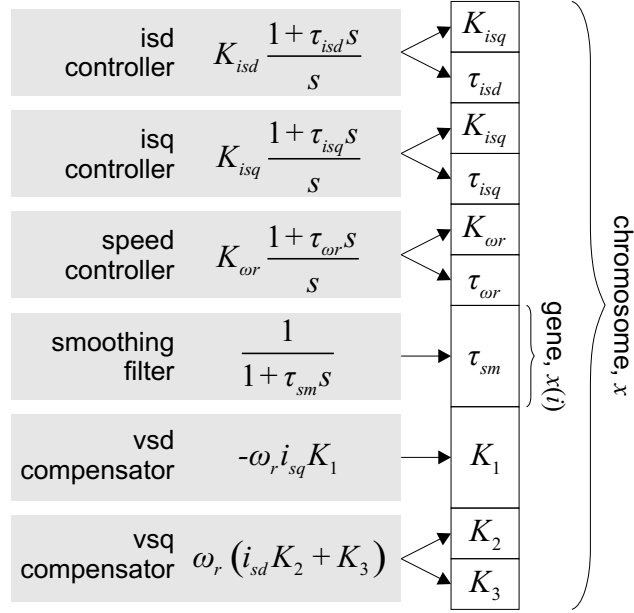


Figure 3.3: The parameters to be optimized, encoded as chromosome.

3.1 Representation

An individual, or potential solution to the design problem is denoted by \mathbf{x} and is encoded as chromosome, or string. The floating point representation has been chosen [45], hence the chromosome is a vector of floating point numbers, known as genes, one for each control parameter

$$\mathbf{x} = \left[K_{isd}, \tau_{isd}, K_{isq}, \tau_{isq}, K_{\omega_r}, \tau_{\omega_r}, \tau_{sm}, K_1, K_2, K_3 \right] \quad (3.1)$$

as shown in fig. 3.3.

3.2 Search Domain

There are two conflicting demands when the search domain is defined. The search domain has to be wide enough to ensure a good result, but the wider the search domain the slower the optimization. A priori knowledge can be helpful in order to assign a search interval for each parameter. For example, a PI optimization can start from an approximate solution and the searching can be limited to a suitable rectangle as depicted in fig. 3.4. Extending this concept to the 10-element vector which represents the chromosome (fig. 3.3), the search domain is a 10-dimensional hyper-rectangle given by the Cartesian product of the intervals which each parameter is limited to. The upper and lower bounds of each interval have been set as follows

$$\mathbf{x}_{lb}(i) = \mathbf{x}_0(i) (1 - \mathbf{LB}(i) / 100) \quad (3.2)$$

$$\mathbf{x}_{ub}(i) = \mathbf{x}_0(i) (1 + \mathbf{UB}(i) / 100) \quad (3.3)$$

where $\mathbf{UB}(i)$ and $\mathbf{LB}(i)$ are the upper and lower bounds in terms of percentage of $\mathbf{x}_0(i)$, and \mathbf{x}_0 is the vector of the parameter values of the sub-optimal solution obtained by the initial commissioning calculated in section II. Typical search ranges would be from $\pm 30\%$ to $\pm 70\%$ of pre-tuned control parameters [55], but these values are chosen as a compromise between the search time and best solution.

As a first step of the EA whose block diagram is shown in fig. 3.5, N_{ind} individuals are randomly sampled over the search domain to create the first population.

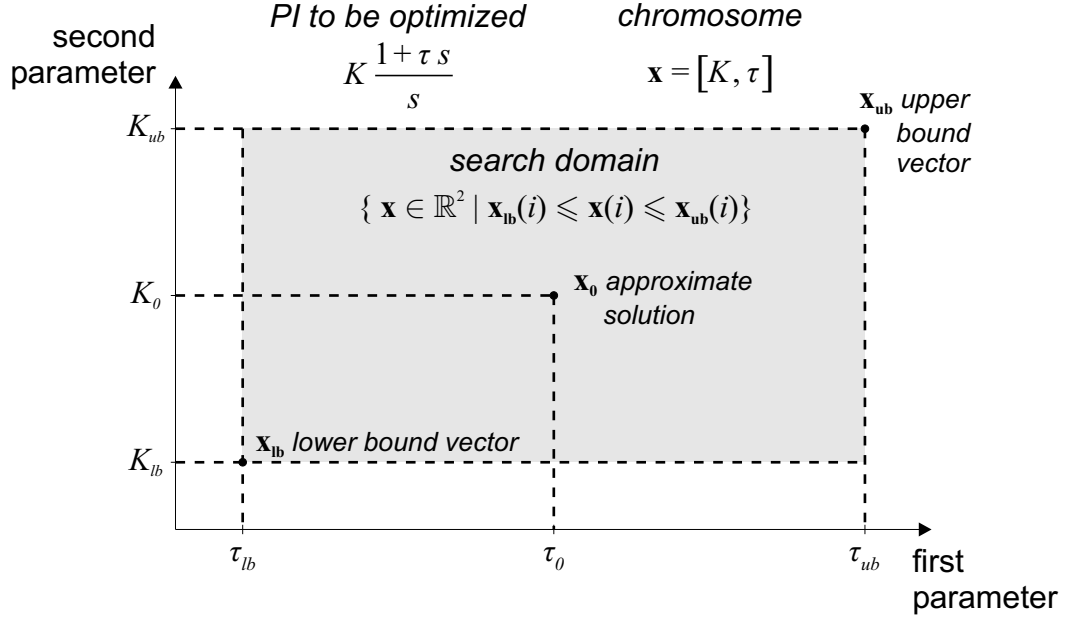


Figure 3.4: Search domain for a PI controller optimization.

3.3 Evaluation

The second step consists of the evaluation of how successful each individual is at solving the problem. The performance evaluation of a potential solution is a crucial point for every optimization method. Because of the nonlinearities of the system, more “best solutions” can be found according to the working conditions. Therefore, the training test and the objective function should be designed in order to consider the specific conditions in which the drive will operate during the normal working. For this reason, a good software for the self-commissioning should allow the final user of the drive to customize the training test and the objective function.

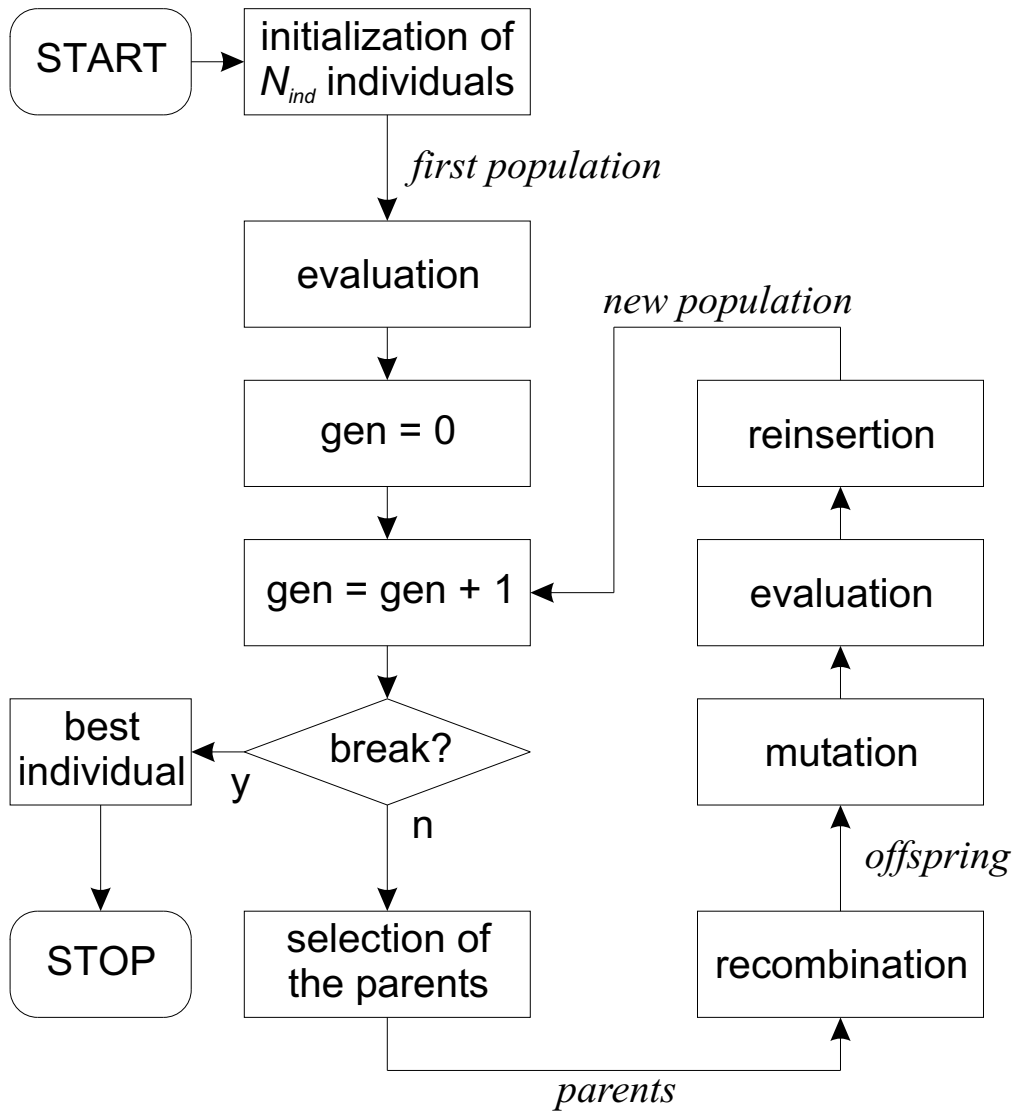


Figure 3.5: Block diagram of EAs.

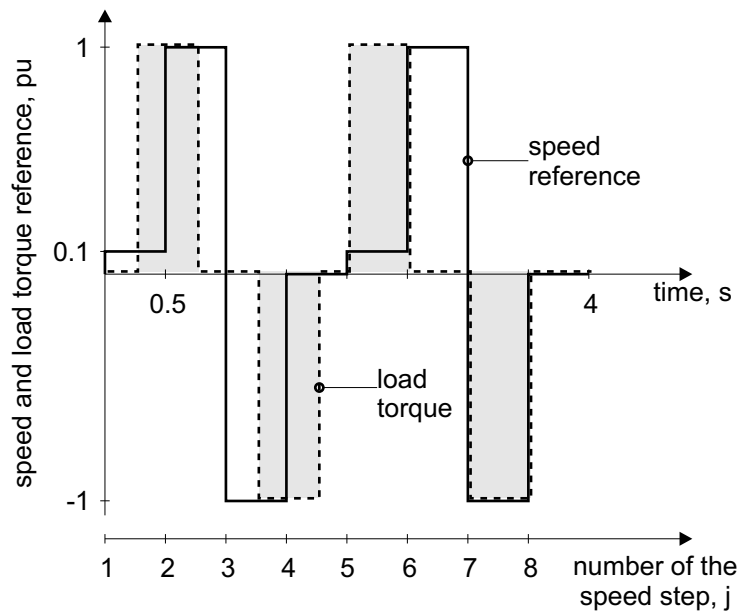


Figure 3.6: The training test is a combination of speed commands (a sequence of 8 speed steps, continuous line) and load torque commands (dashed line).

3.3.1 Training Test

For this research, we propose the training test shown in fig. [3.6](#). It can be noted that the combination of the speed and load torque commands makes it a general-purpose test. Indeed, in addition to low and high speeds, zero-speed and speed reversal are considered at no- and full-load torque. In some applications, it will not be possible to change the load directly, as in this test. However, in many applications (e.g. fan, pump) a change in load will result from a change in speed, and a series of speed changes may provide sufficient training test.

3.3.2 Objective Function

The performance given by each solution is numerically evaluated through the objective function following the conventional weighted-sum approach [52].

The lower the individual's objective value is, the higher the probability of the individual at propagating its genes to the next generation is.

To optimize the overall response of the drive, the objective function to be minimized is

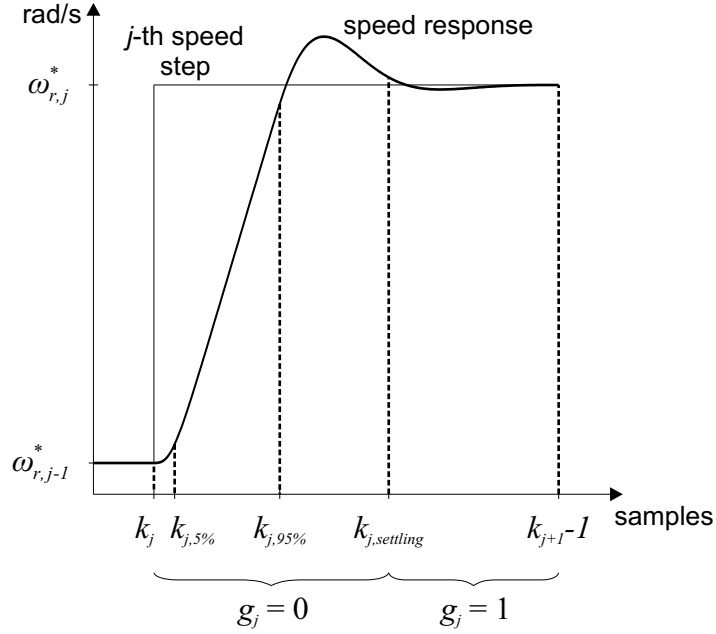
$$Obj = \sum_{i=1}^4 \left(a_i \cdot \sum_{j=1}^8 Obj_{i,j} \right) \quad (3.4)$$

where j indicates the number of the speed steps (the training test is a sequence of 8 speed steps), i indicates the number of the performance index, and a_i is the positive normalization factor of the respective performance index $Obj_{i,j}$. For more clarity, the j -th speed step of the training test is shown in fig. 3.7.

The performance index $Obj_{1,j}$ measures the speed error in the settling phase

$$Obj_{1,j} = \sum_{k=k_j}^{k_{j+1}-1} |\omega_r(k) - \omega_{r,j}^*| g_j(k) \quad (3.5)$$

where $\omega_r(k)$ is the k -th stored sample of the rotor speed, and k_j is the sample at which the set point changes to $\omega_{r,j}^*$. The activation function g_j returns 0 when the speed reference changes, and returns 1 after $k_{j,settling}$ samples. $k_{j,settling}$ is determined from the 5% settling time of the speed response given by the original controller (x_0), and remains fixed throughout

Figure 3.7: j -th speed step of the training.

the optimization. In this way the optimization is directed towards solutions whose settlings are better than those provided by x_0 . In fact, $Obj_{1,j}$ aims to take account of the steady-state speed error which depends mainly on the nonlinearities and disturbance rejections. If some solution provides a settling time greater than that given by x_0 , then $Obj_{1,j}$ will be affected not only by the steady-state speed error, but also by the transient speed error which is numerically bigger.

The overshoot index is

$$Obj_{2,j} = \left| \frac{\max_{k=k_j, \dots, k_{j,settling}} (\omega_r(k) \cdot \text{sign} \Delta \omega_{r,j}^*) - \omega_{r,j}^* \cdot \text{sign} \Delta \omega_{r,j}^*}{\Delta \omega_{r,j}^*} \right| \quad (3.6)$$

where $\Delta \omega_{r,j}^* = \omega_{r,j}^* - \omega_{r,j-1}^*$ is the amplitude of the j -th speed step. It should be noted that, (3.6) works for positive and negative values of $\omega_{r,j}^*$ and $\Delta \omega_{r,j}^*$,

and for under and over damped responses.

The rise time index is

$$Obj_{3,j} = \frac{k_{j,95\%} - k_{j,5\%}}{|\Delta\omega_{r,j}^*|} \quad (3.7)$$

where $k_{j,95\%}$ and $k_{j,5\%}$ are the samples between which the measured speed rises from the 5% to the 95% of the speed step.

Finally, the last performance index is

$$Obj_{4,j} = \sum_{k=k_j}^{k_{j+1}-1} |i_{sd}(k)| \quad (3.8)$$

and takes account of the undesired d-axis-current oscillations which increase losses and vibrations in the motor and drive.

3.3.3 Poorly Performing Solutions

The on-line optimization consists of numerous experiments because each possible solution will be experimentally tested. In this way, the experiment of an unstable (or highly unsatisfactory) solution can severely stress the hardware. While the theoretical guarantees of closed-loop stability on the actual hardware remain open for further research, in this thesis we have adopted an heuristic strategy that effectively overcomes the limitations of model-based analysis. Rather than computing the objective value at the end of each experiment, as is usually done in on-line tuning approaches, each performance index value is monitored and update in real-time, i.e. at each sample time of each experiment. If during an experiment one of the performance indices

exceeds a predefined warning threshold, the current solution is recognized as “bad”. It is immediately replaced by the stable solution x_0 , obtained by the initial commissioning, and the motor is stopped. Then a penalty factor is applied to the objective value according to the rule: “the earlier the bad solution is detected, the higher the penalty factor”. Afterwards the optimization can go on with the next experiment. The value of each warning threshold can be set between 1.5 and 3 times the value of the respective performance index given by x_0 .

