Automatic Torque Ripple Reduction in Permanent Magnet Machines

Joni Airaksinen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 25.6.2020

Supervisor

Prof. Marko Hinkkanen

Advisors

MSc Sina Khamehchi

MSc Victor Mukherjee

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Joni Airaksinen

Title Automatic Torque Ripple Reduction in Permanent Magnet Machines

Degree programme Automation and Electrical Engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC3025

Supervisor Prof. Marko Hinkkanen

Advisors MSc Sina Khamehchi, MSc Victor Mukherjee

Date 25.6.2020 **Number of pages** 56 **Language** English

Abstract

This thesis investigates the feasibility of Q-learning for torque ripple reduction with industrial permanent magnet (PM) machines. A practical Q-learning based compensator is implemented and the ripple reduction performance is compared against conventional PI speed control and iterative learning control (ILC). Performance of the Q-learning based method is evaluated by rigorous simulations and experiments. The results confirm the applicability of the Q-learning based method. The compensation performance is similar to ILC. In certain operating conditions, the Q-learning based method can outperform the ILC compensator.

Keywords Iterative learning control (ILC), permanent magnet synchronous motor (PMSM), Q-learning, torque ripple

Aalto University
School of Electrical Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Joni Airaksinen

Työn nimi Vääntömomenttivärähtelyn automaattinen kompensointi kestomagneettitahtikoneissa

Koulutusohjelma Automaatio ja sähkötekniikka

Pääaine Control, Robotics and Autonomous Systems **Pääaineen koodi** ELEC3025

Työn valvoja Prof. Marko Hinkkanen

Työn ohjaajat MSc Sina Khamehchi, MSc Victor Mukherjee

Päivämäärä 25.6.2020 **Sivumäärä** 56 **Kieli** Englanti

Tiivistelmä

Tässä työssä tutkitaan, mikäli Q-oppimista voidaan hyödyntää kestomagneettitahtikoneiden yhteydessä esiintyvien vääntömomenttivärähdysten minimointiin. Q-oppimiseen pohjautuvan kompensaattorin suorituskykyä verrataan tavanomaiseen PI-nopeussäätöön ja iteratiivisesti oppivaan säätöön (ILC). Vääntömomenttivärähdysten kompensointia testataan sekä kokeellisesti että simuloimalla. Tulokset vahvistavat menetelmän soveltuvuuden. Q-oppimiseen pohjautuvan kompensaattorin suorituskyky on verrannolinen ILC:hen ja tietyissä toimintapisteissä metodi pystyy suoriutumaan paremmin kuin ILC.

Avainsanat Iteratiivinen oppiva säätö (ILC), kestomagneettitahtikone, Q-oppiminen, värähtely, vääntömomentti

Preface

I am grateful to my thesis supervisor Prof. Marko Hinkkanen and instructors Sina Khamehchi and Victor Mukherjee for giving me valuable feedback and guidance. Furthermore, I would like to also thank Samuli Heikkilä and Juha Virtakoivu for helping me with practical matters. Finally, I would like to thank Marko Huikuri, Antti Vilhunen, Teppo Pirttioja and everyone else who helped me to make the thesis better.

Otaniemi, 25.6.2020

Joni Airaksinen

Contents

Symbols and abbreviations

Symbols

B	viscous damping coefficient
dq	direct-quadrature reference frame
f_N	nominal frequency
G_t	discounted reward
I_N	nominal current
i_d, i_q	stator currents of dq-axes
i_q^*	q-axis current reference
J	total inertia
k_t	torque coefficient
N_p	number of poles
N_s	number of stator slots
N_a	number of actions
P_N	nominal power
p	number of polepairs
r_t	single reward
T_N	nominal torque
T_m	electromagnetic torque
T_{cog}	cogging torque
T_{max}	periodical maximum of pulsating torque
T_l	load torque
T_N	torque related to single harmonic
T_Δ	torque related to current measurement errors
U_N	nominal voltage
α	forgetting factor or learning factor
γ	discount factor
Γ	ILC gain
ϵ	probability value
λ	reward weighting factor
ω_N	nominal speed
ω_m	mechanical angular speed of the shaft
θ_e	electrical rotor angle
θ_m	mechanical rotor angle
θ	rotor angle, θ_e / θ_m
Φ	ILC gain
ψ_d, ψ_q	stator flux linkages of dq-axes
ψ_f	rotor flux linkage

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\sum_i$	sum over index i
$\lvert a \rvert$	absolute value of a scalar
$\mathrm{lcm}(a, b)$	least common multiple between a and b
$\leftarrow$	assignment

Abbreviations

AC	alternating current
DC	direct current
FEA	finite element analysis
FEM	finite element method
FFT	fast Fourier transform
ILC	iterative learning control
MDP	Markov decision process
MMF	magnetomotive force
PI	proportional integral
PM	permanent magnet
PMSM	permanent magnet synchronous motor
PMW	pulse width modulation
RAM	random access memory
RL	reinforcement learning
SyRM	synchronous reluctance motor

1 Introduction

Permanent magnet (PM) machines are commonly used in high-performance applications due to numerous favorable characteristics, including high efficiency, power density and reliability. However, PM machines also incorporate inherent torque ripple, which causes mechanical vibrations, acoustic noise and uneven torque production leading to degraded controllability of the machine [1]. Inevitably, the performance of PM machines can be improved by minimizing the torque ripple.

The torque ripple originates from various phenomena. In sight of ripple minimization, the most important ripple sources consist of cogging torque, flux harmonics and current measurement errors. Not only have been these sources commonly acknowledged [1–4], but also their contribution to the ripple can be reduced by improving control of the machine. In addition to these three ripple sources, also others exist, yet these are not considered in this thesis due to significantly lower performance impact.

Many solutions have been proposed for minimizing the torque ripple. In general, these solutions can be divided into two categories [1, 3–6]. The methods in the first category aim to improve the geometry of the machine, thus leading to reduced torque ripples. Although, these methods are ordinarily effective, the approach can be impractical and costly to realize, if the machine has been manufactured already. The methods in the second category aim to compensate for the ripple by improving control algorithms. These approaches have the benefit of being cost-effective, since software changes can be applied to already existing control platform [1]. The thesis focus will be in the methods of the second category, as the objective is to reduce torque ripple occurring in already existing machines.

Iterative learning control (ILC) seems one of the most prominent compensation methods in the light of current research. ILC is easy to take into use, compensation is automatic and the method only requires measuring of the rotor speed. Due to these attractive features, ILC is studied extensively. Unfortunately, ILC also has some shortcomings. For example, the compensator must be tuned carefully and its behaviour is not always fully predictable. Careful gain tuning is time consuming and may require domain expertise. Furthermore, fixed gain values may not work in all operating conditions and hence the gains may require retuning if operating conditions change. Some ILC based compensation schemes have been also patented [7, 8], thus potentially restricting use of ILC.

During recent years, reinforcement learning (RL) has increasingly gained popularity and RL methods have been successfully used for solving various complex problems automatically. In [9, 10], RL was used to solve control problems, such as balancing an inverted pendulum and controlling a gripper in a simulation environment. In [11], RL was used to train an agent to play Atari 2600 games. In many games, the agent was able to achieve a performance level comparable to a professional game tester. In [12], RL was utilized to train an agent capable of defeating the European Go champion. By considering the complexity of the previous problems, it can be reasoned that RL methods could be potentially used for building a compensator.

Q-learning is a reinforcement learning algorithm, which can be used to learn

a policy that compensates for torque pulsations. The compensator based on this algorithm can overcome the weaknesses of ILC while also maintaining its key strengths. Similarly to ILC, the algorithm is computationally light, compensation is automatic and the compensator structure is modular. All control actions are known in advance with Q-learning algorithm, which makes it easy to predict the system behaviour. The adjustable parameters are easy to tune and tuning must be performed only once. Despite the numerous benefits, no work has yet attempted to evaluate the applicability of Q-learning in reducing torque ripple in industrial PM motors.

This thesis aims to demonstrate applicability of Q-learning for building a modular compensator that automatically reduces torque ripple. The compensator is implemented and torque ripple reduction is tested with industrial PM motors. The performance of the Q-learning based compensator is evaluated and compared against ILC and convention PI speed control. Therefore, the thesis will also validate the applicability of the ILC method with industrial PM machines.

Various methods are used in order to produce a comprehensive study. Finite element analysis (FEA) is used to validate that cogging torque, flux harmonics and current measurement errors truly give rise to torque ripple. These simulations are time-consuming, hence a faster lumped-element motor control simulator is used for implementing and evaluating the theoretical performance of the compensators. The performance will be evaluated by comparing torque and speed pulsations in case of using conventional PI speed control, PI with ILC and PI with Q-learning based compensator. Finally, experimental tests are performed with 6-kW and 0.72-kW PM motors. The rotor speed is measured with a resolver to be able to evaluate compensation performance.

This thesis is organized as follows. Chapter 2 reviews the important ripple components and the mathematical model underlying PM synchronous motor (PMSM). Chapter 3 outlines the existing torque ripple minimization methods and their limitations. Chapter 4 describes ILC and its utilization for torque ripple minimization. Chapter 5 discusses the Q-learning based compensation scheme. Chapter 6 presents simulation results for ILC and Q-learning based compensation. Chapter 7 presents the experimental results comparing the compensation of conventional PI-control, ILC and Q-learning methods. Chapter 8 concludes by discussing the results and suggesting future work.

2 Origin of torque ripple

In this chapter, all important factors affecting torque ripple are discussed. FEA is utilized for analyzing torque ripple and its sources. A model of a 160-kW motor is used and its parameters are listed in Table 1. The motor cross section in Fig. 1 shows the magnet placement, stator slots as well as flux density distribution of the PM motor.

Table 1: 160-kW PM motor parameters

Description	Symbol	Value	Unit
Nominal current	I_N	332.1	A
Nominal voltage	U_N	370.0	V
Nominal frequency	f_N	155.0	Hz
Nominal speed	ω_N	3100	rpm
Nominal power	P_N	160.0	kW
Nominal torque	T_N	493.0	Nm
Number of poles	N_p	6	
Number of stator slots	N_s	72	

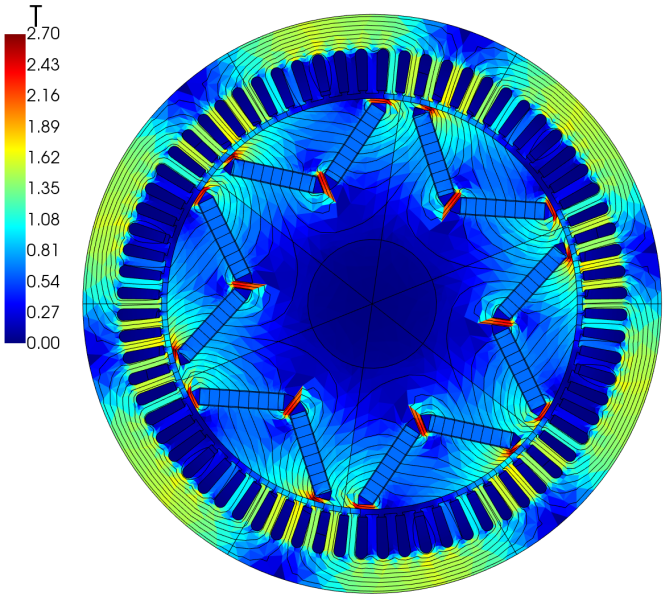


Figure 1: Flux density distribution of the 160-kW PM motor at the nominal point

2.1 PMSM model

Assuming that eddy currents and hysteresis losses are negligible, stator surface is smooth and magnetomotive force (MMF) of the stator windings is sinusoidal, the

flux linkages along the d–q axes in the rotor reference frame are given by [13]

$$\begin{aligned} \psi_d &= L_d i_d + \psi_f \\ \psi_q &= L_q i_q \end{aligned} \tag{1}$$

where i_d and i_q are the dq-axis currents, L_d and L_q are dq-axis inductances and ψ_f is the rotor flux linkage. By further assuming the PMSM to be nonsalient, the L_d and L_q can be considered equal [2]. Stator dq-axis voltages can be expressed as [13]

$$\begin{aligned} u_d &= R_s i_d + \frac{d\psi_d}{dt} - \omega_e \psi_q \\ u_q &= R_s i_q + \frac{d\psi_q}{dt} + \omega_e \psi_d \end{aligned} \tag{2}$$

where ω_e is the electrical angular speed of the rotor and R_s is the stator resistance.

Electromagnetic torque can be expressed as [1, 3, 13]

$$T_m = \frac{3}{2} p(\psi_d i_q - \psi_q i_d) \tag{3}$$

where $p = N_p/2$ is the number of pole pairs. With field-oriented control, the d-axis current is controlled to be zero in order to maximize the output torque with given current [3, 13]. Hence, only q-axis current is used for torque production and with $i_d = 0$, the torque equation reduces to

$$T_m = \frac{3}{2} p \psi_f i_q = k_t i_q \tag{4}$$

where k_t is a torque coefficient.

In order to make the model to reflect non-ideal reality more accurately, also possible disturbances should be considered. Therefore, coefficients describing cogging torque and current measurement errors are added to the electromagnetic torque equation [14]

$$T_m = k_t i_q + T_{cog} + T_{\Delta I} \tag{5}$$

where T_{cog} and $T_{\Delta I}$ are periodic torque pulsations due to cogging torque and current measurement errors, respectively.

The equation of motion is

$$\frac{d\omega_m}{dt} = -\frac{B}{J}\omega_m + \frac{T_m}{J} - \frac{T_l}{J} \tag{6}$$

where $\omega_m = \omega_e/p$ is the mechanical angular speed, T_l is the load torque, B is the viscous damping coefficient and J is the total inertia (motor and load) [2].

2.2 Relation between torque and speed pulsations

The 160-kW motor was run with a constant speed reference in a simulation environment. The FEM simulation was repeatedly stepped forward while instantaneous torque and rotor speed were numerically computed. Motion equation (6) was used in the simulation. Computation result in Fig. 2 shows undesired speed and torque fluctuations, even when the machine speed was controlled with a PI controller and the speed reference was kept constant. Thereby, disturbances must be present, since the system never stabilizes.

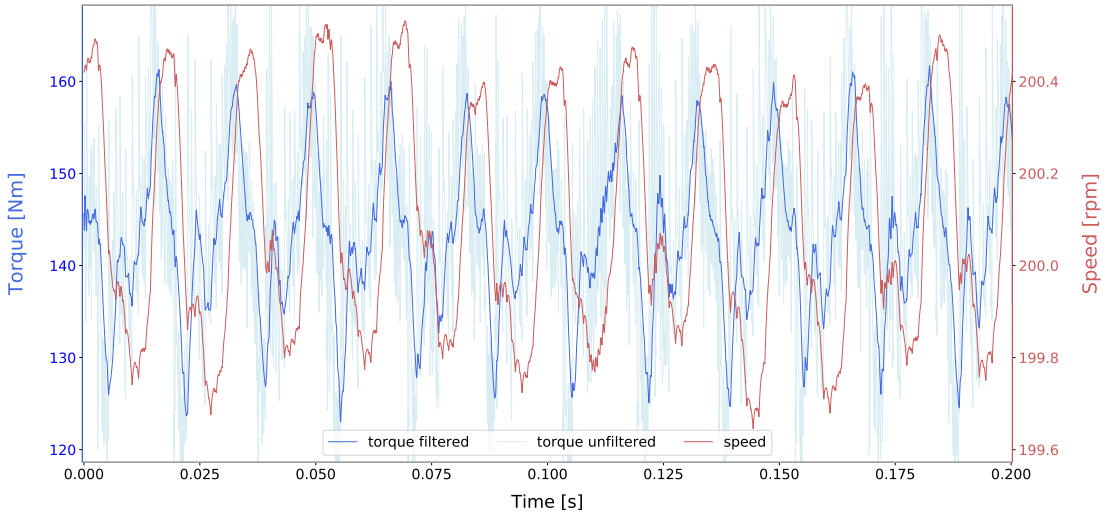


Figure 2: Torque and speed pulsations of the 160-kW motor

Equation (6) allows to derive the transfer function between mechanical angular velocity ω_m and electromagnetic torque T_m

$$\omega_m(s) = \frac{T_m(s) - T_l(s)}{Js + B} \tag{7}$$

It can be concluded that speed oscillates at the same harmonic frequencies as electromagnetic torque [1, 2]. The same can be also observed from Fig. 2 by comparing speed and torque oscillations. The relation between pulsations makes it possible to measure speed signal in order to indirectly conclude whether torque pulsations exist [2]. To minimize the speed pulsations, its source, torque pulsations, must be minimized [1, 14]. This relation makes it possible to utilize speed measuring instruments, such as resolvers and encoders in ripple minimization. When ripple minimization method is based on speed measurements, the compensator becomes more widely applicable, because encoders and resolvers are more commonly available than torque transducers.