3.4 Evolutionary Operators

3.4.1 Selection

After the evaluation of all the individuals of a population, the selection process starts. The selection consists of ranking and sampling.

Ranking

In order to avoid pre-convergence [56], individuals are initially ranked according to their objective values. Particularly the linear ranking algorithm [57] has been adopted.

An example can easily show how it works. Let us suppose that a population of five individuals, $N_{ind} = 5$, is evaluated and then the ranking position of

each individuals is calculated:

$$\mathbf{Obj} = \begin{bmatrix} 11 \\ 506 \\ 343 \\ 62 \\ 97 \end{bmatrix} \Rightarrow \mathbf{Pos} = \begin{bmatrix} 1 \\ 5 \\ 4 \\ 2 \\ 3 \end{bmatrix}, \quad (3.9)$$

where $Obj(i)$ and $Pos(i)$ are the objective and position values of the i -th individual, respectively.

Finally the fitness vector is calculated as

$$\mathbf{Ftn} = (2 - SP) + 2 \frac{SP - 1}{N_{ind} - 1} (\mathbf{Pos} - 1) \quad (3.10)$$

where \mathbf{Ftn} is the fitness vector and $SP \in [1, 2]$ is the selective pressure.

Considering the example data and a selective pressure $SP = 1.5$, the equation (3.10) produces the following fitness vector

$$\mathbf{Ftn} = \begin{bmatrix} 0.5 \\ 1.25 \\ 1.5 \\ 1 \\ 0.75 \end{bmatrix}. \quad (3.11)$$

Fig. 3.8 shows how the linear ranking, according to the selective pressure, is a diversity diminishing and contracting operation. The population diversity, often listed among the advantages of the evolutionary approach, can defeat

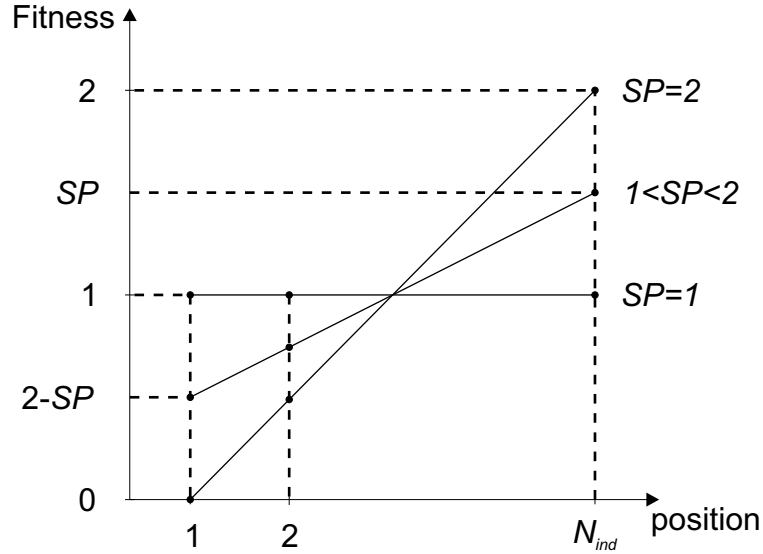


Figure 3.8: Fitness as function of the individual position and the selective pressure.

the optimization when the difference between some individual and the others is too much. This because the evolution tends to use only the few fittest individuals to create the new generation. In other words, the whole population tends to replicate those individuals drastically reducing the exploration and causing the pre-convergence. A comparison between objective values, equation (3.9), and fitness values, equation (3.11), highlights how the ranking has solved the problem. Usually the selective pressure is set between 1.5 and 2 because a too low value causes a too slow convergence rate.

Sampling

The sampling generates the mating pool, i.e. the set of individuals for the reproduction. The N_{pa} individuals of the mating pool are also called parents. The stochastic universal sampling method has been chosen to achieve zero bias and minimum spread [58]. The probability of the individual being

selected is

$$\mathbf{SelP} = \begin{cases} \frac{1}{\sum_1^{N_{ind}} Ftn(i)} \mathbf{Ftn} & \text{formaximizationproblem} \\ 1 - \frac{1}{\sum_1^{N_{ind}} Ftn(i)} \mathbf{Ftn} & \text{forminimizationproblem} \end{cases} . \quad (3.12)$$

In this work the optimization is a minimization problem, i.e., the lower the fitness value the better the individual. Therefore the second selection probability formula is used.

3.4.2 Recombination

The exchange of genetic information is known as recombination. The recombination of the N_{pa} parents generates the same number of new individuals called offspring. The algorithm adopted is known as intermediate recombination [59], and is based on the following formula for offspring calculation

$$\mathbf{x}_{of} = \mathbf{x}_{p1} + \mathbf{B}(\mathbf{x}_{p2} - \mathbf{x}_{p1}) \quad (3.13)$$

where \mathbf{x}_{p1} , \mathbf{x}_{p2} and \mathbf{x}_o are the two parents and the offspring, respectively. \mathbf{B} is a diagonal matrix with diagonal elements b_i randomly chosen in the interval $\left[b^{\min}, b^{\max} \right]$, where typical values are $b^{\min} = -0.25$ and $b^{\max} = 1.25$. If all the b_i have the same value, the intermediate recombination is called linear recombination. The recombination produces offspring in the hypercube defined by

$$\mathbf{x}_{of}^{\min} = \mathbf{x}_{p1} + b^{\min}(\mathbf{x}_{p2} - \mathbf{x}_{p1}), \quad (3.14)$$

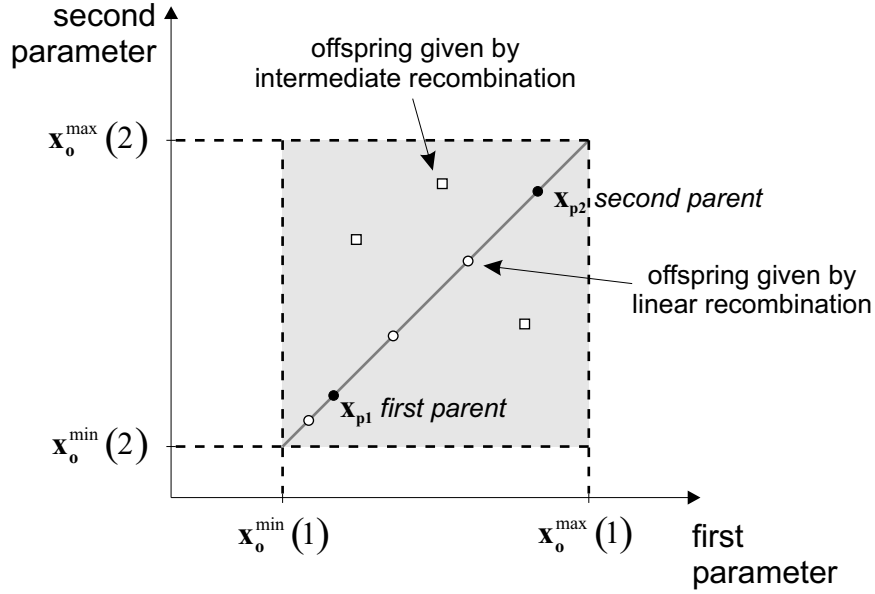


Figure 3.9: Example of intermediate recombination for a two-dimension problem.

$$\mathbf{x}_{\text{of}}^{\max} = \mathbf{x}_{\text{p1}} + b^{\max} (\mathbf{x}_{\text{p2}} - \mathbf{x}_{\text{p1}}). \quad (3.15)$$

For sake of clarity, fig. 3.9 shows the possible offspring for a two-dimension problem.

3.4.3 Mutation

Each offspring's genes is low-probabilistically altered during the mutation process [59]

$$\tilde{\mathbf{x}}_{\text{o}}(i) = \mathbf{x}_{\text{o}}(i) + MP \cdot (\mathbf{x}_{\text{ub}}(i) - \mathbf{x}_{\text{lb}}(i)) \quad (3.16)$$

where MP is called mutation probability and is randomly chosen in the interval $[MP^{\min}, MP^{\max}]$. Usually mutation probability is some percentage, but better results in this case study have been achieved by using $MP^{\min} = -0.2$ and $MP^{\max} = 0.2$.

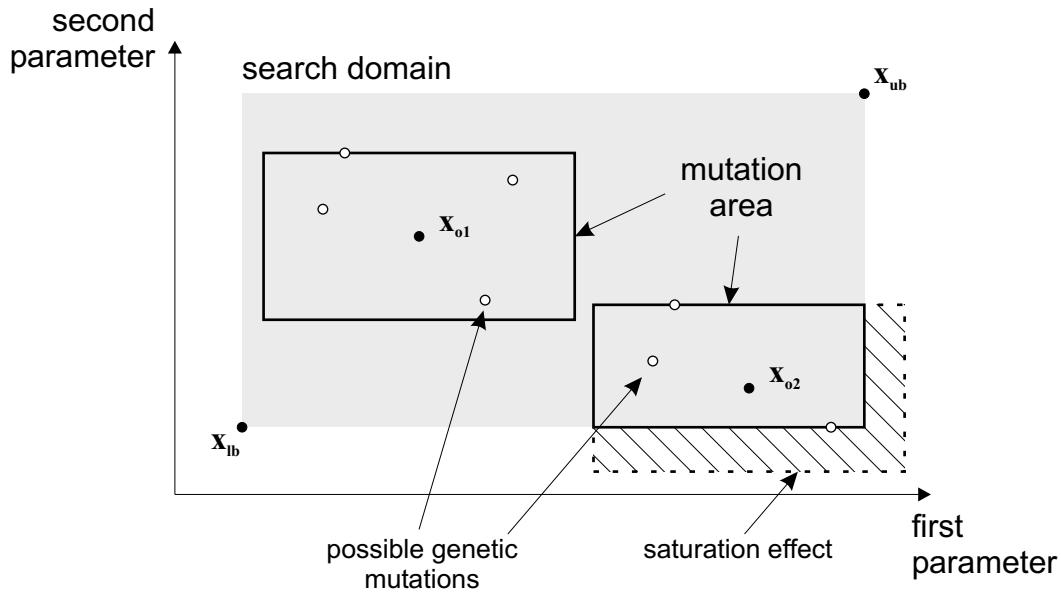


Figure 3.10: Example of mutation for a two-dimension problem.

There is evidence that equation (3.16) can give points beyond the search domain. To avoid this the genes are saturated as follows

$$x_o(i) = \begin{cases} x_{ub}(i) & \text{if } \tilde{x}_o(i) > x_{ub}(i) \\ \tilde{x}_o(i) & \text{if } x_{lb}(i) \leq \tilde{x}_o(i) \leq x_{ub}(i) \\ x_{lb}(i) & \text{if } \tilde{x}_o(i) < x_{lb}(i) \end{cases} \quad (3.17)$$

3.4.4 Reinsertion

Finally, fitness-based reinsertion has been used to create the new population of N_{ind} individuals. The least fit individuals of the previous population are probabilistically replaced by the most fit offspring.

3.5 Termination

At the end of every iteration, or generation, the GA provides a new population whose individuals can generally solve the problem better than the previous ones. Because of the stochastic nature of the GA, the formulation of convergence criteria is still an open problem. For this research the GA is terminated after a predefined number of generations N_{gen} .

Chapter 4

Hybridization of EAs

EAs are guided stochastic search methods that suffer of slow convergence rate. A careful configuration, as discussed in the previous chapter, only partly solves such a problem that lies in the non-deterministic nature of evolutionary operators which locate the optimal “hill”, i.e., the zone where the optimal solution is, but are not able to quickly refine it. On the contrary, classical methods, often classified as hill-climbing, can efficiently exploit local information to speed up the optimization, but properly work when the function to be optimized is smoothed and unimodal. Unfortunately, they often fail in real-world problems that usually are ill-behaved. In order to get the benefits of both the techniques, many hybrid methods were proposed, see fig.

[4.1](#)

A hybrid method consists of a combination between different search methods. Although it is a very general definition, it is the only one that can include the huge number of possibilities. These are given not only by the number of search methods, but also by the hybrid architecture, i.e., the way in which

the different methods are integrated in a framework to cooperate. Among the hundreds paper concerning this subject, [60] is very noteworthy. In addition to a new hybrid approach for GA and simplex method, the authors very well explain the hybrid architectures already proposed in literature. As regards the self-commissioning of electric drives, the present thesis proposes a generalized variant of the elite-based hybrid architecture, proposed in [60]. As shown in fig. 4.2, an initial sub-evolution transforms the initial population, pop_{in} in the intermediate population, pop_{int} . This stage consists of running the EA whose configuration is described in the previous chapter. Consequently, the top-ranking individuals are extracted from pop_{int} creating $pop_{top,in}$. Analogously, the medium-ranking individuals form $pop_{med,in}$. The size of $pop_{top,in}$ top-ranking is chosen between the 10% and 15% of pop_{int} , whilst that of $pop_{med,in}$ is between the 95% and 85%. It has to be noted that $pop_{top,in}$ and $pop_{med,in}$ can be overlapped, that is, can have some individuals in common. The probabilistic multi-directional simplex method, later described, is the local search method that operates on $pop_{top,in}$ to produce $pop_{top,out}$. Again, the $pop_{med,out}$ is obtained by the aforementioned EA with $pop_{med,in}$ as initial population. Finally, the pop_{out} is obtained by a fitness-based merging of $pop_{top,out}$ and $pop_{med,out}$. This is one iteration that can be repeated until a termination criterion, such as a prefixed number of iterations, is satisfied.

The new hybrid architecture better coordinates the global and local search methods in order to save fitness evaluations. Although this is experimentally proved even if not theoretically justifiable, it is author conviction that this is due to the highly noisy fitness. The proposed solution rations the use of

the local search more than that done in [60] or in [61].

For the same reason, the proposed local search method is an emphad-hoc combination of different variants of the simplex method, namely probabilistic multi-directional simplex method. Simplex method was initially proposed in the early sixties, and then popularized by Nelder and Mead [62] with their effective version. It belongs to the class of the direct search methods whose two main properties are:

- no gradient, or any gradient approximation, can be used,
- only the values of the fitness function can be used.

These properties make the direct search methods an efficient alternative to Newton's, and quasi-Newton methods that are impracticable:

- if the fitness evaluation is very time-consuming and noisy such as, when calculated through experimental tests,
- if gradient, Hessian, and first partial derivatives of the fitness function cannot be exactly calculated, and their numerical approximations are too expensive.

The different variants used to implement the probabilistic multi-directional simplex will be described in the next sections. For more detail about the direct search methods please refer to [63, 64, 65].

4.1 Nelder and Mead algorithm

A simplex in a n -dimensional Euclidean space is a Euclidean geometric spatial element having $n + 1$ boundary points, and can be defined by a set of $n + 1$ points. For instance, a simplex is a line segment in one-dimensional space,

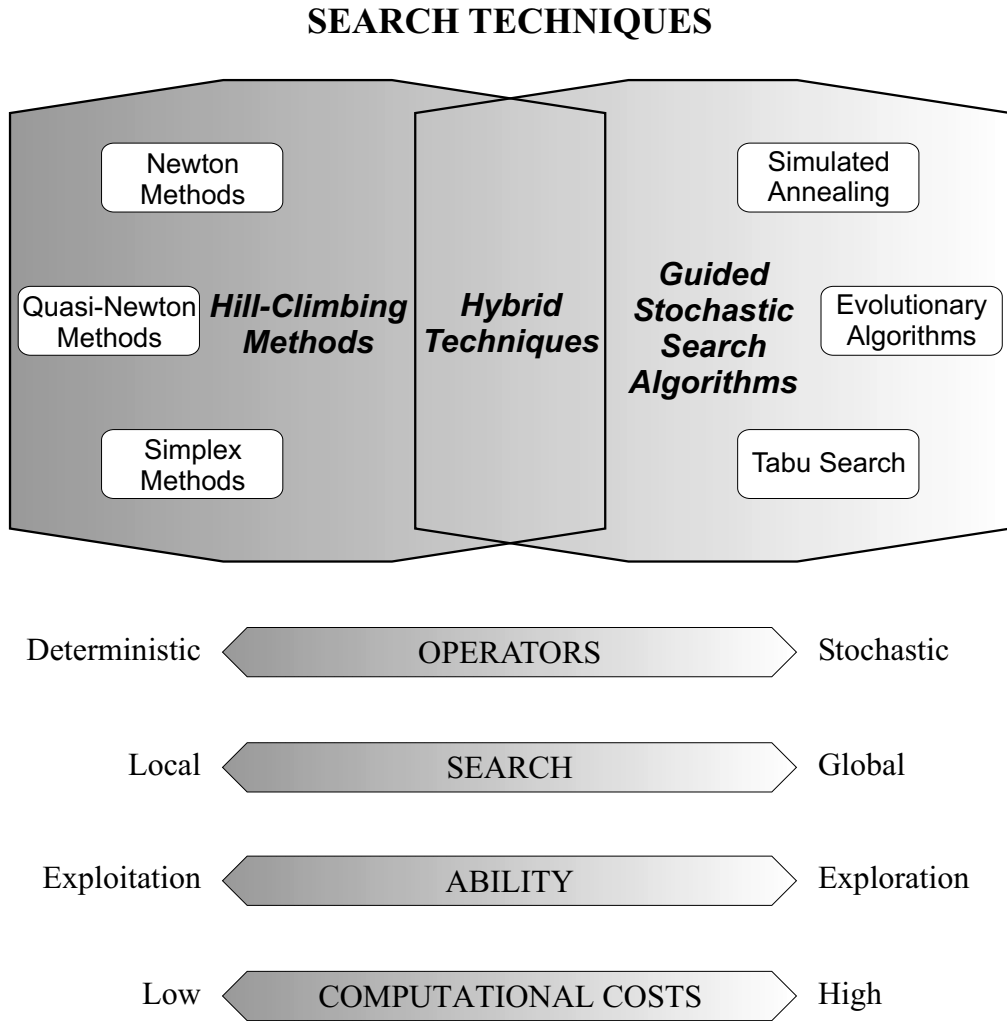


Figure 4.1: Classification of the search techniques.

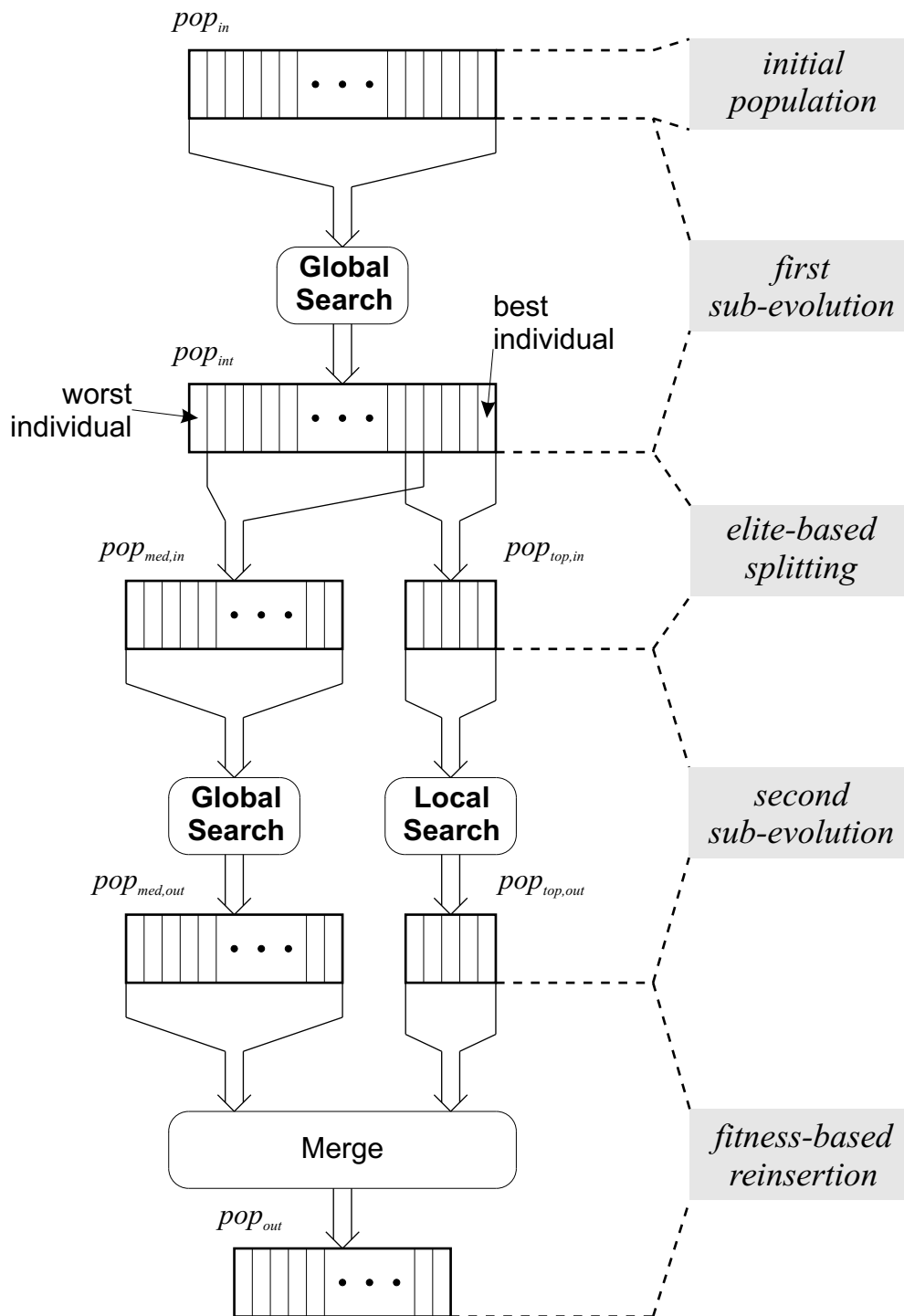


Figure 4.2: Elite-based hybrid architecture.

a triangle in two-dimensional space, or a tetrahedron in three-dimensional space, as shown in fig. 4.3; and the vertices of the spatial elements completely define the simplex itself.

The flowchart of an iteration of this method is shown in fig. 4.4.

4.1.1 Sorting

Generic iteration begins by sorting and indexing the simplex points, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}$, according to their fitness, so that

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \dots \leq f(\mathbf{x}_n) \leq f(\mathbf{x}_{n+1}) \quad (4.1)$$

where f is the fitness function to be minimized, consequently \mathbf{x}_1 is the lower fitness point, i.e., the best point, and \mathbf{x}_{n+1} is the worst point.

4.1.2 Centroid calculation

Then, the centroid is calculated as the average value of the best n points

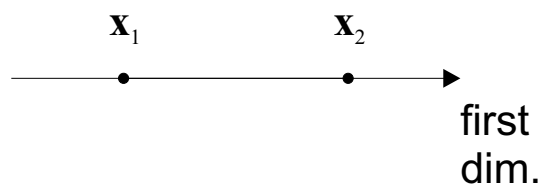
$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i, \quad (4.2)$$

as shown in fig. 4.5a.

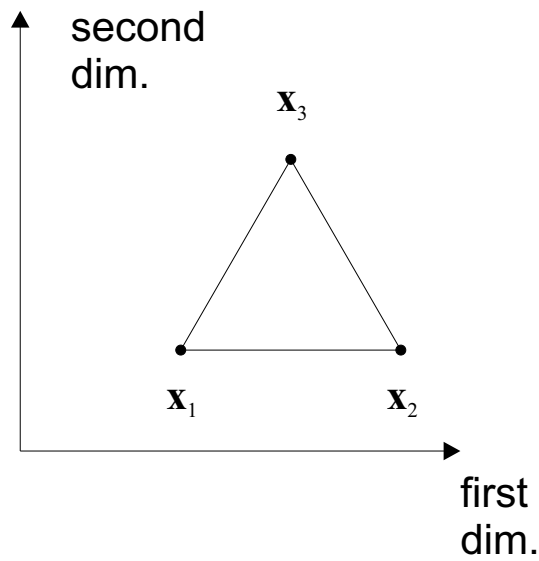
4.1.3 Reflection

In order to replace the worst point \mathbf{x}_{n+1} with a better one, a reflected point is calculated as

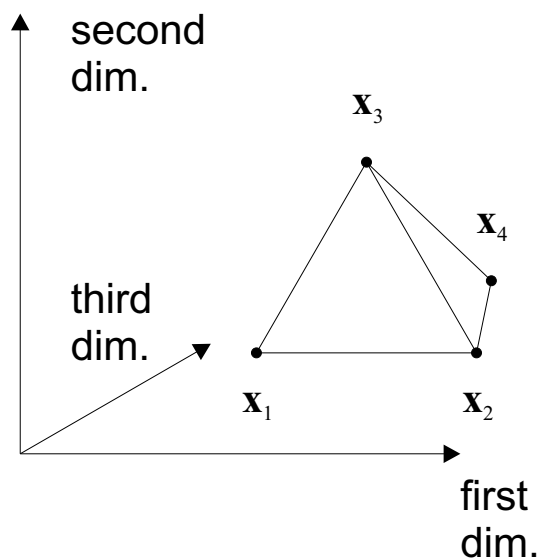
$$\mathbf{x}_{r,n+1} = \bar{\mathbf{x}} + k_r (\bar{\mathbf{x}} - \mathbf{x}_{n+1}) \quad (4.3)$$



(a) One-dimensional space.



(b) Two-dimensional space.



(c) Three-dimensional space.

Figure 4.3: Some example of a simplex.

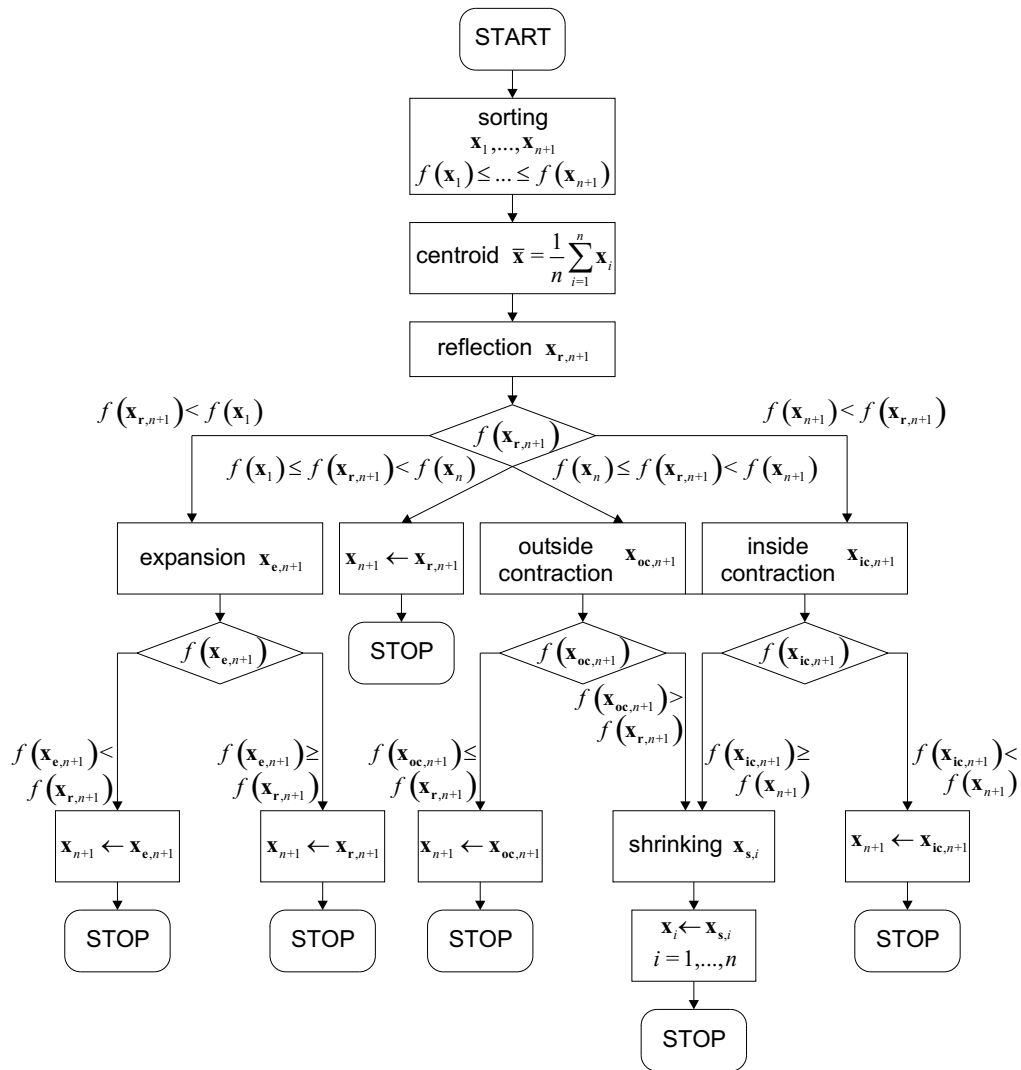


Figure 4.4: Flowchart of an iteration of Nelder and Mead's simplex method.

where $k_r = 1$ is the reflection coefficient, and its standard value is that originally proposed in [62].

The reflection operates a central symmetry, also known as point symmetry, through the point $\bar{\mathbf{x}}$ that sends the point \mathbf{x}_{n+1} to the point at the same distance $\|\bar{\mathbf{x}} - \mathbf{x}_{n+1}\|_2$ on the line determined by $\bar{\mathbf{x}}$ and \mathbf{x}_{n+1} but on the opposite side from $\bar{\mathbf{x}}$. As shown in fig. 4.5b, the idea is to move the worst point towards the more promising zone given by the better n points.

According to the fitness of the reflected point, $f(\mathbf{x}_{r,n+1})$, different decisions can be taken:

- if $f(\mathbf{x}_{r,n+1}) < f(\mathbf{x}_1)$ then the expansion is operated,
- if $f(\mathbf{x}_1) \leq f(\mathbf{x}_{r,n+1}) < f(\mathbf{x}_n)$ then the reflected point replaces the worst, and the iteration terminates,
- if $f(\mathbf{x}_n) \leq f(\mathbf{x}_{r,n+1}) < f(\mathbf{x}_{n+1})$ then the outside contraction is operated,
- if $f(\mathbf{x}_{n+1}) < f(\mathbf{x}_1)$ then the inside contraction is operated.

4.1.4 Expansion

If $f(\mathbf{x}_{r,n+1}) < f(\mathbf{x}_1)$, that is, the reflected point is the new best point, a new point, namely expanded point, is calculated as follows

$$\mathbf{x}_{e,n+1} = \bar{\mathbf{x}} + k_e (\bar{\mathbf{x}} - \mathbf{x}_{n+1}) \quad (4.4)$$

where $k_e = 2$ is the standard expansion coefficient. Fig. 4.5c shows how expansion stretches the simplex in the very promising direction which reflection pointed to. According to the fitness of the expanded point

- if $f(\mathbf{x}_{e,n+1}) < f(\mathbf{x}_{r,n+1})$, then the expanded point replaces the worst, and

the iteration terminates,

- if $f(\mathbf{x}_{\mathbf{e},n+1}) \geq f(\mathbf{x}_{\mathbf{r},n+1})$, then the reflected point replaces the worst, and the iteration terminates.

4.1.5 Outside contraction

If $f(\mathbf{x}_n) \leq f(\mathbf{x}_{\mathbf{r},n+1}) < f(\mathbf{x}_{n+1})$, it means that the pointed direction is not so favorable. Therefore the simplex is contracted by

$$\mathbf{x}_{\mathbf{oc},n+1} = \bar{\mathbf{x}} + k_c (\bar{\mathbf{x}} - \mathbf{x}_{n+1}) \quad (4.5)$$

where $k_c = 0.5$ is the standard contraction coefficient. Fig. 4.5d shows how outside contraction works in the two-dimensional space.

According to $\mathbf{x}_{\mathbf{oc},n+1}$'s fitness,

- if $f(\mathbf{x}_{\mathbf{oc},n+1}) \leq f(\mathbf{x}_{\mathbf{r},n+1})$, then $\mathbf{x}_{\mathbf{oc},n+1}$ replaces the worst \mathbf{x}_{n+1} , and the iteration terminates,
- if $f(\mathbf{x}_{\mathbf{oc},n+1}) > f(\mathbf{x}_{\mathbf{r},n+1})$, then the shrinking is operated.

4.1.6 Inside contraction

If $f(\mathbf{x}_{n+1}) < f(\mathbf{x}_{\mathbf{r},n+1})$, it means that the pointed direction is very unfavorable. The new contracted point is given by

$$\mathbf{x}_{\mathbf{ic},n+1} = \bar{\mathbf{x}} - k_c (\bar{\mathbf{x}} - \mathbf{x}_{n+1}) \quad (4.6)$$

where $k_c = 0.5$ is the contraction coefficient used in (4.6). Fig. 4.5e shows how inside contraction works in the two-dimensional space.

According to $\mathbf{x}_{\mathbf{ic},n+1}$'s fitness,

- if $f(\mathbf{x}_{\mathbf{ic},n+1}) < f(\mathbf{x}_{n+1})$, then $\mathbf{x}_{\mathbf{ic},n+1}$ replaces the worst \mathbf{x}_{n+1} , and the iteration terminates,
- if $f(\mathbf{x}_{\mathbf{ic},n+1}) \geq f(\mathbf{x}_{n+1})$, then the shrinking is operated.