Another FEM computation was performed at different speeds and in open-circuit condition. The result is visualized in Fig. 3. Harmonic orders are given with respect to the mechanical frequency of the rotor. It can be observed that torque amplitudes are equal regardless of the speed. Therefore, the rotation speed is irrelevant when

studying harmonics and data can be visualized in two dimensional space without losing information.

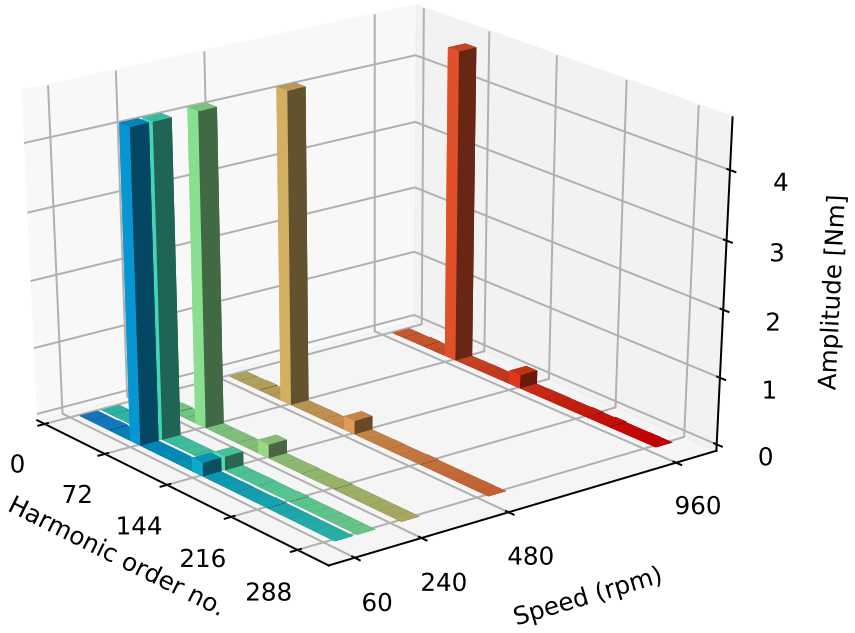


Figure 3: Cogging torque of the 160-kW motor in different speeds

2.3 Cogging torque

Cogging torque arises from interaction between magnets and stator teeth. The magnets create magnetic field around the rotor whilst slots in stator induce permeance variation in airgap. Due to permeance variation, the rotor seeks a position where the permeance of the magnetic circuit is maximized [15, 16]. This creates periodic torque on the shaft, which may generate undesired mechanical vibrations and noise. Since torque pulsations give rise to speed oscillations similarly to Fig. 2, also accurate position control gets more difficult. This explains why ripple minimization is particularly important with servomotors.

Cogging torque can be expressed as Fourier series [4, 16–22]. For a single slot, the cogging torque is given by [16, 19]

$$T_{cslot}(\theta_m) = \sum_{i=1,2,3,\ldots}^{\infty} A_i \sin(N_p i \theta_m) \tag{8}$$

where N_p is the number of poles, θ_m gives the rotor position relative to single stator slot and A_i is the amplitude of the corresponding ith harmonic. Since $\theta_e = (N_p/2)\theta_m$,

harmonics can be also considered as a function of the electrical rotor angle, θ_e [1,3]. The total cogging torque, resulting from N_s stator slots, is given as [16,19]

$$T_{cog}(\theta_m) = \sum_{i=1}^{\infty} \sum_{k=0}^{N_s-1} \cos\left(\frac{2kiN_p\pi}{N_s}\right) A_i \sin(N_p i\theta_m) \tag{9}$$

By mathematically manipulating the equation, it can be deduced that only the terms in which $2i/(N_pN_s)$ are integers, are nonzero. Therefore, the fundamental cogging torque frequency is the least common multiple of poles and stator slots, $\mathrm{lcm}(N_p, N_s)$ [16,19,23]. It could be also shown that all single slot harmonics contribute to the resultant cogging torque, and the periodicity of the cogging torque is identical to single-slot cogging torque waveform, while its amplitude is N_s times greater [19].

FEA was performed for the 160-kW PM motor. The cogging torque of the motor was computed and fast Fourier transform (FFT) was applied in order to obtain the cogging torque harmonics. The harmonics can be seen in Figs. 3 and 4. When ignoring the DC component, the first significant amplitude has the harmonic order of 72. This result is congruent with previous equations and statements as the same order can be also obtained by calculating, $\mathrm{lcm}(N_p, N_s) = 72$. The number of slots and poles are listed in Table 1. The 72nd harmonic has approximately amplitude of 4.6 Nm, which is roughly 1% of the nominal torque.

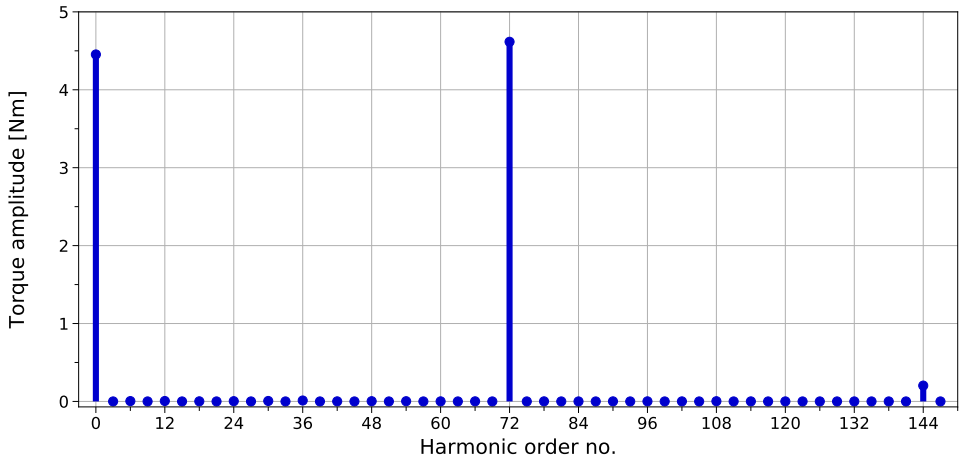


Figure 4: Cogging torque harmonics of the 160-kW motor, when considering the mechanical frequency of the rotor as the fundamental frequency

2.4 Flux harmonics

Flux density distribution in the airgap and the MMF of the stator windings are seldom ideally sinusoidal. Due to nonsinusoidality, the phase flux linkages contain harmonics of the order of $5, 7, 11, \ldots$ [24]. In rotor reference frame, the corresponding spatial harmonics appear as multiples of sixth-order harmonic components, and can

be expressed as [1, 2, 14]

$$\psi_f = \psi_{d0} + \sum_{j=1}^{\infty} \psi_{d6j} \cos(j6\theta_e) \tag{10}$$

where ψ_{d0} is the DC term and ψ_{d6j} are the harmonic terms of the d-axis flux linkage, while θ_e is the electrical angle of the rotor. As before, the flux harmonics could be also considered as a function of the mechanical rotor angle due to relation between mechanical and electrical rotor angles.

Combining (3) and (10) yields

$$T_m = T_0 + \sum_{j=1}^{\infty} T_{6j} \cos(j6\theta_e) \tag{11}$$

where T_0 and T_{6j} are the DC component and harmonic torque amplitudes, respectively. Equation (11) indicates that produced torque pulsations are periodic in nature. FEM computation done at nominal point with constant speed confirms the periodicity of pulsations and it also shows the sixth harmonic multiples to be especially prominent in Fig. 5.

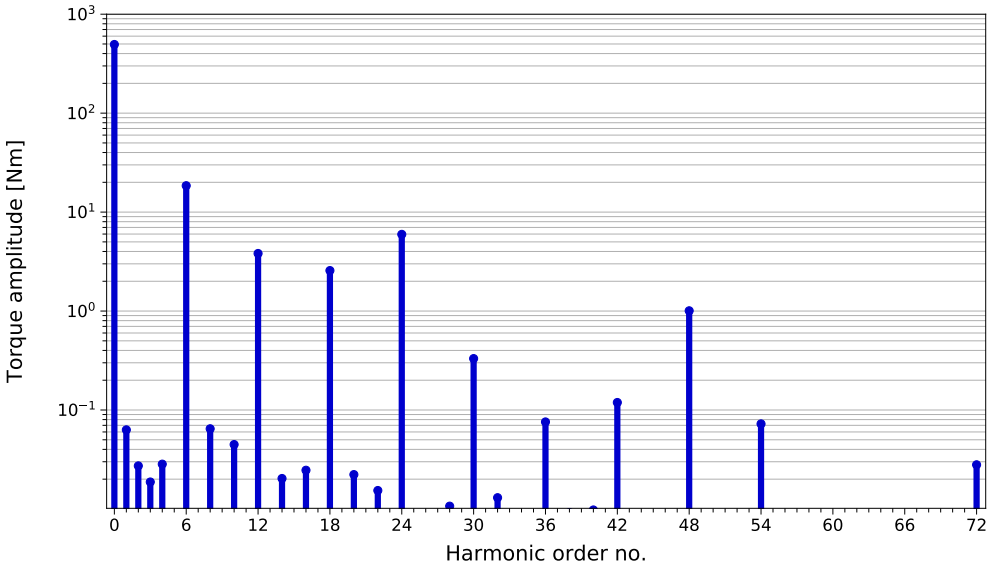


Figure 5: Torque harmonics of the 160-kW motor, when considering the supply frequency (155 Hz) as the fundamental frequency

It is good to note that Figs. 1 and 5 do not only include the effects of the flux harmonics but also cogging torque, because of overlapping frequencies. When rotor flux is calculated in different rotor angles in time stepping simulation, also cogging torque is inevitably captured. Cogging torque could be subtracted from the result, though this serves no purpose, as the objective is to minimize both disturbances simultaneously.

2.5 Current measurement errors

Current measurement errors contribute to the torque ripple [1–3,14,25,26]. The errors can be considered to be the offset and scaling errors [3,14,25,26]. The stator currents are measured and transduced into voltage signals by current sensors. The voltages are then transformed into digital form by analog-to-digital (A/D) converters [2,25], so that the control can utilize the measured values. The current measurement errors emerge during this process.

If there is any unbalance between the phase voltages supplied to the motor, or, if the analog devices include inherent offset, then DC offset current will be introduced. When measuring two phase currents and assuming ideal current control, the current offset related torque error can be described by [25]

$$\Delta T_{m1} = k_t \frac{2}{\sqrt{3}} \sqrt{\Delta i_a^2 + \Delta i_a \Delta i_b + \Delta i_b^2} \cos(\theta_e + \alpha) \tag{12}$$

$$\alpha = \tan^{-1}\left(\frac{\sqrt{3}\Delta i_a}{\Delta i_a + 2\Delta i_b}\right)$$

where Δi_a and Δi_b are a and b phase offsets and α represents angular displacement between the phases. The torque error has relation to θ_e, hence DC offsets give rise to torque oscillations at the fundamental frequency [1,6,14,25].

A FEM computation with intentionally introduced current offset error was performed. The error was added to single phase current, which was used by control algorithms. The simulation result is shown in Fig. 6. The introduced error roughly corresponds to 10 A measurement error, which is approximately 3% of the motor nominal current. With this error, it can be seen that the first harmonic has the greatest amplitude when ignoring the DC component.

The measured phase currents must be scaled to fit in the input range of the A/D converter. The A/D converter output is then re-scaled by the current controller, so that real current values can be obtained. Scaling errors are inevitable [25]. The torque error coming from the scaling errors can be described as [2,25]

$$\Delta T_{m2} = \left(\frac{K_a - K_b}{K_a K_b}\right) \frac{k_t I}{\sqrt{3}} \left[\cos\left(2\theta_e + \frac{\pi}{3}\right) + \frac{1}{2}\right] \tag{13}$$

where K_a and K_b are controller scaling factors for a and b phases and I is the phase current. Since θ_e is multiplied by two, the scaling errors give rise to torque oscillations at twice the fundamental frequency [1,2,6,25].

It can be concluded that torque pulsations resulting from the current measurement errors are periodic with respect to the rotor angle. Therefore, cogging torque, flux harmonics and current measurement errors induce 1st, 2nd, 6th, 12th, etc. torque harmonics and the objective is to suppress these periodic disturbances.

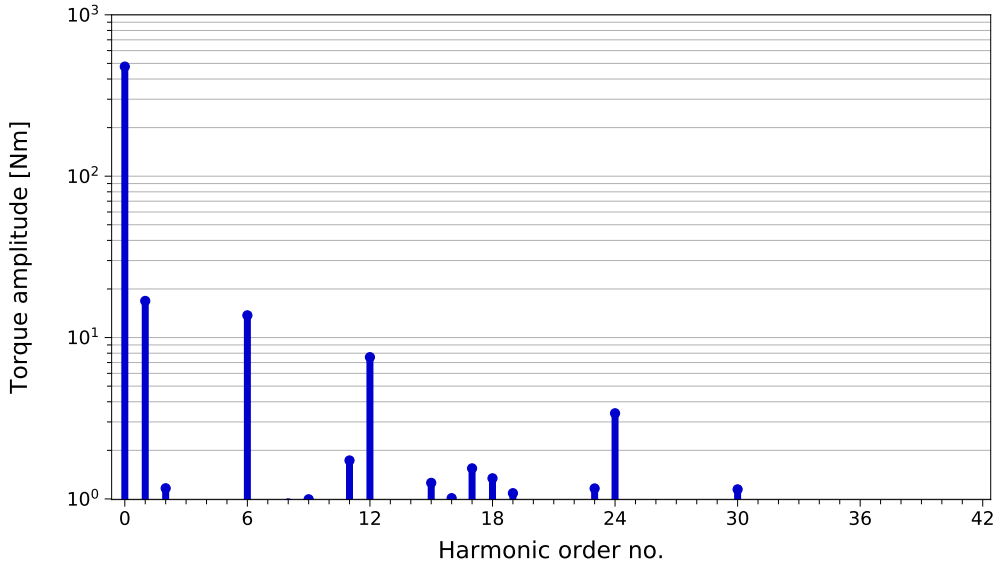


Figure 6: Torque harmonics of the 160-kW motor under current measurement error, when considering the supply frequency (155 Hz) as the fundamental frequency

3 An overview of reduction methods

The torque ripple minimization has been studied extensively for many years and numerous solutions have been already proposed. This chapter discusses the limitations and strengths of the already existing solutions. The focus is on software methods, although the machine design based methods are briefly outlined to provide an extensive overview. The software methods can be divided into three categories: algorithms that exploit foreknowledge, observer based methods and iterative methods. Fig. 7 summarizes the various torque ripple minimization methods presented in this chapter.

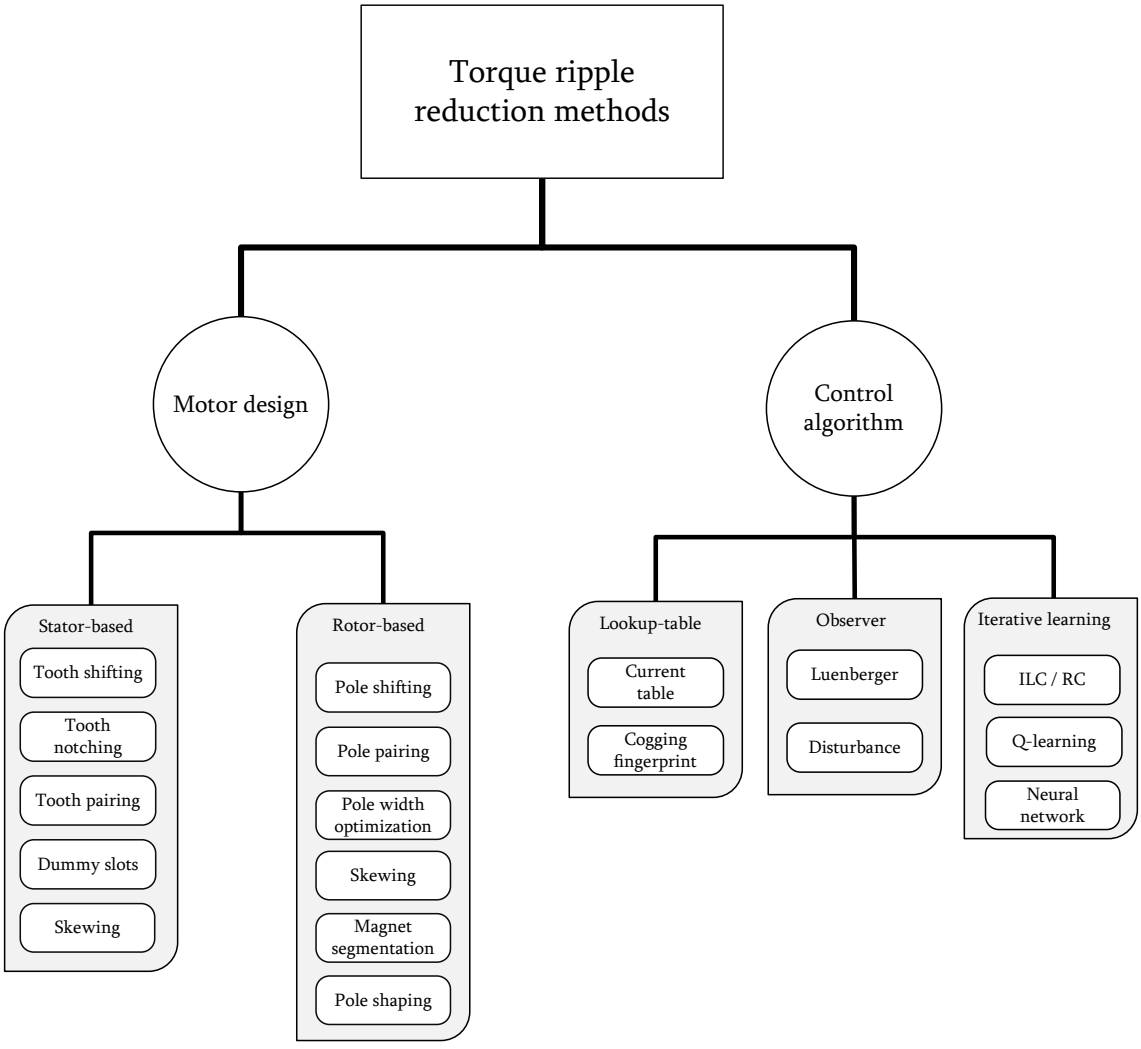


Figure 7: Various torque pulsation reduction techniques

3.1 Machine design optimization

Torque pulsation due to cogging torque emerge even if the machine is not excited by external voltage source. Therefore, it is logical to try reduce torque pulsations

by addressing the sources and then improving the machine geometry appropriately. Design optimization has been found an effective way to reduce ripples. This explains the great number of different methods that can be further categorized to stator and rotor based methods [16]. Rotor geometry optimization is often preferred over stator geometry optimization, because the rotor structure is simpler and thus easier to modify in PM machines [15, 16].