4.1.7 Shrinking

The shrinking is applied when any of previous operation produces a new point better than the worst, and consists of moving all the other vertices $\{\mathbf{x}_i\}_{i=1,\dots,n}$ towards \mathbf{x}_1 . The new points are given by

$$\mathbf{x}_{s,i} = \mathbf{x}_1 + k_s (\mathbf{x}_1 - \mathbf{x}_i), i = 1, \dots, n \quad (4.7)$$

where $k_s = 0.5$ is the standard shrinking coefficient. It has to be noted that also in this case the simplex is contracted and for this reason the shrinking is also known as multi-contraction, fig. [4.5f](#)

4.2 Probabilistic version

Although at a first glance, the way in which N-M's algorithm searches the best solution seems simple, the convergence properties are still discussed by mathematics. However, there is evidence that the optimization process can be modified by changing reflection, expansion, and contraction coefficients. According to the problem, often such coefficients are manually changed in order to improve the convergence rate. Besides such intuitive changes, an innovative idea was proposed in [\[66\]](#). The authors made non-deterministic the

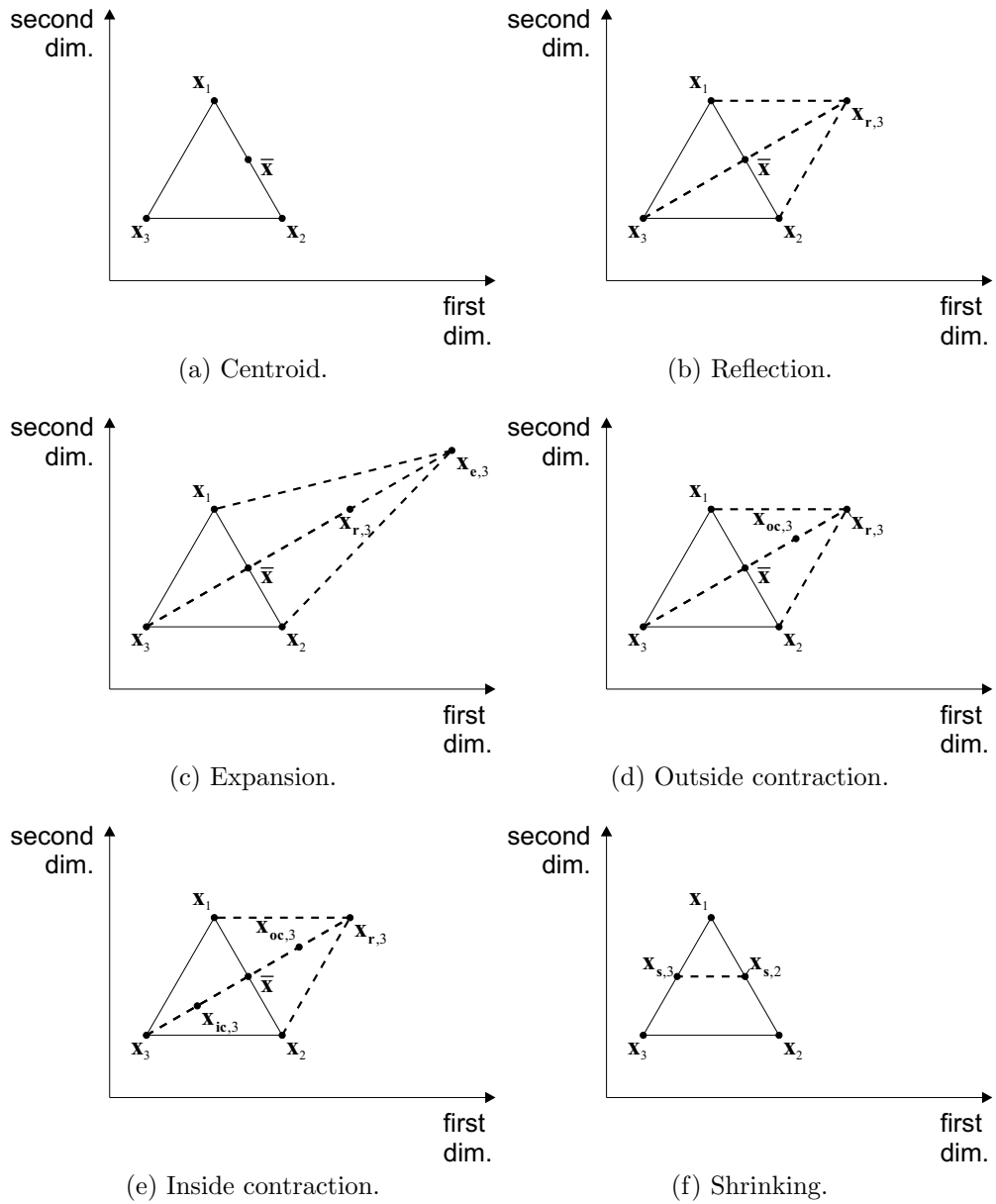


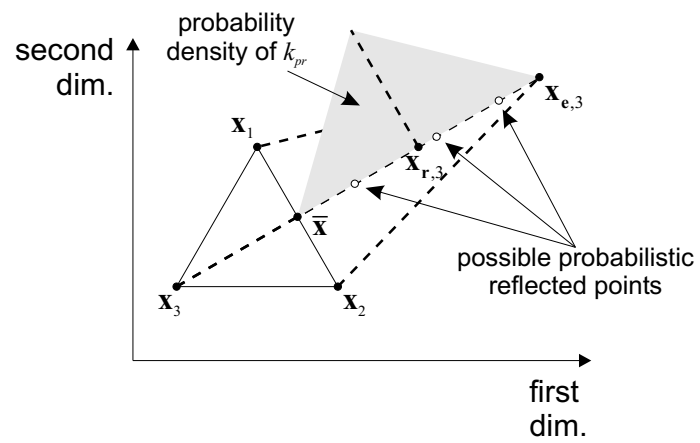
Figure 4.5: Operations of N-M's simplex method.

coefficients of the simplex method to introduce a cost-effective exploration component. This idea has been proved very effective, particularly for online optimization, making the simplex method able to cope with noisy fitness functions.

This variant consolidates reflection, expansion, and outside contraction operations in one probabilistic reflection. In N-M's algorithm, these operations can be seen as a reflection in which the reflection coefficient either $k_r = 1$, or $k_e = 2$, or $k_{oc} = 0.5$ are respectively adopted. As shown in figures [4.6a](#) and [4.6b](#), the probabilistic reflection uses one random variable $k_{pr} \in [0, 2]$ with a triangular probability density function that peaks at 1 and reaches zero at 0 and 2.

Analogously the probabilistic contraction replaces the internal contraction of N-M's algorithm, by using a $k_{pc} \in [0, 1]$ with a triangular probability density function that peaks at 0.5 and reaches zero at 0 and 1, as shown in figures [4.7a](#) and [4.7b](#).

The same authors further introduced a new variant, known as concurrent simplex method, when their aforementioned paper was accepted for journal publication [\[60\]](#). The concurrent simplex methods can be seen as a generalized N-M's algorithm. It starts with $n + m$ points, where $m \geq 1$, the two methods are equivalent for $m = 1$. Sorting, reflection, expansion, and contraction are basically the same, but operates on the worst m points, as shown in fig. [4.8](#). At the expense of a greater number of function evaluations, the search space is better investigated. Furthermore, it has to be noted that the concurrent variation of the simplex method can be applied independently from the probabilistic one.



(a) Probabilistic reflection.

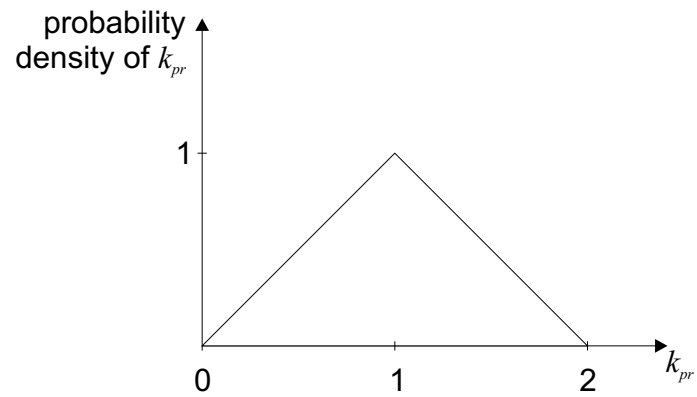
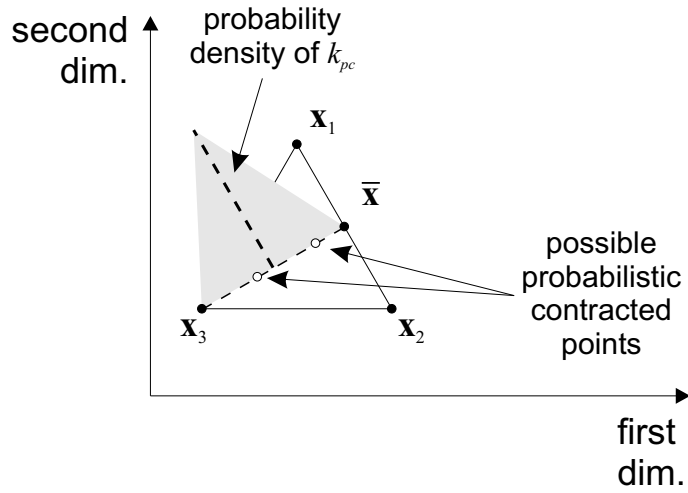
(b) Probability density of k_{pr} .

Figure 4.6: Reflection in the probabilistic simplex method.



(a) Probabilistic contraction.

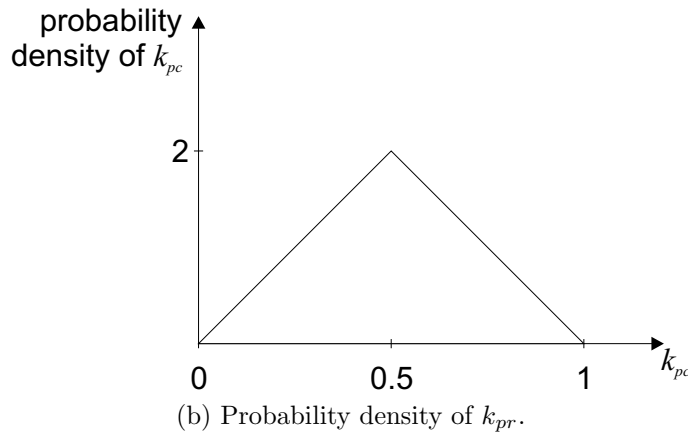
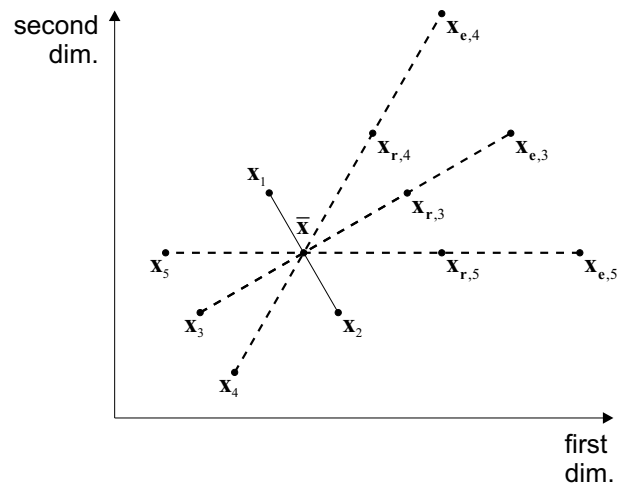
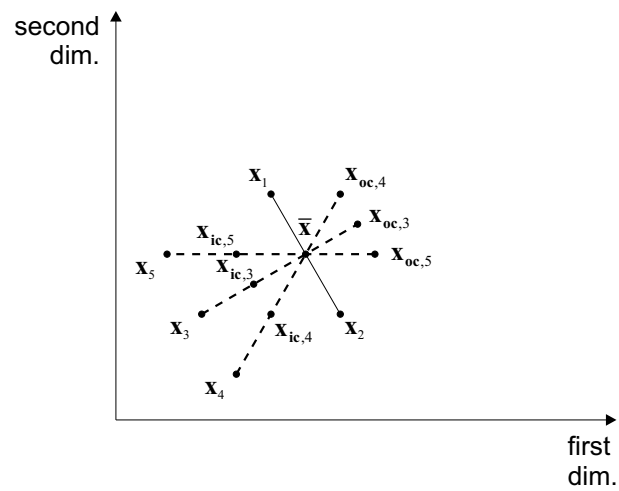


Figure 4.7: Contraction in the probabilistic simplex method.



(a) Reflection and expansion.



(b) Inside and outside contraction.

Figure 4.8: Operations of the concurrent simplex method.

4.3 Multidirectional search algorithm

Among the many recent variants, one of the most important is the multidirectional search algorithm that was initially proposed and then developed in [67]. It was developed to meet the need of efficiency in a parallel computing environment.

Multidirectional search algorithm is a simplex method in which the point of symmetry of reflections and contractions is the best point \mathbf{x}_1 , unlike N-M's version that uses the centroid $\bar{\mathbf{x}}$, see equation (4.2). Reflections or contractions of the remaining n points, $\{\mathbf{x}_i\}_{i=2,\dots,n+1}$, through the best point \mathbf{x}_1 , allow a parallel exploration in different directions. The generic iteration terminates when a new point outperforming \mathbf{x}_1 is found. It has to be noted that this condition is much stronger than that required in N-M's version.

The flowchart of an iteration of multidirectional search method is shown in fig. 4.9.

Sorting

Also in this case an iteration begins by sorting and indexing the simplex points, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}$, according to their fitness, so that

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \dots \leq f(\mathbf{x}_n) \leq f(\mathbf{x}_{n+1}) \quad (4.8)$$

where f is the fitness function to be minimize, consequently \mathbf{x}_1 is the lower fitness point, i.e., the best point, and \mathbf{x}_{n+1} is the worst point.

Reflection

New n points are calculated by reflecting $\{\mathbf{x}_i\}_{i=2,\dots,n+1}$ as follows

$$\mathbf{x}_{r,i} = \mathbf{x}_1 + k_r (\mathbf{x}_1 - \mathbf{x}_i), i = 2, \dots, n + 1 \quad (4.9)$$

where $k_r = 1$ is the reflection coefficient. Fig. [4.10a](#) shows how reflection works in the two-dimensional space.

Comparing the fitness of the best reflected points and with that of \mathbf{x}_1 , different decisions can be taken:

- if $\min_{i=2,\dots,n+1} \{f(\mathbf{x}_{r,i})\} < f(\mathbf{x}_1)$ then the expansion is operated,
- if $\min_{i=2,\dots,n+1} \{f(\mathbf{x}_{r,i})\} \geq f(\mathbf{x}_1)$ then the contraction is operated.

Expansion

If $\min_{i=2,\dots,n+1} \{f(\mathbf{x}_{r,i})\} < f(\mathbf{x}_1)$, the expanded simplex is given by

$$\mathbf{x}_{e,i} = \mathbf{x}_1 + k_e (\mathbf{x}_1 - \mathbf{x}_i), i = 2, \dots, n + 1 \quad (4.10)$$

where $k_e = 2$ is the expansion coefficient. Fig. [4.10b](#) shows how expansion operates in the two-dimensional space.

If $\min_{i=2,\dots,n+1} \{f(\mathbf{x}_{e,i})\} < \min_{i=2,\dots,n+1} \{f(\mathbf{x}_{r,i})\}$, then the expanded points $\{\mathbf{x}_{e,i}\}_{i=2,\dots,n+1}$ will replace the initial $\{\mathbf{x}_i\}_{i=2,\dots,n+1}$ in the next iteration, otherwise the latter will be replaced by the reflected ones $\{\mathbf{x}_{r,i}\}_{i=2,\dots,n+1}$. In any case, the iteration terminates.

Contraction

If $\min_{i=2,\dots,n+1} \{f(\mathbf{x}_{r,i})\} \geq f(\mathbf{x}_1)$ then the contracted simplex is given by

$$\mathbf{x}_{c,i} = \mathbf{x}_1 - k_c (\mathbf{x}_1 - \mathbf{x}_i), i = 2, \dots, n + 1 \quad (4.11)$$

where $k_c = 0.5$ is the standard contraction coefficient. Comparing figures [4.10c](#) and [4.5f](#), equations [\(4.11\)](#) and [\(4.7\)](#), there is evidence that the multidirectional contraction is equivalent to the shrinking of N-M's version.

In this case, the current iteration terminates replacing the initial $\{\mathbf{x}_i\}_{i=2,\dots,n+1}$ by $\{\mathbf{x}_{c,i}\}_{i=2,\dots,n+1}$, regardless their fitness.

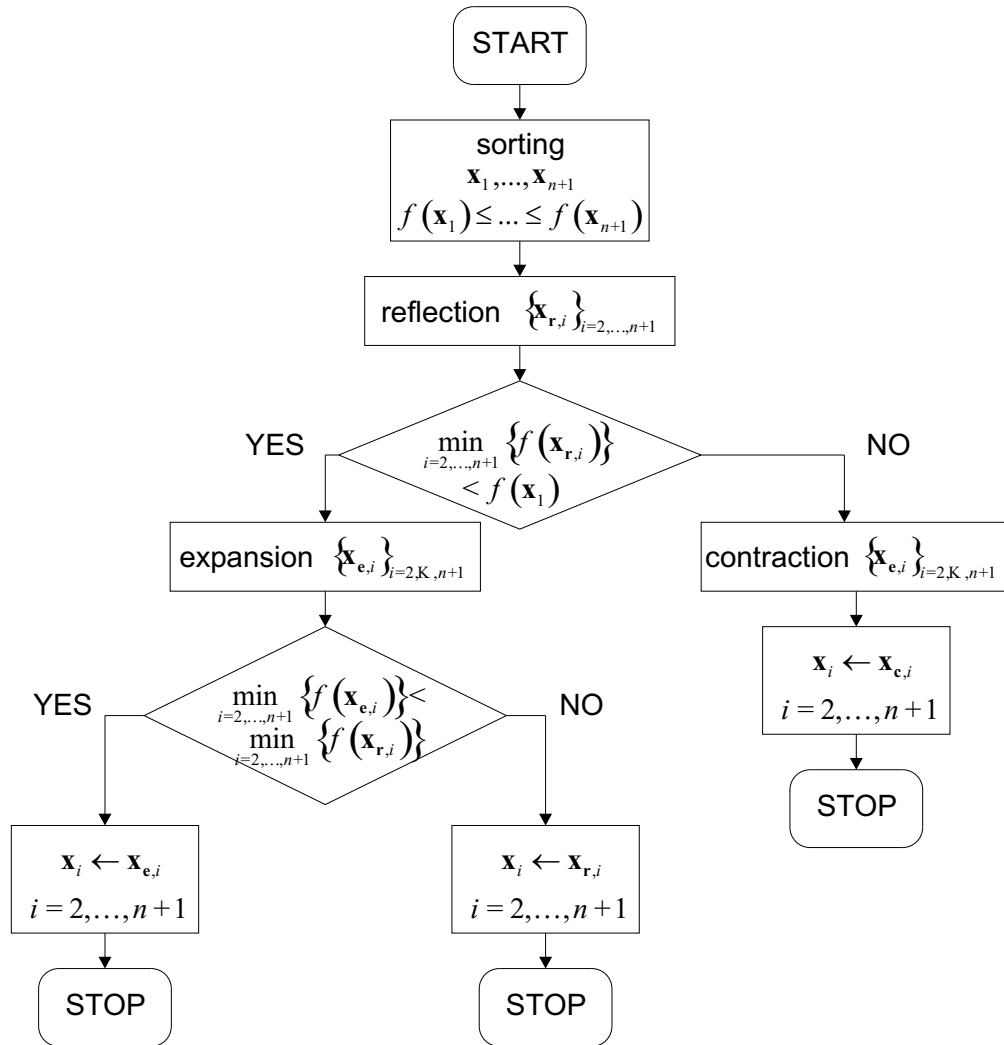


Figure 4.9: Flowchart of the multi-dimensional simplex method.

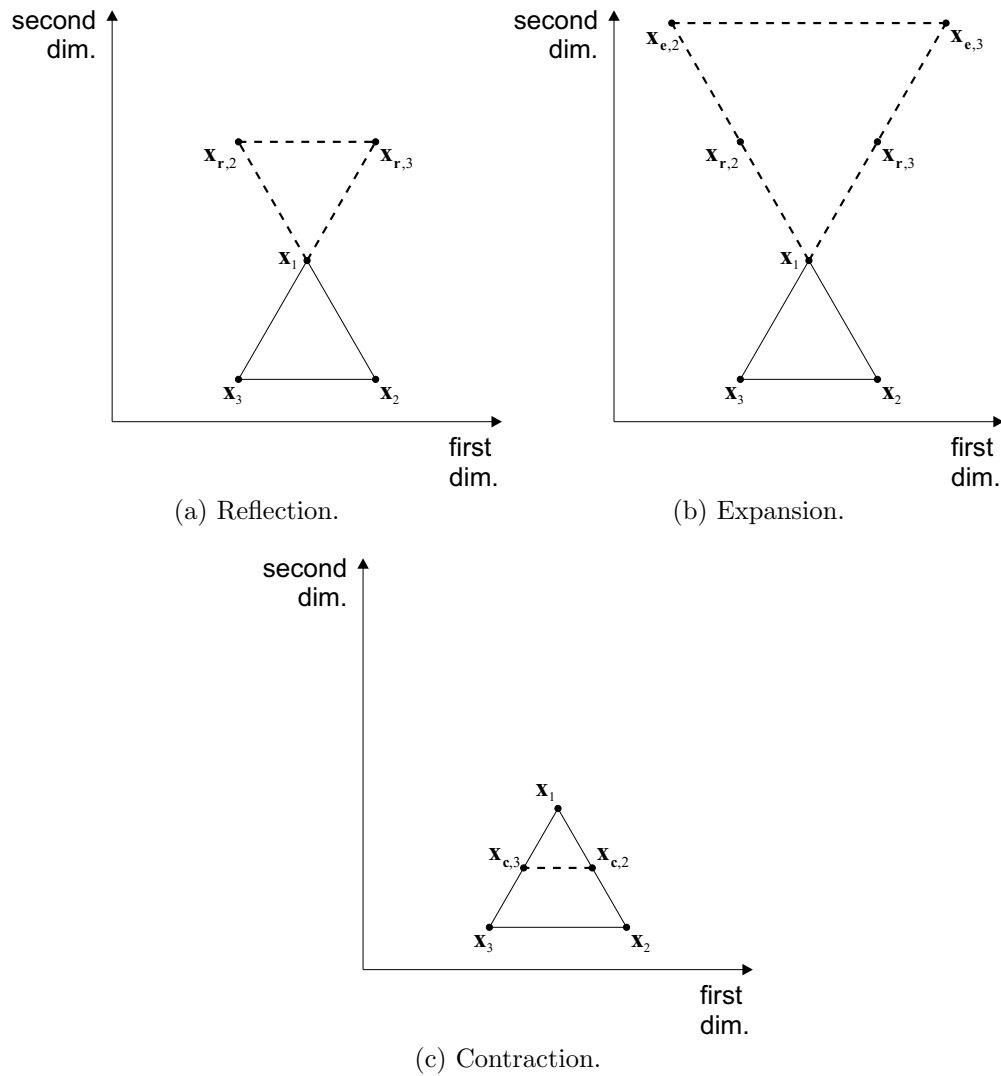


Figure 4.10: Operations of the multi-dimensional simplex method.

Chapter 5

Experimental setup and results

5.1 Hardware and Software Implementation

The implementation of the control system and the self-commissioning procedure is based on an Host PC and a multi-purpose single board dSpace DS1103 [68]. Although the DS1103 has a floating-point PowerPC 604e microprocessor working in conjunction with a TMS320F240 fixed-point DSP, the control platform used in this project employs the fixed-point DSP only, so that control hardware is equivalent to that of a typical industrial drive, as shown in fig. 5.1. The DSP executes the code in the dual-ported memory that can be also accessed by the host PC and the I/O units through the DSP bus. The digital I/O port consists of two 8-bit I/O configurable ports through which the PWM commands and control signals are sent to the inverter. The feedback from the current sensors are A/D converted with 14-bit resolution. Finally, the feedback from the encoder is handled by the encoder interface.

The vector control program has been developed in a Simulink environment and has been compiled by the dSpace code generator. The real-time executable code is downloaded to the DSP memory whose simplified map is shown in fig. [5.2](#). During the normal working of the drive, the vector control code is executed by the DSP which updates the outputs while the I/O units update the feedback values. The control parameters remain static and the control logic flags are used to synchronize the processes and to handle the faults. During the auto-tuning, the evaluation procedure of the HEA changes the control parameters and analyzes the response to the training test. This is performed by accessing the memory locations where the control parameters and the feedback values are stored using the dSpace MLIB Interface. In this way the auto-tuning procedure does not need to recompile the control code. Furthermore the evaluation procedure uses the control flags to make the auto-tuning independent of the training test time and the DSP execution code time.

It should be emphasized that the dSpace board and the Host PC have been used to reduce prototyping time. No extra-hardware has been used for the vector control implementation compared with an industrial drive. Moreover the HEA code has been written by the author in Matlab environment, starting from the Genetic Algorithm Toolbox [\[69\]](#), but for this application it can be optimized to take up a small part of the DSP memory. Hence this solution can be simply embedded in an industrial drive software as a self-commissioning tool.

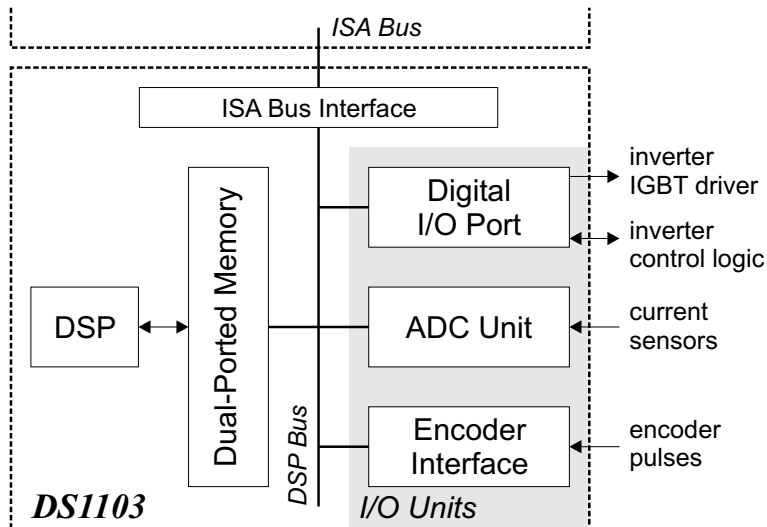


Figure 5.1: Diagram of the hardware implementation.

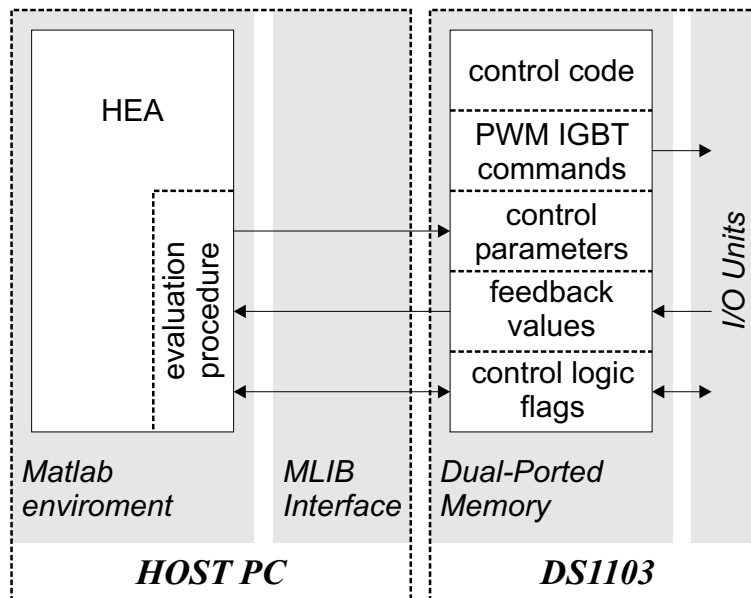


Figure 5.2: Diagram of the software implementation.

Table 5.1: PMSM Nameplate

| | |
|---------------------------|-----------------------------------|
| phase-to-phase resistance | 10.4 Ω |
| phase-to-phase inductance | 0.0087 H |
| voltage constant | 0.35 V/(rad/s) |
| torque constant | 0.40 Nm/A |
| moment of inertia | 0.00012 kg \cdot m ² |
| rated power | 350 W |
| rated speed | 4000 rpm |

5.2 Experimental Results

Besides a PC and the DS1103 board previously described, the experimental setup is constructed from a Technosoft ACPM750 three-phase inverter, a 150 W three-phase PMSM and a 250-pulse incremental encoder. A second torque controlled PMSM (Table 5.1) has been used for the load. The experimental setup is shown in figures 5.3 and 5.4.

As mentioned in chapter 2.5 an initial commissioning has been performed. The AVO and SO criteria have been applied to tune the parameters of the control system after a conventional output-error identification [70]. The control system parameters so obtained are reported in the first column of the Table 5.2 and represent the approximate solution \mathbf{x}_0 . In fact, because of approximations of the modelling and design, \mathbf{x}_0 has proven to be stable but with a poor dynamic response. For a fair comparison with the results given by the proposed on-line hybrid-evolutionary optimization, a further accurate manual calibration has been carried out. The new parameters represent the solution \mathbf{x}_{MC} and are reported in the second column of Table 5.2. The speed



Figure 5.3: Frontal view of the experimental setup.

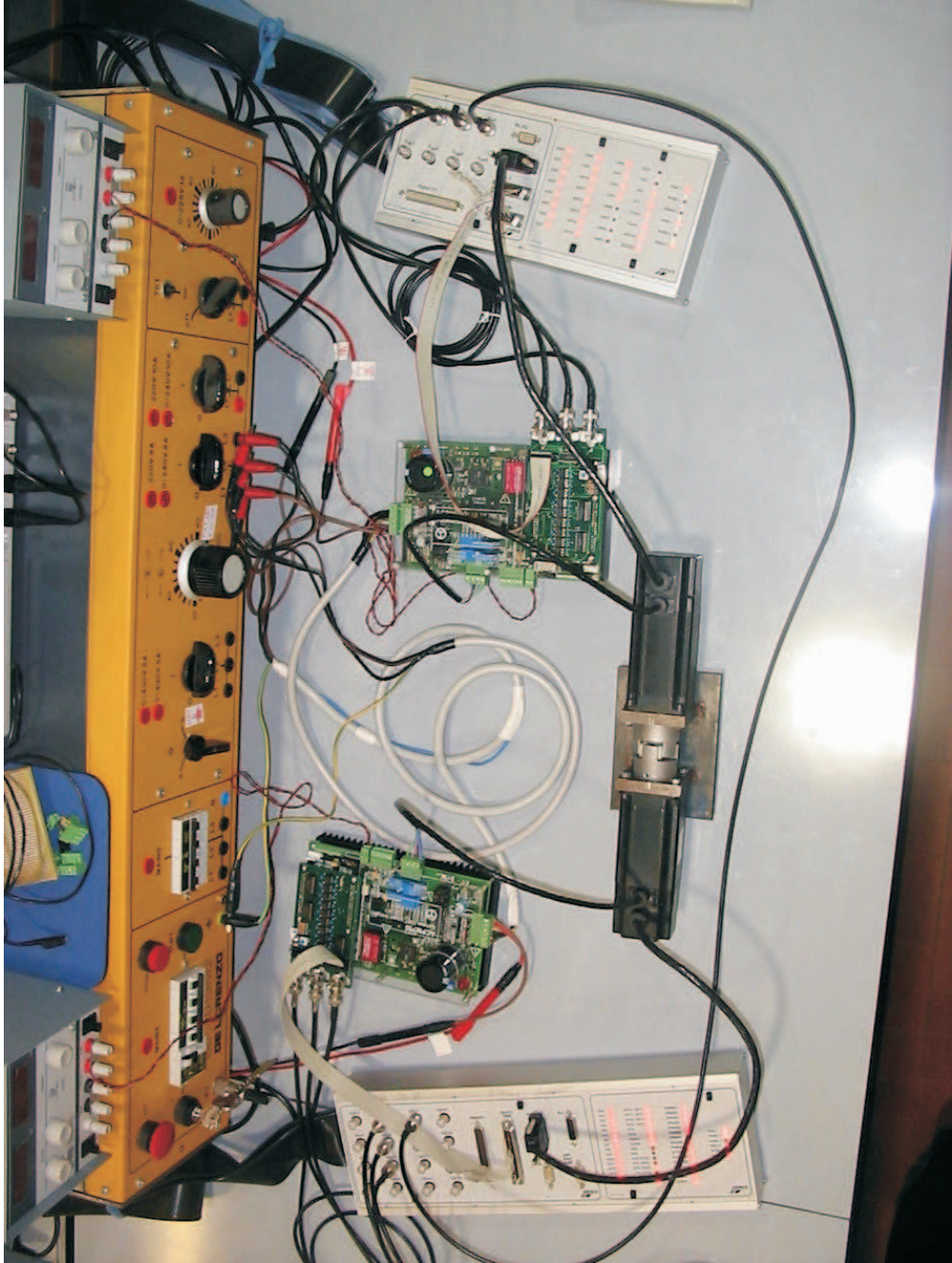


Figure 5.4: Top view of the experimental setup.