Cogging torque should be minimized in order to reduce torque ripple. Minimization techniques include for example, stator and rotor skewing [16, 17, 23, 27, 28], magnet arc width optimization [17, 28, 29], magnet pole shaping [18, 27], magnet segmentation [4, 17, 20], stator tooth notching [17, 27], using odd number of slots [30] and pole and teeth pairing [28]. All these methods aim to improve the geometry of the machine that leads to reduced cogging torque. However, optimization is often a trade-off with other properties of the machine. Therefore, applying all the design improvements simultaneously is not necessarily practical or even possible [16, 17].

Although, the geometry optimization is an effective way to reduce torque ripple, it is rarely an applicable solution when the machine has been manufactured already. Individual parts or the entire machine would have to be re-manufactured in order to enhance the design, which can be costly and impractical to realize.

3.2 Data exploiting algorithms

It is possible to reduce torque ripple by means of active control. The control algorithms can be made to minimize the ripple without any additional hardware investments, because compensation can be implemented on already existing control platform. In addition, torque ripple reduction can be achieved without knowing its sources. This is especially useful in cases where manufacturing imperfections, such as magnet misalignment, causes the ripple. Identifying these ripple sources can be difficult [31].

The early compensation schemes use precomputed stator current excitations to cancel out torque harmonics [2]. Appropriate excitations can be calculated using Fourier transform [5, 32]. The disturbance cancelling harmonic values can be stored into a lookup table which is saved to the memory. The table creation requires accurate foreknowledge of the particular motor, which may be problematic. Without foreknowledge, the method is unusable, as every individual motor can have different torque harmonics. Furthermore, numerical errors in the lookup table can lead to increased ripples due to open-loop feedforward control [1]. Moreover, creating such lookup tables manually for mass produced machines can be troublesome.

Another similar scheme is to automatically identify the cogging torque and fill the compensation lookup table. Cogging torque identification can be done by measuring rotor angle and torque simultaneously. Rotor angle can be measured with a position sensor and torque with a torque transducer. An alternative method to indirectly estimate instantaneous torque is to measure q-axis current, which produces the torque (4). This automatic identification method is less error prone and it is better suited for mass produced machines [26, 33, 34]. This idea can be generalized to other periodic disturbance sources, which allows implementation of an effective compensator.

3.3 Observer based identification and compensation

Evidently, one of the main challenges of automatic methods is to identify the disturbances successfully. Ripple compensation is a relatively simple task, after the disturbances have been accurately identified. Identification can be done by utilizing an observer that tracks the system states. A well suited observer for this problem is a disturbance observer (DOB), which has been already utilized for ripple minimization problem in [31]. The DOB was used to improve the performance of already existing ripple reduction method.

Purely observer based compensation has been studied in [35] and [36]. The studied observers were, Luenberger full-order observer, adaptive observer, DOB, periodic disturbance observer (PDOB) and adaptive PDOB. It was found that compensation can be done by utilizing observers. However, this approach appears to be the most complex of the overviewed methods. Nominal plant model must be known for designing DOB or its variants. System knowledge is also required for designing an effective filter that is used with the DOB [36]. Due to heavy system knowledge utilization, the method is impractical when numerous motor drive systems require reduction of torque pulsations. Therefore, it will be concluded that observer based compensation may work, though there are likely better approaches for this particular problem.

3.4 Iterative control algorithms

In repetitive control (RC) the control input is calculated using the information of the error signal in the preceding periods. Therefore, the RC can be seen as a learning control, which makes it useful for periodic disturbance inputs [37]. The design of the conventional RC requires foreknowledge of the fundamental frequency of the periodic torque ripple and the use of this knowledge makes the RC effective only in constant speeds [38]. Due to this property, the conventional RC is not very practical with variable speed drives. RC can be implemented to use rotor angle instead of the frequency, which allows ripple reduction on variable speeds [38]. The angle based RC can be combined with a disturbance observer which yields an angle-based repetitive observer (ARO) compensator. The ARO can suppress torque ripples on a wide frequency range, it is modular and requires tuning of only two parameters [31].

Another scheme taking advantage of periodicity of pulsations is iterative learning control (ILC). ILC minimizes the ripple iteratively in similar manner to RC. ILC computes an error between reference and actual value and then uses this result on subsequent iterations to improve the control signal [1, 2, 14]. Conventional time based ILC assumes the rotation period to stay constant, which allows ripple reduction only in constant speeds. ILC can be implemented to use rotor angle instead of relying time, which makes it possible to compensate for disturbances on varying speeds [3]. Modular structure allows the ILC compensator to be applied easily, similarly to Fig. 8, to an existing control system.

Comparison between RC and ILC is realized in [39]. The conclusion is that the methods could be often considered the same. From now on, ILC will be favoured over

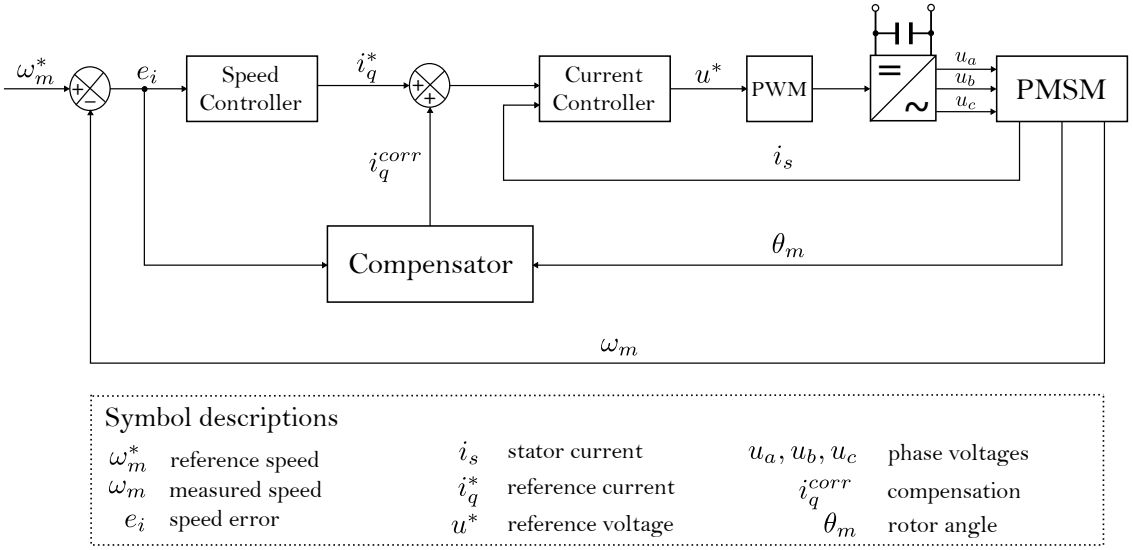


Figure 8: Iterative speed based compensation scheme

RC, because ILC has more strict definition [40] and it is recognized more widely [39] as compared to RC. Since angle-based ILC solves the disturbance compensation effectively, the thesis will investigate ILC more carefully in the following chapters.

Neural networks can be iteratively trained to compensate for periodic disturbances [4]. The compensation concept is similar to the ILC, but instead of continuously calculating the best compensation value, the network is trained once, thus allowing it to immediately output the best value after training. Implementing a neural network to an old hardware, which must be capable of running real-time processes, can be challenging. Besides limited processing power and memory resources, it can also require a lot of effort to build the network, since modern comprehensive code libraries relying on modern standards, cannot be utilized with old devices.

The training could be done using different hardware than is used in drive. With modern embedded devices capable of connecting to internet, the training could happen on remote servers, which can provide a solution to problems emerging from limited computational resources. However, old embedded devices can rarely connect to internet. Alternative training method could be to use a simulator and then move the trained model to the embedded device by using flash drive for example. However, in such scenario, the model performance is heavily dependant on the simulator accuracy. It can be concluded that neural network based compensation may work with modern products, since such compensators have been already built [4], yet the approach can be troublesome with hardware that does not support modern standards and neither is connected to Internet.

4 Iterative learning control

Iterative learning control is an effective scheme used to improve repeatedly executing system performance [1]. Since torque pulsations were found to be periodic, ILC can be used to learn from previous iterations, thus making it possible to identify torque pulsations automatically. ILC based compensator can be used to reduce the ripple with weak process knowledge [41], which makes the method well suited when numerous systems require compensation.

4.1 ILC prerequisites

For ILC to be effective, the following postulates must hold [3, 40, 42]

1) Every iteration ends in a fixed time duration.

2) System invariance is ensured throughout repetition.

3) The system output is measured in a deterministic way.

In addition to previous postulates, a few system related considerations should be accounted. The measured speed (or torque) must not be filtered too heavily, as this introduces compensation delay. Too great delay can annihilate applied compensation. Since ILC learns from previous iterations, memory storage is required. The memory is accessed frequently, which advocates use of random access memory (RAM). The required memory size depends on compensation accuracy that is being pursued. In addition to free memory space, also processing power is needed for computing compensation values.

4.2 Ripple minimization concept

The torque pulsations can be negated by generating an opposing torque. This can be done by computing a correction term, which is injected to a control reference. Ideally, the term has the same magnitude but opposite sign as pulsating torque at some time instant. This creates destructive interference which cancels out pulsations.

Disturbances were found to be periodic, which allows them to be identified iteratively. Identification can be done comparing the measured output value with the reference value. In practice, an error can be calculated by subtracting the measured speed from the speed reference and then the task of ILC is to minimize the error by injecting an appropriate correction term. The correction value can be injected either into torque reference [14] or to q-axis current reference [1, 2] similarly to Fig. 8. ILC tracks complete periods, which allows it to iteratively find appropriate correction terms for every step.

The minimization concept is easy to understand from examples. A simulation result in Fig. 9 shows compensation of arbitrary pulsations. The green signal in plots represents torque produced by the ILC compensator. The blue signal is the pulsating torque, which represents only the sixth harmonic for the sake of clarity. Pulsations are summed to ideal electromagnetic torque, thus leading oscillations of

the red signal. Since ILC is disabled in the upper plot, no compensating torque is produced and torque pulsations are directly visible from the actual torque. In the lower plot, ILC is enabled and compensating torque is produced. Now actual torque has smaller amplitude than pulsating torque, which allows to deduce that the compensator reduces torque pulsations. It can be observed that the compensating signal has roughly the same amplitude as actual torque in the upper plot while the phases differ by π. As a result, majority of torque pulsations cancel out. Part of pulsations remain due to introduction of robustness coefficient [2, 14, 43], which will be explained later.

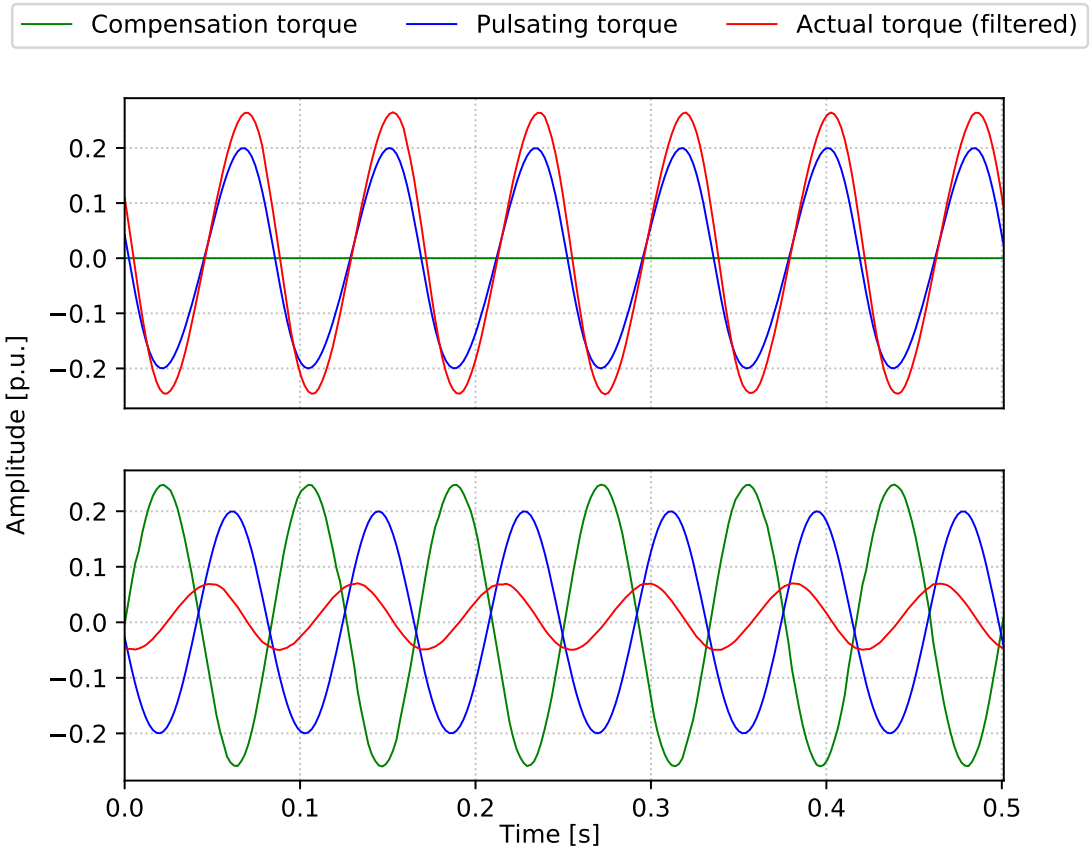


Figure 9: The upper figure shows arbitrary torque pulsations, that are reduced in the lower figure, due to simulated destructive interference caused by the ILC compensator

Figure 10 illustrates retrieval and improvement process of compensation values. Previous iteration execution is shown with dashed lines, whereas solid lines correspond to current iteration. Compensation torque has been too small on the previous iteration, thus letting the speed to oscillate and induce error between the reference and the measured value. ILC has detected the speed error and it is improving the compensation waveform by increasing amplitude of the compensation pattern. The old compensation values are getting replaced with new values. Due to improved compensation, the speed oscillations have gotten smaller on the current iteration. The ILC compensator is able to retrieve compensation values consistently from the

memory by utilizing measured rotor angle. The compensation pattern consists of numerous discrete values that are saved into the memory. It will be discussed how the compensation values can be computed mathematically in the next subsection.