Table 5.2: Values of the Control System Parameters

| | initial solution, \mathbf{x}_0 | manually calibrated solution, \mathbf{x}_{MC} | HEA-designed solution, \mathbf{x}_{HEA} |
|-------------------|-------------------------------------|--|--|
| K_{isd} | 6.93 | 6.93 | 10.51 |
| τ_{isd} | $8.37 \cdot 10^{-4}$ | $8.36 \cdot 10^{-4}$ | $8.96 \cdot 10^{-4}$ |
| K_{isq} | 6.93 | 6.93 | 8.88 |
| τ_{isq} | $8.37 \cdot 10^{-4}$ | $8.36 \cdot 10^{-4}$ | $9.32 \cdot 10^{-4}$ |
| $K_{\omega r}$ | $12.10 \cdot 10^{-3}$ | $10.76 \cdot 10^{-3}$ | $13.90 \cdot 10^{-3}$ |
| $\tau_{\omega r}$ | $1.54 \cdot 10^{-2}$ | $2.24 \cdot 10^{-2}$ | $1.24 \cdot 10^{-2}$ |
| τ_{sm} | $1.54 \cdot 10^{-2}$ | $1.54 \cdot 10^{-2}$ | $1.55 \cdot 10^{-2}$ |
| K_1 | $4.35 \cdot 10^{-3}$ | $3.85 \cdot 10^{-3}$ | $3.75 \cdot 10^{-3}$ |
| K_2 | $4.35 \cdot 10^{-3}$ | $3.85 \cdot 10^{-3}$ | $3.94 \cdot 10^{-3}$ |
| K_3 | $6.36 \cdot 10^{-3}$ | $6.36 \cdot 10^{-3}$ | $5.59 \cdot 10^{-3}$ |

response to the training test is shown in fig. [5.5a](#), whilst the d- and q-axis current responses are shown in fig. [5.5b](#) and [5.5c](#) respectively. The current responses show oscillation, which is particularly large during the first speed reversal. A simply way to remove these oscillations would have consisted of speed dynamics decreasing with consequently deterioration of the speed response. Considering that the first objective is a good speed response, and that the amplitude of the current oscillations is acceptable, \mathbf{x}_{MC} has been chosen as the best solution after the hand-calibration. Essentially, we have retraced the typical steps of an engineer commissioning an industrial drive.

Subsequently the control system parameters have been optimized by the proposed on-line genetic auto-tuning. The solution, \mathbf{x}_{HEA} , is reported in the last column of Table [5.2](#). The repeatability of this final result has been validated by 30 optimization runs. The configuration of the HEA is synthe-

Table 5.3: Performance Indices Scored by \mathbf{x}_{MC} and \mathbf{x}_{HEA} after each speed step

| j | $Ftn_{1,j}$ | | $Ftn_{2,j}$ | | $Ftn_{3,j}$ | | $Ftn_{4,j}$ | |
|-----|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|
| | \mathbf{x}_{MC} | \mathbf{x}_{HEA} | \mathbf{x}_{MC} | \mathbf{x}_{HEA} | \mathbf{x}_{MC} | \mathbf{x}_{HEA} | \mathbf{x}_{MC} | \mathbf{x}_{HEA} |
| 1 | 0.37 | 0.21 | 0.52 | 0.43 | 0.08 | 0.04 | 0.01 | 0.01 |
| 2 | 0.74 | 0.41 | 0.74 | 0.65 | 0.23 | 0.14 | 0.05 | 0.04 |
| 3 | 1.11 | 0.62 | 0.84 | 0.75 | 0.36 | 0.24 | 0.25 | 0.09 |
| 4 | 1.47 | 0.83 | 0.95 | 0.84 | 0.44 | 0.28 | 0.28 | 0.11 |
| 5 | 1.48 | 0.84 | 1.40 | 1.10 | 0.48 | 0.41 | 0.29 | 0.13 |
| 6 | 1.49 | 0.85 | 1.52 | 1.22 | 0.62 | 0.49 | 0.32 | 0.16 |
| 7 | 1.49 | 0.86 | 1.72 | 1.42 | 0.69 | 0.55 | 0.38 | 0.21 |
| 8 | 1.50 | 0.85 | 1.83 | 1.53 | 0.80 | 0.69 | 0.40 | 0.23 |

sized in Table ???. As mentioned in section 3.2, only \mathbf{x}_0 has been used as a priori knowledge and not the more accurate \mathbf{x}_{MC} . The speed response to the training test is shown in fig. 5.6a, whilst the d- and q-axis current responses are shown in fig. 5.6b and 5.6c respectively. At a first glance these results are better than those given by \mathbf{x}_{MC} . However, to allow an easier comparison the values of the performance indices scored by \mathbf{x}_{MC} and \mathbf{x}_{HEA} are reported in Table 5.3. It should be noted that \mathbf{x}_{HEA} has better performance in terms of speed and current response to each speed step.

In particular, consider the details of the no-load speed reversals i.e. the responses given by \mathbf{x}_{MC} and \mathbf{x}_{HEA} to the third speed step of the training test (at $t = 1$ s), fig. 5.7. As regards the \mathbf{x}_{MC} speed response, the 5% settling time is 0.094 s and the overshoot is 11% (in fact the speed step amplitude is 2 p.u., from 1 p.u. to -1 p.u.). The 5% settling time and the overshoot of the \mathbf{x}_{HEA} speed response are 0.077 s and 8%, respectively. The superiority of

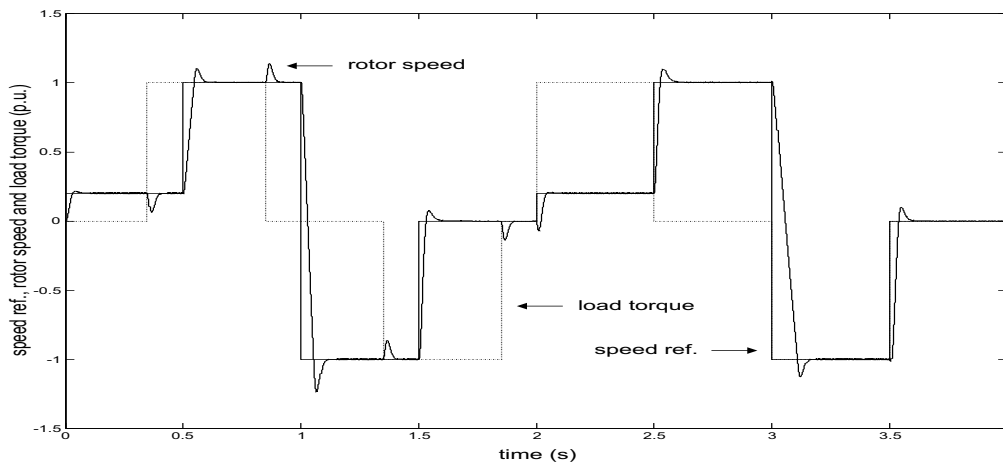
\mathbf{x}_{HEA} is also confirmed by the \mathbf{x}_{MC} current responses which show unwanted oscillations. Also during the full-load speed reversal the performance of \mathbf{x}_{HEA} is better, even though \mathbf{x}_{MC} provides a current response with few oscillations, as shown in fig. 5.8. The full-load speed reversal is the seventh speed step of the training test (at $t = 3$ s). The 5% settling time and the overshoot of \mathbf{x}_{MC} speed response are 0.140 s and 7%, respectively, whilst those of \mathbf{x}_{HEA} are 0.128 s and 5%, respectively.

As mentioned in section 3.3.2, each performance index is updated in real-time. In fig. 5.9 the real-time values of the objective functions (i.e. the weighted sum of the performance indices) given by \mathbf{x}_{MC} and \mathbf{x}_{HEA} are shown. The final objective values scored by \mathbf{x}_{MC} and \mathbf{x}_{HEA} are 4.53 and 3.30 respectively. Hence the overall performance obtained with an accurate hand-calibration is about 37% worse than that achieved with the proposed on-line genetic optimization.

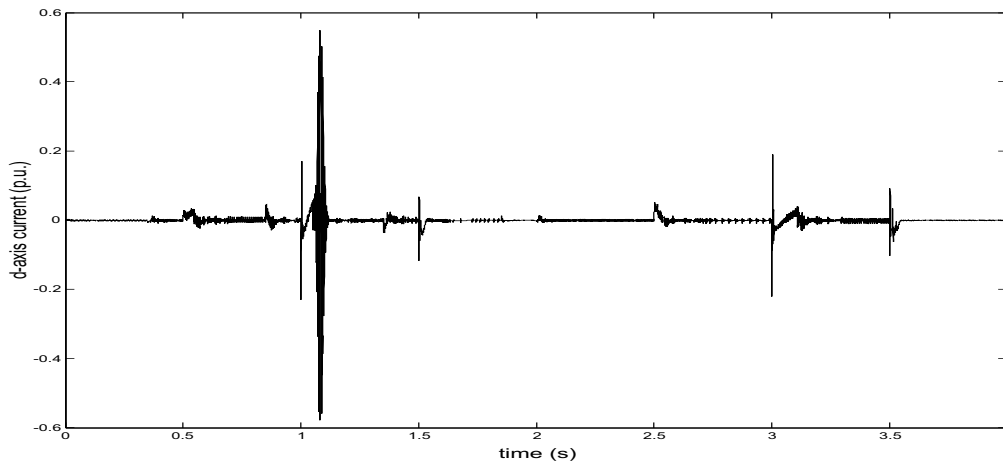
To evaluate the robustness of \mathbf{x}_{MC} and \mathbf{x}_{HEA} , two evaluation tests have been performed. The aim is to stress the control system in conditions differing from those used for its design. The first evaluation test is designed to stress the control system with an alternative load torque. It consists of a set of experiments in which the motor is initially started at half rated speed with no-load torque. After the steady-state is reached, a square-wave load torque with amplitude 0.4 p.u. and frequency from 10 Hz to 26 Hz is applied. For sake of clearness, the speed responses of the experiment with 15 Hz square-wave load torque is shown in fig. 5.10. Each square-wave load torque generates a speed oscillation with the same frequency. The amplitude of each speed oscillation is reported in the Bode diagram in fig. 5.11. The physical implementation of

the load torque does not allow to go beyond a frequency of 30 Hz. In spite of this, it is possible to recognize the \mathbf{x}_{MC} amplitude frequency response as that of an over damped system. In fact, up to 13 Hz the amplitude value is constant, 0.108 p.u., and then is strictly decreasing. As regards the \mathbf{x}_{HEA} amplitude frequency response, the 17 Hz resonant peak shows the behavior of an under damped system. Up to 13 Hz the amplitude value is constant, 0.082 p.u., and after the resonant peak, 0.089 p.u., it is strictly decreasing. This evaluation test confirms the superiority of \mathbf{x}_{HEA} whose amplitude frequency response is always under that of \mathbf{x}_{MC} .

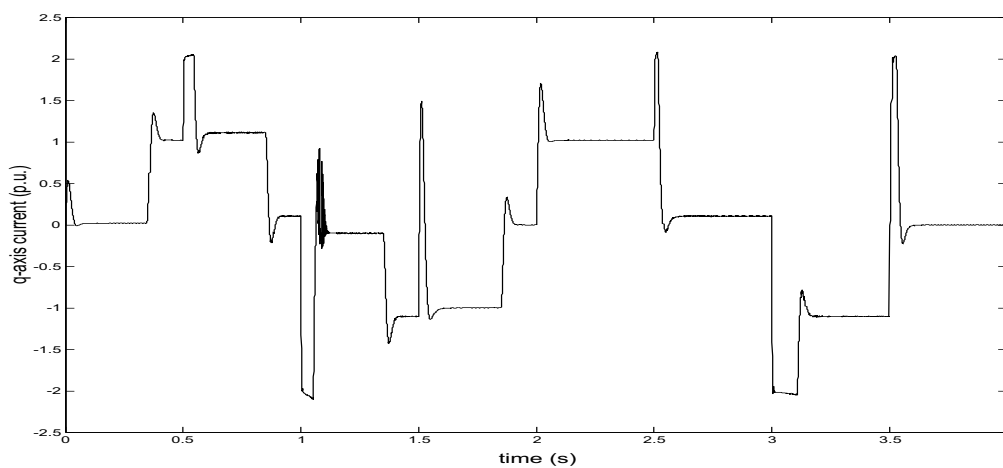
The speed and current responses to the second evaluation test are shown in fig. 5.12a, 5.12b, and 5.12c. Initially, a speed step is applied, whilst the load torque is set to zero. At $t = 0.01$ s the full-load torque is applied, and at $t = 0.05$ s the reference speed is set to zero again. At first, both \mathbf{x}_{MC} and \mathbf{x}_{HEA} responses are quite similar. In fact, both speed responses are ramps in which the acceleration in the first part (from $t = 0$ s to $t = 0.035$ s, no-load torque) is higher than that of the second part (from $t = 0.01$ s to $t = 0.05$ s, full-load torque). At $t = 0.05$ s the speed reference is set to zero whilst the full-load torque is kept. At that time the regulators are still in saturation, in fact the i_{sq} is the double of rated value (choose as maximum value allowable) and the rotor speed is still far enough from the rated speed. It should be noted that the training test does not provide this situation reproduced by the evaluation test. The 5% settling time of \mathbf{x}_{MC} and \mathbf{x}_{HEA} speed responses are equal to 0.106 s and 0.089 s respectively, confirming the superiority of the latter.



(a) Speed response.

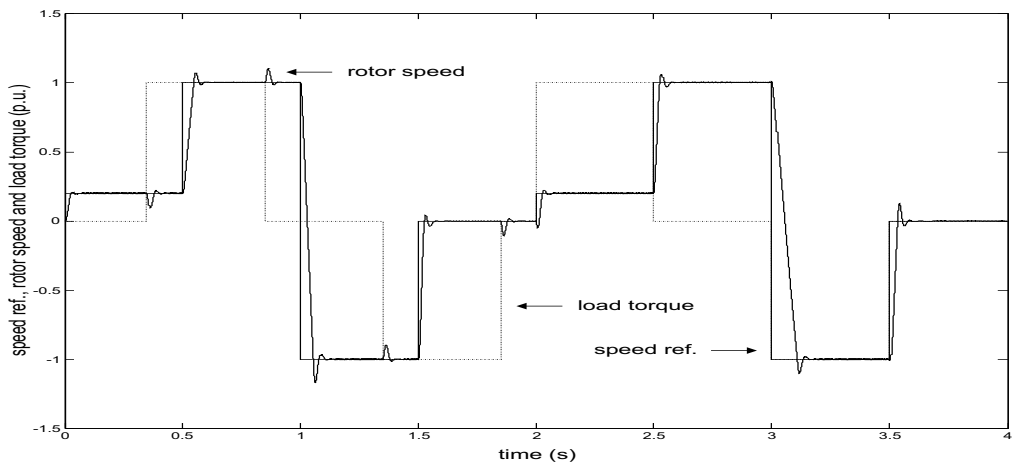


(b) d-axis current response.

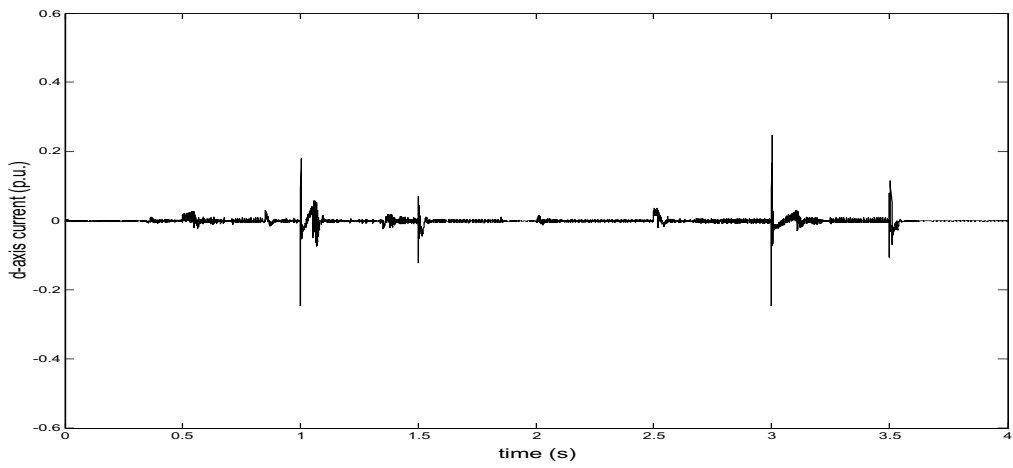


(c) q-axis current response.

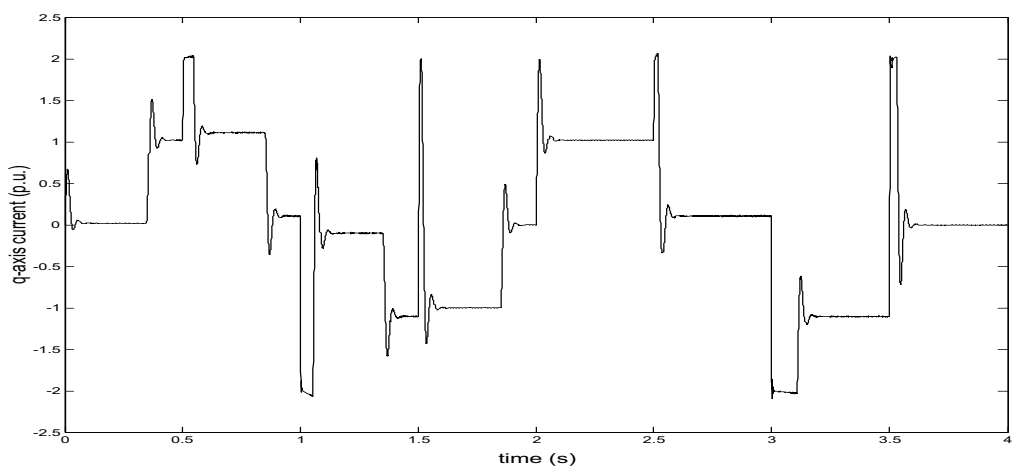
Figure 5.5: Responses given by x_{MC} .



(a) Speed response.

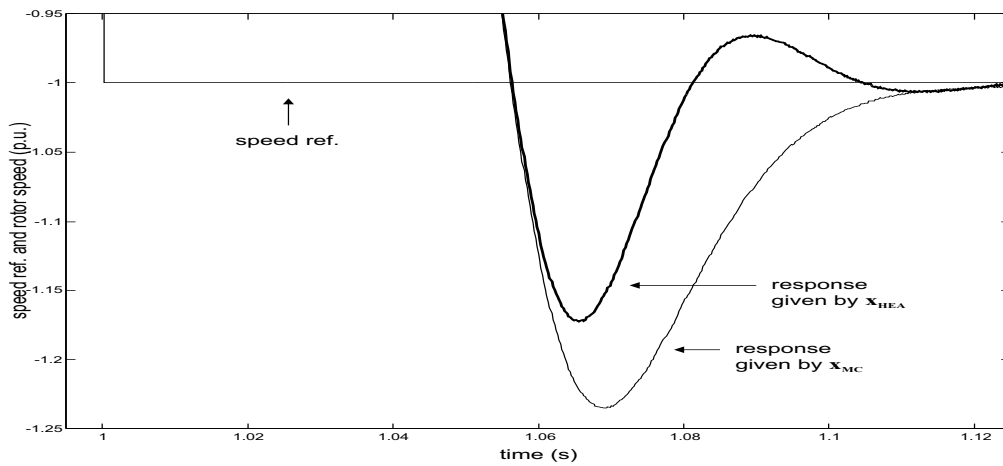


(b) d-axis current.

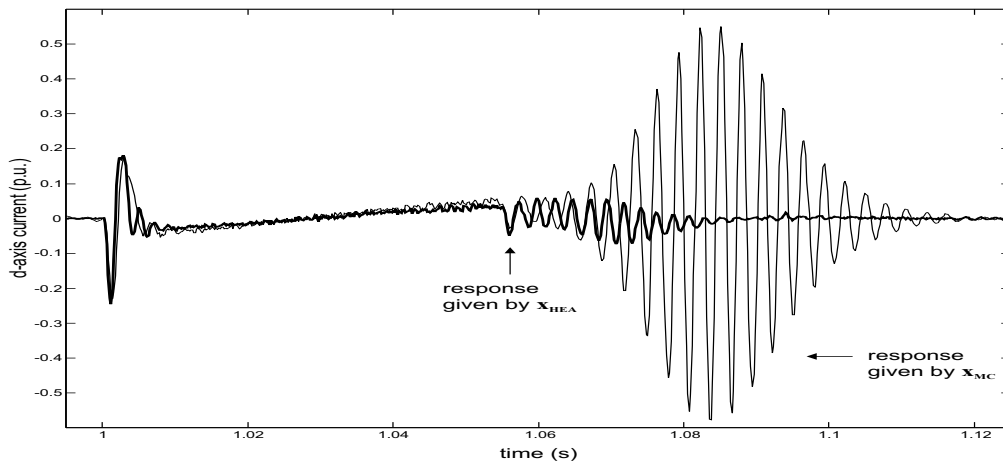


(c) q-axis current.

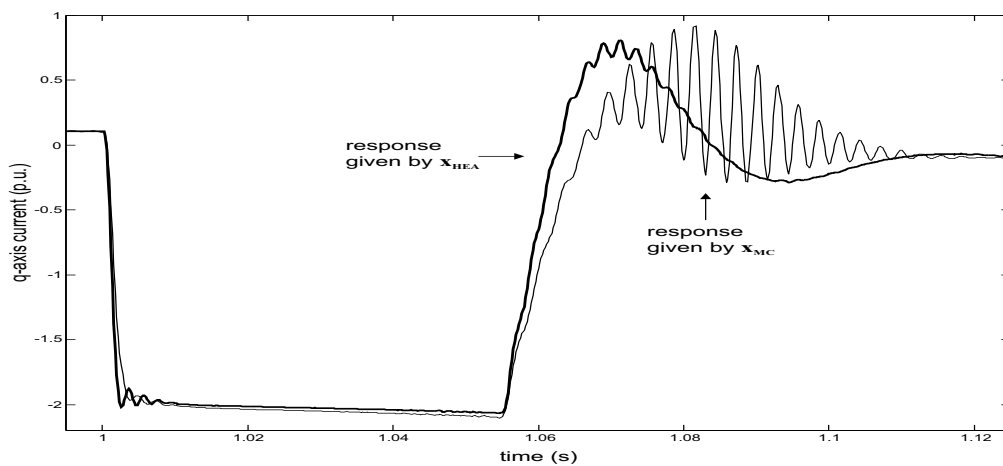
Figure 5.6: Responses given by \mathbf{x}_{HEA} .



(a) Speed responses.

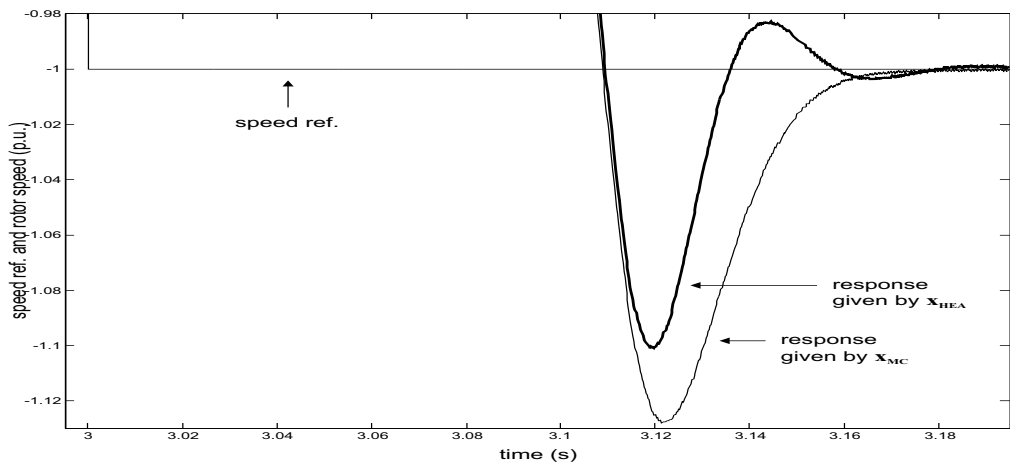


(b) d-axis current responses.

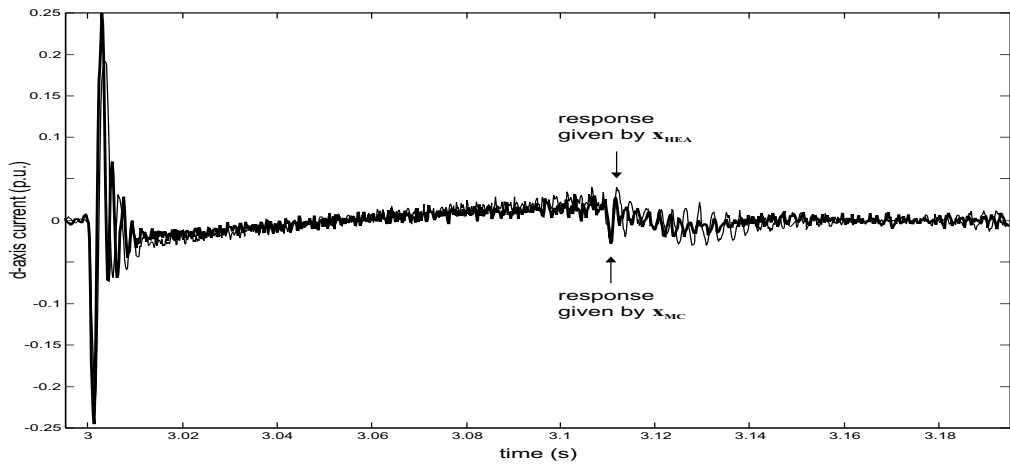


(c) q-axis current responses.

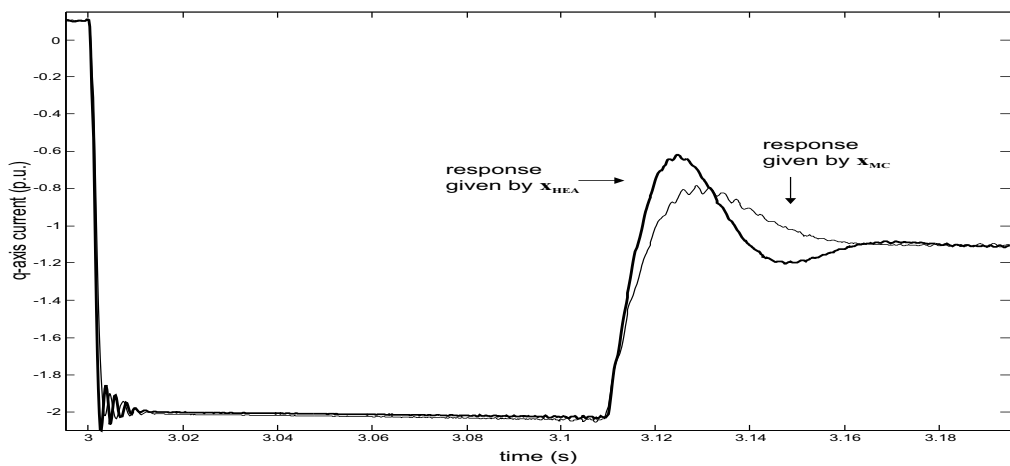
Figure 5.7: Comparison between the responses given by x_{MC} and x_{HEA} to the no-load speed reversal.



(a) Speed responses.



(b) d-axis current responses.



(c) q-axis current responses.

Figure 5.8: Comparison between the responses given by x_{MC} and x_{HEA} to the full-load speed reversal.

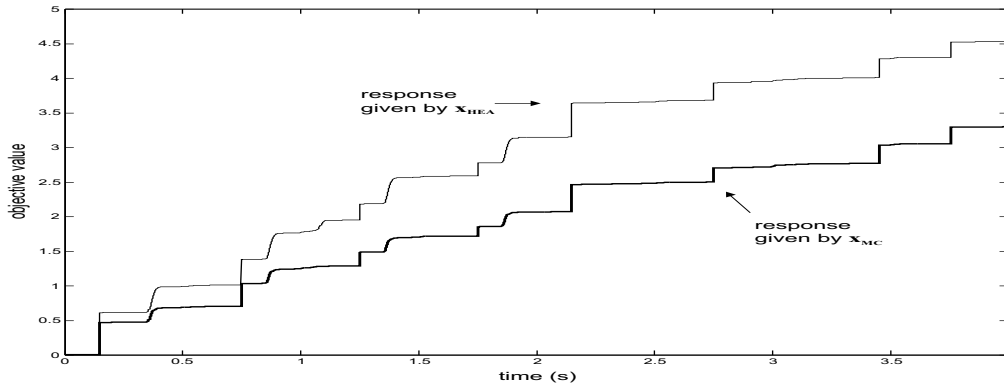


Figure 5.9: Comparison between the objective functions provided by x_{MC} and x_{HEA} .

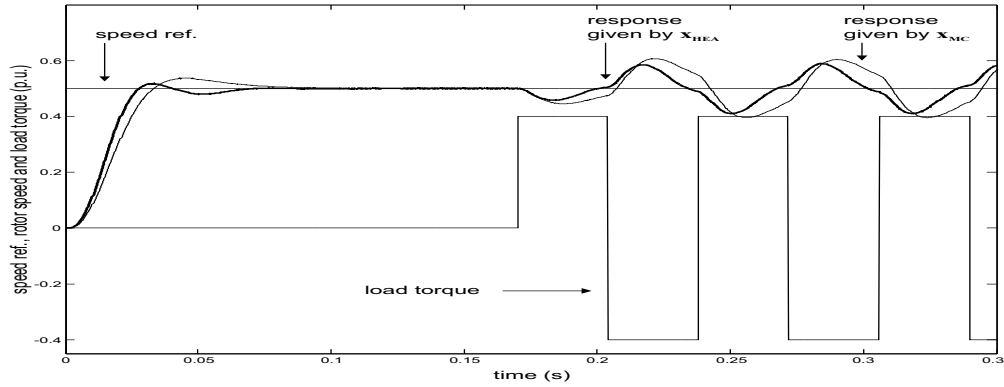


Figure 5.10: Experiment to evaluate the amplitude of the speed oscillation when a 15 Hz square-wave load torque is applied.

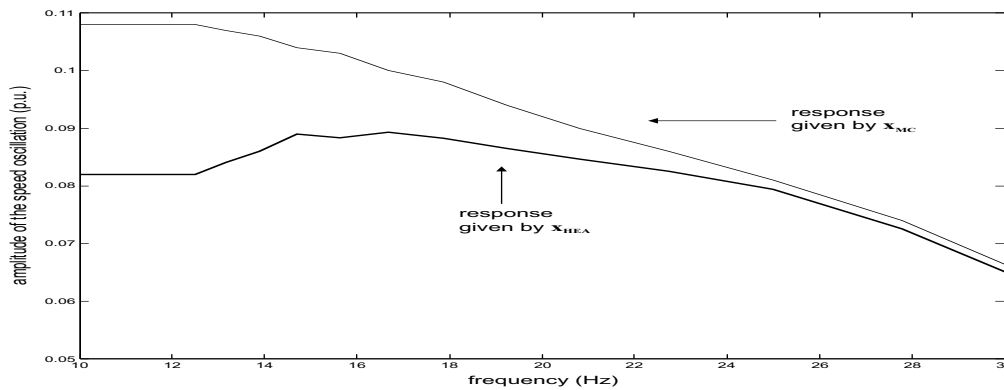
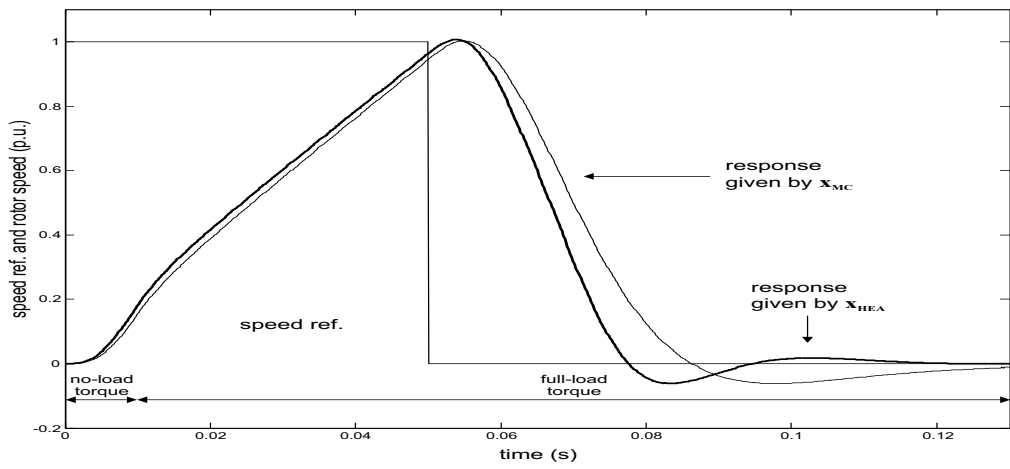
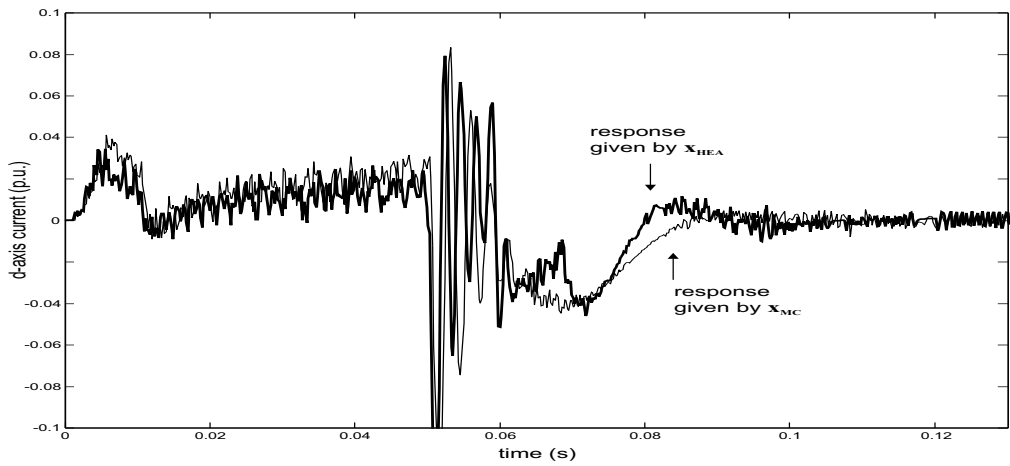


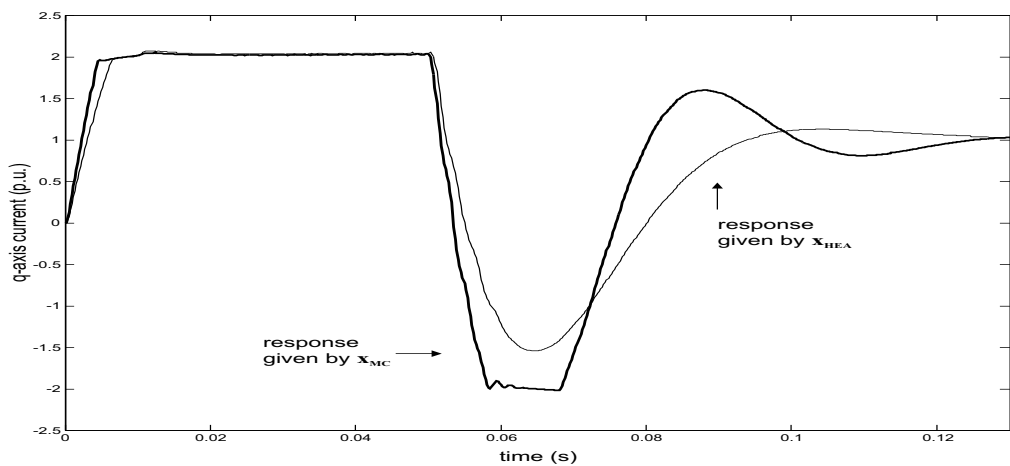
Figure 5.11: Comparison between the amplitude frequency responses provided by x_{MC} and x_{HEA} .