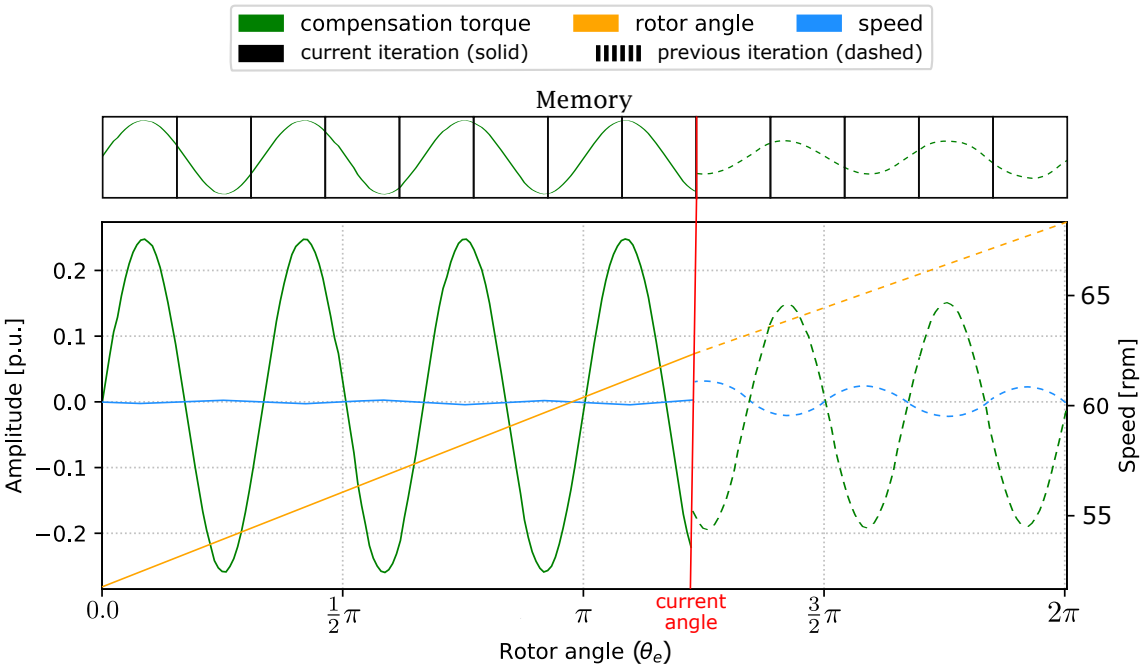


Figure 10: Iterative learning concept of the ILC compensator

4.3 Iterative learning control algorithm

ILC is not a single algorithm, but instead, a methodology consisting of many update laws. The vast majority of ILC literature focuses on either proportional type (P-type) or derivative type (D-type) update laws [40]. Since unnecessary noise build-up in this problem can be avoided by selecting the P-type update law over the D-type [1,2], the focus shall be on a P-type learning controller. In addition, various P-type controllers have been already applied successfully for the speed ripple minimization problem [1–3,14].

The learning law is given by [2,14]

$$u_i(\theta) = (1-\alpha)u_{i-1}(\theta) + \Phi e_{i-1}(\theta) + \Gamma e_i(\theta) \tag{14}$$

where θ is the rotor angle, which can be either mechanical or electrical as there is relation between the two. Φ and Γ are gains used to tune controller, e is the speed error and u is control action, which in this case corresponds the current reference correction signal $i_{q,i}^{corr}$. The iteration index $i = 1, 2, 3, ...$ corresponds either mechanical or electrical period depending on whether mechanical or electrical rotor angle is used. The first measurable error is e_i and e_{i-1} can be obtained from memory after one period. Values in the memory can be initialized to zero [14]. Figure 11 visualizes the structure of the ILC algorithm.

Forgetting factor α can be used to make the P-type algorithm more robust against noise, initialization error and system dynamics fluctuation [2, 43]. Therefore, the parameter α is included into learning law, as this is expected to improve the performance of the controller with real systems. For the same reason, also present error signal $e_i(\theta)$ is included, in addition to typical previous iteration error $e_{i-1}(\theta)$, as this should stabilize the learning in presence of perturbations [14].

The memory usage can be adjusted as long as the memories hold values at least for one full period. Since period length depends on running speed, this leads to the situation where different memory sizes would be needed for different speeds, if memories were accessed with time directly. For this reason, the update law is written with respect to rotor angle, as this allows to have fixed memory size. By quantizing one period into N steps, the values can be stored into memory holding N values. Thus, step size selection has direct effect to memory consumption as well as to compensation performance. The performance suffers if steps are too large, since the algorithm cannot track error accurately.

The fixed size memory is accessed by utilizing rotor angle. Rotor angle quantized to k indices is given by, i.e., $\theta_k = 2\pi k/N$ [3]. The measured rotor angle can be associated to an index by using rounding rule, k =round$(N\theta/2\pi)$ [3]. Use of the index k allows reading and writing to desired memory locations. The same element can be consistently retrieved from the memory every period, hence allowing improvement to happen iteratively. It should be noted that quantization makes it possible to leap over values, especially in case of high speed and many steps. The skipped values should be linearly interpolated [3].

The gain values Φ and Γ should be selected carefully to allow converge. Convergence condition for the controller can be derived starting from the torque equation (5) and the final condition is given as [14]

$$|\rho| = \left| \frac{1 - \alpha - k_t \Phi}{1 + k_t \Gamma} \right| < 1 \tag{15}$$

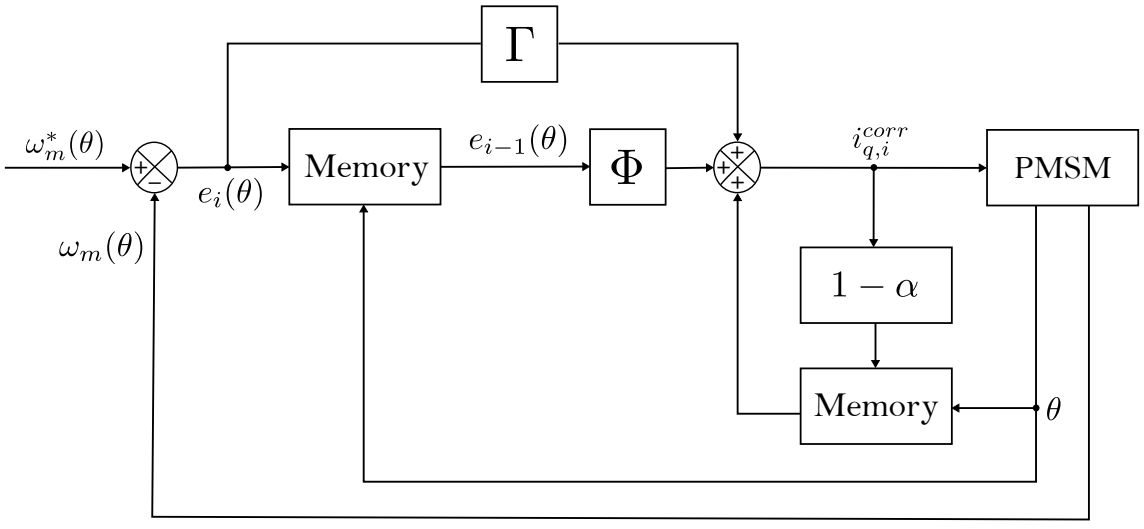


Figure 11: Block diagram of rotor angle-based ILC

Since $k_t > 0$, the condition is satisfied with [14]

$$0 < \alpha + k_t \Phi < 2 \tag{16}$$

If the torque constant k_t is unknown, its upper bound must be estimated in order to set a value for Φ. Prudent upper bound estimate guarantees convergence for a wider range of k_t, but also leads to slower convergence rate [2, 14]. Finding a good value for Φ is important as it considerably affects the compensation performance. The gain Γ influences compensator response between iterations [2]. Too large Γ will cause oscillatory response, hence, a prudent estimate of $\Gamma \leq \Phi$ may suffice for producing satisfying results [14]. However, better compensation performance can be achieved by carefully tuning the controller, instead of using the prudent tuning guidelines.

It can be speculated whether the use of mechanical rotor angle in ILC could provide better compensation results than electrical rotor angle. Mechanical rotor angle allows ILC to observe all possible manufacturing imperfections occurring on one full rotation. Electrical period is only a fraction of mechanical, so asymmetry between poles cannot be fully accounted. However, a shorter period allows to have smaller memory sizes and theoretically ILC should learn faster, because iterations take less time. This gives a good reason to favor electrical angle over the mechanical. During experimental tests, it was observed that use of electrical angle also allows to set slightly higher ILC gains, before instability is encountered. As a direct consequence, the implementation using electrical rotor angle performed slightly better than identical mechanical rotor angle based implementation.

5 Q-learning

Q-learning is one of the earliest reinforcement learning (RL) algorithms. It was discovered in 1989 [44] and the algorithm can be considered to be a significant milestone for RL research. The Q-learning algorithm is computationally light and simple to implement, which makes it perfect for embedded real-time systems. The implementation requires only creation of a structure resembling a lookup table, which can be implemented even on obsolete hardware.

5.1 Modelling the system

The Q-learning algorithm requires the system to be modelled. Markov decision processes (MDP) are widely used for modelling various systems mathematically [45,46]. Control problems can be idealized by using MDPs, which makes it possible to make precise theoretical statements about the system [46]. MDPs formalize sequential decision making [46], that is done in RL. In MDP, the decision maker is called an agent and everything outside the agent is called an environment [46]. The agent continually interacts with the environment by making actions and the environment responds to actions by presenting new situations and outcomes [46]. The outcomes are numerical values that are formally called rewards. The agent seeks to maximize the rewards by always taking the best actions in different situations. The situations can be formalized using states which consist of system parameters. Thereby, the agent-environment interaction can be mathematically presented by declaring a set of states $\mathcal{S}$, actions $\mathcal{A}$ and rewards $\mathcal{R}$. The agent must be always in some state, $s \in \mathcal{S}$ where it can do an action $a \in \mathcal{A}$, which produces reward $r \in \mathcal{R}$.

The states should be defined to represent minimal amount of necessary information about the system. In the case of an electric motor and torque pulsations, the state should at least contain information about the rotor angle. This is because the torque pulsations depend on the rotor angle and the compensation must be done with respect to occurring pulsation. It can be also a good idea to include information about system load to the state. This is because the pulsation magnitudes depend on currents (5) that vary under different loads. However, this is not implemented in order to save memory. The electrical rotor angle θ_e is quantized to 100 possible values, thus allowing the system to be in 100 different states.

In order to compensate for pulsations, an injection causing destructive interference with the pulsations must be produced. The event of making appropriate injection can be expressed using actions. The agent selects a value and sums it to the q-axis current reference i_q^* similarly to ILC and Fig. 8. In Q-learning, the actions, states and state-action values are held in a table, which is stored into the memory [46]. Therefore, actions must be discretized and injection accuracy depends on discretization step size. Step size selection is a trade-off between memory usage, accuracy and learning speed. The effect on learning speed can be explained with the number of different states. With high step size and lower number of actions, the agent needs to interact less with the environment in order to conclude the "goodness" of each action.

5.2 Automatic action generation

Torque pulsations are periodic and continuous, therefore all pulsation values must lie between the peaks. Assuming symmetry of pulsations, then a set of actions can be generated only by knowing the pulsation maximum T_{max}. Specifying number of desired actions N_a, a linearly spaced action set $\mathcal{A} = \{-T_{max}, ..., T_{max}\}$ containing compensation values can be automatically generated. The pulsation maximum is rarely known in practice, hence it must be often estimated. It can be reasoned that it is better to overestimate T_{max} than provide too small value, because potential performance degrade is lower. With too large estimate, it is still possible to include all required values into action set at the cost of worse resolution, whereas too small estimate results into inadequate action set. If learning time and memory space were not concerns, then both parameters could be set arbitrarily high.

It can be observed from FEM simulation result in Fig. 5 that torque harmonics are only a couple of percents of the nominal torque. This will provide an rough estimate of T_{max} values that are appropriate. In case of the 160-kW motor, $T_{max} = 0.02$ p.u. could be used directly, since the pulsation maximum is known from FEA. With the motors used in experiments, appropriate T_{max} value had to be searched by experimenting, yet couple of percents seemed to often work quite well. The value of T_{max} could be potentially fixed to some percentage of nominal torque. Furthermore, to guarantee inclusion of zero action due to set symmetricity, the N_a value should be odd. This allows the compensator to be in inactive state. An example of generated actions can be seen in Fig. 12.

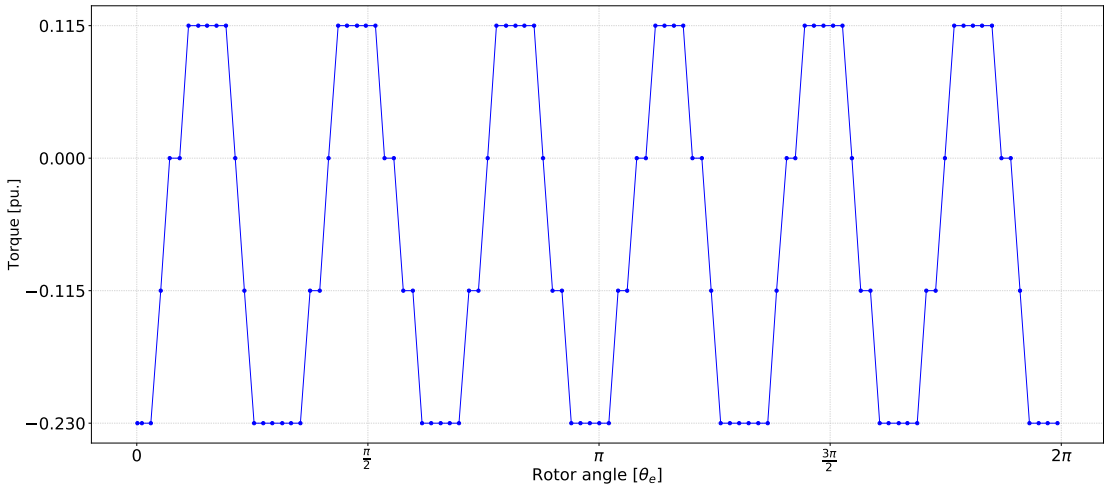


Figure 12: Learned compensation pattern with respect to the rotor angle

5.3 Q-learning algorithm

Q-learning provides a capability for the agent to learn to act optimally in MDP [44]. The agent tries actions and evaluates the consequences in terms of received rewards and estimates of the next state value [44]. By repeatedly trying actions in all states,

the agent eventually learns which actions are the best in overall [44]. The task of the agent is to determine an optimal policy that maximizes total discounted expected reward [44]. The discounted reward is calculated by repeatedly down-weighting recent rewards, by a factor of $\gamma \in [0, 1]$ [46]

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

where t is time step and T is total number of steps and for continuous tasks, $T = \infty$. The number of down-weightings depends on the amount of steps taken and therefore the discounting makes historical rewards less valuable than rewards received recently. The discounting factor γ makes it possible to adjust the discounting rate, which allows to control behaviour of the agent.

During learning, the agent observes the state s_t, selects and performs an action a, observes subsequent state s_{t+1}, receives a reward r and adjusts its Q-values $Q(s_{t+1}, a_{t+1})$ using a learning factor $\alpha \in [0, 1]$. The update can be written as [44]

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{17}$$

where $\max_a Q(s_{t+1}, a)$ corresponds action selection policy. In this case the agent selects the best action it perceives to be able to perform. The α is used to adjust how much weight single sequence gets in update, thus representing update step size. At early stages of learning, the agent policy is far from optimal, but the agent learns iteratively to pick better actions. The learning process can be stopped at any point and the Q-values can be saved for later use. After training, it is adequate to pick the best actions based on the learned and stored Q-values, $Q(s_t, a_t)$.

During the learning phase, the agent must select and test actions in order to learn. The simplest action selection rule is to pick an action producing the highest estimated value, $\max_a Q(s_{t+1}, a)$. Such greedy action selection always exploits the current knowledge, which is problematic in case where better actions exist, but the agent is not aware of these. Better learning performance can be achieved by behaving greedily most of the time, but with small probability ϵ, the agent may randomly test a new action. Action testing is called exploration. The methods allowing exploration with probability ϵ are called ϵ-greedy methods. In noisy environments, these are better than plain greedy methods, as it requires more exploration to find the optimal actions due to noisier rewards. The measured speed signal with electric motors is often quite noisy, which advocates use of ϵ-greedy action selection policy. [46]

The probability ϵ is used to set balance between exploration and exploiting. The action selection policy can be described with [46]

$$a \leftarrow \begin{cases} \max_a Q(s_{t+1}, a), & \text{with probability } 1 - \epsilon. \\ \text{a random action}, & \text{with probability } \epsilon. \end{cases} \tag{18}$$

where ϵ can be a constant value or it can be also gradually decreased. By gradually decreasing ϵ, the agent progressively relies more and more on learnt information. The probability ϵ is allowed to decrease to some small positive number, i.e., 0.01,

which ensures that agent does not completely stop exploring during training. The decay of ϵ can be implemented to happen with respect to training iterations

$$\epsilon \leftarrow \begin{cases} k_\epsilon \,/\, (k_\epsilon + i), & \text{if } \epsilon > 0.01. \\ 0.01, & \text{otherwise.} \end{cases} \tag{19}$$

where the k_ϵ is some constant integer and i is the iteration number. From (18) and (19) it can be seen that high k_ϵ value encourages the agent to explore more than with low k_ϵ value. By gradually decreasing the ϵ, the learning may happen faster than with a constant value.

5.4 Torque based learning scheme

It has been discussed how the torque ripple problem can be formalized and how the agent learns through rewards, but it has been left unclear how the "goodness" of actions can be evaluated and what should be given as a reward. This ultimately depends on the objective and the system. In this study, the system is motor drive and the objective is to minimize torque pulsations. To meet the goal, the rewarding process should connect actions and torque pulsations. This connection allows the agent to perceive the influence of its actions, which makes it possible for the agent to adjust its behaviour in order to maximize the expected reward. A reward function can be used to generate a reward at each time instant.

Assuming that produced torque can be measured, the reward function can be very simple. The reward can be given based on how closely the agent is able to follow the desired torque value, $-|T^* - T_m|$. The expression is multiplied with -1 to make the agent to try to minimize the torque error. In order to maximize the reward and to minimize the error, the agent would need to find a way to compensate disturbances by using predefined actions.

The simple torque based learning scheme was tested in the simulator. The agent was trained to compensate for the sixth harmonic with only five different actions. The learning outcome can be seen in Fig. 12. The rewards obtained by the agent during training were collected and the reward values are visualized in Fig. 13. The reward plot shows that the agent clearly learns to reduce torque pulsations, since the periodical rewards reflecting torque error are getting better. It can be observed that learning gets continually slower as it gets constantly harder for the agent to find better ways to reduce pulsations. Majority of the improvement happens in the first few hundred iterations. It should be also noted that the agent is never able to fully compensate for pulsations with only five actions, hence the reward average will never reach zero. Another important observation from Fig. 13 is the amount of variance in rewards. Not all actions are successful which results to bad rewards. In this particular problem, one bad action can lead to sequence of bad rewards, since recovering takes time. Rewards can be seen to dip even after prolonged training. This implies that the most recent Q-values are not necessarily the best. Therefore, instead of saving the most recent Q-values, the best values should be selected for example using the reward average.

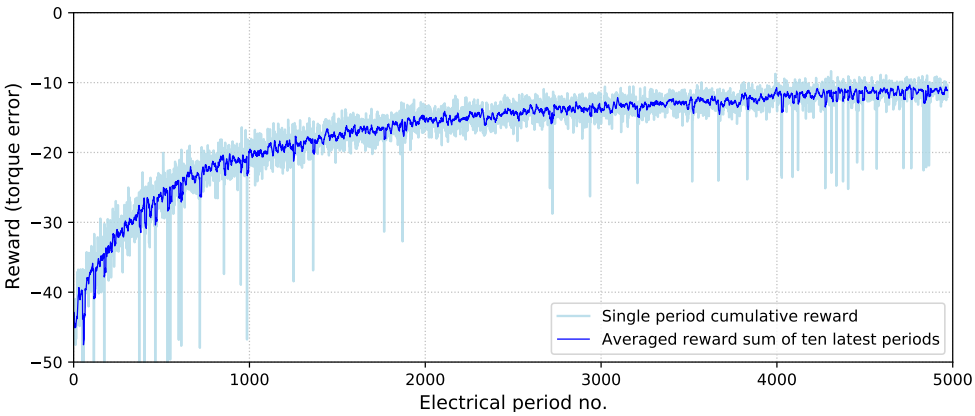


Figure 13: Rewards obtained during simulated training of the agent

The torque based learning scheme is unfortunately highly impractical, since torque measurement or suitable estimate is rarely available. Creating an accurate torque estimator is a complex task and therefore this option should be avoided. It is also good to be aware that already existing torque estimation scheme cannot be utilized. The agent produces torque by injecting to q-axis current reference i_q^* which corrupts the torque signal for learning purposes. If the agent sees its own injections directly from the estimated torque signal, then often the best method to minimize the pulsations is to do nothing.

5.5 Speed based learning scheme

The learning can be done by utilizing the speed signal. A speed based compensation scheme is more practical compared to the torque based counterpart, since speed sensors are more commonly available by default than torque measuring instruments. Furthermore, by inspecting the ILC compensator development trend in [14] and [2], it appears that the speed signal based compensator was found more successful.

The motor speed depends on torque according to (7). By inspecting computed speed and torque in Fig. 2, it can be observed that there is a phase shift between the two signals. This is exceedingly problematic when considering speed based rewarding. If the effects of torque alternation are not immediately visible, then the agent cannot know what consequences its actions had. In the worst case, the agent tries to apply brief torque impulses that are invisible for it, but stress the system in reality. Therefore, the prior rewarding scheme cannot be used directly with speed based learning. The reward function must be modified.

The occurring delay between torque and speed pulsations (7), must be accounted when giving reward. With neural networks, such problems requiring memory could be solved by using recurrent networks [47]. These feed the output signal back into the network, thus allowing the system to memorize events. This would solve the delay related rewarding problem. Unfortunately, these methods cannot be used directly with the tabular representation that the Q-learning uses. Nonetheless, the idea of

saving and reusing the data on subsequent iterations can be utilized. The previous observations can be stored into memory and these can be used in determining reward with the current observations.

It was found best approach to use only the previous and current observations to give reward. Greater amount of rewards attenuate the weight of the previous and current observations, which are the most important ones. The reward function used in the experiments is given by

$$r_t = -(|\omega^*_{m,t} - \omega_{m,t}| + \lambda \cdot |\omega_{m,t} - \omega_{m,t-1}|) \tag{20}$$

where $\omega^*_{m,t}$ is the speed reference at time instant t and λ is a weighting factor. The reward function encourages the agent to select such actions that allow following of the speed reference and the speed should change as little as possible between steps. It was found that indirect learning from the speed signal takes more time than directly using torque values for giving reward.

The agent was found to behave differently depending on how the reward terms are weighted with parameter λ. Simulation results in Fig. 14 show effects of different λ weights. It can be observed that speed pulsations become smaller when λ is increased up to value 32. After going past 32, the pulsations appear to grow again, possibly because the agent gets punished so little from not staying in the reference.

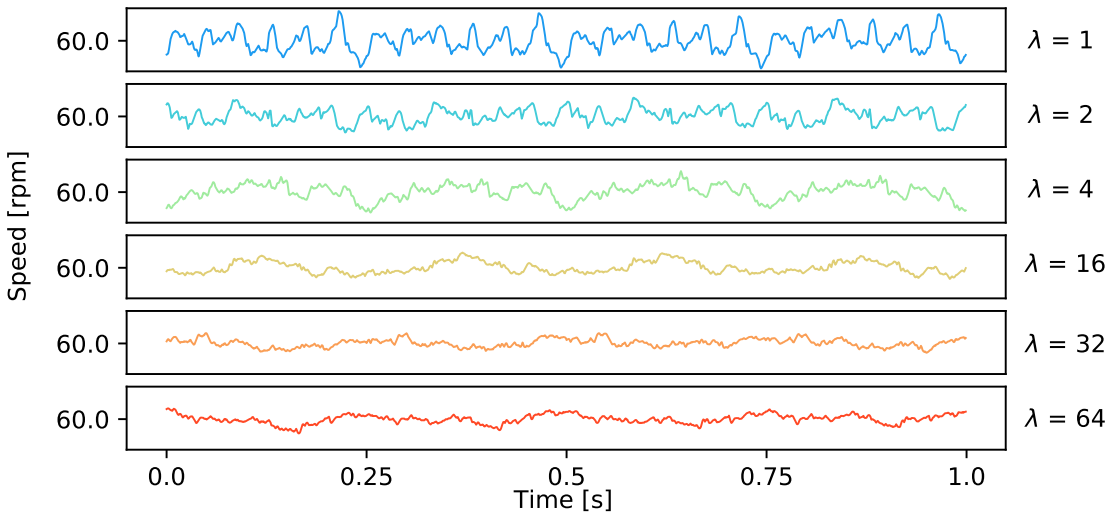


Figure 14: Simulated effects of different λ weights

5.6 Hyperparameters

Hyperparameters are parameters that are set before training and have an effect on the learning rate and outcome. Such parameters are the discount factor γ, learning rate α, weighing factor λ and coefficient k_ϵ used to set epsilon decay rate. These parameters must be set only once for a particular system. The same hyperparameters

are likely to work with variety of different PM motors as long as the whole system stays similar. In [11] the same hyperparameters were successfully used to train the agent to play various video games. In simulations, the same hyperparameters were successfully used to train the compensator to work with all four tested PM motors. By using the same hyperparameters, the training process may not be optimal, but this approach can still yield satisfying learning results.

Figures 15, 16 and 17 show averaged cumulative sums of rewards obtained during one mechanical revolution. Averaging makes the trends more clear, since instaneous rewards can be very noisy. Nevertheless, the most important information that the plots provide are the relative growth rates between the different scenarios and this information can be used for concluding approximately the best hyperparameter values for the system. In general, such hyperparameter values should be selected that provide the best learning outcome and make the learning happen in the shortest time. When considering the learning rate, the first few hundred iterations should be emphasized if the agent can cause superfluous stress to the system. Periodical rewards in Fig. 13 illustrate that most of nonsensical actions are performed during early phase of training. Therefore, the risk of damaging the system is the greatest when training is started. Since unnecessarily large current injections generate heat, it is desired to select such hyperparameter values that let the agent to reach sufficient performance level quickly, even at the expense of slightly decreasing total learning rate.

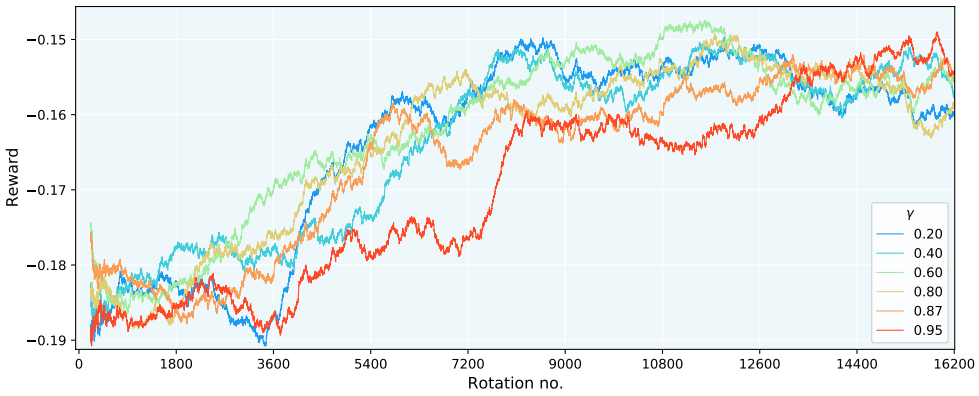


Figure 15: Learning simulated with different γ values

The parameter γ defines discount rate for rewarding. It has effect on how the agent perceives rewards and how shortsighted its actions are. Figure 15 shows that early learning takes longer with high γ values. With more moderate values the agent starts to learn more quickly, thus making less completely nonsensical actions. The parameter $\gamma = 0.6$ was selected to be used in simulations and experiments, as it appears to make the early learning quick and stable.

In order to demonstrate the effects of the learning rate α in the simulation environment, noise had to be generated. Therefore, white noise having maximum amplitude of 30% of the measured speed, was injected to the compensator speed input

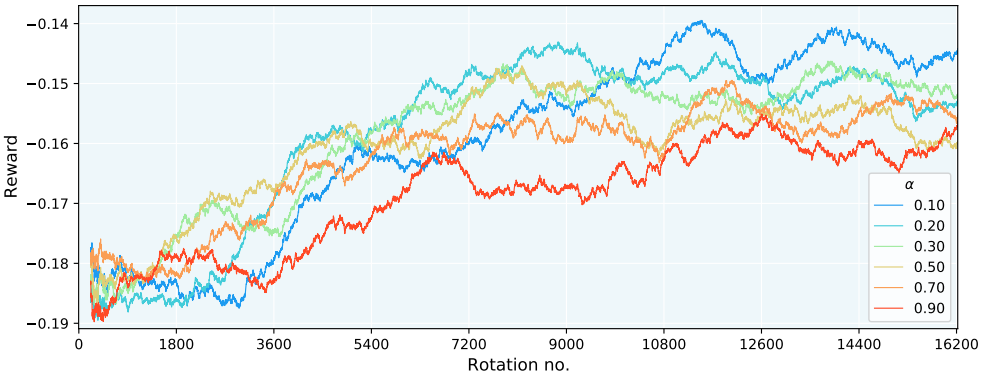


Figure 16: Learning simulated with different α values

signal. With constant 60 rpm speed, the compensator speed input can vary between $42 - 78$ rpm due to added noise. Figure 16 shows that learning is slower, when doing considerable updates with large α. With smaller update steps, the learning is faster in noisy environment. While considering the early learning speed, the value of $\alpha = 0.3$ was selected to be used.

The ϵ-greedy policy is used for training. The different ϵ decrease rates lead to different learning behaviour. Figure 17 illustrates learning with different k_ϵ values. It appears that there is no significant differences between selections. Value of $k_\epsilon = 300$ was selected to be used in simulations.

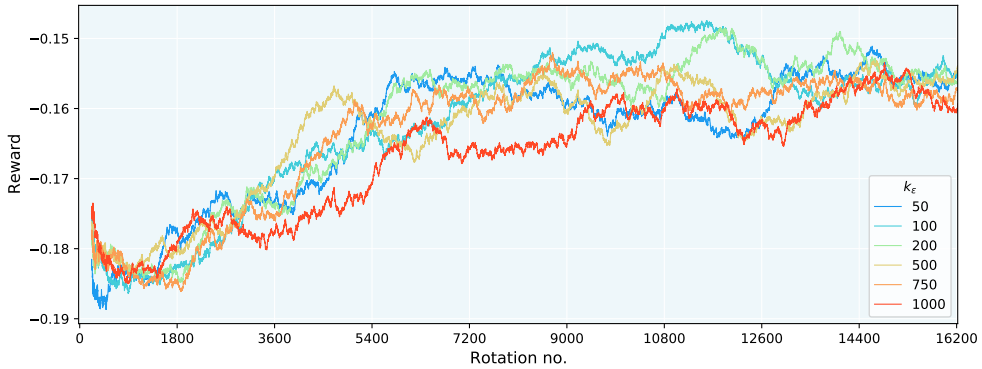


Figure 17: Learning simulated with different k_ϵ values

6 Simulation results

The PMSM model introduced in chapter 2.1 is used in the simulations. The flux linkages are defined as (1) and the stator voltages as (2). Iron losses are not modelled. Simulations are performed by time-stepping. Speed calculations are based on the equation of motion (6).

Evaluation was done by capturing data in three different cases. In the first case, the data was captured using only a PI speed controller. In the second and third cases, the data was captured by having the PI-controller coupled with ILC and Q-learning based compensators, respectively. Everything else was kept unchanged. Hence, the data captured using only the conventional PI-controller, works as a basis for the performance evaluation.

The speed and torque ripples can be measured, which allows fair evaluation of compensation effectiveness. The evaluation is done using a speed ripple factor (SRF) and a torque ripple factor (TRF). These are defined as [2]

$$\text{SRF} = \frac{\omega_{max} - \omega_{min}}{\omega_N} \cdot 100\% \qquad \text{TRF} = \frac{T_{max} - T_{min}}{T_N} \cdot 100\% \tag{21}$$

where peak-to-peak values are obtained from a single period. In addition to the SRF and the TRF, the amplitudes of 1st, 2nd, 6th, 12th harmonics are compared as these are pertinent to assessment. Simulations are performed with the motor described in Table 2.

Table 2: MS4887 motor nameplate values

Description	Symbol	Value	Unit
Nominal current (rms)	I_N	18.1	A
Nominal voltage (rms, line-line)	U_N	195.1	V
Nominal frequency	f_N	133.3	Hz
Nominal speed	ω_N	2000	rpm
Nominal power	P_N	6	kW
Nominal torque	T_N	28.6	Nm
Number of poles	N_p	8	

6.1 Torque pulsation model

In order to be able to evaluate compensation performance in the simulator, disturbances must be generated. Since torque pulsation needs to be only periodic, these can be modelled by using Fourier series

$$T_m(\theta_e) = \sum_{n=1}^{\infty} T_n \cos(n\theta_e + \phi_n) \tag{22}$$

where T_n is the magnitude of the nth harmonic and ϕ_n is the phase shift. The phase shift can be assumed to be zero for simplicity, as ϕ_n has no effect to compensation

Table 3: Modelled harmonics with respect to nominal torque

Harmonic order	1st	2nd	6th	12th
Magnitude (%)	1.37	0.25	0.45	0.09

results. In order to make the model more practical, only harmonics incorporating significant amount of energy, i.e., $\{1, 2, 6, 12\}$ are used for modelling the pulsations. This scheme requires only a few amplitudes to be specified in order to model pulsations. With more complex models, it was found that lack of details, such as stator slot number, can prevent modelling entirely.

The harmonic magnitude T_n must be measured, computed or estimated by utilizing available information. For realistic simulations, the magnitudes were measured at high load from one of the motors used in the experiments. Harmonics T_1, T_2, T_6 and T_{12} were obtained using a torque transducer and then calculating FFT. Harmonic magnitudes are listed in Table 3. It can be observed that the 12th harmonic already incorporates very little energy. Therefore, exclusion of less significant harmonics should not introduce much error. A similar conclusion was made in [6].