(a) Speed responses.








(b) d-axis current responses.



(c) q-axis current responses.

Figure 5.12: Comparison between the responses given by x_{MC} and x_{HEA} to the second evaluation test.

Bibliography

- [1] E. I. Administrator, Ed., *Monthly Energy Review*. Washington DC, USA: U.S. Department of Energy, Dec. 2004. [Online]. Available: <http://www.eia.doe.gov/mer> 
- [2] (2001, Feb.) Company news from frost and sullivan, electric drives market prove to be dynamic. [Online]. Available: <http://www.manufacturingtalk.com> 
- [3] (2003, Sept.) News from control engineering, economic turnaround to boost europe's electric drives market. [Online]. Available: <http://www.manufacturing.net/ctl> 
- [4] R. Sanders, "Physicists build worlds smallest motor using nanotubes and etched silicon," *Berkeley University of California, UC Berkeley News, NewsCenter*, Dec. 2004. [Online]. Available: http://www.berkeley.edu/news/media/releases/2003/07/23_motor.shtml 
- [5] "Energy efficiency: Challenges and opportunities for electric utilities," Office of Technology Assessment, U.S. Congress, Washington DC, USA, Tech. Rep., Sept. 1993. [Online]. Available: <http://www.wws.princeton.edu/cgi-bin/byteserv.prl/~ota> 

- [6] W. Leonhard, *Modern Power Electronics and AC Drives*. Upper Saddle River, NJ, USA: Prentice Hall, 2002. [1.1](#), [2.5](#)
- [7] K. J. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed. USA: Addison-Wesley Publishing Co., 1995. [1.2](#), [2.5](#), [2.5](#)
- [8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley Publishing Co., 1989. [1.2](#), [3](#)
- [9] P. J. Fleming and R. C. Purshouse, “Evolutionary algorithms in control systems engineering: a survey,” *Control Engineering Practice*, vol. 10, pp. 1223–1241, Nov. 2002. [1.2](#)
- [10] T. J. E. Miller and J. R. Hendershot, *Design of Brushless Permanent Magnet Motors*. Clarendon Press, Oxford Science Publications and Magna Physics Publishing, 1994. [2.1](#)
- [11] S. E. Lyshevski, *Electromechanical Systems, Electric Machines, and Applied Mechatronics*. Boca Raton, Florida, USA: CRC Press LCC, 2000. [2.2](#)
- [12] S. Morimoto, Y. Takeda, and T. Hirasaka, “Current phase control methods for pmsm,” *IEEE Transactions on Power Electronics*, vol. 5, no. 2, pp. 133–138, Apr. 1990. [2.3](#)
- [13] P. Pillay and R. Krishnan, “Modeling, simulation, and analysis of permanent-magnet motor drives. i. the permanent-magnet synchronous

- motor drive,” *IEEE Transactions on Industry Applications*, vol. 25, no. 2, pp. 274 – 279, Mar./Apr. 1989. [2.3](#)
- [14] T. M. Jahns, G. B. Kliman, and T. W. Neumann, “Interior permanent -magnet synchronous motors for adjustable-speed drives,” *IEEE Transactions on Industry Applications*, vol. 22, no. 4, pp. 738–747, July/ 1986. [2.3](#)
- [15] R. Krishnan, *Electronic Motor Drives: Modeling, Analysis and Control*. Upper Saddle River, New Jersey, USA: Prentice Hall, Feb. 2001. [2.3](#), [2.4](#)
- [16] R. Colby and D. Novotny, “Efficient operation of pm synchronous motors,” *IEEE Transactions on Industry Applications*, vol. 23, no. 4, pp. 1048–1054, Nov./Dec. 1987. [2.3](#)
- [17] S. Morimoto, Y. Tong, Y. Takeda, and T. Hirasaka, “Loss minimization control of permanent magnet synchronous motor drives,” *IEEE Transactions on Industry Applications*, vol. 41, no. 5, pp. 511–517, Oct. 1994. [2.3](#)
- [18] W. Leonhard, *Control of Electrical Drives*, 2nd ed. Berlin, Germany: Springer-Verlag, 1996. [2.4](#)
- [19] H. Gross, J. Hamann, and G. Wieg, *The control techniques drives and controls handbook*. Cambridge, UK: Cambridge University Press, 2001, vol. 35. [2.4](#)

- [20] H. Schierling, "Self-commissioning-a novel feature of modern inverter-fed induction motor drives," in *Third International Conference on Power Electronics and Variable-Speed Drives*, London, UK, July 1988, pp. 287–290. [2.5](#)
- [21] A. M. Khambadkone and J. Holtz, "Vector-controlled induction motor drive with a self-commissioning scheme," *IEEE Transactions on Industrial Electronics*, vol. 38, no. 5, pp. 322–327, Oct. 1991. [2.5](#)
- [22] M. Sumner and G. M. Asher, "Autocommissioning for voltage referenced voltage-fed vector controlled induction motor drives," *IEE Proceedings-B*, vol. 140, no. 3, pp. 187–200, May 1993. [2.5](#)
- [23] C. Wang, D. W. Novotny, and T. A. Lipo, "An automated rotor time-constant measurement system for indirect field-oriented drives," *IEEE Transactions on Industry Applications*, vol. 24, no. 1, pp. 151–159, Jan./Feb. 1988. [2.5](#)
- [24] M. Sumner, G. M. Asher, and R. Pena, "The experimental investigation of rotor time constant identification for vector controlled induction motor drives during transient operating conditions," in *Proc. of the Fifth European Conference on Power Electronics and Applications*, vol. 5, Brighton, UK, Sept. 1993, pp. 51–56. [2.5](#)
- [25] M. Bertoluzzo, G. S. Buja, and R. Menis, "Self-commissioning of rfo im drives: One-test identification of the magnetization characteristic of the motor," *IEEE Transactions on Industry Applications*, vol. 37, no. 6, pp. 1801–1806, Nov./Dec. 2001. [2.5](#)

- [26] Y.-Y. Tzou, S.-T. Yeh, and H. Wu, “Dsp-based rotor time constant identification and slip gain auto-tuning for indirect vector-controlled induction drives,” in *Proc. of IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, vol. 2, Aug. 1996, pp. 1228–1233. [2.5](#)
- [27] Y.-Y. Tzou, H. Wu, S.-Y. Lin, and J.-Y. Lin, “Auto-tuning self-commissioning control of a universal drive,” in *Proc. of PESC 98 Record, 29th Annual IEEE Power Electronics Specialists Conference*, vol. 1, May 1998, pp. 287–293. [2.5](#)
- [28] T. Senjyu, K. Kinjo, N. Urasaki, and K. Uezato, “Parameter measurement for pmsm using adaptive identification,” in *Proc. of the 2002 IEEE International Symposium on Industrial Electronics*, vol. 3, L’Aquila, Italy, May 2002, pp. 711–716. [2.5](#)
- [29] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ, USA: Pearson Education, 1991. [2.5](#)
- [30] F. Khorrami, P. Krishnamurthy, and H. Melkote, *Modeling and Adaptive Nonlinear Control of Electric Motors*. Heidelberg, Germany: Springer-Verlag, Aug. 2003. [2.5](#)
- [31] L. C. Jain, C. W. D. Silva, and L. C. Jain, *Intelligent Adaptive Control: Industrial Applications*. Boca Raton, FL, USA: CRC Press, Inc., Dec. 1998. [2.5](#)
- [32] F. Cupertino, A. Lattanzi, and L. Salvatore, “A new fuzzy logic-based controller design method for dc and ac impressed-voltage drives,” *IEEE*

- Transactions on Power Electronics*, vol. 15, no. 6, pp. 974–982, Nov. 2000. [2.5](#)
- [33] V. I. Utkin, J. Guldner, and J. Shi, *Sliding Mode Control in Electro-mechanical Systems*. Philadelphia, PA, USA: Taylor & Francis, May 1999. [2.5](#)
- [34] I.-C. Baik, K.-H. Kim, and M.-J. Youn, “Robust nonlinear speed control of pm synchronous motor using adaptive and sliding mode control techniques,” *IEE Proceedings-Electric Power Applications*, vol. 145, no. 4, pp. 369–376, July 1998. [2.5](#)
- [35] K. J. Åström and T. Hägglund, “The future of pid control,” *Control Engineering Practice*, vol. 9, pp. 1163–1175, 2001. [2.5](#)
- [36] H. J. Bremermann, “The evolution of intelligence. the nervous system as a model of its environment,” Dept. of Mathematics, Univ. of Washington, Seattle, USA, Technical Report, contract no. 477(17) 1, July 1958. [3](#)
- [37] J. Holland, “Outline for a logical theory of adaptive systems,” *Journal of Association for Computing Machinery*, vol. 9, no. 3, pp. 297–314, July 1962. [3](#)
- [38] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, Apr. 1992. [3](#)

- [39] ———, “Genetic algorithms,” *Scientific American*, vol. 278, no. 1, pp. 66–72, July 1992. [3](#)
- [40] J. J. Grefenstette, *Foundations of Genetic Algorithms 2*. San Mateo, CA, USA: Morgan Kaufmann Publishers, Inc., 1993, ch. 3. [3](#)
- [41] D. B. Fogel and A. Ghozeil, “Schema processing under proportional selection in the presence of random effects,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 290–293, Nov. 1997. [3](#)
- [42] R. Poli, “Why the schema theorem is correct also in the presence of stochastic effects,” in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, July 2000, pp. 487–492. [3](#)
- [43] N. J. Radcliffe, *Handbook of Evolutionary Computation*. Institute of Physics Publishing, 1997, ch. 2. [3](#)
- [44] D. B. Fogel and R. W. Anderson, “Revisiting bremermann’s genetic algorithm. i. simultaneous mutation of all parameters,” in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 2, July 2000, pp. 1204–1209. [3](#)
- [45] Z. Michalewicz, *Genetic algorithms+data structures=evolution programs*. Berlin, Germany: Springer-Verlag, 1996. [3](#), [3.1](#)
- [46] C. Z. Janikow and Z. Michalewicz, “An experimental comparison of binary and floating point representation in genetic algorithms,” in *Proc. of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, USA, July 1991, pp. 31–36. [3](#)

- [47] H. Schwefel, *Numerical Optimization of Computer Models*. Chichester, England, UK: Wiley, 1981. [3](#)
- [48] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Dec. 1992. [3](#)
- [49] A. Geyer-Schultz, “Holland classier systems,” in *Proc. of the International Conference on Applied Programming Languages*, San Antonio, USA, June 1995, pp. 45–55. [3](#)
- [50] L. J. Fogel, *Intelligence Through Simulated Evolution : Forty Years of Evolutionary Programming*. New York, NY, USA: Wiley-Interscience Publication, July 1999. [3](#)
- [51] W. G. da Silva, P. P. Acarnley, and J. W. Finch, “Application of genetic algorithms to the online tuning of electric drive speed controllers,” *IEEE Transactions on Industrial Electronics*, vol. 47, pp. 217–219, Feb. 2000. [3](#)
- [52] F. Cupertino, V. Giordano, D. Naso, B. Turchiano, and L. Salvatore, “On-line genetic design of fuzzy controllers for dc drives with variable load,” *Electronics Letters*, vol. 39, pp. 479–480, Mar. 2003. [3](#), [3.3.2](#)
- [53] F. Cupertino, E. Mininno, D. Naso, B. Turchiano, and L. Salvatore, “On-line genetic design of anti-windup unstructured controllers for electric drives with variable load,” *IEEE Transaction on Evolutionary Computation*, vol. 8, pp. 347–364, Aug. 2004. [3](#)

- [54] ———, “On-line genetic optimization of unstructured controllers for electric drives,” *IEEE Transaction on Evolutionary Computation*, vol. 8, pp. 347–364, Aug. 2004. [3](#)
- [55] P. Schroder, B. Green, N. Grum, and P. J. Fleming, “On-line evolution of robust control systems: An industrial active magnetic bearing application,” *Control Engineering Practice*, vol. 9, pp. 37–49, Jan. 2001. [3.2](#)
- [56] J. E. Baker, “Adaptive selection methods for genetic algorithms,” in *Proc. of the First International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 101–111. [3.4.1](#)
- [57] D. Whitley, “The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best,” in *Proc. of the Third International Conference on Genetic Algorithms*, 1989, pp. 116–121. [3.4.1](#)
- [58] J. E. Baker, “Reducing bias and inefficiency in the selection algorithm,” in *Proc. of the Second International Conference on Genetic Algorithms*, 1987, pp. 14–21. [3.4.1](#)
- [59] H. Mühlenbein and D. Schlierkamp-Voosen, “Predictive models for the breeder genetic algorithms: I. continuous parameter optimization,” *Evolutionary Computation*, vol. 1. [3.4.2](#), [3.4.3](#)
- [60] J. Yen, J. C. Liao, B. Lee, and D. Randolph, “A hybrid approach to modeling metabolic systems using genetic algorithms and the simplex method,” *IEEE Transaction on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 28, pp. 112–147, Apr. 1998. [4](#), [4.2](#)

- [61] J. Renders and S. P. Flasse, “Hybrid methods using genetic algorithms for global optimization,” *IEEE Transaction on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 26, pp. 243–258, Apr. 1996. [4](#)
- [62] A. Nelder and R. Mead, “A simplex method for function optimization,” *Computation Journal*, vol. 7, pp. 308–313, 1965. [4](#), [4.1.3](#)
- [63] V. Torczon, “On the convergence of the multidirectional search algorithm,” *SIAM Journal on Optimization*, vol. 1, pp. 123–145, Feb. 1991. [4](#)
- [64] J. E. Denis and V. Torczon, “Direct search methods on parallel machines,” *SIAM Journal on Optimization*, vol. 1, pp. 448–474, Nov. 1991. [4](#)
- [65] T. G. Kolda, R. M. Lewis, and V. Torczon, “Optimization by direct search: new perspectives on some classical and modern methods,” *SIAM Review*, vol. 45, pp. 385–482, 2003. [4](#)
- [66] J. Yen, J. C. Liao, B. Lee, and D. Randolph, “A hybrid approach to modeling metabolic systems using genetic algorithms and the simplex method,” Los Angeles, CA, USA, Feb. 1995, pp. 277–283. [4.2](#)
- [67] V. Torczon, “Multi-directional search: A direct search algorithm for parallel machines,” Ph.D. dissertation, Rice University, Huston, Texas, USA, 1989. [4.3](#)
- [68] *DS1103 PPC Controller Board Hardware Reference*, dSPACE GmbH, Paderborne, Germany, 2003. [5.1](#)

- [69] Department of Automatic Control and Systems Engineering of The University of Sheffield. (1994) Genetic algorithm toolbox. [Online]. Available: <http://www.shef.ac.uk/~gaipp/ga-toolbox/> 5.1
- [70] L. Ljung, *System Identification: a Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1992. 5.2