All simulations were made at 60 rpm speed, because pulsations degrade motor performance the most at low speeds. This can be reasoned by inspecting (7) and its frequency response. With higher frequencies, the speed oscillations diminish spontaneously, thus making low speed compensation more critical. In order to make simulations more realistic, white noise was added to the speed signal. This was done similarly to chapter 5.6, though more restrained 10% amplitude was used for generating the noise. Due to added noise, the compensator speed input can vary between $54 - 66$ rpm when the measured speed stays constant 60 rpm.

6.2 Results with fine tuned compensators

In the first simulation, both compensators were fine tuned. Many test simulations were performed in order to find good parameter values before evaluation. It was found that ILC performs well with parameter values of $\Phi = 9$, $\Gamma = 3$ and $\alpha = 0.05$. ILC memories were set to store 750 values. According to simulation results in Figs. 14, 16, 15 and 17, the Q-learning based compensator performs well with parameter values given in Table 4. In addition to the table values, the torque pulsation maximum was set to $T_{max} = 0.014$. Seven actions and 100 states were used in order to keep memory consumption roughly equivalent to ILC, thus allowing more fair comparison. Better compensation performance could be achieved with both compensators, if more memory was reserved for storing values.

The ripple factors were calculated from simulation data. When using only a conventional PI controller, the measures are SRF = 0.21% and TRF = 11.8%. With ILC enabled, the corresponding measures are SRF = 0.04% and TRF = 11.6%. With Q-learning, SRF = 0.05 % and TRF = 10.8%. Ripple factors show that pulsations do get significantly reduced when compensators are enabled. SRF reduction is much greater than TRF. Reason for this can be observed from Fig. 18, which shows that

the torque signal is much noisier than the speed signal due to system dynamics (7). The speed signal behaves similarly to a low-pass filtered signal, which makes the SRF measure more reliable than TRF. Peak values used for calculating the ripple factor are more consistent with the speed signal than they are with torque. The overall ripple is clearly getting reduced even with the torque signal, but singular spikes in the data can still keep the TRF measure high. It can be concluded that the SRF measure should be favoured over the TRF, since the SRF measure is more consistent.

Harmonic magnitudes in Fig. 19 allow to make the same conclusion as can be made from the SRF and TRF measures. Compensators reduce pulsations, since harmonics are much smaller when compensators are enabled. Speed harmonics are reduced more than torque harmonics similarly to SRF and TRF measures. It can be observed that Q-learning based method is not able to compensate the first speed harmonic as well as ILC. Since the first harmonic has the greatest magnitude, it has the greatest effect on the ripple. This explains why ILC manages to compensate pulsations more according to the SRF measure. In case of torque harmonics, the Q-learning based method appears to be able to compensate torque harmonics better in overall than ILC, hence leading to smaller TRF.

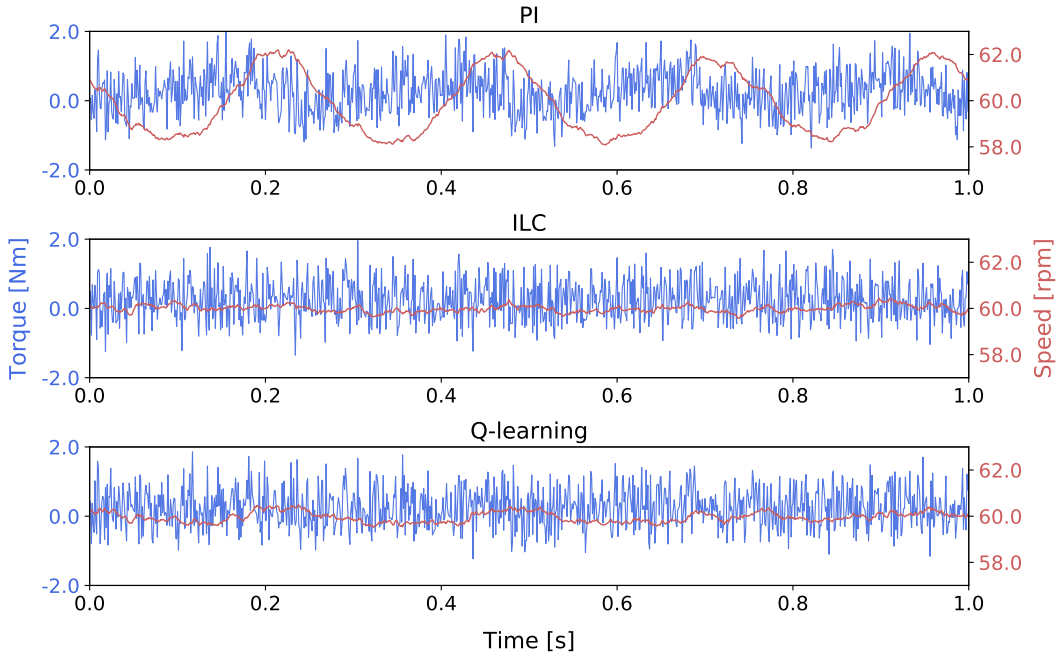
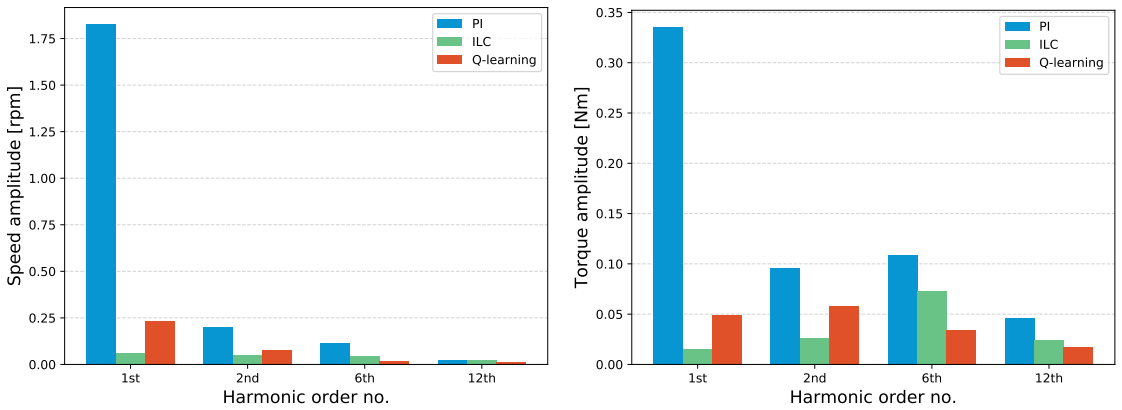


Figure 18: Simulated torque and speed pulsations with PI, ILC and Q-learning compensation schemes

Table 4: Parameter values used with Q-learning

Parameter	α	γ	k_ϵ	λ	N_a
Value	0.3	0.6	300	32	7



(a) Speed harmonics

(b) Torque harmonics

Figure 19: Speed and torque harmonics of fine tuned compensators

The simulation result supports previous studies. In [2], ILC was tested experimentally. The SRF measure without compensation was 0.65% and with compensation 0.15%. A ripple reduction ratio can be calculated from these percentages, which allows to compare the performance with the simulation result. By calculating ratios, $0.15\%/0.65\% \approx 0.23$ and $0.043\%/0.206\% \approx 0.21$ it can be observed that compensation performance is quite similar.

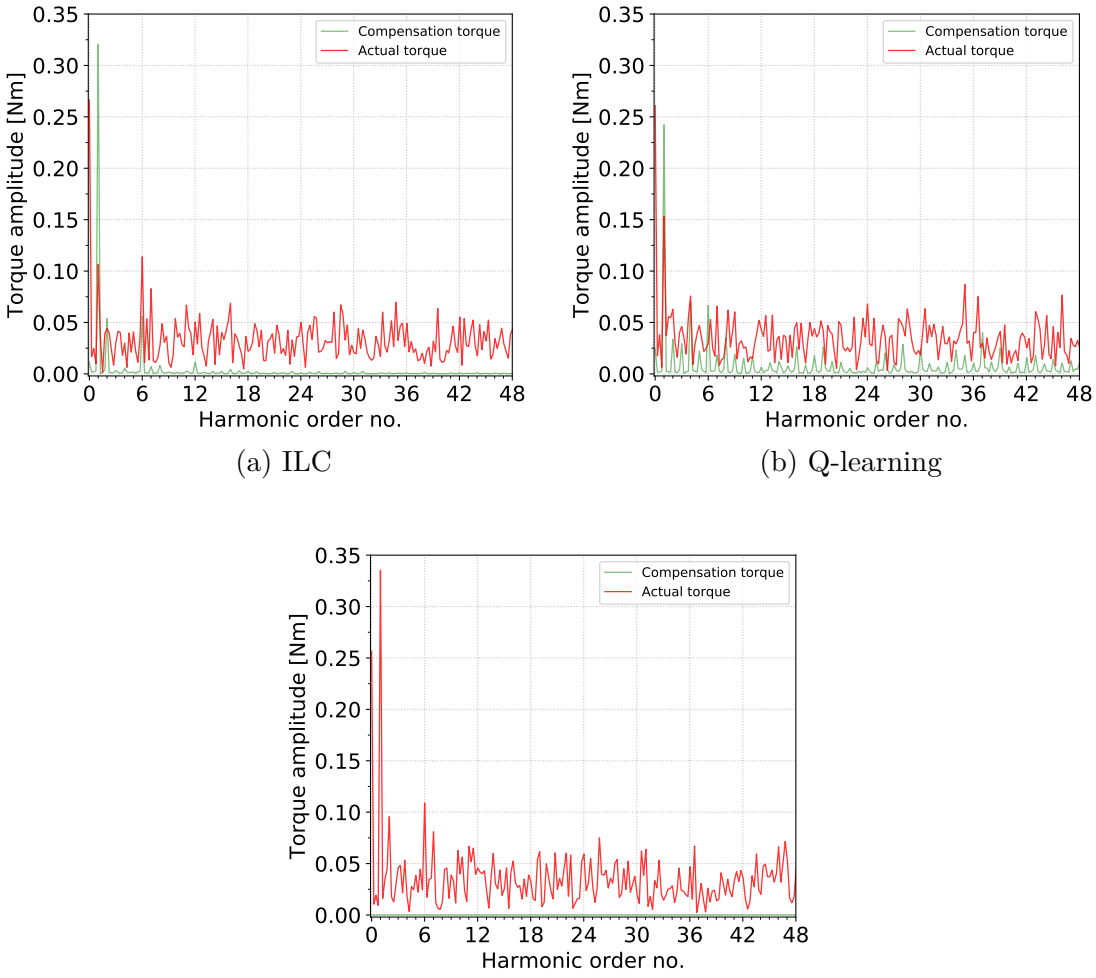
6.3 Results with nonideal parameters

Another simulation was made with more realistic parameter values. It is unfeasible to assume that maximum torque amplitude is known beforehand, therefore nonideal value of $T_{max} = 0.01$ was intentionally used. The ILC gains in the previous simulation were higher than it is usually possible to use with a real motor. Hence, lower gains of $\Phi = 4.75$ and $\lambda = 1.0$ were selected for the simulation. Additionally, a higher forgetting factor $\alpha = 0.15$ value was used with ILC. Other parameters and operating conditions were kept the same as in the previous simulation.

Figure 20 shows that disturbances are getting reduced when compensators are used. The 1st and 2nd harmonics can be clearly seen to get smaller when comparing Figs. 20a and 20b to 20c. Also 6th harmonic is getting reduced when Q-learning based compensator is used. The 12th harmonic reduction is not clear due to noisy appearance of the torque signal and low energy of the 12th harmonic. Since harmonics are getting smaller, it can be concluded that compensators are able to reduce torque ripple with nonideal parameter values. The compensation performance is worse than with fine compensators in Fig. 19, but this is fully expected, as some q-axis current reference injections are simply too small.

The green signal in Figs. 20a and 20b illustrates the torque that gets produced when compensators correct the q-axis current reference, i_q^*. The compensation torque allows to observe that both compensators successfully identify periodic pulsations, because produced torque harmonics overlap with harmonic orders of the disturbances. Besides 1st, 2nd, 6th and 12th harmonics, the Q-learning based method can be seen

to generate also other minor harmonics. The rise of these minor harmonics can be explained with the rough compensation pattern, which is visualized in Fig. 12. The compensator cannot generate ideal waveform needed for compensation due to quantization of the actions. ILC can inject any real value, which explains why ILC generates less these superfluous compensation harmonics. The almost nonexistent superfluous harmonics in case of ILC are caused by noise, which prevents precise disturbance identification. Anyhow, these superfluous harmonics are harmless in both cases, since their amplitude is tiny compared to prominent harmonics.



(a) ILC

(b) Q-learning

(c) PI control

Figure 20: Simulated torque harmonics when (a) ILC compensator is used (b) Q-learning based compensator is used and (c) conventional PI speed control is used

7 Experimental results

The motor setup used in the experiments is shown in Fig. 21. A PM motor is on the left and a synchronous reluctance motor (SyRM) on the right. The SyRM was used as a load machine, which allowed the compensator performance measurements under different load conditions. A brushless resolver, with 10 kHz excitation signal frequency, was connected to the PM motor and was used for measuring the speed. The PM motor parameters are listed in Table 2.

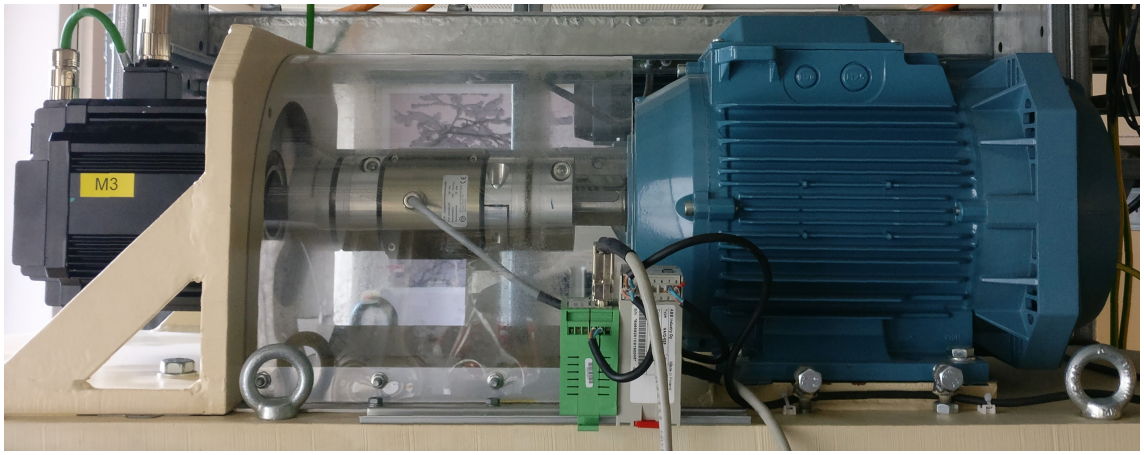


Figure 21: Main motor setup used for the experiments

Experimental results were collected and evaluated in a similar manner to the simulation results. Three measurements with all control methods were taken in each operating point and then these results were compared. The performance comparison was conducted comparing amplitudes of speed harmonics and using the SRF (21) measure.

7.1 Measurement considerations

Compensation performance was tested in different load conditions. The load machine was used to produce loads corresponding 0%, 10%, 50% and 80% of the nominal torque of the PM machine. The speed reference was kept at constant 60 rpm.

During the drive startup, current offsets are automatically identified and removed. This process is performed automatically after every drive startup. Therefore, the Q-learning and ILC measurements taken on different times, are not necessarily directly comparable, since there can be variation between harmonics. Especially, the first harmonic induced by current offsets may be slightly different. For this reason, the ILC and Q-learning measurements are visualized separately. PI speed control measurements were taken twice in conjunction with compensator measurements. Hence, PI speed control measurements allow calculation of reduction ratio, which makes it possible to compare compensation methods rigorously.

It was also found that there is misalignment between the motor shafts, which leads to mechanical oscillations visible in the Fourier spectrum. The frequency of

these oscillations seemed to be different from the harmonics being studied.

The torque transducer visible in Fig. 21 was planned to be used for measuring torque. Unfortunately, the readings were found to drift and torque measurements had to be discarded as these cannot be considered to be reliable. Therefore, only speed measurements are used for making conclusions. On the bright side, the speed signal was found to be more reliable than torque signal in the chapter 6.2. For this reason, the torque measurements can be considered to be redundant and these could have been only used for verification. Now the verification of results is done by testing the compensation with supplementary motor setup.

7.2 ILC measurements

When testing ILC, the tuning strategy was to find a maximum value for gain Φ, which still keeps the system stable, as this maximizes the compensation performance. The α and Γ gains were adjusted as little as possible between different operating points in order to make it easy to see effects of Φ adjustment. However, a higher α value was used, if this made it possible to use higher Φ gain value. The gain Γ was initially set low to make it possible to keep a higher Φ gain value at the cost of slower learning rate. The gains used in the experiments are listed in Table 5.

Table 5: ILC gains used in 60 rpm measurements

	Motoring				Generating			
Load	0%	10%	50%	80%	0%	10%	50%	80%
α	0.15	0.15	0.15	0.25	0.10	0.10	0.10	0.10
Γ	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Φ	4.75	3.0	2.8	1.25	3.0	2.9	3.0	3.0

It can be observed from Table 5 that the compensator behaviour was not identical in motoring and generating modes. The generating mode was easier for the compensator to handle, since gains could be kept almost identical between different operating points, whereas the motoring mode required more re-tuning. The control algorithms in both operation modes are similar, but not identical. Hence, software differences may provide an answer to different behaviour, as well as, differences in the machine dynamics between motoring and generating modes. Nevertheless, the gains could not be kept the same in all operating points, even when considering only one operation mode. Therefore, a gain scheduling scheme would be needed in order to be able to use ILC in different operating points automatically. Without gain scheduling, the system can become unstable when operating point is changed. Another solution to the problem is to use gain values that work in the most demanding operating point, but then the compensation performance is compromised in less demanding conditions.

At low load conditions, the compensation is almost nonexistent. With two decimal accuracy, the PI and ILC SRF is 0.14% at 0% load. The same applies, when the load

is increased to 10%. The SRF is 0.19% with PI and ILC. When load is increased to 50% level, the pulsations become more visible, which makes identification and compensation easier. The compensator reduces SRF from 0.56% to 0.34%. At 80% load, the ILC manages to reduce SRF from 0.97% to 0.84%. Compensation performance is worse at 80% load than at 50% load, even when the magnitude of pulsations is greater. One possible reason for worse performance are the gain values that do not allow ILC to inject enough high correction value to the q-axis current reference. Another possible explanation is the altered type of torque and speed pulsations. Under heavy load, noninteger harmonics become greater due to non-ideal load mechanism when DC generator is excited for loading purposes [48]. The induced noninteger harmonics cannot be compensated by ILC [2, 48], which results to worse overall performance.

The speed harmonics in Fig. 23 are consistent with SRF measures. With 80% load, the compensation performance is truly worse than with 50% load. With 50% load, the first compensated harmonic can be seen to be less than half of the original amplitude, wherease with 80% load, it is roughly half of the original amplitude. At 0% and 10% loads, ILC manages to reduce disturbance harmonics only slightly in overall. However, the pulsations are also small with low loads. With higher loads the pulsations become more noticeable and ILC is able to reduce the harmonics significantly.

7.3 Q-learning measurements

The measurements with the Q-learning based compensator were taken in a similar manner to the ILC measurements. The parameters used in measurements are the same as in the table 4 with exceptions of $\lambda = 1$ and $k_\epsilon = 750$. The pulsation maximum T_{max} was adjusted between different operating points. All other parameters were kept constant. The parameter T_{max} value was set by testing few different values and then selecting the one producing the best results. In practice, values 0.01, 0.02 and 0.035 were used. These values are unlikely to be ideal and better compensation results could be achieved with better value selection.

Figure 24 shows measured speed with conventional PI speed control and Q-learning based compensator. It can be observed that speed pulsations are smaller at higher loads when Q-learning based compensator is used. Since speed pulsations are getting reduced, also torque pulsations must be getting smaller according to (7). Therefore, it can be concluded that the method can compensate torque pulsations successfully.

Speed ripple factors were calculated in different operating conditions. With 0% load, the SRF is 0.15% with conventional PI speed control and 0.18% when the compensator is used. Figure 24 verifies the result of the ripple not getting reduced when the Q-learning based compensator is enabled at 0% load. However, at 10% load, the compensator is able to reduce the speed ripple slightly. With PI control, the SRF is 0.25% and with Q-learning, it is 0.24%. Similarly to ILC, the ripple reduction is more significant at 50% and 80% loads when magnitude of pulsations is much greater than with lower loads. At 50% load, the compensator reduces SRF from 0.50% to 0.43%. Unlike to ILC, the ripple reduction is even greater when load

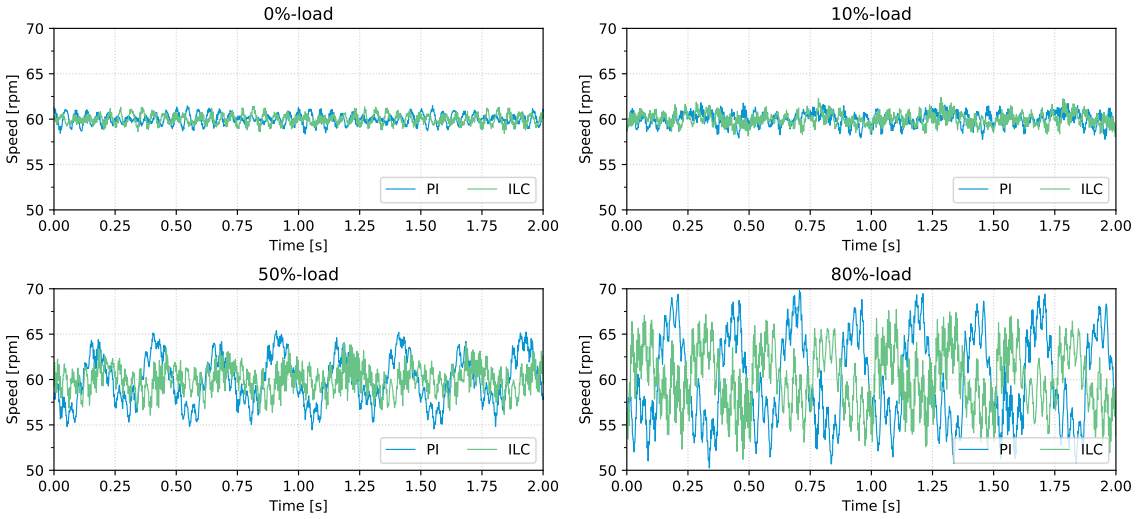


Figure 22: Speed pulsations encountered with PI control getting reduced when ILC based compensator is used at 60 rpm speed

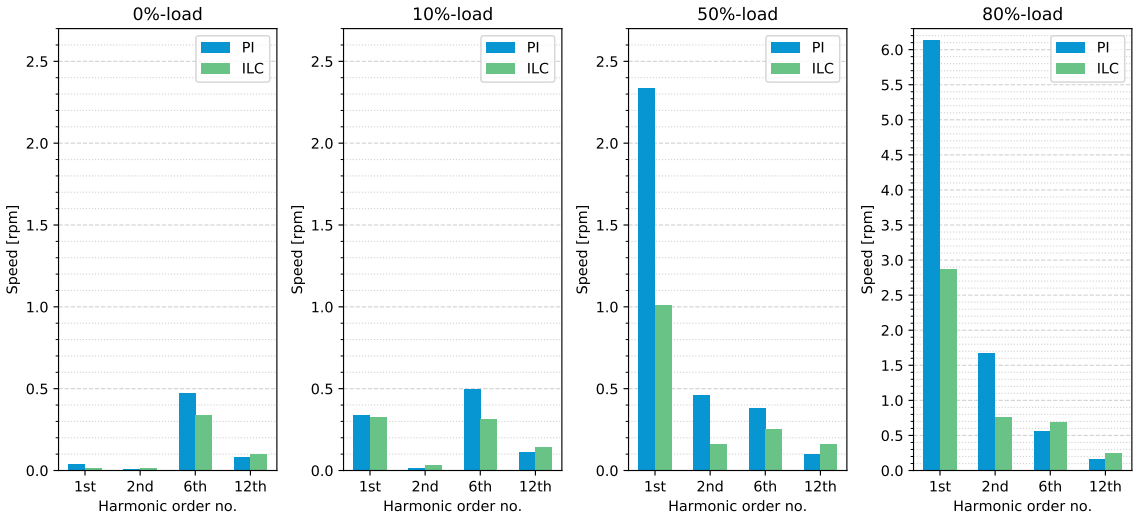


Figure 23: Speed harmonics with PI control and ILC at 60 rpm speed

is increased from 50% to 80%. At 80% load, the SRF with PI is 0.74% and with Q-learning it is 0.56%.

Figure 25 shows speed harmonics that were obtained by computing FFT from the speed measurements. In overall, the harmonics are congruent with the SRF measures. Disturbances are getting reduced in all studied operating points when compensation is used. Therefore, the compensator seems to be functioning properly.

7.4 ILC and Q-learning performance evaluation

Table 6 shows reduction ratios that have been calculated from SRF measures. The ratios show how much smaller the ripple became when compensator was enabled. For example, reduction ratio for Q-learning at 10% load is calculated as $0.235/0.249 = 0.94$. The ratio allows to conclude that the ripple became 6% smaller when Q-learning based compensation was used.

According to reduction ratios, ILC and Q-learning based compensators should not be used if pulsations are minimial, since the ripple is not getting reduced. When pulsations become notable, then both compensators do reduce the ripple remarkably. ILC performs better at 50% load region than Q-learning, but at 80% load, the situation is opposite. ILC may get limited by gain values, which explains why Q-learning based compensator performs better at 80% load. However, current reference injections made by ILC are more accurate than corrections done by Q-learning method, which explains better performance of ILC at 50% load where it is not getting limited as much. ILC can inject any real value whereas Q-learning method uses quantized actions. Therefore, ILC can theoretically achieve better performance.

Table 6: Speed ripple reduction ratios with different loads

Load	ILC	Q-learning
0%	1.0	1.15
10%	1.05	0.94
50%	0.61	0.87
80%	0.86	0.76

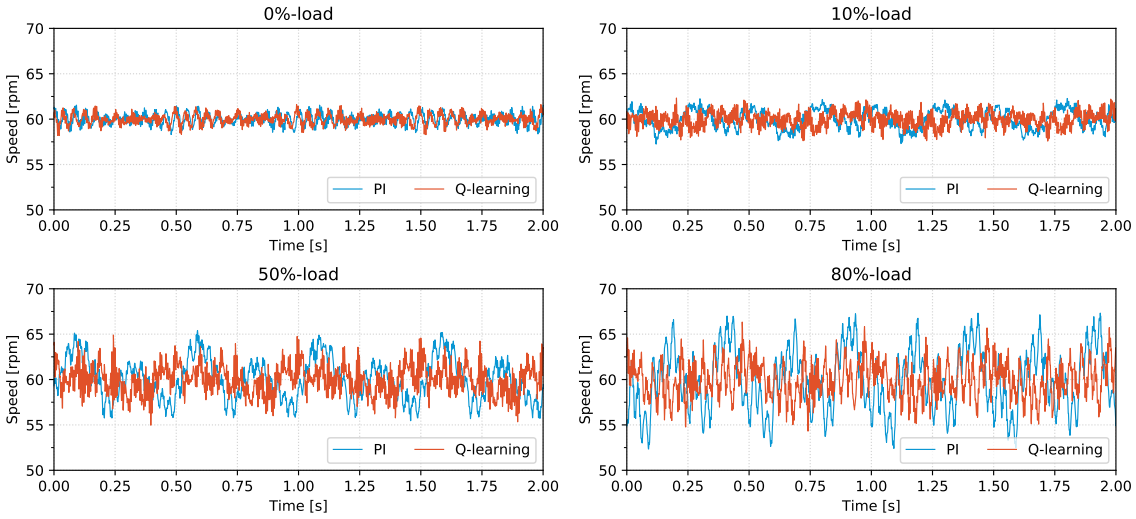


Figure 24: Speed pulsations encountered with PI control getting reduced when Q-learning based compensator is used at 60 rpm speed

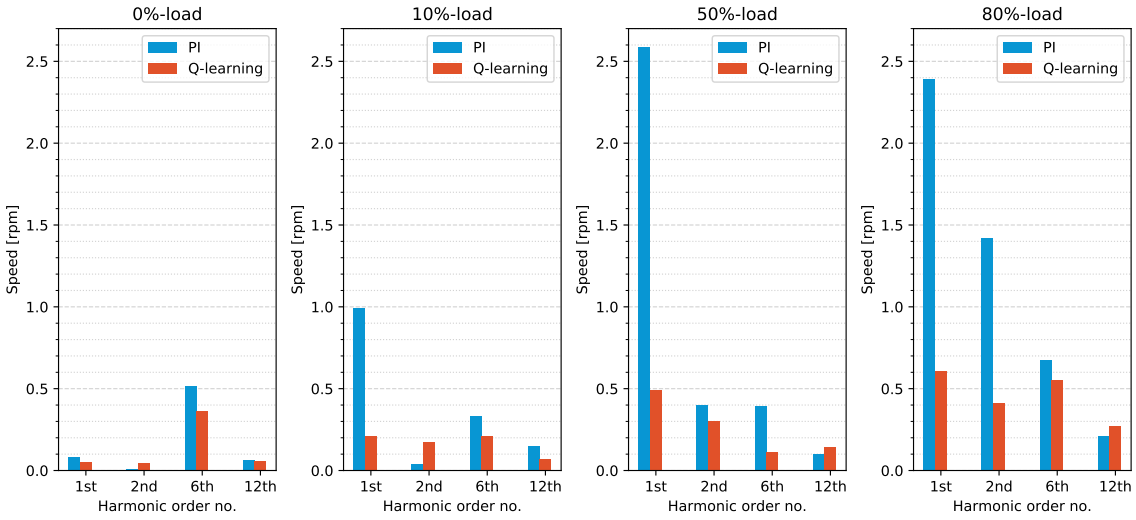


Figure 25: Speed harmonics with PI control and Q-learning at 60 rpm speed

7.5 Cogging torque compensation

A PM servomotor was used to verify that compensation works with different motors. The servomotor is known to have high cogging torque, which makes it ideal for testing, as this reflects typical use case for compensators. The servomotor was not connected to any load. Hence, many potential error sources were eliminated, but at the same time, only no-load tests were possible. A resolver was used for measuring speed pulsations. The servomotor parameters are listed in Table 7.

Table 7: SDM251 servomotor nameplate values

Description	Symbol	Value	Unit
Nominal current (rms)	I_N	1.5	A
Nominal voltage (rms, line-line)	U_N	330.0	V
Nominal frequency	f_N	150.0	Hz
Nominal speed	ω_N	3000	rpm
Nominal power	P_N	0.72	kW
Nominal torque	T_N	2.30	Nm
Number of poles	N_p	6	

At 60 rpm speed, the pulsations were noticeable and it was possible to hear the motor having troubles rotating smoothly due to high cogging torque. However, when compensation was enabled, the uneven noise disappeared and the motor was clearly able to keep more constant speed. The SRF measure with conventional PI control was 2.31%. ILC reduced SRF to 0.68% and Q-learning based approach to 0.62%. The effects of compensation can be observed from Figs. 26 and 27 that show measured shaft speed. When compensators have been trained and compensation values have been saved into memory, the compensation is instantaneous with both methods. ILC appears to be more graceful for the control than Q-learning method when compensators are suddenly enabled, as it takes a few seconds for the system to stabilize with the Q-learning method. The produced waveform is also different between the methods.

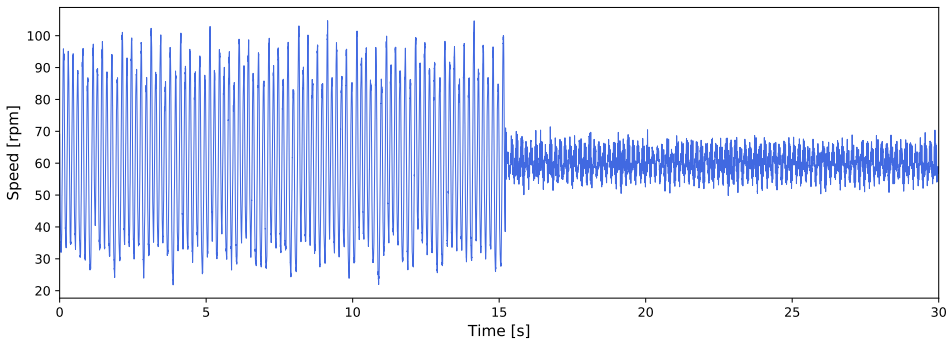


Figure 26: Speed ripple getting reduced when ILC based compensator is enabled

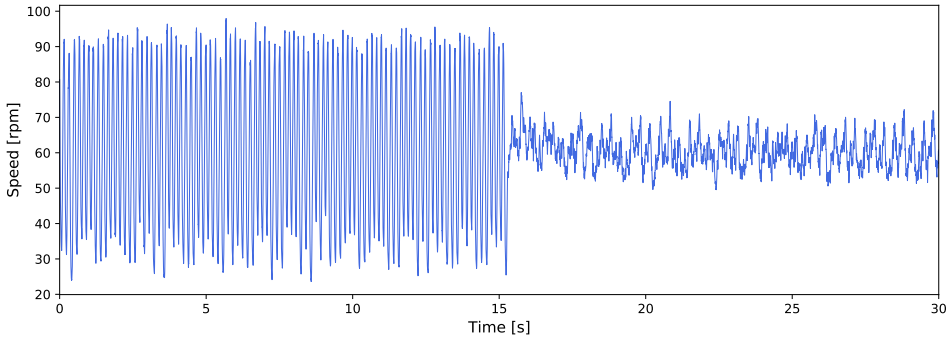


Figure 27: Speed ripple getting reduced when Q-learning based compensator is enabled

The ILC parameters used in the experiment, were $\Phi = 1.5$, $\Gamma = 0.9$ and $\alpha = 0.15$. These values were experimentally found to produce best results. With Q-learning, $T_{max} = 0.03$ was used in addition to parameters values given in Table 4.

7.6 Compensation with noisy systems

The negative effects of noise can be reduced with filtering or by utilizing the forgetting factor with ILC. It was found that the forgetting factor is superior compared to filtering of the speed signal. This was expected result as filtering leads to phase shift, which results to delayed compensation. In the worst case, it is possible to align the compensation crests with the disturbance crests, which creates constructive inference that transforms the compensator to a disturbance amplifier. The negative effects of the filtering can be seen in Fig. 28 where the same setup, with the same ILC gains, was run by using three different speed filtering coefficients. It can be observed that the performance is best when the speed signal is not filtered at all. Hence, filtering should be avoided. The forgetting factor solves the noise issue without introducing additional phase shift. Therefore, the forgetting factor cannot amplify pulsations in any circumstances. However, the use of forgetting factor is still a tradeoff. Forgetting factor makes it impossible to achieve perfect compensation [2, 14]. Therefore, as small α value should be used as possible. It was found experimentally that use of the forgetting factor yields better results with noisy motor setup than heavy filtering.

7.7 Compensation at various speeds

Figure 29 shows compensation performance at different low speeds. Near 60 rpm speed, both compensators reduce speed pulsations tremendously. When moving to higher speeds, ILC seems to have troubles compensating pulsations and it can be even seen to degrade the performance. The Q-learning based compensator is able to compensate in all operating points, although the performance gets worse at higher speeds. Therefore, The Q-learning based method can be better if the machine is run at various low speed operating points.

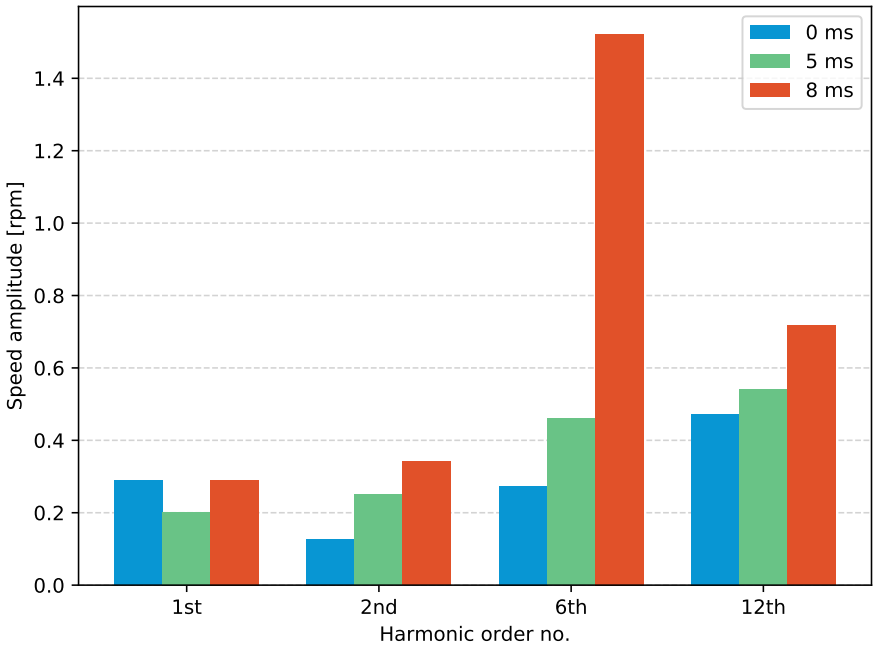


Figure 28: ILC based compensation with different speed filtering time constants

However, the compensation is not really important at higher speeds, since pulsations do get reduced automatically due to inertia, as can be seen from the Fig. 29. The pulsations become smaller even with conventional PI speed control when speed is increased. Therefore, ILC is still totally valid compensation method and the compensators should be used only at low speeds where torque pulsations are truly problematic.

It was found that compensation performance of the Q-learning method is better at high speeds, if lower rotation speed is used for training and the speed is increased after training. This observation can be explained with state skips that occur at high speeds. Due to fast rotation speed, not all states are visited and reduced amount of Q-values are updated. With lower rotation speed, no such skipping occurs. The point where speeds starts to have effect on learning, depends on the number of states and the algorithm update frequency. For example, with 100 states the rotor should always rotate less than $0.01 \cdot 2\pi \approx 0.06$ rad during single iteration in order to not jump over states. If algorithm is updated every $500\mu s$, then the speed should stay lower than 120 rpm during training.

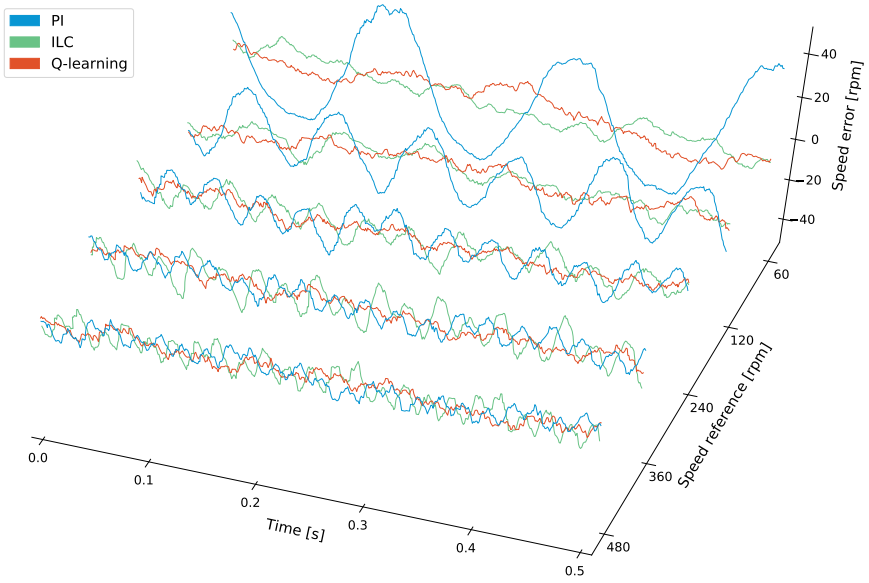


Figure 29: Speed pulsations getting reduced in different speeds with the servomotor

8 Conclusion

ILC based compensation was tested with two PM motors. It was verified that ILC can be used for compensation with industrial PM motors. When testing with different loads, it was noticed that the same gain values may not work in every operating point. This is a significant drawback, since manual gain tuning was found tedious. Therefore, a gain scheduling algorithm may be required if operating conditions change and maximum compensation performance is pursued with high gain values.

It was demonstrated that Q-learning can be used for ripple reduction. Experimental tests show that compensation performance is similar to ILC, which is one of the best automatic compensation methods. Therefore, it is safe conclude that the Q-learning based method is effective. With the servomotor, the Q-learning method managed to reduced speed ripple 73%. The compensator is modular and it can be easily applied to the existing control loop. Compensation is automatic, which allows the method to be utilized with weak system knowledge. The compensator does not have dependency on motor parameters, which allows it to be easily used with various motors. However, the current implementation still requires rough estimate of the maximum pulsation magnitude.

Many improvements could be made to the compensators. The Q-learning based compensator could be simplified by reducing parameters, as some of these seemed to have only minimal effect on the performance. It would be also interesting to test if the maximum amplitude parameter T_{max} could be fixed to some percentage of the motor nominal torque. In contrast to simplifying, it would be also intriguing to test if existing double Q-learning or periodic Q-learning algorithms could provide even better results than the conventional Q-learning algorithm. Finally, it would be also interesting to implement a tabular ILC compensator, which saves compensation values to a table similarly to Q-learning.

References

[1] J. Liu, H. Li, and Y. Deng, "Torque ripple minimization of pmsm based on robust ilc via adaptive sliding mode control," *IEEE Transactions on Power Electronics*, vol. 33, no. 4, pp. 3655–3671, April 2018.

[2] Weizhe Qian, S. K. Panda, and J. X. Xu, "Speed ripple minimization in pm synchronous motor using iterative learning control," *IEEE Transactions on Energy Conversion*, vol. 20, no. 1, pp. 53–61, March 2005.

[3] Y. Yuan, F. Auger, L. Loron, S. Moisy, and M. Hubert, "Torque ripple reduction in permanent magnet synchronous machines using angle-based iterative learning control," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 2518–2523.

[4] T. Pajchrowski, "Application of neural networks for compensation of torque ripple in high performance pmsm motor," in *2017 19th European Conference on Power Electronics and Applications (EPE'17 ECCE Europe)*, Sep. 2017, pp. P.1–P.8.

[5] J. Y. Hung and Z. Ding, "Design of currents to reduce torque ripple in brushless permanent magnet motors," *IEE Proceedings B - Electric Power Applications*, vol. 140, no. 4, pp. 260–266, July 1993.

[6] G. Ferretti, G. Magnani, and P. Rocco, "Modeling, identification, and compensation of pulsating torque in permanent magnet ac motors," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 6, pp. 912–920, Dec 1998.

[7] Y. Deng, H. Li, and J. Liu, "Method for suppressing torque ripple of permanent magnet synchronous motor based on robust iterative learning control," CN Patent CN107070341B, May 10, 2019.

[8] H. Injung, K. Younghun, and S. Goansoo, "Apparatus for controlling speed of a rotary motor," EP Patent EP0805383B1, May 19, 1999.

[9] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[13] N. P. Quang and J.-A. Dittrich, *Vector Control of Three-Phase AC Machines: System Development in the Practice*, 2nd ed. Springer Publishing Company, Incorporated, 2016.

[14] Jian-Xin Xu, S. K. Panda, Ya-Jun Pan, Tong Heng Lee, and B. H. Lam, "A modular control scheme for pmsm speed control with pulsating torque minimization," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 3, pp. 526–536, June 2004.

[15] C. A. Borghi, D. Casadei, A. Cristofolini, M. Fabbri, and G. Serra, "Application of a multiobjective minimization technique for reducing the torque ripple in permanent-magnet motors," *IEEE Transactions on Magnetics*, vol. 35, no. 5, pp. 4238–4246, Sep. 1999.

[16] W. Fei and Z. Q. Zhu, "Comparison of cogging torque reduction in permanent magnet brushless machines by conventional and herringbone skewing techniques," *IEEE Transactions on Energy Conversion*, vol. 28, no. 3, pp. 664–674, Sep. 2013.

[17] N. Bianchi and S. Bolognani, "Design techniques for reducing the cogging torque in surface-mounted pm motors," *IEEE Transactions on Industry Applications*, vol. 38, no. 5, pp. 1259–1265, Sep. 2002.

[18] M. S. Islam, S. Mir, and T. Sebastian, "Issues in reducing the cogging torque of mass-produced permanent-magnet brushless dc motor," *IEEE Transactions on Industry Applications*, vol. 40, no. 3, pp. 813–820, May 2004.

[19] Z. Q. Zhu, S. Ruangsinchaiwanich, and D. Howe, "Synthesis of cogging-torque waveform from analysis of a single stator slot," *IEEE Transactions on Industry Applications*, vol. 42, no. 3, pp. 650–657, May 2006.

[20] T. Liu, S. Huang, and J. Gao, "A method for reducing cogging torque by magnet shifting in permanent magnet machines," in *2010 International Conference on Electrical Machines and Systems*, Oct 2010, pp. 1073–1076.

[21] L. Gašparin and R. Fišer, "Cogging torque sensitivity to permanent magnet tolerance combinations," *Archives of Electrical Engineering*, vol. 62, 09 2013.

[22] J. Gao, G. Wang, X. Liu, W. Zhang, S. Huang, and H. Li, "Cogging torque reduction by elementary-cogging-unit shift for permanent magnet machines," *IEEE Transactions on Magnetics*, vol. 53, no. 11, pp. 1–5, Nov 2017.

[23] L. Zhu, S. Z. Jiang, Z. Q. Zhu, and C. C. Chan, "Analytical methods for minimizing cogging torque in permanent-magnet machines," *IEEE Transactions on Magnetics*, vol. 45, no. 4, pp. 2023–2031, April 2009.

[24] Se-Kyo Chung, Hyun-Soo Kim, Chang-Gyun Kim, and Myung-Joong Youn, "A new instantaneous torque control of pm synchronous motor for high-performance

direct-drive applications," *IEEE Transactions on Power Electronics*, vol. 13, no. 3, pp. 388–400, May 1998.

[25] Dae-Woong Chung and Seung-Ki Sul, "Analysis and compensation of current measurement error in vector-controlled ac motor drives," *IEEE Transactions on Industry Applications*, vol. 34, no. 2, pp. 340–345, March 1998.

[26] K. C. Yeo, G. Heins, F. De Boer, and B. Saunders, "Adaptive feedforward control to compensate cogging torque and current measurement errors for pmsms," in *2011 IEEE International Electric Machines Drives Conference (IEMDC)*, May 2011, pp. 942–947.

[27] Sangmoon Hwang and D. K. Lieu, "Design techniques for reduction of reluctance torque in brushless permanent magnet motors," *IEEE Transactions on Magnetics*, vol. 30, no. 6, pp. 4287–4289, Nov 1994.

[28] Sang-Moon Hwang, Jae-Boo Eom, Yoong-Ho Jung, Deug-Woo Lee, and Beom-Soo Kang, "Various design techniques to reduce cogging torque by controlling energy variation in permanent magnet motors," *IEEE Transactions on Magnetics*, vol. 37, no. 4, pp. 2806–2809, July 2001.

[29] Z. Q. Zhu, S. Ruangsinchaiwanich, N. Schofield, and D. Howe, "Reduction of cogging torque in interior-magnet brushless machines," *IEEE Transactions on Magnetics*, vol. 39, no. 5, pp. 3238–3240, Sep. 2003.

[30] D. G. Dorrell and M. Popescu, "Odd stator slot numbers in brushless dc machines—an aid to cogging torque reduction," *IEEE Transactions on Magnetics*, vol. 47, no. 10, pp. 3012–3015, Oct 2011.

[31] M. Tang, A. Formentini, S. A. Odhano, and P. Zanchetta, "Torque ripple reduction of pmsms using a novel angle-based repetitive observer," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 4, pp. 2689–2699, April 2020.

[32] D. C. Hanselman, "Minimum torque ripple, maximum efficiency excitation of brushless permanent magnet motors," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 3, pp. 292–300, June 1994.

[33] M. Piccoli and M. Yim, "Cogging torque ripple minimization via position based characterization," in *Robotics: Science and Systems*, 2014.

[34] M. Sumega, P. Rafajdus, G. Scelba, and M. Stulrajter, "Control strategies for the identification and reduction of cogging torque in pm motors," in *2019 International Conference on Electrical Drives Power Electronics (EDPE)*, Sep. 2019, pp. 74–80.

[35] M. Ruderman, A. Ruderman, and T. Bertram, "Observer-based compensation of additive periodic torque disturbances in permanent magnet motors," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 1130–1138, May 2013.

[36] H. Muramatsu and S. Katsura, "An adaptive periodic-disturbance observer for periodic-disturbance suppression," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4446–4456, Oct 2018.

[37] S. Hara, Y. Yamamoto, T. Omata, and M. Nakano, "Repetitive control system: a new type servo system for periodic exogenous signals," *IEEE Transactions on Automatic Control*, vol. 33, no. 7, pp. 659–668, July 1988.

[38] M. Tang, A. Gaeta, A. Formentini, and P. Zanchetta, "A fractional delay variable frequency repetitive control for torque ripple reduction in pmsms," *IEEE Transactions on Industry Applications*, vol. 53, no. 6, pp. 5553–5562, Nov 2017.

[39] Y. Wang, F. Gao, and F. J. Doyle III, "Survey on iterative learning control, repetitive control, and run-to-run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[40] Z. Bien and J.-X. Xu, *Iterative learning control: analysis, design, integration and applications*. Springer Science & Business Media, 1998.

[41] H. Ahn, Y. Chen, and K. L. Moore, "Iterative learning control: Brief survey and categorization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1099–1121, Nov 2007.

[42] H.-S. Ahn, K. Moore, and Y. Chen, *Iterative learning control: Robustness and monotonic convergence for interval systems*. Springer Science & Business Media, 01 2007.

[43] S. Arimoto, T. Naniwa, and H. Suzuki, "Robustness of p-type learning control with a forgetting factor for robotic motions," in *29th IEEE Conference on Decision and Control*, Dec 1990, pp. 2640–2645 vol.5.

[44] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[45] S. Singh, T. Jaakkola, M. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, pp. 287–308, 03 2000.

[46] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[47] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," 2015.

[48] J.-X. Xu, S. Panda, and T. Lee, *Real-time iterative learning control. Design and applications*. Springer Science & Business Media, 01 2009.